

# SUSE® Rancher Manager

Version v2.15, Printed 2026-11-23 16:25 UTC

# Release Notes

## Current Version

| Version | Release Notes | Support Matrix | Prime | Community |
|---|---|---|---|---|
| v2.15.3 | View | N/A | ⬔ | ⬔ |

## Past Versions

| Version | Release Notes | Support Matrix | Prime | Community |
|---|---|---|---|---|
| v2.15.2 | View | View [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/rancher-v2-15-2/] | ⬔ | ⬔ |
| v2.15.1 | View | View [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/rancher-v2-15-1/] | ⬔ | ⬔ |
| v2.15.0 | GitHub Release [https://github.com/rancher/rancher/releases/tag/v2.15.0] | N/A | N/A | ⬔ |

## Release v2.15.3



- If you are using Active Directory Federation Service (AD FS), upgrading to Rancher v2.10.1 or later may cause issues with authentication, requiring manual intervention. These issues are due to the AD FS Relying Party Trust not being able to pick up a signature verification certificate from the metadata. For more information, see #48655 [https://github.com/rancher/rancher/issues/48655]. These issues can be corrected by either of two methods:
  - Updating the Relying Party Trust information from federation metadata (Relying Party Trust → Update from Federation Metadata...).
  - Directly adding the certificate (Relying Party Trust → Properties → Signature tab → Add → Select the certificate).

Rancher v2.15.3 is the latest patch release of Rancher. This is a Prime version release that introduces maintenance updates and bug fixes. To learn more about Rancher Prime, see our page on the Rancher Prime Platform [https://www.rancher.com/products/rancher-platform].

For more information on new features in the general minor release, see the v2.15.1 release notes.

### Security Fixes for Vulnerabilities

This release addresses the following security issues:

### Rancher

- Fixed a security vulnerability in the Rancher meta proxy endpoint where the destination allowlist was evaluated in a location that could not prevent a request from being forwarded. An authenticated user could cause the Rancher server to send requests to internal services that the allowlist is intended to block and receive the responses. The allowlist is now enforced in the request handler, which rejects disallowed destinations before any connection is made. For more information, see CVE-2026-88809 [https://github.com/rancher/rancher/security/advisories/GHSA-f9gf-wh3x-jx3v].
- Fixed a security vulnerability in Rancher's RBAC validation where the `inheritedNamespacedRules` field of a `GlobalRole` was not checked for privilege escalation. A user with delegated permissions to manage global roles and global role bindings could grant permissions in downstream clusters that exceeded their own. The `GlobalRole` and `GlobalRoleBinding` admission webhooks now validate this field in the same way as the other permission-carrying fields. For more information, refer to CVE-2026-88810 [https://github.com/rancher/webhook/security/advisories/GHSA-f898-2c8h-r7jc].
- Fixed a security vulnerability in the Rancher Manager meta proxy where the caller's own Rancher credentials were not consistently removed from requests forwarded to third-party cloud provider hosts. A signed-in user's session credential could be sent to a cloud provider during ordinary use of provisioning views, and an API client's Rancher token could be sent when it omitted the proxy's credential selection headers. The proxy now strips the caller's credentials on every request path. For more information, refer to CVE-2026-88811 [https://github.com/rancher/rancher/security/advisories/GHSA-x82x-2qj6-5rm2].
- Fixed a security vulnerability in Rancher where the authorization check that protects the local cluster shell was not applied to every API route that could reach it. Any authenticated user, including one with no permissions on the `local` cluster, could obtain an unprivileged shell pod on the management cluster, breaking the expected isolation of the management plane. The authorization check is now enforced consistently for all request forms. For more information, see CVE-2026-104623 [https://github.com/rancher/rancher/security/advisories/GHSA-8x6f-2v58-x6p7].

### Fleet

- Fixed a privilege escalation vulnerability in Fleet's agent management controller, where a Fleet cluster object created by an ordinary user could reference administrative credentials for the Rancher management (`local`) cluster. This allowed an authenticated user holding only the default `user` global role, with no permissions on the local cluster, to gain cluster-admin on the Rancher management cluster and thereby on all managed clusters. Fleet now only accepts a kubeconfig secret from another namespace when the secret is labeled for use by the referencing namespace, and only deploys agents onto the management cluster for its own system clusters. For more information, refer to CVE-2026-94290 [https://github.com/rancher/fleet/security/advisories/GHSA-qfmw-gw96-ffcr].
- Fixed a security vulnerability in Fleet where the cluster identity claimed during cluster registration was not bound to the registration token used for the request. A holder of a valid workspace registration token who knew another cluster's client identifier could obtain credentials for that cluster's Fleet management namespace and disrupt its legitimate agent. Fleet now records the registration token a cluster is registered with and rejects registration requests that claim the same cluster identity with a different token. For more information, refer to CVE-2026-104615 [https://github.com/rancher/fleet/security/advisories/GHSA-gx8v-2929-9q95].

For additional details, see the Security Advisories and CVEs page in Rancher's documentation or in Rancher's GitHub repository [https://github.com/rancher/rancher/security/advisories].

## Rancher general

### Major bug fixes

- Fixed an issue where Rancher settings configured using environment variables retained stale values and the `source: env` attribute after the environment variable was removed. When an environment variable is removed, Rancher now clears the stored environment source and value on startup. The setting then reverts to its default value. See See #58075 [https://github.com/rancher/rancher/issues/58075].

## Cluster provisioning

### Major bug fixes

- Fixed an issue where imported clusters could become unavailable after upgrading Rancher if the cluster agent was redeployed before its registration token was migrated, leaving the agent with an empty token. See #58028 [https://github.com/rancher/rancher/issues/58028].
- Fixed an issue where provisioned clusters with S3 etcd snapshots could remain stuck during deletion. See #57506 [https://github.com/rancher/rancher/issues/57506].

## Changes Since v2.15.2

See the full list of changes [https://github.com/rancher/rancher/compare/v2.15.2%E2%80%A6v2.15.3].

## Install/Upgrade Notes

If you're installing Rancher for the first time, your environment must fulfill the installation requirements.



- Chart name change for Rancher Prime.
  - The chart name change introduced in Rancher Prime v2.13.1 has been reverted. The chart name `rancher` should be used for all installations and upgrades. As an example, the installation command is now `helm install rancher rancher-prime/rancher`.
- Rancher now requires the cluster it runs on to have the Kubernetes API Aggregation Layer [https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/] enabled. This is because Rancher extends Kubernetes with additional APIs by registering its own extension API server. Please note that all versions of Kubernetes supported in this Rancher versions K8s distributions (RKE2/K3s) will have the aggregation layer configured and enabled by default.
  - Refer to the Extension API Server documentation and #50400 [https://github.com/rancher/rancher/issues/50400] for more information.
- Rancher Kubernetes Engine (RKE/RKE1) has reached end of life as of **July 31, 2025**. Rancher versions 2.12.0 and later no longer support provisioning or managing downstream RKE1 clusters. We recommend replatforming RKE1 clusters to RKE2 to ensure continued support and security updates. Learn more about the transition here [https://www.suse.com/support/kb/doc/?id=000021518].
  - Rancher now has a pre-upgrade validation check for RKE1 resources which fails and lists the RKE1 resources if present. Refer to #50286 [https://github.com/rancher/rancher/issues/50286] for more information.

- It is crucial that you review the available disk space on your nodes and plan accordingly before upgrading to Rancher v2.12.0 and later to avoid potential disk pressure and pod eviction issues.
  - For additional information, refer to the UI Server Side Pagination - Disk Space documentation.
- Rancher now has an enablement option called `AUDIT_LOG_ENABLED` for API Audit Logs for a Rancher installation.
  - In Rancher versions 2.11.x and earlier, only the `AUDIT_LEVEL` could be set and the default log level (`0`) would disable the audit log. In Rancher versions 2.12.x and later, the default log level (`0`) now only contains the log request and response metadata, and can be set when configuring `AUDIT_LOG_ENABLED`. If installing or upgrading via Helm you can enable the API Audit Logs and specify the log level by applying the following setting to your Helm command: `--set auditLog.enabled=true --set auditLog.level=0`. See the Enabling the API Audit Log to Record System Events documentation and #48941 [https://github.com/rancher/rancher/issues/48941].

## Changes in Image Artifacts

Image artifact digests are renamed in Rancher v2.12.0, v2.11.4 and v2.10.8. Up until this change, separate image digests files for each operating system and architecture have been maintained for compatibility reasons. With this change, only one file for each operating system is to be provided:

- The `rancher-images-digests-linux-amd64.txt` and `rancher-images-digests-linux-arm64.txt` files are to be renamed to `rancher-images-digests-linux.txt`.
- The `rancher-images-digests-windows-ltsc2019.txt` and `rancher-images-digests-windows-ltsc2022.txt` files are to be renamed to `rancher-images-digests-windows.txt`.

## Upgrade Requirements

- **Creating backups:** Create a backup before you upgrade Rancher. To roll back Rancher after an upgrade, you must first back up and restore Rancher to the previous Rancher version. Because Rancher will be restored to the same state as when the backup was created, any changes post-upgrade will not be included after the restore.
- **Helm version requirements:**
  - To manage Rancher 2.12.x and later, you must upgrade your Helm client to version 3.18 or newer.
  - This change is required to reflect the addition of Kubernetes 1.33 support with this release.
  - Currently, the official Helm Version Support Policy [https://helm.sh/docs/topics/version_skew/] dictates that only Helm 3.18 supports the proper Kubernetes version range for Rancher 2.12.
- **CNI requirements:**
  - For Kubernetes v1.19 and later, disable firewalld as it's incompatible with various CNI plugins. See #28840 [https://github.com/rancher/rancher/issues/28840].
  - When upgrading or installing a Linux distribution that uses nf_tables as the backend packet filter, such as SLES 15, RHEL 8, Ubuntu 20.10, Debian 10, or later, upgrade to RKE v1.19.2 or later to get Flannel v0.13.0. Flannel v0.13.0 supports nf_tables. See Flannel #1317 [https://github.com/flannel-io/flannel/issues/1317].

- **Requirements for air-gapped environments:**
  - When using a proxy in front of an air-gapped Rancher instance, you must pass additional parameters to `NO_PROXY`. See the documentation and issue #2725 [https://github.com/rancher/docs/issues/2725#issuecomment-702454584].
  - When installing Rancher with Docker in an air-gapped environment, you must supply a custom `registries.yaml` file to the `docker run` command, as shown in the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/private-registry.html]. If the registry has certificates, then you'll also need to supply those. See #28969 [https://github.com/rancher/rancher/issues/28969#issuecomment-694474229].

## Versions

### Images

- rancher/rancher:v2.15.3

### Tools

- CLI - v2.15.3 [https://github.com/rancher/cli/releases/tag/v2.15.3]

### Kubernetes Versions for RKE2/K3s

- v1.36.5 (Default)
- v1.35.9
- v1.34.12

### Rancher Helm Chart Versions

In Rancher v2.6.0 and later, in the **Apps & Marketplace** UI, many Rancher Helm charts are named with a major version that starts with *100*. This avoids simultaneous upstream changes and Rancher changes from causing conflicting version increments. This also complies with semantic versioning (SemVer), which is a requirement for Helm. You can see the upstream version number of a chart in the build metadata, for example: `100.0.0+up2.1.0`. See #32294 [https://github.com/rancher/rancher/issues/32294].

## Previous Rancher Behavior Changes

### Previous Rancher Behavior Changes - Rancher General

- **Rancher v2.15.0:**
  - Rancher v2.15 removes support for Kubernetes v1.33. See #55306 [https://github.com/rancher/rancher/issues/55306].

### Previous Rancher Behavior Changes Changes - Rancher App (Global UI)

- **Rancher v2.15.0:**
  - The option to disable the feature flag `ui-sql-cache` / Vai / Server-Side Pagination (SSP) in the Rancher UI has been removed in preparation for Rancher v2.16, where the feature will always be enabled. See #16822 [https://github.com/rancher/dashboard/issues/16822].
  - **Removal**: Support for Ember-based UI plugins for cluster and node drivers (deprecated since

Rancher v2.11.0) is removed in Rancher v2.15.0. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see #14005 [https://github.com/rancher/dashboard/issues/14005].

#### Previous Rancher Behavior Changes - K3s and RKE2 Provision

- **Rancher v2.15.0:**
  - By default, the `Standard User` and `Create Clusters` global permissions now allow creating and then updating CAPI infrastructure objects in the `fleet-default` namespace. Note that manually created infrastructure provider identity objects, such as `AWSClusterStaticIdentity` or `VSphereClusterIdentity`, that target the `fleet-default` namespace can be used through CAPI infrastructure objects such as `AWSCluster` or `VsphereCluster`. See #53777 [https://github.com/rancher/rancher/issues/53777].
  - When using native CAPI infrastructure providers with v2prov, Rancher will now generate the bootstrap cloud-config in the Jinja format [https://docs.cloud-init.io/en/latest/explanation/format/jinja.html]. User-data for regular node drivers with v2prov is unaffected. See #53777 [https://github.com/rancher/rancher/issues/53777].

#### Previous Rancher Behavior Changes - Continuous Delivery (Fleet)

- **Rancher v2.15.0:**
  - **Deprecation**: Fleet will be removing support for `GitRepoRestriction` in a future release, making the use of the Policy resource for configuring multi-tenant environments the only option. See #55652 [https://github.com/rancher/rancher/issues/55652].
  - **Deprecation**: Fleet support for unauthenticated webhook calls will be removed in a future release, making the setup of a webhook secret [https://documentation.suse.com/cloudnative/continuous-delivery/latest/en/how-tos-for-users/webhook.html#_3_optional_configure_a_webhook_secret] the only option. See #55633 [https://github.com/rancher/rancher/issues/55633]. == Long-standing Known Issues

#### Long-standing Known Issues - Cluster Provisioning

- **Rancher v2.15.1:**
  - Creating or editing v2prov clusters with native CAPI infrastructure providers from the UI can under some failure conditions leave unused infrastructure resources in the management cluster that must be cleaned up manually. See #55705 [https://github.com/rancher/rancher/issues/55705].
- **Rancher v2.15.0:**
  - Imported clusters cannot reconnect to Rancher if their registration token is rotated while they are disconnected and the grace period ends before they reconnect. The cluster agent logs `400 Bad Request` with the response body `cluster not found`. As a workaround, re-run the registration command from the cluster **Registration** tab on the downstream cluster. To prevent the issue, increase the `crt-default-grace-period-minutes` setting to cover the expected downtime, or disable the `crt-token-ttl-rotation` feature flag. See #57078 [https://github.com/rancher/rancher/issues/57078].
- **Rancher v2.14.0:**
  - Upgrading from Rancher v2.13.x to v2.14 or v2.15 leaves the old `rke2-machineconfig-cleanup` CronJob, ServiceAccount, and ConfigMap in `fleet-default`. Rancher also creates these resources in `fleet-local`, leaving two CronJobs with the same name. The old CronJob uses the v2.13.x agent image, contains a copy of the local cluster registration token, and runs without permissions. To work

around this issue, verify each old object has the annotation `objectset.rio.cattle.io/id=rke2-machine-config-cleanup`. Then delete the old CronJob, ServiceAccount, and ConfigMap from `fleet-default`. Do not delete the shared ClusterRole or ClusterRoleBinding because the `fleet-local` CronJob uses them. See #58290 [https://github.com/rancher/rancher/issues/58290].

- Not all cluster tools can be installed on a hardened cluster.
- **Rancher v2.8.1:**
  - When you attempt to register a new etcd/controlplane node in a CAPR-managed cluster after a failed etcd snapshot restoration, the node can become stuck in a perpetual paused state, displaying the error message `[ERROR] 000 received while downloading Rancher connection information. Sleeping for 5 seconds and trying again`. As a workaround, you can unpause the cluster by running `kubectl edit clusters.cluster clustername -n fleet-default` and set `spec.unpaused` to `false`. See #43735 [https://github.com/rancher/rancher/issues/43735].

#### Long-standing Known Issues - RKE2 Provisioning

- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #42834 [https://github.com/rancher/rancher/issues/42834].
- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].

#### Long-standing Known Issues - K3s Provisioning

- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].
- **Rancher v2.7.2:**
  - Clusters remain in an `Updating` state even when they contain nodes in an `Error` state. See #39164 [https://github.com/rancher/rancher/issues/39164].

#### Long-standing Known Issues - Rancher App (Global UI)

- **Rancher v2.10.0:**
  - After deleting a Namespace or Project in the Rancher UI, the Namespace or Project remains visible. As a workaround, refresh the page. See #12220 [https://github.com/rancher/dashboard/issues/12220].
- **Rancher v2.9.2:**
  - Although system mode node pools must have at least one node, the Rancher UI allows a minimum node count of zero. Inputting a zero minimum node count through the UI can cause cluster creation to fail due to an invalid parameter error. To prevent this error from occurring, enter a minimum node count at least equal to the node count. See #11922 [https://github.com/rancher/dashboard/issues/11922].

- **Rancher v2.7.7:**
  - When creating a cluster in the Rancher UI it does not allow the use of an underscore `_` in the `Cluster Name` field. See #9416 [https://github.com/rancher/dashboard/issues/9416].
- **Rancher v2.6.4:**
  - Deployment securityContext section is missing when a new workload is created. This prevents pods from starting when Pod Security Policy Support is enabled. See #4815 [https://github.com/rancher/dashboard/issues/4815].

#### Long-standing Known Issues - EKS

- **Rancher v2.7.0:**
  - EKS clusters on Kubernetes v1.21 or below on Rancher v2.7 cannot be upgraded. See #39392 [https://github.com/rancher/rancher/issues/39392].

#### Long-standing Known Issues - Authentication

- **Rancher v2.9.0:**
  - There are some known issues with the OpenID Connect provider support:
    - When the generic OIDC auth provider is enabled, and you attempt to add auth provider users to a cluster or project, users are not populated in the dropdown search bar. This is expected behavior as the OIDC auth provider alone is not searchable. See #46104 [https://github.com/rancher/rancher/issues/46104].
    - When the generic OIDC auth provider is enabled, auth provider users that are added to a cluster/project by their username are not able to access resources upon logging in. A user will only have access to resources upon login if the user is added by their userID. See #46105 [https://github.com/rancher/rancher/issues/46105].
    - When the generic OIDC auth provider is enabled and an auth provider user in a nested group is logged into Rancher, the user will see the following error when they attempt to create a Project: `projectroletemplatebindings.management.cattle.io is forbidden: User "u-gcxatwsnku" cannot create resource "projectroletemplatebindings" in API group "management.cattle.io" in the namespace "p-9t5pg"`. However, the project is still created. See #46106 [https://github.com/rancher/rancher/issues/46106].

#### Long-standing Known Issues - Rancher Webhook

- **Rancher v2.7.2:**
  - A webhook is installed in all downstream clusters. There are several issues that users may encounter with this functionality:
    - If you rollback from a version of Rancher v2.7.2 or later, to a Rancher version earlier than v2.7.2, the webhooks will remain in downstream clusters. Since the webhook is designed to be 1:1 compatible with specific versions of Rancher, this can cause unexpected behaviors to occur downstream. The Rancher team has developed a script [https://github.com/rancher/webhook/wiki/Remove-Webhook-from-downstream-clusters] which should be used after rollback is complete (meaning after a Rancher version earlier than v2.7.2 is running). This removes the webhook from affected downstream clusters. See #40816 [https://github.com/rancher/rancher/issues/40816].

#### Long-standing Known Issues - Virtualization Management (Harvester)

- **Rancher v2.13.1:**
  - When upgrading to Rancher v2.13.1 or higher while using Harvester v1.6.1, users may encounter an issue with their Load Balancers in downstream clusters using the Harvester Cloud Provider, and must perform the following workaround [https://documentation.suse.com/cloudnative/virtualization/latest/en/troubleshooting/rancher.html#_guest_cluster_load_balancer_ip_is_not_reachable] to instruct Calico to not use any of the IP/interface managed by `kube-vip`. See #9767 [https://github.com/harvester/harvester/issues/9767].
- **Rancher v2.7.2:**
  - If you're using Rancher v2.7.2 with Harvester v1.1.1 clusters, you won't be able to select the Harvester cloud provider when deploying or updating guest clusters. The Harvester release notes [https://github.com/harvester/release-notes/blob/main/v1.1.2.md#important-information-about-rancher-support] contain instructions on how to resolve this. See #3750 [https://github.com/harvester/harvester/issues/3750].

#### Long-standing Known Issues - Backup/Restore

- When migrating to a cluster with the Rancher Backup feature, the server-url cannot be changed to a different location. It must continue to use the same URL.
- **Rancher v2.13.0:**
  - When performing a rollback from Rancher v2.13.0 to v2.12.3 using the backup and restore operator (BRO), the restore does not complete successfully. See #844 [https://github.com/rancher/backup-restore-operator/issues/844]. To work around this issue, you must scale down your Rancher deployment and uninstall the Webhook chart before performing the restore. For details, refer to this Knowledge Base article [https://support.scc.suse.com/s/kb/Rolling-back-from-Rancher-v2-13-0-to-v2-12-3?language=en_US].
- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #42834 [https://github.com/rancher/rancher/issues/42834].

# Release v2.15.2



- If you are using Active Directory Federation Service (AD FS), upgrading to Rancher v2.10.1 or later may cause issues with authentication, requiring manual intervention. These issues are due to the AD FS Relying Party Trust not being able to pick up a signature verification certificate from the metadata. For more information, see #48655 [https://github.com/rancher/rancher/issues/48655]. These issues can be corrected by either of two methods:
  - Updating the Relying Party Trust information from federation metadata (Relying Party Trust → Update from Federation Metadata...).
  - Directly adding the certificate (Relying Party Trust → Properties → Signature tab → Add → Select the certificate).

Rancher v2.15.2 is the latest patch release of Rancher. This is a Prime version release that introduces maintenance updates and bug fixes. To learn more about Rancher Prime, see our page on the Rancher Prime

Platform [https://www.rancher.com/products/rancher-platform].

## Security Fixes for Vulnerabilities

This release addresses the following security issues:

### Rancher

- Fixed a security vulnerability in Rancher where logging out cleared the browser session cookies but did not delete the corresponding public API session token on the server. A previously captured session token therefore remained valid until its natural expiry, defeating logout as a revocation control. Only deployments using public API session tokens in, for example, OIDC-based authentication, were affected. For more information, refer to CVE-2026-88805 [https://github.com/rancher/rancher/security/advisories/GHSA-6vpq-mf48-9794].
- Fixed a security vulnerability in Rancher where a small set of public UI settings served by unauthenticated, read-only API routes could be modified by a remote unauthenticated user. Because one of those settings is rendered as raw HTML on the login page, attacker-controlled content could be stored and executed in the browser of users accessing the login page, potentially exposing the local administrator bootstrap password or an active administrator session. The unauthenticated settings routes are now strictly read-only. For more information, see CVE-2026-88804 [https://github.com/rancher/rancher/security/advisories/GHSA-992f-xh8r-jg2f].

### Fleet

- Fixed a security vulnerability in Fleet where Helm `valuesFiles` entries in `fleet.yaml` were not confined to the bundle directory. A user able to supply untrusted bundle content could cause Fleet to read files outside the repository checkout and expose their contents through the generated `Bundle`, including credential material the user had no permission to read. Fleet now rejects values file paths that escape the bundle directory. For more information, refer to CVE-2026-93537 [https://github.com/rancher/fleet/security/advisories/GHSA-wpfm-r97v-3j4h].
- Fixed an issue where the Fleet agent copied the Secrets and ConfigMaps listed in a Bundle's `downstreamResources`, and created the deployment's target namespace, using the agent's own credentials instead of the ServiceAccount pinned to the deployment. Those writes are now gated by the tenant's downstream RBAC. Deployments that do not pin a ServiceAccount are unchanged as an agent operation. Rancher v2.15.x is affected in a default installation, because the resource copy is always enabled there. After upgrading, a deployment that pins a ServiceAccount without permission to create the target namespace, or to write the copied Secrets and ConfigMaps in it, will no longer deploy. It reports "not ready" and retries until the permission is granted. Grant the ServiceAccount the missing permissions, or pre-create the target namespace. For more information, refer to CVE-2026-88808 [https://github.com/rancher/fleet/security/advisories/GHSA-q9v4-358v-r8q5].
- Fixed a security vulnerability in Fleet's Git webhook receiver where when a webhook secret was not configured, an unauthenticated caller with network access to the webhook service could change the polling interval of `GitRepo` resources in any namespace, delaying the rollout of legitimate Git changes. Fleet now only applies webhook-triggered changes to `GitRepo` specifications for requests verified against a configured webhook secret. For more information, refer to CVE-2026-93539 [https://github.com/rancher/fleet/security/advisories/GHSA-8vfv-33cg-g75q].

## Rancher General

### Major Bug Fixes

- Fixed a known issue with Kubernetes v1.36 clusters where **Rancher Backups** is not listed under the **Cluster Tools** due to a Backups chart issue. See #1055 [https://github.com/rancher/backup-restore-operator/pull/1055].

## Authentication

### Major Bug Fixes

- The ext.Kubeconfig store has been updated to align with standard Kubernetes API resource behavior. See #56341 [https://github.com/rancher/rancher/issues/56341].

## Role-Based Access Control (RBAC)

### Major Bug Fixes

- Fixed the reconciler loop for Global Roles that contain the `inheritedFleetWorkspacePermissions` field. The fix prevents Rancher logs from being spammed with create/delete operations. See #57050 [https://github.com/rancher/rancher/issues/57050].
- Extended Rancher to better protect its managed resources for limit ranges and resource quotas from interference by regular users.
  - The webhook now rejects attempts from outside of Rancher to edit these managed resources, to promote an unmanaged resource to managed, or to demote a managed resource to unmanaged, while still allowing Rancher itself full access. See #56773 [https://github.com/rancher/rancher/issues/56773].

## Cluster Provisioning

### Major Bug Fixes

- Fixed an issue where the GKE provisioning UI offered Kubernetes versions that are not supported by any GKE release channel. Rancher now shows only valid GKE versions for release-channel-based provisioning, preventing users from selecting versions that cannot be used to provision new clusters. See #18928 [https://github.com/rancher/dashboard/issues/18928] and #56879 [https://github.com/rancher/rancher/issues/56879].

### Known Issues

- There is a known issue with provisioned clusters deletion where such clusters get stuck if they have s3 snapshots. A workaround is to delete all left over s3 `etcdsnapshots.rke.cattle.io` objects belonging to the cluster that are present within the local cluster. See #57506 [https://github.com/rancher/rancher/issues/57506] and the issue comment [https://github.com/rancher/rancher/issues/57506#issuecomment-5784025617] for more information.

## Changes Since v2.15.1

See the full list of changes [https://github.com/rancher/rancher/compare/v2.15.1%E2%80%A6v2.15.2].

## Install/Upgrade Notes

If you're installing Rancher for the first time, your environment must fulfill the installation requirements.

- Chart name change for Rancher Prime.
  - The chart name change introduced in Rancher Prime v2.13.1 has been reverted. The chart name `rancher` should be used for all installations and upgrades. As an example, the installation command is now `helm install rancher rancher-prime/rancher`.
- Rancher now requires the cluster it runs on to have the Kubernetes API Aggregation Layer [https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/] enabled. This is because Rancher extends Kubernetes with additional APIs by registering its own extension API server. Please note that all versions of Kubernetes supported in this Rancher versions K8s distributions (RKE2/K3s) will have the aggregation layer configured and enabled by default.
  - Refer to the Extension API Server documentation and #50400 [https://github.com/rancher/rancher/issues/50400] for more information.
- Rancher Kubernetes Engine (RKE/RKE1) has reached end of life as of **July 31, 2025**. Rancher versions 2.12.0 and later no longer support provisioning or managing downstream RKE1 clusters. We recommend replatforming RKE1 clusters to RKE2 to ensure continued support and security updates. Learn more about the transition here [https://www.suse.com/support/kb/doc/?id=000021518].



  - Rancher now has a pre-upgrade validation check for RKE1 resources which fails and lists the RKE1 resources if present. Refer to #50286 [https://github.com/rancher/rancher/issues/50286] for more information.
- It is crucial that you review the available disk space on your nodes and plan accordingly before upgrading to Rancher v2.12.0 and later to avoid potential disk pressure and pod eviction issues.
  - For additional information, refer to the UI Server Side Pagination - Disk Space documentation.
- Rancher now has an enablement option called `AUDIT_LOG_ENABLED` for API Audit Logs for a Rancher installation.
  - In Rancher versions 2.11.x and earlier, only the `AUDIT_LEVEL` could be set and the default log level (`0`) would disable the audit log. In Rancher versions 2.12.x and later, the default log level (`0`) now only contains the log request and response metadata, and can be set when configuring `AUDIT_LOG_ENABLED`. If installing or upgrading via Helm you can enable the API Audit Logs and specify the log level by applying the following setting to your Helm command: `--set auditLog.enabled=true --set auditLog.level=0`. See the Enabling the API Audit Log to Record System Events documentation and #48941 [https://github.com/rancher/rancher/issues/48941].

## Changes in Image Artifacts

Image artifact digests are renamed in Rancher v2.12.0, v2.11.4 and v2.10.8. Up until this change, separate image digests files for each operating system and architecture have been maintained for compatibility reasons. With this change, only one file for each operating system is to be provided:

- The `rancher-images-digests-linux-amd64.txt` and `rancher-images-digests-linux-arm64.txt` files are to be renamed to `rancher-images-digests-linux.txt`.
- The `rancher-images-digests-windows-ltsc2019.txt` and `rancher-images-digests-windows-ltsc2022.txt` files are to be renamed to `rancher-images-digests-windows.txt`.

## Upgrade Requirements

- **Creating backups:** Create a backup before you upgrade Rancher. To roll back Rancher after an upgrade, you must first back up and restore Rancher to the previous Rancher version. Because Rancher will be restored to the same state as when the backup was created, any changes post-upgrade will not be included after the restore.
- **Helm version requirements:**
  - To manage Rancher 2.12.x and later, you must upgrade your Helm client to version 3.18 or newer.
  - This change is required to reflect the addition of Kubernetes 1.33 support with this release.
  - Currently, the official Helm Version Support Policy [https://helm.sh/docs/topics/version_skew/] dictates that only Helm 3.18 supports the proper Kubernetes version range for Rancher 2.12.
- **CNI requirements:**
  - For Kubernetes v1.19 and later, disable firewalld as it's incompatible with various CNI plugins. See #28840 [https://github.com/rancher/rancher/issues/28840].
  - When upgrading or installing a Linux distribution that uses nf_tables as the backend packet filter, such as SLES 15, RHEL 8, Ubuntu 20.10, Debian 10, or later, upgrade to RKE v1.19.2 or later to get Flannel v0.13.0. Flannel v0.13.0 supports nf_tables. See Flannel #1317 [https://github.com/flannel-io/flannel/issues/1317].
- **Requirements for air-gapped environments:**
  - When using a proxy in front of an air-gapped Rancher instance, you must pass additional parameters to `NO_PROXY`. See the documentation and issue #2725 [https://github.com/rancher/docs/issues/2725#issuecomment-702454584].
  - When installing Rancher with Docker in an air-gapped environment, you must supply a custom `registries.yaml` file to the `docker run` command, as shown in the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/private-registry.html]. If the registry has certificates, then you'll also need to supply those. See #28969 [https://github.com/rancher/rancher/issues/28969#issuecomment-694474229].

## Versions

### Images

- rancher/rancher:v2.15.2

### Tools

- CLI - v2.15.2 [https://github.com/rancher/cli/releases/tag/v2.15.2]

### Kubernetes Versions for RKE2/K3s

- v1.36.4 (Default)
- v1.35.8

- v1.34.11

#### Rancher Helm Chart Versions

In Rancher v2.6.0 and later, in the **Apps & Marketplace** UI, many Rancher Helm charts are named with a major version that starts with *100*. This avoids simultaneous upstream changes and Rancher changes from causing conflicting version increments. This also complies with semantic versioning (SemVer), which is a requirement for Helm. You can see the upstream version number of a chart in the build metadata, for example: `100.0.0+up2.1.0`. See #32294 [https://github.com/rancher/rancher/issues/32294].

## Previous Rancher Behavior Changes

#### Previous Rancher Behavior Changes - Rancher General

- **Rancher v2.15.0:**
  - Rancher v2.15 removes support for Kubernetes v1.33. See #55306 [https://github.com/rancher/rancher/issues/55306].

#### Previous Rancher Behavior Changes Changes - Rancher App (Global UI)

- **Rancher v2.15.0:**
  - The option to disable the feature flag `ui-sql-cache` / Vai / Server-Side Pagination (SSP) in the Rancher UI has been removed in preparation for Rancher v2.16, where the feature will always be enabled. See #16822 [https://github.com/rancher/dashboard/issues/16822].
  - **Removal**: Support for Ember-based UI plugins for cluster and node drivers (deprecated since Rancher v2.11.0) is removed in Rancher v2.15.0. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see #14005 [https://github.com/rancher/dashboard/issues/14005].

#### Previous Rancher Behavior Changes - K3s and RKE2 Provision

- **Rancher v2.15.0:**
  - By default, the `Standard User` and `Create Clusters` global permissions now allow creating and then updating CAPI infrastructure objects in the `fleet-default` namespace. Note that manually created infrastructure provider identity objects, such as `AWSClusterStaticIdentity` or `VSphereClusterIdentity`, that target the `fleet-default` namespace can be used through CAPI infrastructure objects such as `AWSCluster` or `VsphereCluster`. See #53777 [https://github.com/rancher/rancher/issues/53777].
  - When using native CAPI infrastructure providers with v2prov, Rancher will now generate the bootstrap cloud-config in the Jinja format [https://docs.cloud-init.io/en/latest/explanation/format/jinja.html]. User-data for regular node drivers with v2prov is unaffected. See #53777 [https://github.com/rancher/rancher/issues/53777].

#### Previous Rancher Behavior Changes - Continuous Delivery (Fleet)

- **Rancher v2.15.0:**
  - **Deprecation**: Fleet will be removing support for `GitRepoRestriction` in a future release, making the use of the Policy resource for configuring multi-tenant environments the only option. See #55652 [https://github.com/rancher/rancher/issues/55652].

- **Deprecation**: Fleet support for unauthenticated webhook calls will be removed in a future release, making the setup of a webhook secret [https://documentation.suse.com/cloudnative/continuous-delivery/latest/en/how-tos-for-users/webhook.html#_3_optional_configure_a_webhook_secret] the only option. See #55633 [https://github.com/rancher/rancher/issues/55633]. == Long-standing Known Issues

#### Long-standing Known Issues - Cluster Provisioning

- **Rancher v2.15.1:**
  - Creating or editing v2prov clusters with native CAPI infrastructure providers from the UI can under some failure conditions leave unused infrastructure resources in the management cluster that must be cleaned up manually. See #55705 [https://github.com/rancher/rancher/issues/55705].
- Not all cluster tools can be installed on a hardened cluster.
- **Rancher v2.8.1:**
  - When you attempt to register a new etcd/controlplane node in a CAPR-managed cluster after a failed etcd snapshot restoration, the node can become stuck in a perpetual paused state, displaying the error message `[ERROR] 000 received while downloading Rancher connection information. Sleeping for 5 seconds and trying again`. As a workaround, you can unpause the cluster by running `kubectl edit clusters.cluster clustername -n fleet-default` and set `spec.unpaused` to `false`. See #43735 [https://github.com/rancher/rancher/issues/43735].

#### Long-standing Known Issues - RKE2 Provisioning

- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #34518 [https://github.com/rancher/rancher/issues/34518] and #42834 [https://github.com/rancher/rancher/issues/42834].
- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].

#### Long-standing Known Issues - K3s Provisioning

- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].
- **Rancher v2.7.2:**
  - Clusters remain in an `Updating` state even when they contain nodes in an `Error` state. See #39164 [https://github.com/rancher/rancher/issues/39164].

#### Long-standing Known Issues - Rancher App (Global UI)

- **Rancher v2.10.0:**

- After deleting a Namespace or Project in the Rancher UI, the Namespace or Project remains visible. As a workaround, refresh the page. See #12220 [https://github.com/rancher/dashboard/issues/12220].

- **Rancher v2.9.2:**
  - Although system mode node pools must have at least one node, the Rancher UI allows a minimum node count of zero. Inputting a zero minimum node count through the UI can cause cluster creation to fail due to an invalid parameter error. To prevent this error from occurring, enter a minimum node count at least equal to the node count. See #11922 [https://github.com/rancher/dashboard/issues/11922].

- **Rancher v2.7.7:**
  - When creating a cluster in the Rancher UI it does not allow the use of an underscore `_` in the `Cluster Name` field. See #9416 [https://github.com/rancher/dashboard/issues/9416].

- **Rancher v2.6.4:**
  - Deployment securityContext section is missing when a new workload is created. This prevents pods from starting when Pod Security Policy Support is enabled. See #4815 [https://github.com/rancher/dashboard/issues/4815].

#### Long-standing Known Issues - EKS

- **Rancher v2.7.0:**
  - EKS clusters on Kubernetes `v1.21` or below on Rancher v2.7 cannot be upgraded. See #39392 [https://github.com/rancher/rancher/issues/39392].

#### Long-standing Known Issues - Authentication

- **Rancher v2.9.0:**
  - There are some known issues with the OpenID Connect provider support:
    - When the generic OIDC auth provider is enabled, and you attempt to add auth provider users to a cluster or project, users are not populated in the dropdown search bar. This is expected behavior as the OIDC auth provider alone is not searchable. See #46104 [https://github.com/rancher/rancher/issues/46104].
    - When the generic OIDC auth provider is enabled, auth provider users that are added to a cluster/project by their username are not able to access resources upon logging in. A user will only have access to resources upon login if the user is added by their userID. See #46105 [https://github.com/rancher/rancher/issues/46105].
    - When the generic OIDC auth provider is enabled and an auth provider user in a nested group is logged into Rancher, the user will see the following error when they attempt to create a Project: `projectroletemplatebindings.management.cattle.io is forbidden: User "u-gcxatwsnku" cannot create resource "projectroletemplatebindings" in API group "management.cattle.io" in the namespace "p-9t5pg"`. However, the project is still created. See #46106 [https://github.com/rancher/rancher/issues/46106].

#### Long-standing Known Issues - Rancher Webhook

- **Rancher v2.7.2:**
  - A webhook is installed in all downstream clusters. There are several issues that users may encounter with this functionality:
    - If you rollback from a version of Rancher v2.7.2 or later, to a Rancher version earlier than v2.7.2,

the webhooks will remain in downstream clusters. Since the webhook is designed to be 1:1 compatible with specific versions of Rancher, this can cause unexpected behaviors to occur downstream. The Rancher team has developed a script [https://github.com/rancher/webhook/wiki/Remove-Webhook-from-downstream-clusters] which should be used after rollback is complete (meaning after a Rancher version earlier than v2.7.2 is running). This removes the webhook from affected downstream clusters. See #40816 [https://github.com/rancher/rancher/issues/40816].

**Long-standing Known Issues - Virtualization Management (Harvester)**

- **Rancher v2.13.1:**
  - When upgrading to Rancher v2.13.1 or higher while using Harvester v1.6.1, users may encounter an issue with their Load Balancers in downstream clusters using the Harvester Cloud Provider, and must perform the following workaround [https://documentation.suse.com/cloudnative/virtualization/latest/en/troubleshooting/rancher.html#_guest_cluster_load_balancer_ip_is_not_reachable] to instruct Calico to not use any of the IP/interface managed by `kube-vip`. See #9767 [https://github.com/harvester/harvester/issues/9767].
- **Rancher v2.7.2:**
  - If you're using Rancher v2.7.2 with Harvester v1.1.1 clusters, you won't be able to select the Harvester cloud provider when deploying or updating guest clusters. The Harvester release notes [https://github.com/harvester/release-notes/blob/main/v1.1.2.md#important-information-about-rancher-support] contain instructions on how to resolve this. See #3750 [https://github.com/harvester/harvester/issues/3750].

**Long-standing Known Issues - Backup/Restore**

- When migrating to a cluster with the Rancher Backup feature, the server-url cannot be changed to a different location. It must continue to use the same URL.
- **Rancher v2.13.0:**
  - When performing a rollback from Rancher v2.13.0 to v2.12.3 using the backup and restore operator (BRO), the restore does not complete successfully. See #844 [https://github.com/rancher/backup-restore-operator/issues/844]. To work around this issue, you must scale down your Rancher deployment and uninstall the Webhook chart before performing the restore. For details, refer to this Knowledge Base article [https://support.scc.suse.com/s/kb/Rolling-back-from-Rancher-v2-13-0-to-v2-12-3?language=en_US].
- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #34518 [https://github.com/rancher/rancher/issues/34518] and #42834 [https://github.com/rancher/rancher/issues/42834].

# Release v2.15.1



- If you are using Active Directory Federation Service (AD FS), upgrading to Rancher v2.10.1 or later may cause issues with authentication, requiring manual intervention. These issues are due to the AD FS Relying Party Trust not being able to pick up a signature verification certificate from the metadata. For more information, see #48655 [https://github.com/rancher/rancher/issues/48655]. These issues can be corrected by either of two methods:

- Updating the Relying Party Trust information from federation metadata (Relying Party Trust → Update from Federation Metadata...).
- Directly adding the certificate (Relying Party Trust → Properties → Signature tab → Add → Select the certificate).

Rancher v2.15.1 is the latest patch release of Rancher. This is a Prime version release that introduces new features, enhancements, and bug fixes. To learn more about Rancher Prime, see our page on the Rancher Prime Platform [https://www.rancher.com/products/rancher-platform].

## What's New - v2.15.1

- **Kubernetes Support and Lifecycle Updates**
  - Rancher v2.15.1 adds support for **Kubernetes v1.36** while removing support for Kubernetes v1.33.
  - **Rancher App Chart Retention Policy**: Enforces a new lifecycle policy preserving chart versions in the "Rancher" repository for the **7 most recent minor releases** (approx. 2.5 years). Older chart versions have been removed.
- **Cluster Provisioning**
  - **Cluster API (CAPI) GA**: Support for native CAPI infrastructure providers within the V2 provisioning framework (v2prov) has been promoted to **General Availability (GA)**.
  - Updated CAPI support to **v1.13.2**.
- **Authentication**
  - **Inherited Namespace Rules for GlobalRoles**: Administrators can now define `inheritedNamespacedRules` in GlobalRoles to automatically propagate specific namespace RBAC permissions across all downstream clusters.
  - **Azure AD SSO Logout**: Added support for Single Logout (SLO) with Azure AD, allowing users to terminate their Azure AD session directly upon logging out of Rancher, with support for custom sovereign-cloud endpoints.
  - **Hide Local Auth Provider**: Introduced a new `hide-local-auth-provider` feature flag to restrict local UI authentication once an External Authentication Provider (EAP) is configured.
- **Dashboard**
  - **Workloads Overview UI**: Replaces the legacy all-in-one Workloads list in the Rancher Dashboard with a high-performance **Workloads Overview** UI view designed for large-scale environments.

## Security Fixes for Vulnerabilities

This release addresses the following security issues:

### Rancher

- Fixed a security vulnerability in Rancher's SAML authentication handler where the assertion replay protection was ineffective in high-availability (multi-replica) deployments. Because each replica tracked consumed assertion IDs in a separate in-memory cache, a captured SAML assertion could be replayed against other replicas to create additional authenticated sessions as the victim. The fix tracks consumed assertion IDs across all replicas. All SAML providers were affected. For more information, refer to CVE-2026-75034 [https://github.com/rancher/rancher/security/advisories/GHSA-j637-8xgr-436x].
- Fixes a security vulnerability in the Rancher project-scoped `secretsController` where Project

Secrets propagate to namespaces based only on the `field.cattle.io/projectId` annotation without validating that the referenced project belongs to the same cluster. An authenticated user with namespace creation permissions on one downstream cluster could spoof this annotation to leak Project Secrets from an unauthorized project in a different cluster. The controller now validates that the annotation references a project within the same cluster before propagating secrets. For more information, see CVE-2026-75033 [https://github.com/rancher/rancher/security/advisories/GHSA-5hf4-f4mp-g6h4].

- Fixed a security vulnerability in the Rancher imperative ext Token API (`ext.cattle.io/v1` tokens) where any authenticated user can list and watch other users' tokens by supplying a crafted label selector, disclosing token metadata and the stored bearer token hash. The store now enforces per-user scoping unconditionally so callers see only their own tokens, and read responses no longer return the stored token hash. For more information, see CVE-2026-75035 [https://github.com/rancher/rancher/security/advisories/GHSA-h923-rr66-g2w5].
- Fixed a security vulnerability in Rancher's GlobalRole controller where a user with delegated `create` or `update` permissions for `GlobalRole` resources could set the `authz.management.cattle.io/cr-name` annotation to target an arbitrary existing `ClusterRole` (such as the built-in `cluster-admin`). The controller then overwrites the rules without an ownership check, stripping permissions from all bound principals and causing a persistent, cluster-wide RBAC lockout on the local management cluster. For more information, refer to CVE-2026-71404 [https://github.com/rancher/rancher/security/advisories/GHSA-92jp-phmj-wv65].
- Fixed a security vulnerability in the SUSE Rancher Norman `/v3/users` API where the update path did not enforce immutability of the `username`, `principalIds`, and `displayName` fields on a `User` resource. A caller holding `update` permissions on `users` could rebind a user's external principal identity, enabling account takeover during the next external login or creating a denial-of-service condition that left existing role bindings inaccessible. The API store and admission-webhook layers were updated to reject post-creation changes to these identity fields. For more information, see CVE-2026-71403 [https://github.com/rancher/rancher/security/advisories/GHSA-wfvm-w99r-cjgc].

**Fleet**

- Fixed a security vulnerability in Fleet's Helm template preprocessing where templates evaluated by the Fleet controller could reach network resources. A user who can supply untrusted bundle content can expose cluster metadata and information about hosts reachable from the controller. Fleet now rejects bundles whose Helm template values require network access, reporting an error for the corresponding `GitRepo`. For more information, refer to CVE-2026-75036 [https://github.com/rancher/fleet/security/advisories/GHSA-x9m6-xcjr-hpjp].
- Fixed a security vulnerability in Fleet where namespace labels and annotations requested by a bundle through `namespaceLabels` and `namespaceAnnotations` were applied without the authorization of the service account the bundle was pinned to. In multi-tenant deployments this allowed a bundle to modify namespace metadata that its own identity was not permitted to change. The namespace update now runs as the deployment's service account and is governed by downstream Kubernetes RBAC. For more information, refer to CVE-2026-93540 [https://github.com/rancher/fleet/security/advisories/GHSA-m93g-8438-2cgg].
- Fixed a security vulnerability in Fleet where cluster labels supplied by an agent during agent-initiated cluster registration were applied to the resulting `Cluster` object, including reserved `management.cattle.io/` labels used for bundle targeting. In deployments where multiple trust domains share a Fleet workspace namespace, this allowed a registering cluster to match another tenant's `Bundle` or `GitRepo` targets and receive that tenant's manifests and resolved values Secrets. Fleet now discards reserved labels supplied at agent-initiated registration. Standalone Rancher deployments are

not affected in their default configuration, as Rancher uses manager-initiated registration. For more information, refer to CVE-2026-93538 [https://github.com/rancher/fleet/security/advisories/GHSA-h9p5-fp5h-qpqr].

## Rancher General

### Features and Enhancements

- Rancher now supports Kubernetes v1.36. See #54303 [https://github.com/rancher/rancher/issues/54303] for information on Rancher support for Kubernetes v1.36. You can view the upstream Kubernetes changelog [https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.36.md] for v1.36 for a complete list of changes.
- Added a cap on the maximum number of public endpoint annotations (`field.cattle.io/publicEndpoints`) written for Ingress objects. This prevents workloads with extensive custom domain lists from exceeding Kubernetes' 256 KB metadata annotation size limit and triggering API validation errors. See #42058 [https://github.com/rancher/rancher/issues/42058].
- Optimized `rancher-images.txt` to include only the latest patch release per Kubernetes minor version compatible with the target Rancher release. Release Candidate (RC) pre-releases and superseded patch/minor versions are now filtered out, significantly reducing the size of air-gapped image tarballs while ensuring `system-agent-installer` and auxiliary images remain fully up to date. See #46174 [https://github.com/rancher/rancher/issues/46174].

### Behavior Changes

- Rancher v2.15 removes support for Kubernetes v1.33. See #55306 [https://github.com/rancher/rancher/issues/55306].

#### Retention Policy for Rancher App Charts

To improve repository performance, Rancher is introducing a lifecycle management policy for charts available in the Apps feature of Rancher, specifically in the "Rancher" repository.

- **The Policy:** Rancher will transition from a cumulative model (retaining all historical versions forever) to a retention model that preserves chart versions for the **seven (7) most recent Rancher minor releases** (approximately a 2.5-year window).
- **Timeline - Rancher v2.13 & v2.14:** Legacy chart versions (older than the 7-version window) remain available.
- **Rancher v2.15.1:** This release is the first version to enforce the policy. Versions falling outside the 7-version window and older than two years will no longer be available.

**Impact:** This change is non-destructive for existing Rancher installations. Historical versions will remain accessible but will not be available in newer release branches once they age out of the 7-version window. You are advised to upgrade your applications before upgrading to Rancher v2.15.1. Uninstallation after v2.15.1 and replacement with an updated version will still be possible.

### Major Bug Fixes

- Fixed an issue where `rancher-save-images.sh` failed to pull OCI-based Helm charts when the local Docker daemon was configured to use the `overlay2` storage driver. See #55068 [https://github.com/rancher/rancher/issues/55068].

#### Known Issues

- A known issue exists with Rancher v2.15.1 and Kubernetes v1.36 clusters where **Rancher Backups** is not listed under the **Cluster Tools** due to a Backups chart issue. See #1055 [https://github.com/rancher/backup-restore-operator/pull/1055].

## Rancher App (Global UI)

#### Features and Enhancements

- The Rancher Dashboard replaces the legacy all-in-one Workloads list with a new **Workloads Overview** UI view to solve major performance and scalability issues when viewing workload resources. See #11513 [https://github.com/rancher/dashboard/issues/11513].

##### Rancher Monitoring Dashboards



Starting in Rancher v2.15.1, new monitoring installations use the decoupled `rancher-monitoring-dashboards` chart.

Rancher maintains the legacy `rancher-monitoring` chart to support existing deployments, but it does not receive new updates. Use the `rancher-monitoring-dashboards` chart for ongoing updates and support.

The `rancher-monitoring-dashboards` chart provides Grafana dashboards and Rancher UI integration for external Prometheus monitoring stacks, such as `kube-prometheus-stack`.

- **No bundled runtime:** Operates without depending on Rancher-shipped `kube-prometheus-stack` images or components. It operates independently and does not require installing the legacy `rancher-monitoring` chart. It installs a dashboard-only version of `rancher-monitoring`.
- **Prerequisites:** Requires uninstalling the `rancher-monitoring` chart and using a Prometheus infrastructure. Although several configurations are supported, installing `kube-prometheus-stack` is the simplest method.

For more detailed information regarding installation and migrations check the Monitoring and Dashboards documentation, and refer to the evolution of SUSE Rancher Observability [https://www.suse.com/c/the-path-to-modern-observability-why-were-evolving-the-suse-rancher-observability-stack/] blog post for further context. See #221 [https://github.com/rancher/ob-team-charts/issues/221] and #265 [https://github.com/rancher/ob-team-charts/issues/265].

#### Behavior Changes

- The option to disable the feature flag `ui-sql-cache` / Vai / Server-Side Pagination (SSP) in the Rancher UI has been removed in preparation for Rancher v2.16, where the feature will always be enabled. See #16822 [https://github.com/rancher/dashboard/issues/16822].
- **Removal**: Support for Ember-based UI plugins for cluster and node drivers (deprecated since Rancher v2.11.0) is removed in Rancher v2.15.1. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see #14005 [https://github.com/rancher/dashboard/issues/14005].

#### Major Bug Fixes

- Fixed an issue where selecting **Enable Dual-Stack** with **Stack Preference** set to **IPv6** omitted the required UI warning. Dual-stack clusters require the **Stack Preference** option to be set to **Dual**. For

details, see #16788 [https://github.com/rancher/dashboard/issues/16788].

## Authentication

### Features and Enhancements

- **Azure AD SSO Logout**: Rancher now supports Single Logout (SLO) for the Azure AD auth provider. When enabled, logging out of Rancher also terminates the user's Azure AD session by redirecting to Azure AD's end-session endpoint. A `LogoutEndpoint` override field is available for sovereign-cloud deployments (Azure US Government, China). An optional `LogoutAllForced` flag can prevent users from bypassing SSO logout. See #54566 [https://github.com/rancher/rancher/issues/54566].
- Added the `hide-local-auth-provider` feature flag (disabled by default) to restrict local authentication usage once an External Authentication Provider (EAP) is active. When enabled, the flag hides the local login interface and removes the ability to create, edit, or delete local users from the UI. Note that password rotation for required administrative local accounts remains allowed to support emergency fallback workflows. See #46078 [https://github.com/rancher/rancher/issues/46078].

## Cluster Provisioning

### Features and Enhancements

- Rancher v2.15.1 adds support for Cluster API (CAPI) v1.13.2. See #54304 [https://github.com/rancher/rancher/issues/54304].
- Users with the `clusters-create` role can now view the CAPIprovider resource. See #55167 [https://github.com/rancher/rancher/issues/55167].
- Rancher now supports mixed-architecture (x86_64 and ARM64) Linux deployments for both the Rancher Management Server and downstream clusters.

### Known Issues

- Google Cloud has deprecated the "No channel" option for GKE clusters as of August 21, 2026. All clusters must now use a release channel [https://docs.cloud.google.com/kubernetes-engine/docs/concepts/release-channels#channels]. Currently, GCP's automatic out-of-band upgrades can cause state drift in Rancher, which may block cluster edits, break node pool updates, or cause provisioning failures. An upcoming out-of-band GKE chart release will prevent these issues by automatically assigning new clusters to a release channel by default. See #56879 [https://github.com/rancher/rancher/issues/56879].

## K3s and RKE2 Provisioning

### Features and Enhancements

- **General Availability**: Promoted support for native Cluster API (CAPI) infrastructure providers within Rancher's V2 provisioning framework (v2prov) from Tech Preview to General Availability (GA). See #53777 [https://github.com/rancher/rancher/issues/53777].
  - If the CAPA infrastructure provider is installed, Rancher will mirror cloud credentials for AWS and create a corresponding `AWSClusterStaticIdentity` object that targets the `fleet-default` namespace.
- Rancher introduces a new `rke2-provisioning-prime-default` setting, which defaults to `true`. This setting enables the RKE2 `prime` server argument for newly provisioned clusters, ensuring that new

clusters use Prime-specific images and chart values by default. Existing clusters remain unchanged after an upgrade. You can override this behavior per cluster by using a new annotation. See #54336 [https://github.com/rancher/rancher/issues/54336].

#### Behavior Changes

- By default, the `Standard User` and `Create Clusters` global permissions now allow creating and then updating CAPI infrastructure objects in the `fleet-default` namespace. Note that manually created infrastructure provider identity objects, such as `AWSClusterStaticIdentity` or `VSphereClusterIdentity`, that target the `fleet-default` namespace can be used through CAPI infrastructure objects such as `AWSCluster` or `VsphereCluster`. See #53777 [https://github.com/rancher/rancher/issues/53777].
- When using native CAPI infrastructure providers with v2prov, Rancher will now generate the bootstrap cloud-config in the Jinja format [https://docs.cloud-init.io/en/latest/explanation/format/jinja.html]. User-data for regular node drivers with v2prov is unaffected. See #53777 [https://github.com/rancher/rancher/issues/53777].

#### Known Issues

- A known issue exists related to CAPI infrastructure providers within Rancher's V2 provisioning framework. Creating or editing v2prov clusters with native infrastructure providers (such as CAPA or CAPV) from the UI can under some failure conditions leave unused, orphaned infrastructure resources (e.g., `AWSMachineTemplate` or `AWSCluster`) in the management cluster which need to be cleaned up manually. See #55752 [https://github.com/rancher/rancher/issues/55752].

### Continuous Delivery (Fleet)

#### Features and Enhancements

- SUSE Continuous Delivery introduces native support for the SUSE Application Collection as a dedicated deployment flow under App Bundles. Users can authenticate using their apps.rancher.io access credentials to browse, configure, and deploy curated application charts directly to downstream clusters, with Fleet automatically handling imagePullSecrets management for target workloads. See #14627 [https://github.com/rancher/dashboard/issues/14627].

#### Behavior Changes

- **Deprecation**: Fleet will be removing support for `GitRepoRestriction` in a future release, making the use of the Policy resource for configuring multi-tenant environments the only option. See #55652 [https://github.com/rancher/rancher/issues/55652].
- **Deprecation**: Fleet support for unauthenticated webhook calls will be removed in a future release, making the setup of a webhook secret [https://documentation.suse.com/cloudnative/continuous-delivery/latest/en/how-tos-for-users/webhook.html#_3_optional_configure_a_webhook_secret] the only option. See #55633 [https://github.com/rancher/rancher/issues/55633].

### Role-Based Access Control (RBAC)

#### Features and Enhancements

- Rancher introduces a new `inheritedNamespacedRules` field for GlobalRoles, allowing administrators to automatically grant users RBAC permissions to specific namespaces across all

downstream clusters (for example, shared monitoring or logging namespaces). Rules defined in `inheritedNamespacedRules` automatically propagate to all downstream clusters managed by Rancher. If the namespace doesn't exist on a cluster no rules are added. However, if the namespace does not exist but gets created, the controller triggers and creates the rules. Please note that these inherited rules apply exclusively to downstream clusters and do not affect the local management cluster.

- Configuring `inheritedNamespacedRules` currently requires editing the GlobalRole YAML directly or via the API, as direct UI inputs for this specific map are not yet available. Example YAML configuration for a GlobalRole with `inheritedNamespacedRules`:

```
apiVersion: management.cattle.io/v3
kind: GlobalRole
metadata:
namespace: default
name: test-gr-1
rules:
- apiGroups:
- catalog.cattle.io
resources:
- apps
verbs:
- create
inheritedNamespacedRules:
shared-namespace:
    - apiGroups:
    - catalog.cattle.io
    resources:
    - apps
    verbs:
    - edit
```

See #50663 [https://github.com/rancher/rancher/issues/50663] for more information.

## Virtualization Management (Harvester)

### Features and Enhancements

- This release adds support for the `harvester-baremetal-container-workload` feature flag, disabled by default, within Rancher's Fleet continuous delivery targeting mechanism. When enabled, the UI stops filtering out Harvester clusters, allowing Fleet bundles and applications to target and deploy container workloads directly to them. After enabling the `harvester-baremetal-container-workload` feature flag, please validate existing Fleet Bundles to ensure they now select, or don't select, the required clusters. When disabled, the UI maintains the default behavior and excludes Harvester clusters from Fleet targets. See #12518 [https://github.com/rancher/dashboard/issues/12518].

## Changes Since v2.15.0

See the full list of changes [https://github.com/rancher/rancher/compare/v2.15.0%E2%80%A6v2.15.1].

## Install/Upgrade Notes

If you're installing Rancher for the first time, your environment must fulfill the installation requirements.

- Chart name change for Rancher Prime.
  - The chart name change introduced in Rancher Prime v2.13.1 has been reverted. The chart name `rancher` should be used for all installations and upgrades. As an example, the installation command is now `helm install rancher rancher-prime/rancher`.
- Rancher now requires the cluster it runs on to have the Kubernetes API Aggregation Layer [https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/] enabled. This is because Rancher extends Kubernetes with additional APIs by registering its own extension API server. Please note that all versions of Kubernetes supported in this Rancher versions K8s distributions (RKE2/K3s) will have the aggregation layer configured and enabled by default.
  - Refer to the Extension API Server documentation and #50400 [https://github.com/rancher/rancher/issues/50400] for more information.
- Rancher Kubernetes Engine (RKE/RKE1) has reached end of life as of **July 31, 2025**. Rancher versions 2.12.0 and later no longer support provisioning or managing downstream RKE1 clusters. We recommend replatforming RKE1 clusters to RKE2 to ensure continued support and security updates. Learn more about the transition here [https://www.suse.com/support/kb/doc/?id=000021518].



  - Rancher now has a pre-upgrade validation check for RKE1 resources which fails and lists the RKE1 resources if present. Refer to #50286 [https://github.com/rancher/rancher/issues/50286] for more information.
- It is crucial that you review the available disk space on your nodes and plan accordingly before upgrading to Rancher v2.12.0 and later to avoid potential disk pressure and pod eviction issues.
  - For additional information, refer to the UI Server Side Pagination - Disk Space documentation.
- Rancher now has an enablement option called `AUDIT_LOG_ENABLED` for API Audit Logs for a Rancher installation.
  - In Rancher versions 2.11.x and earlier, only the `AUDIT_LEVEL` could be set and the default log level (`0`) would disable the audit log. In Rancher versions 2.12.x and later, the default log level (`0`) now only contains the log request and response metadata, and can be set when configuring `AUDIT_LOG_ENABLED`. If installing or upgrading via Helm you can enable the API Audit Logs and specify the log level by applying the following setting to your Helm command: `--set auditLog.enabled=true --set auditLog.level=0`. See the Enabling the API Audit Log to Record System Events documentation and #48941 [https://github.com/rancher/rancher/issues/48941].

## Changes in Image Artifacts

Image artifact digests are renamed in Rancher v2.12.0, v2.11.4 and v2.10.8. Up until this change, separate image digests files for each operating system and architecture have been maintained for compatibility reasons. With this change, only one file for each operating system is to be provided:

- The `rancher-images-digests-linux-amd64.txt` and `rancher-images-digests-linux-arm64.txt` files are to be renamed to `rancher-images-digests-linux.txt`.
- The `rancher-images-digests-windows-ltsc2019.txt` and `rancher-images-digests-windows-ltsc2022.txt` files are to be renamed to `rancher-images-digests-windows.txt`.

## Upgrade Requirements

- **Creating backups:** Create a backup before you upgrade Rancher. To roll back Rancher after an upgrade, you must first back up and restore Rancher to the previous Rancher version. Because Rancher will be restored to the same state as when the backup was created, any changes post-upgrade will not be included after the restore.
- **Helm version requirements:**
  - To manage Rancher 2.12.x and later, you must upgrade your Helm client to version 3.18 or newer.
  - This change is required to reflect the addition of Kubernetes 1.33 support with this release.
  - Currently, the official Helm Version Support Policy [https://helm.sh/docs/topics/version_skew/] dictates that only Helm 3.18 supports the proper Kubernetes version range for Rancher 2.12.
- **CNI requirements:**
  - For Kubernetes v1.19 and later, disable firewalld as it's incompatible with various CNI plugins. See #28840 [https://github.com/rancher/rancher/issues/28840].
  - When upgrading or installing a Linux distribution that uses nf_tables as the backend packet filter, such as SLES 15, RHEL 8, Ubuntu 20.10, Debian 10, or later, upgrade to RKE v1.19.2 or later to get Flannel v0.13.0. Flannel v0.13.0 supports nf_tables. See Flannel #1317 [https://github.com/flannel-io/flannel/issues/1317].
- **Requirements for air-gapped environments:**
  - When using a proxy in front of an air-gapped Rancher instance, you must pass additional parameters to `NO_PROXY`. See the documentation and issue #2725 [https://github.com/rancher/docs/issues/2725#issuecomment-702454584].
  - When installing Rancher with Docker in an air-gapped environment, you must supply a custom `registries.yaml` file to the `docker run` command, as shown in the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/private-registry.html]. If the registry has certificates, then you'll also need to supply those. See #28969 [https://github.com/rancher/rancher/issues/28969#issuecomment-694474229].

## Versions

### Images

- rancher/rancher:v2.15.1

### Tools

- CLI - v2.15.1 [https://github.com/rancher/cli/releases/tag/v2.15.1]

### Kubernetes Versions for RKE2/K3s

- v1.36.3 (Default)
- v1.35.7

- v1.34.10

#### Rancher Helm Chart Versions

In Rancher v2.6.0 and later, in the **Apps & Marketplace** UI, many Rancher Helm charts are named with a major version that starts with *100*. This avoids simultaneous upstream changes and Rancher changes from causing conflicting version increments. This also complies with semantic versioning (SemVer), which is a requirement for Helm. You can see the upstream version number of a chart in the build metadata, for example: `100.0.0+up2.1.0`. See #32294 [https://github.com/rancher/rancher/issues/32294].

## Long-standing Known Issues

#### Long-standing Known Issues - Cluster Provisioning

- Not all cluster tools can be installed on a hardened cluster.
- **Rancher v2.8.1:**
  - When you attempt to register a new etcd/controlplane node in a CAPR-managed cluster after a failed etcd snapshot restoration, the node can become stuck in a perpetual paused state, displaying the error message `[ERROR] 000 received while downloading Rancher connection information. Sleeping for 5 seconds and trying again`. As a workaround, you can unpause the cluster by running `kubectl edit clusters.cluster clustername -n fleet-default` and set `spec.unpaused` to `false`. See #43735 [https://github.com/rancher/rancher/issues/43735].

#### Long-standing Known Issues - RKE2 Provisioning

- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #34518 [https://github.com/rancher/rancher/issues/34518] and #42834 [https://github.com/rancher/rancher/issues/42834].
- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].

#### Long-standing Known Issues - K3s Provisioning

- **Rancher v2.7.6:**
  - Provisioning RKE2/K3s clusters with added (not built-in) custom node drivers causes provisioning to fail. As a workaround, fix [https://github.com/rancher/rancher/issues/37074#issuecomment-1664722305] the added node drivers after activating. See #37074 [https://github.com/rancher/rancher/issues/37074].
- **Rancher v2.7.2:**
  - Clusters remain in an `Updating` state even when they contain nodes in an `Error` state. See #39164 [https://github.com/rancher/rancher/issues/39164].

#### Long-standing Known Issues - Rancher App (Global UI)

- **Rancher v2.10.0:**
  - After deleting a Namespace or Project in the Rancher UI, the Namespace or Project remains visible. As a workaround, refresh the page. See #12220 [https://github.com/rancher/dashboard/issues/12220].
- **Rancher v2.9.2:**
  - Although system mode node pools must have at least one node, the Rancher UI allows a minimum node count of zero. Inputting a zero minimum node count through the UI can cause cluster creation to fail due to an invalid parameter error. To prevent this error from occurring, enter a minimum node count at least equal to the node count. See #11922 [https://github.com/rancher/dashboard/issues/11922].
- **Rancher v2.7.7:**
  - When creating a cluster in the Rancher UI it does not allow the use of an underscore `_` in the `Cluster Name` field. See #9416 [https://github.com/rancher/dashboard/issues/9416].
- **Rancher v2.6.4:**
  - Deployment securityContext section is missing when a new workload is created. This prevents pods from starting when Pod Security Policy Support is enabled. See #4815 [https://github.com/rancher/dashboard/issues/4815].

#### Long-standing Known Issues - EKS

- **Rancher v2.7.0:**
  - EKS clusters on Kubernetes v1.21 or below on Rancher v2.7 cannot be upgraded. See #39392 [https://github.com/rancher/rancher/issues/39392].

#### Long-standing Known Issues - Authentication

- **Rancher v2.9.0:**
  - There are some known issues with the OpenID Connect provider support:
    - When the generic OIDC auth provider is enabled, and you attempt to add auth provider users to a cluster or project, users are not populated in the dropdown search bar. This is expected behavior as the OIDC auth provider alone is not searchable. See #46104 [https://github.com/rancher/rancher/issues/46104].
    - When the generic OIDC auth provider is enabled, auth provider users that are added to a cluster/project by their username are not able to access resources upon logging in. A user will only have access to resources upon login if the user is added by their userID. See #46105 [https://github.com/rancher/rancher/issues/46105].
    - When the generic OIDC auth provider is enabled and an auth provider user in a nested group is logged into Rancher, the user will see the following error when they attempt to create a Project: `projectroletemplatebindings.management.cattle.io is forbidden: User "u-gcxatwsnku" cannot create resource "projectroletemplatebindings" in API group "management.cattle.io" in the namespace "p-9t5pg"`. However, the project is still created. See #46106 [https://github.com/rancher/rancher/issues/46106].

#### Long-standing Known Issues - Rancher Webhook

- **Rancher v2.7.2:**
  - A webhook is installed in all downstream clusters. There are several issues that users may encounter

with this functionality:

- If you rollback from a version of Rancher v2.7.2 or later, to a Rancher version earlier than v2.7.2, the webhooks will remain in downstream clusters. Since the webhook is designed to be `1:1` compatible with specific versions of Rancher, this can cause unexpected behaviors to occur downstream. The Rancher team has developed a script [https://github.com/rancher/webhook/wiki/Remove-Webhook-from-downstream-clusters] which should be used after rollback is complete (meaning after a Rancher version earlier than v2.7.2 is running). This removes the webhook from affected downstream clusters. See #40816 [https://github.com/rancher/rancher/issues/40816].

**Long-standing Known Issues - Virtualization Management (Harvester)**

- **Rancher v2.13.1:**
  - When upgrading to Rancher v2.13.1 or higher while using Harvester v1.6.1, users may encounter an issue with their Load Balancers in downstream clusters using the Harvester Cloud Provider, and must perform the following workaround [https://documentation.suse.com/cloudnative/virtualization/latest/en/troubleshooting/rancher.html#_guest_cluster_load_balancer_ip_is_not_reachable] to instruct Calico to not use any of the IP/interface managed by `kube-vip`. See #9767 [https://github.com/harvester/harvester/issues/9767].
- **Rancher v2.7.2:**
  - If you're using Rancher v2.7.2 with Harvester v1.1.1 clusters, you won't be able to select the Harvester cloud provider when deploying or updating guest clusters. The Harvester release notes [https://github.com/harvester/release-notes/blob/main/v1.1.2.md#important-information-about-rancher-support] contain instructions on how to resolve this. See #3750 [https://github.com/harvester/harvester/issues/3750].

**Long-standing Known Issues - Backup/Restore**

- When migrating to a cluster with the Rancher Backup feature, the server-url cannot be changed to a different location. It must continue to use the same URL.
- **Rancher v2.13.0:**
  - When performing a rollback from Rancher v2.13.0 to v2.12.3 using the backup and restore operator (BRO), the restore does not complete successfully. See #844 [https://github.com/rancher/backup-restore-operator/issues/844]. To work around this issue, you must scale down your Rancher deployment and uninstall the Webhook chart before performing the restore. For details, refer to this Knowledge Base article [https://support.scc.suse.com/s/kb/Rolling-back-from-Rancher-v2-13-0-to-v2-12-3?language=en_US].
- **Rancher v2.7.7:**
  - Due to the backoff logic in various components, downstream provisioned K3s and RKE2 clusters may take longer to re-achieve **Active** status after a migration. If you see that a downstream cluster is still updating or in an error state immediately after a migration, please let it attempt to resolve itself. This might take up to an hour to complete. See #34518 [https://github.com/rancher/rancher/issues/34518] and #42834 [https://github.com/rancher/rancher/issues/42834].

# About Rancher

## What is SUSE Rancher Prime?

Rancher is a Kubernetes management tool to deploy and run clusters anywhere and on any provider.

Rancher can provision Kubernetes from a hosted provider, provision compute nodes and then install Kubernetes onto them, or import existing Kubernetes clusters running anywhere.

Rancher adds significant value on top of Kubernetes, first by centralizing authentication and role-based access control (RBAC) for all of the clusters, giving global admins the ability to control cluster access from one location.

It then enables detailed monitoring and alerting for clusters and their resources, ships logs to external providers, and integrates directly with Helm via the Application Catalog. If you have an external CI/CD system, you can plug it into Rancher, but if you don't, Rancher even includes Fleet to help you automatically deploy and upgrade workloads.

Rancher is a *complete* container management platform for Kubernetes, giving you the tools to successfully run Kubernetes anywhere.

## SUSE Rancher Prime: The Open Infrastructure Platform

SUSE Rancher Prime is the enterprise edition of Rancher. It gives platform teams one place to manage Kubernetes clusters, containers and virtual machines, whether they run in the cloud, in your data center or at the edge.

Rancher Prime uses the same open source code as the Rancher project which remains fully open source. You can adopt Prime without being locked in to SUSE or any other vendor. What Prime adds is what you need to run Rancher in production, including a trusted software supply chain, longer support lifecycles and enterprise support from SUSE.

### The SUSE Rancher Prime Difference

#### One control plane for any Kubernetes

Rancher Manager lets you provision, govern and audit every cluster from a single console. It manages SUSE's RKE2 and K3s distributions and Virtual Clusters. It also manages any CNCF-certified distribution, including Amazon EKS, Azure AKS and Google GKE. Access control, security policies and audit logs work the same way across all of your environments, air-gapped sites included. Rancher Manager connects to the identity providers you already use, such as Active Directory, OpenLDAP and Okta.

#### Built for production

Prime installation assets and container images come from a trusted registry that SUSE manages. Images are signed and include SBOMs and provenance data, so your security team can confirm exactly what is running.

RKE2 and K3s receive up to five years of long-term support. This gives you time to plan upgrades on your own schedule.

SUSE Kubernetes experts are available at any moment under defined SLAs. Prime customers also get

Kubernetes advisories and reference architectures.

#### Add what you need

The platform grows with your requirements, and everything you add is managed from Rancher Manager. SUSE Virtualization runs virtual machines alongside containers. SUSE Storage provides distributed block storage for your clusters. SUSE Security protects containers from build through runtime. SUSE Observability gives you full-stack visibility and is built on OpenTelemetry. SUSE Application Collection and Developer Access give developers hardened application images and a self-service option to get what they need.

#### Open by design

Rancher Prime doesn't require a specific operating system, storage product or proprietary tooling. You decide which components to use and where they run.

### Get started

To learn more, contact SUSE. For technical details, see What is Rancher?

# Overview

Rancher is a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere, meet IT requirements, and empower DevOps teams.

### Run Kubernetes Everywhere

Kubernetes has become the container orchestration standard. Most cloud and virtualization vendors now offer it as standard infrastructure. Rancher users have the choice of creating Kubernetes clusters with Rancher Kubernetes Engine (RKE) or cloud Kubernetes services, such as GKE, AKS, and EKS. Rancher users can also import and manage their existing Kubernetes clusters created using any Kubernetes distribution or installer.

### Meet IT Requirements

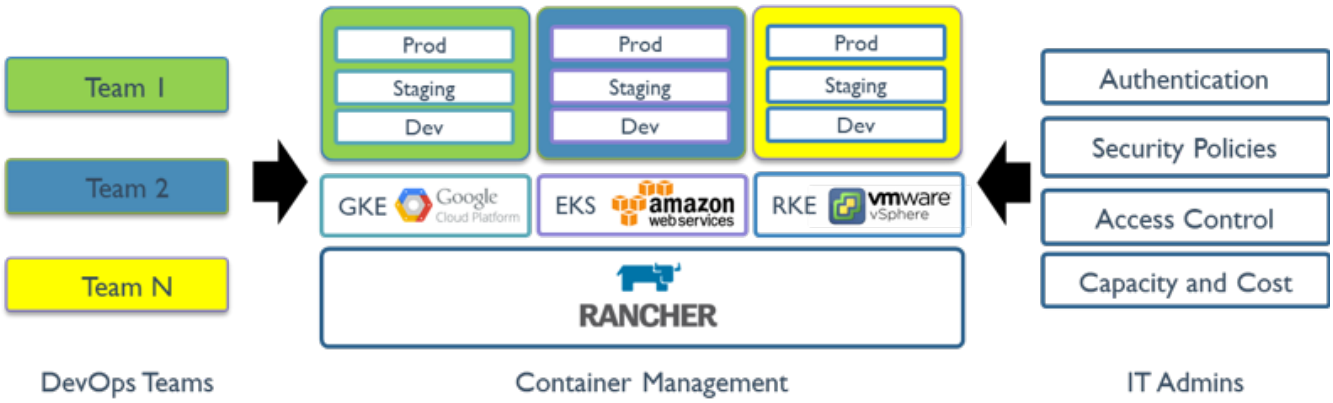
Rancher supports centralized authentication, access control, and monitoring for all Kubernetes clusters under its control. For example, you can:

- Use your Active Directory credentials to access Kubernetes clusters hosted by cloud vendors, such as GKE.
- Setup and enforce access control and security policies across all users, groups, projects, clusters, and clouds.
- View the health and capacity of your Kubernetes clusters from a single-pane-of-glass.

### Empower DevOps Teams

Rancher provides an intuitive user interface for DevOps engineers to manage their application workload. The user does not need to have in-depth knowledge of Kubernetes concepts to start using Rancher. Rancher catalog contains a set of useful DevOps tools. Rancher is certified with a wide selection of cloud native ecosystem products, including, for example, security tools, monitoring systems, container registries, and storage and networking drivers.

The following figure illustrates the role Rancher plays in IT and DevOps organizations. Each team deploys their applications on the public or private clouds they choose. IT administrators gain visibility and enforce policies across all users, clusters, and clouds.




## Features of the Rancher API Server

The Rancher API server is built on top of an embedded Kubernetes API server and an etcd database. It implements the following functionalities:

### Authorization and Role-Based Access Control

- **User management:** The Rancher API server manages user identities that correspond to external authentication providers like Active Directory or GitHub, in addition to local users.
- **Authorization:** The Rancher API server manages access control and security standards.

### Working with Kubernetes

- **Provisioning Kubernetes clusters:** The Rancher API server can provision Kubernetes on existing nodes, or perform Kubernetes upgrades.
- **Catalog management:** Rancher provides the ability to use a catalog of Helm charts that make it easy to repeatedly deploy applications.
- **Managing projects:** A project is a group of multiple namespaces and access control policies within a cluster. A project is a Rancher concept, not a Kubernetes concept, which allows you to manage multiple namespaces as a group and perform Kubernetes operations in them. The Rancher UI provides features for project administration and for managing applications within projects.
- **Fleet Continuous Delivery:** Within Rancher, you can leverage SUSE® Rancher Prime: Continuous Delivery to deploy applications from git repositories, without any manual operation, to targeted downstream Kubernetes clusters.
- **Istio:** Our integration with Istio is designed so that a Rancher operator, such as an administrator or cluster owner, can deliver Istio to developers. Then developers can use Istio to enforce security policies, troubleshoot problems, or manage traffic for green/blue deployments, canary deployments, or A/B testing.

### Working with Cloud Infrastructure

- **Tracking nodes:** The Rancher API server tracks identities of all the nodes in all clusters.
- **Setting up infrastructure:** When configured to use a cloud provider, Rancher can dynamically provision new nodes and persistent storage in the cloud.

**Cluster Visibility**

- **Logging:** Rancher can integrate with a variety of popular logging services and tools that exist outside of your Kubernetes clusters.
- **Monitoring:** Using Rancher, you can monitor the state and processes of your cluster nodes, Kubernetes components, and software deployments through integration with Prometheus, a leading open-source monitoring solution.
- **Alerting:** To keep your clusters and applications healthy and driving your organizational productivity forward, you need to stay informed of events occurring in your clusters and projects, both planned and unplanned.

## Editing Downstream Clusters with Rancher

The options and settings available for an existing cluster change based on the method that you used to provision it. For example, only clusters provisioned by RKE have **Cluster Options** available for editing.

After a cluster is created with Rancher, a cluster administrator can manage cluster membership or manage node pools, among xref:[other options.]

The following table summarizes the options and settings available for each cluster type:

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Using kubectl and a kubeconfig file to Access a Cluster | ⬕ | ⬕ | ⬕ | ⬕ |
| Managing Cluster Members | ⬕ | ⬕ | ⬕ | ⬕ |
| Editing and Upgrading Clusters | ⬕ | ⬕ | ⬕ | ⬕[2] |
| Managing Nodes | ⬕ | ⬕ | ⬕ | ⬕[3] |
| Managing Persistent Volumes and Storage Classes | ⬕ | ⬕ | ⬕ | ⬕ |
| Managing Projects, Namespaces and Workloads | ⬕ | ⬕ | ⬕ | ⬕ |
| Using App Catalogs | ⬕ | ⬕ | ⬕ | ⬕ |
| Configuring Tools (Alerts, Notifiers, Monitoring, Logging, Istio) | ⬕ | ⬕ | ⬕ | ⬕ |
| Running Security Scans | ⬕ | ⬕ | ⬕ | ⬕ |
| Ability to rotate certificates | ⬕ | ⬕ | | |
| Ability to backup and restore Rancher-launched clusters | ⬕ | ⬕ | | ⬕[4] |

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Cleaning Kubernetes components when clusters are no longer reachable from Rancher | ◪ | | | |

1. Registered EKS, GKE and AKS clusters have the same options available as EKS, GKE and AKS clusters created from the Rancher UI. The difference is that when a registered cluster is deleted from the Rancher UI, it is not destroyed.
2. Cluster configuration options can't be edited for registered clusters, except for K3s and RKE2 clusters.
3. For registered cluster nodes, the Rancher UI exposes the ability to cordon, drain, and edit the node.
4. For registered clusters using etcd as a control plane, snapshots must be taken manually outside of the Rancher UI to use for backup and recovery.

# SUSE Rancher Prime Architecture

This section focuses on the Rancher server and its components and how Rancher communicates with downstream Kubernetes clusters.

For information on the different ways that Rancher can be installed, refer to the overview of installation options.

For a list of main features of the Rancher API server, refer to the overview section.

For guidance about setting up the underlying infrastructure for the Rancher server, refer to the architecture recommendations.



This section assumes a basic familiarity with Docker and Kubernetes. For a brief explanation of how Kubernetes components work together, refer to the concepts page.

## Architecture Recommendations

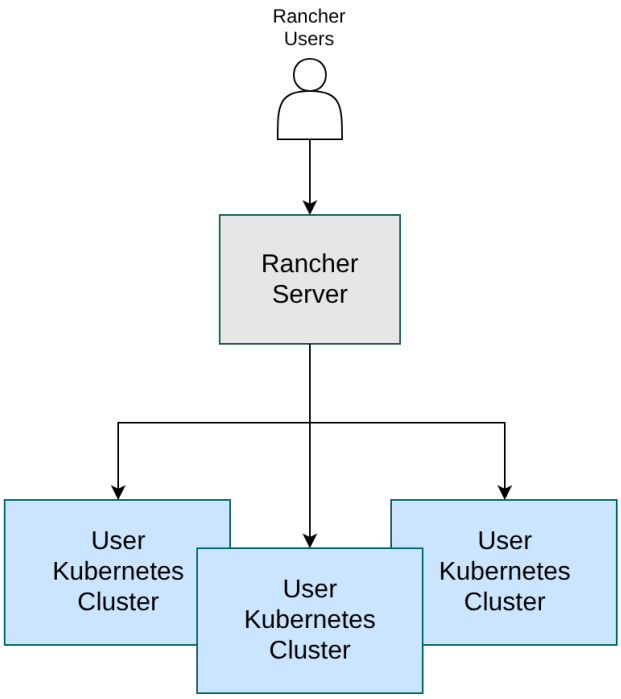
If you are installing Rancher on a single node, the main architecture recommendation that applies to your installation is that the node running Rancher should be separate from downstream clusters.

#### Separation of Rancher and User Clusters

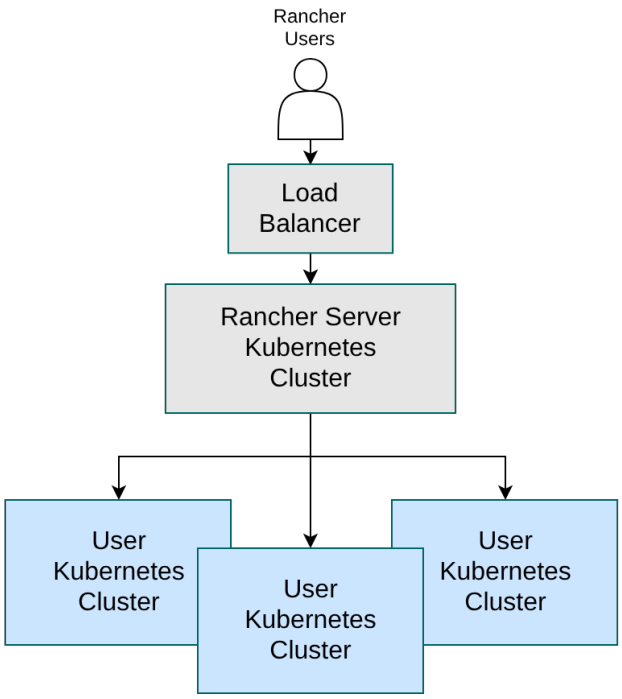
A user cluster is a downstream Kubernetes cluster that runs your apps and services.

If Rancher is intended to manage downstream Kubernetes clusters, the Kubernetes cluster that the Rancher server runs on should also be separate from the downstream user clusters.

Separation of Single Node Rancher Server and User Clusters




Separation of High-availability Rancher Server and User Clusters




**Why HA is Better for Rancher in Production**

We recommend installing the Rancher server on a high-availability Kubernetes cluster, primarily because it protects the Rancher server data. In a high-availability installation, a load balancer serves as the single point of contact for clients, distributing network traffic across multiple servers in the cluster and helping to prevent any one server from becoming a point of failure.

**SUSE® Rancher Prime: K3s Kubernetes Cluster Installations**

One option for the underlying Kubernetes cluster is to use K3s Kubernetes. K3s is Rancher's CNCF certified Kubernetes distribution. It is easy to install and uses half the memory of Kubernetes, all in a binary of less than 100 MB. Another advantage of K3s is that it allows an external datastore to hold the cluster data, allowing the K3s server nodes to be treated as ephemeral.

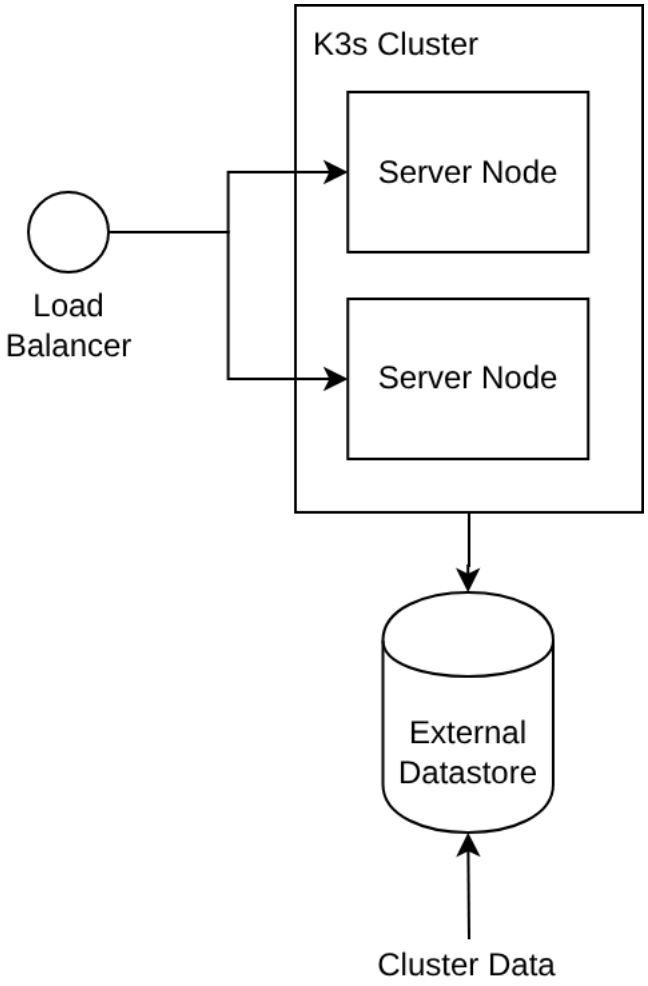



*Figure 1. Architecture of a K3s Kubernetes Cluster Running the Rancher Management Server*

**Recommended Load Balancer Configuration for Kubernetes Installations**

We recommend the following configurations for the load balancer and Ingress controllers:

- The DNS for Rancher should resolve to a Layer 4 load balancer (TCP).
- The Load Balancer should forward port TCP/80 and TCP/443 to all 3 nodes in the Kubernetes cluster.
- The Ingress controller will redirect HTTP to HTTPS and terminate SSL/TLS on port TCP/443.
- The Ingress controller will forward traffic to port TCP/80 on the pod in the Rancher deployment.

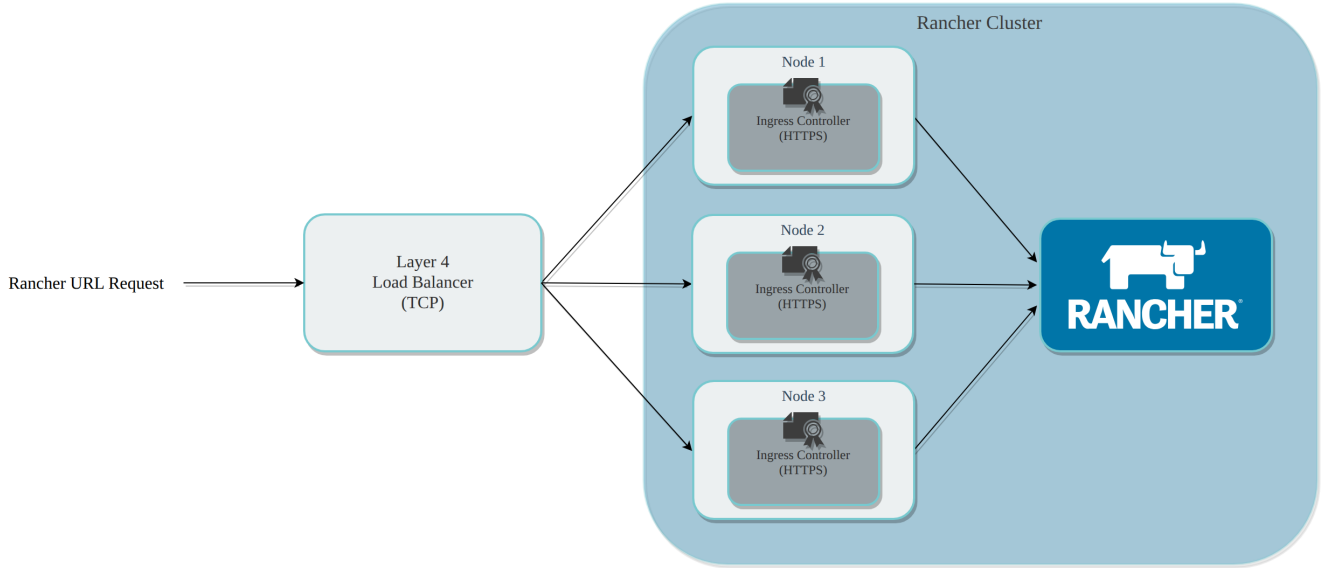



*Figure 2. Rancher installed on a Kubernetes cluster with layer 4 load balancer, depicting SSL termination at Ingress controllers*

### Environment for Kubernetes Installations

It is strongly recommended to install Rancher on a Kubernetes cluster on hosted infrastructure such as Amazon's EC2 or Google Compute Engine.

For the best performance and greater security, we recommend a dedicated Kubernetes cluster for the Rancher management server. Running user workloads on this cluster is not advised. After deploying Rancher, you can create or import clusters for running your workloads.

### Recommended Node Roles for Kubernetes Installations

The below recommendations apply when Rancher is installed on a K3s Kubernetes cluster.

#### SUSE® Rancher Prime: K3s Cluster Roles

In K3s clusters, there are two types of nodes: server nodes and agent nodes. Both servers and agents can have workloads scheduled on them. Server nodes run the Kubernetes master.

For the cluster running the Rancher management server, we recommend using two server nodes. Agent nodes are not required.

### Architecture for an Authorized Cluster Endpoint (ACE)

If you are using an authorized cluster endpoint (ACE), we recommend creating an FQDN pointing to a load balancer which balances traffic across our nodes with the `controlplane` role.

If you are using private CA signed certificates on the load balancer, you have to supply the CA certificate, which will be included in the generated kubeconfig file to validate the certificate chain. See the documentation on kubeconfig files and API keys for more information.

ACE support is available for registered RKE2 and K3s clusters. To view the manual steps to perform on the downstream cluster to enable the ACE, click here.

## Communicating with Downstream User Clusters

This section describes how Rancher provisions and manages the downstream user clusters that run your apps and services.

The below diagram shows how the cluster controllers, cluster agents, and Rancher system agent allow Rancher to control downstream clusters.

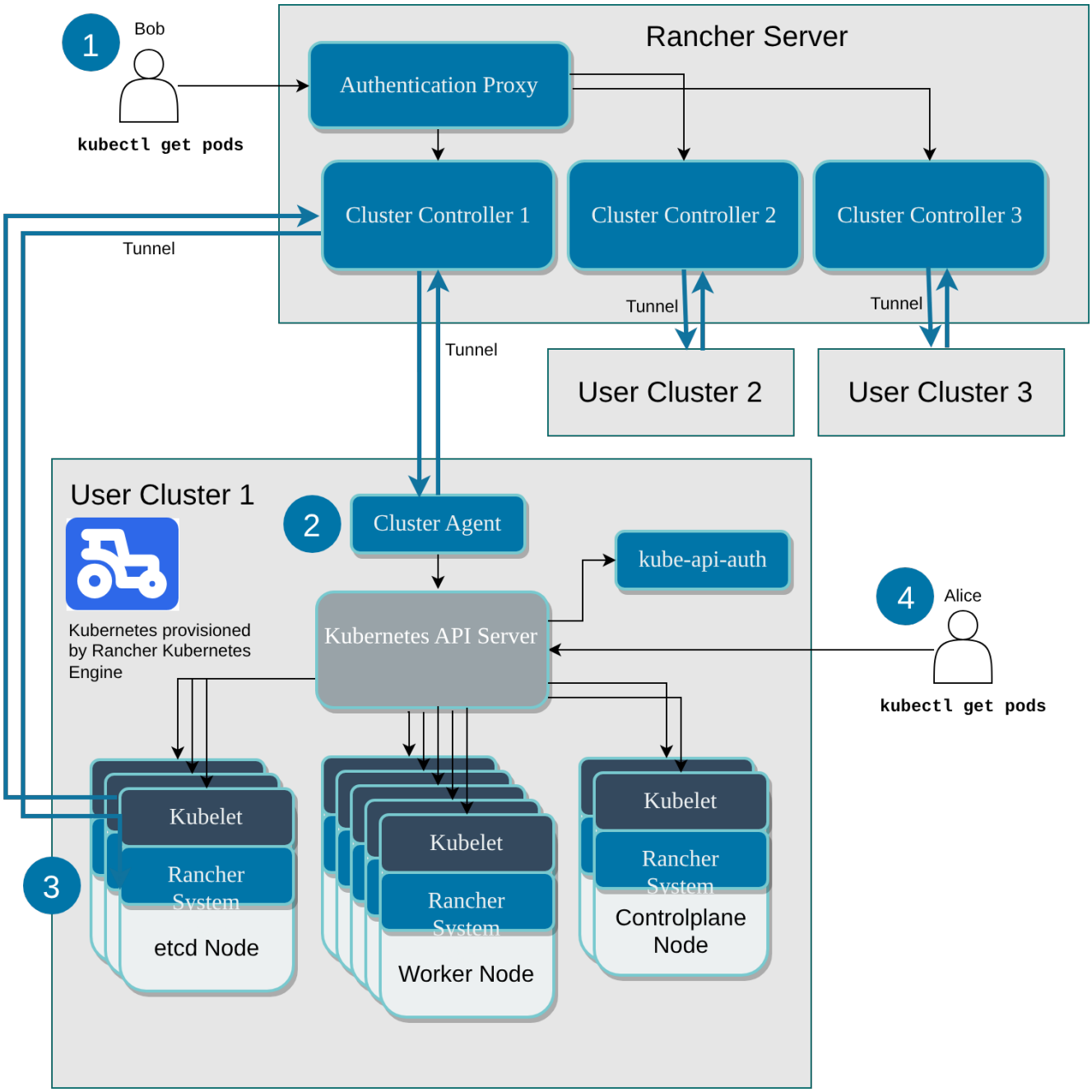



*Figure 3. Communicating with Downstream Clusters*

The following descriptions correspond to the numbers in the diagram above:

1. The Authentication Proxy
2. Cluster Controllers and Cluster Agents
3. Rancher System Agent
4. Authorized Cluster Endpoint

### 1. The Authentication Proxy

In this diagram, a user named Bob wants to see all pods running on a downstream user cluster called User Cluster 1. From within Rancher, he can run a `kubectl` command to see the pods. Bob is authenticated through Rancher's authentication proxy.

The authentication proxy forwards all Kubernetes API calls to downstream clusters. It integrates with authentication services like local authentication, Active Directory, and GitHub. On every Kubernetes API call, the authentication proxy authenticates the caller and sets the proper Kubernetes impersonation headers before forwarding the call to Kubernetes masters.

Rancher communicates with Kubernetes clusters using a service account [https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/]. Each user account in Rancher correlates with an equivalent service account in the downstream cluster. Rancher uses the service account to impersonate [https://kubernetes.io/docs/reference/access-authn-authz/authentication/#user-impersonation] the user, which provides all the permissions the user is intended to have.

By default, Rancher generates a kubeconfig file that contains credentials for proxying through the Rancher server to connect to the Kubernetes API server on a downstream user cluster. The kubeconfig file (`kube_config_rancher-cluster.yml`) contains full access to the cluster.

### 2. Cluster Controllers and Cluster Agents

Each downstream user cluster has a cluster agent, which opens a tunnel to the corresponding cluster controller within the Rancher server.

There is one cluster controller and one cluster agent for each downstream cluster. Each cluster controller:

- Watches for resource changes in the downstream cluster
- Brings the current state of the downstream cluster to the desired state
- Configures access control policies to clusters and projects
- Provisions clusters by calling the required Docker machine drivers and Kubernetes engines, such as GKE

By default, to enable Rancher to communicate with a downstream cluster, the cluster controller connects to the cluster agent. If the cluster agent is not available, the cluster controller can connect to a Rancher system agent instead.

The cluster agent, also called `cattle-cluster-agent`, is a component that runs in a downstream user cluster. It performs the following tasks:

- Connects to the Kubernetes API of Rancher-launched Kubernetes clusters
- Manages workloads, pod creation and deployment within each cluster
- Applies the roles and bindings defined in each cluster's global policies
- Communicates between the cluster and Rancher server (through a tunnel to the cluster controller) about events, stats, node info, and health

### 3. Rancher System Agent

If the cluster agent (also called `cattle-cluster-agent`) is not available, the Rancher system agent creates a tunnel to the cluster controller to communicate with Rancher.

The `rancher-system-agent` runs on every node in RKE2 and K3s Kubernetes clusters. It is used to interact with the nodes when performing cluster operations. Examples of cluster operations include upgrading the Kubernetes version and creating or restoring etcd snapshots.

#### 4. Authorized Cluster Endpoint

An authorized cluster endpoint (ACE) allows users to connect to the Kubernetes API server of a downstream cluster without having to route their requests through the Rancher authentication proxy.

> ACE is available on RKE2 and K3s clusters that are provisioned or registered with Rancher. It's not available on clusters in a hosted Kubernetes provider, such as Amazon's EKS.

There are two main reasons why a user might need the authorized cluster endpoint:

- To access a downstream user cluster while Rancher is down
- To reduce latency in situations where the Rancher server and downstream cluster are separated by a long distance

The `kube-api-auth` microservice is deployed to provide the user authentication functionality for the authorized cluster endpoint. When you access the user cluster using `kubectl`, the cluster's Kubernetes API server authenticates you by using the `kube-api-auth` service as a webhook.

Like the authorized cluster endpoint, the `kube-api-auth` authentication service is also only available for Rancher-launched Kubernetes clusters.

> **Example scenario:** Let's say that the Rancher server is located in the United States, and User Cluster `1` is located in Australia. A user, Alice, also lives in Australia. Alice can manipulate resources in User Cluster `1` by using the Rancher UI, but her requests will have to be sent from Australia to the Rancher server in the United States, then be proxied back to Australia, where the downstream user cluster is. The geographical distance may cause significant latency, which Alice can reduce by using the authorized cluster endpoint.

With this endpoint enabled for the downstream cluster, Rancher generates an extra Kubernetes context in the kubeconfig file in order to connect directly to the cluster. This file has the credentials for `kubectl` and `helm`.



To use the ACE context in your kubeconfig, run `kubectl use-context <ace context name>` after enabling it.

For more information, refer to the section on accessing your cluster with kubectl and the kubeconfig file.

We recommend exporting the kubeconfig file so that if Rancher goes down, you can still use the credentials in the file to access your cluster.

#### Impersonation

Users technically exist only on the upstream cluster. Rancher creates RoleBindings and ClusterRoleBindings

[https://kubernetes.io/docs/reference/access-authn-authz/rbac/#rolebinding-and-clusterrolebinding] that refer to Rancher users, even though there is no actual User resource [https://kubernetes.io/docs/reference/access-authn-authz/authentication/#users-in-kubernetes] on the downstream cluster.

When users interact with a downstream cluster through the authentication proxy, there needs to be some entity downstream to serve as the actor for those requests. Rancher creates service accounts to be that entity. Each service account is only granted one permission, which is to **impersonate** the user they belong to. If there was only one service account that could impersonate any user, then it would be possible for a malicious user to corrupt that account and escalate their privileges by impersonating another user. This issue was the basis for a CVE [https://github.com/rancher/rancher/security/advisories/GHSA-pvxj-25m6-7vqr].

#### Impersonation Troubleshooting

On the downstream cluster, five resources handle impersonation:

- namespace: `cattle-impersonation-system`
- service account: `cattle-impersonation-system/cattle-impersonation-<user ID>`
- account token secret: `cattle-impersonation-system/cattle-impersonation-<user ID>-token-<hash>`
- cluster role: `cattle-impersonation-<user ID>`
- cluster role binding: `cattle-impersonation-<user ID>`

In this example of a typical impersonation cluster role, the system is configured to use `github` as the auth provider:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 creationTimestamp: "2021-10-06T18:20:13Z"
 labels:
   authz.cluster.cattle.io/impersonator: "true"
   cattle.io/creator: norman
 name: cattle-impersonation-user-abcde
 resourceVersion: "3528"
 uid: a7478731-72a0-4343-b09f-c3bf12552d77
rules:
# allowed to impersonate user user-abcde
- apiGroups:
  - ""
  resourceNames:
  - user-abcde
  resources:
  - users
  verbs:
  - impersonate
# allowed to impersonate listed groups
- apiGroups:
```

```
  - ""
  resourceNames:
  - github_team://123 # group from GitHub auth provider
  - system:authenticated # automatic group from Kubernetes
  - system:cattle:authenticated # automatic group from Rancher
  resources:
  - groups
  verbs:
  - impersonate
# allowed to impersonate principal ID github_user://098
- apiGroups:
  - authentication.k8s.io
  resourceNames:
  - github_user://098 # principal ID from GitHub auth provider
  resources:
  - userextras/principalid
  verbs:
  - impersonate
# allowed to impersonate username example
- apiGroups:
  - authentication.k8s.io
  resourceNames:
  - example # username from GitHub auth provider
  resources:
  - userextras/username
  verbs:
  - impersonate
```

When you troubleshoot impersonation issues, check whether these resources exist for the user, and whether the rules in the cluster role look similar to the above. For example:

```
kubectl --namespace cattle-impersonation-system get serviceaccount
cattle-impersonation-<user ID>
kubectl --namespace cattle-impersonation-system get secret cattle-
impersonation-<user ID>-token-<hash>
kubectl get clusterrole cattle-impersonation-<user ID> --output yaml
kubectl get clusterrolebinding cattle-impersonation-<user ID>
```

If you see an error related to "impersonation" in the UI, pay close attention to the *end* of the error message, which should indicate the real reason that the request failed.

#### Important Files

The files mentioned below are needed to maintain, troubleshoot and upgrade your cluster:

- `config.yaml`: The RKE2 and K3s cluster configuration file.
- `rke2.yaml` or `k3s.yaml`: The Kubeconfig file for your RKE2 or K3s cluster. This file contains credentials for full access to the cluster. You can use this file to authenticate with a Rancher-launched Kubernetes cluster if Rancher goes down.

For more information on connecting to a cluster without the Rancher authentication proxy and other configuration options, refer to the kubeconfig file documentation.

### Tools for Provisioning Kubernetes Clusters

The tools that Rancher uses to provision downstream user clusters depends on the type of cluster that is being provisioned.

#### Rancher Launched Kubernetes for Nodes Hosted in an Infrastructure Provider

Rancher can dynamically provision nodes in a provider such as Amazon EC2, DigitalOcean, Azure, or vSphere, then install Kubernetes on them.

Rancher provisions this type of cluster using docker-machine. [https://github.com/rancher/machine]

#### Rancher Launched Kubernetes for Custom Nodes

When setting up this type of cluster, Rancher installs Kubernetes on existing nodes, which creates a custom cluster.

Rancher provisions this type of cluster using RKE2 [https://github.com/rancher/rke2] or K3s [https://github.com/rancher/k3s].

#### Hosted Kubernetes Providers

When setting up this type of cluster, Kubernetes is installed by providers such as Google Kubernetes Engine, Amazon Elastic Container Service for Kubernetes, or Azure Kubernetes Service.

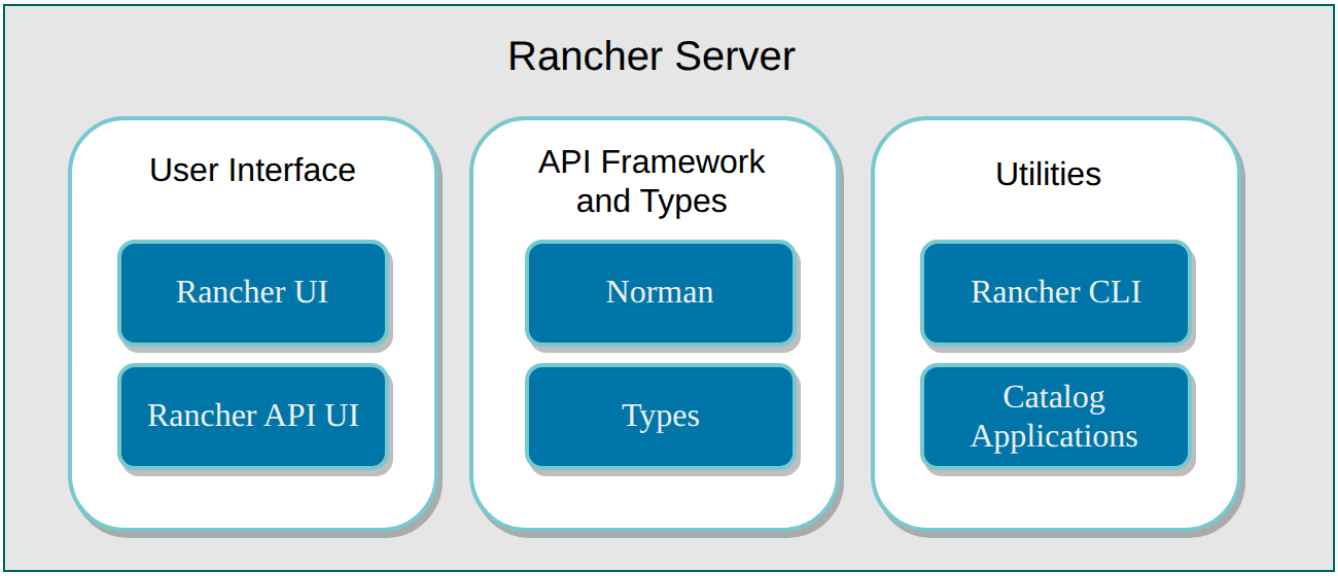
Rancher provisions this type of cluster using kontainer-engine. [https://github.com/rancher/kontainer-engine]

#### Imported Kubernetes Clusters

In this type of cluster, Rancher connects to a Kubernetes cluster that has already been set up. Therefore, Rancher does not provision Kubernetes, but only sets up the Rancher agents to communicate with the cluster.

### Rancher Server Components and Source Code

This diagram shows each component that the Rancher server is composed of:




The GitHub repositories for Rancher can be found at the following links:

- Main Rancher server repository [https://github.com/rancher/rancher]
- Rancher UI [https://github.com/rancher/ui]
- Rancher API UI [https://github.com/rancher/api-ui]
- Norman, [https://github.com/rancher/norman] Rancher's API framework
- Types [https://github.com/rancher/types]
- Rancher CLI [https://github.com/rancher/cli]
- Catalog applications [https://github.com/rancher/helm]

This is a partial list of the most important Rancher repositories. For more details about Rancher source code, refer to the section on contributing to Rancher. To see all libraries and projects used in Rancher, see the `go.mod` file [https://github.com/rancher/rancher/blob/master/go.mod] in the `rancher/rancher` repository.

## SUSE Rancher Prime Server and Components

The majority of Rancher 2.x software runs on the Rancher Server. Rancher Server includes all the software components used to manage the entire Rancher deployment.

The diagram below illustrates the high-level architecture of Rancher 2.x. The figure depicts a Rancher Server installation that manages two downstream Kubernetes clusters: one created by RKE2 and another created by Amazon EKS (Elastic Kubernetes Service).

For the best performance and security, we recommend a dedicated Kubernetes cluster for the Rancher management server. Running user workloads on this cluster is not advised. After deploying Rancher, you can create or import clusters for running your workloads.

The diagram shows how users can manipulate both Rancher-launched Kubernetes clusters and hosted Kubernetes clusters through Rancher's authentication proxy:

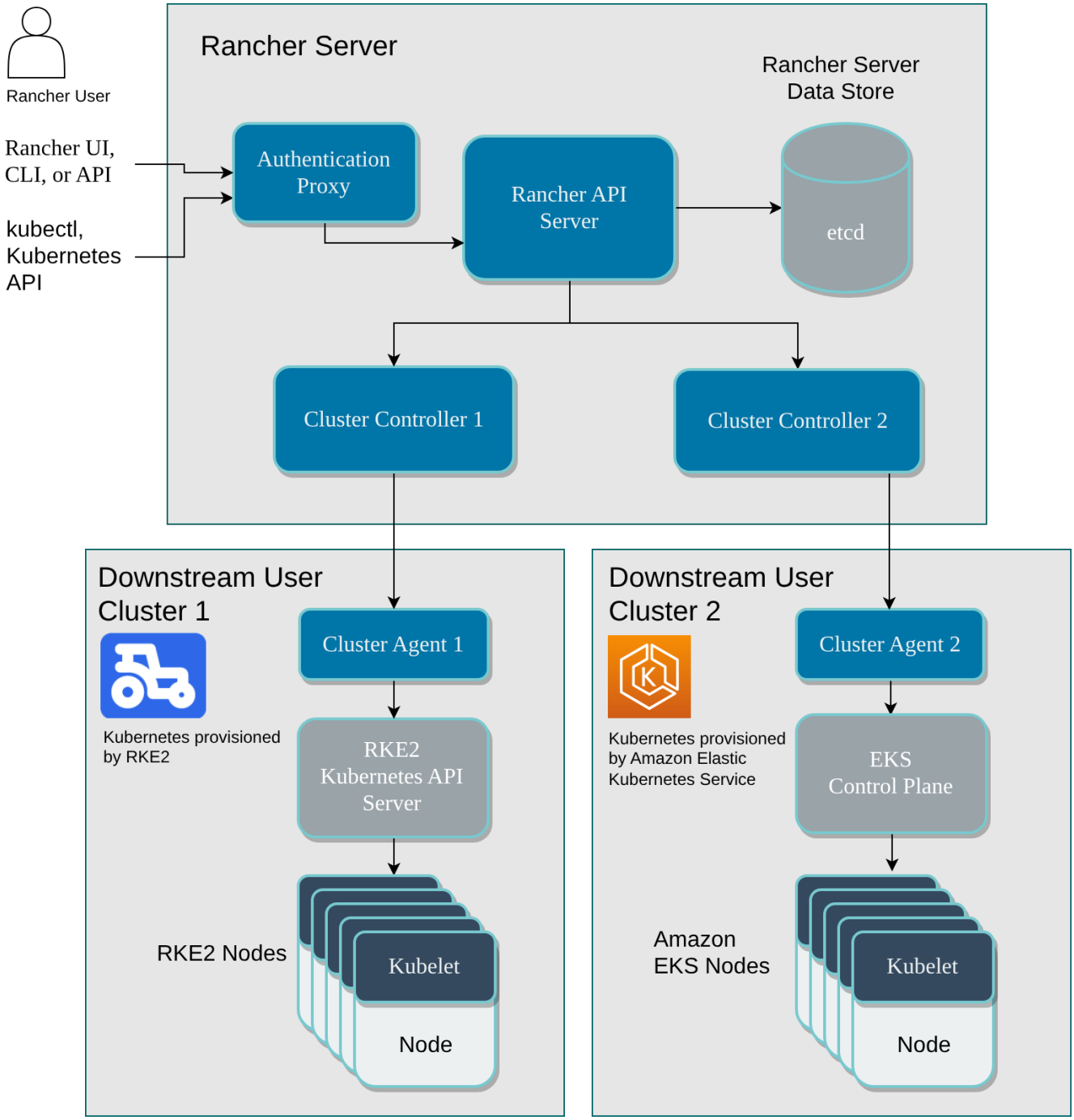



*Figure 4. Managing Kubernetes Clusters through Rancher's Authentication Proxy*

You can install Rancher on a single node, or on a high-availability Kubernetes cluster.

A high-availability Kubernetes installation is recommended for production.

A Docker installation of Rancher is recommended only for development and testing purposes. The ability to migrate Rancher to a high-availability cluster depends on the Rancher version.

To see deployed resources, run the following command:

```
kubectl get all -n <namespace>
```

If you have administrator privileges, you can also see these resources listed in the Rancher UI:

1. Click ☰ and select a cluster.
2. Select **More Resources** from the side navigation menu to view deployed resources by type.
3. Select **Cluster** > **Projects/Namespaces** from the side navigation menu, then select a namespace, to view deployed resources by namespace. The Rancher backup operator can be used to migrate Rancher from the single Docker container install to an installation on a high-availability Kubernetes cluster. For details, refer to the documentation on migrating Rancher to a new cluster.

The Rancher server, regardless of the installation method, should always run on nodes that are separate from the downstream user clusters that it manages. If Rancher is installed on a high-availability Kubernetes cluster, it should run on a separate cluster from the cluster(s) it manages.

# Kubernetes Concepts

This page explains concepts related to Kubernetes that are important for understanding how Rancher works. The descriptions below provide a simplified overview of Kubernetes components. For more details, refer to the official documentation on Kubernetes components. [https://kubernetes.io/docs/concepts/overview/components/]

## About Docker

Docker is a container packaging and runtime standard. Developers can build container images from Dockerfiles and distribute container images from Docker registries. Docker Hub [https://hub.docker.com] is a popular public registry. Many organizations also set up private Docker registries. Docker is primarily used to manage containers on individual nodes.

## About Kubernetes

Kubernetes is the container cluster management standard. YAML files specify containers and other resources that form an application. Kubernetes performs functions such as scheduling, scaling, service discovery, health check, secret management, and configuration management.

## What is a Kubernetes Cluster?

A cluster is a group of computers that work together as a single system.

A *Kubernetes Cluster* is a cluster that uses the Kubernetes container-orchestration system [https://kubernetes.io/] to deploy, maintain, and scale containers, allowing your organization to automate application operations.

## Roles for Nodes in Kubernetes Clusters

Each computing resource in a Kubernetes cluster is called a *node*. Nodes can be either bare-metal servers or virtual machines. Kubernetes classifies nodes into three types: *etcd* nodes, *control plane* nodes, and *worker* nodes.

A Kubernetes cluster consists of at least one etcd, controlplane, and worker node.

### etcd Nodes

Rancher uses etcd as a data store in both single node and high-availability installations. In Kubernetes, etcd is also a role for nodes that store the cluster state.

The state of a Kubernetes cluster is maintained in etcd. [https://kubernetes.io/docs/concepts/overview/components/#etcd] The etcd nodes run the etcd database.

The etcd database component is a distributed key-value store used as Kubernetes storage for all cluster data, such as cluster coordination and state management. It is recommended to run etcd on multiple nodes so that there's always a backup available for failover.

Although you can run etcd on just one node, etcd requires a majority of nodes, a quorum, to agree on updates to the cluster state. The cluster should always contain enough healthy etcd nodes to form a quorum. For a cluster with n members, a quorum is (n/2)+1. For any odd-sized cluster, adding one node will always increase the number of nodes necessary for a quorum.

Three etcd nodes is generally sufficient for smaller clusters and five etcd nodes for large clusters.

#### Controlplane Nodes

Controlplane nodes run the Kubernetes API server, scheduler, and controller manager. These nodes take care of routine tasks to ensure that your cluster maintains your configuration. Because all cluster data is stored on your etcd nodes, control plane nodes are stateless. You can run control plane on a single node, although three or more nodes are recommended for redundancy. Additionally, a single node can share the control plane and etcd roles.

#### Worker Nodes

Each worker node [https://kubernetes.io/docs/concepts/architecture/nodes/] runs the following:

- **Kubelets:** An agent that monitors the state of the node, ensuring your containers are healthy.
- **Workloads:** The containers and pods that hold your apps, as well as other types of deployments.

Worker nodes also run storage and networking drivers, and ingress controllers when required. You create as many worker nodes as necessary to run your workloads.

### About Helm

For high-availability installations of Rancher, Helm is the tool used to install Rancher on a Kubernetes cluster.

Helm is the package management tool of choice for Kubernetes. Helm charts provide templating syntax for Kubernetes YAML manifest documents. With Helm we can create configurable deployments instead of just using static files. For more information about creating your own catalog of deployments, check out the docs at https://helm.sh.

For more information on service accounts and cluster role binding, refer to the Kubernetes documentation. [https://kubernetes.io/docs/reference/access-authn-authz/rbac/]

## Glossary

This page covers Rancher-specific terminology and symbols which might be unfamiliar, or which differ between Rancher versions.

## Icons & Symbols



The navigation menu toggle, a button in the far top left corner of the Rancher UI. Clicking the  opens a sidebar that provides access to the main sections of the Rancher interface.



The submenu toggle, a button found at the far right of many table rows throughout the Rancher UI. Clicking the  opens a submenu containing a contextual list of tasks related to the item in that row.

## A

### Apps

**Versions:** Rancher v2.6.5 and later When describing navigation within Rancher, the section of the Rancher UI where you manage Helm charts and Helm chart repositories. In earlier versions of Rancher, this section was called *Apps & Marketplace*, or *Catalogs*.

**Synonyms:** *Apps & Marketplace, Catalogs*

### Apps & Marketplace

**Versions:** Rancher v2.5.0—2.6.5 The section of the Rancher UI where you manage Helm charts and Helm chart repositories. In earlier versions of Rancher, this section was called *Catalogs*. In later versions, it is called *Apps*.

**Synonyms:** *Apps, Catalogs*

## C

### Catalogs

**Versions:** 1. Rancher v2.4 and earlier 2. Rancher v2.7 and later The section of the Rancher UI where you manage Helm charts and Helm chart repositories. In later versions of Rancher, this section is called *Apps & Marketplace*, or *Apps*.

**Synonyms:** *Apps, Apps & Marketplace*

**Related terms:** *Extension catalogs*

### Community

A build of Rancher that's available to the entire Rancher community.

**Related terms:** *Prime, Rancher Prime*

## D

### Downstream cluster

An alternative name for a *managed cluster*, a Kubernetes cluster that is managed by a Rancher server.

**Synonyms:** *Managed cluster*

**Related terms:** *Hosted cluster, Imported cluster, Rancher server, Registered cluster, Upstream cluster*

## E

### Extension

**Versions:** Rancher v2.7.0 and later Helm charts that can extend and enhance the Rancher UI, and tailor Rancher to your specific environment. Some Extensions require manual installation on a cluster, while others are built into Rancher. Built-in Extensions include Fleet and Harvester. Manually installed Extensions include Kubewarden and Elemental.

**Related terms:** *Apps & Marketplace, Catalogs, Integrations*

### Extension catalogs

**Versions:** Rancher v2.7.0 and later Container image repositories that can be imported to add UI extensions to your Rancher instance.

## H

### Hosted cluster

A managed cluster that is hosted in the cloud, by a Kubernetes provider. Rancher supports a variety of hosted Kubernetes providers, including Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), and Google Kubernetes Engine (GKE). Rancher integrates with the hosted Kubernetes provider's cloud APIs, so that you can manage the cluster from the Rancher UI.

**Related terms:** *Downstream cluster, Imported cluster, Managed cluster, Registered cluster*

## I

### Imported cluster

**Versions:** Rancher v2.0—2.4 An existing cluster that was provisioned by a third party, then set up to be managed by Rancher. *Registered clusters* in later versions of Rancher are roughly synonymous with imported clusters. The main difference is that there are more management features exposed in the Rancher UI for registered clusters than for imported clusters.

**Related terms:** *Downstream cluster, Hosted cluster, Managed cluster, Reistered cluster*

## K

### K3s

A lightweight Kubernetes distribution chiefly designed for workloads that run unattended under tight resource constraints.

**Related terms:** *RKE, RKE2*

## L

### Local cluster

An alternative name for a *Rancher server*, the Kubernetes cluster that hosts Rancher and manages other clusters. Here, *local* refers to the location of the Rancher installation, not to the user's local workstation. Local clusters can be hosted on-premises, in air-gapped environments, or in the cloud.

**Synonyms:** *Rancher server, Upstream cluster*

**Related terms:** *Downstream cluster, Hosted cluster, Imported cluster, Managed cluster, Registered cluster*

## M

### Managed cluster

A Kubernetes cluster that is managed by a Rancher server. Managed clusters can run on a variety of infrastructure and can be located anywhere, including in the cloud, on-premises, or in air-gapped environments.

**Synonyms:** *Downstream cluster*

**Related terms:** *Hosted cluster, Imported cluster, Rancher server, Registered cluster*

## R

### Rancher

A Kubernetes management tool to deploy and run clusters anywhere and on any provider. Rancher includes all the software and downstream cluster components used to manage the entire Rancher deployment.

### Rancher Kubernetes Engine (RKE)

A CNCF-certified Kubernetes distribution that runs entirely within Docker containers. There are two main versions of RKE available. The 1.x software line is sometimes called *RKE1*, or simply *RKE*. The next-generation line is called *RKE2*.

**Synonyms:** *RKE1*

**Related terms:** *K3s, RKE2*

### Rancher Prime

Rancher Prime is a new edition of the commercial, enterprise offering built on the same source code. Installation assets are hosted on a trusted registry owned and managed by Rancher with additional value coming in from security assurances, extended lifecycles, access to focused architectures and Kubernetes advisories.

### Rancher server

The Kubernetes cluster that hosts Rancher and manages the other clusters in the Rancher deployment.

**Synonyms:** *Local cluster, Upstream cluster*

**Related terms:** *Downstream cluster, Hosted cluster, Imported cluster, Managed cluster, Registered cluster*

### Registered cluster

**Versions:** Rancher v2.5 and later An existing cluster that was provisioned by a third party, then set up to be managed by Rancher. Older versions of Rancher use *Imported cluster* to refer to a similar concept. The main difference is that there are more management features exposed in the Rancher UI for registered clusters than for imported clusters.

**Related terms:** *Downstream cluster, Hosted cluster, Imported cluster, Managed cluster*

### RKE1

An alternative name for the v1.x software line of *Rancher Kubernetes Engine (RKE)*. Sometimes called *RKE*

for short. RKE1 is a certified Kubernetes distribution and CLI/library which creates and manages a Kubernetes cluster.

**Synonyms:** *Rancher Kubernetes Engine*

**Related terms:** *K3s, RKE2*

### RKE2

**Versions:** Rancher v2.6.0 and later (to provision clusters), Rancher 2.5.0 and later (to register clusters) A fully conformant version of the Rancher Kubernetes Engine (RKE) that is designed for security and compliance with US federal government standards.

**Related terms:** *K3s, RKE*

### Roles

Roles determine what actions a Rancher user can make within a cluster or project.

## U

### Upstream cluster

An alternative name for the *Rancher server*, the Kubernetes cluster that hosts Rancher and manages the other clusters in the Rancher deployment.

**Synonyms:** *Local cluster, Rancher server*

**Related terms:** *Downstream cluster, Hosted cluster, Imported cluster, Managed cluster, Registered cluster*

### User

A Rancher resource `users.management.cattle.io` that defines a user within Rancher.

## W

### Workload

Objects that set deployment rules for pods. Based on these rules, Kubernetes performs the deployment and updates the workload with the current state of the application. Workloads let you define the rules for application scheduling, scaling, and upgrade.

# Installation and Upgrades

This section provides an overview of the architecture options of installing Rancher, describing advantages of each option.

## Terminology

- **The Rancher server** manages and provisions Kubernetes clusters. You can interact with downstream Kubernetes clusters through the Rancher server's user interface. The Rancher management server can be installed on any Kubernetes cluster, including hosted clusters, such as Amazon EKS clusters.
- **K3s (Lightweight Kubernetes)** is also a fully compliant Kubernetes distribution. It is newer than RKE, easier to use, and more lightweight, with a binary size of less than 100 MB.
- **RKE2** is a fully conformant Kubernetes distribution that focuses on security and compliance within the U.S. Federal Government sector.

## Overview of Installation Options

Rancher can be installed on these main architectures.

### High-availability Kubernetes Install with the Helm CLI

We recommend using Helm, a Kubernetes package manager, to install Rancher on multiple nodes on a dedicated Kubernetes cluster. For RKE clusters, three nodes are required to achieve a high-availability cluster. For K3s clusters, only two nodes are required.

### Rancher on EKS Install

Rancher can be installed on to Amazon Elastic Kubernetes Service (EKS). For details, refer to installation-and-upgrade/hosted-kubernetes/rancher-on-amazon-eks.pdf.

You can also use SUSE Rancher for AWS [https://documentation.suse.com/en-us/cloudnative/rancher-srfa/] as a distinct, fully managed Software-as-a-Service (SaaS) offering through the AWS Marketplace.

### Single-node Kubernetes Install

Rancher can be installed on a single-node Kubernetes cluster. In this case, the Rancher server doesn't have high availability, which is important for running Rancher in production.

However, this option is useful if you want to save resources by using a single node in the short term, while preserving a high-availability migration path. In the future, you can add nodes to the cluster to get a high-availability Rancher server.

### Other Options

There are also separate instructions for installing Rancher in an air gap environment or behind an HTTP proxy:

| Level of Internet Access | Kubernetes Installation - Strongly Recommended |
|---|---|
| With direct access to the Internet | Docs |

| Level of Internet Access | Kubernetes Installation - Strongly Recommended |
|---|---|
| Behind an HTTP proxy | Docs |
| In an air gap environment | Docs |

We recommend installing Rancher on a Kubernetes cluster, because in a multi-node cluster, the Rancher management server becomes highly available. This high-availability configuration helps maintain consistent access to the downstream Kubernetes clusters that Rancher will manage.

For that reason, we recommend that for a production-grade architecture, you should set up a high-availability Kubernetes cluster, then install Rancher on it. After Rancher is installed, you can use Rancher to deploy and manage Kubernetes clusters.

Our instructions for installing Rancher on Kubernetes describe how to first use K3s or RKE to create and manage a Kubernetes cluster, then install Rancher onto that cluster.

When the nodes in your Kubernetes cluster are running and fulfill the node requirements, you will use Helm to deploy Rancher onto Kubernetes. Helm uses Rancher's Helm chart to install a replica of Rancher on each node in the Kubernetes cluster. We recommend using a load balancer to direct traffic to each replica of Rancher in the cluster.

For a longer discussion of Rancher architecture, refer to the architecture overview, recommendations for production-grade architecture, or our best practices guide.

# Prerequisites

Before installing Rancher, make sure that your nodes fulfill all of the installation requirements.

# Architecture Tip

For the best performance and greater security, we recommend a separate, dedicated Kubernetes cluster for the Rancher management server. Running user workloads on this cluster is not advised. After deploying Rancher, you can create or import clusters for running your workloads.

For more architecture recommendations, refer to this page.

### More Options for Installations on a Kubernetes Cluster

Refer to the Helm chart options for details on installing Rancher on a Kubernetes cluster with other configurations, including:

- With API auditing to record all transactions
- With TLS termination on a load balancer
- With a custom Ingress

In the Rancher installation instructions, we recommend using K3s or RKE to set up a Kubernetes cluster before installing Rancher on the cluster. Both K3s and RKE have many configuration options for customizing the Kubernetes cluster to suit your specific environment. For the full list of their capabilities, refer to their documentation:

- RKE configuration options [https://rancher.com/docs/rke/latest/en/config-options/]

- K3s configuration options [https://rancher.com/docs/k3s/latest/en/installation/install-options/]

# SUSE Rancher Prime Deployment Quick Start Guides



The intent of these guides is to quickly launch a sandbox that you can use to evaluate Rancher. These guides are not intended for production environments. For comprehensive setup instructions, see Installation.

Use this section of the docs to jump start your deployment and testing of Rancher 2.x. It contains instructions for a simple Rancher setup and some common use cases. We plan on adding more content to this section in the future.

We have Quick Start Guides for:

- Deploying Rancher Server: Get started running Rancher using the method most convenient for you.
- Deploying Workloads: Deploy a simple workload [https://kubernetes.io/docs/concepts/workloads/] and expose it, letting you access it from outside the cluster.

## Deploying SUSE Rancher Prime Server

Use one of the following guides to deploy and provision Rancher and a Kubernetes cluster in the provider of your choice.

- Amazon EKS

If you prefer, the following guide will take you through the same process in individual steps. Use this if you want to run Rancher in a different provider, on prem, or if you would just like to see how easy it is.

- Manual Install

Alternatively, you can also use SUSE Rancher for AWS [https://documentation.suse.com/en-us/cloudnative/rancher-srfa/] as a distinct, fully managed Software-as-a-Service (SaaS) offering through the AWS Marketplace.

#### Helm CLI Quick Start

These instructions capture a quick way to set up a proof-of-concept Rancher installation.

These instructions assume you have a Linux virtual machine that you will communicate with from your local workstation. Rancher will be installed on the Linux machine. You will need to retrieve the IP address of that machine so that you can access Rancher from your local workstation. Rancher is designed to manage Kubernetes clusters remotely, so any Kubernetes cluster that Rancher manages in the future will also need to be able to reach this IP address.

We don't recommend installing Rancher locally because it creates a networking problem. Installing Rancher on localhost does not allow Rancher to communicate with downstream Kubernetes clusters, so on localhost you wouldn't be able to test Rancher's cluster provisioning or cluster management functionality.

Your Linux machine can be anywhere. It could be an Amazon EC2 instance, a Digital Ocean droplet, or an Azure virtual machine, to name a few examples. Other Rancher docs often use 'node' as a generic term for all of these. One possible way to deploy a Linux machine is by setting up an Amazon EC2 instance as shown in this tutorial.

The full installation requirements are here.

#### Install SUSE® Rancher Prime: K3s on Linux

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the Rancher Support Matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/].

To specify the K3s (Kubernetes) version, use the INSTALL_K3S_VERSION (e.g., `INSTALL_K3S_VERSION="v1.24.10+k3s1"`) environment variable when running the K3s installation script.

Install a K3s cluster by running this command on the Linux machine:

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=<VERSION> sh -s -
server --cluster-init
```

Using `--cluster-init` allows K3s to use embedded etcd as the datastore and has the ability to convert to an HA setup. Refer to High Availability with Embedded DB [https://rancher.com/docs/k3s/latest/en/installation/ha-embedded/].

Save the IP of the Linux machine.

#### Save the kubeconfig to your workstation

The kubeconfig file is important for accessing the Kubernetes cluster. Copy the file at `/etc/rancher/k3s/k3s.yaml` from the Linux machine and save it to your local workstation in the directory `~/.kube/config`. One way to do this is by using the `scp` tool and run this command on your local machine:

**Mac and Linux**

```
scp root@<IP_OF_LINUX_MACHINE>:/etc/rancher/k3s/k3s.yaml
~/.kube/config
```

In some cases it may need to make sure that your shell has the environment variable `KUBECONFIG=~/.kube/config` defined, for instance, it can be exported in your profile or rc files.

**Windows**

By default, "scp" is not a recognized command, so we need to install a module first.

In Windows Powershell:

```
Find-Module Posh-SSH
Install-Module Posh-SSH

## Get the remote kubeconfig file
scp root@<IP_OF_LINUX_MACHINE>:/etc/rancher/k3s/k3s.yaml
```

```
$env:USERPROFILE\.kube\config
```

#### Edit the Rancher server URL in the kubeconfig

In the kubeconfig file, you will need to change the value of the `server` field to `<IP_OF_LINUX_NODE>:6443`. The Kubernetes API server will be reached at port 6443, while the Rancher server will be reached at ports 80 and 443. This edit is needed so that when you run Helm or kubectl commands from your local workstation, you will be able to communicate with the Kubernetes cluster that Rancher will be installed on.

**Mac and Linux**

One way to open the kubeconfig file for editing is to use Vim:

```
vi ~/.kube/config
```

Press `i` to put Vim in insert mode. To save your work, press `Esc`. Then press `:wq` and press `Enter`.

**Windows**

In Windows Powershell, you can use `notepad.exe` for editing the kubeconfig file:

```
notepad.exe $env:USERPROFILE\.kube\config
```

Once edited, either press `ctrl+s` or go to `File > Save` to save your work.

#### Install Rancher with Helm

Then from your local workstation, run the following commands. You will need to have kubectl [https://kubernetes.io/docs/tasks/tools/#kubectl] and helm [https://helm.sh/docs/intro/install/] installed.



To see options on how to customize the cert-manager install (including for cases where your cluster uses PodSecurityPolicies), see the cert-manager docs [https://artifacthub.io/packages/helm/cert-manager/cert-manager#configuration].

```
helm repo add rancher-prime <helm-chart-repo-url>
```



To learn more about the Rancher Prime Helm chart repository URL, see our Prime-only documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#chart-repo-url]. Authentication is required. Use your SUSE Customer Center (SCC) [https://scc.suse.com/] credentials to log in.

```
kubectl create namespace cattle-system

kubectl apply -f https://github.com/cert-manager/cert-
```

```
manager/releases/download/<VERSION>/cert-manager.crds.yaml

helm repo add jetstack https://charts.jetstack.io

helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace

# Windows Powershell
helm install cert-manager jetstack/cert-manager `
  --namespace cert-manager `
  --create-namespace
```

The final command to install Rancher is below. The command requires a domain name that forwards traffic to the Linux machine. For the sake of simplicity in this tutorial, you can use a fake domain name to create your proof-of-concept. An example of a fake domain name would be `<IP_OF_LINUX_NODE>.sslip.io`.

To install a specific Rancher version, use the `--version` flag (e.g., `--version 2.6.6`). Otherwise, the latest Rancher is installed by default.

See Setting up the Bootstrap Password for password requirements.

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=<IP_OF_LINUX_NODE>.sslip.io \
  --set replicas=1 \
  --set bootstrapPassword=<PASSWORD_FOR_RANCHER_ADMIN>

# Windows Powershell

helm install rancher rancher-prime/rancher `
  --namespace cattle-system `
  --set hostname=<IP_OF_LINUX_NODE>.sslip.io `
  --set replicas=1 `
  --set bootstrapPassword=<PASSWORD_FOR_RANCHER_ADMIN>
```

Now if you navigate to `<IP_OF_LINUX_NODE>.sslip.io` in a web browser, you should see the Rancher UI.

To make these instructions simple, we used a fake domain name and self-signed certificates to do this installation. Therefore, you will probably need to add a security exception to your web browser to see the Rancher UI. Note that for production installs, you would need a high-availability setup with a load balancer, a real domain name and real certificates.

These instructions also left out the full installation requirements and other installation options. If you have

any issues with these steps, refer to the full Helm CLI installation docs.

To launch new Kubernetes clusters with your new Rancher server, you may need to set up cloud credentials in Rancher. For more information, see Launching Kubernetes clusters with Rancher.

#### Rancher for AWS

SUSE Rancher for AWS (SRA/SRFA) is a fully managed Software-as-a-Service (SaaS) offering available through the AWS Marketplace. It provides a centralized control plane to manage, monitor, and scale Kubernetes clusters within an AWS environment.

For deployment prerequisites, architecture requirements, and subscription details, refer to the AWS Marketplace listing [https://aws.amazon.com/marketplace/pp/prodview-yrzugbpzuukww]. For more information, refer to the SUSE Rancher for AWS documentation [https://documentation.suse.com/cloudnative/rancher-srfa/].



SUSE Rancher for AWS is an offering distinct from deploying and managing the Rancher server manually on Amazon EKS. If you intend to install and manage your own Rancher deployment on an EKS cluster, refer to Installing Rancher on Amazon EKS.

## Deploying Workloads

These guides walk you through the deployment of an application, including how to expose the application for use outside of the cluster.

- Workload with Ingress
- Workload with NodePort

#### Workload with Ingress Quick Start

##### Prerequisite

You have a running cluster with at least `1` node.

##### 1. Deploying a Workload

You're ready to create your first Kubernetes workload [https://kubernetes.io/docs/concepts/workloads/]. A workload is an object that includes pods along with other files and info needed to deploy your application.

For this workload, you'll be deploying the application Rancher Hello-World.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click **Workload**.
4. Click **Create**.
5. Click **Deployment**.
6. Enter a **Name** for your workload.
7. From the **Container Image** field, enter `rancher/hello-world`. This field is case-sensitive.
8. Click **Add Port** and `Cluster IP` for the `Service Type` and enter `80` in the **Private Container Port** field. You may leave the `Name` blank or specify any name that you wish. Adding a port enables access to

the application inside and outside of the cluster. For more information, see Services.

9. Click **Create**.

**Result:**

- Your workload is deployed. This process might take a few minutes to complete.
- When your workload completes deployment, it's assigned a state of **Active**. You can view this status from the project's **Workloads** page.

#### 2. Expose The Application Via An Ingress

Now that the application is up and running, it needs to be exposed so that other services can connect.

1. Click ☰ **> Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click **Service Discovery › Ingresses**.
4. Click **Create.**
5. When choosing **Namespace**, ensure it is the same as the one used when you created your deployment. Otherwise, your deployment will not be available when you attempt to select **Target Service**, as in Step 8 below.
6. Enter a **Name**, such as **hello**.
7. Specify your **Path**, such as `/hello`.
8. In the **Target Service** field, drop down the list and choose the name that you set for your service.
9. In the **Port** field, drop down the list and select `80`.
10. Click **Create** at the bottom right.

**Result:** The application is assigned a `sslip.io` address and exposed. It may take a minute or two to populate.

#### View Your Application

From the **Deployments** page, find the **Endpoints** column for your deployment and click on an endpoint. The endpoints available will depend on how you configured the port you added to your deployment. For endpoints where you do not see a randomly assigned port, append the path you specified when creating the ingress to the IP address. For example, if your endpoint looks like `xxx.xxx.xxx.xxx` or `https://xxx.xxx.xxx.xxx` change it to `xxx.xxx.xxx.xxx/hello` or `https://xxx.xxx.xxx.xxx/hello`.

Your application will open in a separate window.

#### Finished

Congratulations! You have successfully deployed a workload exposed via an ingress.

#### What's Next?

When you're done using your sandbox, destroy the Rancher Server and your cluster.

### Workload with NodePort Quick Start

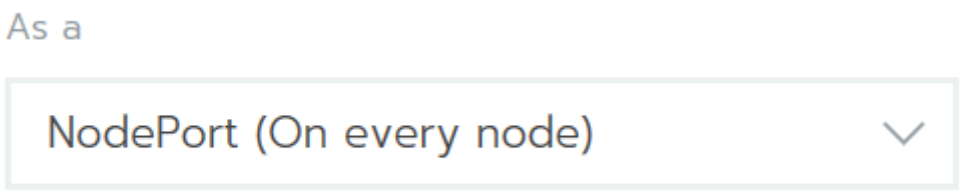
You must have a running cluster with at least 1 node.

#### 1. Deploying a Workload

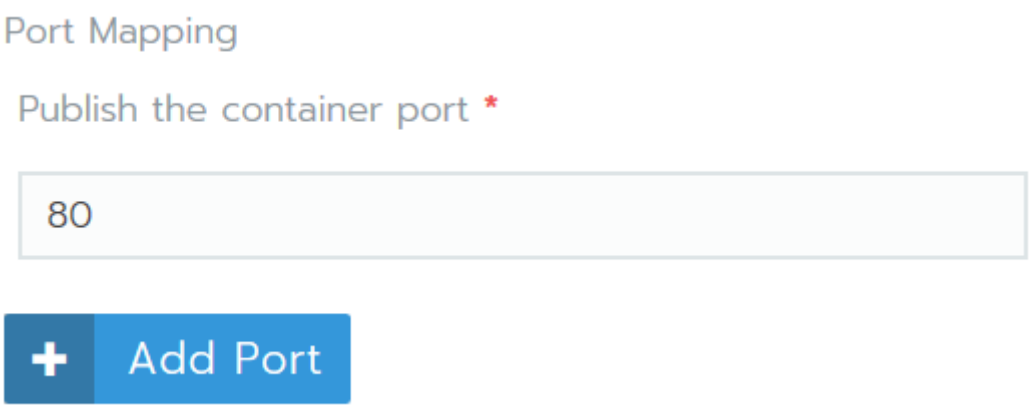
You're ready to create your first Kubernetes workload [https://kubernetes.io/docs/concepts/workloads/]. A workload is an object that includes pods along with other files and info needed to deploy your application.

For this workload, you'll be deploying the application Rancher Hello-World.

1. Click ☰ > **Cluster Management**.
2. From the **Clusters** page, go to the cluster where the workload should be deployed and click **Explore**.
3. Click **Workload**.
4. Click **Create**.
5. Enter a **Name** for your workload.
6. From the **Container Image** field, enter `rancher/hello-world`. This field is case-sensitive.
7. Click **Add Port**.
8. From the **Service Type** drop-down, make sure that **NodePort** is selected.




9. From the **Publish the container port** field, enter port `80`.




10. Click **Create**.

**Result:**

- Your workload is deployed. This process might take a few minutes to complete.
- When your workload completes deployment, it's assigned a state of **Active**. You can view this status from the project's **Workloads** page.

#### 2. Viewing Your Application

From the **Workloads** page, click the link underneath your workload. If your deployment succeeded, your application opens.

#### Attention: Cloud-Hosted Sandboxes

When using a cloud-hosted virtual machine, you may not have access to the port running the container. In this event, you can test Nginx in an ssh session on the local machine using `Execute Shell`. Use the port number after the `:` in the link under your workload if available, which is `31568` in this example.

```
gettingstarted@rancher:~$ curl http://localhost:31568
<!DOCTYPE html>
<html>
  <body>
    <img id="logo" src="img/rancher-logo.svg" alt="Rancher logo"
width=400 />
    <h1>Hello world!</h1>
    <h3>My hostname is hello-world-66b4b9d88b-78bhx</h3>
    <div id='Services'>
      <h3>k8s services found 2</h3>

      <b>INGRESS_D1E1A394F61C108633C4BD37AEDDE757</b>
tcp://10.43.203.31:80<br />

      <b>KUBERNETES</b> tcp://10.43.0.1:443<br />

    </div>
    <br />

    <div id='rancherLinks' class="row social">
      <a class="p-a-xs" href="https://rancher.com/docs"><img
src="img/favicon.png" alt="Docs" height="25" width="25"></a>
      <a class="p-a-xs" href="https://slack.rancher.io/"><img
src="img/icon-slack.svg" alt="slack" height="25" width="25"></a>
      <a class="p-a-xs" href="https://github.com/rancher/rancher"><img
src="img/icon-github.svg" alt="github" height="25" width="25"></a>
      <a class="p-a-xs" href="https://twitter.com/Rancher_Labs"><img
src="img/icon-twitter.svg" alt="twitter" height="25" width="25"></a>
      <a class="p-a-xs" href=
"https://www.facebook.com/rancherlabs/"><img src="img/icon-facebook.svg"
alt="facebook" height="25" width="25"></a>
      <a class="p-a-xs"
href="https://www.linkedin.com/groups/6977008/profile"><img
src="img/icon-linkedin.svg" height="25" alt="linkedin" width="25"></a>
```

```
    </div>
    <br />
    <button class='button' onclick='myFunction()'>Show request
details</button>
    <div id="reqInfo" style='display:none'>
      <h3>Request info</h3>
      <b>Host:</b> 172.22.101.111:31411 <br />
      <b>Pod:</b> hello-world-66b4b9d88b-78bhx </b><br />

      <b>Accept:</b> [*/*]<br />

      <b>User-Agent:</b> [curl/7.47.0]<br />

    </div>
    <br />
    <script>
      function myFunction() {
          var x = document.getElementById("reqInfo");
          if (x.style.display === "none") {
              x.style.display = "block";
          } else {
              x.style.display = "none";
          }
      }
    </script>
  </body>
</html>
gettingstarted@rancher:~$
```

#### Finished

Congratulations! You have successfully deployed a workload exposed via a NodePort.

#### What's Next?

When you're done using your sandbox, destroy the Rancher Server and your cluster.

## Installation Requirements

This page describes the software, hardware, and networking requirements for the nodes where the Rancher server will be installed. The Rancher server can be installed on a single node or a high-availability Kubernetes cluster.



If you install Rancher on a Kubernetes cluster, requirements are different from the node requirements for downstream user clusters, which will run your apps and services.

The Rancher UI works best in Firefox or Chromium based browsers (Chrome, Edge, Opera, Brave, etc).

See our page on best practices for a list of recommendations for running a Rancher server in production.

## Kubernetes Compatibility with Rancher

Rancher needs to be installed on a supported Kubernetes version. Consult the Rancher support matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions] to ensure that your intended version of Kubernetes is supported.

Regardless of version and distribution, the Kubernetes cluster must have the aggregation API layer properly configured to support the extension API used by Rancher.

### Install Rancher on a Hardened Kubernetes Cluster

If you install Rancher on a hardened Kubernetes cluster, check the Exempting Required Rancher Namespaces section for detailed requirements.

### Install Rancher on an IPv6-only or Dual-stack Kubernetes Cluster

You can deploy Rancher on an IPv6-only or dual-stack Kubernetes cluster.

For details on Rancher's IPv6-only and dual-stack support, see the IPv4/IPv6 Dual-stack page.

## Operating Systems and Container Runtime Requirements

Rancher works with any modern Linux distribution. All supported operating systems can run in x86_64, ARM64, or mixed-architecture environments.

### Operating Systems and CPU Architectures

Here are the requirements for configuring the Rancher Management Server:

- **Rancher Management Server**:
  - **OS Requirement**: Must run exclusively on Linux.
  - **Supported Architectures**: `x86_64`, `ARM64`, or mixed-architecture node pools.

### Container Runtime Requirements for Rancher Management Server

Because the Rancher Management Server is deployed via Helm onto an underlying Kubernetes cluster, host container runtime requirements are determined by that cluster's distribution:

- **RKE2 and K3s (Recommended)**: Use an embedded `containerd` runtime. No host-level container runtime installation is required prior to deployment.
- **Hosted Kubernetes (EKS, AKS, GKE)**: Uses the cloud provider's default CRI-compliant container runtime (such as `containerd` or `CRI-O`).

### Multi-Architecture Image Handling

All core Rancher Management Server container images (including `rancher/rancher`, `rancher-webhook`, and `fleet`) are published as multi-architecture container manifests (`linux/amd64` and `linux/arm64`).

The host container runtime automatically detects the CPU architecture of each node and pulls the matching `x86_64` or `ARM64` image variant at runtime. No architecture-specific image tags or custom Helm

configuration flags are required.

#### Node Time Synchronization (NTP)

Ensure the `ntp` package (or an equivalent service like `chrony` or `systemd-timesyncd`) is installed and active on all host nodes. Synchronized time across cluster nodes prevents TLS certificate validation errors and communication failures.

#### Networking and Firewall Requirements

Default host firewall rules on some Linux distributions can block internal Kubernetes cluster communication:

- **firewalld**: Must be disabled on nodes, as default rules conflict with Kubernetes Container Network Interface (CNI) plugins.
- **Alternative firewalld Configuration**: If your security policy requires `firewalld` to remain active, you must configure a dedicated `firewalld` zone with an `ACCEPT` policy targeting the Pod CIDR network and internal node communication.

#### SUSE® Rancher Prime: RKE2 Specific Requirements

RKE2 bundles its own container runtime, containerd.

For details on which OS versions were tested with RKE2, refer to the Rancher support matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions].

#### SUSE® Rancher Prime: K3s Specific Requirements

For the container runtime, K3s bundles its own containerd by default. Alternatively, you can configure K3s to use an already installed Docker runtime. For more information on using K3s with Docker see the K3s documentation. [https://documentation.suse.com/cloudnative/k3s/latest/en/advanced.html#_using_docker_as_the_container_runtime]

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the Rancher support matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions]. To specify the K3s version, use the INSTALL_K3S_VERSION environment variable when running the K3s installation script.

If you are installing Rancher on a K3s cluster with **Raspbian Buster**, follow these steps [https://rancher.com/docs/k3s/latest/en/advanced/#enabling-legacy-iptables-on-raspbian-buster] to switch to legacy iptables.

If you are installing Rancher on a K3s cluster with Alpine Linux, follow these steps [https://rancher.com/docs/k3s/latest/en/advanced/#additional-preparation-for-alpine-linux-setup] for additional setup.

## Hardware Requirements

The following sections describe the CPU, memory, and I/O requirements for nodes where Rancher is installed. Requirements vary based on the size of the infrastructure.

#### Practical Considerations

Rancher's hardware footprint depends on a number of factors, including:

- Size of the managed infrastructure (e.g., node count, cluster count).

- Complexity of the desired access control rules (e.g., `RoleBinding` object count).
- Number of workloads (e.g., Kubernetes deployments, Fleet deployments).
- Usage patterns (e.g., subset of functionality actively used, frequency of use, number of concurrent users).

Since there are a high number of influencing factors that may vary over time, the requirements listed here should be understood as reasonable starting points that work well for most use cases. Nevertheless, your use case may have different requirements. For inquiries about a specific scenario please contact Rancher [https://rancher.com/contact/] for further guidance.

In particular, requirements on this page are subject to typical use assumptions, which include:

- Under 60,000 total Kubernetes resources, per type.
- Up to 120 pods per node.
- Up to 200 CRDs in the upstream (local) cluster.
- Up to 100 CRDs in downstream clusters.
- Up to 50 Fleet deployments.

Higher numbers are possible but requirements might be higher. If you have more than 20,000 resources of the same type, loading time of the whole list through the Rancher UI might take several seconds.



*Evolution:*

Rancher's codebase evolves, use cases change, and the body of accumulated Rancher experience grows every day.

Hardware requirement recommendations are subject to change over time, as guidelines improve in accuracy and become more concrete.

If you find that your Rancher deployment no longer complies with the listed recommendations, contact Rancher [https://rancher.com/contact/] for a re-evaluation.

**SUSE® Rancher Prime: RKE2 Kubernetes**

The following table lists minimum CPU and memory requirements for each node in the upstream cluster.

Please note that a highly available setup with at least three nodes is required for production.

| Managed Infrastructure Size | Maximum Number of Clusters | Maximum Number of Nodes | vCPUs | RAM |
|---|---|---|---|---|
| Small | 150 | 1500 | 4 | 16 GB |
| Medium | 300 | 3000 | 8 | 32 GB |
| Large (*) | 500 | 5000 | 16 | 64 GB |
| Larger (†) | (†) | (†) | (†) | (†) |

(*): Large deployments require that you follow best practices for adequate performance.

(†): Larger deployment sizes are generally possible with ad-hoc hardware recommendations and tuning. You can contact Rancher [https://rancher.com/contact/] for a custom evaluation.

Refer to RKE2 documentation for more detailed information on RKE2 general requirements

[https://documentation.suse.com/cloudnative/rke2/latest/en/install/requirements.html].

#### SUSE® Rancher Prime: K3s Kubernetes

The following table lists minimum CPU and memory requirements for each node in the upstream cluster.

Please note that a highly available setup with at least three nodes is required for production.

| Managed Infrastructure Size | Maximum Number of Clusters | Maximum Number of Nodes | vCPUs | RAM | External Database Host (*) |
|---|---|---|---|---|---|
| Small | 150 | 1500 | 4 | 16 GB | 2 vCPUs, 8 GB + 1000 IOPS |
| Medium | 300 | 3000 | 8 | 32 GB | 4 vCPUs, 16 GB + 2000 IOPS |
| Large (†) | 500 | 5000 | 16 | 64 GB | 8 vCPUs, 32 GB + 4000 IOPS |

(*): External Database Host refers to hosting the K3s cluster data store on an dedicated external host [https://documentation.suse.com/cloudnative/k3s/latest/en/datastore/datastore.html]. This is optional. Exact requirements depend on the external data store.

(†): Large deployments require that you follow best practices for adequate performance.

Refer to the K3s documentation for more detailed information on general requirements [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/requirements.html].

#### Hosted Kubernetes

The following table lists minimum CPU and memory requirements for each node in the upstream cluster.

Please note that a highly available setup with at least three nodes is required for production.

These requirements apply to hosted Kubernetes clusters such as Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), or Google Kubernetes Engine (GKE). They don't apply to Rancher SaaS solutions such as Rancher Prime Hosted [https://www.rancher.com/products/rancher].

| Managed Infrastructure Size | Maximum Number of Clusters | Maximum Number of Nodes | vCPUs | RAM |
|---|---|---|---|---|
| Small | 150 | 1500 | 4 | 16 GB |
| Medium | 300 | 3000 | 8 | 32 GB |
| Large (*) | 500 | 5000 | 16 | 64 GB |

(*): Large deployments require that you follow best practices for adequate performance.

### Ingress

Each node in the Kubernetes cluster that Rancher is installed on should run an Ingress.

The Ingress should be deployed as DaemonSet to ensure your load balancer can successfully route traffic to all nodes.

For RKE2 and K3s installations, you don't have to install the Ingress manually because it is installed by

default.

For hosted Kubernetes clusters (EKS, GKE, AKS), you will need to set up the ingress.

- **Amazon EKS:** For details on how to install Rancher on Amazon EKS, including how to install an ingress so that the Rancher server can be accessed, refer to this page.
- **AKS:** For details on how to install Rancher with Azure Kubernetes Service, including how to install an ingress so that the Rancher server can be accessed, refer to this page.
- **GKE:** For details on how to install Rancher with Google Kubernetes Engine, including how to install an ingress so that the Rancher server can be accessed, refer to this page.

## Disks

Rancher performance depends on etcd in the cluster performance. To ensure optimal speed, we recommend always using SSD disks to back your Rancher management Kubernetes cluster. On cloud providers, you will also want to use the minimum size that allows the maximum IOPS. In larger clusters, consider using dedicated storage devices for etcd data and wal directories.

## Networking Requirements

This section describes the networking requirements for the node(s) where the Rancher server is installed.



If a server containing Rancher has the `X-Frame-Options=DENY` header, some pages in the new Rancher UI will not be able to render after upgrading from the legacy UI. This is because some legacy pages are embedded as iFrames in the new UI.

### Node IP Addresses

Each node used should have a static IP configured, regardless of whether you are installing Rancher on a single node or on an HA cluster. In case of DHCP, each node should have a DHCP reservation to make sure the node gets the same IP allocated.

### Port Requirements

To operate properly, Rancher requires a number of ports to be open on Rancher nodes and on downstream Kubernetes cluster nodes. Port Requirements lists all the necessary ports for Rancher and Downstream Clusters for the different cluster types.

### Load Balancer Requirements

If you use a load balancer, it should be HTTP/2 compatible.

To receive help from SUSE Support, Rancher Prime customers who use load balancers (or any other middleboxes such as firewalls), must use one that is HTTP/2 compatible.

When HTTP/2 is not available, Rancher falls back to HTTP/1.1. However, since HTTP/2 offers improved web application performance, using HTTP/1.1 can create performance issues.

## Port Requirements

To operate properly, Rancher requires a number of ports to be open on Rancher nodes and on downstream

Kubernetes cluster nodes.

### Rancher Nodes

The following table lists the ports that need to be open to and from nodes that are running the Rancher server.

The port requirements differ based on the Rancher server architecture.

Rancher can be installed on any Kubernetes cluster. For Rancher installs on a K3s, RKE, or RKE2 Kubernetes cluster, refer to the tabs below. For other Kubernetes distributions, refer to the distribution's documentation for the port requirements for cluster nodes.



- Rancher nodes may also require additional outbound access for any external authentication provider which is configured (LDAP for example).
- Kubernetes recommends TCP 30000-32767 for node port services.
- For firewalls, traffic may need to be enabled within the cluster and pod CIDR.
- Rancher nodes may also need outbound access to an external S3 location which is used for storing cluster backups (Minio for example).

#### Ports for Rancher Server Nodes on SUSE® Rancher Prime: K3s

*Click to expand*

The K3s server needs port 6443 to be accessible by the nodes.

The nodes need to be able to reach other nodes over UDP port 8472 when Flannel VXLAN is used. The node should not listen on any other port. K3s uses reverse tunneling such that the nodes make outbound connections to the server and all kubelet traffic runs through that tunnel. However, if you do not use Flannel and provide your own custom CNI, then port 8472 is not needed by K3s.

If you wish to utilize the metrics server, you will need to open port 10250 on each node.



The VXLAN port on nodes should not be exposed to the world as it opens up your cluster network to be accessed by anyone. Run your nodes behind a firewall/security group that disables access to port 8472.

The following tables break down the port requirements for inbound and outbound traffic:

*Table 1. Inbound Rules for Rancher Server Nodes*

| Protocol | Port | Source | Description |
|---|---|---|---|
| TCP | 80 | Load balancer/proxy that does external SSL termination | Rancher UI/API when external SSL termination is used |
| TCP | 443 | • K3s server and agent nodes<br>• Hosted/Registered Kubernetes<br>• Any source that needs to be able to use the Rancher UI or API | Rancher agent, Rancher UI/API, kubectl |

| Protocol | Port | Source | Description |
|---|---|---|---|
| TCP | 6443 | K3s server nodes | Kubernetes API |
| UDP | 8472 | K3s server and agent nodes | Required only for Flannel VXLAN. |
| TCP | 10250 | K3s server and agent nodes | kubelet |

*Table 2. Outbound Rules for Rancher Nodes*

| Protocol | Port | Destination | Description |
|---|---|---|---|
| TCP | 22 | Any node IP from a node created using Node Driver | SSH provisioning of nodes using Node Driver |
| TCP | 443 | git.rancher.io | Rancher catalog |
| TCP | 2376 | Any node IP from a node created using Node driver | Docker daemon TLS port used by Docker Machine |
| TCP | 6443 | Hosted/Imported Kubernetes API | Kubernetes API server |

#### Ports for Rancher Server Nodes on SUSE® Rancher Prime: RKE2

*Click to expand*

The RKE2 server needs port 6443 and 9345 to be accessible by other nodes in the cluster.

All nodes need to be able to reach other nodes over UDP port 8472 when Flannel VXLAN is used.

If you wish to utilize the metrics server, you will need to open port 10250 on each node.



The VXLAN port on nodes should not be exposed to the world as it opens up your cluster network to be accessed by anyone. Run your nodes behind a firewall/security group that disables access to port 8472.

*Table 3. Inbound Rules for RKE2 Server Nodes*

| Protocol | Port | Source | Description |
|---|---|---|---|
| TCP | 9345 | RKE2 server and agent nodes | Node registration. Port should be open on all server nodes to all other nodes in the cluster. |
| TCP | 6443 | RKE2 agent nodes | Kubernetes API |
| UDP | 8472 | RKE2 server and agent nodes | Required only for Flannel VXLAN |
| TCP | 10250 | RKE2 server and agent nodes | kubelet |
| TCP | 2379 | RKE2 server nodes | etcd client port |
| TCP | 2380 | RKE2 server nodes | etcd peer port |
| TCP | 30000-32767 | RKE2 server and agent nodes | NodePort port range. Can use TCP or UDP. |
| TCP | 5473 | Calico-node pod connecting to typha pod | Required when deploying with Calico |
| HTTP | 80 | Load balancer/proxy that does external SSL termination | Rancher UI/API when external SSL termination is used |

| Protocol | Port | Source | Description |
|---|---|---|---|
| HTTPS | 443 | • Hosted/Registered Kubernetes<br>• Any source that needs to be able to use the Rancher UI or API | Rancher agent, Rancher UI/API, kubectl. Not needed if you have a load balancer doing TLS termination. |

Typically all outbound traffic is allowed.

#### Ports for Rancher Server in Docker

*Click to expand*

The following tables break down the port requirements for Rancher nodes, for inbound and outbound traffic:

*Table 4. Inbound Rules for Rancher Node*

| Protocol | Port | Source | Description |
|---|---|---|---|
| TCP | 80 | Load balancer/proxy that does external SSL termination | Rancher UI/API when external SSL termination is used |
| TCP | 443 | • Hosted/Registered Kubernetes<br>• Any source that needs to be able to use the Rancher UI or API | Rancher agent, Rancher UI/API, kubectl |

*Table 5. Outbound Rules for Rancher Node*

| Protocol | Port | Source | Description |
|---|---|---|---|
| TCP | 22 | Any node IP from a node created using Node Driver | SSH provisioning of nodes using Node Driver |
| TCP | 443 | git.rancher.io | Rancher catalog |
| TCP | 2376 | Any node IP from a node created using a node driver | Docker daemon TLS port used by Docker Machine |
| TCP | 6443 | Hosted/Imported Kubernetes API | Kubernetes API server |

### Downstream Kubernetes Cluster Nodes

Downstream Kubernetes clusters run your apps and services. This section describes what ports need to be opened on the nodes in downstream clusters so that Rancher can communicate with them.

The port requirements differ depending on how the downstream cluster was launched. Each of the tabs below list the ports that need to be opened for different cluster types.

The following diagram depicts the ports that are opened for each cluster type.

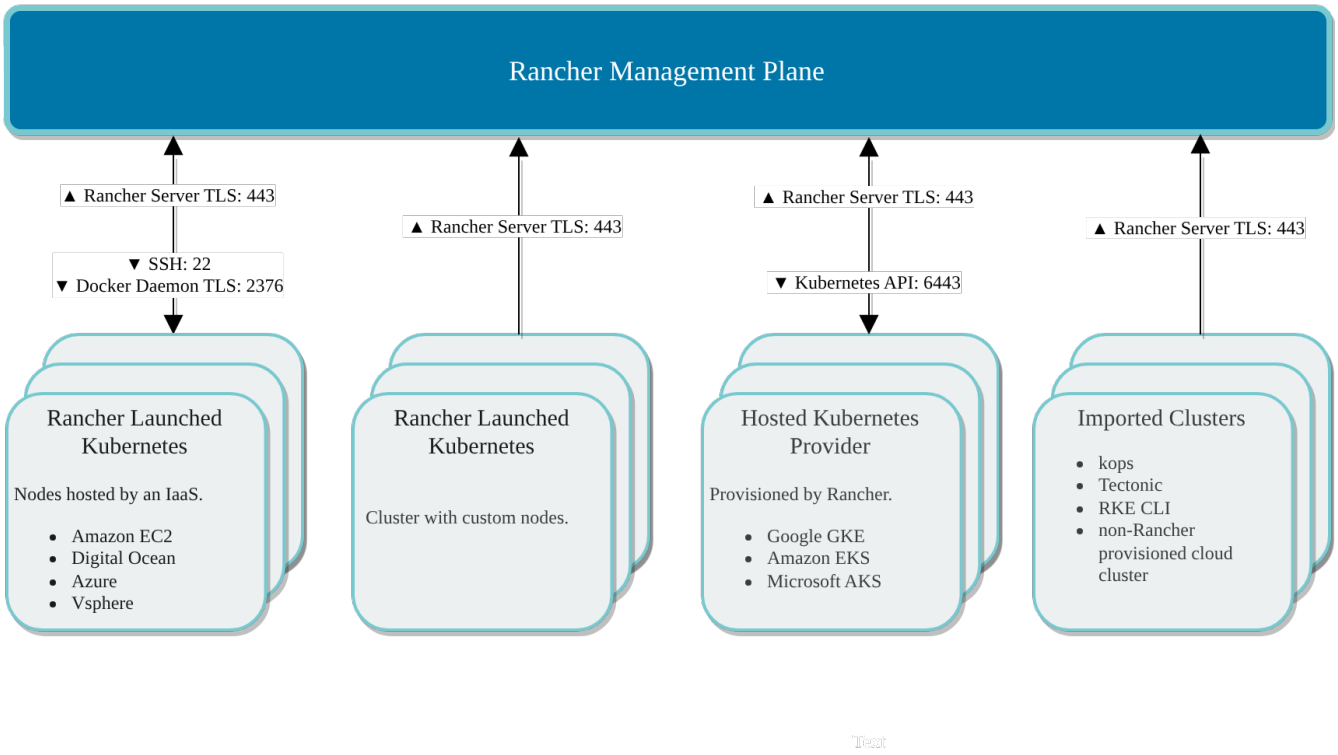



*Figure 5. Port Requirements for the Rancher Management Plane*

If security isn't a large concern and you're okay with opening a few additional ports, you can use the table in Commonly Used Ports as your port reference instead of the comprehensive tables below.

#### Ports for SUSE Virtualization Clusters

Refer to the SUSE Virtualization Integration Overview for more information on Harvester port requirements.

#### Ports for Rancher Launched Kubernetes Clusters using Node Pools

*Click to expand*

The following table depicts the port requirements for Rancher Launched Kubernetes with nodes created in an Infrastructure Provider.

The required ports are automatically opened by Rancher during creation of clusters in cloud providers like Amazon EC2 or DigitalOcean.

| From / To | Rancher Nodes | etcd Plane Nodes | Control Plane Nodes | Worker Plane Nodes | External Rancher Load Balancer | Internet |
|---|---|---|---|---|---|---|
| Rancher Nodes [1] | | 22 TCP | | | | git.rancher.io |
| | | 2376 TCP | | | | |
| etcd Plane Nodes | 443 TCP [3] | 2379 TCP | | | 443 TCP | |
| | | 2380 TCP | | | | |
| | | | 6443 TCP | | | |
| | | 8472 UDP | | | | |
| | | 9099 TCP [4] | | | | |

| From / To | Rancher Nodes | etcd Plane Nodes | Control Plane Nodes | Worker Plane Nodes | External Rancher Load Balancer | Internet |
|---|---|---|---|---|---|---|
| Control Plane Nodes | 443 TCP [3] | 2379 TCP | | | 443 TCP | |
| | | 2380 TCP | | | | |
| | | | 6443 TCP | | | |
| | | 8472 UDP | | | | |
| | | 10250 TCP | | | | |
| | | | 9099 TCP [4] | | | |
| | | | 10254 TCP [4] | | | |
| Worker Plane Nodes | 443 TCP [3] | | 6443 TCP | | 443 TCP | |
| | | 8472 UDP | | | | |
| | | | | 9099 TCP [4] | | |
| | | | | 10254 TCP [4] | | |
| Kubernetes API Clients | | | 6443 TCP [5] | | | |
| Workload Clients or Load Balancer | | | | 30000-32767 TCP / UDP (nodeport) | | |
| | | | | 80 TCP (Ingress) | | |
| | | | | 443 TCP (Ingress) | | |

Notes:

1. Nodes running standalone server or Rancher HA deployment.
2. Required to fetch Rancher chart library.
3. Only without external load balancer in front of Rancher.
4. Local traffic to the node itself (not across nodes).
5. Only if Authorized Cluster Endpoints are activated.

**Ports for Rancher Launched Kubernetes Clusters using Custom Nodes**

*Click to expand*

The following table depicts the port requirements for Rancher Launched Kubernetes with Custom Nodes.

| From / To | Rancher Nodes | etcd Plane Nodes | Control Plane Nodes | Worker Plane Nodes | External Rancher Load Balancer | Internet |
|---|---|---|---|---|---|---|
| Rancher Nodes [1] | | | | | | git.rancher.io |
| etcd Plane Nodes | 443 TCP [3] | 2379 TCP | | | 443 TCP | |
| | | 2380 TCP | | | | |
| | | | 6443 TCP | | | |
| | | 8472 UDP | | | | |
| | | 4789 UDP [6] | | | | |
| | | 9099 TCP [4] | | | | |

| From / To | Rancher Nodes | etcd Plane Nodes | Control Plane Nodes | Worker Plane Nodes | External Rancher Load Balancer | Internet |
|---|---|---|---|---|---|---|
| Control Plane Nodes | 443 TCP [3] | 2379 TCP | | | 443 TCP | |
| | | 2380 TCP | | | | |
| | | | 6443 TCP | | | |
| | | 8472 UDP | | | | |
| | | 4789 UDP [6] | | | | |
| | | 10250 TCP | | | | |
| | | | 9099 TCP [4] | | | |
| | | | 10254 TCP [4] | | | |
| Worker Plane Nodes | 443 TCP [3] | | 6443 TCP | | 443 TCP | |
| | | 8472 UDP | | | | |
| | | 4789 UDP [6] | | | | |
| | | | | 9099 TCP [4] | | |
| | | | | 10254 TCP [4] | | |
| Kubernetes API Clients | | | 6443 TCP [5] | | | |
| Workload Clients or Load Balancer | | | | 30000-32767 TCP / UDP (nodeport) | | |
| | | | | 80 TCP (Ingress) | | |
| | | | | 443 TCP (Ingress) | | |

Notes:

1. Nodes running standalone server or Rancher HA deployment.
2. Required to fetch Rancher chart library.
3. Only without external load balancer in front of Rancher.
4. Local traffic to the node itself (not across nodes), if you've enabled optional features such as Rancher Monitoring.
5. Only if Authorized Cluster Endpoints are activated.
6. Only if using Overlay mode on Windows cluster.

**Ports for Hosted Kubernetes Clusters**

*Click to expand*

The following table depicts the port requirements for hosted clusters.

| From / To | Rancher Nodes | Hosted / Imported Cluster | External Rancher Load Balancer | Internet |
|---|---|---|---|---|
| Rancher Nodes [1] | 80 TCP | Kubernetes API Endpoint Port [2] | | git.rancher.io |
| | | 8443 TCP | | |
| | | 9443 TCP | | |
| Hosted / Imported Cluster | 443 TCP [4][5] | | 443 TCP [5] | |

| From / To | Rancher Nodes | Hosted / Imported Cluster | External Rancher Load Balancer | Internet |
|---|---|---|---|---|
| Kubernetes API Clients | | Cluster / Provider Specific [6] | | |
| Workload Client | | Cluster / Provider Specific [7] | | |

Notes:

1. Nodes running standalone server or Rancher HA deployment.
2. Only for hosted clusters.
3. Required to fetch Rancher chart library.
4. Only without external load balancer.
5. From worker nodes.
6. For direct access to the Kubernetes API without Rancher.
7. Usually Ingress backed by infrastructure load balancer and/or nodeport.

#### Ports for Registered Clusters



Registered clusters were called imported clusters before Rancher v2.5.

*Click to expand*

The following table depicts the port requirements for registered clusters.

| From / To | Rancher Nodes | Hosted / Imported Cluster | External Rancher Load Balancer | Internet |
|---|---|---|---|---|
| Rancher Nodes [1] | 80 TCP | Kubernetes API Endpoint Port [2] | | git.rancher.io |
| | | 8443 TCP | | |
| | | 9443 TCP | | |
| Hosted / Imported Cluster | 443 TCP [4][5] | | 443 TCP [5] | |
| Kubernetes API Clients | | Cluster / Provider Specific [6] | | |
| Workload Client | | Cluster / Provider Specific [7] | | |

Notes:

1. Nodes running standalone server or Rancher HA deployment.
2. Only for hosted clusters.
3. Required to fetch Rancher chart library.
4. Only without external load balancer.
5. From worker nodes.
6. For direct access to the Kubernetes API without Rancher.
7. Usually Ingress backed by infrastructure load balancer and/or nodeport.

### Other Port Considerations

#### Commonly Used Ports

These ports are typically opened on your Kubernetes nodes, regardless of what type of cluster it is.

| Protocol | Port | Description |
|---|---|---|
| TCP | 22 | Node driver SSH provisioning |
| TCP | 179 | Calico BGP Port |
| TCP | 2376 | Node driver Docker daemon TLS port |
| TCP | 2379 | etcd client requests |
| TCP | 2380 | etcd peer communication |
| UDP | 8472 | Canal/Flannel VXLAN overlay networking |
| UDP | 4789 | Flannel VXLAN overlay networking on Windows cluster |
| TCP | 8443 | Rancher webhook |
| TCP | 9099 | Canal/Flannel livenessProbe/readinessProbe |
| TCP | 9443 | Rancher webhook |
| TCP | 9796 | Default port required by Monitoring to scrape metrics from Linux and Windows node-exporters |
| TCP | 6783 | Weave Port |
| UDP | 6783-6784 | Weave UDP Ports |
| TCP | 10250 | Metrics server communication with all nodes API |
| TCP | 10254 | Ingress controller livenessProbe/readinessProbe |
| TCP/UDP | 30000-32767 | NodePort port range |

#### Local Node Traffic

Ports marked as `local traffic` (i.e., `9099 TCP`) in the above requirements are used for Kubernetes healthchecks (`livenessProbe` and`readinessProbe`). These healthchecks are executed on the node itself. In most cloud environments, this local traffic is allowed by default.

However, this traffic may be blocked when:

- You have applied strict host firewall policies on the node.
- You are using nodes that have multiple interfaces (multihomed).

In these cases, you have to explicitly allow this traffic in your host firewall, or in case of public/private cloud hosted machines (i.e. AWS or OpenStack), in your security group configuration. Keep in mind that when using a security group as source or destination in your security group, explicitly opening ports only applies to the private interface of the nodes / instances.

#### Rancher AWS EC2 Security Group

When using the AWS EC2 node driver to provision cluster nodes in Rancher, you can choose to let Rancher create a security group called `rancher-nodes`. The following rules are automatically added to this security group.

| Type | Protocol | Port Range | Source/Destination | Rule Type |
|---|---|---|---|---|
| SSH | TCP | 22 | 0.0.0.0/0 and ::/0 | Inbound |
| HTTP | TCP | 80 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom TCP Rule | TCP | 443 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom TCP Rule | TCP | 8443 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom TCP Rule | TCP | 2376 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom TCP Rule | TCP | 6443 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom TCP Rule | TCP | 179 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 5473 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 9345 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 2379-2380 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 10250-10252 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 10256 | sg-xxx (rancher-nodes) | Inbound |
| Custom UDP Rule | UDP | 4789 | sg-xxx (rancher-nodes) | Inbound |
| Custom UDP Rule | UDP | 8472 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 9796 | sg-xxx (rancher-nodes) | Inbound |
| Custom TCP Rule | TCP | 30000-32767 | 0.0.0.0/0 and ::/0 | Inbound |
| Custom UDP Rule | UDP | 30000-32767 | 0.0.0.0/0 and ::/0 | Inbound |
| All traffic | All | All | 0.0.0.0/0 and ::/0 | Outbound |

#### Opening SUSE Linux Ports

SUSE Linux may have a firewall that blocks all ports by default. To open the ports needed for adding the host to a custom cluster,

**SLES 15 / openSUSE Leap 15**

1. SSH into the instance.
2. Start YaST in text mode:

```
sudo yast2
```

3. Navigate to **Security and Users** > **Firewall** > **Zones:public** > **Ports**. To navigate within the interface, follow these instructions [https://doc.opensuse.org/documentation/leap/reference/html/book-reference/cha-yast-text.html#sec-yast-cli-navigate].
4. To open the required ports, enter them into the **TCP Ports** and **UDP Ports** fields. In this example, ports 9796 and 10250 are also opened for monitoring. The resulting fields should look similar to the following:

```
 TCP Ports
 22, 80, 443, 2376, 2379, 2380, 6443, 9099, 9796, 10250, 10254,
30000-32767
 UDP Ports
```

```
8472, 30000-32767
```

5. When all required ports are enter, select **Accept**.

**SLES 12 / openSUSE Leap 42**

1. SSH into the instance.
2. Edit `/etc/sysconfig/SuSEfirewall2` and open the required ports. In this example, ports 9796 and 10250 are also opened for monitoring:

```
FW_SERVICES_EXT_TCP="22 80 443 2376 2379 2380 6443 9099 9796
10250 10254 30000:32767"
FW_SERVICES_EXT_UDP="8472 30000:32767"
FW_ROUTE=yes
```

3. Restart the firewall with the new ports:

```
SuSEfirewall2
```

**Result:** The node has the open ports required to be added to a custom cluster.

## Helm Version Requirements

This section contains the requirements for Helm, which is the tool used to install Rancher on a high-availability Kubernetes cluster.

#### Identifying the Proper Helm Version

Select a Helm version compatible with your Kubernetes version range from the Rancher Support Matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions].

Helm is backward compatible with three previous minor versions of Kubernetes (n-3). It does not guarantee forward compatibility with newer Kubernetes versions.

To apply this rule, you may need to reference the following two external resources:

- **Helm Version Compatibility:** Refer to the Helm Version Support Policy [https://helm.sh/docs/topics/version_skew/] and select the version matching the rule for your Rancher target version.
- **Rancher's Kubernetes Support Range:** Use the Rancher Support Matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions] to identify the Kubernetes versions supported by your target Rancher version.

##### Applying Compatibility Rules

- **Scenario:** You are targeting a specific Rancher version that supports a range of Kubernetes versions.
- **Application:** You need a Helm version that supports this entire Kubernetes range. Helm does not provide forward compatibility. You must verify compatibility with the highest supported Kubernetes version.

- **Result:** You find multiple Helm versions that support your target Kubernetes range.
- When multiple versions work, use the newest Helm version that supports the Kubernetes range.

#### Additional Notes

- When using tools that run Helm commands for you (like Terraform), you must make sure they are configured to use the correct Helm version.

# Best Practices for the SUSE Rancher Prime Server

This guide contains our recommendations for running the Rancher server, and is intended to be used in situations in which Rancher manages downstream Kubernetes clusters.

## Recommended Architecture and Infrastructure

Refer to this guide for our general advice for setting up the Rancher server for a production installation.

## Deployment Strategies

This guide is designed to help you choose whether a regional deployment strategy or a hub-and-spoke deployment strategy is better for a Rancher server that manages downstream Kubernetes clusters.

## Installing Rancher in a VMware vSphere Environment

This guide outlines a reference architecture for installing Rancher in a vSphere environment, in addition to standard vSphere best practices as documented by VMware.

## SUSE Rancher Prime Deployment Strategy

There are two recommended deployment strategies for a Rancher instance that manages downstream Kubernetes clusters. Each one has its own pros and cons. Read more about which one would fit best for your use case.

#### Hub & Spoke Strategy

---

In this deployment scenario, there is a single Rancher instance managing Kubernetes clusters across the globe. The Rancher instance would be run on a high-availability Kubernetes cluster, and there would be impact due to latencies.

#### Pros

- Single control plane interface to view/see all regions and environments.
- Kubernetes does not require Rancher to operate and can tolerate losing connectivity to the Rancher instance.

#### Cons

- Subject to network latencies.
- If Rancher goes down, global provisioning of new services is unavailable until it is restored. However,

each Kubernetes cluster can continue to be managed individually.

#### Regional Strategy

---

In the regional deployment model a Rancher instance is deployed in close proximity to the downstream Kubernetes clusters.

##### Pros

- Rancher functionality in regions stay operational if a Rancher instance in another region goes down.
- Network latency between Rancher and downstream clusters is greatly reduced, improving the performance of functionality in Rancher.
- Upgrades of Rancher can be done independently per region.

##### Cons

- Overhead of managing multiple Rancher installations.
- Visibility into Kubernetes clusters in different regions requires multiple interfaces/panes of glass.

## Tips for Running SUSE Rancher Prime

This guide is geared toward use cases where Rancher is used to manage downstream Kubernetes clusters. The high-availability setup is intended to prevent losing access to downstream clusters if the Rancher server is not available.

A high-availability Kubernetes installation, defined as an installation of Rancher on a Kubernetes cluster with at least three nodes, should be used in any production installation of Rancher, as well as any installation deemed "important." Multiple Rancher instances running on multiple nodes ensure high availability that cannot be accomplished with a single node environment.

If you are installing Rancher in a vSphere environment, refer to the best practices documented here.

When you set up your high-availability Rancher installation, consider the following:

#### Minimize Third-Party Software on the Upstream Cluster

We generally recommend running Rancher on a dedicated cluster, free of other workloads, to avoid potential performance and compatibility issues.

Rancher, especially when managing a growing number of clusters, nodes, and workloads, places a significant load on core Kubernetes components like `etcd` and `kube-apiserver` on the upstream cluster. Third-party software can interfere with the performance of these components and Rancher, potentially leading to instability.

Furthermore, third-party software can functionally interfere with Rancher. To minimize compatibility risks, deploy only essential Kubernetes system components and Rancher on the upstream cluster.

The following applications and components generally do not interfere with Rancher or the Kubernetes system and installing them on the upstream cluster is supported:

- Rancher internal components, such as Fleet

- Rancher extensions
- Cluster API components
- CNIs, CPIs, CSIs
- Cloud controller managers
- Observability and monitoring tools (except prometheus-rancher-exporter)

Note that each of these components has its own minimum resource requirements, which must be met in addition to Rancher's. For high-scale deployments, also consider dedicating separate nodes to non-Rancher software using taints and tolerations [https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/] to minimize interference.

Other software can interfere with Rancher and is therefore not supported on the upstream cluster.

Note in particular the following software is known to interfere with Rancher performance:

- CrossPlane [https://www.crossplane.io/]
- Argo CD [https://argoproj.github.io/cd/]
- Flux [https://fluxcd.io/]
- prometheus-rancher-exporter [https://github.com/David-VTUK/prometheus-rancher-exporter] (see issue 33 [https://github.com/David-VTUK/prometheus-rancher-exporter/issues/33])
- Container registries such as SUSE Private Registry, which can require significant bandwidth for serving images

##### Guidance for Container Registries

Container registries, such as SUSE Private Registry, can consume significant network bandwidth when serving images. This demand increases with the number of images, the frequency of image pulls, and the quantity of clusters and container runtimes they serve. Due to this potential for interference with Rancher UI and API traffic, we recommend against running container registries on the same cluster as the Rancher management server.

Regardless of your deployment strategy for a container registry, ensure sufficient bandwidth is available, ideally reserved using Quality of Service (QoS) mechanisms.

Consider the following recommendations based on your needs:

- **Simple Setups (HA Not a Primary Concern):** A container registry deployed as a single Virtual Machine (VM) can be a viable solution.
- **High Availability (HA) Requirements:** We recommend running the registry in a dedicated Kubernetes cluster. All other clusters should then be configured to pull images from this centralized, HA registry.
- **Very Large-Scale or Complex Network Topologies:** Multiple registry clusters might be necessary. These can be deployed in a hierarchical or federated model to efficiently distribute images and manage traffic.

#### Make sure nodes are configured correctly for Kubernetes

It's important to follow K8s and etcd best practices when deploying your nodes, including disabling swap, double checking you have full network connectivity between all machines in the cluster, using unique hostnames, MAC addresses, and product_uids for every node, checking that all correct ports are opened, and deploying with ssd backed etcd. More details can be found in the kubernetes docs [https://kubernetes.io/

docs/setup/production-environment/tools/kubeadm/install-kubeadm/#before-you-begin] and etcd's performance op guide [https://etcd.io/docs/v3.5/op-guide/performance/].

#### Run All Nodes in the Cluster in the Same Datacenter

For best performance, run all three of your nodes in the same geographic datacenter. If you are running nodes in the cloud, such as AWS, run each node in a separate Availability Zone. For example, launch node 1 in us-west-2a, node 2 in us-west-2b, and node 3 in us-west-2c.

#### Development and Production Environments Should be Similar

It's strongly recommended to have a "staging" or "pre-production" environment of the Kubernetes cluster that Rancher runs on. This environment should mirror your production environment as closely as possible in terms of software and hardware configuration.

#### Monitor Your Clusters to Plan Capacity

The Rancher server's Kubernetes cluster should run within the system and hardware requirements as closely as possible. The more you deviate from the system and hardware requirements, the more risk you take.

However, metrics-driven capacity planning analysis should be the ultimate guidance for scaling Rancher, because the published requirements take into account a variety of workload types.

Using Rancher, you can monitor the state and processes of your cluster nodes, Kubernetes components, and software deployments through integration with Prometheus, a leading open-source monitoring solution, and Grafana, which lets you visualize the metrics from Prometheus.

After you enable monitoring in the cluster, you can set up alerts to let you know if your cluster is approaching its capacity. You can also use the Prometheus and Grafana monitoring framework to establish a baseline for key metrics as you scale.

## Tuning and Best Practices for SUSE Rancher Prime at Scale

This guide describes the best practices and tuning approaches to scale Rancher setups and the associated challenges with doing so. As systems grow, performance will naturally reduce, but there are steps that can minimize the load put on Rancher and optimize Rancher's ability to manage larger infrastructures.

#### Optimizing Rancher Performance

- Keep Rancher up to date with patch releases. We are continuously improving Rancher with performance enhancements and bug fixes. The latest Rancher release contains all accumulated improvements to performance and stability, plus updates based on developer experience and user feedback.
- Always scale up gradually, and monitor and observe any changes in behavior while doing do. It is usually easier to resolve performance problems as soon as they surface, before other problems obscure the root cause.
- Reduce network latency between the upstream Rancher cluster and downstream clusters to the extent possible. Note that latency is, among other factors, a function of geographic distance - if you require clusters or nodes spread across the world, consider multiple Rancher installations.

#### Minimizing Load on the Upstream Cluster

When scaling up Rancher, one typical bottleneck is resource growth in the upstream (local) Kubernetes

cluster. The upstream cluster contains information for all downstream clusters. Many operations that apply to downstream clusters create new objects in the upstream cluster and require computation from handlers running in the upstream cluster.

#### Minimizing Third-Party Software on the Upstream Cluster

Recommendations outlined in the general Rancher recommendations are particularly important in a high scale context.

#### Managing Your Object Counts

Etcd is the backing database for Kubernetes and for Rancher. The database may eventually encounter limitations to the number of a single Kubernetes resource type it can store. Exact limits vary and depend on a number of factors. However, experience indicates that performance issues frequently arise once a single resource type's object count exceeds 60,000. Often that type is `RoleBinding`.

This is typical in Rancher, as many operations create new `RoleBinding` objects in the upstream cluster as a side effect.

You can reduce the number of `RoleBindings` in the upstream cluster in the following ways:

- Only add users to clusters and projects when necessary.
- Remove clusters and projects when they are no longer needed.
- Only use custom roles if necessary.
- Use as few rules as possible in custom roles.
- Consider whether adding a role to a user is redundant.
- Consider using less, but more powerful, clusters.
- Kubernetes permissions are always "additive" (allow-list) rather than "subtractive" (deny-list). Try to minimize configurations that gives access to all but one aspect of a cluster, project, or namespace, as that will result in the creation of a high number of `RoleBinding` objects.
- Experiment to see if creating new projects or clusters manifests in fewer `RoleBindings` for your specific use case.

#### Using External Authentication

If you have fifty or more users, you should configure an external authentication provider. This is necessary for better performance.

After you configure external authentication, make sure to assign permissions to groups instead of to individual users. This helps reduce the `RoleBinding` object count.

#### RoleBinding Count Estimation

Predicting how many `RoleBinding` objects a given configuration will create is complicated. However, the following considerations can offer a rough estimate:

- For a minimum estimate, use the formula `32C + U + 2UaC + 8P + 5Pa`.
  - `C` is the total number of clusters.
  - `U` is the total number of users.

- `Ua` is the average number of users with a membership on a cluster.
- `P` is the total number of projects.
- `Pa` is the average number of users with a membership on a project.

- The number of `RoleBindings` increases linearly with the number of clusters, projects, and users.

#### Using New Apps Over Legacy Apps

Rancher uses two Kubernetes app resources: `apps.projects.cattle.io` and `apps.cattle.cattle.io`. Legacy apps, represented by `apps.projects.cattle.io`, were introduced with the former Cluster Manager UI and are now outdated. Current apps, represented by `apps.catalog.cattle.io`, are found in the Cluster Explorer UI for their respective cluster. `Apps.cattle.cattle.io` apps are preferable because their data resides in downstream clusters, which frees up resources in the upstream cluster.

You should remove any remaining legacy apps that appear in the Cluster Manager UI, and replace them with apps in the Cluster Explorer UI. Create any new apps only in the Cluster Explorer UI.

#### Using the Authorized Cluster Endpoint (ACE)

An Authorized Cluster Endpoint (ACE) provides access to the Kubernetes API of Rancher-provisioned RKE2 and K3s clusters. When enabled, the ACE adds a context to kubeconfig files generated for the cluster. The context uses a direct endpoint to the cluster, thereby bypassing Rancher. This reduces load on Rancher for cases where unmediated API access is acceptable or preferable. See Authorized Cluster Endpoint for more information and configuration instructions.

#### Reducing Event Handler Executions

The bulk of Rancher's logic occurs on event handlers. These event handlers run on an object whenever the object is updated, and when Rancher is started. Additionally, they run every 10 hours when Rancher syncs caches. In scaled setups these scheduled runs come with huge performance costs because every handler is being run on every applicable object. However, the scheduled handler execution can be disabled with the `CATTLE_SYNC_ONLY_CHANGED_OBJECTS` environment variable. If resource allocation spikes are seen every 10 hours, this setting can help.

The value for `CATTLE_SYNC_ONLY_CHANGED_OBJECTS` can be a comma separated list of the following options. The values refer to types of handlers and controllers (the structures that contain and run handlers). Adding the controller types to the variable disables that set of controllers from running their handlers as part of cache resyncing.

- `mgmt` refers to management controllers which only run on one Rancher node.
- `user` refers to user controllers which run for every cluster. Some of these run on the same node as management controllers, while others run in the downstream cluster. This option targets the former.
- `scaled` refers to scaled controllers which run on every Rancher node. You should avoid setting this value, as the scaled handlers are responsible for critical functions and changes may disrupt cluster stability.

In short, if you notice CPU usage peaks every 10 hours, add the `CATTLE_SYNC_ONLY_CHANGED_OBJECTS` environment variable to your Rancher deployment (in the `spec.containers.env` list) with the value `mgmt,user`

### Optimizations Outside of Rancher

Important influencing factors are the underlying cluster's own performance and configuration. The upstream cluster, if misconfigured, can introduce a bottleneck Rancher software has no chance to resolve.

#### Manage Upstream Cluster Nodes Directly with SUSE® Rancher Prime: RKE2

As Rancher can be very demanding on the upstream cluster, especially at scale, you should have full administrative control of the cluster's configuration and nodes. To identify the root cause of excess resource consumption, use standard Linux troubleshooting techniques and tools. This can aid in distinguishing between whether Rancher, Kubernetes, or operating system components are causing issues.

Although managed Kubernetes services make it easier to deploy and run Kubernetes clusters, they are discouraged for the upstream cluster in high scale scenarios. Managed Kubernetes services typically limit access to configuration and insights on individual nodes and services.

Use RKE2 for large scale use cases.

#### Keep all Upstream Cluster Nodes co-located

To provide high availability, Kubernetes is designed to run nodes and control components in different zones. However, if nodes and control plane components are located in different zones, network traffic may be slower.

Traffic between Rancher components and the Kubernetes API is especially sensitive to network latency, as is etcd traffic between nodes.

To improve performance, run all upstream node clusters in the same location. In particular, make sure that latency between etcd nodes and Rancher is as low as possible.

#### Keeping Kubernetes Versions Up to Date

You should keep the local Kubernetes cluster up to date. This will ensure that your cluster has all available performance enhancements and bug fixes.

#### Optimizing etcd

Etcd is the backend database for Kubernetes and for Rancher. It plays a very important role in Rancher performance.

The two main bottlenecks to etcd performance [https://etcd.io/docs/v3.5/op-guide/performance/] are disk and network speed. Etcd should run on dedicated nodes with a fast network setup and with SSDs that have high input/output operations per second (IOPS). For more information regarding etcd performance, see Slow etcd performance (performance testing and optimization) [https://www.suse.com/support/kb/doc/?id=000020100] and Tuning etcd for Large Installations. Information on disks can also be found in the Installation Requirements.

It's best to run etcd on exactly three nodes, as adding more nodes will reduce operation speed. This may be counter-intuitive to common scaling approaches, but it's due to etcd's replication mechanisms [https://etcd.io/docs/v3.5/faq/#what-is-maximum-cluster-size].

Etcd performance will also be negatively affected by network latency between nodes as that will slow down network communication. Etcd nodes should be located together with Rancher nodes.

#### Browser Requirements

At high scale, Rancher transfers more data from the upstream cluster to UI components running in the browser, and those components also need to perform more processing.

For best performance, ensure that the host running the hardware meets these requirements:

- 2020 i5 10th generation Intel (4 cores) or equivalent
- 8 GB RAM
- Total network bandwidth to the upstream cluster: 72 Mb/s (equivalent to a single 802.11n Wi-Fi 4 link stream, ~8 MB/s http download throughput)
- Round-trip time (ping time) from browser to upstream cluster: 150 ms or less

### Tuning etcd for Large Installations

When Rancher is used to manage a large infrastructure it is recommended to increase the default keyspace for etcd from the default 2 GB. The maximum setting is 8 GB and the host should have enough RAM to keep the entire dataset in memory. When increasing this value you should also increase the size of the host. The keyspace size can also be adjusted in smaller installations if you anticipate a high rate of change of pods during the garbage collection interval.

The etcd data set is automatically cleaned up on a five-minute interval by Kubernetes. There are situations, e.g. deployment thrashing, where enough events could be written to etcd and deleted before garbage collection occurs and cleans things up causing the keyspace to fill up. If you see `mvcc: database space exceeded` errors, in the etcd logs or Kubernetes API server logs, you should consider increasing the keyspace size. This can be accomplished by setting the quota-backend-bytes [https://etcd.io/docs/latest/op-guide/maintenance/#space-quota] setting on the etcd servers.

#### Example: This Snippet of the RKE2/K3s config.yaml file Increases the Keyspace Size to 5GB

*RKE2/K3s config.yaml*

```
etcd-arg:
  - "quota-backend-bytes=5368709120"
```

#### Scaling etcd Disk Performance

For RKE2 and K3s-based Rancher management clusters, achieving optimal etcd performance and stability is paramount. It is highly recommended to mount a dedicated, high-performance flash device (SSD or NVMe) at the database path. This volume should be formatted with a robust, high-performance filesystem such as XFS. This configuration isolates critical etcd I/O operations from the host operating system and any other applications, which is a foundational best practice for preventing I/O contention and ensuring low-latency writes.

- For RKE2, the path is `/var/lib/rancher/rke2/server/db`.
- For K3s, the path is `/var/lib/rancher/k3s/server/db`.



Do not attempt to change the default etcd data directory using etcd-arg or other configuration flags. RKE2 and K3s require the etcd database to remain at its default location. Modifying this path will break built-in cluster operations, such as snapshots and

> restores. To utilize dedicated storage, you must mount the device directly to the default paths listed above.

Additionally, you can follow the recommendations from the etcd docs [https://etcd.io/docs/latest/tuning/#disk] on how to tune the disk priority on the host.

#### Advanced WAL Separation

Based on etcd best practices, mirroring RAID configurations are unnecessary because etcd replicates data between the nodes in the cluster. You can use striping RAID configurations to increase available IOPS.

If you are using dedicated devices to separate the data and `wal` directories, you must mount your `wal` disk to the default `wal` path and explicitly declare the `wal-dir` in your configuration file using that default path (as shown in the examples below). Without specifying it, the etcd process will try to manipulate the underlying `wal` mount with insufficient permissions.

*RKE2 config.yaml*

```
etcd-arg:
  - "data-dir=/var/lib/rancher/rke2/server/db/etcd"
  - "wal-dir=/var/lib/rancher/rke2/server/db/etcd/wal"
```

*K3s config.yaml*

```
etcd-arg:
  - "data-dir=/var/lib/rancher/k3s/server/db/etcd"
  - "wal-dir=/var/lib/rancher/k3s/server/db/etcd/wal"
```

## About High-availability Installations

We recommend using Helm, a Kubernetes package manager, to install Rancher on a dedicated Kubernetes cluster. This is called a high-availability Kubernetes installation because increased availability is achieved by running Rancher on multiple nodes.

In a standard installation, Kubernetes is first installed on three nodes that are hosted in an infrastructure provider such as Amazon's EC2 or Google Compute Engine.

Then Helm is used to install Rancher on top of the Kubernetes cluster. Helm uses Rancher's Helm chart to install a replica of Rancher on each of the three nodes in the Kubernetes cluster. We recommend using a load balancer to direct traffic to each replica of Rancher in the cluster, in order to increase Rancher's availability.

The Rancher server data is stored on etcd. This etcd database also runs on all three nodes, and requires an odd number of nodes so that it can always elect a leader with a majority of the etcd cluster. If the etcd database cannot elect a leader, etcd can fail, requiring the cluster to be restored from backup.

For information on how Rancher works, regardless of the installation method, refer to the architecture section.

### Recommended Architecture

- DNS for Rancher should resolve to a layer 4 load balancer
- The Load Balancer should forward port TCP/80 and TCP/443 to all 3 nodes in the Kubernetes cluster.
- The Ingress controller will redirect HTTP to HTTPS and terminate SSL/TLS on port TCP/443.
- The Ingress controller will forward traffic to port TCP/80 on the pod in the Rancher deployment.

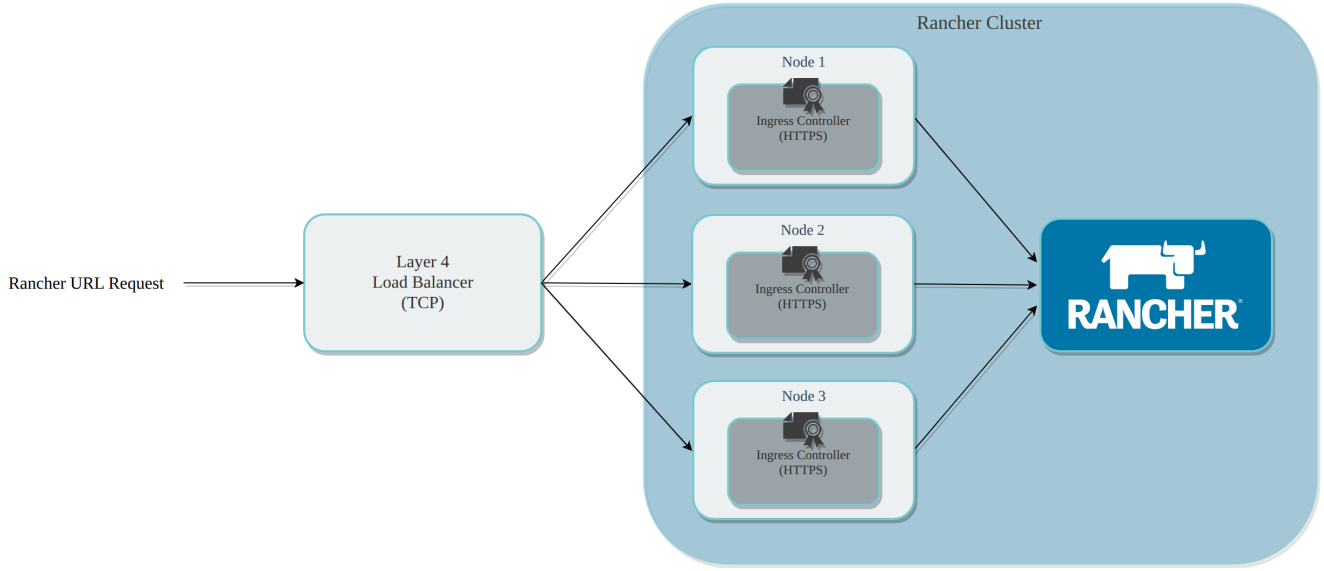



*Figure 6. Kubernetes Rancher install with layer 4 load balancer, depicting SSL termination at ingress controllers*

## Installing SUSE Rancher Prime in a VMware vSphere Environment

This guide outlines a reference architecture for installing Rancher on an RKE Kubernetes cluster in a VMware vSphere environment. It also desctibes standard vSphere best practices as documented by VMware.

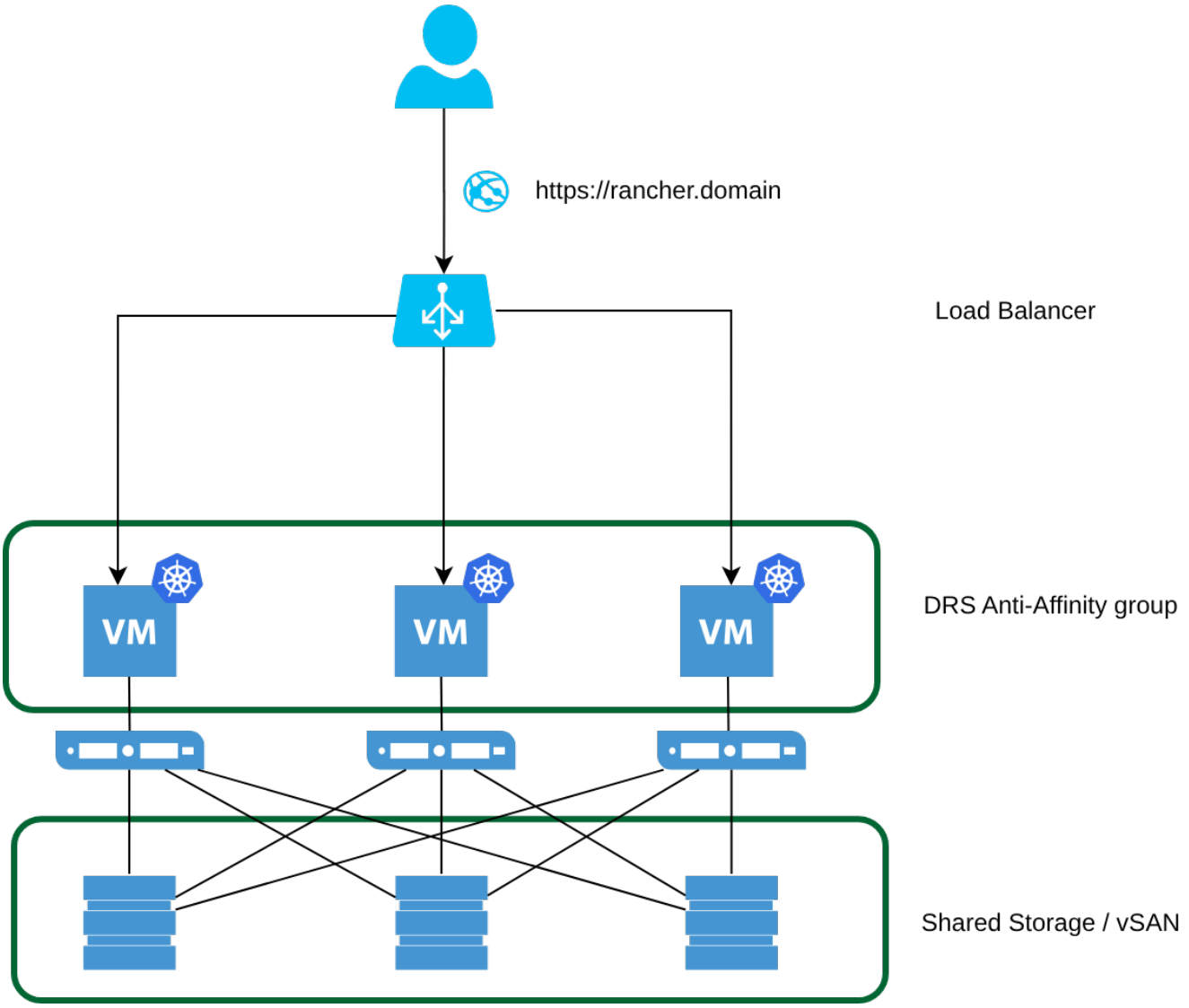



*Figure 7. Solution Overview*

### 1. Load Balancer Considerations

A load balancer is required to direct traffic to the Rancher workloads residing on the RKE nodes.

#### Leverage Fault Tolerance and High Availability

Leverage the use of an external (hardware or software) load balancer that has inherit high-availability functionality (F5, NSX-T, Keepalived, etc).

#### Back Up Load Balancer Configuration

In the event of a Disaster Recovery activity, availability of the Load balancer configuration will expedite the recovery process.

#### Configure Health Checks

Configure the Load balancer to automatically mark nodes as unavailable if a health check is failed. For example, NGINX can facilitate this with:

```
max_fails=3 fail_timeout=5s
```

#### Leverage an External Load Balancer

Avoid implementing a software load balancer within the management cluster.

#### Secure Access to Rancher

Configure appropriate Firewall / ACL rules to only expose access to Rancher

### 2. VM Considerations

#### Size the VM's According to Rancher Documentation

See Installation Requirements.

#### Leverage VM Templates to Construct the Environment

To facilitate the consistency of Virtual Machines deployed across the environment, consider the use of "Golden Images" in the form of VM templates. Packer can be used to accomplish this, adding greater customization options.

#### Leverage DRS Anti-Affinity Rules (Where Possible) to Separate Rancher Cluster Nodes Across ESXi Hosts

Doing so will ensure node VM's are spread across multiple ESXi hosts - preventing a single point of failure at the host level.

#### Leverage DRS Anti-Affinity Rules (Where Possible) to Separate Rancher Cluster Nodes Across Datastores

Doing so will ensure node VM's are spread across multiple datastores - preventing a single point of failure at the datastore level.

#### Configure VM's as Appropriate for Kubernetes

It's important to follow K8s and etcd best practices when deploying your nodes, including disabling swap, double-checking you have full network connectivity between all machines in the cluster, using unique hostnames, MAC addresses, and product_uuids for every node.

### 3. Network Considerations

#### Leverage Low Latency, High Bandwidth Connectivity Between ETCD Nodes

Deploy etcd members within a single data center where possible to avoid latency overheads and reduce the likelihood of network partitioning. For most setups, 1Gb connections will suffice. For large clusters, 10Gb connections can reduce the time taken to restore from backup.

#### Consistent IP Addressing for VM's

Each node used should have a static IP configured. In the case of DHCP, each node should have a DHCP reservation to make sure the node gets the same IP allocated.

### 4. Storage Considerations

#### Leverage SSD Drives for ETCD Nodes

ETCD is very sensitive to write latency. Therefore, leverage SSD disks where possible.

#### 5. Backups and Disaster Recovery

##### Perform Regular Management Cluster Backups

Rancher stores its data in the ETCD datastore of the Kubernetes cluster it resides on. Like with any Kubernetes cluster, perform frequent, tested backups of this cluster.

##### Back up Rancher Cluster Node VMs

Incorporate the Rancher management node VM's within a standard VM backup policy.

### Other Resources

The purpose of this section is to consolidate best practices for Rancher implementations. This also includes recommendations for related technologies, such as Kubernetes, Docker, containers, and more. The objective is to improve the outcome of a Rancher implementation using the operational experience of Rancher and its customers.

If you have any questions about how these might apply to your use case, please contact your Customer Success Manager or Support.

Use the navigation bar on the left to find the current best practices for managing and deploying the Rancher Server.

For more guidance on best practices, you can consult these resources:

- Security
- Rancher Blog [https://www.suse.com/c/rancherblog/]
- Rancher Forum [https://forums.rancher.com/]
- Rancher Users Slack [https://slack.rancher.io/]
- Rancher Labs YouTube Channel - Online Meetups, Demos, Training, and Webinars [https://www.youtube.com/channel/UCh5Xtp82q8wjijP8npkVTBA/featured]

# Installation References

Please see the following reference guides for other installation resources: Rancher Helm chart options, TLS settings, and feature flags.

### SUSE Rancher Prime Helm Chart Options

This page is a configuration reference for the Rancher Helm chart.

For information on enabling experimental features, refer to Enabling Experimental Features.

#### Common Options

| Option | Default Value | Description |
| --- | --- | --- |
| `bootstrapPassword` | " " | `string` - Set the bootstrap password for the first admin user. After logging in, the admin should reset their password. A randomly generated bootstrap password is used if this value is not set. |
| `hostname` | " " | `string` - the Fully Qualified Domain Name for your Rancher Server |
| `ingress.tls.source` | "rancher" | `string` - Where to get the cert for the ingress. - "rancher, letsEncrypt, secret" |
| `letsEncrypt.email` | " " | `string` - Your email address |
| `letsEncrypt.environment` | "production" | `string` - Valid options: "staging, production" |
| `privateCA` | false | `bool` - Set to true if your cert is signed by a private CA |

### Advanced Options

| Option | Default Value | Description |
| --- | --- | --- |
| `additionalTrustedCAs` | false | `bool` - See Additional Trusted CAs |
| `addLocal` | "true" | `string` - Have Rancher detect and import the "local" (upstream) Rancher server cluster. *Note: This option is no longer available in v2.5.0. Consider using the `restrictedAdmin` option to prevent users from modifying the local cluster.* |
| `agentTLSMode` | "" | `string` - either `system-store` or `strict`. See Agent TLS Enforcement |
| `antiAffinity` | "preferred" | `string` - AntiAffinity rule for Rancher pods - "preferred, required" |
| `auditLog.destination` | "sidecar" | `string` - Stream to sidecar container console or hostPath volume - "sidecar, hostPath" |
| `auditLog.hostPath` | "/var/log/rancher/audit" | `string` - log file destination on host (only applies when `auditLog.destination` is set to `hostPath`) |
| `auditLog.enabled` | false | `bool` - Enables / disables audit logging. |
| `auditLog.level` | 0 | `int` - set the API Audit Log level. [0-3] |
| `auditLog.maxAge` | 1 | `int` - maximum number of days to retain old audit log files (only applies when `auditLog.destination` is set to `hostPath`) |
| `auditLog.maxBackup` | 1 | `int` - maximum number of audit log files to retain (only applies when `auditLog.destination` is set to `hostPath`) |

| Option | Default Value | Description |
|---|---|---|
| auditLog.maxSize | 100 | int - maximum size in megabytes of the audit log file before it gets rotated (only applies when auditLog.destination is set to hostPath) |
| auditLog.image.repository | "registry.suse.com/bci/bci-micro" | string - Location for the image used to collect audit logs. |
| auditLog.image.tag | "15.4.14.3" | string - Tag for the image used to collect audit logs. |
| auditLog.image.pullPolicy | "IfNotPresent" | string - Override imagePullPolicy for auditLog images - "Always", "Never", "IfNotPresent". |
| busyboxImage | "" | string - Image location for busybox image used to collect audit logs. *Note: This option is deprecated use auditLog.image.repository to control auditing sidecar image.* |
| certmanager.version | "" | string - set cert-manager compatibility |
| debug | false | bool - set debug flag on rancher server |
| extraEnv | [] | list - set additional environment variables for Rancher |
| imagePullSecrets | [] | list - list of names of Secret resource containing private registry credentials |
| ingress.configurationSnippet | "" | string - additional Nginx configuration. Can be used for proxy configuration. |
| ingress.extraAnnotations | {} | map - additional annotations to customize the ingress |
| ingress.enabled | true | string - When set to false, Helm will not install a Rancher ingress. Set the option to false to deploy your own ingress. |
| letsEncrypt.ingress.class | "" | string - optional ingress class for the cert-manager acmesolver ingress that responds to the Let's Encrypt ACME challenges. Options: traefik, nginx. |
| noProxy | "127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16,.svc,.cluster.local,cattle-system.svc" | string - comma separated list of hostnames or ip address not to use the proxy |
| proxy | "" | string - HTTP[S] proxy server for Rancher |
| image.registry | "" | string - Override rancher image-specific registry, e.g, http://registry.example.com/ |
| image.repository | "rancher/rancher" | string - rancher image source |
| image.pullPolicy | "IfNotPresent" | string - Override imagePullPolicy for rancher server images - "Always", "Never", "IfNotPresent" |

| Option | Default Value | Description |
| --- | --- | --- |
| `image.tag` | "" | `string` - Defaults to .Chart.appVersion for rancher/rancher image tag |
| `replicas` | 3 | `int` - Number of Rancher server replicas. Setting to `-1` will dynamically choose 1, 2, or 3 based on the number of available nodes in the cluster. |
| `resources` | {} | `map` - rancher pod resource requests & limits |
| `systemDefaultRegistry` | "" | `string` - private registry to be used for all system container images, e.g., `https://registry.example.com/` |
| `tls` | "ingress" | `string` - See External TLS Termination for details. - "ingress, external" |
| `useBundledSystemChart` | `false` | `bool` - select to use the system-charts packaged with Rancher server. This option is used for air gapped installations. |
| `systemRegistryInheritsPullSecrets` | `false` | `bool` - Sets the default value of the `system-default-registry-pull-secrets` global setting to a comma-separated list of the `imagePullSecrets` used when installing Rancher. For details, see configuring a global default private registry. |

#### Bootstrap Password

You can set a specific bootstrap password during Rancher installation. If you don't set a specific bootstrap password, Rancher randomly generates a password for the first admin account.

When you log in for the first time, use the bootstrap password you set to log in. If you did not set a bootstrap password, the Rancher UI shows commands that can be used to retrieve the bootstrap password. Run those commands and log in to the account. After you log in for the first time, you are asked to reset the admin password.

#### API Audit Log

Enabling the API Audit Log.

You can collect this log as you would any container log. Enable logging for the `System` Project on the Rancher server cluster.

```
--set auditLog.enabled=true --set auditLog.level=1
```

By default enabling Audit Logging will create a sidecar container in the Rancher pod. This container (`rancher-audit-log`) will stream the log to `stdout`. You can collect this log as you would any container log. When using the sidecar as the audit log destination, the `hostPath`, `maxAge`, `maxBackups`, and `maxSize` options do not apply. It's advised to use your OS or Docker daemon's log rotation features to control disk space use. Enable logging for the Rancher server cluster or System Project.

Set the `auditLog.destination` to `hostPath` to forward logs to volume shared with the host system instead of streaming to a sidecar container. When setting the destination to `hostPath` you may want to adjust the other auditLog parameters for log rotation.

#### Setting Extra Environment Variables

You can set extra environment variables for Rancher server using `extraEnv`. This list is passed to the Rancher deployment in its YAML format. It is embedded under `env` for the Rancher container. Refer to the Kubernetes documentation for setting container environment variables, `extraEnv` can use any of the keys referenced in Define Environment Variables for a Container [https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/#define-an-environment-variable-for-a-container].

Consider an example that uses the `name` and `value` keys:

```
--set 'extraEnv[0].name=CATTLE_TLS_MIN_VERSION'
--set 'extraEnv[0].value=1.0'
```

If passing sensitive data as the value for an environment variable, such as proxy authentication credentials, it is strongly recommended that a secret reference is used. This will prevent sensitive data from being exposed in Helm or the Rancher deployment.

Consider an example that uses the `name`, `valueFrom.secretKeyRef.name`, and `valueFrom.secretKeyRef.key` keys. See example in HTTP Proxy

#### TLS Settings

When you install Rancher inside of a Kubernetes cluster, TLS is offloaded at the cluster's ingress controller. The possible TLS settings depend on the used ingress controller.

See TLS settings for more information and options.

#### Import `local` Cluster

By default Rancher server will detect and import the `local` cluster it's running on. User with access to the `local` cluster will essentially have "root" access to all the clusters managed by Rancher server.



If you turn addLocal off, most Rancher v2.5 features won't work, including the EKS provisioner.

If this is a concern in your environment you can set this option to "false" on your initial install.

This option is only effective on the initial Rancher install. See Issue `16522` [https://github.com/rancher/rancher/issues/16522] for more information.

```
--set addLocal="false"
```

#### Customizing your Ingress

To customize or use a different ingress with Rancher server you can set your own Ingress annotations.

Please refer to the Traefik documentation for the full list of Ingress NGINX annotations that are supported [https://doc.traefik.io/traefik/reference/routing-configuration/kubernetes/ingress-nginx/#annotations-support] and unsupported [https://doc.traefik.io/traefik/reference/routing-configuration/kubernetes/ingress-nginx/#unsupported-annotations] by Traefik's kubernetesIngressNginx provider.

Example on setting a custom certificate issuer:

```
--set ingress.extraAnnotations.'cert-manager\.io/cluster-issuer'=issuer-name
```

#### HTTP Proxy

Rancher requires internet access for some functionality (Helm charts). Use `proxy` to set your proxy server or use `extraEnv` to set the `HTTPS_PROXY` environment variable to point to your proxy server.

Add your IP exceptions to the `noProxy` chart value as a comma separated list. Make sure you add the following values:

- Pod cluster IP range (default: `10.42.0.0/16`).
- Service cluster IP range (default: `10.43.0.0/16`).
- Internal cluster domains (default: `.svc,.cluster.local`).
- Any worker cluster `controlplane` nodes. Rancher supports CIDR notation ranges in this list.

When not including sensitive data, the `proxy` or `extraEnv` chart options can be used. When using `extraEnv` the `noProxy` Helm option is ignored. Therefore, the `NO_PROXY` environment variable must also be set with `extraEnv`.

The following is an example of setting proxy using the `proxy` chart option:

```
--set proxy="http://<proxy_url:proxy_port>/"
```

Example of setting proxy using the `extraEnv` chart option:

```
--set extraEnv[1].name=HTTPS_PROXY
--set extraEnv[1].value="http://<proxy_url>:<proxy_port>/"
--set extraEnv[2].name=NO_PROXY
--set
extraEnv[2].value="127.0.0.0/8\,10.0.0.0/8\,172.16.0.0/12\,192.168.0.0/16\,.svc\,.cluster.local"
```

When including sensitive data, such as proxy authentication credentials, use the `extraEnv` option with `valueFrom.secretRef` to prevent sensitive data from being exposed in Helm or the Rancher deployment.

The following is an example of using `extraEnv` to configure proxy. This example secret would contain the value `"http://<username>:<password>@<proxy_url>:<proxy_port>/"` in the secret's `"https-proxy-url"` key:

```
--set extraEnv[1].name=HTTPS_PROXY
--set extraEnv[1].valueFrom.secretKeyRef.name=secret-name
--set extraEnv[1].valueFrom.secretKeyRef.key=https-proxy-url
--set extraEnv[2].name=NO_PROXY
--set
extraEnv[2].value="127.0.0.0/8\,10.0.0.0/8\,172.16.0.0/12\,192.168.0.0/1
6\,.svc\,.cluster.local"
```

To learn more about how to configure environment variables, refer to Define Environment Variables for a Container [https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/#define-an-environment-variable-for-a-container].

#### Additional Trusted CAs

If you have private registries, catalogs or a proxy that intercepts certificates, you may need to add more trusted CAs to Rancher.

```
--set additionalTrustedCAs=true
```

Once the Rancher deployment is created, copy your CA certs in pem format into a file named `ca-additional.pem` and use `kubectl` to create the `tls-ca-additional` secret in the `cattle-system` namespace.

```
kubectl -n cattle-system create secret generic tls-ca-additional --from
-file=ca-additional.pem=./ca-additional.pem
```

#### Private Registry and Air Gap Installs

For details on installing Rancher with a private registry, see the air gap installation docs.

### External TLS Termination

We recommend configuring your load balancer as a Layer 4 balancer, forwarding plain 80/tcp and 443/tcp to the Rancher Management cluster nodes. The Ingress Controller on the cluster will redirect http traffic on port 80 to https on port 443.

You may terminate the SSL/TLS on a L7 load balancer external to the Rancher cluster (ingress). Use the `--set tls=external` option and point your load balancer at port http 80 on all of the Rancher cluster nodes. This will expose the Rancher interface on http port 80. Be aware that clients that are allowed to connect directly to the Rancher cluster will not be encrypted. If you choose to do this we recommend that you restrict direct access at the network level to just your load balancer.



If you are using a Private CA signed certificate (or if `agent-tls-mode` is set to `strict`), add `--set privateCA=true` and see Adding TLS Secrets - Using a Private CA Signed Certificate to add the CA cert for Rancher.

Your load balancer must support long lived websocket connections and will need to insert proxy headers so

Rancher can route links correctly.

#### Required Headers

- `Host`
- `X-Forwarded-Proto`
- `X-Forwarded-Port`
- `X-Forwarded-For`

#### Recommended Timeouts

- Read Timeout: `1800 seconds`
- Write Timeout: `1800 seconds`
- Connect Timeout: `30 seconds`

#### Health Checks

Rancher will respond `200` to health checks on the `/healthz` endpoint.

## TLS Settings

Changing the default TLS settings depends on the chosen installation method.

### Running Rancher in a highly available Kubernetes cluster

When you install a Rancher managed Kubernetes cluster, TLS is offloaded at the cluster's ingress controller. Traefik is the default ingress for K3s and can be used with RKE2, refer to TLS Options [https://doc.traefik.io/traefik/https/tls/#tls-options] for further information.

### Running Rancher in a single Docker container

The default TLS configuration only accepts TLS 1.2 and secure TLS cipher suites. You can change this by setting the following environment variables:

| Parameter | Description | Default | Available options |
|---|---|---|---|
| `CATTLE_TLS_MIN_VERSION` | Minimum TLS version | `1.2` | `1.0`, `1.1`, `1.2`, `1.3` |
| `CATTLE_TLS_CIPHERS` | Allowed TLS cipher suites | `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`, `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`, `TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305`, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`, `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`, `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305` | See Golang tls constants [https://golang.org/pkg/crypto/tls/#pkg-constants] |

### Agent TLS Enforcement

The `agent-tls-mode` setting controls how Rancher's agents (`cluster-agent`, `fleet-agent`, and `system-agent`) validate Rancher's certificate.

When the value is set to `strict`, Rancher's agents only trust certificates generated by the Certificate Authority contained in the `cacerts` setting. When the value is set to `system-store`, Rancher's agents trust any certificate generated by a public Certificate Authority contained in the operating system's trust store including those signed by authorities such as Let's Encrypt. This can be a security risk, since any certificate generated by these external authorities, which are outside the user's control, are considered valid in this state.

While the `strict` option enables a higher level of security, it requires Rancher to have access to the CA which generated the certificate visible to the agents. In the case of certain certificate configurations (notably, external certificates), this is not automatic, and extra configuration is needed. See the installation guide for more information on which scenarios require extra configuration.

In Rancher v2.9.0 and later, this setting defaults to `strict` on new installs. For users installing or upgrading from a prior Rancher version, it is set to `system-store`.

#### Preparing for the Setting Change

Each cluster contains a condition in the status field called `AgentTlsStrictCheck`. If `AgentTlsStrictCheck` is set to `"True"`, this indicates that the agents for the cluster are ready to operate in `strict` mode. You can manually inspect each cluster to see if they are ready using the Rancher UI or a kubectl command such as the following:

```
## the below command skips outputs $CLUSTER_NAME,$STATUS for all non-
local clusters
kubectl get cluster.management.cattle.io -o jsonpath='{range
.items[?(@.metadata.name!="local")]}{.metadata.name},{.status.conditions
[?(@.type=="AgentTlsStrictCheck")].status}{"\n"}{end}'
```

#### Changing the Setting

You can change the setting using the Rancher UI or the `agentTLSMode` helm chart option.



If you specify the value through the Helm chart, you may only modify the value with Helm.



Depending on your cert setup, additional action may be required, such as uploading the Certificate Authority which signed your certs. Review the installation guide before changing the setting to see if any additional requirements apply to your setup.

To change the setting's value through the UI, navigate to the **Global Settings** page, and find the `agent-tls-mode` setting near the bottom of the page. When you change the setting through the UI, Rancher first checks that all downstream clusters have the condition `AgentTlsStrictCheck` set to `"True"` before allowing the request. This prevents outages from a certificate mismatch.

#### Overriding the Setting Validation Checks

In some cases, you may want to override the check ensuring all agents can accept the new TLS configuration:



Rancher checks the status of all downstream clusters to prevent outages. Overriding this check is not recommended, and should be done with great caution.

1. As an admin, generate a kubeconfig for the local cluster. In the below examples, this was saved to the `local_kubeconfig.yaml` file.
2. Retrieve the current setting and save it to `setting.yaml`:

```
kubectl get setting agent-tls-mode -o yaml --kubeconfig
=local_kubeconfig.yaml > setting.yaml
```

3. Update the `setting.yaml` file, replacing `value` with `strict`. Adding the `cattle.io/force: "true"` annotation overrides the cluster condition check, and should only be done with great care:



Including the `cattle.io/force` annotation with any value (including, for example `"false"`) overrides the cluster condition check.

```
apiVersion: management.cattle.io/v3
customized: false
default: strict
kind: Setting
metadata:
  name: agent-tls-mode
  annotations:
    cattle.io/force: "true"
source: ""
value: strict
```

1. Apply the new version of the setting:

```
kubectl apply -f setting.yaml --kubeconfig=local_kubeconfig.yaml
```

### Feature Flags

With feature flags, you can try out optional or experimental features, and enable legacy features that are being phased out.

To learn more about feature values and how to enable them, see Enabling Experimental Features.



Some feature flags require a restart of the Rancher container. Features that require a restart are marked in the Rancher UI.

The following is a list of feature flags available in Rancher. If you've upgraded from a previous Rancher version, you may see additional flags in the Rancher UI, such as `proxy` or `dashboard` (both discontinued):

- `aggregated-roletemplates`: Use cluster role aggregation architecture for RoleTemplates, ProjectRoleTemplateBindings, and ClusterRoleTemplateBindings. See RoleTemplate Aggregation for more information.
- `clean-stale-secrets`: Removes stale secrets from the `cattle-impersonation-system` namespace. This slowly cleans up old secrets which are no longer being used by the impersonation system.
- `cluster-shell`: Enables the Rancher cluster shell feature, which allows users to access a kubectl shell directly from the Rancher UI to manage cluster resources.
- `continuous-delivery`: Allows Fleet GitOps to be disabled separately from Fleet. See Continuous Delivery. for more information.
- `crt-token-ttl-rotation`: This feature flag enables automatic CRT token expiration and rotation, configurable via `CRTDefaultTTL`/`CRTDefaultGracePeriod` settings or per-CRT overrides. On expiry, a new token is generated; both old and new tokens remain valid during the grace period giving agents time to pick up the new credential. Rotation is paired with new detection logic in `clusterDeploy` that recognizes when a cluster's token has changed and automatically redeploys the cluster agent with the new credential. Token rotation alone would otherwise never reach a running agent. The feature flag is opt-in for 2.13 and 2.14, and enabled by default starting in 2.15.
- `fleet`: The Rancher provisioning framework in v2.6 and later requires Fleet. The flag will be automatically enabled when you upgrade, even if you disabled this flag in an earlier version of Rancher. See Continuous Delivery with Fleet for more information.
- `harvester`: Manages access to the Virtualization Management page, where users can navigate directly to Harvester clusters and access the Harvester UI. See SUSE Virtualization Integration Overview for more information.
- `harvester-baremetal-container-workload`: This feature flag is disabled by default. When enabled the Rancher UI stops filtering out Harvester clusters, allowing Fleet bundles and applications to target and deploy container workloads directly to them. After enabling the `harvester-baremetal-container-workload` feature flag, please validate existing Fleet Bundles to ensure they now select, or don't select, the required clusters. When disabled, the UI maintains the default behavior and excludes Harvester clusters from Fleet targets.
- `hide-local-auth-provider`: When enabled, hides the local auth provider from users if one or more external auth providers are configured and active. This prevents login through the UI, API, and CLI. It does not revoke API keys that were already issued for `local-auth-provider` users. Creation or editing of `local-auth-provider users` in the UI is also disabled, administrators and users with the `manage-users` permission will continue to be able to create or edit users via API or CLI. This flag is disabled by default.



If the external auth provider becomes unavailable whilst this flag is active, it will not be possible to log in via UI, API or CLI. So when `hide-local-auth-provider` is enabled, make sure an administrator has a valid, non-expiring-soon API key or `kubeconfig` token for the local cluster. These can be used to recover the system by using either the API or `kubectl` to disable the `hide-local-auth-provider` flag.

- `imperative-api-extension`: Enables Rancher's extension API server [https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/] to register new APIs to Kubernetes. This flag is enabled by default. See the Extension API Server page for more information.

- `imported-day-2-ops`: Enables day 2 operations for imported RKE2 and K3s clusters. This flag is active by default and requires a Rancher restart when its value changes. See Day 2 Operations for Imported RKE2 and K3s Clusters for more information.
- `istio-virtual-service-ui`: Enables a visual interface to create, read, update, and delete Istio virtual services and destination rules, which are Istio traffic management features.
- `legacy`: Enables a set of features from 2.5.x and earlier, that are slowly being phased out in favor of newer implementations. These are a mix of deprecated features as well as features that will eventually be available to newer versions. This flag is disabled by default on new Rancher installations. If you're upgrading from a previous version of Rancher, this flag is enabled.
- `managed-system-upgrade-controller`: Enables the installation of the system-upgrade-controller app in downstream imported RKE2/K3s clusters, as well as in the local cluster if it is an RKE2/K3s cluster.



This `managed-system-upgrade-controller` flag is intended for **internal use only** and does not have an associated Feature CR. Use with caution.

To control whether Rancher should manage the Kubernetes version of imported RKE2/K3s clusters, it is recommended to use the imported-cluster-version-management feature that is available in Rancher v2.11.0 or newer.



If the `managed-system-upgrade-controller` flag was **disabled** in Rancher v2.10.x, and any imported RKE2/K3s clusters were upgraded **outside of Rancher**, follow the steps below to prevent the unexpected installation of the system-upgrade-controller app and to ensure the imported-cluster-version-management feature works correctly:

1. Upgrade Rancher to v2.11.0 or newer, making sure to **retain** the `managed-system-upgrade-controller=false` feature flag in Helm values if it was set during the v2.10.x installation.
2. After Rancher is fully up and running, disable the `imported-cluster-version-management` setting. You can do this either through the Rancher UI by clicking ☰ > **Global Settings > Settings > imported-cluster-version-management**, or by editing the corresponding `Setting.management.cattle.io/v3` custom resource via kubectl.
3. Perform a second Helm upgrade, this time omitting the `managed-system-upgrade-controller=false` feature flag.

Now, the imported cluster version management is disabled by default, and Rancher no longer installs the system-upgrade-controller app on imported clusters automatically.

You can enable this feature on a per-cluster basis. For more information, please refer to the documentation.

- `multi-cluster-management`: Allows multi-cluster provisioning and management of Kubernetes clusters. This flag can only be set at install time. It can't be enabled or disabled later.
- `node-shell`: Enables the Rancher node shell feature, which allows users to SSH into provisioned cluster nodes directly from the Rancher UI.
- `pod-shell`: Enables the Rancher pod shell feature, which allows users to execute shell commands inside pods directly from the Rancher UI.

- `rke2`: Enables provisioning RKE2 clusters. This flag is enabled by default.
- `token-hashing`: Enables token hashing. Once enabled, existing tokens will be hashed and all new tokens will be hashed automatically with the SHA256 algorithm. Once a token is hashed it can't be undone. This flag can't be disabled after its enabled. See API Tokens for more information.
- `uiextension`: Enables UI extensions. This flag is enabled by default. Enabling or disabling the flag forces the Rancher pod to restart. The first time this flag is set to `Active`, it creates a CRD and enables the controllers and endpoints necessary for the feature to work. If set to `Disabled`, it disables the previously mentioned controllers and endpoints. Setting `uiextension` to `Disabled` has no effect on the CRD — it does not create a CRD if it does not yet exist, nor does it delete the CRD if it already exists.* `unsupported-storage-drivers`: Enables types for storage providers and provisioners that aren't enabled by default. See Allow Unsupported Storage Drivers for more information.
- `ui-sql-cache`: Enables an SQLite-based cache for UI tables and Server-Side Pagination. This feature flag can no longer be disabled through the UI. See UI Server-Side Pagination for more information.

The following table shows the availability and default values for some feature flags in Rancher. Features marked "GA" are generally available:

| Feature Flag Name | Default Value | Status | Available As Of | Additional Information |
|---|---|---|---|---|
| `aggregated-roletemplates` | `Disabled` | Experimental | v2.11.0 | This flag value is locked on install and can't be changed. |
| `clean-stale-secrets` | `Active` | GA | v2.11.0 | |
| `cluster-shell` | `Active` | GA | v2.15.0 | |
| `continuous-delivery` | `Active` | GA | v2.6.0 | |
| `crt-token-ttl-rotation` | `Enabled` | GA | v2.15.0 | |
| `external-rules` | v2.7.14: `Disabled`, v2.8.5: `Active` | Removed | v2.7.14, v2.8.5 | This flag affected external `RoleTemplate` behavior. It is removed in Rancher v2.9.0 and later as the behavior is enabled by default. |
| `fleet` | `Active` | Can no longer be disabled | v2.6.0 | |
| `fleet` | `Active` | GA | v2.5.0 | |
| `harvester` | `Active` | Experimental | v2.6.1 | |
| `harvester-baremetal-container-workload` | `Disabled` | GA | v2.15.0 | |
| `hide-local-auth-provider` | `Disabled` | GA | v2.15.0 | See the warning above about the risk of being locked out of the UI, API, and CLI. |
| `imperative-api-extension` | `Active` | GA | v2.11.0 | |

| Feature Flag Name | Default Value | Status | Available As Of | Additional Information |
|---|---|---|---|---|
| `imported-day-2-ops` | `Active` | GA | v2.15.0 | Enables day 2 operations for imported RKE2 and K3s clusters. Changing this flag requires a Rancher restart. See Day 2 Operations for Imported RKE2 and K3s Clusters for more information. |
| `legacy` | `Disabled` for new installs, `Active` for upgrades | GA | v2.6.0 | |
| `managed-system-upgrade-controller` | `Active` | GA | v2.10.0 | |
| `node-shell` | `Active` | GA | v2.15.0 | |
| `pod-shell` | `Active` | GA | v2.15.0 | |
| `rke2` | `Active` | Experimental | v2.6.0 | |
| `token-hashing` | `Disabled` for new installs, `Active` for upgrades | GA | v2.6.0 | |
| `uiextension` | `Active` | GA | v2.9.0 | |
| `ui-sql-cache` | `Active` | GA. This feature flag can no longer be disabled through the Rancher UI. | v2.9.0 | |

# Resources

## Air-Gapped Installations

Follow these steps to install the Rancher server in an air gapped environment.

An air gapped environment could be where Rancher server will be installed offline, behind a firewall, or behind a proxy.

## Advanced Options

When installing Rancher, there are several advanced options that can be enabled during installation. Within each install guide, these options are presented. Learn more about these options:

- Custom CA Certificate
- API Audit Log
- TLS Settings
- etcd configuration
- | Local System Charts for Air Gap Installations | v2.3.0 |

### Setting up the Bootstrap Password

When you install Rancher, you can set a bootstrap password for the first admin account.

If you choose not to set a bootstrap password, Rancher randomly generates a bootstrap password for the first admin account.

For details on how to set the bootstrap password, see below.

#### Password Requirements

The bootstrap password can be any length.

When you reset the first admin account's password after first login, the new password must be at least 12 characters long.

You can customize the minimum password length for user accounts, within limitations.

Minimum password length can be any positive integer value between 2 and 256. Decimal values and leading zeroes are not allowed.

#### Specifying the Bootstrap Password

During Rancher installation, set `bootstrapPassword` alongside any other flags for the Rancher Helm chart. For example:

```
helm install rancher rancher-<chart-repo>/rancher \
  --set bootstrapPassword=<password>
```

# Retrieving the Bootstrap Password

The bootstrap password is stored in the Docker container logs. After Rancher is installed, the UI shows instructions for how to retrieve the password based on your installation method.

```
kubectl get secret --namespace cattle-system bootstrap-secret -o go-
template='{{ .data.bootstrapPassword|base64decode}}{{ "\n" }}'
```

### Adding TLS Secrets

Kubernetes will create all the objects and services for Rancher, but it will not become available until we populate the `tls-rancher-ingress` secret in the `cattle-system` namespace with the certificate and key.

Combine the server certificate followed by any intermediate certificate(s) needed into a file named `tls.crt`. Copy your certificate key into a file named `tls.key`.

For example, acme.sh [https://acme.sh] provides server certificate and CA chains in `fullchain.cer` file. This `fullchain.cer` should be renamed to `tls.crt` & certificate key file as `tls.key`.

Use `kubectl` with the `tls` secret type to create the secrets.

```
kubectl -n cattle-system create secret tls tls-rancher-ingress \
  --cert=tls.crt \
  --key=tls.key
```



If you want to replace the certificate, you can delete the `tls-rancher-ingress` secret using `kubectl -n cattle-system delete secret tls-rancher-ingress` and add a new one using the command shown above. If you are using a private CA signed certificate, replacing the certificate is only possible if the new certificate is signed by the same CA as the certificate currently in use.

#### Using a Private CA Signed Certificate

If you are using a private CA, Rancher requires a copy of the private CA's root certificate or certificate chain, which the Rancher Agent uses to validate the connection to the server.

Create a file named `cacerts.pem` that only contains the root CA certificate or certificate chain from your private CA, and use `kubectl` to create the `tls-ca` secret in the `cattle-system` namespace.

```
kubectl -n cattle-system create secret generic tls-ca \
  --from-file=cacerts.pem
```



The configured `tls-ca` secret is retrieved when Rancher starts. On a running Rancher installation the updated CA will take effect after new Rancher pods are started.

The certificate chain must be properly formatted, or components may fail to download resources from the Rancher server.

#### Adding Additional CA Certificates

If you are using a node driver that makes API requests with a different CA than the one configured for Rancher, you can add additional root certificates and certificate chains.

Create a unique file ending in `.pem` for each certificate that is required, and use kubectl to create the `tls-additional` secret in the `cattle-system` namespace.

```
kubectl -n cattle-system create secret generic tls-additional \
  --from-file=cacertsl.pem=cacertsl.pem --from-file=cacerts2.pem
=cacerts2.pem
```

Rancher mounts these CA root certificates and certificate chains into the node driver pod during provisioning.

#### Updating a Private CA Certificate

Follow the steps on this page to update the SSL certificate of the ingress in a Rancher high availability Kubernetes installation or to switch from the default self-signed certificate to a custom certificate.

## About Custom CA Root Certificates

If you're using Rancher in an internal production environment where you aren't exposing apps publicly, use a certificate from a private certificate authority (CA).

Services that Rancher needs to access are sometimes configured with a certificate from a custom/internal CA root, also known as self signed certificate. If the presented certificate from the service cannot be validated by Rancher, the following error displays: `x509: certificate signed by unknown authority`.

To validate the certificate, the CA root certificates need to be added to Rancher. As Rancher is written in Go, we can use the environment variable `SSL_CERT_DIR` to point to the directory where the CA root certificates are located in the container. The CA root certificates directory can be mounted using the Docker volume option (`-v host-source-directory:container-destination-directory`) when starting the Rancher container.

Examples of services that Rancher can access:

- Catalogs
- Authentication providers
- Accessing hosting/cloud API when using Node Drivers

#### Installing with the custom CA Certificate

For details on starting a Rancher container with your private CA certificates mounted, refer to the installation docs:

- Kubernetes install options for Additional Trusted CAs

## Updating the SUSE Rancher Prime Certificate

#### Updating a Private CA Certificate

Follow these steps to rotate an SSL certificate and private CA used by Rancher installed on a Kubernetes cluster, or migrate to an SSL certificate signed by a private CA.

A summary of the steps is as follows:

1. Create or update the `tls-rancher-ingress` Kubernetes secret object with the new certificate and private key.
2. Create or update the `tls-ca` Kubernetes secret object with the root CA certificate (only required when using a private CA).
3. Update the Rancher installation using the Helm CLI.
4. Reconfigure the Rancher agents to trust the new CA certificate.
5. Select Force Update of Fleet clusters to connect fleet-agent to Rancher.

The details of these instructions are below.

#### 1. Create/update the certificate secret object

First, concatenate the server certificate followed by any intermediate certificate(s) to a file named `tls.crt` and provide the corresponding certificate key in a file named `tls.key`.

Use the following command to create the `tls-rancher-ingress` secret object in the Rancher (local) management cluster:

```
kubectl -n cattle-system create secret tls tls-rancher-ingress \
  --cert=tls.crt \
  --key=tls.key
```

Alternatively, to update an existing `tls-rancher-ingress` secret:

```
kubectl -n cattle-system create secret tls tls-rancher-ingress \
  --cert=tls.crt \
  --key=tls.key \
  --dry-run --save-config -o yaml | kubectl apply -f -
```

#### 2. Create/update the CA certificate secret object

If the new certificate was signed by a private CA, you will need to copy the corresponding root CA certificate into a file named `cacerts.pem` and create or update the `tls-ca` secret in the `cattle-system` namespace. If the certificate was signed by an intermediate CA, then the `cacerts.pem` must contain both the intermediate and root CA certificates (in this order).

To create the initial `tls-ca` secret:

```
kubectl -n cattle-system create secret generic tls-ca \
  --from-file=cacerts.pem
```

To update an existing `tls-ca` secret:

```
kubectl -n cattle-system create secret generic tls-ca \
  --from-file=cacerts.pem \
  --dry-run --save-config -o yaml | kubectl apply -f -
```

#### 3. Reconfigure the Rancher deployment

If the certificate source remains the same (for example, `secret`), please follow the steps in Step 3a.

However, if the certificate source is changing (for example, `letsEncrypt` to `secret`), follow the steps in 3b.

#### 3a. Redeploy the Rancher pods

This step is required when the certificate source remains the same, but the CA certificate is being updated.

In this scenario a redeploy of the Rancher pods is needed, this is because the `tls-ca` secret is read by the Rancher pods when starting.

The command below can be used to redeploy the Rancher pods:

```
kubectl rollout restart deploy/rancher -n cattle-system
```

When the change is completed, navigate to `https://<rancher_server_url>/v3/settings/cacerts` to verify that the value matches the CA certificate written in the `tls-ca` secret earlier. The CA `cacerts` value may not update until all of the redeployed Rancher pods start.

#### 3b. Update the Helm values for Rancher

This step is required if the certificate source is changing. If Rancher was previously configured to use the default self-signed certificate (`ingress.tls.source=rancher`) or Let's Encrypt (`ingress.tls.source=letsEncrypt`), and is now using a certificate signed by a private CA (`ingress.tls.source=secret`).

The below steps update the Helm values for the Rancher chart, so the Rancher pods and ingress are reconfigured to use the new private CA certificate created in Step 1 & 2.

1. Adjust the values that were used during initial installation, store the current values with:

   ```
   helm get values rancher -n cattle-system -o yaml > values.yaml
   ```

2. Retrieve the version string of the currently deployed Rancher chart to use below:

   ```
   helm ls -n cattle-system
   ```

3. Update the current Helm values in the `values.yaml` file to contain:

   ```
   ingress:
     tls:
       source: secret
   privateCA: true
   ```



```
As the certificate is signed by a private CA, it is
important to ensure {xref-helm-chart-
options}#_common_options[`privateCA: true`] is set in
the `values.yaml` file.
```

4. Upgrade the Helm application instance using the `values.yaml` file and the current chart version. The version must match to prevent an upgrade of Rancher.

```
helm upgrade rancher rancher-prime/rancher \
  --namespace cattle-system \
  -f values.yaml \
  --version <DEPLOYED_RANCHER_VERSION>
```

When the change is completed, navigate to `https://<rancher_server_url>/v3/settings/cacerts` to verify that the value matches the CA certificate written in the `tls-ca` secret earlier. The CA `cacerts` value may not update until all Rancher pods start.

#### 4. Reconfigure Rancher agents to trust the private CA

This section covers three methods to reconfigure Rancher agents to trust the private CA. This step is required if either of the following is true:

- Rancher was previously configured to use the Rancher self-signed certificate (`ingress.tls.source=rancher`) or with a Let's Encrypt issued certificate (`ingress.tls.source=letsEncrypt`)
- The certificate was signed by a different private CA

##### Why is this step required?

When Rancher is configured with a certificate signed by a private CA, the CA certificate chain is trusted by Rancher agent containers. Agents compare the checksum of the downloaded certificate against the `CATTLE_CA_CHECKSUM` environment variable. This means that, when the private CA certificate used by Rancher has changed, the environment variable `CATTLE_CA_CHECKSUM` must be updated accordingly.

##### Which method should I choose?

Method 1 is the easiest, but requires all clusters to be connected to Rancher after the certificates have been rotated. This is usually the case if the process is performed right after updating or redeploying the Rancher deployment (Step 3).

If the clusters have lost connection to Rancher but Authorized Cluster Endpoint (ACE) is enabled on all clusters, then go with method 2.

Method 3 can be used as a fallback if method 1 and 2 are not possible.

##### Method 1: Force a redeploy of the Rancher agents

For each downstream cluster run the following command using the Kubeconfig file of the Rancher (local) management cluster.

```
kubectl annotate clusters.management.cattle.io <CLUSTER_ID>
io.cattle.agent.force.deploy=true
```



Locate the cluster ID (c-xxxxx) for the downstream cluster, this can be seen in the browser URL bar when viewing the cluster in the Rancher UI, under Cluster Management.

This command will cause the agent manifest to be reapplied with the checksum of the new certificate.

#### Method 2: Manually update the checksum environment variable

Manually patch the agent Kubernetes objects by updating the `CATTLE_CA_CHECKSUM` environment variable to the value matching the checksum of the new CA certificate. Generate the new checksum value like so:

```
curl -k -s -fL <RANCHER_SERVER_URL>/v3/settings/cacerts | jq -r .value |
sha256sum | awk '{print $1}'
```

Using a Kubeconfig for each downstream cluster update the environment variable for the two agent deployments. If the ACE is enabled for the cluster, the kubectl context can be adjusted to connect directly to the downstream cluster.

```
kubectl edit -n cattle-system ds/cattle-node-agent
kubectl edit -n cattle-system deployment/cattle-cluster-agent
```

#### Method 3: Manually redeploy the Rancher agents

With this method the Rancher agents are reapplied by running a set of commands on a control plane node of each downstream cluster.

Repeat the below steps for each downstream cluster:

1. Retrieve the agent registration kubectl command:
    a. Locate the cluster ID (c-xxxxx) for the downstream cluster, this can be seen in the URL when viewing the cluster in the Rancher UI under Cluster Management
    b. Add the Rancher server URL and cluster ID to the following URL: `https://<rancher_server_url>/v3/clusterregistrationtokens?clusterId=<CLUSTER_ID>`
    c. Copy the command from the `insecureCommand` field, this command is used because a private CA is un use
2. Run the kubectl command from the previous step using a kubeconfig for the downstream cluster with one of the following methods:
    a. If the ACE is enabled for the cluster, the context can be adjusted to connect directly to the downstream cluster
    b. Alternatively, SSH into the control plane node:
        - RKE: Use the steps in the document here [https://github.com/rancherlabs/support-tools/tree/master/how-to-retrieve-kubeconfig-from-custom-cluster] to generate a kubeconfig
        - RKE2/K3s: Use the kubeconfig populated during installation

#### 5. Force Update SUSE® Rancher Prime: Continuous Delivery clusters to reconnect the fleet-agent to Rancher

Select 'Force Update' for the clusters within the Continuous Delivery view of the Rancher UI to allow the fleet-agent in downstream clusters to successfully connect to Rancher.

#### Why is this step required?

Fleet agents in Rancher managed clusters store a kubeconfig that is used to connect to Rancher. The kubeconfig contains a `certificate-authority-data` field containing the CA for the certificate used by Rancher. When changing the CA, this block needs to be updated to allow the fleet-agent to trust the certificate used by Rancher.

### Updating from a Private CA Certificate to a Public CA Certificate

Follow these steps to perform the opposite procedure as shown above, to change from a certificate issued by a private CA, to a public or self-signed CA.

#### 1. Create/update the certificate secret object

First, concatenate the server certificate followed by any intermediate certificate(s) to a file named `tls.crt` and provide the corresponding certificate key in a file named `tls.key`.

Use the following command to create the `tls-rancher-ingress` secret object in the Rancher (local) management cluster:

```
kubectl -n cattle-system create secret tls tls-rancher-ingress \
  --cert=tls.crt \
  --key=tls.key
```

Alternatively, to update an existing `tls-rancher-ingress` secret:

```
kubectl -n cattle-system create secret tls tls-rancher-ingress \
  --cert=tls.crt \
  --key=tls.key \
  --dry-run --save-config -o yaml | kubectl apply -f -
```

#### 2. Delete the CA certificate secret object

You will delete the `tls-ca` secret in the `cattle-system` namespace as it is no longer needed. You may also optionally save a copy of the `tls-ca` secret if desired.

To save the existing `tls-ca` secret:

```
kubectl -n cattle-system get secret tls-ca -o yaml > tls-ca.yaml
```

To delete the existing `tls-ca` secret:

```
kubectl -n cattle-system delete secret tls-ca
```

#### 3. Reconfigure the Rancher deployment

This step is required if the certificate source is changing. In this scenario it's likely only changing because Rancher was previously configured to use the default self-signed certificate (`ingress.tls.source=rancher`).

The below steps update the Helm values for the Rancher chart, so the Rancher pods and ingress are reconfigured to use the new certificate created in Step 1.

1. Adjust the values that were used during initial installation, store the current values with:

   ```
   helm get values rancher -n cattle-system -o yaml > values.yaml
   ```

2. Also get the version string of the currently deployed Rancher chart:

   ```
   helm ls -n cattle-system
   ```

3. Update the current Helm values in the `values.yaml` file:
   a. As a private CA is no longer being used, remove the `privateCA: true` field, or set this to `false`
   b. Adjust the `ingress.tls.source` field as necessary. Please refer to the chart options for more details. Here are some examples:
      i. If using a public CA continue with a value of: `secret`
      ii. If using Let's Encrypt update the value to: `letsEncrypt`
4. Update the Helm values for the Rancher chart using the `values.yaml` file, and the current chart version to prevent an upgrade:

   ```
   helm upgrade rancher rancher-prime/rancher \
     --namespace cattle-system \
     -f values.yaml \
     --version <DEPLOYED_RANCHER_VERSION>
   ```

#### 4. Reconfigure Rancher agents for the non-private/common certificate

As a private CA is no longer being used, the `CATTLE_CA_CHECKSUM` environment variable on the downstream cluster agents should be removed or set to "" (an empty string).

#### 5. Force Update SUSE® Rancher Prime: Continuous Delivery clusters to reconnect the fleet-agent to Rancher

Select 'Force Update' for the clusters within the Continuous Delivery view of the Rancher UI to allow the fleet-agent in downstream clusters to successfully connect to Rancher.

#### Why is this step required?

Fleet agents in Rancher managed clusters store a kubeconfig that is used to connect to Rancher. The kubeconfig contains a `certificate-authority-data` field containing the CA for the certificate used by Rancher. When changing the CA, this block needs to be updated to allow the fleet-agent to trust the certificate used by Rancher.

## Upgrading Cert-Manager

Rancher is compatible with the API version cert-manager.io/v1 and was last tested with cert-manager version v1.13.1.

Rancher uses cert-manager to automatically generate and renew TLS certificates for HA deployments of Rancher. As of Fall 2019, three important changes to cert-manager are set to occur that you need to take action on if you have an HA deployment of Rancher:

1. Let's Encrypt will be blocking cert-manager instances older than 0.8.0 starting November 1st 2019. [https://community.letsencrypt.org/t/blocking-old-cert-manager-versions/98753]
2. Cert-manager is deprecating and replacing the certificate.spec.acme.solvers field [https://cert-manager.io/docs/installation/upgrading/upgrading-0.7-0.8/]. This change has no exact deadline.
3. Cert-manager is deprecating `v1alphal` API and replacing its API group [https://cert-manager.io/docs/installation/upgrading/upgrading-0.10-0.11/]

To address these changes, this guide will do two things:

1. Document the procedure for upgrading cert-manager
2. Explain the cert-manager API changes and link to cert-manager's official documentation for migrating your data



If you are upgrading cert-manager to the latest version from a version older than 1.5, follow the steps via the method Upgrade cert-manager from Versions 1.5 and Below. Note that you do not need to reinstall Rancher to perform this upgrade.

#### cert-manager Upgrade Methods

The namespace used in these instructions depends on the namespace cert-manager is currently installed in. If it is in kube-system use that in the instructions below. You can verify by running `kubectl get pods --all-namespaces` and checking which namespace the cert-manager-* pods are listed in. Do not change the namespace cert-manager is running in or this can cause issues.

In order to upgrade cert-manager, follow these instructions:

##### Upgrade cert-manager Using SUSE Application Collection

*Click to expand*

The cert-manager chart can be upgraded in-place using the Helm `upgrade` workflow to a specific version [https://apps.rancher.io/applications/cert-manager] listed in the SUSE Application Collection, below is an example command:

```
helm upgrade <cert-manager-release-version>
```

```
oci://dp.apps.rancher.io/charts/cert-manager \
  --set global.imagePullSecrets={application-collection}
```

To reuse the existing values and only upgrade the chart version:

```
helm upgrade <cert-manager-release-version>
oci://dp.apps.rancher.io/charts/cert-manager \
  --reuse-values
```

#### Upgrade cert-manager from Versions 1.5 and Below

*Click to expand*

Previously, in order to upgrade cert-manager from an older version, an uninstall and reinstall of Rancher was recommended. Using the method below, you may upgrade cert-manager without those additional steps in order to better preserve your production environment:

1. Install `cmctl`, the cert-manager CLI tool, using the installation guide [https://cert-manager.io/docs/usage/cmctl/#installation].
2. Ensure that any cert-manager custom resources that may have been stored in etcd at a deprecated API version get migrated to `v1`:

   ```
   cmctl upgrade migrate-api-version
   ```

   Refer to the API version migration docs [https://cert-manager.io/docs/usage/cmctl/#migrate-api-version] for more information. Please also see the docs to upgrade from 1.5 to 1.6 [https://cert-manager.io/docs/installation/upgrading/upgrading-1.5-1.6/] and the docs to upgrade from 1.6. to 1.7 [https://cert-manager.io/docs/installation/upgrading/upgrading-1.6-1.7/] if needed.

3. Upgrade cert-manager to v1.7.1 with a normal `helm upgrade`. You may go directly from version 1.5 to 1.7 if desired.
4. Follow the Helm tutorial to update the API version of a release manifest [https://helm.sh/docs/topics/kubernetes_apis/#updating-api-versions-of-a-release-manifest]. The chart release name is `release_name=rancher` and the release namespace is `release_namespace=cattle-system`.
5. In the decoded file, search for `cert-manager.io/v1beta1` and **replace it** with `cert-manager.io/v1`.
6. Upgrade Rancher normally with `helm upgrade`.

#### Verify the Deployment

Once you've installed cert-manager, you can verify it is deployed correctly by checking the kube-system namespace for running pods:

```
kubectl get pods --namespace cert-manager

NAME                                       READY   STATUS    RESTARTS
```

```
AGE
cert-manager-5c6866597-zw7kh              1/1     Running   0
2m
cert-manager-cainjector-577f6d9fd7-tr77l  1/1     Running   0
2m
cert-manager-webhook-787858fcdb-nlzsq     1/1     Running   0
2m
```

**Cert-Manager API change and data migration**

---

Rancher now supports cert-manager versions 1.6.2 and 1.7.1. We recommend v1.7.x because v 1.6.x will reach end-of-life on March 30, 2022. To read more, see the cert-manager docs. For instructions on upgrading cert-manager from version 1.5 to 1.6, see the upstream cert-manager documentation here [https://cert-manager.io/docs/installation/upgrading/upgrading-1.5-1.6/]. For instructions on upgrading cert-manager from version 1.6 to 1.7, see the upstream cert-manager documentation here [https://cert-manager.io/docs/installation/upgrading/upgrading-1.6-1.7/].

---

Cert-manager has deprecated the use of the `certificate.spec.acme.solvers` field and will drop support for it completely in an upcoming release.

Per the cert-manager documentation, a new format for configuring ACME certificate resources was introduced in v0.8. Specifically, the challenge solver configuration field was moved. Both the old format and new are supported as of v0.9, but support for the old format will be dropped in an upcoming release of cert-manager. The cert-manager documentation strongly recommends that after upgrading you update your ACME Issuer and Certificate resources to the new format.

Details about the change and migration instructions can be found in the cert-manager v0.7 to v0.8 upgrade instructions [https://cert-manager.io/docs/installation/upgrading/upgrading-0.7-0.8/].

The v0.11 release marks the removal of the `v1alpha1` API that was used in previous versions of cert-manager, as well as our API group changing to be cert-manager.io instead of certmanager.k8s.io.

We have also removed support for the old configuration format that was deprecated in the v0.8 release. This means you must transition to using the new solvers style configuration format for your ACME issuers before upgrading to v0.11. For more information, see the upgrading to v0.8 guide [https://cert-manager.io/docs/installation/upgrading/upgrading-0.7-0.8/].

Details about the change and migration instructions can be found in the cert-manager v0.10 to v0.11 upgrade instructions [https://cert-manager.io/docs/installation/upgrading/upgrading-0.10-0.11/].

More info about cert-manager upgrade information [https://cert-manager.io/docs/installation/upgrade/].

# Infrastructure Setup

Don't have infrastructure for your Kubernetes cluster? Try one of these tutorials.

To set up infrastructure for a high-availability K3s Kubernetes cluster with an external database, refer to this

page.

## Setting up Infrastructure for a High Availability SUSE® Rancher Prime: K3s Kubernetes Cluster

This tutorial is intended to help you provision the underlying infrastructure for a Rancher management server.

The recommended infrastructure for the Rancher-only Kubernetes cluster differs depending on whether Rancher will be installed on a K3s Kubernetes cluster, an RKE2 Kubernetes cluster, or a single Docker container.

For more information about each installation option, refer to this page.



These nodes must be in the same region. You may place these servers in separate availability zones (datacenter).

To install the Rancher management server on a high-availability K3s cluster, we recommend setting up the following infrastructure:

- **Two Linux nodes,** typically virtual machines, in the infrastructure provider of your choice.
- **An external database** to store the cluster data. We recommend MySQL.
- **A load balancer** to direct traffic to the two nodes.
- **A DNS record** to map a URL to the load balancer. This will become the Rancher server URL, and downstream Kubernetes clusters will need to reach it.

#### 1. Set up Linux Nodes

Make sure that your nodes fulfill the general installation requirements for OS, container runtime, hardware, and networking.

For an example of one way to set up Linux nodes, refer to this tutorial for setting up nodes as instances in Amazon EC2.

#### 2. Set up External Datastore

The ability to run Kubernetes using a datastore other than etcd sets K3s apart from other Kubernetes distributions. This feature provides flexibility to Kubernetes operators. The available options allow you to select a datastore that best fits your use case.

For a high-availability K3s installation, you will need to set a MySQL [https://www.mysql.com/] external database. Rancher has been tested on K3s Kubernetes clusters using MySQL version 5.7 as the datastore.

When you install Kubernetes using the K3s installation script, you will pass in details for K3s to connect to the database.

For an example of one way to set up the MySQL database, refer to this tutorial for setting up MySQL on Amazon's RDS service.

For the complete list of options that are available for configuring a K3s cluster datastore, refer to the K3s documentation. [https://rancher.com/docs/k3s/latest/en/installation/datastore/]

### 3. Set up the Load Balancer

You will also need to set up a load balancer to direct traffic to the Rancher replica on both nodes. That will prevent an outage of any single node from taking down communications to the Rancher management server.

When Kubernetes gets set up in a later step, the K3s tool will deploy a Traefik Ingress controller. This controller will listen on ports 80 and 443 of the worker nodes, answering traffic destined for specific hostnames.

When Rancher is installed (also in a later step), the Rancher system creates an Ingress resource. That Ingress tells the Traefik Ingress controller to listen for traffic destined for the Rancher hostname. The Traefik Ingress controller, when receiving traffic destined for the Rancher hostname, will forward that traffic to the running Rancher pods in the cluster.

For your implementation, consider if you want or need to use a Layer-4 or Layer-7 load balancer:

- **A layer-4 load balancer** is the simpler of the two choices, in which you are forwarding TCP traffic to your nodes. We recommend configuring your load balancer as a Layer 4 balancer, forwarding traffic to ports TCP/80 and TCP/443 to the Rancher management cluster nodes. The Ingress controller on the cluster will redirect HTTP traffic to HTTPS and terminate SSL/TLS on port TCP/443. The Ingress controller will forward traffic to port TCP/80 to the Ingress pod in the Rancher deployment.
- **A layer-7 load balancer** is a bit more complicated but can offer features that you may want. For instance, a layer-7 load balancer is capable of handling TLS termination at the load balancer, as opposed to Rancher doing TLS termination itself. This can be beneficial if you want to centralize your TLS termination in your infrastructure. Layer-7 load balancing also offers the capability for your load balancer to make decisions based on HTTP attributes such as cookies, etc. that a layer-4 load balancer is not able to concern itself with. If you decide to terminate the SSL/TLS traffic on a layer-7 load balancer, you will need to use the `--set tls=external` option when installing Rancher in a later step. For more information, refer to the Rancher Helm chart options.

For an example showing how to set up an NGINX load balancer, refer to this page.

For a how-to guide for setting up an Amazon ELB Network Load Balancer, refer to this page.



Do not use this load balancer (i.e, the `local` cluster Ingress) to load balance applications other than Rancher following installation. Sharing this Ingress with other applications may result in websocket errors to Rancher following Ingress configuration reloads for other apps. We recommend dedicating the `local` cluster to Rancher and no other applications.

### 4. Set up the DNS Record

Once you have set up your load balancer, you will need to create a DNS record to send traffic to this load balancer.

Depending on your environment, this may be an A record pointing to the load balancer IP, or it may be a CNAME pointing to the load balancer hostname. In either case, make sure this record is the hostname that you intend Rancher to respond on.

You will need to specify this hostname in a later step when you install Rancher, and it is not possible to change it later. Make sure that your decision is a final one.

For a how-to guide for setting up a DNS record to route domain traffic to an Amazon ELB load balancer, refer

to the official AWS documentation. [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer]

## Setting up Infrastructure for a High Availability SUSE® Rancher Prime: RKE2 Kubernetes Cluster

This tutorial is intended to help you provision the underlying infrastructure for a Rancher management server.

The recommended infrastructure for the Rancher-only Kubernetes cluster differs depending on whether Rancher will be installed on a RKE2 Kubernetes cluster, an RKE Kubernetes cluster, or a single Docker container.



These nodes must be in the same region. You may place these servers in separate availability zones (datacenter).

To install the Rancher management server on a high-availability RKE2 cluster, we recommend setting up the following infrastructure:

- **Three Linux nodes,** typically virtual machines, in the infrastructure provider of your choice.
- **A load balancer** to direct traffic to the nodes.
- **A DNS record** to map a URL to the load balancer. This will become the Rancher server URL, and downstream Kubernetes clusters will need to reach it.

### 1. Set up Linux Nodes

Make sure that your nodes fulfill the general installation requirements for OS, container runtime, hardware, and networking.

For an example of one way to set up Linux nodes, refer to this tutorial for setting up nodes as instances in Amazon EC2.

### 2. Set up the Load Balancer

You will also need to set up a load balancer to direct traffic to the Rancher replica on all nodes. That will prevent an outage of any single node from taking down communications to the Rancher management server.



**Ingress-NGINX EOL:** The community `ingress-nginx` controller reached End-of-Life (EOL) March 2026 and is deprecated in RKE2 v1.36. SUSE Rancher Prime provisioned RKE2 versions will continue to receive `ingress-nginx` support and CVE fixes (8+) for the entire lifecycle of RKE2 v1.32, v1.33, v1.34, v1.35 and v1.36. Traefik is the recommended migration path for SUSE Rancher RKE2 environments, it is recommended to migrate as soon as possible. Refer to Migrating from Ingress NGINX to Traefik in a Rancher provisioned RKE2 Cluster [https://support.scc.suse.com/s/kb/Migrating-from-Ingress-NGINX-to-Traefik-in-a-Rancher-provisioned-RKE2-Cluster?language=en_US] for further details.

When Kubernetes gets set up in a later step, the RKE2 tool will deploy an Nginx Ingress controller. This controller will listen on ports 80 and 443 of the worker nodes, answering traffic destined for specific hostnames.

When Rancher is installed (also in a later step), the Rancher system creates an Ingress resource. That Ingress

tells the Nginx Ingress controller to listen for traffic destined for the Rancher hostname. The Nginx Ingress controller, when receiving traffic destined for the Rancher hostname, will forward that traffic to the running Rancher pods in the cluster.

For your implementation, consider if you want or need to use a Layer-4 or Layer-7 load balancer:

- **A layer-4 load balancer** is the simpler of the two choices, in which you are forwarding TCP traffic to your nodes. We recommend configuring your load balancer as a Layer 4 balancer, forwarding traffic to ports TCP/80 and TCP/443 to the Rancher management cluster nodes. The Ingress controller on the cluster will redirect HTTP traffic to HTTPS and terminate SSL/TLS on port TCP/443. The Ingress controller will forward traffic to port TCP/80 to the Ingress pod in the Rancher deployment.
- **A layer-7 load balancer** is a bit more complicated but can offer features that you may want. For instance, a layer-7 load balancer is capable of handling TLS termination at the load balancer, as opposed to Rancher doing TLS termination itself. This can be beneficial if you want to centralize your TLS termination in your infrastructure. Layer-7 load balancing also offers the capability for your load balancer to make decisions based on HTTP attributes such as cookies, etc. that a layer-4 load balancer is not able to concern itself with. If you decide to terminate the SSL/TLS traffic on a layer-7 load balancer, you will need to use the `--set tls=external` option when installing Rancher in a later step. For more information, refer to the Rancher Helm chart options.

For an example showing how to set up an NGINX load balancer, refer to this page.

For a how-to guide for setting up an Amazon ELB Network Load Balancer, refer to this page.



Do not use this load balancer (i.e, the `local` cluster Ingress) to load balance applications other than Rancher following installation. Sharing this Ingress with other applications may result in websocket errors to Rancher following Ingress configuration reloads for other apps. We recommend dedicating the `local` cluster to Rancher and no other applications.

**3. Set up the DNS Record**

Once you have set up your load balancer, you will need to create a DNS record to send traffic to this load balancer.

Depending on your environment, this may be an A record pointing to the load balancer IP, or it may be a CNAME pointing to the load balancer hostname. In either case, make sure this record is the hostname that you intend Rancher to respond on.

You will need to specify this hostname in a later step when you install Rancher, and it is not possible to change it later. Make sure that your decision is a final one.

For a how-to guide for setting up a DNS record to route domain traffic to an Amazon ELB load balancer, refer to the official AWS documentation. [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer]

## Setting up Nodes in Amazon EC2

In this tutorial, you will learn one way to set up Linux nodes for the Rancher management server. These nodes will fulfill the node requirements for OS, hardware, and networking.

If the Rancher server will be installed on a K3s Kubernetes cluster, you only need to provision two instances.

**1. Optional Preparation**

- **Create IAM role:** To allow Rancher to manipulate AWS resources, such as provisioning new storage or new nodes, you will need to configure Amazon as a cloud provider. There are several things you'll need to do to set up the cloud provider on EC2, but part of this process is setting up an IAM role for the Rancher server nodes. For the full details on setting up the cloud provider, refer to this page.
- **Create security group:** We also recommend setting up a security group for the Rancher nodes that complies with the port requirements for Rancher nodes.

**2. Provision Instances**

1. Log into the Amazon AWS EC2 Console [https://console.aws.amazon.com/ec2/] to get started. Make sure to take note of the **Region** where your EC2 instances (Linux nodes) are created, because all of the infrastructure for the Rancher management server should be in the same region.
2. In the left panel, click **Instances**.
3. Click **Launch Instance**.
4. In the section called **Step 1: Choose an Amazon Machine Image (AMI),** we will use Ubuntu 18.04 as the Linux OS, using `ami-0dlcd67c26f5fcal9 (64-bit x86)`. Go to the Ubuntu AMI and click **Select**.
5. In the **Step 2: Choose an Instance Type** section, select the `t2.medium` type.
6. Click **Next: Configure Instance Details**.
7. In the **Number of instances** field, enter the number of instances. A high-availability K3s cluster requires only two instances, while a high-availability RKE cluster requires three instances.
8. Optional: If you created an IAM role for Rancher to manipulate AWS resources, select the new IAM role in the **IAM role** field.
9. Click **Next: Add Storage, Next: Add Tags,** and **Next: Configure Security Group**.
10. In **Step 6: Configure Security Group,** select a security group that complies with the port requirements for Rancher nodes.
11. Click **Review and Launch**.
12. Click **Launch**.
13. Choose a new or existing key pair that you will use to connect to your instance later. If you are using an existing key pair, make sure you already have access to the private key.
14. Click **Launch Instances**.

**Result:** You have created Rancher nodes that satisfy the requirements for OS, hardware, and networking.

## Setting up a MySQL Database in Amazon RDS

This tutorial describes how to set up a MySQL database in Amazon's Relational Database Service (RDS).

This database can later be used as an external datastore for a high-availability K3s Kubernetes cluster.

1. Log into the Amazon AWS RDS Console [https://console.aws.amazon.com/rds/] to get started. Make sure to select the **Region** where your EC2 instances (Linux nodes) are created.
2. In the left panel, click **Databases**.
3. Click **Create database**.

4. In the **Engine type** section, click **MySQL**.
5. In the **Version** section, choose **MySQL 5.7.22**.
6. In **Settings** section, under **Credentials Settings,** enter a master password for the **admin** master username. Confirm the password.
7. Expand the **Additional configuration** section. In the **Initial database name** field, enter a name. The name can have only letters, numbers, and underscores. This name will be used to connect to the database.
8. Click **Create database**.

You'll need to capture the following information about the new database so that the K3s Kubernetes cluster can connect to it.

To see this information in the Amazon RDS console, click **Databases,** and click the name of the database that you created.

- **Username:** Use the admin username.
- **Password:** Use the admin password.
- **Hostname:** Use the **Endpoint** as the hostname. The endpoint is available in the **Connectivity & security** section.
- **Port:** The port should be 3306 by default. You can confirm it in the **Connectivity & security** section.
- **Database name:** Confirm the name by going to the **Configuration** tab. The name is listed under **DB name**.

This information will be used to connect to the database in the following format:

```
mysql://username:password@tcp(hostname:3306)/database-name
```

For more information on configuring the datastore for K3s, refer to the K3s documentation. [https://rancher.com/docs/k3s/latest/en/installation/datastore/]

## Setting up Amazon ELB Network Load Balancer

This how-to guide describes how to set up a Network Load Balancer (NLB) in Amazon's EC2 service that will direct traffic to multiple instances on EC2.

These examples show the load balancer being configured to direct traffic to three Rancher server nodes. If Rancher is installed on a K3s Kubernetes cluster, only two nodes are required.

This tutorial is about one possible way to set up your load balancer, not the only way. Other types of load balancers, such as a Classic Load Balancer or Application Load Balancer, could also direct traffic to the Rancher server nodes.

Rancher only supports using the Amazon NLB when terminating traffic in `tcp` mode for port 443 rather than `tls` mode. This is due to the fact that the NLB does not inject the correct headers into requests when terminated at the NLB. This means that if you want to use certificates managed by the Amazon Certificate Manager (ACM), you should use an ALB.

### Requirements

These instructions assume you have already created Linux instances in EC2. The load balancer will direct traffic to these nodes.

### 1. Create Target Groups

Begin by creating two target groups for the **TCP** protocol, one with TCP port 443 and one regarding TCP port 80 (providing redirect to TCP port 443). You'll add your Linux nodes to these groups.

Your first NLB configuration step is to create two target groups. Technically, only port 443 is needed to access Rancher, but it's convenient to add a listener for port 80, because traffic to port 80 will be automatically redirected to port 443.

The Traefik Ingress should redirect traffic from port 80 to port 443.

1. Log into the Amazon AWS Console [https://console.aws.amazon.com/ec2/] to get started. Make sure to select the **Region** where your EC2 instances (Linux nodes) are created.
2. Select **Services** and choose **EC2**, find the section **Load Balancing** and open **Target Groups**.
3. Click **Create target group** to create the first target group, regarding TCP port 443.



For details on Traefik Ingress health checks, refer to this section.

#### Target Group (TCP port 443)

Configure the first target group according to the table below.

| Option | Setting |
|---|---|
| Target Group Name | `rancher-tcp-443` |
| Target type | `instance` |
| Protocol | `TCP` |
| Port | `443` |
| VPC | Choose your VPC |

Health check settings:

| Option | Setting |
|---|---|
| Protocol | TCP |
| Port | `override`,`80` |
| Healthy threshold | `3` |
| Unhealthy threshold | `3` |
| Timeout | `6 seconds` |
| Interval | `10 seconds` |

Click **Create target group** to create the second target group, regarding TCP port 80.

#### Target Group (TCP port 80)

Configure the second target group according to the table below.

| Option | Setting |
| --- | --- |
| Target Group Name | `rancher-tcp-80` |
| Target type | `instance` |
| Protocol | `TCP` |
| Port | `80` |
| VPC | Choose your VPC |

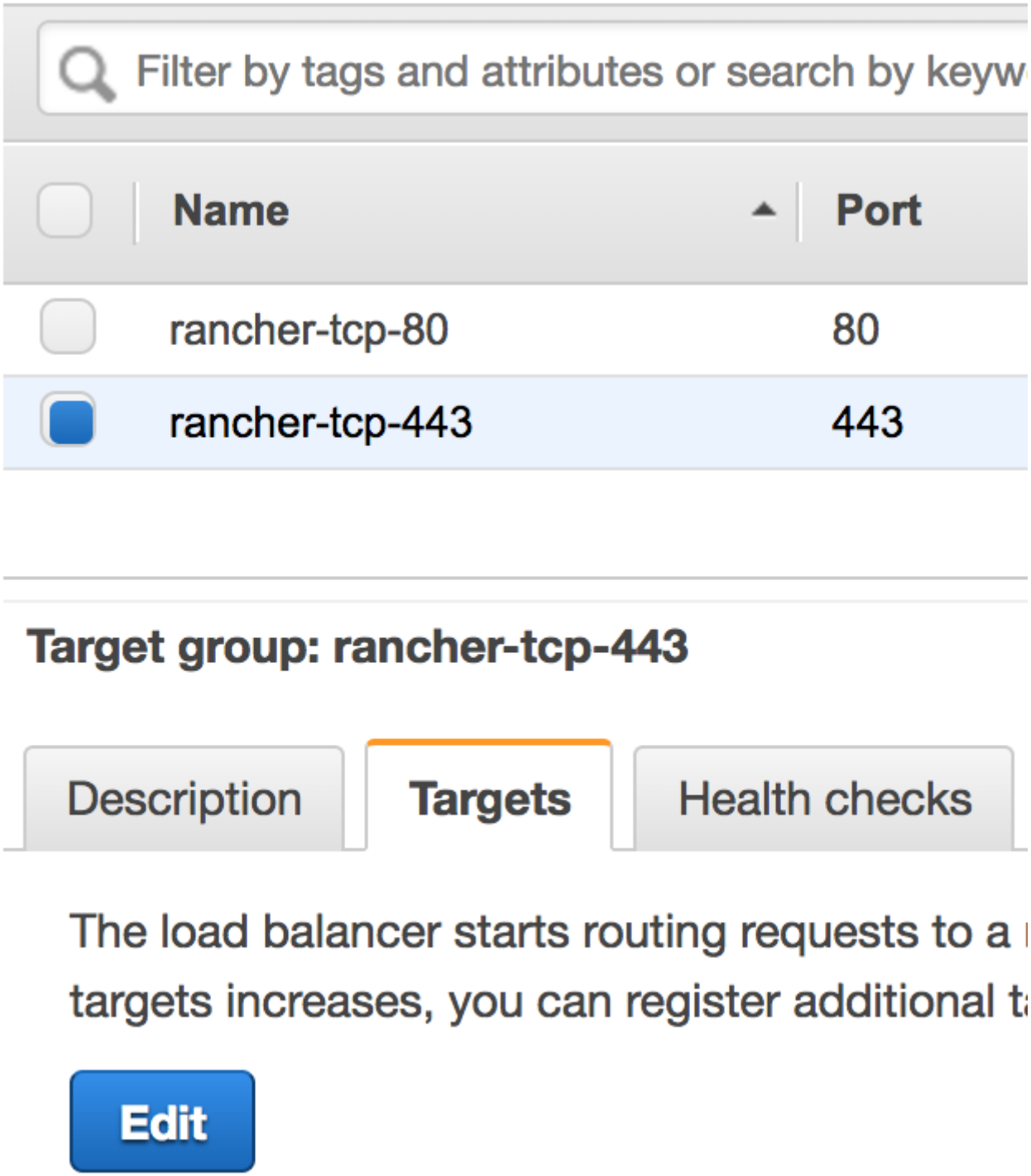
Health check settings:

| Option | Setting |
| --- | --- |
| Protocol | TCP |
| Port | `traffic port` |
| Healthy threshold | `3` |
| Unhealthy threshold | `3` |
| Timeout | `6 seconds` |
| Interval | `10 seconds` |

**2. Register Targets**

Next, add your Linux nodes to both target groups.

Select the target group named **rancher-tcp-443**, click the tab **Targets** and choose **Edit**.




Select the instances (Linux nodes) you want to add, and click **Add to registered**.

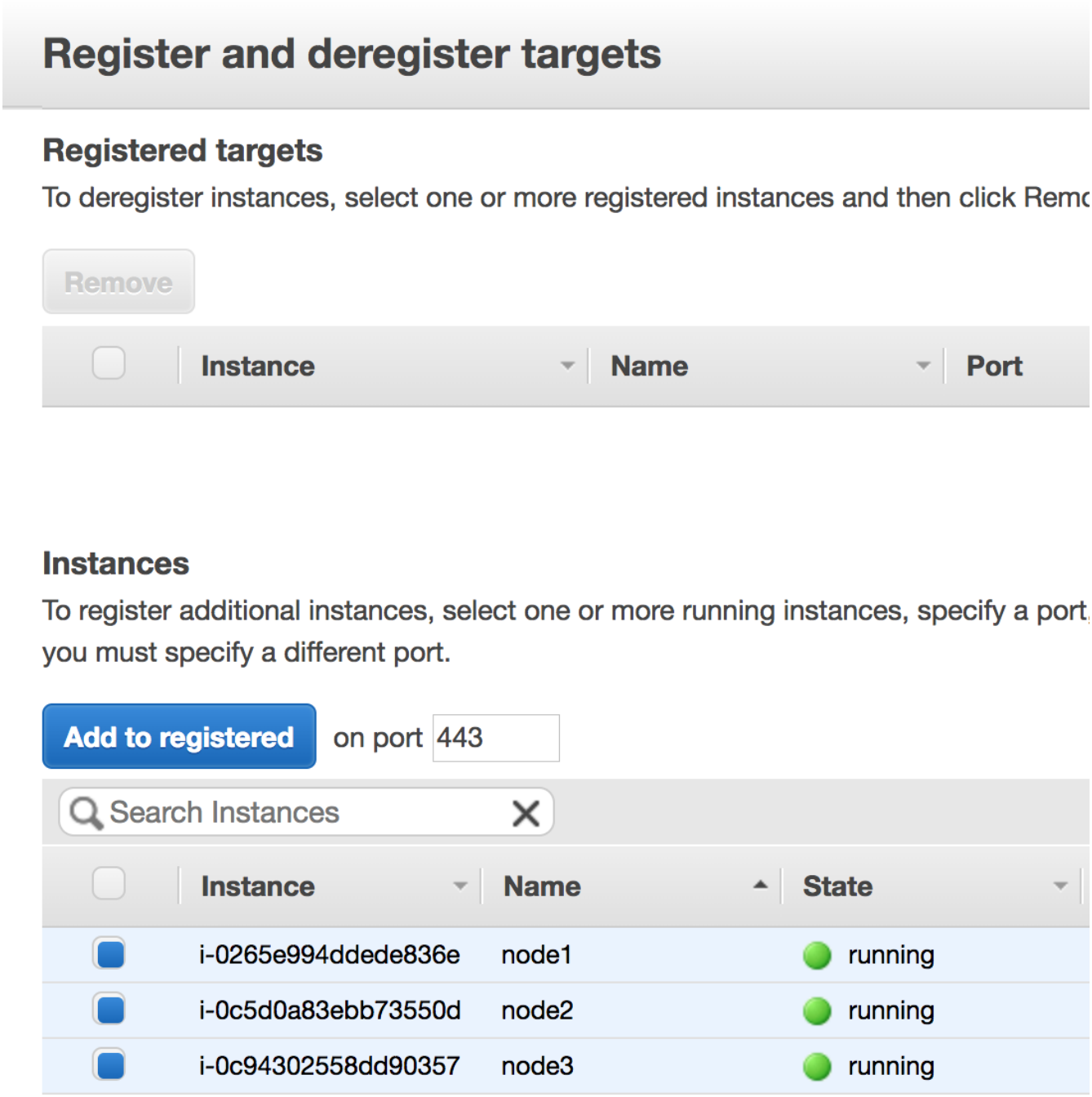



*Figure 8. Screenshot Add targets to target group TCP port 443*

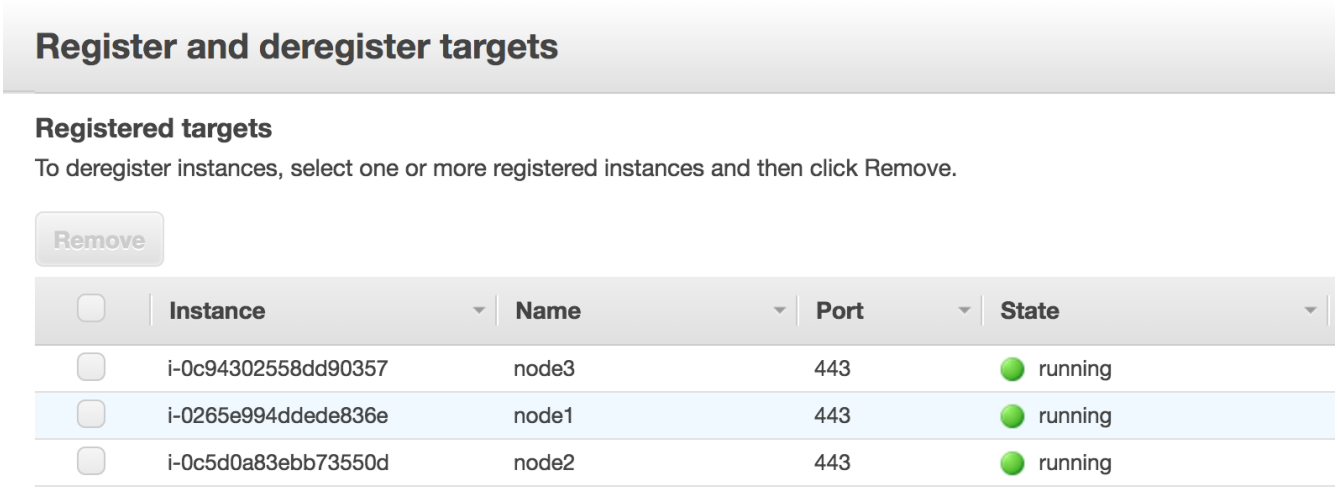



*Figure 9. Screenshot Added targets to target group TCP port 443*

When the instances are added, click **Save** on the bottom right of the screen.

Repeat those steps, replacing **rancher-tcp-443** with **rancher-tcp-80**. The same instances need to be added as targets to this target group.

#### 3. Create Your NLB

Use Amazon's Wizard to create a Network Load Balancer. As part of this process, you'll add the target groups you created in 1. Create Target Groups.

1. From your web browser, navigate to the Amazon EC2 Console [https://console.aws.amazon.com/ec2/].
2. From the navigation pane, choose **LOAD BALANCING** > **Load Balancers**.
3. Click **Create Load Balancer**.
4. Choose **Network Load Balancer** and click **Create**. Then complete each form.
   - Step 1: Configure Load Balancer
   - Step 2: Configure Routing
   - Step 3: Register Targets
   - Step 4: Review

##### Step 1: Configure Load Balancer

Set the following fields in the form:

- **Name:** `rancher`
- **Scheme:** `internal` or `internet-facing`. The scheme that you choose for your NLB is dependent on the configuration of your instances and VPC. If your instances do not have public IPs associated with them, or you will only be accessing Rancher internally, you should set your NLB Scheme to `internal` rather than `internet-facing`.
- **Listeners:** The Load Balancer Protocol should be TCP and the corresponding Load Balancer Port should be set to 443.
- **Availability Zones:** Select Your **VPC** and **Availability Zones**.

#### Step 2: Configure Routing

1. From the **Target Group** drop-down, choose **Existing target group**.
2. From the **Name** drop-down, choose `rancher-tcp-443`.
3. Open **Advanced health check settings**, and configure **Interval** to `10 seconds`.

#### Step 3: Register Targets

Since you registered your targets earlier, all you have to do is click **Next: Review**.

#### Step 4: Review

Look over the load balancer details and click **Create** when you're satisfied.

After AWS creates the NLB, click **Close**.

### 4. Add listener to NLB for TCP port 80

1. Select your newly created NLB and select the **Listeners** tab.
2. Click **Add listener**.
3. Use `TCP:80` as **Protocol** : **Port**
4. Click **Add action** and choose **Forward to...**
5. From the **Forward to** drop-down, choose `rancher-tcp-80`.
6. Click **Save** in the top right of the screen.

### Health Check Paths for Traefik Ingresses

K3s Kubernetes clusters use Traefik as the default Ingress.

The health check path is `/ping`. By default `/ping` is always matched (regardless of Host), and a response from Traefik itself [https://docs.traefik.io/operations/ping/] is always served.

To simulate an accurate health check, it is a best practice to use the Host header (Rancher hostname) combined with `/ping` or `/healthz` wherever possible, to get a response from the Rancher Pods, not the Ingress.

## Using an External Gateway with Rancher

When using the Gateway API network exposure type, Rancher can create and manage its own Gateway resource. However, if you have an existing Gateway that you manage independently (for example, a shared Gateway used by multiple applications), you will need to create your own HTTPRoute resources to route traffic to Rancher.

This section covers how to create the required HTTPRoute resources manually when using an externally managed Gateway.

### Prerequisites

- An existing Gateway resource configured and operational in your cluster
- Knowledge of your Gateway's:

- Name and namespace
- Listener names (sectionName) for HTTP and/or HTTPS traffic

- Rancher installed with `networkExposure.type` set to something other than `gateway` (e.g., `none` or `ingress`)

#### Cross-Namespace Gateway Requirements

If your Gateway is in a different namespace than Rancher (e.g., Gateway in `gateway-system`, Rancher in `cattle-system`), the Gateway must be configured to accept HTTPRoutes from the Rancher namespace. By default, Gateway API only allows routes from the same namespace as the Gateway.

The Gateway owner must configure `allowedRoutes` on the relevant listeners. There are two options:

**Option 1: Allow routes from all namespaces**

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: shared-gateway
  namespace: gateway-system
spec:
  gatewayClassName: example
  listeners:
    - name: https
      port: 443
      protocol: HTTPS
      allowedRoutes:
        namespaces:
          from: All
    - name: http
      port: 80
      protocol: HTTP
      allowedRoutes:
        namespaces:
          from: All
```

**Option 2: Allow routes from specific namespaces (more restrictive)**

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: shared-gateway
  namespace: gateway-system
spec:
  gatewayClassName: example
  listeners:
```

```
    - name: https
      port: 443
      protocol: HTTPS
      allowedRoutes:
        namespaces:
          from: Selector
          selector:
            matchLabels:
              shared-gateway-access: "true"
    - name: http
      port: 80
      protocol: HTTP
      allowedRoutes:
        namespaces:
          from: Selector
          selector:
            matchLabels:
              shared-gateway-access: "true"
```

When using the selector approach, ensure the Rancher namespace has the required label:

```
kubectl label namespace cattle-system shared-gateway-access=true
```



If the Gateway and Rancher are in the same namespace, no additional configuration is needed—the default `allowedRoutes` setting (`from: Same`) will permit the HTTPRoute attachment.

#### Determining Your Rancher Service Values

Before creating HTTPRoute resources, identify the following values from your Rancher installation:

| Value | How to Determine | Example |
|---|---|---|
| **Release Name** | The name used in `helm install <release-name>` | `rancher` |
| **Namespace** | The namespace where Rancher is installed | `cattle-system` |
| **Hostname** | The `hostname` value from your Helm values | `rancher.example.com` |
| **TLS Mode** | The `tls` value from your Helm values | `ingress`, `external`, or `secret` |
| **Service HTTP Disabled** | The `service.disableHTTP` value | `true` or `false` |

The Rancher service name follows the pattern: `<release-name>-rancher` (or just `<release-name>` if the release name already contains "rancher").

### HTTPRoute Configuration

#### Primary HTTPRoute

Create an HTTPRoute to direct traffic from your Gateway to the Rancher service. The configuration depends on your TLS setup:

**When TLS terminates at the Gateway or within Kubernetes (`tls: ingress`, `tls: secret`, or `tls: letsEncrypt`):**

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: rancher
  namespace: cattle-system  # Must match Rancher's namespace
spec:
  parentRefs:
    - name: <your-gateway-name>
      namespace: <your-gateway-namespace>
      sectionName: <your-https-listener-name>  # Your Gateway's HTTPS
listener
  hostnames:
    - <your-rancher-hostname>  # e.g., rancher.example.com
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /
      backendRefs:
        - name: rancher  # Your Rancher service name
          port: 80       # Use 443 if service.disableHTTP=true
```

**When TLS terminates externally (`tls: external`):**

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: rancher
  namespace: cattle-system
spec:
  parentRefs:
    - name: <your-gateway-name>
      namespace: <your-gateway-namespace>
      sectionName: <your-http-listener-name>  # Your Gateway's HTTP
listener
  hostnames:
```

```
    - <your-rancher-hostname>
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /
      backendRefs:
        - name: rancher
          port: 80
```

#### HTTP to HTTPS Redirect Route (Optional)

If TLS terminates at or within Kubernetes (not externally), you may want to redirect HTTP traffic to HTTPS. Create an additional HTTPRoute:

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: rancher-http-redirect
  namespace: cattle-system
spec:
  parentRefs:
    - name: <your-gateway-name>
      namespace: <your-gateway-namespace>
      sectionName: <your-http-listener-name>  # Your Gateway's HTTP
listener
  hostnames:
    - <your-rancher-hostname>
  rules:
    - filters:
        - type: RequestRedirect
          requestRedirect:
            scheme: https
            statusCode: 301
```

### Using extraObjects

You can include these HTTPRoute resources directly in your Rancher Helm installation using the `extraObjects` value. This keeps all resources managed together:

```
# values.yaml
hostname: rancher.example.com
tls: ingress

extraObjects:
```

```yaml
- apiVersion: gateway.networking.k8s.io/v1
  kind: HTTPRoute
  metadata:
    name: rancher
  spec:
    parentRefs:
      - name: my-shared-gateway
        namespace: gateway-system
        sectionName: https
    hostnames:
      - rancher.example.com
    rules:
      - matches:
          - path:
              type: PathPrefix
              value: /
        backendRefs:
          - name: rancher
            port: 80

- apiVersion: gateway.networking.k8s.io/v1
  kind: HTTPRoute
  metadata:
    name: rancher-http-redirect
  spec:
    parentRefs:
      - name: my-shared-gateway
        namespace: gateway-system
        sectionName: http
    hostnames:
      - rancher.example.com
    rules:
      - filters:
          - type: RequestRedirect
            requestRedirect:
              scheme: https
              statusCode: 301
```

#### Backend Port Selection

The port in `backendRefs` depends on your `service.disableHTTP` setting:

| `service.disableHTTP` | Backend Port |
|---|---|
| `false` (default) | `80` |
| `true` | `443` |

#### Listener Selection Summary

| TLS Configuration | Primary Route Listener | Redirect Route |
|---|---|---|
| `tls: external` | HTTP listener | Not needed |
| `tls: ingress` | HTTPS listener | HTTP listener (optional) |
| `tls: secret` | HTTPS listener | HTTP listener (optional) |
| `tls: letsEncrypt` | HTTPS listener | HTTP listener (optional) |

#### Troubleshooting

**HTTPRoute not being accepted:**

- Verify the Gateway name and namespace are correct
- Ensure the `sectionName` matches an existing listener on your Gateway
- Check that the listener allows routes from the Rancher namespace (see Gateway's `allowedRoutes` configuration)

**Connection refused or timeouts:**

- Confirm the Rancher service exists and has endpoints: `kubectl get endpoints rancher -n cattle-system`
- Verify the backend port matches your `service.disableHTTP` setting

**Certificate errors:**

- If using `tls: ingress` or `tls: secret`, ensure your Gateway's HTTPS listener has the appropriate certificate configured
- Verify the certificate covers your Rancher hostname

## Setting up an NGINX Load Balancer

NGINX will be configured as Layer 4 load balancer (TCP) that forwards connections to one of your Rancher nodes.

In this configuration, the load balancer is positioned in front of your nodes. The load balancer can be any host capable of running NGINX.



Do not use one of your Rancher nodes as the load balancer.

> These examples show the load balancer being configured to direct traffic to three Rancher server nodes. If Rancher is installed on an RKE Kubernetes cluster, three nodes are required. If Rancher is installed on a K3s Kubernetes cluster, only two nodes are required.

#### Install NGINX

Start by installing NGINX on the node you want to use as a load balancer. NGINX has packages available for all known operating systems. The versions tested are `1.14` and `1.15`. For help installing NGINX, refer to

their install documentation [https://www.nginx.com/resources/wiki/start/topics/tutorials/install/].

The `stream` module is required, which is present when using the official NGINX packages. Please refer to your OS documentation on how to install and enable the NGINX `stream` module on your operating system.

#### Create NGINX Configuration

After installing NGINX, you need to update the NGINX configuration file, `nginx.conf`, with the IP addresses for your nodes.

1. Copy and paste the code sample below into your favorite text editor. Save it as `nginx.conf`.
2. From `nginx.conf`, replace both occurrences (port 80 and port 443) of `<IP_NODE_1>`, `<IP_NODE_2>`, and `<IP_NODE_3>` with the IPs of your nodes.



See NGINX Documentation: TCP and UDP Load Balancing [https://docs.nginx.com/nginx/admin-guide/load-balancer/tcp-udp-load-balancer/] for all configuration options.

*Example NGINX config*

```
worker_processes 4;
worker_rlimit_nofile 40000;

events {
    worker_connections 8192;
}

stream {
    upstream rancher_servers_http {
        least_conn;
        server <IP_NODE_1>:80 max_fails=3 fail_timeout=5s;
        server <IP_NODE_2>:80 max_fails=3 fail_timeout=5s;
        server <IP_NODE_3>:80 max_fails=3 fail_timeout=5s;
    }
    server {
        listen 80;
        proxy_pass rancher_servers_http;
    }

}

http {

    upstream rancher_servers_https {
        least_conn;
        server <IP_NODE_1>:443 max_fails=3 fail_timeout=5s;
        server <IP_NODE_2>:443 max_fails=3 fail_timeout=5s;
        server <IP_NODE_3>:443 max_fails=3 fail_timeout=5s;
```

```
    }
    server {
        listen 443 ssl;
        ssl_certificate /path/to/tls.crt;
        ssl_certificate_key /path/to/key.key;
        location / {
            proxy_pass https://rancher_servers_https;
            proxy_set_header Host <rancher UI URL>;
            proxy_ssl_server_name on;
            proxy_ssl_name <rancher UI URL>
        }
    }
}
```

3. Save `nginx.conf` to your load balancer at the following path: `/etc/nginx/nginx.conf`.
4. Load the updates to your NGINX configuration by running the following command:

```
# nginx -s reload
```

**Option - Run NGINX as Docker container**

Instead of installing NGINX as a package on the operating system, you can rather run it as a Docker container. Save the edited **Example NGINX config** as `/etc/nginx.conf` and run the following command to launch the NGINX container:

```
docker run -d --restart=unless-stopped \
  -p 80:80 -p 443:443 \
  -v /etc/nginx.conf:/etc/nginx/nginx.conf \
  nginx:1.14
```

# Setting up a Kubernetes Cluster for SUSE Rancher Prime Server

Don't have a Kubernetes cluster? Try one of these tutorials.

This section contains information on how to install a Kubernetes cluster that the Rancher server can be installed on.

Rancher can run on any Kubernetes cluster.

## Setting up a High-availability SUSE® Rancher Prime: K3s Kubernetes Cluster for SUSE Rancher Prime

This section describes how to install a Kubernetes cluster according to the best practices for the Rancher server environment.

For systems without direct internet access, refer to the air gap installation instructions.



*Single-node Installation Tip:*

In a single-node Kubernetes cluster, the Rancher server does not have high availability, which is important for running Rancher in production. However, installing Rancher on a single-node cluster can be useful if you want to save resources by using a single node in the short term, while preserving a high-availability migration path.

To set up a single-node K3s cluster, run the Rancher server installation command on just one node instead of two nodes.

In both single-node setups, Rancher can be installed with Helm on the Kubernetes cluster in the same way that it would be installed on any other cluster.

**Prerequisites**

These instructions assume you have set up two nodes, a load balancer, a DNS record, and an external MySQL database as described in this section.

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the Rancher Support Matrix [https://rancher.com/support-maintenance-terms/].

To specify the K3s (Kubernetes) version, use the INSTALL_K3S_VERSION (e.g., `INSTALL_K3S_VERSION="v1.24.10+k3s1"`) environment variable when running the K3s installation script.

**Installing Kubernetes**

**1. Install Kubernetes and Set up the SUSE® Rancher Prime: K3s Server**

When running the command to start the K3s Kubernetes API server, you will pass in an option to use the external datastore that you set up earlier.

1. Connect to one of the Linux nodes that you have prepared to run the Rancher server.
2. On the Linux node, run this command to start the K3s server and connect it to the external datastore:

```
 curl -sfL https://get.k3s.io |  INSTALL_K3S_VERSION=<VERSION> sh -s
- server \
   --datastore-endpoint="<DATASTORE_ENDPOINT>"
```

   Where `<DATASTORE_ENDPOINT>` is the connection URI for your datastore. For example, `mysql://username:password@tcp(hostname:3306)/database-name` if you're using MySQL. Valid datastores include etcd, MySQL, PostgreSQL, or SQLite (default).



   The datastore endpoint can also be passed in using the environment variable `$K3S_DATASTORE_ENDPOINT`.

3. Get main server node token:

```
cat /var/lib/rancher/k3s/server/token
```

4. Run command on your second K3s server node:

```
   curl -sfL https://get.k3s.io |  INSTALL_K3S_VERSION=<VERSION> sh
-s - server \
     --datastore-endpoint="<DATASTORE_ENDPOINT>" \
     --token "<MAIN_SERVER_NODE_TOKEN>"
```

#### 2. Confirm that SUSE® Rancher Prime: K3s is Running

To confirm that K3s has been set up successfully, run the following command on either of the K3s server nodes:

```
sudo k3s kubectl get nodes
```

Then you should see two nodes with the master role:

```
ubuntu@ip-172-31-60-194:~$ sudo k3s kubectl get nodes
NAME               STATUS   ROLES    AGE    VERSION
ip-172-31-60-194   Ready    master   44m    v1.17.2+k3s1
ip-172-31-63-88    Ready    master   6m8s   v1.17.2+k3s1
```

Then test the health of the cluster pods:

```
sudo k3s kubectl get pods --all-namespaces
```

**Result:** You have successfully set up a K3s Kubernetes cluster.

#### 3. Save and Start Using the kubeconfig File

When you installed K3s on each Rancher server node, a `kubeconfig` file was created on the node at `/etc/rancher/k3s/k3s.yaml`. This file contains credentials for full access to the cluster, and you should save this file in a secure location.

To use this `kubeconfig` file,

1. Install kubectl, [https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl] a Kubernetes command-line tool.
2. Copy the file at `/etc/rancher/k3s/k3s.yaml` and save it to the directory `~/.kube/config` on your local machine.
3. In the kubeconfig file, the `server` directive is defined as localhost. Configure the server as the DNS of your load balancer, referring to port 6443. (The Kubernetes API server will be reached at port 6443, while the Rancher server will be reached at ports 80 and 443.) Here is an example `k3s.yaml`:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [CERTIFICATE-DATA]
    server: [LOAD-BALANCER-DNS]:6443 # Edit this line
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    password: [PASSWORD]
    username: admin
```

**Result:** You can now use `kubectl` to manage your K3s cluster. If you have more than one kubeconfig file, you can specify which one you want to use by passing in the path to the file when using `kubectl`:

```
kubectl --kubeconfig ~/.kube/config/k3s.yaml get pods --all-namespaces
```

For more information about the `kubeconfig` file, refer to the K3s documentation [https://rancher.com/docs/k3s/latest/en/cluster-access/] or the official Kubernetes documentation [https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/] about organizing cluster access using `kubeconfig` files.

#### 4. Check the Health of Your Cluster Pods

Now that you have set up the `kubeconfig` file, you can use `kubectl` to access the cluster from your local machine.

Check that all the required pods and containers are healthy are ready to continue:

```
ubuntu@ip-172-31-60-194:~$ sudo kubectl get pods --all-namespaces
NAMESPACE       NAME                                      READY   STATUS
RESTARTS   AGE
kube-system     metrics-server-6d684c7b5-bw59k            1/1
Running   0          8d
kube-system     local-path-provisioner-58fb86bdfd-fmkvd   1/1
Running   0          8d
kube-system     coredns-d798c9dd-ljjnf                    1/1
```

```
Running   0          8d
```

**Result:** You have confirmed that you can access the cluster with `kubectl` and the K3s cluster is running successfully. Now the Rancher management server can be installed on the cluster.

## Setting up a High-availability SUSE® Rancher Prime: RKE2 Kubernetes Cluster for SUSE Rancher Prime

This section describes how to install a Kubernetes cluster according to the best practices for the Rancher server environment.

### Prerequisites

These instructions assume you have set up three nodes, a load balancer, and a DNS record, as described in this section.

Note that in order for RKE2 to work correctly with the load balancer, you need to set up two listeners: one for the supervisor on port 9345, and one for the Kubernetes API on port 6443.

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the support maintenance terms. [https://rancher.com/support-maintenance-terms/] To specify the RKE2 version, use the INSTALL_RKE2_VERSION environment variable when running the RKE2 installation script.

### Installing Kubernetes

#### 1. Install Kubernetes and Set up the SUSE® Rancher Prime: RKE2 Server

RKE2 server runs with embedded etcd so you will not need to set up an external datastore to run in HA mode.

On the first node, you should set up the configuration file with your own pre-shared secret as the token. The token argument can be set on startup.

If you do not specify a pre-shared secret, RKE2 will generate one and place it at /var/lib/rancher/rke2/server/node-token.

To avoid certificate errors with the fixed registration address, you should launch the server with the tls-san parameter set. This option adds an additional hostname or IP as a Subject Alternative Name in the server's TLS cert, and it can be specified as a list if you would like to access via both the IP and the hostname.

First, you must create the directory where the RKE2 config file is going to be placed:

```
mkdir -p /etc/rancher/rke2/
```

Next, create the RKE2 config file at `/etc/rancher/rke2/config.yaml` using the following example:

```
token: my-shared-secret
tls-san:
  - my-kubernetes-domain.com
```

```
  - another-kubernetes-domain.com
```

After that, you need to run the install command and enable and start rke2:

```
curl -sfL https://get.rke2.io | sh -
systemctl enable rke2-server.service
systemctl start rke2-server.service
```

1. To join the rest of the nodes, you need to configure each additional node with the same shared token or the one generated automatically. Here is an example of the configuration file:

   ```
   token: my-shared-secret
   server: https://<DNS-DOMAIN>:9345
   tls-san:
     - my-kubernetes-domain.com
     - another-kubernetes-domain.com After that, you need to run the
  installer and enable, then start, rke2:

   curl -sfL https://get.rke2.io | sh -
   systemctl enable rke2-server.service
   systemctl start rke2-server.service
   ```

2. Repeat the same command on your third RKE2 server node.

#### 2. Confirm that SUSE® Rancher Prime: RKE2 is Running

Once you've launched the rke2 server process on all server nodes, ensure that the cluster has come up properly with

```
/var/lib/rancher/rke2/bin/kubectl \
        --kubeconfig /etc/rancher/rke2/rke2.yaml get nodes
You should see your server nodes in the Ready state.
```

Then test the health of the cluster pods:

```
/var/lib/rancher/rke2/bin/kubectl \
        --kubeconfig /etc/rancher/rke2/rke2.yaml get pods --all
-namespaces
```

**Result:** You have successfully set up a RKE2 Kubernetes cluster.

#### 3. Save and Start Using the kubeconfig File

When you installed RKE2 on each Rancher server node, a `kubeconfig` file was created on the node at

`/etc/rancher/rke2/rke2.yaml`. This file contains credentials for full access to the cluster, and you should save this file in a secure location.

To use this `kubeconfig` file,

1. Install kubectl, [https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl] a Kubernetes command-line tool.
2. Copy the file at `/etc/rancher/rke2/rke2.yaml` and save it to the directory `~/.kube/config` on your local machine.
3. In the kubeconfig file, the `server` directive is defined as localhost. Configure the server as the DNS of your control-plane load balancer, on port 6443. (The RKE2 Kubernetes API Server uses port 6443, while the Rancher server will be served via the Traefik Ingress on ports 80 and 443. Refer to Migrating from Ingress NGINX to Traefik in a Rancher provisioned RKE2 Cluster [https://support.scc.suse.com/s/kb/Migrating-from-Ingress-NGINX-to-Traefik-in-a-Rancher-provisioned-RKE2-Cluster?language=en_US] for further details.) Here is an example `rke2.yaml`:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [CERTIFICATE-DATA]
    server: [LOAD-BALANCER-DNS]:6443 # Edit this line
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    password: [PASSWORD]
    username: admin
```

**Result:** You can now use `kubectl` to manage your RKE2 cluster. If you have more than one kubeconfig file, you can specify which one you want to use by passing in the path to the file when using `kubectl`:

```
kubectl --kubeconfig ~/.kube/config/rke2.yaml get pods --all-namespaces
```

For more information about the `kubeconfig` file, refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/cluster_access.html] or the official Kubernetes documentation [https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/] about organizing cluster access using `kubeconfig` files.

#### 4. Check the Health of Your Cluster Pods

Now that you have set up the `kubeconfig` file, you can use `kubectl` to access the cluster from your local machine.

Check that all the required pods and containers are healthy are ready to continue:

```
/var/lib/rancher/rke2/bin/kubectl --kubeconfig
/etc/rancher/rke2/rke2.yaml get pods -A
NAMESPACE     NAME
READY   STATUS      RESTARTS   AGE
kube-system   cloud-controller-manager-rke2-server-1
1/1     Running     0          2m28s
kube-system   cloud-controller-manager-rke2-server-2
1/1     Running     0          61s
kube-system   cloud-controller-manager-rke2-server-3
1/1     Running     0          49s
kube-system   etcd-rke2-server-1
1/1     Running     0          2m13s
kube-system   etcd-rke2-server-2
1/1     Running     0          87s
kube-system   etcd-rke2-server-3
1/1     Running     0          56s
kube-system   helm-install-rke2-canal-hs6sx
0/1     Completed   0          2m17s
kube-system   helm-install-rke2-coredns-xmzm8
0/1     Completed   0          2m17s
kube-system   helm-install-traefik-crd-z8vsz
0/1     Completed   0          2m17s
kube-system   helm-install-traefik-flwnl
0/1     Completed   0          2m17s
kube-system   helm-install-rke2-metrics-server-7sggn
0/1     Completed   0          2m17s
kube-system   kube-apiserver-rke2-server-1
1/1     Running     0          116s
kube-system   kube-apiserver-rke2-server-2
1/1     Running     0          66s
kube-system   kube-apiserver-rke2-server-3
1/1     Running     0          48s
kube-system   kube-controller-manager-rke2-server-1
1/1     Running     0          2m30s
kube-system   kube-controller-manager-rke2-server-2
1/1     Running     0          57s
kube-system   kube-controller-manager-rke2-server-3
1/1     Running     0          42s
kube-system   kube-proxy-rke2-server-1
```

```
1/1     Running     0          2m25s
kube-system   kube-proxy-rke2-server-2
1/1     Running     0          59s
kube-system   kube-proxy-rke2-server-3
1/1     Running     0          85s
kube-system   kube-scheduler-rke2-server-1
1/1     Running     0          2m30s
kube-system   kube-scheduler-rke2-server-2
1/1     Running     0          57s
kube-system   kube-scheduler-rke2-server-3
1/1     Running     0          42s
kube-system   rke2-canal-b9lvm
2/2     Running     0          91s
kube-system   rke2-canal-khwp2
2/2     Running     0          2m5s
kube-system   rke2-canal-swfmq
2/2     Running     0          105s
kube-system   rke2-coredns-rke2-coredns-547d5499cb-6tvwb
1/1     Running     0          92s
kube-system   rke2-coredns-rke2-coredns-547d5499cb-rdttj
1/1     Running     0          2m8s
kube-system   rke2-coredns-rke2-coredns-autoscaler-65c9bb465d-85sq5
1/1     Running     0          2m8s
kube-system   traefik-7c844b766f-m9p2w
1/1     Running     0          52s
kube-system   traefik-7c844b766f-s4l9k
1/1     Running     0          52s
kube-system   rke2-metrics-server-6564db4569-vdfkn
1/1     Running     0          66s
```

**Result:** You have confirmed that you can access the cluster with `kubectl` and the RKE2 cluster is running successfully. Now the Rancher management server can be installed on the cluster.

## Installing SUSE Rancher Prime on Azure Kubernetes Service

This page covers how to install Rancher on Microsoft's Azure Kubernetes Service (AKS).

The guide uses command line tools to provision an AKS cluster with an ingress. If you prefer to provision your cluster using the Azure portal, refer to the official documentation [https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough-portal].

If you already have an AKS Kubernetes cluster, skip to the step about installing an ingress. Then install the Rancher Helm chart following the instructions on this page.

### Prerequisites



Deploying to Microsoft Azure will incur charges.

- Microsoft Azure Account [https://azure.microsoft.com/en-us/free/]: A Microsoft Azure Account is required to create resources for deploying Rancher and Kubernetes.
- Microsoft Azure Subscription [https://docs.microsoft.com/en-us/azure/cost-management-billing/manage/create-subscription#create-a-subscription-in-the-azure-portal]: Use this link to follow a tutorial to create a Microsoft Azure subscription if you don't have one yet.
- Micsoroft Azure Tenant [https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-create-new-tenant]: Use this link and follow instructions to create a Microsoft Azure tenant.
- Your subscription has sufficient quota for at least 2 vCPUs. For details on Rancher server resource requirements, refer to this section
- When installing Rancher with Helm in Azure, use the L7 load balancer to avoid networking issues. For more information, refer to the documentation on Azure load balancer limitations [https://docs.microsoft.com/en-us/azure/load-balancer/components#_limitations].

**1. Prepare your Workstation**

Install the following command line tools on your workstation:

- The Azure CLI, **az:** For help, refer to these installation steps. [https://docs.microsoft.com/en-us/cli/azure/]
- **kubectl:** For help, refer to these installation steps. [https://kubernetes.io/docs/tasks/tools/#kubectl]
- **helm:** For help, refer to these installation steps. [https://helm.sh/docs/intro/install/]

**2. Create a Resource Group**

After installing the CLI, you will need to log in with your Azure account.

```
az login
```

Create a resource group [https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal] to hold all relevant resources for your cluster. Use a location that applies to your use case.

```
az group create --name rancher-rg --location eastus
```

**3. Create the AKS Cluster**

To create an AKS cluster, run the following command. Use a VM size that applies to your use case. Refer to this article [https://docs.microsoft.com/en-us/azure/virtual-machines/sizes] for available sizes and options. When choosing a Kubernetes version, be sure to first consult the support matrix [https://rancher.com/support-matrix/] to find the highest version of Kubernetes that has been validated for your Rancher version.

```
az aks create \
  --resource-group rancher-rg \
  --name rancher-server \
  --kubernetes-version <VERSION> \
  --node-count 3 \
```

```
  --node-vm-size Standard_D2_v3
```

The cluster will take some time to be deployed.

#### 4. Get Access Credentials

After the cluster is deployed, get the access credentials.

```
az aks get-credentials --resource-group rancher-rg --name rancher-server
```

This command merges your cluster's credentials into the existing kubeconfig and allows `kubectl` to interact with the cluster.

#### 5. Install an Ingress

The cluster needs an Ingress so that Rancher can be accessed from outside the cluster. Installing an Ingress requires allocating a public IP address. Ensure you have sufficient quota, otherwise it will fail to assign the IP address. Limits for public IP addresses are applicable at a regional level per subscription. You can use a managed ingress controller provided by Azure or a third-party ingress controller like Traefik.



It is not recommended to install a third-party ingress controller, like Traefik, if a managed ingress controller is already being used.



**Ingress-NGINX EOL:** The community `ingress-nginx` controller reaches End-of-Life (EOL) in March 2026. This page uses Traefik, which is the recommended migration path for Rancher environments.

Traefik includes a native Ingress NGINX provider. This allows you to migrate from NGINX without rewriting your existing Ingress objects, as Traefik will automatically interpret `nginx.ingress.kubernetes.io` annotations. If you are upgrading a cluster that is already using `ingress-nginx`, follow this guide [https://doc.traefik.io/traefik/migrate/nginx-to-traefik/] for more information.

To install Traefik, we highly recommend using the Traefik application in the Rancher Prime Application Collection [https://apps.rancher.io/applications/traefik-proxy]. Check their documentation to integrate with Rancher Manager [https://docs.apps.rancher.io/howto-guides/integrate-with-rancher-manager].

#### 6. Get Load Balancer IP

To get the address of the load balancer, run:

```
kubectl get service traefik --namespace=traefik
```

The result should look similar to the following:

```
NAME                      TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)
 AGE
```

```
traefik  LoadBalancer   10.0.116.18    40.31.180.83
80:31229/TCP,443:31050/TCP 67s
```

Save the `EXTERNAL-IP`.

#### 7. Set up DNS

External traffic to the Rancher server will need to be directed at the load balancer you created.

Set up a DNS to point at the `EXTERNAL-IP` that you saved. This DNS will be used as the Rancher server URL.

There are many valid ways to set up the DNS. For help, refer to the Azure DNS documentation [https://docs.microsoft.com/en-us/azure/dns/]

#### 8. Install the Rancher Helm Chart

Next, install the Rancher Helm chart by following the instructions on this page. The Helm instructions are the same for installing Rancher on any Kubernetes distribution.

Use that DNS name from the previous step as the Rancher server URL when you install Rancher. It can be passed in as a Helm option. For example, if the DNS name is `rancher.my.org`, you could run the Helm installation command with the option `--set hostname=rancher.my.org`.

When installing Rancher on top of this setup, you will also need to pass the value below into the Rancher Helm install command in order to set the name of the ingress controller to be used with Rancher's ingress resource:

```
--set ingress.ingressClassName=traefik
```

Refer here for the Helm install command for your chosen certificate option.

### Installing SUSE Rancher Prime on Amazon EKS

This page covers installing Rancher on an Amazon EKS cluster. You can also use SUSE Rancher for AWS [https://documentation.suse.com/en-us/cloudnative/rancher-srfa/] as a distinct, fully managed Software-as-a-Service (SaaS) offering through the AWS Marketplace.

If you already have an EKS Kubernetes cluster, skip to the step about installing an ingress. Then install the Rancher Helm chart following the instructions on this page.

#### Creating an EKS Cluster for the Rancher Server

In this section, you'll install an EKS cluster with an ingress by using command line tools. This guide may be useful if you want to use fewer resources while trying out Rancher on EKS.



*Prerequisites:*

- You should already have an AWS account.
- It is recommended to use an IAM user instead of the root AWS account. You will need the IAM user's access key and secret key to configure the AWS command line interface.

- The IAM user needs the minimum IAM policies described in the official eksctl documentation. [https://eksctl.io/usage/minimum-iam-policies/]

#### 1. Prepare your Workstation

Install the following command line tools on your workstation:

- **The AWS CLI v2:** For help, refer to these installation steps. [https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html]
- **eksctl:** For help, refer to these installation steps. [https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html]
- **kubectl:** For help, refer to these installation steps. [https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html]
- **helm:** For help, refer to these installation steps. [https://helm.sh/docs/intro/install/]

#### 2. Configure the AWS CLI

To configure the AWS CLI, run the following command:

```
aws configure
```

Then enter the following values:

| Value | Description |
|---|---|
| AWS Access Key ID | The access key credential for the IAM user with EKS permissions. |
| AWS Secret Access Key | The secret key credential for the IAM user with EKS permissions. |
| Default region name | An AWS region [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html#Concepts.RegionsAndAvailabilityZones.Regions] where the cluster nodes will be located. |
| Default output format | Enter `json`. |

#### 3. Create the EKS Cluster

To create an EKS cluster, run the following command. Use the AWS region that applies to your use case. When choosing a Kubernetes version, be sure to first consult the support matrix [https://rancher.com/support-matrix/] to find the highest version of Kubernetes that has been validated for your Rancher version.

```
eksctl create cluster \
  --name rancher-server \
  --version <VERSION> \
  --region us-west-2 \
  --nodegroup-name rancherнodes \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
```

```
  --managed
```

The cluster will take some time to be deployed with CloudFormation.

#### 4. Test the Cluster

To test the cluster, run:

```
eksctl get cluster
```

The result should look like the following:

```
eksctl get cluster
2021-03-18 15:09:35 [ℹ]  eksctl version 0.40.0
2021-03-18 15:09:35 [ℹ]  using region us-west-2
NAME        REGION      EKSCTL CREATED
rancher-server-cluster      us-west-2   True
```

#### 5. Install an Ingress

The cluster needs an Ingress so that Rancher can be accessed from outside the cluster. Installing an Ingress requires allocating a public IP address. Ensure you have sufficient quota, otherwise it will fail to assign the IP address. Limits for public IP addresses are applicable at a regional level per subscription. You can use a managed ingress controller provided by AWS (ALB) or a third-party ingress controller like Traefik.



It is not recommended to install a third-party ingress controller, like Traefik, if a managed ingress controller is already being used.



**Ingress-NGINX EOL:** The community `ingress-nginx` controller reaches End-of-Life (EOL) in March 2026. This page uses Traefik, which is the recommended migration path for Rancher environments.

Traefik includes a native Ingress NGINX provider. This allows you to migrate from NGINX without rewriting your existing Ingress objects, as Traefik will automatically interpret `nginx.ingress.kubernetes.io` annotations. If you are upgrading a cluster that is already using `ingress-nginx`, follow this guide [https://doc.traefik.io/traefik/migrate/nginx-to-traefik/] for more information.

To install Traefik, we highly recommend using the Traefik application in the Rancher Prime Application Collection [https://apps.rancher.io/applications/traefik-proxy]. Check their documentation to integrate with Rancher Manager [https://docs.apps.rancher.io/howto-guides/integrate-with-rancher-manager].

#### 6. Get Load Balancer IP

To get the address of the load balancer, run:

```
kubectl get service traefik --namespace=traefik
```

The result should look similar to the following:

```
NAME                      TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)
 AGE
traefik  LoadBalancer   10.0.116.18    40.31.180.83
80:31229/TCP,443:31050/TCP 27m
```

Save the `EXTERNAL-IP`.

#### 7. Set up DNS

External traffic to the Rancher server will need to be directed at the load balancer you created.

Set up a DNS to point at the external IP that you saved. This DNS will be used as the Rancher server URL.

There are many valid ways to set up the DNS. For help, refer to the AWS documentation on routing traffic to an ELB load balancer. [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer.html]

#### 8. Install the Rancher Helm Chart

Next, install the Rancher Helm chart by following the instructions on this page. The Helm instructions are the same for installing Rancher on any Kubernetes distribution.

Use that DNS name from the previous step as the Rancher server URL when you install Rancher. It can be passed in as a Helm option. For example, if the DNS name is `rancher.my.org`, you could run the Helm installation command with the option `--set hostname=rancher.my.org`.

When installing Rancher on top of this setup, you will also need to pass the value below into the Rancher Helm install command in order to set the name of the ingress controller to be used with Rancher's ingress resource:

```
--set ingress.ingressClassName=traefik
```

Refer here for the Helm install command for your chosen certificate option.

## Installing SUSE Rancher Prime on a Google Kubernetes Engine Cluster

In this section, you'll learn how to install Rancher using Google Kubernetes Engine.

If you already have a GKE Kubernetes cluster, skip to the step about installing an ingress. Then install the Rancher Helm chart following the instructions on this page.

### Prerequisites

- You will need a Google account.
- You will need a Google Cloud billing account. You can manage your Cloud Billing accounts using the Google Cloud Console. For more information about the Cloud Console, visit General guide to the console. [https://support.google.com/cloud/answer/3465889?hl=en&ref_topic=3340599]

- You will need a cloud quota for at least one in-use IP address and at least 2 CPUs. For more details about hardware requirements for the Rancher server, refer to this section.

**1. Enable the Kubernetes Engine API**

Take the following steps to enable the Kubernetes Engine API:

1. Visit the Kubernetes Engine page [https://console.cloud.google.com/projectselector/kubernetes?_ga=2.169595943.767329331.1617810440-856599067.1617343886] in the Google Cloud Console.
2. Create or select a project.
3. Open the project and enable the Kubernetes Engine API for the project. Wait for the API and related services to be enabled. This can take several minutes.
4. Make sure that billing is enabled for your Cloud project. For information on how to enable billing for your project, refer to the Google Cloud documentation. [https://cloud.google.com/billing/docs/how-to/modify-project#enable_billing_for_a_project]

**2. Open the Cloud Shell**

Cloud Shell is a shell environment for managing resources hosted on Google Cloud. Cloud Shell comes preinstalled with the `gcloud` command-line tool and kubectl command-line tool. The `gcloud` tool provides the primary command-line interface for Google Cloud, and `kubectl` provides the primary command-line interface for running commands against Kubernetes clusters.

The following sections describe how to launch the cloud shell from the Google Cloud Console or from your local workstation.

##### Cloud Shell

To launch the shell from the Google Cloud Console, [https://console.cloud.google.com] go to the upper-right corner of the console and click the terminal button. When hovering over the button, it is labeled **Activate Cloud Shell**.

##### Local Shell

To install `gcloud` and `kubectl`, perform the following steps:

1. Install the Cloud SDK by following these steps. [https://cloud.google.com/sdk/docs/install] The Cloud SDK includes the `gcloud` command-line tool. The steps vary based on your OS.
2. After installing Cloud SDK, install the `kubectl` command-line tool by running the following command:

   ```
   gcloud components install kubectl
   ```

   In a later step, `kubectl` will be configured to use the new GKE cluster.
3. Install Helm 3 [https://helm.sh/docs/intro/install/] if it is not already installed.
4. Enable Helm experimental support for OCI images [https://github.com/helm/community/blob/master/hips/hip-0006.md] with the `HELM_EXPERIMENTAL_OCI` variable. Add the following line to `~/.bashrc` (or `~/.bash_profile` in macOS, or wherever your shell stores environment variables):

```
export HELM_EXPERIMENTAL_OCI=1
```

5. Run the following command to load your updated `.bashrc` file:

```
source ~/.bashrc
```

If you are running macOS, use this command:

```
source ~/.bash_profile
```

**3. Configure the gcloud CLI**

Set up default gcloud settings using one of the following methods:

- Using gcloud init, if you want to be walked through setting defaults.
- Using gcloud config, to individually set your project ID, zone, and region.

**Using gcloud init**

1. Run gcloud init and follow the directions:

```
gcloud init
```

If you are using SSH on a remote server, use the --console-only flag to prevent the command from launching a browser:

```
gcloud init --console-only
```

2. Follow the instructions to authorize gcloud to use your Google Cloud account and select the new project that you created.

**Using gcloud config**

**4. Confirm that gcloud is configured correctly**

Run:

```
gcloud config list
```

The output should resemble the following:

```
[compute]
```

```
region = us-west1 # Your chosen region
zone = us-west1-b # Your chosen zone
[core]
account = <Your email>
disable_usage_reporting = True
project = <Your project ID>

Your active configuration is: [default]
```

**5. Create a GKE Cluster**

The following command creates a three-node cluster.

Replace `cluster-name` with the name of your new cluster.

When choosing a Kubernetes version, be sure to first consult the support matrix [https://rancher.com/support-matrix/] to find the highest version of Kubernetes that has been validated for your Rancher version.

To successfully create a GKE cluster with Rancher, your GKE must be in Standard mode. GKE has two modes of operation when creating a Kubernetes cluster, Autopilot and Standard mode. The cluster configuration for Autopilot mode has restrictions on editing the kube-system namespace. However, Rancher needs to create resources in the kube-system namespace during installation. As a result, you will not be able to install Rancher on a GKE cluster created in Autopilot mode. For more information about the difference between GKE Autopilot mode and Standard mode, visit Compare GKE Autopilot and Standard. [https://cloud.google.com/kubernetes-engine/docs/resources/autopilot-standard-feature-comparison]

```
gcloud container clusters create cluster-name --num-nodes=3 --cluster
-version=<VERSION>
```

**6. Get Authentication Credentials**

After creating your cluster, you need to get authentication credentials to interact with the cluster:

```
gcloud container clusters get-credentials cluster-name
```

This command configures `kubectl` to use the cluster you created.

**7. Install an Ingress**

The cluster needs an Ingress so that Rancher can be accessed from outside the cluster. Installing an Ingress requires allocating a public IP address. Ensure you have sufficient quota, otherwise it will fail to assign the IP address. Limits for public IP addresses are applicable at a regional level per subscription. You can use a managed ingress controller provided by GCP or a third-party ingress controller like Traefik.



It is not recommended to install a third-party ingress controller, like Traefik, if a managed ingress controller is already being used

**Ingress-NGINX EOL:** The community `ingress-nginx` controller reaches End-of-Life (EOL) in March 2026. This page uses Traefik, which is the recommended migration path for Rancher environments.

Traefik includes a native Ingress NGINX provider. This allows you to migrate from NGINX without rewriting your existing Ingress objects, as Traefik will automatically interpret `nginx.ingress.kubernetes.io` annotations. If you are upgrading a cluster that is already using `ingress-nginx`, follow this guide [https://doc.traefik.io/traefik/migrate/nginx-to-traefik/] for more information.

To install Traefik, we highly recommend using the Traefik application in the Rancher Prime Application Collection [https://apps.rancher.io/applications/traefik-proxy]. Check their documentation to integrate with Rancher Manager [https://docs.apps.rancher.io/howto-guides/integrate-with-rancher-manager].

#### 8. Get Load Balancer IP

To get the address of the load balancer, run:

```
kubectl get service traefik --namespace=traefik
```

The result should look similar to the following:

```
NAME                      TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)
 AGE
traefik  LoadBalancer   10.0.116.18    40.31.180.83
80:31229/TCP,443:31050/TCP 27m
```

Save the `EXTERNAL-IP`.

#### 9. Set up DNS

External traffic to the Rancher server will need to be directed at the load balancer you created.

Set up a DNS to point at the external IP that you saved. This DNS will be used as the Rancher server URL.

There are many valid ways to set up the DNS. For help, refer to the Google Cloud documentation about managing DNS records. [https://cloud.google.com/dns/docs/records]

#### 10. Install the Rancher Helm chart

Next, install the Rancher Helm chart by following the instructions on this page. The Helm instructions are the same for installing Rancher on any Kubernetes distribution.

Use the DNS name from the previous step as the Rancher server URL when you install Rancher. It can be passed in as a Helm option. For example, if the DNS name is `rancher.my.org`, you could run the Helm installation command with the option `--set hostname=rancher.my.org`.

When installing Rancher on top of this setup, you will also need to set the name of the ingress controller to be used with Rancher's ingress resource:

```
--set ingress.ingressClassName=traefik
```

Refer here for the Helm install command for your chosen certificate option.

In Rancher v2.7.5, if you intend to use the default GKE ingress on your cluster without enabling VPC-native cluster mode, you need to set the following flag:

```
--set service.type=NodePort
```

This is necessary because of compatibility issues between this setup and ClusterIP, the default type for `cattle-system/rancher`.

## Cloud Marketplace Integration

Rancher offers integration with cloud marketplaces to easily purchase support for installations hosted on certain cloud providers. In addition, this integration also provides the ability to generate a supportconfig bundle which can be provided to rancher support.

This integration only supports AWS.

### AWS Marketplace Integration

#### Overview

Rancher offers an integration with the AWS Marketplace which allows users to purchase a support contract with SUSE. This integration allows you easily adjust your support needs as you start to support more clusters.

#### Limitations

- You must be running Rancher v2.6.7 or higher
- Rancher must be deployed with additional metrics enabled.
- Rancher must be installed on an EKS cluster.
- You must purchase at least one entitlement to Rancher support through AWS Marketplace.
- You may need additional setup to support proxy/airgap use cases. See the prerequisites for more information.

#### How to Use

1. Complete the prerequisite steps.
2. Install the CSP Adapter.

#### FAQ

**Can I purchase support for more nodes later on?**

Yes. Simply go to the AWS Marketplace entry that you used to initially purchase support and increase the number of entitlements.

### Can I use multiple instances of Rancher in the same AWS account?

Yes. However, each cluster that Rancher is installed in will need to adhere to the prerequisites.

In addition, keep in mind that a given entitlement can only be used by one Rancher management server at a time.

#### Prerequisites

##### 1. Setting Up License Manager and Purchasing Support

First, complete the first step [https://docs.aws.amazon.com/license-manager/latest/userguide/getting-started.html] of the license manager one-time setup. Next, go to the AWS Marketplace. Locate the "Rancher Premium Support Billing Container Starter Pack". Purchase at least one entitlement.

If you have installed Rancher using the "Rancher Setup" AWS Marketplace offering, skip to Step 4.



Each entitlement grants access to support for a certain amount of nodes. You can purchase more licenses as necessary later on.

##### 2. Create an EKS Cluster

Follow the Rancher docs to create an EKS cluster. When you get to the final step to install Rancher, **stop and return to this page**. This cluster will need to meet the following requirements:

- EKS version 1.22.
- Each node in the cluster has access to the registry containing Rancher and its related images.
- Each node in the cluster has access to the ECR repo storing the CSP Adapter.
- Each node in the cluster has access to the license manager service.
- Each node in the cluster has access to global endpoints for the STS service.

##### 3. Install Rancher

In addition to the options specified to install Rancher in the Rancher docs, you will also need to enable extra metrics. This can be done through the Helm CLI through the following options:

```
--set extraEnv\[0\].name="CATTLE_PROMETHEUS_METRICS" --set-string
extraEnv\[0\].value=true
```

You can also use a values.yaml like the below:

```
extraEnv:
  - name: "CATTLE_PROMETHEUS_METRICS"
    value: "true"
```

You will also need to install Rancher version 2.6.7 or higher.

#### 4. Create an OIDC Provider

Follow the AWS documentation [https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html] to create an OIDC provider for the cluster specified in the previous section.

#### 5. Create an IAM Role

An IAM role is required for the CSP adapter to check-in/check-out entitlements.

First, configure the trust policy as below. Replace `MY_AWS_ACC` with your AWS account number, `MY_AWS_REGION` with your AWS region, and `MY_OIDC_PROVIDER` with the id of your OIDC provider:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::${MY_AWS_ACC}:oidc-
provider/oidc.eks.${MY_AWS_REGION}.amazonaws.com/id/${MY_OIDC_PROVIDER}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {

"oidc.eks.${MY_AWS_REGION}.amazonaws.com/id/${MY_OIDC_PROVIDER}:sub":
"system:serviceaccount:cattle-csp-adapter-system:rancher-csp-adapter",

"oidc.eks.${MY_AWS_REGION}.amazonaws.com/id/${MY_OIDC_PROVIDER}:aud":
"sts.amazonaws.com"
                }
            }
        }
    ]
}
```

Next, use a policy for the role which has the following permissions:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RancherCSPAdapterPermissions",
            "Effect": "Allow",
            "Action": [
                "license-manager:ListReceivedLicenses",
```

```
                "license-manager:CheckoutLicense",
                "license-manager:ExtendLicenseConsumption",
                "license-manager:CheckInLicense",
                "license-manager:GetLicense",
                "license-manager:GetLicenseUsage"
            ],
            "Resource": "*"
        }
    ]
}
```

Save the name of the role. You will need it later on when installing the CSP adapter.

#### Installing the Adapter



If you are attempting to re-install the adapter, you may experience errant out-of-compliance messages for up to an hour.

#### Rancher vs. Adapter Compatibility Matrix



Different versions of the CSP adapter rely on features found in specific versions of Rancher. In order to deploy and run the adapter successfully, you need to ensure its version corresponds to the necessary version of Rancher.

| Rancher Version | Adapter Version |
|---|---|
| v2.15.3 | v110.0.0+up10.0.1 |
| v2.15.2 | v110.0.0+up10.0.0 |
| v2.15.1 | v110.0.0+up10.0.0 |
| v2.15.0 | v110.0.0+up10.0.0 |

#### 1. Gain Access to the Local Cluster



Only admin users should have access to the local cluster. Because the CSP adapter must be installed in the local cluster, this installation must be carried out by an admin user.

First, click on the local cluster and download a kubeconfig token. You can then configure your CLI to use this new token with the following command, replacing `$TOKEN_PATH` with the path on your filesystem to the downloaded token:

```
export KUBECONFIG=$TOKEN_PATH
```

#### 2. Create the Adapter Namespace

Create the namespace that the adapter will be installed in.

```
kubectl create ns cattle-csp-adapter-system
```

#### 3. Create Certificate Secrets

The adapter requires access to the root CA that Rancher is using to communicate with the Rancher server. You can read more about which certificate options Rancher supports in the chart options page.

If your Rancher install uses a certificate signed by a recognized Certificate Authority such as Let's Encrypt, then you can safely skip to Step 4.

However, if your Rancher install uses a custom certificate such as a Rancher-generated certificate or one signed by a private Certificate Authority, you will need to provide the certificate for this authority in PEM-encoded format so that the adapter can communicate with Rancher.

First, retrieve the certificate that Rancher is using and place in a file named `ca-additional.pem`. If you are using the Rancher-generated certs option, this can be done with the following command:

```
kubectl get secret tls-rancher -n cattle-system -o jsonpath=
"{.data.tls\.crt}" | base64 -d  >> ca-additional.pem
```

Then, create a secret which uses this cert:

```
kubectl -n cattle-csp-adapter-system create secret generic tls-ca-
additional --from-file=ca-additional.pem
```



Do not change the names of the file or of the created secret. Making changes to these values may result in errors when the adapter runs.

#### 4. Install the Chart

First, add the `rancher/charts` repo using the following command:

```
helm repo add rancher-charts https://charts.rancher.io
```

Next, install the CSP adapter. You must specify several values, including the account number, and the name of the role created in the prerequisites.

Ensure that you use the version of the CSP adapter that matches the version of Rancher that you are running, as defined above.

For the below instructions, replace `$MY_ACC_NUM` with your AWS account number and `$MY_ROLE_NAME` with the name of the role created in the prerequisites. In addition, replace `$CSP_ADAPTER_VERSION` with the version that matches your Rancher version in the version matrix.



If you use shell variables, do not specify quotation marks. For example, MY_ACC_NUM=123456789012 will work, but MY_ACC_NUM="123456789012" will fail.

Accounts using the AWS Marketplace listing for the EU and the UK will need to specify an additional `--set image.repository=rancher/rancher-csp-adapter-eu` option. To see if your account needs this option when installing the adapter, refer to the usage instructions of the marketplace listing.

**Let's Encrypt/ Public Certificate Authority**

```
  helm install rancher-csp-adapter rancher-charts/rancher-csp-
adapter --namespace cattle-csp-adapter-system --set aws.enabled
=true --set aws.roleName=$MY_ROLE_NAME --set-string
aws.accountNumber=$MY_ACC_NUM --version $CSP_ADAPTER_VERSION
```

Alternatively, you can use a `values.yaml` and specify options like below:

```
  aws:
    enabled: true
    accountNumber: "$MY_ACC_NUM"
    roleName: $MY_ROLE_NAME
```

The account number needs to be specified in a string format, like the above, or the installation will fail.

You can then install the adapter with the following command:

```
  helm install rancher-csp-adapter rancher-charts/rancher-csp-
adapter -f values.yaml --version $CSP_ADAPTER_VERSION
```

**Private CA Authority / Rancher-generated Certificates**

```
  helm install rancher-csp-adapter rancher-charts/rancher-csp-
adapter --namespace cattle-csp-adapter-system --set aws.enabled
=true --set aws.roleName=$MY_ROLE_NAME --set-string
aws.accountNumber=$MY_ACC_NUM --set additionalTrustedCAs=true
--version $CSP_ADAPTER_VERSION
```

Alternatively, you can use a `values.yaml` and specify options the below:

```
  aws:
    enabled: true
    accountNumber: "$MY_ACC_NUM"
    roleName: $MY_ROLE_NAME
  additionalTrustedCAs: true
```

The account number needs to be specified in a string format, like the above, or the installation will fail.

You can then install the adapter with the following command:

```
  helm install rancher-csp-adapter rancher-charts/rancher-csp-
adapter -f values.yaml --version $CSP_ADAPTER_VERSION
```

**5. Managing Certificate Updates**

If you had to create a secret storing a custom cert in Step 3, you will need to update this secret over time as the certificate is rotated.

First, delete the original secret in the cattle-csp-adapter-system namespace, using the below command:

```
kubectl delete secret tls-ca-additional -n cattle-csp-adapter-system
```

Then, follow the original installation steps in Step 3 to replace the content of the secret with the updated value.

Finally, restart the rancher-csp-adapter deployment to ensure that the updated value is made available to the adapter:

```
kubectl rollout restart deploy rancher-csp-adapter -n cattle-csp-
adapter-system
```

Methods such as cert-manager's trust operator [https://cert-manager.io/docs/projects/trust/] allow you to automate some of these tasks. Although these methods aren't officially supported, they can reduce how often you need to manually rotate certificates.

**Uninstalling The Adapter**

**1. Uninstall the adapter chart using helm.**

```
helm uninstall rancher-csp-adapter -n cattle-csp-adapter-system
```

**2. Remove the namespace created for the adapter.**

```
kubectl delete ns cattle-csp-adapter-system
```

**3. (Optional) remove any outstanding user notifications.**

```
kubectl delete RancherUserNotification csp-compliance
```

#### Common Issues

**After installing the adapter, a banner message appears in Rancher that says "AWS Marketplace Adapter: Unable to run the adapter, please check the adapter logs"**

This error indicates that while the adapter was installed into the cluster, an error has occurred which prevents it from properly checking-in/checking-out licenses.

This often occurs because the IAM role was not set up properly. Review the prerequisites and verify that:

- An OIDC provider has been created/associated with the cluster Rancher is running on.
- The IAM role has been configured to trust this OIDC provider.
- The IAM role has at least the permissions outlined in the policy.

If all of the above have been configured correctly, reach out to support for assistance.

**I see a banner message that states, "AWS Marketplace Adapter: You have exceeded your licensed node count. At least x more license(s) are required in AWS to become compliant"**

This message indicates that you do not have enough entitlements for the amount of nodes Rancher is currently managing.

Keep in mind the following limitations:

- Each entitlement is valid for a certain number of nodes.
- Every node currently managed by Rancher counts toward your usage total (with exception of nodes in the cluster rancher is installed on).
- Each entitlement can be used by at most one Rancher instance. For example, if you have two running Rancher instances in your account (each installed on a separate EKS cluster), then you will need at least two entitlements.

You may also have recently uninstalled/re-installed the adapter. If the adapter loses track of the licenses that it is currently managing, it can take up to an hour to resolve the actual state of the licenses.

### Supportconfig Bundle

After installing the CSP adapter, you will have the ability to generate a supportconfig bundle. This bundle is a tar file which can be used to quickly provide information to support.

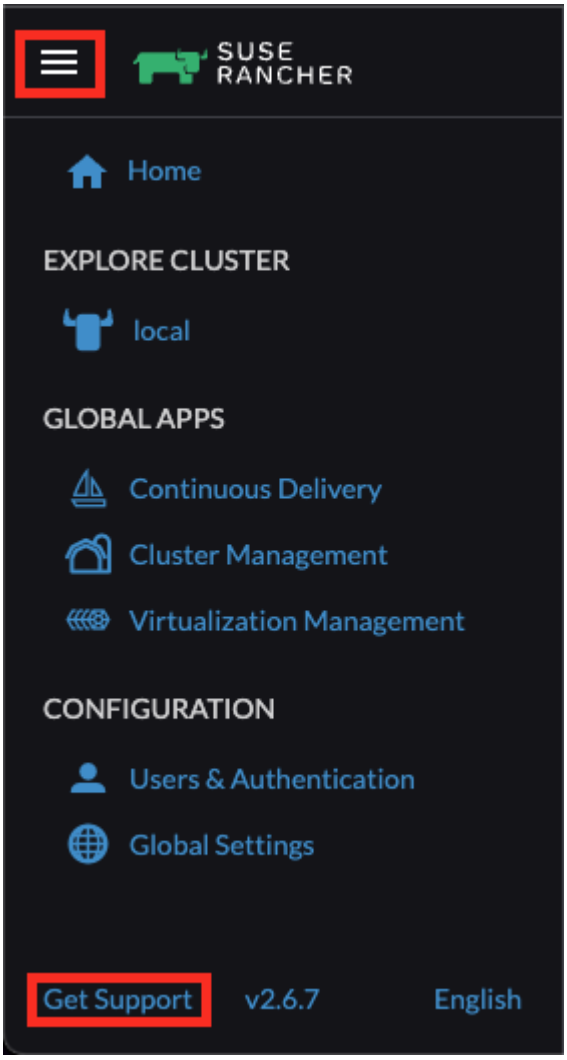
These bundles can be created through Rancher or through direct access to the cluster that Rancher is installed on. Note that accessing through Rancher is preferred.



Only admin users can generate/download supportconfig bundles, regardless of method.

#### Accessing Through Rancher

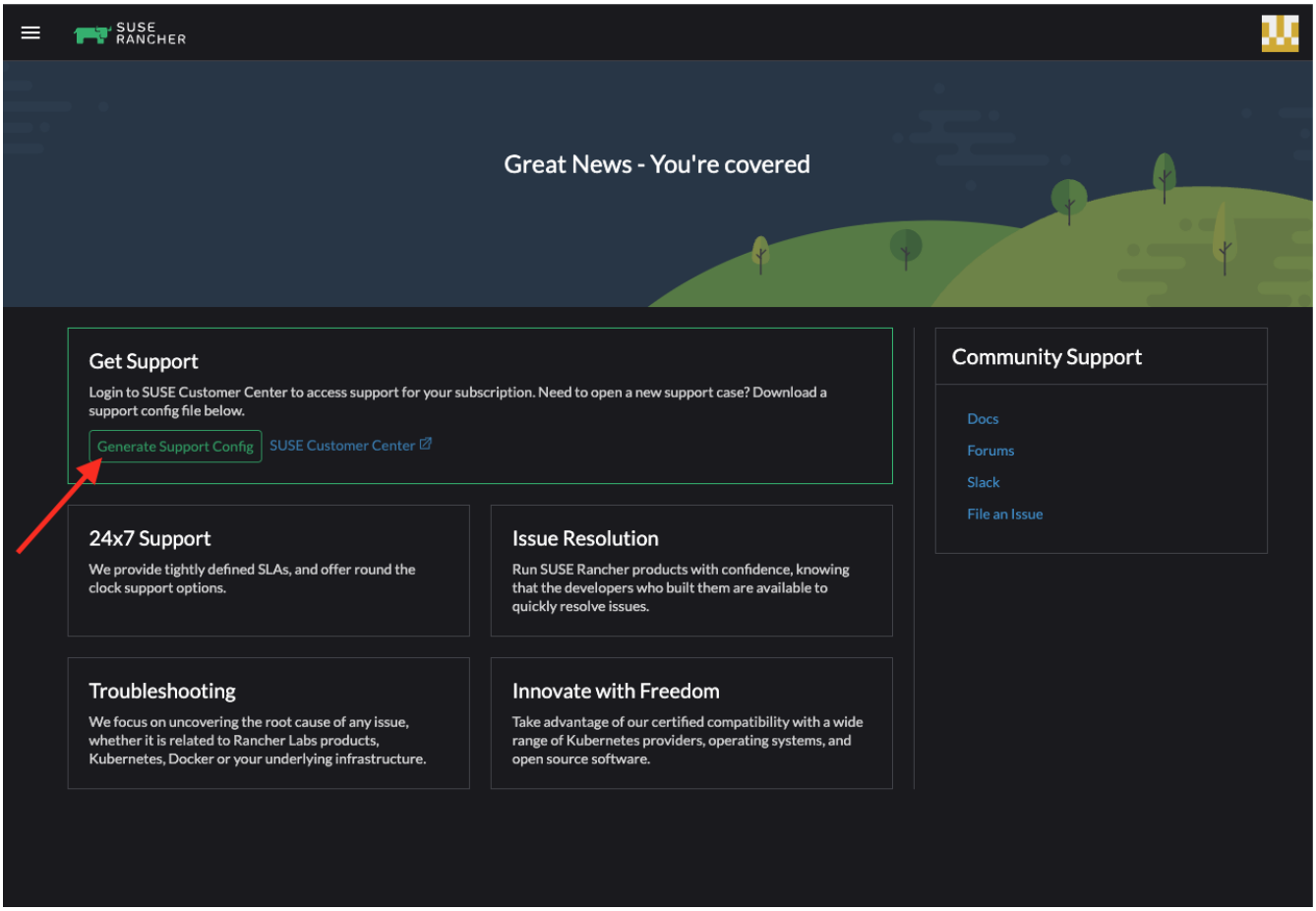
First, click on the hamburger menu. Then click the `Get Support` button.




In the next page, click on the `Generate Support Config` button.



If the adapter is not installed, the option to generate the supportconfig bundle will not be present. You must install the CSP adapter to generate a supportconfig bundle.




#### Accessing Without Rancher

First, generate a kubeconfig for the cluster that Rancher is installed on.



If Rancher is down, you will not be able to use a kubeconfig token generated by Rancher to access the cluster.

Configure your shell environment to use this kubeconfig token:

```
export KUBECONFIG=$MY_KUBECONFIG_PATH
```

It is recommended to create a temporary working directory while running this command, like below:

```
mkdir temp && cd temp
```

Then, retrieve the supportconfig bundle:

```
mkdir rancher && kubectl get configmap csp-config -n cattle-csp-adapter-system -o=jsonpath='{.data.data}' >> rancher/config.json && tar -c -f supportconfig_rancher.tar rancher && rm -rf rancher
```

This will create a `supportconfig_rancher.tar` file in your current directory.

Users who run these commands on Mac may experience issues due to incompatibilities between gnu-tar and bsd-tar. If support has issues reading a supportconfig that you produce, try using the below command after making gnu-tar accessible as `gtar` on your path:

```
mkdir rancher && kubectl get configmap csp-config -n cattle-csp-adapter-system -o=jsonpath='{.data.data}' >> rancher/config.json && gtar -c -f supportconfig_rancher.tar rancher && rm -rf rancher
```

# Other Installation Methods

## Air Gapped Installations

Follow these steps to install the Rancher server in an air gapped environment.

An air gapped environment could be where Rancher server will be installed offline, behind a firewall, or behind a proxy.

## Docker Installations

The xref:[single-node Docker installation] is for Rancher users that are wanting to test out Rancher. Instead of running on a Kubernetes cluster using Helm, you install the Rancher server component on a single node using a `docker run` command.

The Docker installation is for development and testing environments only.

Since there is only one node and a single Docker container, if the node goes down, there is no copy of the etcd data available on other nodes and you will lose all the data of your Rancher server.

The Rancher backup operator can be used to migrate Rancher from the single Docker container install to an installation on a high-availability Kubernetes cluster. For details, refer to the documentation on migrating Rancher to a new cluster.

## Air-Gapped Helm CLI Install

This section is about using the Helm CLI to install the Rancher server in an air gapped environment. An air gapped environment could be where Rancher server will be installed offline, behind a firewall, or behind a proxy.

For more information on each installation option, refer to this page.

Throughout the installation instructions, there will be *tabs* for each installation option.



If you install Rancher following the Docker installation guide, there is no upgrade path to transition your Docker Installation to a Kubernetes Installation.

**Installation Outline**

1. Set up infrastructure and private registry
2. Collect and publish images to your private registry

3. Set up a Kubernetes cluster
4. Install Rancher

**Upgrades**

To upgrade Rancher with Helm CLI in an air gap environment, follow this procedure.

**Next: Prepare your Node(s)**

**1. Set up Infrastructure and Private Registry**

This section describes how to:

- Provision the underlying infrastructure for a Rancher management server in an air-gapped environment.
- Setup the private container image registry required to distribute images to your Rancher nodes.

Although Rancher can be installed on any CNCF-certified Kubernetes cluster, the specific infrastructure requirements depend on whether you are deploying SUSE Rancher Prime on SUSE® Rancher Prime: K3s or SUSE® Rancher Prime: RKE2. For more information on the available installation options, refer to Installing SUSE Rancher Prime.

The following tabs provide infrastructure recommendations and tutorials for K3s and RKE2.

**K3s**

To install the Rancher management server on a high-availability K3s cluster, set up the following infrastructure:

- **Linux nodes:** Two Linux nodes (when using an external datastore) or three Linux nodes (when using embedded etcd), typically virtual machines hosted by your infrastructure provider.
- **Datastore:** Stores cluster data using either an external database (PostgreSQL, MySQL, or etcd) or an embedded etcd datastore.
- **Load balancer:** Directs incoming traffic across your nodes.
- **DNS record:** Maps an FQDN to the load balancer. Downstream Kubernetes clusters must be able to reach this Rancher server URL.
- **Private image registry:** Distributes container images to your nodes.

```
<h3><a class="anchor" id="_k3s_node_requirements"
href="#_k3s_node_requirements"></a>Node Requirements</h3>
```

Node requirements for a high-availability K3s cluster depend on your chosen datastore architecture:

- **External datastore:** Requires a minimum of two control plane nodes. Because state and quorum are managed by your external database (such as PostgreSQL or MySQL), the K3s control plane nodes remain stateless.
- **Embedded etcd:** Requires a minimum of three control plane nodes to maintain etcd quorum.

In an embedded etcd configuration, Rancher server data is stored in etcd across all three nodes. An etcd cluster requires an odd number of nodes to maintain quorum and elect a leader. If etcd loses quorum, a split-brain condition occurs, which can require restoring the cluster from a backup. In a

three-node setup, if one node fails, the remaining two nodes retain a majority and continue operating without service disruption

<h3><a class="anchor" id="_k3s_1_set_up_linux_nodes" href="#_k3s_1_set_up_linux_nodes"></a>1. Set up Linux Nodes</h3>

These hosts are disconnected from the internet, but must be able to access your private registry.

Ensure that your nodes meet the general installation requirements for operating systems, container runtimes, hardware, and networking.

For an example of setting up Linux nodes as Amazon EC2 instances, see Setting Up Nodes in Amazon EC2.

<h3><a class="anchor" id="_k3s_2_set_up_external_datastore" href="#_k3s_2_set_up_external_datastore"></a>2. Set up External Datastore</h3>

K3s supports datastores other than etcd, providing flexibility to select a database that fits your environment.

For a high-availability K3s installation using an external datastore, set up one of the following databases:

- PostgreSQL [https://www.postgresql.org/] (certified for versions 10.7 and 11.5)
- MySQL [https://www.mysql.com/] (certified for version 5.7)
- etcd [https://etcd.io/] (certified for version 3.3.15)

During K3s installation, pass the database connection parameters to K3s.

For an example of setting up a MySQL database on Amazon RDS, see MySQL Database in Amazon RDS.

For a complete list of datastore configuration options, see the SUSE® Rancher Prime: K3s Datastore Documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/datastore/datastore.html].

<h3><a class="anchor" id="_k3s_3_set_up_the_load_balancer" href="#_k3s_3_set_up_the_load_balancer"></a>3. Set up the Load Balancer</h3>

Set up a load balancer to direct traffic to the Rancher replicas on both nodes. This prevents a single node outage from interrupting communication with the Rancher management server.

When you set up Kubernetes, K3s deploys a Traefik Ingress controller. This controller listens on ports 80 and 443 of the worker nodes and handles traffic destined for specific hostnames.

When you install Rancher, the system creates an Ingress resource. This resource instructs the Traefik Ingress controller to listen for traffic destined for the Rancher hostname. When the Traefik Ingress controller receives traffic for the Rancher hostname, it forwards the traffic to the running Rancher pods in the cluster.

Determine whether to use a Layer 4 or Layer 7 load balancer for your deployment:

- **Layer 4 load balancer:** Forwards raw TCP traffic to your nodes. Layer 4 load balancing is recommended. Configure the load balancer to forward traffic to ports TCP/80 and TCP/443 on the Rancher management cluster nodes. The cluster Ingress controller redirects HTTP traffic to HTTPS, terminates TLS on port TCP/443, and forwards HTTP traffic on port TCP/80 to the Rancher Ingress pod.
- **Layer 7 load balancer:** Provides advanced routing features, such as centralizing TLS termination at the load balancer or making routing decisions based on HTTP attributes. If terminating TLS at a Layer 7 load balancer, use the `--set tls=external` option during Rancher installation. For details, see Rancher Helm Chart Options.

For load balancer configuration examples, see:

- NGINX Load Balancer Setup
- Amazon ELB Load Balancer Setup



Do not use the `local` cluster load balancer or Ingress for applications other than Rancher. Sharing this Ingress with other workloads can cause WebSocket errors during configuration reloads. Dedicate the `local` cluster exclusively to Rancher.

```
<h3><a class="anchor" id="_k3s_4_set_up_the_dns_record"
href="#_k3s_4_set_up_the_dns_record"></a>4. Set up the DNS
Record</h3>
```

After setting up the load balancer, create a DNS record to route traffic to it. Depending on your environment, create either a record pointing to the load balancer IP address or a CNAME record pointing to the load balancer hostname. Ensure that this record matches the exact hostname intended for the Rancher server.

Specify this hostname during Rancher installation. This hostname cannot be changed after installation.

For instructions on routing domain traffic to an Amazon ELB load balancer, see the AWS Route 53 Documentation [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer].

```
<h3><a class="anchor" id="_k3s_5_set_up_a_private_image_registry"
href="#_k3s_5_set_up_a_private_image_registry"></a>5. Set up a
Private Image Registry</h3>
```

Rancher supports air-gapped installations using a private registry. Ensure that you have a private registry or an alternative mechanism to distribute container images to your nodes.

During K3s cluster setup, create a url-k3s-private-registry-configuration[private registry configuration file] containing your registry configuration details.

To deploy a new private registry, see the documentation for your container runtime:

- Containerd CRI Registry Configuration [https://github.com/containerd/containerd/blob/main/docs/cri/config.md#registry-configuration]
- nerdctl Registry Documentation [https://github.com/containerd/nerdctl/blob/main/docs/registry.md]
- Docker Registry Deployment Guide [https://docs.docker.com/registry/deploying/]

**RKE2**

To install the Rancher management server on a high-availability RKE2 cluster, set up the following infrastructure:

- **Linux nodes:** Three Linux nodes, typically virtual machines hosted by an infrastructure provider such as Amazon EC2, Google Compute Engine, or VMware vSphere.
- **Load balancer:** Directs incoming traffic across all three nodes.
- **DNS record:** Maps an FQDN to the load balancer. Downstream Kubernetes clusters must be able to reach this Rancher server URL.
- **Private image registry:** Distributes container images to your nodes.

Deploy all nodes in the same region or data center. Nodes can be distributed across separate availability zones within that region.

<h3><a class="anchor" id="_rke2_node_requirements" href="#_rke2_node_requirements"></a>Node Requirements</h3>

In a high-availability RKE2 cluster, Rancher server data is stored in etcd across all three nodes.

An etcd cluster requires an odd number of nodes to maintain quorum and elect a leader. If etcd loses quorum, a split-brain condition occurs, which can require restoring the cluster from a backup. In a three-node setup, if one node fails, the remaining two nodes retain a majority and continue operating without service disruption.

<h3><a class="anchor" id="_rke2_1_set_up_linux_nodes" href="#_rke2_1_set_up_linux_nodes"></a>1. Set up Linux Nodes</h3>

These nodes are disconnected from the Internet, but must be able to access your private registry.

Ensure that your nodes meet the general installation requirements for operating systems, container runtimes, hardware, and networking.

For an example of setting up Linux nodes as Amazon EC2 instances, see Setting Up Nodes in Amazon EC2.

<h3><a class="anchor" id="_rke2_2_set_up_the_load_balancer" href="#_rke2_2_set_up_the_load_balancer"></a>2. Set up the Load Balancer</h3>

Set up a load balancer to distribute traffic to the Rancher replicas across all three nodes. This prevents a single node outage from interrupting access to the Rancher management server.

During cluster setup, RKE2 deploys an NGINX Ingress controller that listens on ports 80 and 443 of the worker nodes for incoming host-based traffic.

When you install Rancher, the installer creates an Ingress resource instructing the NGINX Ingress controller to route requests for the Rancher hostname directly to the Rancher pods in the cluster.

Determine whether to use a Layer 4 or Layer 7 load balancer for your deployment:

- **Layer 4 load balancer:** Forwards raw TCP traffic to your nodes. Layer 4 load balancing is

recommended. Configure the load balancer to forward traffic to ports TCP/80 and TCP/443 on the Rancher management cluster nodes. The cluster Ingress controller redirects HTTP traffic to HTTPS, terminates TLS on port TCP/443, and forwards HTTP traffic on port TCP/80 to the Rancher Ingress pod.

- **Layer 7 load balancer:** Provides advanced routing features, such as centralizing TLS termination at the load balancer or making routing decisions based on HTTP attributes. If terminating TLS at a Layer 7 load balancer, use the `--set tls=external` option during Rancher installation. For details, see Rancher Helm Chart Options.

For load balancer configuration examples, see:

- NGINX Load Balancer Setup
- Amazon ELB Load Balancer Setup



Do not use the `local` cluster load balancer or Ingress for applications other than Rancher. Sharing this Ingress with other workloads can cause WebSocket errors during configuration reloads. Dedicate the `local` cluster exclusively to Rancher.

```
<h3><a class="anchor" id="_rke2_3_set_up_the_dns_record"
href="#_rke2_3_set_up_the_dns_record"></a>3. Set up the DNS
Record</h3>
```

After setting up the load balancer, create a DNS record to route traffic to it. Depending on your environment, create either an A record pointing to the load balancer IP address or a CNAME record pointing to the load balancer hostname. Ensure that this record matches the exact hostname intended for the Rancher server.

Specify this hostname during Rancher installation. This hostname cannot be changed after installation.

For instructions on routing domain traffic to an Amazon ELB load balancer, see the AWS Route 53 Documentation [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer].

```
<h3><a class="anchor" id="_rke2_4_set_up_a_private_image_registry"
href="#_rke2_4_set_up_a_private_image_registry"></a>4. Set up a
Private Image Registry</h3>
```

Rancher supports air-gapped installations using a private registry. Ensure that you have a private registry or an alternative mechanism to distribute container images to your nodes.

During RKE2 cluster setup, create a url-rke2-private-registry-configuration[private registry configuration file] containing your registry details.

To deploy a new private registry, see the documentation for your container runtime:

- Containerd CRI Registry Configuration [https://github.com/containerd/containerd/blob/main/docs/cri/config.md#registry-configuration]
- nerdctl Registry Documentation [https://github.com/containerd/nerdctl/blob/main/docs/registry.md]
- Docker Registry Deployment Guide [https://docs.docker.com/registry/deploying/]

### 2. Collect and Publish Images to your Private Registry

This page outlines steps to collect and publish container images to a private registry for an air-gapped installation of SUSE Rancher Prime.



You can use any Rancher Prime version listed on the Prime Artifacts URL for the assets mentioned in these steps. To learn more about the Prime Artifacts URL, see our Prime-only documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#prime-artifacts-url]. Authentication is required. Use your SUSE Customer Center (SCC) [https://scc.suse.com/home] credentials to log in.

By default, all images used to provision Kubernetes clusters or launch any tools in Rancher, e.g. monitoring, pipelines, alerts, are pulled from Docker Hub. In an air gapped installation of Rancher, you need a private registry that is accessible by your Rancher server to load all the images.

Populating the private registry with images is the same process for installing Rancher with Docker and for installing Rancher on a Kubernetes cluster.

The steps in this section differ depending on whether or not you are planning to use Rancher to provision a downstream cluster with Windows nodes or not. By default, we provide the steps of how to populate your private registry assuming that Rancher will provision downstream Kubernetes clusters with only Linux nodes. But if you plan on provisioning any downstream Kubernetes clusters using Windows nodes, there are separate instructions to support the images needed.



*Prerequisites:*

You must have a private registry [https://docs.docker.com/registry/deploying/#run-an-externally-accessible-registry] available to use.

If the registry has certs, reference the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/private-registry.html] about adding a private registry. The certs and registry configuration files need to be mounted into the Rancher container.

**Linux Only Clusters**

For Rancher servers that will only provision Linux clusters, these are the steps to populate your private registry.

1. Find the required assets for your Rancher version
2. Collect the cert-manager image (unless you are bringing your own certificates or terminating TLS on a load balancer)
3. Save the images to your workstation
4. Populate the private registry

```
<h3><a id="_prerequisites"></a>Prerequisites</h3>
```

These steps expect you to use a Linux workstation that has internet access, access to your private registry, and at least 20 GB of disk space.

If you will use ARM64 hosts, the registry must support manifests. As of April 2020, Amazon Elastic Container Registry does not support manifests.



`rancher-images.txt` may include OCI Helm chart references (entries containing `rancher/charts/`). Docker can pull and push these natively only when the daemon is configured to use the containerd image store snapshotter [https://docs.docker.com/engine/storage/containerd/#enable-containerd-image-store-on-docker-engine]. If your Docker daemon does not use the containerd snapshotter, install the Helm CLI [https://helm.sh/docs/intro/install/] (v3.8.0 or later) on your workstation. In that case, `rancher-save-images.sh` and `rancher-load-images.sh` automatically detect the missing snapshotter and fall back to `helm pull`/`helm push` for the OCI Helm charts, storing them in a `./rancher-oci-charts/` directory alongside `rancher-images.tar.gz`. If neither the containerd snapshotter nor the Helm CLI in a required version is available, the scripts exit with an error.

<h3><a id="_1_find_the_required_assets_for_your_rancher_version"></a>1. Find the required assets for your Rancher version</h3>

1. Go to the Prime Artifacts URL [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#prime-artifacts-url] and find the Rancher v2.x.x release that you want to install, and click **show**.
2. To install Rancher in an air-gapped environment, you must download the following files:

| Release File | Description |
|---|---|
| `rancher-images.txt` | This file contains a list of images needed to install Rancher, provision clusters and user Rancher tools. |
| `rancher-save-images.sh` | This script pulls all the images in the `rancher-images.txt` from Docker Hub and saves all of the images as `rancher-images.tar.gz`. |
| `rancher-load-images.sh` | This script loads images from the `rancher-images.tar.gz` file and pushes them to your private registry. |

<h3><a class="anchor" id="_2_collect_the_cert_manager_image" href="#_2_collect_the_cert_manager_image"></a>2. Collect the cert-manager image</h3>



Skip this step if you are using your own certificates, or if you are terminating TLS on an external load balancer.

In a Kubernetes Install, if you elect to use the Rancher default self-signed TLS certificates, you must add the `cert-manager` [https://artifacthub.io/packages/helm/cert-manager/cert-manager] image to `rancher-images.txt` as well.

1. Fetch the latest `cert-manager` Helm chart and parse the template for image details:



Recent changes to cert-manager require an upgrade. If you are upgrading Rancher and using a version of cert-manager older than v0.12.0, please see our upgrade documentation.

```
helm repo add jetstack https://charts.jetstack.io
helm repo update
helm fetch jetstack/cert-manager
helm template ./cert-manager-<version>.tgz | awk '$1 ~ /image:/
{print $2}' | sed s/\"//g >> ./rancher-images.txt
```

2. Sort and unique the images list to remove any overlap between the sources:

```
sort -u rancher-images.txt -o rancher-images.txt
```

<h3><a id="_3_save_the_images_to_your_workstation"></a>3. Save the images to your workstation</h3>

1. **(Optional)** Verify the image list before pulling:

```
wc -l rancher-images.txt
head rancher-images.txt
```

2. Make `rancher-save-images.sh` an executable:

```
chmod +x rancher-save-images.sh
```

3. Run `rancher-save-images.sh` with the `rancher-images.txt` image list to create a tarball of all the required images:

```
./rancher-save-images.sh --image-list ./rancher-images.txt
```

4. **(Optional)** Specify a custom output file:

```
./rancher-save-images.sh \
  --image-list ./rancher-images.txt \
  --images rancher-images-custom.tar.gz
```

   **Result:** Docker begins pulling the images used for an air-gap install. Be patient. This process takes a few minutes. If the image list contains OCI Helm charts and the Helm CLI fallback is used, the charts are saved to a `./rancher-oci-charts/` directory instead of the tarball.

5. Verify that the tarball was created:

```
ls -lh rancher-images.tar.gz
```

If some images fail to pull, review the output and retry after resolving any issues.

<h3><a id="_linux_4_populate_the_private_registry"></a>4. Populate the private registry</h3>

Next, you will move the images in the `rancher-images.tar.gz` to your private registry using the scripts to load the images.

Move the images in the `rancher-images.tar.gz` to your private registry using the scripts to load the images.

The `rancher-images.txt` is expected to be on the workstation in the same directory that you are running the `rancher-load-images.sh` script. The `rancher-images.tar.gz` should also be in the same directory. If a `./rancher-oci-charts/` directory was created when saving the images, transfer it to this workstation as well and place it in the same directory before running `rancher-load-images.sh`. The `./rancher-oci-charts/` directory can be safely removed after the script finishes.

1. Log into your private registry if required:

```
docker login <REGISTRY.YOURDOMAIN.COM:PORT>
```

2. Make `rancher-load-images.sh` an executable:

```
chmod +x rancher-load-images.sh
```

3. Use `rancher-load-images.sh` to extract, tag and push `rancher-images.txt` and `rancher-images.tar.gz` to your private registry:

```
./rancher-load-images.sh --image-list ./rancher-images.txt
--registry <REGISTRY.YOURDOMAIN.COM:PORT>
```

**Linux and Windows Clusters**

For Rancher servers that will provision Linux and Windows clusters, there are distinctive steps to populate your private registry for the Windows images and the Linux images. Since a Windows cluster is a mix of Linux and Windows nodes, the Linux images pushed into the private registry are manifests.

<h2><a id="_windows_steps"></a>Windows Steps</h2>

The Windows images need to be collected and pushed from a Windows server workstation.

1. Find the required assets for your Rancher version
2. Save the images to your Windows Server workstation
3. Prepare the Docker daemon
4. Populate the private registry

<h3><a id="_prerequisites"></a>Prerequisites</h3>

These steps expect you to use a Windows Server 1809 workstation that has internet access, access to your private registry, and at least 50 GB of disk space.

The workstation must have Docker 18.02+ in order to support manifests, which are required when provisioning Windows clusters.

Your registry must support manifests. As of April 2020, Amazon Elastic Container Registry does not support manifests.

<h3><a id="_1_find_the_required_assets_for_your_rancher_version"></a>1. Find the required assets for your Rancher version</h3>

1. Go to the Prime Artifacts URL [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#prime-artifacts-url] and find the Rancher v2.x.x release that you want to install, and click **show**.
2. To install Rancher in an air-gapped environment, you must download the following files:

| Release File | Description |
|---|---|
| `rancher-windows-images.txt` | This file contains a list of Windows images needed to provision Windows clusters. |
| `rancher-save-images.psl` | This script pulls all the images in the `rancher-windows-images.txt` from Docker Hub and saves all of the images as `rancher-windows-images.tar.gz`. |
| `rancher-load-images.psl` | This script loads the images from the `rancher-windows-images.tar.gz` file and pushes them to your private registry. |

<h3><a id="_2_save_the_images_to_your_windows_server_workstation"></a>2. Save the images to your Windows Server workstation</h3>

1. Using `powershell`, go to the directory that has the files that were downloaded in the previous step.
2. Run `rancher-save-images.psl` to create a tarball of all the required images:

```
./rancher-save-images.psl
```

**Result:** Docker begins pulling the images used for an air-gap install. Be patient. This process takes a few minutes. When the process completes, your current directory will output a tarball named `rancher-windows-images.tar.gz`. Check that the output is in the directory.

<h3><a id="_3_prepare_the_docker_daemon"></a>3. Prepare the Docker daemon</h3>

Append your private registry address to the `allow-nondistributable-artifacts` config field

in the Docker daemon (`C:\ProgramData\Docker\config\daemon.json`). Since the base image of Windows images are maintained by the `mcr.microsoft.com` registry, this step is required as the layers in the Microsoft registry are missing from Docker Hub and need to be pulled into the private registry.

```
{
  ...
  "allow-nondistributable-artifacts": [
    ...
    "<REGISTRY.YOURDOMAIN.COM:PORT>"
  ]
  ...
}
```

<h3><a id="_win_4_populate_the_private_registry"></a>4. Populate the private registry</h3>

Move the images in the `rancher-windows-images.tar.gz` to your private registry using the scripts to load the images.

The `rancher-windows-images.txt` is expected to be on the workstation in the same directory that you are running the `rancher-load-images.psl` script. The `rancher-windows-images.tar.gz` should also be in the same directory.

1. Using `powershell`, log into your private registry if required:

   ```
   docker login <REGISTRY.YOURDOMAIN.COM:PORT>
   ```

2. Using `powershell`, use `rancher-load-images.psl` to extract, tag and push the images from `rancher-images.tar.gz` to your private registry:

   ```
   ./rancher-load-images.psl --registry
   <REGISTRY.YOURDOMAIN.COM:PORT>
   ```

<h2><a id="_linux_steps"></a>Linux Steps</h2>

The Linux images need to be collected and pushed from a Linux host, but *must be done after* populating the Windows images into the private registry. These step are different from the Linux only steps as the Linux images that are pushed will actually manifests that support Windows and Linux images.

<h3><a id="_prerequisites"></a>Prerequisites</h3>

You must populate the private registry with the Windows images before populating the private registry with Linux images. If you have already populated the registry with Linux images, you will need to follow these instructions again as they will publish manifests that support Windows and Linux images.

These steps expect you to use a Linux workstation that has internet access, access to your private registry, and at least 20 GB of disk space.

The workstation must have Docker 18.02+ in order to support manifests, which are required when provisioning Windows clusters.



`rancher-images.txt` may include OCI Helm chart references (entries containing `rancher/charts/`). Docker can pull and push these natively only when the daemon is configured to use the containerd image store snapshotter [https://docs.docker.com/engine/storage/containerd/#enable-containerd-image-store-on-docker-engine]. If your Docker daemon does not use the containerd snapshotter, install the Helm CLI [https://helm.sh/docs/intro/install/] (v3.8.0 or later) on your workstation. In that case, `rancher-save-images.sh` and `rancher-load-images.sh` automatically detect the missing snapshotter and fall back to `helm pull`/`helm push` for the OCI Helm charts, storing them in a `./rancher-oci-charts/` directory alongside `rancher-images.tar.gz`. If neither the containerd snapshotter nor the Helm CLI in a required version is available, the scripts exit with an error.

<h3><a id="_1_find_the_required_assets_for_your_rancher_version"></a>1. Find the required assets for your Rancher version</h3>

1. Browse to our releases page [https://github.com/rancher/rancher/releases] and find the Rancher v2.x.x release that you want to install. Don't download releases marked `rc` or `Pre-release`, as they are not stable for production environments. Click **Assets**.
2. From the release's **Assets** section, download the following files:

| Release File | Description |
|---|---|
| `rancher-images.txt` | This file contains a list of images needed to install Rancher, provision clusters and user Rancher tools. |
| `rancher-windows-images.txt` | This file contains a list of images needed to provision Windows clusters. |
| `rancher-save-images.sh` | This script pulls all the images in the `rancher-images.txt` from Docker Hub and saves all of the images as `rancher-images.tar.gz`. |
| `rancher-load-images.sh` | This script loads images from the `rancher-images.tar.gz` file and pushes them to your private registry. |

<h3><a id="_2_collect_all_the_required_images"></a>2. Collect all the required images</h3>

**For Kubernetes Installs using Rancher Generated Self-Signed Certificate:** In a Kubernetes Install, if you elect to use the Rancher default self-signed TLS certificates, you must add the `cert-manager` [https://artifacthub.io/packages/helm/cert-manager/cert-manager] image to `rancher-images.txt` as well. You skip this step if you are using you using your own certificates.

1. Fetch the latest `cert-manager` Helm chart and parse the template for image details:

Recent changes to cert-manager require an upgrade. If you are upgrading Rancher and using a version of cert-manager older than v0.12.0, please see our upgrade documentation.

```
helm repo add jetstack https://charts.jetstack.io
helm repo update
helm fetch jetstack/cert-manager
helm template ./cert-manager-<version>.tgz | awk '$1 ~ /image:/ {print $2}' | sed s/\"//g >> ./rancher-images.txt
```

2. Sort and unique the images list to remove any overlap between the sources:

```
sort -u rancher-images.txt -o rancher-images.txt
```

<h3><a id="_3_save_the_images_to_your_workstation"></a>3. Save the images to your workstation</h3>

1. Make `rancher-save-images.sh` an executable:

```
chmod +x rancher-save-images.sh
```

2. Run `rancher-save-images.sh` with the `rancher-images.txt` image list to create a tarball of all the required images:

```
./rancher-save-images.sh --image-list ./rancher-images.txt
```

**Result:** Docker begins pulling the images used for an air-gap install. Be patient. This process takes a few minutes. When the process completes, your current directory will output a tarball named `rancher-images.tar.gz`. Check that the output is in the directory. If the image list contains OCI Helm charts and the Helm CLI fallback is used, the charts are saved to a `./rancher-oci-charts/` directory instead of the tarball.

<h3><a id="_4_populate_the_private_registry"></a>4. Populate the private registry</h3>

Move the images in the `rancher-images.tar.gz` to your private registry using the `rancher-load-images.sh script` to load the images.

The image list, `rancher-images.txt` or `rancher-windows-images.txt`, is expected to be on the workstation in the same directory that you are running the `rancher-load-images.sh` script. The `rancher-images.tar.gz` should also be in the same directory. If a `./rancher-oci-charts/` directory was created when saving the images, transfer it to this workstation as well and place it in the same directory before running `rancher-load-images.sh`. The `./rancher-oci-`

`charts/` directory can be safely removed after the script finishes.

1. Log into your private registry if required:

```
docker login <REGISTRY.YOURDOMAIN.COM:PORT>
```

2. Make `rancher-load-images.sh` an executable:

```
chmod +x rancher-load-images.sh
```

3. Use `rancher-load-images.sh` to extract, tag and push the images from `rancher-images.tar.gz` to your private registry:

```
./rancher-load-images.sh --image-list ./rancher-images.txt \
   --windows-image-list ./rancher-windows-images.txt \
   --registry <REGISTRY.YOURDOMAIN.COM:PORT>
```

**Next step for Kubernetes Installs - Launch a Kubernetes Cluster**

### 3. Install Kubernetes

This section describes how to install a Kubernetes cluster according to our best practices for the Rancher server environment. This cluster should be dedicated to run only the Rancher server.

Rancher can be installed on any Kubernetes cluster, including hosted Kubernetes providers.

The steps to set up an air-gapped Kubernetes cluster on RKE2 or K3s are shown below.

**K3s**

In this guide, we are assuming you have created your nodes in your air-gapped environment and have a secure private container registry (or OCI-compliant registry) on your bastion server.

<h2><a id="_1_prepare_images_directory"></a>1. Prepare Images Directory</h2>

Obtain the images tar file for your architecture from the releases [https://github.com/k3s-io/k3s/releases] page for the version of K3s you will be running.

Place the tar file in the `images` directory before starting K3s on each node, for example:

```
sudo mkdir -p /var/lib/rancher/k3s/agent/images/
sudo cp ./k3s-airgap-images-$ARCH.tar
/var/lib/rancher/k3s/agent/images/
```

<h2><a id="_2_create_registry_yaml"></a>2. Create Registry YAML</h2>

Create the registries.yaml file at `/etc/rancher/k3s/registries.yaml`. This will tell K3s the necessary details to connect to your private registry.

The registries.yaml file should look like this before plugging in the necessary information:

```
---
mirrors:
  customreg:
    endpoint:
      - "https://ip-to-server:5000"
configs:
  customreg:
    auth:
      username: xxxxxx # this is the registry username
      password: xxxxxx # this is the registry password
    tls:
      cert_file: <path to the cert file used in the registry>
      key_file:  <path to the key file used in the registry>
      ca_file: <path to the ca file used in the registry>
```

Note, at this time only secure registries are supported with K3s (SSL with custom CA).

For more information on private registries configuration file for K3s, refer to the K3s documentation. [https://rancher.com/docs/k3s/latest/en/installation/private-registry/]



The `registries.yaml` file configures registry credentials at the node level for K3s and RKE2. If you install Rancher on another Kubernetes distribution, or cannot configure node-level registries, see Helm Chart Options for Air Gap Installations to configure credentials using the `system-default-regsitry` and `system-default-registry-pull-secrets` global settings instead.

<h2><a id="_3_install_k3s"></a>3. Install K3s</h2>

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the Rancher Support Matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/].

To specify the K3s (Kubernetes) version, use the INSTALL_K3S_VERSION (e.g., `INSTALL_K3S_VERSION="v1.24.10+k3s1"`) environment variable when running the K3s installation script.

Obtain the K3s binary from the releases [https://github.com/k3s-io/k3s/releases] page, matching the same version used to get the airgap images tar. Also obtain the K3s install script at https://get.k3s.io

Place the binary in `/usr/local/bin` on each node. Place the install script anywhere on each node, and name it `install.sh`.

Install K3s on each server:

```
INSTALL_K3S_SKIP_DOWNLOAD=true INSTALL_K3S_VERSION=<VERSION>
./install.sh
```

Install K3s on each agent:

```
INSTALL_K3S_SKIP_DOWNLOAD=true INSTALL_K3S_VERSION=<VERSION>
K3S_URL=https://<SERVER>:6443 K3S_TOKEN=<TOKEN> ./install.sh
```

Where `<SERVER>` is the IP or valid DNS of the server and `<TOKEN>` is the node-token from the server found at `/var/lib/rancher/k3s/server/node-token`.



K3s additionally provides a `--resolv-conf` flag for kubelets, which may help with configuring DNS in air-gapped networks.

`<h2><a id="_4_save_and_start_using_the_kubeconfig_file"></a>4. Save and Start Using the kubeconfig File</h2>`

When you installed K3s on each Rancher server node, a `kubeconfig` file was created on the node at `/etc/rancher/k3s/k3s.yaml`. This file contains credentials for full access to the cluster, and you should save this file in a secure location.

To use this `kubeconfig` file,

1. Install kubectl, [https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl] a Kubernetes command-line tool.
2. Copy the file at `/etc/rancher/k3s/k3s.yaml` and save it to the directory `~/.kube/config` on your local machine.
3. In the kubeconfig file, the `server` directive is defined as localhost. Configure the server as the DNS of your load balancer, referring to port 6443. (The Kubernetes API server will be reached at port 6443, while the Rancher server will be reached at ports 80 and 443.) Here is an example `k3s.yaml`:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [CERTIFICATE-DATA]
    server: [LOAD-BALANCER-DNS]:6443 # Edit this line
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
```

```
preferences: {}
users:
- name: default
  user:
    password: [PASSWORD]
    username: admin
```

**Result:** You can now use `kubectl` to manage your K3s cluster. If you have more than one kubeconfig file, you can specify which one you want to use by passing in the path to the file when using `kubectl`:

```
kubectl --kubeconfig ~/.kube/config/k3s.yaml get pods --all
-namespaces
```

For more information about the `kubeconfig` file, refer to the K3s documentation [https://rancher.com/docs/k3s/latest/en/cluster-access/] or the official Kubernetes documentation [https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/] about organizing cluster access using `kubeconfig` files.

<h2><a id="_note_on_upgrading"></a>Note on Upgrading</h2>

Upgrading an air-gap environment can be accomplished in the following manner:

1. Download the new air-gap images (tar file) from the releases [https://github.com/k3s-io/k3s/releases] page for the version of K3s you will be upgrading to. Place the tar in the `/var/lib/rancher/k3s/agent/images/` directory on each node. Delete the old tar file.
2. Copy and replace the old K3s binary in `/usr/local/bin` on each node. Copy over the install script at https://get.k3s.io (as it is possible it has changed since the last release). Run the script again just as you had done in the past with the same environment variables.
3. Restart the K3s service (if not restarted automatically by installer).

**RKE2**

In this guide, we are assuming you have created your nodes in your air-gapped environment and have a secure private container registry (or OCI-compliant registry) on your bastion server.

<h2><a id="_1_create_rke2_configuration"></a>1. Create RKE2 configuration</h2>

Create the config.yaml file at `/etc/rancher/rke2/config.yaml`. This will contain all the configuration options necessary to create a highly available RKE2 cluster.

On the first server the minimum config is:

```
token: my-shared-secret
tls-san:
  - loadbalancer-dns-domain.com
```

On each other server the config file should contain the same token and tell RKE2 to connect to the existing first server:

```
server: https://ip-of-first-server:9345
token: my-shared-secret
tls-san:
  - loadbalancer-dns-domain.com
```

For more information, refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/ha.html].



RKE2 additionally provides a `resolv-conf` option for kubelets, which may help with configuring DNS in air-gap networks.

<h2><a id="_2_create_registry_yaml"></a>2. Create Registry YAML</h2>

Create the registries.yaml file at `/etc/rancher/rke2/registries.yaml`. This will tell RKE2 the necessary details to connect to your private registry.

The registries.yaml file should look like this before plugging in the necessary information:

```
---
mirrors:
  customreg:
    endpoint:
      - "https://ip-to-server:5000"
configs:
  customreg:
    auth:
      username: xxxxxx # this is the registry username
      password: xxxxxx # this is the registry password
    tls:
      cert_file: <path to the cert file used in the registry>
      key_file:  <path to the key file used in the registry>
      ca_file: <path to the ca file used in the registry>
```

For more information on private registries configuration file for RKE2, refer to the RKE2 documentation. [https://documentation.suse.com/cloudnative/rke2/latest/en/install/containerd_registry_configuration.html]

<h2><a id="_3_install_rke2"></a>3. Install RKE2</h2>

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the support maintenance terms. [https://rancher.com/support-maintenance-terms/]

Download the install script, rke2, rke2-images, and sha256sum archives from the release and

upload them into a directory on each server:

```
mkdir /tmp/rke2-artifacts && cd /tmp/rke2-artifacts/
wget
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2
/rke2-images.linux-amd64.tar.zst
wget
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2
/rke2.linux-amd64.tar.gz
wget
https://github.com/rancher/rke2/releases/download/v1.21.5%2Brke2r2
/sha256sum-amd64.txt
curl -sfL https://get.rke2.io --output install.sh
```

Next, run install.sh using the directory on each server, as in the example below:

```
INSTALL_RKE2_ARTIFACT_PATH=/tmp/rke2-artifacts sh install.sh
```

Then enable and start the service on all servers:

` systemctl enable rke2-server.service systemctl start rke2-server.service `

For more information, refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/airgap.html].

<h2><a id="_4_save_and_start_using_the_kubeconfig_file"></a>4. Save and Start Using the kubeconfig File</h2>

When you installed RKE2 on each Rancher server node, a `kubeconfig` file was created on the node at `/etc/rancher/rke2/rke2.yaml`. This file contains credentials for full access to the cluster, and you should save this file in a secure location.

To use this `kubeconfig` file,

1. Install kubectl [https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl], a Kubernetes command-line tool.
2. Copy the file at `/etc/rancher/rke2/rke2.yaml` and save it to the directory `~/.kube/config` on your local machine.
3. In the kubeconfig file, the `server` directive is defined as localhost. Configure the server as the DNS of your load balancer, referring to port 6443. (The Kubernetes API server will be reached at port 6443, while the Rancher server will be reached at ports 80 and 443.) Here is an example `rke2.yaml`:

```
apiVersion: v1
clusters:
- cluster:
```

```
    certificate-authority-data: [CERTIFICATE-DATA]
    server: [LOAD-BALANCER-DNS]:6443 # Edit this line
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: default
  user:
    password: [PASSWORD]
    username: admin
```

**Result:** You can now use `kubectl` to manage your RKE2 cluster. If you have more than one kubeconfig file, you can specify which one you want to use by passing in the path to the file when using `kubectl`:

```
kubectl --kubeconfig ~/.kube/config/rke2.yaml get pods --all
-namespaces
```

For more information about the `kubeconfig` file, refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/cluster_access.html] or the official Kubernetes documentation [https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/] about organizing cluster access using `kubeconfig` files.

<h2><a id="_note_on_upgrading"></a>Note on Upgrading</h2>

Upgrading an air-gap environment can be accomplished in the following manner:

1. Download the new air-gap artifacts and install script from the releases [https://github.com/rancher/rke2/releases] page for the version of RKE2 you will be upgrading to.
2. Run the script again just as you had done in the past with the same environment variables.
3. Restart the RKE2 service.

#### Issues or Errors?

See the Troubleshooting page.

### 4. Install SUSE Rancher Prime

This section is about how to deploy Rancher for your air gapped environment in a high-availability Kubernetes installation. An air gapped environment could be where Rancher server will be installed offline,

behind a firewall, or behind a proxy.

#### Privileged Access for Rancher

When the Rancher server is deployed in the Docker container, a local Kubernetes cluster is installed within the container for Rancher to use. Because many features of Rancher run as deployments, and privileged mode is required to run containers within containers, you will need to install Rancher with the `--privileged` option.

#### Docker Instructions

If you want to continue the air gapped installation using Docker commands, skip the rest of this page and follow the instructions on xref:[this page.]

#### Kubernetes Instructions

Rancher recommends installing Rancher on a Kubernetes cluster. A highly available Kubernetes install is comprised of three nodes running the Rancher server components on a Kubernetes cluster. The persistence layer (etcd) is also replicated on these three nodes, providing redundancy and data duplication in case one of the nodes fails.

#### 1. Add the Helm Chart Repository

From a system that has access to the internet, fetch the latest Helm chart and copy the resulting manifests to a system that has access to the Rancher server cluster.

1. If you haven't already, install `helm` locally on a workstation that has internet access. Note: Refer to the Helm version requirements to choose a version of Helm to install Rancher.
2. Use the `helm repo add` command to add the Helm chart repository that contains charts to install Rancher Prime.

```
helm repo add rancher-prime <helm-chart-repo-url>
```



To learn more about the Rancher Prime Helm chart repository URL, see our Prime-only documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#chart-repo-url]. Authentication is required. Use your SUSE Customer Center (SCC) [https://scc.suse.com/] credentials to log in.

3. Fetch the SUSE Rancher Prime chart. This will pull down the chart and save it in the current directory as a `.tgz` file.
   - To fetch the latest version:

```
helm fetch rancher-prime/rancher
```

   - To fetch a specific version:
     a. Check to see which versions of Rancher are available.

```
helm search repo --versions rancher-prime
```

b. Fetch a specific version by specifying the `--version` parameter:

```
helm fetch rancher-prime/rancher --version=<version>
```

#### 2. Choose your SSL Configuration

Rancher Server is designed to be secure by default and requires SSL/TLS configuration.

When Rancher is installed on an air gapped Kubernetes cluster, there are two recommended options for the source of the certificate.



If you want to terminate SSL/TLS externally, see TLS termination on an External Load Balancer.

| Configuration | Chart option | Description | Requires cert-manager |
|---|---|---|---|
| Rancher Generated Self-Signed Certificates | `ingress.tls.source=rancher` | Use certificates issued by Rancher's generated CA (self signed)<br>This is the **default** and does not need to be added when rendering the Helm template. | yes |
| Certificates from Files | `ingress.tls.source=secret` | Use your own certificate files by creating Kubernetes Secret(s).<br>This option must be passed when rendering the Rancher Helm template. | no |

#### Helm Chart Options for Air Gap Installations

When setting up the Rancher Helm template, there are several options in the Helm chart that are designed specifically for air gap installations.

| Chart Option | Chart Value | Description |
|---|---|---|
| `certmanager.version` | `<version>` | Configure proper Rancher TLS issuer depending of running cert-manager version. |
| `systemDefaultRegistry` | `<REGISTRY.YOURDOMAIN.COM:PORT>` | Configure Rancher server to always pull from your private registry when provisioning clusters. |
| `systemRegistryInheritsPullSecrets` | `<boolean>` | Sets the default value of the `system-default-registry-pull-secrets` global setting to the image pull secrets used during Rancher installation. Use this option when node-level registry configuration is not possible and you want to propagate these credentials to all downstream clusters by default. |

| Chart Option | Chart Value | Description |
|---|---|---|
| `useBundledSystemChart` | `true` | Configure Rancher server to use the packaged copy of Helm charts. The charts repository contains all the catalog items required for features such as monitoring, logging, alerting, and global DNS. These Helm charts are located in the Rancher Charts GitHub repository [https://github.com/rancher/charts] or served via the main mirror at `https://git.rancher.io/charts`. In an air-gapped environment, using the bundled charts within Rancher is much easier than setting up a Git mirror. |

#### 3. Fetch the Cert-Manager Chart

Based on the choice your made in 2. Choose your SSL Configuration, complete one of the procedures below.

##### Option A: Default Self-Signed Certificate

##### Fetch from SUSE Application Collection

Pull the security-hardened cert-manager chart listed in the SUSE Application Collection [https://apps.rancher.io/applications/cert-manager] (AppCo) available to Prime users.

1. Log in to AppCo registry on an internet-connected machine:

   Use your enterprise credentials [https://docs.apps.rancher.io/get-started/authentication] to log in to the secure SUSE Application Collection registry:

   ```
   helm registry login dp.apps.rancher.io -u <username-or-service-account-username> -p <access-token-or-service-account-token>
   ```

2. Export and download the OCI chart archive:

   Use `helm pull` to download the chart directly from the OCI registry as a local tarball:

   ```
   helm pull oci://dp.apps.rancher.io/charts/cert-manager --version <cert-manager-version>
   ```

   This saves `cert-manager-<cert-manager-version>.tgz` to your current working directory.

3. Populate the Private Registry with AppCo Images.

   Before installing cert-manager inside the air-gapped cluster, you must pull the secure container images from SUSE on your internet-connected machine, save them, transfer them via secure media (like a USB drive), and push them to your local registry.

   The below command pulls the secure container images from SUSE:

```
docker pull dp.apps.rancher.io/cert-manager-controller:<cert-manager-
version>
docker pull dp.apps.rancher.io/cert-manager-cainjector:<cert-manager-
version>
docker pull dp.apps.rancher.io/cert-manager-webhook:<cert-manager-
version>
```

The below command saves, transfers, and retags the images for your internal air-gapped registry:

```
# Save to tarball
docker save -o cert-manager-appco.tar \
   dp.apps.rancher.io/cert-manager-controller:<cert-manager-version>
\
   dp.apps.rancher.io/cert-manager-cainjector:<cert-manager-version>
\
   dp.apps.rancher.io/cert-manager-webhook:<cert-manager-version>

# Transfer the tar file and chart .tgz file into the air-gapped
network

# Load on air-gapped machine
docker load -i cert-manager-appco.tar

# Retag for your internal registry (e.g., registry.internal.corp)
docker tag dp.apps.rancher.io/cert-manager-controller:<cert-manager-
version> <registry.internal.corp>/cert-manager-controller:<cert-
manager-version>
docker tag dp.apps.rancher.io/cert-manager-cainjector:<cert-manager-
version> <registry.internal.corp>/cert-manager-cainjector:<cert-
manager-version>
docker tag dp.apps.rancher.io/cert-manager-webhook:<cert-manager-
version> <registry.internal.corp>/cert-manager-webhook:<cert-manager-
version>

# Push to internal registry
docker push <registry.internal.corp>/cert-manager-controller:<cert-
manager-version>
docker push <registry.internal.corp>/cert-manager-cainjector:<cert-
manager-version>
docker push <registry.internal.corp>/cert-manager-webhook:<cert-
manager-version>
```

#### 4. Install Rancher

Copy the fetched charts to a system that has access to the Rancher server cluster to complete installation.

#### 1. Install cert-manager

Install cert-manager with the same options you would use to install the chart. Remember to set the `image.repository` option to pull the image from your private registry.



To see options on how to customize the cert-manager install (including for cases where your cluster uses PodSecurityPolicies), see the cert-manager docs [https://artifacthub.io/packages/helm/cert-manager/cert-manager#configuration].

*Click to expand - SUSE Application Collection cert-manager Installation Instructions*

1. Create the namespace.

```
kubectl create namespace cert-manager
```

2. Install using the local chart tarball file.

```
helm install cert-manager ./<cert-manager-version>.tgz \
  --namespace cert-manager \
  --set installCRDs=true \
  --set image.repository=<registry.internal.corp>/cert-manager-
controller \
  --set webhook.image.repository=<registry.internal.corp>/cert-
manager-webhook \
  --set cainjector.image.repository=<registry.internal.corp>/cert-
manager-cainjector
```

#### 2. Install Rancher

First, refer to Adding TLS Secrets to publish the certificate files so Rancher and the ingress controller can use them.

Then, create the namespace for Rancher using kubectl:

```
kubectl create namespace cattle-system
```

Next, install Rancher, declaring your chosen options. Use the reference table below to replace each placeholder. Rancher needs to be configured to use the private registry in order to provision any Rancher launched Kubernetes clusters or Rancher tools.

| Placeholder | Description |
|---|---|
| `<VERSION>` | The version number of the output tarball. |
| `<RANCHER.YOURDOMAIN.COM>` | The DNS name you pointed at your load balancer. |

| Placeholder | Description |
| --- | --- |
| <REGISTRY.YOURDOMAIN.COM:PORT> | The DNS name for your private registry. |
| <CERTMANAGER_VERSION> | Cert-manager version running on k8s cluster. |

```
   helm install rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set certmanager.version=<CERTMANAGER_VERSION> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

**Optional**: To install a specific Rancher version, set the `image.tag` value, example: `--set image.tag=v2.15.3`

#### Option B: Certificates From Files Using Kubernetes Secrets

##### 1. Create Secrets

Create Kubernetes secrets from your own certificates for Rancher to use. The common name for the cert will need to match the `hostname` option in the command below, or the ingress controller will fail to provision the site for Rancher.

##### 2. Install Rancher

Install Rancher, declaring your chosen options. Use the reference table below to replace each placeholder. Rancher needs to be configured to use the private registry in order to provision any Rancher launched Kubernetes clusters or Rancher tools.

| Placeholder | Description |
| --- | --- |
| <VERSION> | The version number of the output tarball. |
| <RANCHER.YOURDOMAIN.COM> | The DNS name you pointed at your load balancer. |
| <REGISTRY.YOURDOMAIN.COM:PORT> | The DNS name for your private registry. |

```
   helm install rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set ingress.tls.source=secret \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

If you are using a Private CA signed cert, add `--set privateCA=true` following `--set ingress.tls.source=secret`:

```
   helm install rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set ingress.tls.source=secret \
    --set privateCA=true \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

The installation is complete.

#### Optional: Install Rancher with an Authenticated Default Registry

Starting in Rancher v2.15.0, Rancher supports an authenticated `systemDefaultRegistry` using Kubernetes image pull secrets. This configuration differs from the top-level `imagePullSecrets` and `image.repository` values, which apply only to the primary Rancher deployment. For details on working with global registries and pull secrets across local and downstream clusters, see Authenticated Private Registries.

#### Using an Authenticated Global Registry with Pull Secrets

If your private registry requires authentication, you can configure the `system-default-registry` and `system-default-registry-pull-secrets` settings. Use this option when you cannot configure node-level credentials or want to distribute credentials to downstream clusters automatically. This option configures the local cluster and all downstream clusters to pull from the authenticated registry by default.

You can configure default pull secrets in two ways when installing Rancher with Helm:

**Configuration 1: Using the `systemRegistryInheritsPullSecrets` Helm option**

When set to `true`, Rancher sets the default value of the `system-default-registry-pull-secrets` global setting to the image pull secrets used during the Rancher installation. You can edit this setting after installation in **Global Settings** in the Rancher UI.

```
helm install rancher ./rancher-<VERSION>.tgz \
  --namespace cattle-system \
  --set hostname=<RANCHER.YOURDOMAIN.COM> \
  --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \
  --set systemRegistryInheritsPullSecrets=true \
  --set useBundledSystemChart=true
```

**Configuration 2: Using the `CATTLE_BASE_REGISTRY_PULL_SECRETS` environment variable**

Use this environment variable to set the default value of the `system-default-registry-pull-secrets` global setting to specific pull secrets in the `cattle-system` namespace. You can edit this setting after installation in **Global Settings** in the Rancher UI.

```
helm install rancher ./rancher-<VERSION>.tgz \
  --namespace cattle-system \
  --set hostname=<RANCHER.YOURDOMAIN.COM> \
  --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \
  --set extraEnv[0].name=CATTLE_BASE_REGISTRY_PULL_SECRETS \
  --set extraEnv[0].value='<SECRET1>\,<SECRET2>' \
  --set useBundledSystemChart=true
```



These configurations set the default values for the `system-default-registry` and `system-default-registry-pull-secrets` settings. To clear these settings, you must remove the associated Helm value (such as `systemDefaultRegistry`) or environment variable (such as `CATTLE_BASE_REGISTRY_PULL_SECRETS`) from your Helm installation.

#### Additional Resources

These resources could be helpful when installing Rancher:

- Importing and installing extensions in an air-gapped environment
- Rancher Helm chart options
- Adding TLS secrets
- Troubleshooting Rancher Kubernetes Installations

### Setting up Local System Charts for Air Gapped Installations

The Charts [https://github.com/rancher/charts] repository contains all the Helm catalog items required for features such as monitoring, logging, alerting and Istio.

In an air gapped installation of Rancher, you will need to configure Rancher to use a local copy of the system charts. This section describes how to use local system charts using a CLI flag.

#### Using Local System Charts

A local copy of `system-charts` has been packaged into the `rancher/rancher` container. To be able to use these features in an air gap install, you will need to run the Rancher install command with an extra environment variable, `CATTLE_SYSTEM_CATALOG=bundled`, which tells Rancher to use the local copy of the charts instead of attempting to fetch them from GitHub.

Example commands for a Rancher installation with a bundled `system-charts` are included in the air gap installation instructions for Docker and Helm installs.

### Upgrading in an Air-Gapped Environment



These instructions assume you have already followed the instructions for a Kubernetes

> upgrade on this page, including the prerequisites, up until step 3. Upgrade Rancher.

#### Rancher Helm Upgrade Options

To upgrade with Helm, apply the same options that you used when installing Rancher. Refer to the reference table below to replace each placeholder. Rancher needs to be configured to use the private registry in order to provision any Rancher launched Kubernetes clusters or Rancher tools.

Based on the choice you made during installation, complete one of the procedures below.

| Placeholder | Description |
|---|---|
| `<VERSION>` | The version number of the output tarball. |
| `<RANCHER.YOURDOMAIN.COM>` | The DNS name you pointed at your load balancer. |
| `<REGISTRY.YOURDOMAIN.COM:PORT>` | The DNS name for your private registry. |
| `<CERTMANAGER_VERSION>` | Cert-manager version running on k8s cluster. |

#### Option A: Default Self-signed Certificate

```
helm upgrade rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set certmanager.version=<CERTMANAGER_VERSION> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

#### Resolving UPGRADE FAILED Error

If you encounter the error message, `Error: UPGRADE FAILED: "rancher" has no deployed releases`, Rancher might have been installed via the `helm template` command. To successfully upgrade Rancher, use the following command instead:

```
helm template rancher ./rancher-<VERSION>.tgz --output-dir . \
    --no-hooks \ # prevent files for Helm hooks from being generated
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set certmanager.version=<CERTMANAGER_VERSION> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

After you run the Helm command, apply the rendered template:

```
kubectl -n cattle-system apply -R -f ./rancher
```

**Option B: Certificates from Files using Kubernetes Secrets**

```
helm upgrade rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set ingress.tls.source=secret \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

If you are using a Private CA signed cert, add `--set privateCA=true` following `--set ingress.tls.source=secret`:

```
helm upgrade rancher ./rancher-<VERSION>.tgz \
    --namespace cattle-system \
    --set hostname=<RANCHER.YOURDOMAIN.COM> \
    --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
    --set ingress.tls.source=secret \
    --set privateCA=true \
    --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
    --set useBundledSystemChart=true # Use the packaged Rancher system
charts
```

**Verify the Upgrade**

Log into Rancher to confirm that the upgrade succeeded.



Having network issues following upgrade?

See xref:[Restoring Cluster Networking].

**Known Upgrade Issues**

A list of known issues for each Rancher version can be found in the release notes on GitHub [https://github.com/rancher/rancher/releases] and on the Rancher forums. [https://forums.rancher.com/c/announcements/12]

## Installing SUSE Rancher Prime behind an HTTP Proxy

In a lot of enterprise environments, servers or VMs running on premise do not have direct Internet access,

but must connect to external services through a HTTP(S) proxy for security reasons. This tutorial shows step by step how to set up a highly available Rancher installation in such an environment.

Alternatively, it is also possible to set up Rancher completely air-gapped without any Internet access. This process is described in detail in the Rancher docs.

**Installation Outline**

1. Set up infrastructure
2. Set up a Kubernetes cluster
3. Install Rancher

**1. Set up Infrastructure**

In this section, you will provision the underlying infrastructure for your Rancher management server with internet access through a HTTP proxy.

To install the Rancher management server on a high-availability RKE2/K3s cluster, we recommend setting up the following infrastructure:

- **Three Linux nodes,** typically virtual machines, in an infrastructure provider such as Amazon's EC2, Google Compute Engine, or vSphere.
- **A load balancer** to direct front-end traffic to the three nodes.
- **A DNS record** to map a URL to the load balancer. This will become the Rancher server URL, and downstream Kubernetes clusters will need to reach it.

These nodes must be in the same region/data center. You may place these servers in separate availability zones.

**Why Three Nodes?**

In an RKE2 or K3s cluster, SUSE Rancher Prime server data is stored in etcd, which runs on all three nodes. Achieving fault tolerance in a high-availability (HA) setup requires a minimum of three etcd nodes.

Expanding the cluster from three to four nodes increases the required quorum from two to three without improving fault tolerance (both configurations tolerate only a single node failure). A three-node cluster provides the baseline for high availability. If one node fails, the remaining two maintain a majority quorum ((N / 2) + 1) to elect a leader, as described in the [etcd documentation on cluster size](https://etcd.io/docs/v3.6/faq/#why-an-odd-number-of-cluster-members).

**1. Set up Linux Nodes**

These hosts will connect to the internet through an HTTP proxy.

Make sure that your nodes fulfill the general installation requirements for OS, container runtime, hardware, and networking.

For an example of one way to set up Linux nodes, refer to this tutorial for setting up nodes as instances in Amazon EC2.

#### 2. Set up the Load Balancer

You will also need to set up a load balancer to direct traffic to the Rancher replica on both nodes. That will prevent an outage of any single node from taking down communications to the Rancher management server.

When Kubernetes gets set up in a later step, the RKE2/K3s tool will deploy an NGINX Ingress controller. This controller will listen on ports 80 and 443 of the worker nodes, answering traffic destined for specific hostnames.

When Rancher is installed (also in a later step), the Rancher system creates an Ingress resource. That Ingress tells the NGINX Ingress controller to listen for traffic destined for the Rancher hostname. The NGINX Ingress controller, when receiving traffic destined for the Rancher hostname, will forward that traffic to the running Rancher pods in the cluster.

For your implementation, consider if you want or need to use a Layer-4 or Layer-7 load balancer:

- **A layer-4 load balancer** is the simpler of the two choices, in which you are forwarding TCP traffic to your nodes. We recommend configuring your load balancer as a Layer 4 balancer, forwarding traffic to ports TCP/80 and TCP/443 to the Rancher management cluster nodes. The Ingress controller on the cluster will redirect HTTP traffic to HTTPS and terminate SSL/TLS on port TCP/443. The Ingress controller will forward traffic to port TCP/80 to the Ingress pod in the Rancher deployment.
- **A layer-7 load balancer** is a bit more complicated but can offer features that you may want. For instance, a layer-7 load balancer is capable of handling TLS termination at the load balancer, as opposed to Rancher doing TLS termination itself. This can be beneficial if you want to centralize your TLS termination in your infrastructure. Layer-7 load balancing also offers the capability for your load balancer to make decisions based on HTTP attributes such as cookies, etc. that a layer-4 load balancer is not able to concern itself with. If you decide to terminate the SSL/TLS traffic on a layer-7 load balancer, you will need to use the `--set tls=external` option when installing Rancher in a later step. For more information, refer to the Rancher Helm chart options.

For an example showing how to set up an NGINX load balancer, refer to this page.

For a how-to guide for setting up an Amazon ELB Network Load Balancer, refer to this page.



Do not use this load balancer (i.e, the `local` cluster Ingress) to load balance applications other than Rancher following installation. Sharing this Ingress with other applications may result in websocket errors to Rancher following Ingress configuration reloads for other apps. We recommend dedicating the `local` cluster to Rancher and no other applications.

#### 3. Set up the DNS Record

Once you have set up your load balancer, you will need to create a DNS record to send traffic to this load balancer.

Depending on your environment, this may be an A record pointing to the LB IP, or it may be a CNAME pointing to the load balancer hostname. In either case, make sure this record is the hostname that you intend Rancher to respond on.

You will need to specify this hostname in a later step when you install Rancher, and it is not possible to change it later. Make sure that your decision is a final one.

For a how-to guide for setting up a DNS record to route domain traffic to an Amazon ELB load balancer, refer to the official AWS documentation. [https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-

load-balancer]

**Next: Set up a Kubernetes cluster**

### 2. Install Kubernetes

Once the infrastructure is ready, you can continue with setting up a Kubernetes cluster to install Rancher in.

The steps to set up RKE2 or K3s are shown below.

For convenience, export the IP address and port of your proxy into an environment variable and set up the `HTTP_PROXY` variables for your current shell on every node:



The `NO_PROXY` environment variable is not standardized, and the accepted format of the value can differ between applications. When configuring the `NO_PROXY` variable for Rancher, the value must adhere to the format expected by Golang.

Specifically, the value should be a comma-delimited string which only contains IP addresses, CIDR notation, domain names, or special DNS labels (e.g. `*`). For a full description of the expected value format, refer to the **upstream Golang documentation** [https://pkg.go.dev/golang.org/x/net/http/httpproxy#Config].

```
export proxy_host="10.0.0.5:8888"
export HTTP_PROXY=http://${proxy_host}
export HTTPS_PROXY=http://${proxy_host}
export NO_PROXY=127.0.0.0/8,10.0.0.0/8,cattle-
system.svc,172.16.0.0/12,192.168.0.0/16
```

**K3s**

First configure the HTTP proxy settings on the K3s systemd service, so that K3s's containerd can pull images through the proxy:

```
cat <<'EOF' | sudo tee /etc/default/k3s > /dev/null
HTTP_PROXY=http://${proxy_host}
HTTPS_PROXY=http://${proxy_host}
NO_PROXY=127.0.0.0/8,10.0.0.0/8,cattle-
system.svc,172.16.0.0/12,192.168.0.0/16,.svc,.cluster.local
EOF
```

Rancher needs to be installed on a supported Kubernetes version. To find out which versions of Kubernetes are supported for your Rancher version, refer to the Rancher Support Matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/].

To specify the K3s (Kubernetes) version, use the INSTALL_K3S_VERSION (e.g., `INSTALL_K3S_VERSION="v1.24.10+k3s1"`) environment variable when running the K3s installation script.

On the first node, create a new cluster:

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=<VERSION>
K3S_TOKEN=<TOKEN> sh -s - server --cluster-init
```

And then join the other nodes:

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION=<VERSION>
K3S_TOKEN=<TOKEN> sh -s - server --server https://<SERVER>:6443
```

Where `<SERVER>` is the IP or valid DNS of the server and `<TOKEN>` is the node-token from the server found at `/var/lib/rancher/k3s/server/node-token`.

For more information on installing K3s see the K3s installation docs [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/installation.html].

To have a look at your cluster run:

```
kubectl cluster-info
kubectl get pods --all-namespaces
```

**RKE2**

On every node, run the RKE2 installation script. Ensure that the RKE2 version you are installing is supported by Rancher [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions/].

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_CHANNEL=v1.xx sh -
```

Then you have to configure the HTTP proxy settings on the RKE2 systemd service, so that RKE2's containerd can pull images through the proxy:

```
cat <<'EOF' | sudo tee /etc/default/rke2-server > /dev/null
HTTP_PROXY=http://${proxy_host}
HTTPS_PROXY=http://${proxy_host}
NO_PROXY=127.0.0.0/8,10.0.0.0/8,cattle-
system.svc,172.16.0.0/12,192.168.0.0/16,.svc,.cluster.local
EOF
```

Next create the RKE2 configuration file on every node following the RKE2 High Availability documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/ha.html].

After that start and enable the `rke2-server` service:

```
systemctl enable rke2-server.service
```

```
systemctl start rke2-server.service
```

For more information on installing RKE2 see the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html].

To have a look at your cluster run:

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
alias kubectl=/var/lib/rancher/rke2/bin/kubectl
kubectl cluster-info
kubectl get pods --all-namespaces
```

#### Issues or errors?

See the Troubleshooting page.

### 3. Install SUSE Rancher Prime

Now that you have a running RKE2/K3s cluster, you can install Rancher in it. For security reasons all traffic to Rancher must be encrypted with TLS. For this tutorial you are going to automatically issue a self-signed certificate through cert-manager [https://cert-manager.io/]. In a real-world use-case you will likely use Let's Encrypt or provide your own certificate.

#### Install the Helm CLI

Install the Helm [https://helm.sh/docs/intro/install/] CLI on a host where you have a kubeconfig to access your Kubernetes cluster:

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod +x get_helm.sh
sudo ./get_helm.sh
```

#### Install cert-manager

SUSE Application Collection [https://docs.apps.rancher.io/] provides a SUSE Rancher Prime compatible cert-manager chart, follow the instructions below for installation.

1. Log in to the secure SUSE Application Collection registry:

   Before installing, authenticate your Helm CLI [https://docs.apps.rancher.io/get-started/authentication] against the Application Collection OCI registry using your enterprise credentials:

   ```
   helm registry login dp.apps.rancher.io -u <username-or-service-
   account-username> -p <access-token-or-service-account-token>
   ```

2. Create the target namespace `cert-manager`:

```
kubectl create namespace cert-manager
```

3. Create the Image Pull Secret:

   Because the Application Collection is a private registry, your cluster's nodes need a secret to pull the images. Generate this secret in the `cert-manager` namespace:

```
kubectl create secret docker-registry application-collection \
  --namespace cert-manager \
  --docker-server=dp.apps.rancher.io \
  --docker-username=<YOUR_APPCO_USERNAME> \
  --docker-password=<YOUR_APPCO_TOKEN>
```

4. Install cert-manager via the OCI chart path:

   Deploy the chart directly from the OCI registry path. Ensure you pass the `global.imagePullSecrets` flag so the pods can authenticate:

```
helm install <cert-manager-release-version>
oci://dp.apps.rancher.io/charts/cert-manager \
    --namespace cert-manager \
    --set global.imagePullSecrets={application-collection} \
    --set crds.enabled=true
```



If you are deploying in a cluster that requires FIPS cryptography compliance, you can substitute the chart path with `oci://dp.apps.rancher.io/charts/cert-manager-fips`.

5. Verify the Installation.

   Once you've installed cert-manager, you can verify it is deployed correctly by checking the `cert-manager` namespace for running pods:

```
kubectl get pods --namespace cert-manager
```

```
NAME                                       READY   STATUS    RESTARTS
AGE
cert-manager-56cc584bd4-nhjx7              1/1     Running   0
3m
cert-manager-cainjector-7cfc74b84b-kg7m2   1/1     Running   0
3m
cert-manager-webhook-784f6dd68-69dvn       1/1     Running   0
```

```
3m
```

#### Install SUSE Rancher Prime

Next you can install SUSE Rancher Prime itself. First, add the Helm repository:

```
helm repo add rancher-prime <helm-chart-repo-url>
```



To learn more about the Rancher Prime Helm chart repository URL, see our Prime-only documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#chart-repo-url]. Authentication is required. Use your SUSE Customer Center (SCC) [https://scc.suse.com/] credentials to log in.

Create a namespace:

```
kubectl create namespace cattle-system
```

And install Rancher with Helm. Rancher also needs a proxy configuration so that it can communicate with external application catalogs or retrieve Kubernetes version update metadata:

```
helm upgrade --install rancher rancher-prime/rancher \
   --namespace cattle-system \
   --set hostname=rancher.example.com \
   --set proxy=http://${proxy_host} \
   --set noProxy=127.0.0.0/8\\,10.0.0.0/8\\,cattle-
system.svc\\,172.16.0.0/12\\,192.168.0.0/16\\,.svc\\,.cluster.local
```

After waiting for the deployment to finish:

```
kubectl rollout status deployment -n cattle-system rancher
```

You can now navigate to `https://rancher.example.com` and start using Rancher.

#### Additional Resources

These resources could be helpful when installing Rancher:

- Rancher Helm chart options
- Adding TLS secrets
- Troubleshooting Rancher Kubernetes Installations

## Install/Upgrade SUSE Rancher Prime on a Kubernetes Cluster

In this section, you'll learn how to deploy Rancher on a Kubernetes cluster using the Helm CLI.

## Prerequisites

- Kubernetes Cluster
- Ingress Controller
- CLI Tools

### Kubernetes Cluster

Set up the Rancher server's local Kubernetes cluster.

Rancher can be installed on any Kubernetes cluster. This cluster can use upstream Kubernetes, or it can use one of Rancher's Kubernetes distributions, or it can be a managed Kubernetes cluster from a provider such as Amazon EKS.

For maximum uptime and stability, SUSE Rancher Prime server clusters require three nodes, each assigned all roles (`etcd`, `controlplane`, and `worker`). For further optimization in large environments, see Tuning and Best Practices for SUSE Rancher Prime at Scale.

For help setting up a Kubernetes cluster, we provide these tutorials:

- **K3s:** For the tutorial to install a K3s Kubernetes cluster, refer to this page. For help setting up the infrastructure for a high-availability K3s cluster, refer to this page.
- **RKE2:** For the tutorial to install an RKE2 Kubernetes cluster, refer to this page. For help setting up the infrastructure for a high-availability RKE2 cluster, refer to this page.
- **Amazon EKS:** For details on how to install Rancher on Amazon EKS, including how to install an Ingress controller so that the Rancher server can be accessed, refer to this page.
- **AKS:** For details on how to install Rancher with Azure Kubernetes Service, including how to install an Ingress controller so that the Rancher server can be accessed, refer to this page.
- **GKE:** For details on how to install Rancher with Google Kubernetes Engine, including how to install an Ingress controller so that the Rancher server can be accessed, refer to this page. GKE has two modes of operation when creating a Kubernetes cluster, Autopilot and Standard mode. The cluster configuration for Autopilot mode has restrictions on editing the kube-system namespace. However, Rancher needs to create resources in the kube-system namespace during installation. As a result, you will not be able to install Rancher on a GKE cluster created in Autopilot mode.

### Ingress Controller

The Rancher UI and API are exposed through an Ingress. This means the Kubernetes cluster that you install Rancher in must contain an Ingress controller.

For RKE2 and K3s installations, you don't have to install the Ingress controller manually because one is installed by default.

For distributions that do not include an Ingress Controller by default, like a hosted Kubernetes cluster such as EKS, GKE, or AKS, you have to deploy an Ingress controller first. Note that the Rancher Helm chart does not set an `ingressClassName` on the ingress by default. Because of this, you have to configure the Ingress controller to also watch ingresses without an `ingressClassName`.

Examples are included in the **Amazon EKS**, **AKS**, and **GKE** tutorials above.

#### CLI Tools

The following CLI tools are required for setting up the Kubernetes cluster. Please make sure these tools are installed and available in your $PATH.

- kubectl [https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl] - Kubernetes command-line tool.
- helm [https://docs.helm.sh/using_helm/#installing-helm] - Package management for Kubernetes. Refer to the Helm version requirements to choose a version of Helm to install Rancher. Refer to the instructions provided by the Helm project [https://helm.sh/docs/intro/install/] for your specific platform.

### Install the Rancher Helm Chart

Rancher is installed using the Helm [https://helm.sh/] package manager for Kubernetes. Helm charts provide templating syntax for Kubernetes YAML manifest documents. With Helm, we can create configurable deployments instead of just using static files.

For systems without direct internet access, see Air Gap: Kubernetes install.

To choose a Rancher version to install, refer to xref:[Choosing a Rancher Version.]

To choose a version of Helm to install Rancher with, refer to the Helm version requirements



The installation instructions assume you are using Helm 3.

To set up Rancher,

1. Add the Helm chart repository
2. Create a namespace for Rancher
3. Choose your SSL configuration
4. Install cert-manager (unless you are bringing your own certificates, or TLS will be terminated on a load balancer)
5. Install Rancher with Helm and your chosen certificate option
6. Verify that the Rancher server is successfully deployed
7. Save your options

#### 1. Add the Helm Chart Repository

Use the `helm repo add` command to add the Helm chart repository that contains charts to install Rancher Prime.

```
helm repo add rancher-prime <helm-chart-repo-url>
```



To learn more about the Rancher Prime Helm chart repository URL, see our Prime-only documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/reference-guide.html#chart-repo-url]. Authentication is required. Use your SUSE Customer Center (SCC) [https://scc.suse.com/] credentials to log in.

### 2. Create a Namespace for Rancher

We'll need to define a Kubernetes namespace where the resources created by the Chart should be installed. This should always be `cattle-system`:

```
kubectl create namespace cattle-system
```

### 3. Choose your SSL Configuration

The Rancher management server is designed to be secure by default and requires SSL/TLS configuration.



If you want to externally terminate SSL/TLS, see TLS termination on an External Load Balancer. As outlined on that page, this option does have additional requirements for TLS verification.

There are three recommended options for the source of the certificate used for TLS termination at the Rancher server:

- **Rancher-generated TLS certificate:** In this case, you will need to install `cert-manager` into the cluster. Rancher utilizes `cert-manager` to issue and maintain its certificates. Rancher will generate a CA certificate of its own, and sign a cert using that CA. `cert-manager` is then responsible for managing that certificate. No extra action is needed when `agent-tls-mode` is set to strict. More information can be found on this setting in Agent TLS Enforcement.
- **Let's Encrypt:** The Let's Encrypt option also uses `cert-manager`. However, in this case, cert-manager is combined with a special Issuer for Let's Encrypt that performs all actions (including request and validation) necessary for getting a Let's Encrypt issued cert. This configuration uses HTTP validation (`HTTP-01`), so the load balancer must have a public DNS record and be accessible from the internet. When setting `agent-tls-mode` to `strict`, you must also specify `--privateCA=true` and upload the Let's Encrypt CA as described in Adding TLS Secrets. More information can be found on this setting in Agent TLS Enforcement.
- **Bring your own certificate:** This option allows you to bring your own public- or private-CA signed certificate. Rancher will use that certificate to secure websocket and HTTPS traffic. In this case, you must upload this certificate (and associated key) as PEM-encoded files with the name `tls.crt` and `tls.key`. If you are using a private CA, you must also upload that certificate. This is due to the fact that this private CA may not be trusted by your nodes. Rancher will take that CA certificate, and generate a checksum from it, which the various Rancher components will use to validate their connection to Rancher. If `agent-tls-mode` is set to `strict`, the CA must be uploaded, so that downstream clusters can successfully connect. More information can be found on this setting in Agent TLS Enforcement.

| Configuration | Helm Chart Option | Requires cert-manager |
|---|---|---|
| Rancher Generated Certificates (Default) | `ingress.tls.source=rancher` | yes |
| Let's Encrypt | `ingress.tls.source=letsEncrypt` | yes |
| Certificates from Files | `ingress.tls.source=secret` | no |

### 4. Install cert-manager

SUSE Application Collection [https://docs.apps.rancher.io/] provides a SUSE Rancher Prime compatible cert-manager chart, follow the instructions below for installation.

1. Log in to the secure SUSE Application Collection registry:

   Before installing, authenticate your Helm CLI [https://docs.apps.rancher.io/get-started/authentication] against the Application Collection OCI registry using your enterprise credentials:

   ```
   helm registry login dp.apps.rancher.io -u <username-or-service-
   account-username> -p <access-token-or-service-account-token>
   ```

2. Create the target namespace `cert-manager`:

   ```
   kubectl create namespace cert-manager
   ```

3. Create the Image Pull Secret:

   Because the Application Collection is a private registry, your cluster's nodes need a secret to pull the images. Generate this secret in the `cert-manager` namespace:

   ```
   kubectl create secret docker-registry application-collection \
     --namespace cert-manager \
     --docker-server=dp.apps.rancher.io \
     --docker-username=<YOUR_APPCO_USERNAME> \
     --docker-password=<YOUR_APPCO_TOKEN>
   ```

4. Install cert-manager via the OCI chart path:

   Deploy the chart directly from the OCI registry path. Ensure you pass the `global.imagePullSecrets` flag so the pods can authenticate:

   ```
   helm install <cert-manager-release-version>
   oci://dp.apps.rancher.io/charts/cert-manager \
       --namespace cert-manager \
       --set global.imagePullSecrets={application-collection} \
       --set crds.enabled=true
   ```

   If you are deploying in a cluster that requires FIPS cryptography compliance, you can substitute the chart path with `oci://dp.apps.rancher.io/charts/cert-manager-fips`.

5. Verify the Installation.

   Once you've installed cert-manager, you can verify it is deployed correctly by checking the `cert-manager` namespace for running pods:

   ```
   kubectl get pods --namespace cert-manager
   ```

```
NAME                                       READY   STATUS    RESTARTS
AGE
cert-manager-56cc584bd4-nhjx7              1/1     Running   0
3m
cert-manager-cainjector-7cfc74b84b-kg7m2   1/1     Running   0
3m
cert-manager-webhook-784f6dd68-69dvn       1/1     Running   0
3m
```

#### 5. Install Rancher with Helm and Your Chosen Certificate Option

The exact command to install Rancher differs depending on the certificate configuration.

However, irrespective of the certificate configuration, the name of the Rancher installation in the `cattle-system` namespace should always be `rancher`.



*Testing and Development:*

This final command to install Rancher requires a domain name that forwards traffic to Rancher. If you are using the Helm CLI to set up a proof-of-concept, you can use a fake domain name when passing the `hostname` option. An example of a fake domain name would be `<IP_OF_LINUX_NODE>.sslip.io`, which would expose Rancher on an IP where it is running. Production installs would require a real domain name.

**Rancher-generated Certificates**

The default is for Rancher to generate a CA and uses `cert-manager` to issue the certificate for access to the Rancher server interface.

Because `rancher` is the default option for `ingress.tls.source`, we are not specifying `ingress.tls.source` when running the `helm install` command.

- Set the `hostname` to the DNS name you pointed at your load balancer.
- Set the `bootstrapPassword` to something unique for the `admin` user.
- To install a specific Rancher version, use the `--version` flag, example: `--version 2.7.0`

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set bootstrapPassword=admin
```

Wait for Rancher to be rolled out:

```
kubectl -n cattle-system rollout status deploy/rancher
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated
replicas are available...
```

```
deployment "rancher" successfully rolled out
```

#### Let's Encrypt

This option uses `cert-manager` to automatically request and renew Let's Encrypt [https://letsencrypt.org/] certificates. This is a free service that provides you with a valid certificate as Let's Encrypt is a trusted CA.



You need to have port 80 open as the HTTP-01 challenge can only be done on port 80.

In the following command,

- `hostname` is set to the public DNS record,
- Set the `bootstrapPassword` to something unique for the `admin` user.
- `ingress.tls.source` is set to `letsEncrypt`
- `letsEncrypt.email` is set to the email address used for communication about your certificate (for example, expiry notices)
- Set `letsEncrypt.ingress.class` to whatever your ingress controller is, e.g., `traefik`, `nginx`, `haproxy`, etc.



When `agent-tls-mode` is set to `strict` (the default value for new installs of Rancher starting from v2.9.0), you must supply the `privateCA=true` chart value (e.x. through `--set privateCA=true`) and upload the Let's Encrypt Certificate Authority as outlined in Adding TLS Secrets. Information on identifying the Let's Encrypt Root CA can be found in the Let's Encrypt docs [https://letsencrypt.org/certificates/]. If you don't upload the CA, then Rancher may fail to connect to new or existing downstream clusters.

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set bootstrapPassword=admin \
  --set ingress.tls.source=letsEncrypt \
  --set letsEncrypt.email=me@example.org \
  --set letsEncrypt.ingress.class=nginx
```

Wait for Rancher to be rolled out:

```
kubectl -n cattle-system rollout status deploy/rancher
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated
replicas are available...
deployment "rancher" successfully rolled out
```

**Certificates from Files**

In this option, Kubernetes secrets are created from your own certificates for Rancher to use.

When you run this command, the `hostname` option must match the `Common Name` or a `Subject Alternative Names` entry in the server certificate or the Ingress controller will fail to configure correctly.

Although an entry in the `Subject Alternative Names` is technically required, having a matching `Common Name` maximizes compatibility with older browsers and applications.



If you want to check if your certificates are correct, see How do I check Common Name and Subject Alternative Names in my server certificate? [../../../faq/technical-items.adoc#_how_do_i_check_common_name_and_subject_alternative_names_in_my_server_certificate]

- Set the `hostname`.
- Set the `bootstrapPassword` to something unique for the `admin` user.
- Set `ingress.tls.source` to `secret`.

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set bootstrapPassword=admin \
  --set ingress.tls.source=secret
```

If you are using a Private CA signed certificate , add `--set privateCA=true` to the command:

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set bootstrapPassword=admin \
  --set ingress.tls.source=secret \
  --set privateCA=true
```

Now that Rancher is deployed, see Adding TLS Secrets to publish the certificate files so Rancher and the Ingress controller can use them.

The Rancher chart configuration has many options for customizing the installation to suit your specific environment. Here are some common advanced scenarios.

- HTTP Proxy
- Private Container Image Registry
- TLS Termination on an External Load Balancer

See the Chart Options for the full list of options.

#### 6. Verify that the Rancher Server is Successfully Deployed

After adding the secrets, check if Rancher was rolled out successfully:

```
kubectl -n cattle-system rollout status deploy/rancher
Waiting for deployment "rancher" rollout to finish: 0 of 3 updated
replicas are available...
deployment "rancher" successfully rolled out
```

If you see the following error: `error: deployment "rancher" exceeded its progress deadline`, you can check the status of the deployment by running the following command:

```
kubectl -n cattle-system get deploy rancher
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
rancher   3         3         3            3           3m
```

It should show the same count for `DESIRED` and `AVAILABLE`.

#### 7. Save Your Options

Make sure you save the `--set` options you used. You will need to use the same options when you upgrade Rancher to new versions with Helm.

#### Finishing Up

That's it. You should have a functional Rancher server.

In a web browser, go to the DNS name that forwards traffic to your load balancer. Then you should be greeted by the colorful login page.

Doesn't work? Take a look at the Troubleshooting Page

# Upgrades

The following instructions will guide you through upgrading a Rancher server that was installed on a Kubernetes cluster with Helm. These steps also apply to air-gapped installs with Helm.

For the instructions to upgrade Rancher installed with Docker, refer to xref:[this page.]

## Prerequisites

#### Access to kubeconfig

Helm should be run from the same location as your kubeconfig file, or the same location where you run your kubectl commands from.

If you installed Kubernetes with RKE2/K3s, the Kubeconfig is stored in the `/etc/rancher/rke2/rke2.yaml` or `/etc/rancher/k3s/k3s.yaml` directory depending on your chosen distribution.

The kubeconfig can also be manually targeted for the intended cluster with the `--kubeconfig` tag (see: https://helm.sh/docs/helm/helm/)

#### Review Known Issues

Review the list of known issues for each Rancher version, which can be found in the release notes on GitHub [https://github.com/rancher/rancher/releases] and on the Rancher forums. [https://forums.rancher.com/c/announcements/12]

#### Upgrade Path



The only tested and supported Rancher upgrade path between minor versions (e.g. v2.13.x to v2.14.x) is to upgrade from the latest available patch version of your current running minor release to the latest available patch version of the next minor release.

Before initiating a minor version upgrade, verify that you are running the most recent patch release of your current version.

You can query the available chart versions with the Helm CLI:

1. Update your local Helm repo cache.

```
helm repo update
```

2. Search for available versions in the repository:

```
helm search repo rancher-prime/rancher --versions
```

If your installation is not on the latest patch version of the current minor release, you must upgrade to that version before proceeding to the next minor version.

#### Helm Version

The upgrade instructions assume you are using Helm 3.

#### For air-gapped installs: Populate private registry

For air-gapped installs only, collect and populate images for the new Rancher server version. Follow the guide to populate your private registry with the images for the Rancher version that you want to upgrade to.

#### For upgrades with cert-manager older than 0.8.0

Let's Encrypt will be blocking cert-manager instances older than 0.8.0 starting November 1st 2019. [https://community.letsencrypt.org/t/blocking-old-cert-manager-versions/98753] Upgrade cert-manager to the latest version by following these instructions.

### Upgrade Outline

Follow the steps to upgrade Rancher server:

**1. Back up Your Kubernetes Cluster that is Running Rancher Server**

Use the backup application to back up Rancher.

You'll use the backup as a restore point if something goes wrong during upgrade.

**2. Update the Helm chart repository**

1. Update your local Helm repo cache.

   ```
   helm repo update
   ```

2. Get the repository name that you used to install Rancher.

   ```
   helm repo list

   NAME                  URL
   rancher-prime         <helm-chart-repo-url>
   ```

3. Fetch the latest chart to install Rancher from the Helm chart repository.

   This command will pull down the latest charts and save it in the current directory as a `.tgz` file.

   ```
   helm fetch rancher-prime/rancher
   ```

   You can fetch the chart for the specific version you are upgrading to by adding in the `--version=` tag. For example:

   ```
   helm fetch rancher-prime/rancher --version={current-patch-version}
   ```

**3. Review Rancher Feature Chart Versions Before Upgrade**

Rancher feature charts follow specific release lines that align with Rancher versions. Major versions of feature charts correspond to Rancher minor versions and follow a defined versioning scheme.

Before upgrading Rancher, review any installed Rancher feature charts and upgrade them to the latest available version within their current chart release line. This helps ensure compatibility and avoids potential issues during or after the Rancher upgrade.

To review installed feature charts:

1. In the Rancher UI, go to **Apps & Marketplace**.
2. Select **Installed Apps**.
3. Review the chart versions and upgrade to the latest patch release within the same chart major version if needed.

For more information, see the Helm Charts in Rancher – Versioning Scheme.

#### 4. Upgrade Rancher

This section describes how to upgrade normal (Internet-connected) or air-gapped installations of Rancher with Helm.

*Air Gap Instructions:*

If you are installing Rancher in an air-gapped environment, skip the rest of this page and render the Helm template by following the instructions on this page.

*Upgrading from Rancher v2.13.1*

In Rancher v2.13.2, the Helm chart name has reverted from `rancher-prime` back to `rancher`. If you are upgrading from v2.13.1, the existing Ingress resource created by the previous chart conflicts with the new one, causing the upgrade to fail with an error similar to:

```
Error: UPGRADE FAILED: failed to create resource: admission
webhook "validate.nginx.ingress.kubernetes.io" denied the
request: host "rancher.my.org" and path "/" is already
defined in ingress cattle-system/rancher-rancher-prime
```

To prevent or resolve this issue, you must manually delete the old Ingress before completing the upgrade:

```
kubectl delete ingress rancher-rancher-prime -n cattle-
system
```

Get the values, which were passed with `--set`, from the current Rancher Helm chart that is installed.

```
helm get values rancher -n cattle-system

hostname: rancher.my.org
```

There will be more values that are listed with this command. This is just an example of one of the values.

Your deployment name may vary; for example, if you're deploying Rancher through the AWS Marketplace, the deployment name is 'rancher-prime'.

Thus:

```
helm get values rancher-prime -n cattle-system

hostname: rancher.my.org
```

If you are upgrading cert-manager to the latest version from v1.5 or below, follow the cert-manager upgrade docs to learn how to upgrade cert-manager without needing to perform an uninstall or reinstall of Rancher. Otherwise, follow the steps to upgrade Rancher below.

#### Steps to Upgrade Rancher

Upgrade Rancher to the latest version with all your settings.

Take all the values from the previous step and append them to the command using `--set key=value`.

```
helm upgrade rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org
```



The above is an example, there may be more values from the previous step that need to be appended.



If you deploy Rancher through the AWS Marketplace, the deployment name is 'rancher-prime'.

Thus:

```
helm upgrade rancher-prime rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org
```

Alternatively, it's possible to export the current values to a file and reference that file during upgrade. For example, to only change the Rancher version:

1. Export the current values to a file:

   ```
   helm get values rancher -n cattle-system -o yaml > values.yaml
   ```

2. Update only the Rancher version:

   ```
   helm upgrade rancher rancher-prime/rancher \
     --namespace cattle-system \
     -f values.yaml \
     --version={current-patch-version}
   ```

#### 5. Verify the Upgrade

Log into Rancher to confirm that the upgrade succeeded.

### Known Upgrade Issues

A list of known issues for each Rancher version can be found in the release notes on GitHub [https://github.com/rancher/rancher/releases] and on the Rancher forums. [https://forums.rancher.com/c/announcements/12]

# Rollbacks

This page outlines how to roll back Rancher to a previous version after an upgrade.



- This is only possible by restoring to a previous version from a backup instance. Changing versions using kubectl or Helm is not supported.
- When rolling back, ensure the version being restored is under active support. Rolling back to versions that are end-of-life is not supported. Refer to SUSE's Product Support Lifecycle [https://www.suse.com/lifecycle/#suse-rancher-prime] page to verify key lifecycle dates for your given version.

Follow the instructions from this page when:

- The running Rancher instance has been upgraded to a newer version after the backup was made.
- The upstream (local) cluster is the same as where the backup was made.



- Follow these steps to migrate Rancher.
- If you need to restore Rancher to its previous state at the same Rancher version, see the restore documentation.

To roll back Rancher to a lower/previous version, use the **Rancher Backups** application and restore Rancher from backup.

Rancher has to be started with the lower/previous version after a rollback.

A restore is performed by creating a Restore custom resource.

### Step 1: Create the Restore Custom Resource

1. Click ☰ **> Cluster Management**.
2. Go to the local cluster and click **Explore**.
3. In the left navigation bar, click **Rancher Backups › Restore**.



If the Rancher Backups app is not visible, you will need to install it from the Charts page in **Apps**. Refer here for more information.

1. Click **Create**.
2. Create the Restore with the form or with YAML. For help creating the Restore resource using the online form, refer to the configuration reference and to the examples.
3. To use the YAML editor, you can click **Create › Create from YAML.** Enter the Restore YAML. The following is an example Restore custom resource:

```
 apiVersion: resources.cattle.io/v1
 kind: Restore
 metadata:
   name: restore-migration
 spec:
   backupFilename: backup-b0450532-cee1-4aa1-a881-f5f48a007b1c-2020-
09-15T07-27-09Z.tar.gz
   encryptionConfigSecretName: encryptionconfig
   storageLocation:
     s3:
       credentialSecretName: s3-creds
       credentialSecretNamespace: default
       bucketName: rancher-backups
       folder: rancher
       region: us-west-2
       endpoint: s3.us-west-2.amazonaws.com
```

For help configuring the Restore, refer to the configuration reference and to the examples.

4. Click **Create**.

**Result:** The backup file is created and updated to the target storage location. The resources are restored in this order:

1. Custom Resource Definitions (CRDs)
2. Cluster-scoped resources
3. Namespaced resources

To check how the restore is progressing, you can check the logs of the operator. Follow these steps to get the logs:

```
kubectl get pods -n cattle-resources-system
kubectl logs -n cattle-resources-system -f
```

### Step 2: Roll Back to a Previous Rancher Version

Rancher can be rolled back using the Helm CLI. To roll back to the previous version:

```
helm rollback rancher -n cattle-system
```

If the previous revision is not the intended target, you can specify a revision to roll back to. To see the deployment history:

```
helm history rancher -n cattle-system
```

When the target revision is determined, perform the rollback. This example will roll back to revision 3:

```
helm rollback rancher 3 -n cattle-system
```

### Alternative Steps for Special Scenarios

Alternative steps need to be performed for rollbacks in the following scenarios:

- Rolling back from v2.14.0 and later to an earlier version of v2.13.x.

In Rancher v2.13.0, Rancher Turtles became the default manager for CAPI resources, replacing the previously embedded cluster-api controllers, and in Rancher v2.14.0 the embedded cluster-api was removed entirely. As a result, if you roll back from Rancher v2.14.0 and later to an earlier version of Rancher v2.13.x and do not intend to continue using Rancher Turtles to manage CAPI resources, additional manual steps may be required to use the embedded cluster-api controllers. From Rancher v2.14.0 onward, Rancher Turtles is the only supported manager for CAPI resources.

In Rancher v2.14.0, the cluster-api module is upgraded from v1.10.6 to v1.12.2. The cluster-api v1.12.2, in turn, upgrades the apiVersions of its Custom Resource Definitions (CRDs) from `cluster.x-k8s.io/v1beta1` to `cluster.x-k8s.io/v1beta2`. Rancher backup files include Cluster API CRDs. When restoring backup data from Rancher v2.13.x to a local cluster after upgrading to v2.14.0, the Rancher Backup application first restores the `v1beta1` CRDs. This fails because the `v1beta2` version cannot be removed from the CRDs while `v1beta2` custom resources are present in the cluster.



- Follow the instructions from this page for restoring Rancher on the same cluster where it was backed up from. In order to migrate Rancher to a new cluster, follow the steps to migrate Rancher.
- While restoring Rancher on the same setup, the Rancher deployment is manually scaled down before the restore starts, then the operator will scale it back up once the restore completes. As a result, Rancher and its UI will be unavailable until the restore is complete. While the UI is unavailable, use the original cluster kubeconfig with the restore YAML file: `kubectl create -f restore.yaml`.

#### Step 1: Clean Up the Upstream (Local) Cluster

To avoid rollback failure, follow these instructions [https://github.com/rancher/rancher-cleanup/blob/main/README.md] to run the scripts **before** you attempt a restore operation or rollback:

- `cleanup.sh`: Cleans up the cluster.
- `verify.sh`: Checks for any Rancher-related resources in the cluster.



There will be downtime while `cleanup.sh` runs, since the script deletes resources created by Rancher.

**Result:** All Rancher-related resources should be cleaned up on the upstream (local) cluster. See the rancher/rancher-cleanup repo [https://github.com/rancher/rancher-cleanup] for more details and source code.

#### Step 2: Restore the Backup and Bring Up Rancher

At this point, there should be no Rancher-related resources on the upstream cluster. Therefore, the next step will be the same as if you were migrating Rancher to a new cluster that contains no Rancher resources. Follow these instructions to install the Rancher-Backup Helm chart and restore Rancher to its previous state.

Please keep in mind that:

1. Step 3 can be skipped, because the Cert-Manager app should still exist on the upstream (local) cluster if it was installed before.
2. At Step 4, install the Rancher version you intend to roll back to.

# Troubleshooting the SUSE Rancher Prime Server Kubernetes Cluster

This section describes how to troubleshoot an installation of Rancher on a Kubernetes cluster.

## Relevant Namespaces

Most of the troubleshooting will be done on objects in these 3 namespaces.

- `cattle-system` - `rancher` deployment and pods.
- `traefik` - Ingress controller pods and services.
- `cert-manager` - `cert-manager` pods.

## "default backend - 404"

A number of things can cause the ingress-controller not to forward traffic to your rancher instance. Most of the time its due to a bad ssl configuration.

Things to check

- Is Rancher Running
- Cert CN is "Kubernetes Ingress Controller Fake Certificate"

## Check if Rancher is Running

Use `kubectl` to check the `cattle-system` system namespace and see if the Rancher pods are in a Running state.

```
kubectl -n cattle-system get pods

NAME                                READY     STATUS    RESTARTS   AGE
pod/rancher-784d94f59b-vgqzh   1/1       Running   0          10m
```

If the state is not `Running`, run a `describe` on the pod and check the Events.

```
kubectl -n cattle-system describe pod

...
Events:
  Type     Reason                  Age   From                Message
  ----     ------                  ----  ----                -------
  Normal   Scheduled               11m   default-scheduler   Successfully
assigned rancher-784d94f59b-vgqzh to localhost
  Normal   SuccessfulMountVolume   11m   kubelet, localhost
MountVolume.SetUp succeeded for volume "rancher-token-dj4mt"
  Normal   Pulling                 11m   kubelet, localhost  pulling
image "rancher/rancher:v2.0.4"
  Normal   Pulled                  11m   kubelet, localhost  Successfully
pulled image "rancher/rancher:v2.0.4"
  Normal   Created                 11m   kubelet, localhost  Created
container
  Normal   Started                 11m   kubelet, localhost  Started
container
```

### Check the Rancher Logs

Use `kubectl` to list the pods.

```
kubectl -n cattle-system get pods

NAME                            READY     STATUS    RESTARTS   AGE
pod/rancher-784d94f59b-vgqzh    1/1       Running   0          10m
```

Use `kubectl` and the pod name to list the logs from the pod.

```
kubectl -n cattle-system logs -f rancher-784d94f59b-vgqzh
```

### Cert CN is "Kubernetes Ingress Controller Fake Certificate"

Use your browser to check the certificate details. If it says the Common Name is "Kubernetes Ingress Controller Fake Certificate", something may have gone wrong with reading or issuing your SSL cert.



If you are using LetsEncrypt to issue certs, it can sometimes take a few minutes to issue the cert.

### Checking for issues with cert-manager issued certs (Rancher Generated or LetsEncrypt)

`cert-manager` has 3 parts.

- `cert-manager` pod in the `cert-manager` namespace.
- `Issuer` object in the `cattle-system` namespace.
- `Certificate` object in the `cattle-system` namespace.

Work backwards and do a `kubectl describe` on each object and check the events. You can track down what might be missing.

For example there is a problem with the Issuer:

```
kubectl -n cattle-system describe certificate
...
Events:
  Type     Reason          Age                 From          Message
  ----     ------          ----                ----          -------
  Warning  IssuerNotReady  18s (x23 over 19m)  cert-manager  Issuer
rancher not ready
```

```
kubectl -n cattle-system describe issuer
...
Events:
  Type     Reason         Age                From          Message
  ----     ------         ----               ----          -------
  Warning  ErrInitIssuer  19m (x12 over 19m)  cert-manager  Error
initializing issuer: secret "tls-rancher" not found
  Warning  ErrGetKeyPair  9m (x16 over 19m)   cert-manager  Error
getting keypair for CA issuer: secret "tls-rancher" not found
```

### Checking for Issues with Your Own SSL Certs

Your certs get applied directly to the Ingress object in the `cattle-system` namespace.

Check the status of the Ingress object and see if its ready.

```
kubectl -n cattle-system describe ingress
```

If its ready and the SSL is still not working you may have a malformed cert or secret.

Check the nginx-ingress-controller logs. Because the nginx-ingress-controller has multiple containers in its pod you will need to specify the name of the container.

```
kubectl logs -n traefik traefik-6b94b8b688-bngw2
...
W0705 23:04:58.240571       7 backend_ssl.go:49] error obtaining PEM
from secret cattle-system/tls-rancher-ingress: error retrieving secret
cattle-system/tls-rancher-ingress: secret cattle-system/tls-rancher-
ingress was not found
```

### No matches for kind "Issuer"

The SSL configuration option you have chosen requires cert-manager to be installed before installing Rancher or else the following error is shown:

```
Error: validation failed: unable to recognize "": no matches for kind
"Issuer" in version "certmanager.k8s.io/vlalphal"
```

Install cert-manager and try installing Rancher again.

### Canal Pods show READY 2/3

The most common cause of this issue is port 8472/UDP is not open between the nodes. Check your local firewall, network routing or security groups.

Once the network issue is resolved, the `canal` pods should timeout and restart to establish their connections.

### Failed to dial to /var/run/docker.sock: ssh: rejected: administratively prohibited (open failed)

Some causes of this error include:

- User specified to connect with does not have permission to access the Docker socket. This can be checked by logging into the host and running the command `docker ps`:

  ```
  $ ssh user@server
  user@server$ docker ps
  CONTAINER ID        IMAGE               COMMAND
  CREATED             STATUS              PORTS
  NAMES
  ```

See Manage Docker as a non-root user [https://docs.docker.com/install/linux/linux-postinstall/#manage-docker-as-a-non-root-user] how to set this up properly.

- When using RedHat/CentOS as operating system, you cannot use the user `root` to connect to the nodes because of Bugzilla #1527565 [https://bugzilla.redhat.com/show_bug.cgi?id=1527565]. You will need to add a separate user and configure it to access the Docker socket. See Manage Docker as a non-root user [https://docs.docker.com/install/linux/linux-postinstall/#manage-docker-as-a-non-root-user] how to set this up

properly.

- SSH server version is not version 6.7 or higher. This is needed for socket forwarding to work, which is used to connect to the Docker socket over SSH. This can be checked using `sshd -V` on the host you are connecting to, or using netcat:

```
$ nc xxx.xxx.xxx.xxx 22
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.10
```

### Failed to dial ssh using address [xxx.xxx.xxx.xxx:xx]: ssh: handshake failed: ssh: unable to authenticate, attempted methods [none publickey], no supported methods remain

The key file specified as `ssh_key_path` is not correct for accessing the node. Double-check if you specified the correct `ssh_key_path` for the node and if you specified the correct user to connect with.

### Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

The node is not reachable on the configured `address` and `port`.

### Agent reports TLS errors

When using Rancher, you may encounter error messages from the `fleet-agent`, `system-agent`, or `cluster-agent`, such as the message below:

```
tls: failed to verify certificate: x509: failed to load system roots and
no roots provided; readdirent /dev/null: not a directory
```

This occurs when Rancher was configured with `agent-tls-mode` set to `strict`, but couldn't find cacerts in the `cacert` setting. To resolve the issue, set the `agent-tls-mode` to `system-store`, or upload the CA for Rancher as described in Adding TLS Secrets.

### New Cluster Deployment is stuck in "Waiting for Agent to check in"

When Rancher has `agent-tls-mode` set to `strict`, new clusters may fail to provision and report a generic "Waiting for Agent to check in" error message. The root cause of this is similar to the above case of TLS errors - Rancher's agent can't determine which CA Rancher is using (or can't verify that Rancher's cert is actually signed by the specified certificate authority).

To resolve the issue, set the `agent-tls-mode` to `system-store` or upload the CA for Rancher as described in Adding TLS Secrets.

### Troubleshooting Certificates

#### How Do I Know if My Certificates are in PEM Format?

You can recognize the PEM format by the following traits:

- The file begins with the following header:

```
-----BEGIN CERTIFICATE-----
```

- The header is followed by a long string of characters.
- The file ends with a footer:

----END CERTIFICATE----

PEM Certificate Example:

```
----BEGIN CERTIFICATE-----
MIIGVDCCBDygAwIBAgIJAMiIrEm29kRLMA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNV
... more lines
VWQqljhfacYPgp8KJUJENQ9h5hZ2nSCrI+W00Jcw4QcEdCI8HL5wmg==
-----END CERTIFICATE-----
```

PEM Certificate Key Example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIGVDCCBDygAwIBAgIJAMiIrEm29kRLMA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNV
... more lines
VWQqljhfacYPgp8KJUJENQ9h5hZ2nSCrI+W00Jcw4QcEdCI8HL5wmg==
-----END RSA PRIVATE KEY-----
```

If your key looks like the example below, see Converting a Certificate Key From PKCS8 to PKCS1.

```
-----BEGIN PRIVATE KEY-----
MIIGVDCCBDygAwIBAgIJAMiIrEm29kRLMA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNV
... more lines
VWQqljhfacYPgp8KJUJENQ9h5hZ2nSCrI+W00Jcw4QcEdCI8HL5wmg==
-----END PRIVATE KEY-----
```

**Converting a Certificate Key From PKCS8 to PKCS1**

If you are using a PKCS8 certificate key file, Rancher will log the following line:

```
ListenConfigController cli-config [listener] failed with : failed to
read private key: asnl: structure error: tags don't match (2 vs {class:0
tag:16 length:13 isCompound:true})
```

To make this work, you will need to convert the key from PKCS8 to PKCS1 using the command below:

```
openssl rsa -in key.pem -out convertedkey.pem
```

You can now use `convertedkey.pem` as certificate key file for Rancher.

#### What is the Order of Certificates if I Want to Add My Intermediate(s)?

The order of adding certificates is as follows:

```
-----BEGIN CERTIFICATE-----
%YOUR_CERTIFICATE%
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
%YOUR_INTERMEDIATE_CERTIFICATE%
-----END CERTIFICATE-----
```

#### How Do I Validate My Certificate Chain?

You can validate the certificate chain by using the `openssl` binary. If the output of the command (see the command example below) ends with `Verify return code: 0 (ok)`, your certificate chain is valid. The `ca.pem` file must be the same as you added to the `rancher/rancher` container.

When using a certificate signed by a recognized Certificate Authority, you can omit the `-CAfile` parameter.

Command:

```
openssl s_client -CAfile ca.pem -connect rancher.yourdomain.com:443
...
    Verify return code: 0 (ok)
```

## SUSE Rancher Prime HA

The commands/steps listed on this page can be used to check your Rancher Kubernetes Installation.

Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kube_config_cluster.yml`).

#### Check Rancher Pods

Rancher pods are deployed as a Deployment in the `cattle-system` namespace.

Check if the pods are running on all nodes:

```
kubectl -n cattle-system get pods -l app=rancher -o wide
```

Example output:

```
NAME                       READY   STATUS    RESTARTS   AGE   IP
NODE
rancher-7dbd7875f7-n6t5t   1/1     Running   0          8m    x.x.x.x
x.x.x.x
rancher-7dbd7875f7-qbj5k   1/1     Running   0          8m    x.x.x.x
x.x.x.x
rancher-7dbd7875f7-qw7wb   1/1     Running   0          8m    x.x.x.x
x.x.x.x
```

If a pod is unable to run (Status is not **Running**, Ready status is not showing `1/1` or you see a high count of Restarts), check the pod details, logs and namespace events.

#### Pod Details

```
kubectl -n cattle-system describe pods -l app=rancher
```

#### Pod Container Logs

```
kubectl -n cattle-system logs -l app=rancher
```

#### Namespace Events

```
kubectl -n cattle-system get events
```

### Check Ingress

Ingress should have the correct `HOSTS` (showing the configured FQDN) and `ADDRESS` (host address(es) it will be routed to).

```
kubectl -n cattle-system get ingress
```

Example output:

```
NAME      HOSTS                 ADDRESS                   PORTS
AGE
rancher   rancher.yourdomain.com   x.x.x.x,x.x.x.x,x.x.x.x   80, 443
2m
```

### Check Ingress Controller Logs

When accessing your configured Rancher FQDN does not show you the UI, check the ingress controller logging to see what happens when you try to access Rancher:

```
kubectl -n traefik logs -l app=traefik
```

#### Leader Election

The leader is determined by a leader election process. After the leader has been determined, the leader (`holderIdentity`) is saved in the `cattle-controllers` Lease in the `kube-system` namespace (in this example, `rancher-dbc7ff869-gvg6k`).

```
kubectl -n kube-system get lease cattle-controllers
```

Example output:

```
NAME                 HOLDER                          AGE
cattle-controllers   rancher-dbc7ff869-gvg6k         6h10m
```

##### Configuration

*Available as of Rancher 2.8.3*

If the Kubernetes API experiences latency, the Rancher replica holding the leader lock may not be able to renew the lease before the lease becomes invalid, which can be observed in the Rancher logs:

```
E0629 04:13:07.293461       34 leaderelection.go:364] Failed to update
lock: Put
"https://172.17.0.1:443/apis/coordination.k8s.io/v1/namespaces/kube-
system/leases/cattle-controllers?timeout=15m0s": context deadline
exceeded
I0629 04:13:07.293594       34 leaderelection.go:280] failed to renew
lease kube-system/cattle-controllers: timed out waiting for the
condition
...
2024/06/29 04:13:10 [FATAL] leaderelection lost for cattle-controllers
```

To mitigate this, you can set environment variables in the `rancher` Deployment to modify the default parameters for leader election:

- `CATTLE_ELECTION_LEASE_DURATION`: The lease duration [https://pkg.go.dev/k8s.io/client-go/tools/leaderelection#LeaderElectionConfig.LeaseDuration]. The default value is 45s.
- `CATTLE_ELECTION_RENEW_DEADLINE`: The renew deadline [https://pkg.go.dev/k8s.io/client-go/tools/leaderelection#LeaderElectionConfig.RenewDeadline]. The default value is 30s.
- `CATTLE_ELECTION_RETRY_PERIOD`: The retry period [https://pkg.go.dev/k8s.io/client-go/tools/leaderelection#LeaderElectionConfig.RetryPeriod]. The default value is 2s.

Example:

```
kubectl -n cattle-system set env deploy/rancher
CATTLE_ELECTION_LEASE_DURATION=2m CATTLE_ELECTION_RENEW_DEADLINE=90s
CATTLE_ELECTION_RETRY_PERIOD=10s
```

This will temporarily increase the lease duration, renew deadline and retry period to 120, 90 and 10 seconds respectively. Alternatively, in order to make such changes permanent, these environment variables can be set by using Helm values instead.

# Rancher Administration

After installation, the system administrator should configure Rancher to configure authentication, authorization, security, default settings, security policies, and drivers.

## First Log In

After you log into Rancher for the first time, Rancher will prompt you for a **Rancher Server URL**. You should set the URL to the main entry point to the Rancher Server. When a load balancer sits in front a Rancher Server cluster, the URL should resolve to the load balancer. The system will automatically try to infer the Rancher Server URL from the IP address or host name of the host running the Rancher Server. This is only correct if you are running a single node Rancher Server installation. In most cases, therefore, you need to set the Rancher Server URL to the correct value yourself.



After you set the Rancher Server URL, we do not support updating it. Set the URL with extreme care.

## Authentication

One of the key features that Rancher adds to Kubernetes is centralized user authentication. This feature allows to set up local users and/or connect to an external authentication provider. By connecting to an external authentication provider, you can leverage that provider's user and groups.

For more information how authentication works and how to configure each provider, see Authentication.

## Authorization

Within Rancher, each person authenticates as a *user*, which is a login that grants you access to Rancher. Once the user logs in to Rancher, their *authorization*, or their access rights within the system, is determined by the user's role. Rancher provides built-in roles to allow you to easily configure a user's permissions to resources, but Rancher also provides the ability to customize the roles for each Kubernetes resource.

For more information how authorization works and how to customize roles, see Roles Based Access Control (RBAC).

## User Retention

In Rancher v2.8.5 and later, you can enable user retention. This feature automatically removes inactive users after a configurable period of time.

The user retention feature is disabled by default.

For more information, see Enabling User Retention.

## Provisioning Drivers

Drivers in Rancher allow you to manage which providers can be used to provision hosted Kubernetes clusters or nodes in an infrastructure provider to allow Rancher to deploy and manage Kubernetes.

For more information, see Provisioning Drivers.

# Adding Kubernetes Versions into Rancher

With this feature, you can upgrade to the latest version of Kubernetes as soon as it is released, without upgrading Rancher. This feature allows you to easily upgrade Kubernetes patch versions (i.e. `v1.15.X`), but not intended to upgrade Kubernetes minor versions (i.e. `v1.X.0`) as Kubernetes tends to deprecate or add APIs between minor versions.

The information that Rancher uses to provision RKE clusters is now located in the Rancher Kubernetes Metadata. For details on metadata configuration and how to change the Kubernetes version used for provisioning RKE clusters, see Rancher Kubernetes Metadata.

Rancher Kubernetes Metadata contains Kubernetes version information which Rancher uses to provision RKE clusters.

For more information on how metadata works and how to configure metadata config, see Rancher Kubernetes Metadata.

# Global Settings

Options that control certain global-level Rancher settings are available from the top navigation bar.

Click ☰ in the top left corner, then select **Global Settings**, to view and configure the following settings:

- **Settings**: Various Rancher defaults, such as the minimum length for a user's password (`password-min-length`). You should be cautious when modifying these settings, as invalid values may break your Rancher installation.
- **Feature Flags**: Rancher features that can be toggled on or off. Some of these flags are for experimental features.
- **Banners**: Elements you can add to fixed locations on the portal. For example, you can use these options to set a custom banner for users when they login to Rancher.
- **Branding**: Rancher UI design elements that you can customize. You can add a custom logo or favicon, and modify UI colors.
- **Performance**: Performance settings for the Rancher UI, such as incremental resource loading.
- **Home Links**: Links displayed on the Rancher UI **Home** page. You can modify visibility for the default links or add your own links.

### Enabling Experimental Features

Rancher includes some features that are experimental and/or disabled by default. Feature flags allow you to enable these features. For more information, refer to the section about feature flags.

# SUSE Customer Center Registration

SUSE Rancher Prime customers can register Rancher Manager with the SUSE Customer Center (SCC) [https://scc.suse.com/home].

Registering your Rancher Manager instance with the SUSE Customer Center (SCC) unlocks unified visibility

and control over your enterprise infrastructure, giving you a single pane of glass to track subscription usage alongside your SLES landscape. While your software currently continues to function seamlessly with full capabilities and zero operational lockouts if you choose not to register, completing this step now ensures you get the full management benefits, insights, and compliance backing for your ecosystem.

When logging in to SCC, you may see multiple registration codes listed under your organization. Ensure you select a code corresponding to an active SUSE Rancher Prime subscription [https://scc.suse.com/docs/userguide#UG-Activating-and-Managing-Subscriptions].



There is no limit on the number of Rancher Manager instances that can be registered using the same registration code, and each instance is registered independently.

## Prerequisites

- Registering your Rancher Manager instance with SCC requires a SUSE account. If you don't have a SUSE account yet, create one [https://www.suse.com/account/create/].
- You must have access to your organization. Ask your organization's administrator to add you to the account. See the SCC documentation [https://scc.suse.com/docs/userguide#UG-Requesting-Access-to-an-Organizations-Account] and SCC FAQ [https://scc.suse.com/docs/help#how-to-add-user] for details. If you know your organization's registration code, enter it when prompted.

## Register with SUSE Customer Center

From the Rancher UI, go to **Global Settings** > **Registration**. You can also check the Notification Center (bell icon) in the upper-right corner for a "Rancher Prime Registration" notification and select **Register Now**. This redirects you to the **Registration** page under **Global Settings**.

You have the option to register your Rancher Manager instance with SCC online or offline.

### Online Registration

You can register online through the Rancher UI or with Helm.

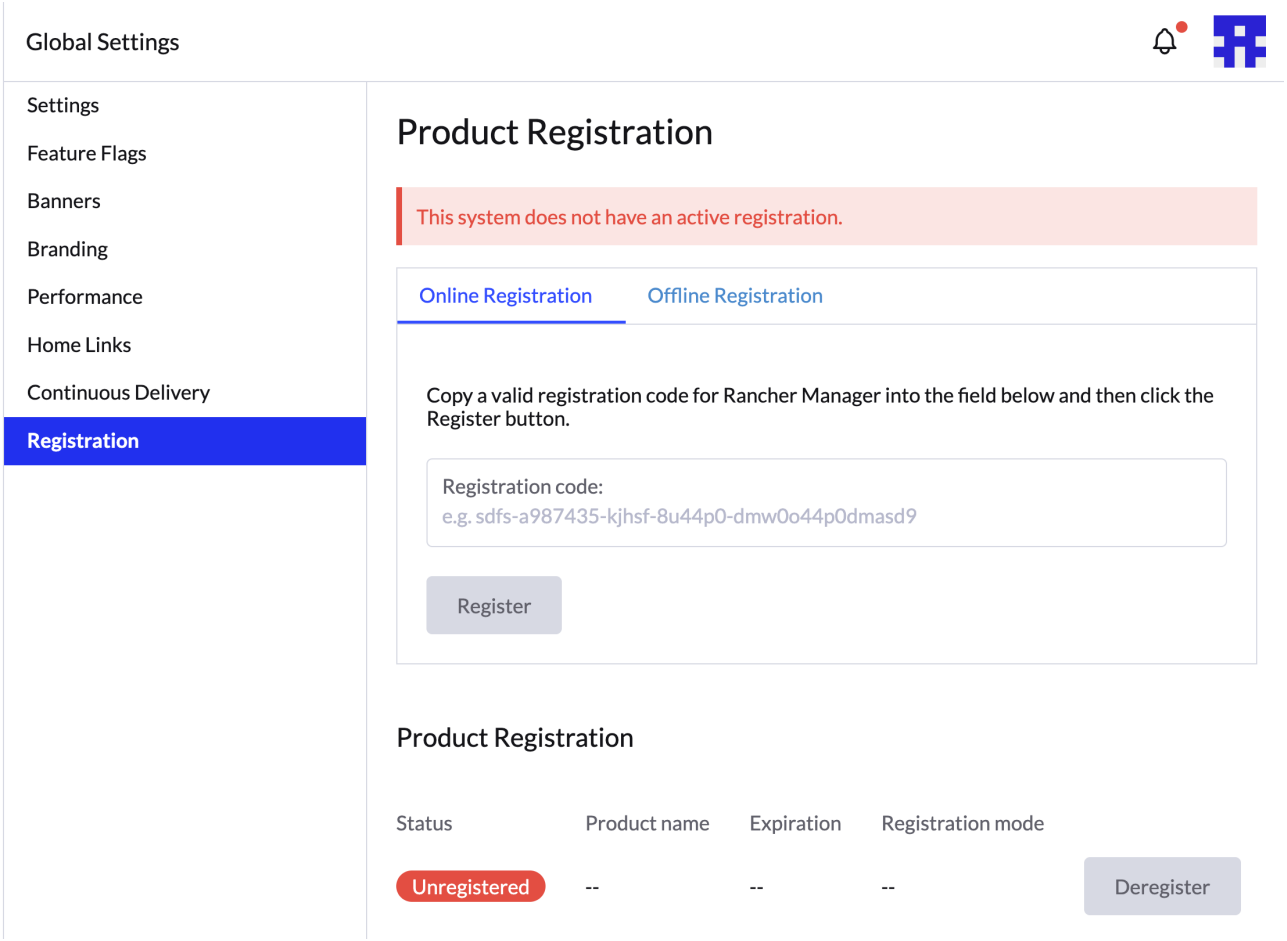
#### Obtain Registration Code

Before proceeding, you must obtain the registration code for your Rancher Manager subscription on SCC. To get the registration code:

1. Log in to SCC.
2. Navigate to **My Organizations** and select the organization with your Rancher Manager subscription.
3. Find your Registration code under **Organization** > **Subscriptions** > **Subscription Information**.

#### Rancher UI

1. From the Rancher UI, copy the registration code into the **Registration code** field and select **Register**. If successful, the **Status** becomes **Active**, and the **Product Name**, **Expiration date** and **Registration mode** are displayed.




**Helm**

After you have obtained your registration code from SCC, you can register your Rancher Manager instance through Helm.

**Entrypoint Secret**

To follow security best practices in production or GitOps environments, avoid storing registration codes in plain text. Instead, manually create an entrypoint secret before installation to allow the SCC Operator to detect and consume the registration data automatically.

1. Create the SCC system namespace. The SCC Operator expects registration data to reside in the `cattle-scc-system` namespace.

```
kubectl create namespace cattle-scc-system
```

2. Create a secret named `scc-registration` containing your registration code. This secret can be managed through external secret managers to keep it encrypted in Git.

```
kubectl create secret generic scc-registration \
  --namespace cattle-scc-system \
  --from-literal=regCode=<REGISTRATION-CODE-FROM-SCC>
```

3. Enable registration in your `values.yaml` but leave the `regCode` field empty. When

`registration.enabled` is `true` and no code is provided in the values, the operator looks for an existing `scc-registration` secret in the `cattle-scc-system` namespace.

```
registration:
  enabled: true
  regCode: ""
```

4. Deploy Rancher using your updated `values.yaml`:

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --values values.yaml
```

5. To verify that the registration was successful using the manually created secret, check the logs of the SCC-Operator or the status of the registration in the Rancher UI under **Global Settings** > **Registration**.

#### Helm Parameters

For testing or non-production environments, you can pass the registration code directly during the Helm install command:

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set registration.enabled=true \
  --set registration.regCode=<REGISTRATION-CODE-FROM-SCC>
```

Alternatively, you can update `values.yaml` to enable registration and add your registration code.

```
registration:
  enabled: true
  regCode: "<REGISTRATION-CODE>"
```

Then, deploy Rancher using the updated values file:

```
helm install rancher rancher-prime/rancher
  --namespace cattle-system
  --values values.yaml
```

### Offline Registration

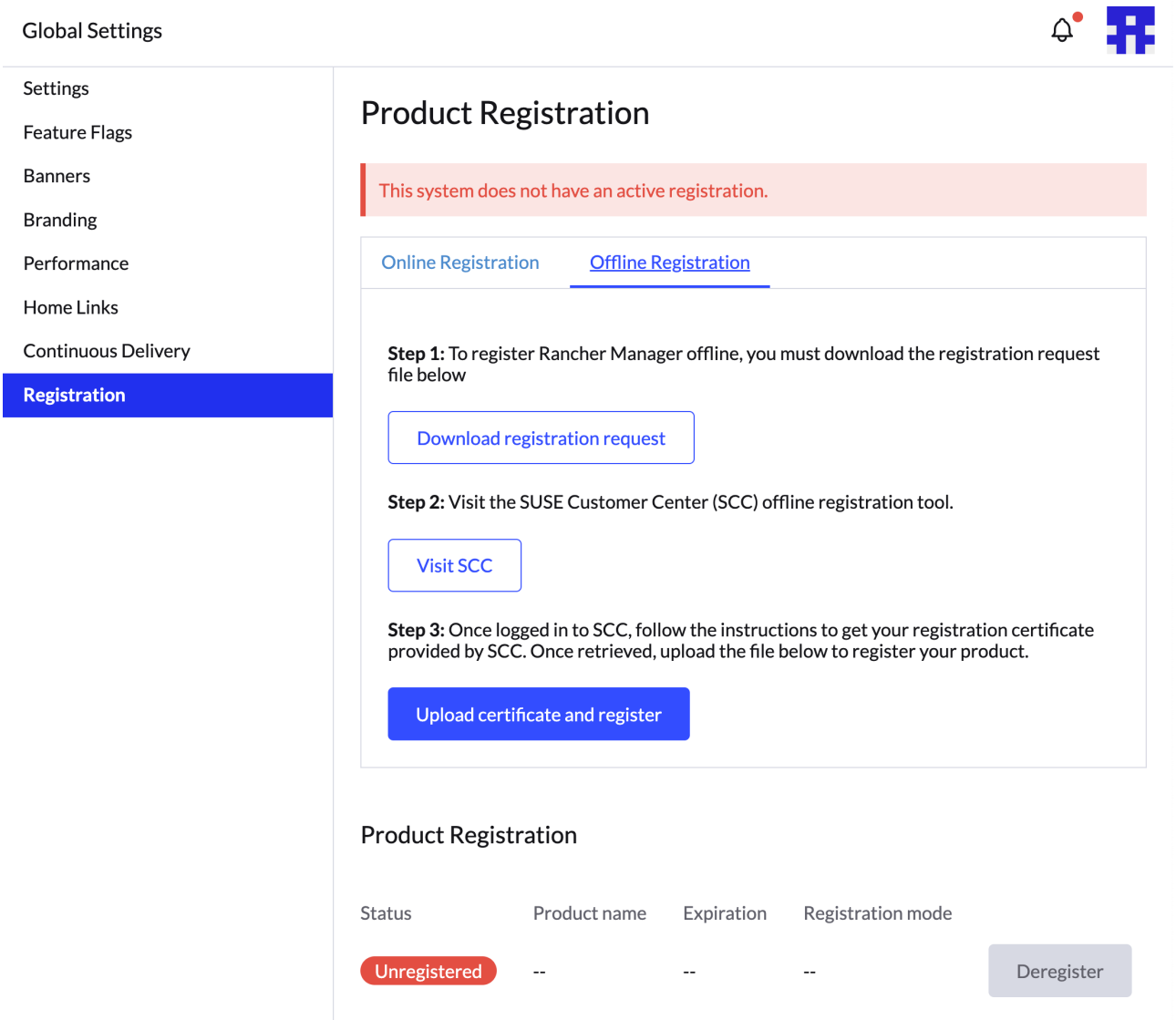
If you are using Rancher in an air-gapped environment, you can register offline through the Rancher UI or with Helm.

**Rancher UI**



An SCC Organization Admin must get your registration certificate, as regular users do not have access.

1. From the Rancher UI, select **Download registration request** to download the registration request file.
2. Select **Visit SCC** to use the SCC offline registration tool and get the registration certificate.
   a. Upload the registration request file.
   b. Select your **Organization** and **Subscription**.
   c. Select **Download Offline Certificate** and save the registration certificate file.
3. Return to the Rancher UI, select **Upload certificate and register** to upload the registration certificate. If successful, the **Status** becomes **Active**, and the **Product Name**, **Expiration date** and **Registration mode** are displayed.




**Helm**

1. In `values.yaml`, enable registration, comment out `regCode` and set the `mode` to `offline`.

```
registration:
```

```
  enabled: true
  # regCode: ""
  mode: offline
```

2. Run the Helm install command `helm install rancher rancher-prime/rancher` and verify installation is successful.
3. Before proceeding, use the following Bash command to check that the SCC-Operator has successfully created a secret in the `cattle-scc-system` namespace. After the secret is created, it outputs `Secret created!`.

```
for i in {1..100}; do kubectl wait --for=create secret/scc-
registration -n cattle-scc-system --timeout=3s &> /dev/null && echo
"Secret created!" && break || { echo "Not ready yet..."; sleep 3; };
done
```

4. Using `kubectl`, retrieve the registration request file and save it locally.

```
kubectl get secrets -n cattle-scc-system \
-l scc.cattle.io/secret-role=offline-request,scc.cattle.io/scc-hash
=$(kubectl get secrets \
-n cattle-scc-system scc-registration -o yaml|yq
'.metadata.labels["scc.cattle.io/scc-hash"]') \
-o yaml | yq '.items[0].data.request' > offline.base64
```

5. Select **Visit SCC** to use the SCC offline registration tool.
   a. Upload the registration request file.
   b. Select your **Organization** and **Subscription**.
   c. Select **Download Offline Certificate** and save the registration certificate file.
6. Upload the registration certificate file to your Rancher Manager instance using `kubectl` or Helm.

   **`kubectl`:**

   ```
   kubectl patch secret -n cattle-scc-system scc-registration \
   --type='merge' -p "{\"data\": {\"certificate\": \"$(cat <your
   Offline Cert file> )\"}}"
   ```

   **Helm:**

   Add your registration certificate using the `certificate` value field, and then `helm upgrade` Rancher.

   ```
   certificate: {}
   ```

## Post-Registration

After registering your Rancher Manager instance with SCC, several updates occur across both the local Rancher Manager instance and the central SUSE Customer Center.

### Rancher Application

- **Status Update:** Under **Global Settings** > **Registration**, the subscription status updates to **Active**, displaying your Product Name, Expiration Date, and Registration Mode.
- **Background Telemetry Sync:** The embedded `scc-operator` begins transmitting technical metrics (such as downstream cluster counts, vCPU counts, node capacity, and the server URL) to SCC at regular intervals over HTTPS.

### Viewing Registered Instances in SCC

You can centrally manage and track your organization's deployed Rancher Manager instances in SCC.

To view registered Rancher Manager instances in SCC:

1. Log in to SUSE Customer Center [https://scc.suse.com] and select your organization.
2. Select **Systems** in the top navigation menu.
3. Locate your relevant Rancher Manager instance in the system list and select the instance name to open a detailed overview.

Alternatively, you can navigate to the **Subscriptions** view in the top navigation menu, select your **SUSE Rancher Prime** subscription, and view the **Registered Systems** to view all instances under that specific subscription.

## Deregister from SUSE Customer Center

Organization administrators can deregister Rancher Manager instances from SCC. Deregistering a system removes it from being tracked in the SCC server inventory.



If you deregister a system behind the Repository Mirroring Tool (RMT) registration proxy, you must also remove the system from the proxy itself. Otherwise, active systems will reappear in SCC.

### Online Deregistration Steps

1. Go to **Global Settings** > **Registration**.
2. Find your Product Registration under **Online Registration** or **Offline Registration** and select **Deregister**.

### Offline Deregistration Steps

1. Log in to SUSE Customer Center [https://scc.suse.com] and your Rancher application.
2. In the Rancher UI, select **Cluster Management**, then select your local cluster.
3. Search for `Registration` on your local cluster using the search field to find the Registration object.
4. View the Registration objects YAML to identify the SCC System ID (`.status.sccSystemID`).
5. From the SCC portal, select the **Systems** tab and locate the matching Rancher Manager instance using

the SCC System ID.

6. Select **Deregister** to remove the system from SCC.
7. Return to Rancher and navigate to **Global Settings** > **Registration**.
8. Select **Deregister** to remove the registration from Rancher.

### Telemetry

To improve service and stability, your Rancher Manager server securely sends technical telemetry, including deployment details, version, and usage metrics to SUSE.

The key benefit of this is gaining a unified view in the SUSE Customer Center, allowing you to manage your Rancher installations, monitor subscription compliance, and track subscription usage alongside your existing SLES landscape in a single, comprehensive pane.

The following table details the system attributes included in the telemetry data.

| System attribute | Example |
|---|---|
| Product name | Rancher Manager Prime |
| Product version | 2.15.3 |
| Architecture | x86_64 |
| Server URL | The management URL of the RMS |
| Cluster UUID | UUID of the k8s cluster hosting the RMS instance |
| Install UUID | UUID of the RMS instance |
| Git hash | Hash of the RMS build |
| Feature Flags | Enabled on the RMS instance |

## User Management

### Configuring Authentication

One of the key features that Rancher adds to Kubernetes is centralized user authentication. This feature allows your users to use one set of credentials to authenticate with any of your Kubernetes clusters.

This centralized user authentication is accomplished using the Rancher authentication proxy, which is installed along with the rest of Rancher. This proxy authenticates your users and forwards their requests to your Kubernetes clusters using a service account.



The account used to enable the external provider will be granted admin permissions. If you use a test account or non-admin account, that account will still be granted admin-level permissions. See External Authentication Configuration and Principal Users to understand why.

#### External vs. Local Authentication

External authentication provides centralized, enterprise-grade security through a third-party identity provider, while local authentication serves small test environments.

- **External Authentication (Recommended for Production)**: The primary and preferred method for

managing user access. External providers offer the robust security controls, strict password requirements, and centralized identity management necessary for enterprise environments.

- **Local Authentication (Small Deployments)**: Rancher also provides local authentication. This is a built-in provider active by default. It is intended strictly for small deployments and testing. In production environments, local authentication should only be used as an emergency backup to ensure you can access Rancher if your external authentication provider goes down.

The Rancher authentication proxy integrates with the following external authentication services.

| Auth Service |
| --- |
| Microsoft Active Directory |
| GitHub |
| Microsoft Entra ID |
| FreeIPA |
| OpenLDAP |
| Microsoft AD FS |
| PingIdentity |
| Keycloak (OIDC) |
| Keycloak (SAML) |
| Okta |
| Google OAuth |
| Shibboleth |
| Generic (OIDC) |

#### Users and Groups



- Local authentication does not support creating or managing groups.
- After an external authentication provider is configured, note that local Rancher scoped administrative users only display resources such as users and groups that they are a member of in the respective authentication provider.

Rancher relies on users and groups to determine who is allowed to log in to Rancher and which resources they can access. When authenticating with an external provider, groups are provided from the external provider based on the user. These users and groups are given specific roles to resources like clusters and projects. When you give access to a group, all users who are a member of that group in the authentication provider will be able to access the resource with the permissions that you've specified. For more information on roles and permissions, see Role Based Access Control.

For more information, see Users and Groups.

#### Scope of Rancher Authorization

After you configure Rancher to allow sign on using an external authentication service, you should configure who should be allowed to log in and use Rancher. The following options are available:

| Access Level | Description |
|---|---|
| Allow any valid Users | *Any* user in the authorization service can access Rancher. We generally discourage use of this setting! |
| Allow members of Clusters, Projects, plus Authorized Users and Organizations | Any user in the authorization service and any group added as a **Cluster Member** or **Project Member** can log in to Rancher. Additionally, any user in the authentication service or group you add to the **Authorized Users and Organizations** list may log in to Rancher. |
| Restrict access to only Authorized Users and Organizations | Only users in the authentication service or groups added to the Authorized Users and Organizations can log in to Rancher. |



Only trusted admin-level users should have access to the local cluster, which manages all of the other clusters in a Rancher instance. Rancher is directly installed on the local cluster, and Rancher's management features allow admins on the local cluster to provision, modify, connect to, and view details about downstream clusters. Since the local cluster is key to a Rancher instance's architecture, inappropriate access carries security risks.

To set the Rancher access level for users in the authorization service, follow these steps:

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Auth Provider**.
3. After setting up the configuration details for an auth provider, use the **Site Access** options to configure the scope of user authorization. The table above explains the access level for each option.
4. Optional: If you choose an option other than **Allow any valid Users,** you can add users to the list of authorized users and organizations by searching for them in the text field that appears.
5. Click **Save**.

**Result:** The Rancher access configuration settings are applied.



*SAML Provider Caveats*

- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

#### External Authentication Configuration and Principal Users

Configuring external authentication requires:

- A local user assigned the administrator role, called hereafter the *local principal*.
- An external user that can authenticate with your external authentication service, called hereafter the *external principal*.

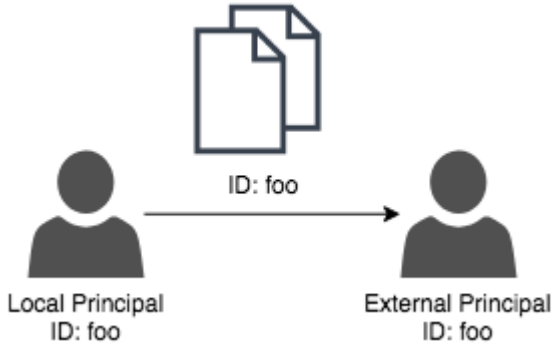
The configuration of external authentication also affects how principal users are managed within Rancher. Specifically, when a user account enables an external provider, it is granted admin-level permissions. This is because the local principal and external principal share the same user ID and access rights.

The following instructions demonstrate these effects:

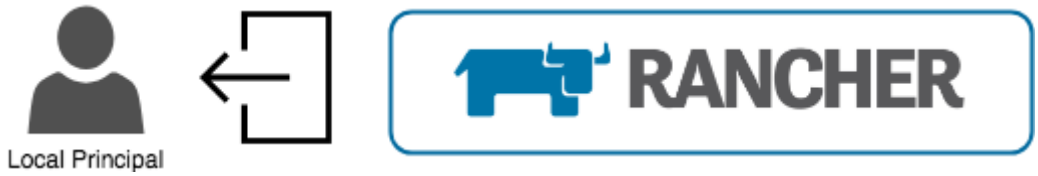
1. Sign into Rancher as the local principal and complete configuration of external authentication.



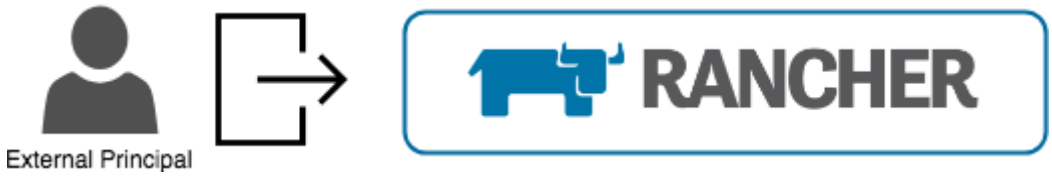

2. Rancher associates the external principal with the local principal. These two users share the local principal's user ID.



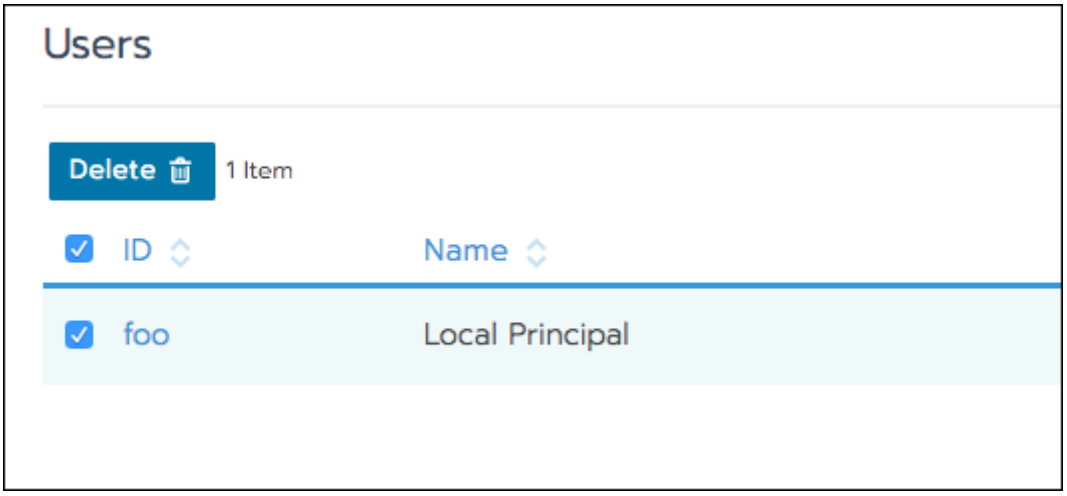

3. After you complete configuration, Rancher automatically signs out the local principal.




4. Then, Rancher automatically signs you back in as the external principal.




5. Because the external principal and the local principal share an ID, no unique object for the external principal displays on the Users page.




6. The external principal and the local principal share the same access rights.



*Reconfiguring a previously set up auth provider*

If you need to reconfigure or disable then re-enable a provider that had been previously set up, ensure that the user who attempts to do so is logged in to Rancher as an external user, not the local admin.

**Disabling An Auth Provider**

When you disable an auth provider, Rancher deletes all resources associated with it, such as:

- Secrets.
- Global role bindings.
- Cluster role template bindings.
- Project role template bindings.
- External users associated with the provider, but who never logged in as local users to Rancher.

As this operation may lead to a loss of many resources, you may want to add a safeguard on the provider. To ensure that this cleanup process doesn't run when the auth provider is disabled, add a special annotation to the corresponding auth config.

For example, to add a safeguard to the Microsoft Entra ID provider, annotate the `azuread` authconfig object:

```
kubectl annotate --overwrite authconfig azuread management.cattle.io/auth-provider-cleanup='user-locked'
```

Rancher won't perform cleanup until you set the annotation to `unlocked`.

**Running Resource Cleanup Manually**

Rancher might retain resources from a previously disabled auth provider configuration in the local cluster, even after you configure another auth provider. For example, if you used Provider A, then disabled it and started using Provider B, when you upgrade to a new version of Rancher, you can manually trigger cleanup on resources configured by Provider A.

To manually trigger cleanup for a disabled auth provider, add the `management.cattle.io/auth-`

`provider-cleanup` annotation with the `unlocked` value to its auth config.

### Users and Groups

Rancher relies on users and groups to determine who is allowed to log in to Rancher and which resources they can access. When you configure an external authentication provider, users from that provider will be able to log in to your Rancher server. When a user logs in, the authentication provider will supply your Rancher server with a list of groups to which the user belongs.

Access to clusters and projects can be controlled by adding either individual users or groups to these resources. When you add a group to a resource, all users who are members of that group in the authentication provider, will be able to access the resource with the permissions that you've specified for the group. For more information on roles and permissions, see Role Based Access Control.

#### Managing Members

When adding a user or group to a resource, you can search for users or groups by beginning to type their name. The Rancher server will query the authentication provider to find users and groups that match what you've entered. Searching is limited to the authentication provider that you are currently logged in with. For example, if you've enabled GitHub authentication but are logged in using a local user account, you will not be able to search for GitHub users or groups.

All users, whether they are local users or from an authentication provider, can be viewed and managed. In the upper left corner, click ☰ > **Users & Authentication**. In the left navigation bar, click **Users**.



*SAML Provider Caveats*

- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

#### User Information

Rancher maintains information about each user that logs in through an authentication provider. This information includes whether the user is allowed to access your Rancher server and the list of groups that the user belongs to. Rancher keeps this user information so that the CLI, API, and kubectl can accurately reflect the access that the user has based on their group membership in the authentication provider.

Whenever a user logs in to the UI using an authentication provider, Rancher automatically updates this user information.

#### Automatically Refreshing User Information

Rancher will periodically refresh the user information even before a user logs in through the UI. You can control how often Rancher performs this refresh.

Two settings control this behavior:

- **`auth-user-info-max-age-seconds`**

  This setting controls how old a user's information can be before Rancher refreshes it. If a user makes an API call (either directly or by using the Rancher CLI or kubectl) and the time since the user's last refresh is greater than this setting, then Rancher will trigger a refresh. This setting defaults to `3600` seconds, i.e. `1` hour.

- **`auth-user-info-resync-cron`**

  This setting controls a recurring schedule for resyncing authentication provider information for all users. Regardless of whether a user has logged in or used the API recently, this will cause the user to be refreshed at the specified interval. This setting defaults to `0 0 * * *`, i.e. once a day at midnight. See the Cron documentation [https://en.wikipedia.org/wiki/Cron] for more information on valid values for this setting.

To change these settings,

1. In the upper left corner, click ☰ > **Global Settings**.
2. Go to the setting you want to configure and click ☰ > **Edit Setting**.



Since SAML does not support user lookup, SAML-based authentication providers do not support periodically refreshing user information. User information will only be refreshed when the user logs into the Rancher UI.

#### Manually Refreshing User Information

If you are not sure the last time Rancher performed an automatic refresh of user information, you can perform a manual refresh of all users.

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. On the **Users** page, click on **Refresh Group Memberships**.

**Results:** Rancher refreshes the user information for all users. Requesting this refresh will update which users can access Rancher as well as all the groups that each user belongs to.



Since SAML does not support user lookup, SAML-based authentication providers do not support the ability to manually refresh user information. User information will only be refreshed when the user logs into the Rancher UI.

#### Minimum Password Length

By default, user passwords must be at least 12 characters long. However, you can customize the password length requirement:

1. In the upper left corner, click ☰ > **Global Settings**.
2. Go to **`password-min-length`** and click ☰ > **Edit Setting**.
3. Enter an integer value between 2 and 256, and click **Save**.

#### Session Length

The default length (TTL) of each user session is adjustable. The default session length is 16 hours.

1. In the upper left corner, click ☰ > **Global Settings**.
2. Go to `auth-user-session-ttl-minutes` and click ☰ > **Edit Setting**.
3. Enter the amount of time in minutes a session length should last and click **Save**.

**Result:** Users are automatically logged out of Rancher after the set number of minutes.

### Local Authentication



Rancher's local authentication provider is designed primarily to support small-scale deployments and test environments.

Enterprise customers require stronger security controls, such as advanced password policies and centralized identity management. It is highly recommended to configure an external authentication provider as your primary method for managing user access. In production environments, local authentication should only be used as an emergency backup to ensure you can access Rancher if your external authentication provider goes down.

Local authentication is the default until you configure an external authentication provider. Rancher stores user account information, such as usernames and passwords, locally. By default, the `admin` user that logs in to Rancher for the first time is a local user.

#### Adding Local Users

Regardless of whether you use external authentication, you should create a few local authentication users so that you can continue using Rancher if your external authentication service encounters issues.

1. In the top left corner, click ☰ > **Users & Authentication**.
2. In the left navigation menu, click **Users**.
3. Click **Create**.
4. Complete the **Add User** form.
5. Click **Create**.

### External Authentication

#### JSON Web Token (JWT) Authentication

Many 3rd party integrations available for Kubernetes, such as GitLab and HashiCorp Vault, involve giving an external process access to the Kubernetes API using a native Kubernetes Service Account token for authentication.

In Rancher v2.9.0 and later, service accounts on downstream clusters can now authenticate through a JSON web token (JWT) using the Rancher authentication proxy. In Rancher versions earlier than v2.9.0, only Rancher-issued tokens were supported.

To enable this feature, follow these steps:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Click **Advanced** to open the dropdown menu.

3. Select **JWT Authentication**.
4. Click the checkbox for the cluster you want to enable JWT authentication for, and click **Enable**. Alternatively, you can click ⋮ > **Enable**.

#### Configure Generic OIDC

Generic OpenID Connect (OIDC) allows users to sign in to Rancher using their credentials from their existing account at an OIDC Identity Provider (IdP). Rancher supports integration with the OIDC protocol and the SAML protocol. Both implementations are functionally equivalent when used with Rancher. The following instructions describe how to create an OIDC client and configure Rancher to work with your authentication provider. Users can then sign into Rancher using their login from the OIDC IdP.

##### Prerequisites

##### Identity Provider

In Rancher, Generic OIDC is disabled.



Consult the documentation for your specific IdP to complete the listed prerequisites.

##### OIDC Client

In your IdP, create a new client with the settings below:

| Setting | Value |
|---|---|
| `Client ID` | <CLIENT_ID>(e.g. <code>rancher</code>)</CLIENT_ID> |
| `Name` | <CLIENT_NAME>(e.g. <code>rancher</code>)</CLIENT_NAME> |
| `Client Protocol` | `openid-connect` |
| `Access Type` | `confidential` |
| `Valid Redirect URI` | `https://yourRancherHostURL/verify-auth` |

In the new OIDC client, create mappers to expose the user's fields.



The `groups` and `full_group_path` claims generated by the Groups and Group Path mappers, which you create within the OIDC client in your Identity Provider, should be JSON arrays, e.g. `"groups":["admins","devs","qa"]` and `"full_group_path":["/admins","/devs","/qa"]`.

1. Create a new `Groups Mapper` with the settings below:

| Setting | Value |
|---|---|
| `Name` | `Groups Mapper` |
| `Mapper Type` | `Group Membership` |
| `Token Claim Name` | `groups` |
| `Add to ID token` | `OFF` |
| `Add to access token` | `OFF` |
| `Add to user info` | `ON` |

2. Create a new `Client Audience` with the settings below:

| Setting | Value |
|---|---|
| `Name` | `Client Audience` |
| `Mapper Type` | `Audience` |
| `Included Client Audience` | <CLIENT_NAME></CLIENT_NAME> |
| `Add to access token` | `ON` |

3. Create a new `Groups Path` with the settings below.

| Setting | Value |
|---|---|
| `Name` | `Group Path` |
| `Mapper Type` | `Group Membership` |
| `Token Claim Name` | `full_group_path` |
| `Full group path` | `ON` |
| `Add to user info` | `ON` |



Rancher uses the value received in the "sub" claim to form the PrincipalID which is the unique identifier in Rancher. It is important to make this a value that is unique and immutable.

#### Configuring Generic OIDC in Rancher

1. In the upper left corner of the Rancher UI, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Auth Provider**.
3. Select **Generic OIDC**.
4. Complete the **Configure an OIDC account** form. For help with filling the form, see the configuration reference.
5. Click **Enable**.

   Rancher will redirect you to the IdP login page. Enter your IdP credentials to validate your Rancher Keycloak configuration.



You may need to disable your popup blocker to see the IdP login page.

**Result:** Rancher is configured to work with your provider using the OIDC protocol. Your users can now sign into Rancher using their IdP logins.

#### Custom Claim Mapping

Custom claim mapping within the Generic OIDC configuration is supported for `name`, `email` and `groups` claims. This allows you to manually map these OIDC claims when your IdP doesn't use standard names in tokens.

#### How a Custom Groups Claim Works

A custom groups claim influences how user groups work:

- If both the standard OIDC `groups` claim and the custom groups claim are present in the user's token, the custom claim supplements the list of groups provided by the standard claim.
- If there is no standard groups claim in the token, the groups listed in the custom claim will form the user's only groups.



There is no search functionality available for groups sourced from a custom claim. To assign a role to one of these groups, you must manually enter the group's exact name into the RBAC field.

#### Configuring Custom Claims

When on the **Configure an OIDC account** form:

1. Select **Add custom claims**.
2. Add your custom `name`, `email` or `groups` claims to the appropriate **Custom Claims** field.

For example, if your IdP sends `groups` in a claim called `custom_roles`, enter `custom_roles` into the **Custom Groups Claim** field. Rancher then supplements the standard OIDC `groups` claim or looks for that specific claim when processing the user's token.

#### Configuration Reference

| Field | Description |
|---|---|
| Client ID | The Client ID of your OIDC client. |
| Client Secret | The generated Secret of your OIDC client. |
| Private Key/Certificate | A key/certificate pair to create a secure shell between Rancher and your IdP. Required if HTTPS/SSL is enabled on your OIDC server. |
| Endpoints | Choose whether to use the generated values for the Rancher URL, Issue, and Auth Endpoint fields or to provide manual overrides if incorrect. |
| Rancher URL | The URL for your Rancher Server. |
| Issuer | The URL of your IdP. If your provider has discovery enabled, Rancher uses the Issuer URL to fetch all of the required URLs. |
| Auth Endpoint | The URL where users are redirected to authenticate. |

#### Custom Claims

| Custom Claim Field | Default OIDC Claim | Custom Claim Description |
|---|---|---|
| Custom Name Claim | `name` | The name of the claim in the OIDC token that contains the user's full name or display name. |
| Custom Email Claim | `email` | The name of the claim in the OIDC token that contains the user's email address. |
| Custom Groups Claim | `groups` | The name of the claim in the OIDC token that contains the user's group memberships (used for RBAC). |

#### OIDC Support for PKCE Extension

Rancher supports the Proof Key for Code Exchange (PKCE) extension (RFC 7636) for OIDC authentication providers. SHA-256 (S256) is the only supported PKCE verification method. To enable this feature, your authentication provider must use PKCE with S256 for authorization requests.

You can enable this feature by selecting **Enable PKCE (S256)** in your authentication provider configuration in Rancher. Enabling S256 PKCE token verification allows you to mitigate authorization code interception attacks during OIDC authentication flows.

#### Configuring OIDC Single Logout (SLO)

Rancher supports the ability to configure OIDC Single Logout (SLO). Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options.

#### Prerequisites

Before configuring OIDC SLO, ensure the following is set up on your IdP:

- **SLO Support**: The **Log Out behavior** configuration section only appears if your OIDC IdP allows for `OIDC SLO`.
- **Post-Logout Redirect URI**: Your Rancher Server URL must be configured as an authorized post-logout redirect URI in your IdP's OIDC client settings. This URL is used by the IdP to redirect a user back to Rancher after a successful external logout.

#### OIDC SLO Configuration

Configure the SLO settings when setting up or editing your OIDC authentication provider.

1. Sign in to Rancher using a standard user or an administrator role.
2. In the top left corner, select ☰ > **Users & Authentication**.
3. In the left navigation menu, select **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

5. If you choose to log out of your IdP, provide an **End Session Endpoint**. Rancher uses this URL to initiate the external logout.

#### How to get the End Session Endpoint

The `end_session_endpoint` is one of the specific URLs published within a standardized JSON object containing the IdP's metadata and is retrieved from the OIDC Discovery URL. To get the `end_session_endpoint` from the OIDC Discovery URL, follow these steps:

1. Obtain the Discovery URL by appending the IdP Issuer URL with the well-known path (`.well-known/openid-configuration`).
2. Send an HTTP `GET` request to the Discovery URL.
3. In the JSON object, look for the key named `end_session_endpoint` and retrieve the URL.

   You can also use a `curl` command to retrieve `end_session_endpoint`:

   ```
   curl -s <ISSUER_URL>/.well-known/openid-configuration | jq
   '.end_session_endpoint'
   ```

#### Troubleshooting

If you are experiencing issues while testing the connection to the OIDC server, first double-check the configuration options of your OIDC client. You can also inspect the Rancher logs to help pinpoint what's causing issues. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

All Generic OIDC related log entries are prepended with either `[generic oidc]` or `[oidc]`.

#### You are not redirected to your authentication provider

If you fill out the **Configure a Generic OIDC account** form and click on **Enable**, and you are not redirected to your IdP, verify your OIDC client configuration.

#### The generated `Issuer` and `Auth Endpoint` are incorrect

If the `Issuer` and `Auth Endpoint` are generated incorrectly, open the **Configure an OIDC account** form, change **Endpoints** to `Specify (advanced)` and override the `Issuer` value.

#### Error: "Invalid grant_type"

In some cases, the "Invalid grant_type" error message may be misleading and is actually caused by setting the `Valid Redirect URI` incorrectly.

#### Configure Rancher as an OIDC provider

Rancher can act as an OpenID Connect (OIDC) Identity Provider (IdP) for other applications. This allows you to use Rancher's centralized authentication and role-based access control (RBAC) to manage access to external, third-party applications. This can be used for enabling single sign-on (SSO) across Rancher components. For example, see the documentation [https://documentation.suse.com/cloudnative/suse-observability/latest/en/setup/security/authentication/oidc.html] for configuring the OIDC provider for SUSE Observability.



Because OIDC is a superset of OAuth2, you can use Rancher as an OAuth2 server without requiring full OIDC. This ensures that clients utilizing the OAuth2 aspect, such as the `rancher-ai-mcp` server, are fully supported.

The Rancher OIDC Provider issues access tokens for OAuth2 and OIDC that can be used as standard Bearer tokens (per RFC6750) to authenticate with Rancher. Previously, only an ID token could be used to impersonate and authenticate a user.

The OIDC provider can be enabled with the `oidc-provider` feature flag. When this flag is on the following

endpoints are available:

- `https://{rancher-url}/oidc/authorize`: This endpoint initiates the authentication flow. If a user is already logged into Rancher, it returns an authorization code. Otherwise, it redirects the user to the Rancher login page. Authorization codes and related request information are securely stored in session secrets. Codes are single-use and expire after 10 minutes.
- `https://{rancher-url}/oidc/token`: This endpoint exchanges an authorization code for an `id_token`, `access_token`, and `refresh_token`.
- `https://{rancher-url}/oidc/.well-known/openid-configuration`: This endpoint returns a JSON document containing the OIDC provider's configuration, including endpoint URLs, supported scopes, claims, and other relevant details.
- `https://{rancher-url}/oidc/userinfo`: This endpoint provides information about the authenticated user.

The OIDC provider supports the OIDC Authentication Code Flow with PKCE.

#### Configuring an OIDC Client

An `OIDCClient` represents an external application that will be authenticating against Rancher. To register a client application, you must create an `OIDCClient` custom resource.

#### Configuration Fields

When defining your `OIDCClient` manifest, you must include specific fields to pass CRD validation:

- `spec.tokenExpirationSeconds`: This field is strictly required and will cause a validation error if omitted. It defines the lifespan of the access token.
- `spec.refreshTokenExpirationSeconds`: This field is also strictly required and will cause a validation error if omitted. It defines the lifespan of the refresh token.
- `scopes` (Optional): This field allows you to restrict the scopes that a client can request. If not explicitly configured, the allowed scopes will default to `openid`, `profile`, and `offline_access`.

#### Example OIDC Client Manifest

Below is an example of an `OIDCClient` configuration:



You must include the expiration fields to successfully apply the resource.

```
apiVersion: management.cattle.io/v3
kind: OIDCClient
metadata:
  name: example-client
spec:
  description: "Example OIDC Client"
  redirectUris:
    - "https://example-app.com/callback"
  tokenExpirationSeconds: 3600
  refreshTokenExpirationSeconds: 86400
```

```
  # scopes:
  #   - openid
  #   - profile
  #   - offline_access
```

Save this configuration to a file (e.g., `oidcclient.yaml`) and apply it to your Rancher local cluster:

```
kubectl apply -f oidcclient.yaml
```

Rancher automatically generates a client ID and client secret for each `OIDCClient`. Once the resource is created, Rancher populates the status field with the client id:

```
apiVersion: management.cattle.io/v3
kind: OIDCClient
metadata:
  name: oidc-client-test
spec:
  tokenExpirationSeconds: 600 # expiration of the id_token and
access_token
  refreshTokenExpirationSeconds: 3600 # expiration of the refresh_token
  redirectURIs:
    - "https://myredirecturl.com" # replace with your redirect url
  scopes: # Optional: Restricts the scopes the client can request.
Defaults to openid, profile, and offline_access if omitted.
    - openid
    - profile
    - offline_access
status:
  clientID: client-xxx
  clientSecrets:
    client-secret-1:
      createdAt: "xxx"
      lastFiveCharacters: xxx
```

Rancher automatically generates a Kubernetes `Secret` in the `cattle-oidc-client-secrets` namespace for each `OIDCClient` resource. The Secret's name matches the `OIDCClient` client ID. Initially, the `Secret` contains a single client secret.

To retrieve the client secret:

```
kubectl get secret client-xxx -n cattle-oidc-client-secrets -o jsonpath
="{.data.client-secret-1}" | base64 -d
```

Output:

```
secret-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

You can now use this client ID and client secret in your OIDC client application.

#### Managing Client Secrets

You can manage multiple client secrets per `OIDCClient`. Use annotations on the `OIDCClient` resource to perform secret operations:

- Creation: Adding the `cattle.io/oidc-client-secret-create: true` annotation triggers the creation of a new client secret.
- Removal: Adding the `cattle.io/oidc-client-secret-remove:client-secret-1` annotation removes the specified client secrets.
- Regeneration: Adding the `cattle.io/oidc-client-secret-regenerate:client-secret-1` annotation regenerates the specified client secrets.

#### Rancher UI

Create an OIDCClient:

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **OIDC Apps**.
3. Click **Add Application**. Fill out the **Create OIDC App** form.
4. Click **Add Application**.

#### Managing Client Secrets

In the OIDC App page:

- Creation: Click **Add new secret**.
- Removal: Click ☰ **> Delete**.
- Regeneration: Click ☰ **> Regenerate**.

#### Signing key

A default key pair for signing the `id_token`, `access_token`, and `refresh_token` tokens is created by Rancher in a `Secret` called `oidc-signing-key` in the `cattle-system` namespace. Only one key will be used for signing, but multiple public keys can be returned in the JWKS endpoint in order to avoid disruption when doing a key rotation.

#### Rotation without disruption

In order to create a new key pair for signing you need to manually create a new keypair and add it to the `oidc-signing-key Secret`.

Example:

```
apiVersion: v1
```

```
kind: Secret
metadata:
 name: oidc-signing-key
type: Opaque
data:
 key2.pem: <base64-encoded-new-private-key>
 key1.pub: <base64-encoded-old-public-key>
 key2.pub: <base64-encoded-new-public-key>
```

Rancher will sign tokens using `key2.pem`, while the JWKS endpoint will serve both `key1.pub` and `key2.pub`. This ensures a smooth key rotation from `key1` to `key2` without disrupting existing token verification. Note that only one private key (.pem) can be stored in the secret at a time, and each key pair must share the same base name, differing only by their suffix: .pem for the private key and .pub for the public key.

#### Rotation with disruption

Removing the `oidc-signing-key Secret` will cause Rancher to regenerate the signing key on the next restart.



This will invalidate all previously issued `id_token`, `access_token`, and `refresh_token` tokens making them unusable.

#### Configure Active Directory (AD)

If your organization uses Microsoft Active Directory as central user repository, you can configure Rancher to communicate with an Active Directory server to authenticate users. This allows Rancher admins to control access to clusters and projects based on users and groups managed externally in the Active Directory, while allowing end-users to authenticate with their AD credentials when logging in to the Rancher UI.

Rancher uses LDAP to communicate with the Active Directory server. The authentication flow for Active Directory is therefore the same as for the OpenLDAP authentication integration.



Before you start, please familiarise yourself with the concepts of External Authentication Configuration and Principal Users.

#### Prerequisites

You'll need to create or obtain from your AD administrator a new AD user to use as service account for Rancher. This user must have sufficient permissions to perform LDAP searches and read attributes of users and groups under your AD domain.

Usually a (non-admin) **Domain User** account should be used for this purpose, as by default such user has read-only privileges for most objects in the domain partition.

Note however, that in some locked-down Active Directory configurations this default behaviour may not apply. In such case you will need to ensure that the service account user has at least **Read** and **List Content** permissions granted either on the Base OU (enclosing users and groups) or globally for the domain.



*Using TLS?*

- If the certificate used by the AD server is self-signed or not from a recognized certificate authority, make sure have at hand the CA certificate (concatenated with any intermediate certificates) in PEM format. You will have to paste in this certificate during the configuration so that Rancher is able to validate the certificate chain.
- Upon an upgrade to v2.6.0, authenticating via Rancher against an active directory using TLS can fail if the certificates on the AD server do not support SAN attributes. This is a check enabled by default in Go v1.15.
  - The error received is "Error creating SSL connection: LDAP Result Code 200 "Network Error": x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0".
  - To resolve the error, update or replace the certificates on the AD server with new ones that support the SAN attribute. Alternatively, this error can be ignored by setting `GODEBUG=x509ignoreCN=0` as an environment variable to Rancher server container.

#### Configuration Steps

##### Open Active Directory Configuration

1. Log into the Rancher UI using the initial local `admin` account.
2. In the top left corner, click **☰ > Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **ActiveDirectory**. The **Authentication Provider: ActiveDirectory** form will be displayed.
5. Fill out the form. For help, refer to the details on configuration options below.
6. Click **Enable**.

##### Configure Active Directory Server Settings

In the section titled `1. Configure an Active Directory server`, complete the fields with the information specific to your Active Directory server. Please refer to the following table for detailed information on the required values for each parameter.



If you are unsure about the correct values to enter in the user/group Search Base field, please refer to Identify Search Base and Schema using ldapsearch.

**Table 1: AD Server parameters**

| Parameter | Description |
|---|---|
| Hostname | Specify the hostname or IP address of the AD server |
| Port | Specify the port at which the Active Directory server is listening for connections. Unencrypted LDAP normally uses the standard port of 389, while LDAPS uses port 636. |
| TLS | Check this box to enable LDAP over SSL/TLS (commonly known as LDAPS). |
| Server Connection Timeout | The duration in number of seconds that Rancher waits before considering the AD server unreachable. |

| Parameter | Description |
| --- | --- |
| Service Account Username | Enter the username of an AD account with read-only access to your domain partition (see Prerequisites). The username can be entered in NetBIOS format (e.g. "DOMAIN\serviceaccount") or UPN format (e.g. "serviceaccount@domain.com [mailto:serviceaccount@domain.com]"). |
| Service Account Password | The password for the service account. |
| Default Login Domain | When you configure this field with the NetBIOS name of your AD domain, usernames entered without a domain (e.g. "jdoe") will automatically be converted to a slashed, NetBIOS logon (e.g. "LOGIN_DOMAIN\jdoe") when binding to the AD server. If your users authenticate with the UPN (e.g. "jdoe@acme.com [mailto:jdoe@acme.com]") as username then this field **must** be left empty. |
| User Search Base | The Distinguished Name of the node in your directory tree from which to start searching for user objects. All users must be descendents of this base DN. For example: "ou=people,dc=acme,dc=com". |
| Group Search Base | If your groups live under a different node than the one configured under `User Search Base` you will need to provide the Distinguished Name here. Otherwise leave it empty. For example: "ou=groups,dc=acme,dc=com". |

#### Configure User/Group Schema

In the section titled `2. Customize Schema` you must provide Rancher with a correct mapping of user and group attributes corresponding to the schema used in your directory.

Rancher uses LDAP queries to search for and retrieve information about users and groups within the Active Directory. The attribute mappings configured in this section are used to construct search filters and resolve group membership. It is therefore paramount that the provided settings reflect the reality of your AD domain.



If you are unfamiliar with the schema used in your Active Directory domain, please refer to Identify Search Base and Schema using ldapsearch to determine the correct configuration values.

#### User Schema

The table below details the parameters for the user schema section configuration.

**Table 2: User schema configuration parameters**

| Parameter | Description |
| --- | --- |
| Object Class | The name of the object class used for user objects in your domain. If defined, only specify the name of the object class - *don't* include it in an LDAP wrapper such as &(objectClass=xxxx) |
| Username Attribute | The user attribute whose value is suitable as a display name. |

| Parameter | Description |
|---|---|
| Login Attribute | The attribute whose value matches the username part of credentials entered by your users when logging in to Rancher. If your users authenticate with their UPN (e.g. "jdoe@acme.com [mailto:jdoe@acme.com]") as username then this field must normally be set to `userPrincipalName`. Otherwise for the old, NetBIOS-style logon names (e.g. "jdoe") it's usually `sAMAccountName`. |
| User Member Attribute | The attribute containing the groups that a user is a member of. |
| Search Attribute | When a user enters text to add users or groups in the UI, Rancher queries the AD server and attempts to match users by the attributes provided in this setting. Multiple attributes can be specified by separating them with the pipe ("\|") symbol. To match UPN usernames (e.g. jdoe@acme.com [mailto:jdoe@acme.com]) you should usually set the value of this field to `userPrincipalName`. |
| Search Filter | This filter gets applied to the list of users that is searched when Rancher attempts to add users to a site access list or tries to add members to clusters or projects. For example, a user search filter could be `(\|(memberOf=CN=groupl,CN=Users,DC=testad,DC=rancher,DC=io)(memberOf=CN=group2,CN=Users,DC=testad,DC=rancher,DC=io))`. Note: If the search filter does not use valid AD search syntax, [https://docs.microsoft.com/en-us/windows/win32/adsi/search-filter-syntax] the list of users will be empty. |
| User Enabled Attribute | The attribute containing an integer value representing a bitwise enumeration of user account flags. Rancher uses this to determine if a user account is disabled. You should normally leave this set to the AD standard `userAccountControl`. |
| Disabled Status Bitmask | This is the value of the `User Enabled Attribute` designating a disabled user account. You should normally leave this set to the default value of "2" as specified in the Microsoft Active Directory schema (see here [https://docs.microsoft.com/en-us/windows/desktop/adschema/a-useraccountcontrol#remarks]). |

#### Group Schema

The table below details the parameters for the group schema configuration.

**Table 3: Group schema configuration parameters**

| Parameter | Description |
|---|---|
| Object Class | The name of the object class used for group objects in your domain. If defined, only specify the name of the object class - *don't* include it in an LDAP wrapper such as &(objectClass=xxxx) |
| Name Attribute | The group attribute whose value is suitable for a display name. |

| Parameter | Description |
|---|---|
| Group Member User Attribute | The name of the **user attribute** whose format matches the group members in the `Group Member Mapping Attribute`. |
| Group Member Mapping Attribute | The name of the group attribute containing the members of a group. |
| Search Attribute | Attribute used to construct search filters when adding groups to clusters or projects. See description of user schema `Search Attribute`. |
| Search Filter | This filter gets applied to the list of groups that is searched when Rancher attempts to add groups to a site access list or tries to add groups to clusters or projects. For example, a group search filter could be `(\|(cn=group1)(cn=group2))`. Note: If the search filter does not use valid AD search syntax, [https://docs.microsoft.com/en-us/windows/win32/adsi/search-filter-syntax] the list of groups will be empty. |
| Group DN Attribute | The name of the group attribute whose format matches the values in the user attribute describing a the user's memberships. See `User Member Attribute`. |
| Nested Group Membership | This settings defines whether Rancher should resolve nested group memberships. Use only if your organization makes use of these nested memberships (i.e., you have groups that contain other groups as members. We advise avoiding nested groups when possible to avoid potential performance issues when there is a large amount of nested memberships). |

#### Test Authentication

Once you have completed the configuration, proceed by testing the connection to the AD server **using your AD admin account**. If the test is successful, authentication with the configured Active Directory will be enabled implicitly with the account you test with set as admin.



The AD user pertaining to the credentials entered in this step will be mapped to the local principal account and assigned administrator privileges in Rancher. You should therefore make a conscious decision on which AD account you use to perform this step.

1. Enter the **username** and **password** for the AD account that should be mapped to the local principal account.
2. Click **Authenticate with Active Directory** to finalise the setup.

**Result:**

- Active Directory authentication has been enabled.
- You have been signed into Rancher as administrator using the provided AD credentials.



You will still be able to login using the locally configured `admin` account and password in case of a disruption of LDAP services.

#### Annex: Identify Search Base and Schema using ldapsearch

In order to successfully configure AD authentication it is crucial that you provide the correct configuration pertaining to the hierarchy and schema of your AD server.

The `ldapsearch` [https://manpages.ubuntu.com/manpages/noble/en/man1/ldapsearch.1.html] tool allows you to query your AD server to learn about the schema used for user and group objects.

For the purpose of the example commands provided below we will assume:

- The Active Directory server has a hostname of `ad.acme.com`
- The server is listening for unencrypted connections on port `389`
- The Active Directory domain is `acme`
- You have a valid AD account with the username `jdoe` and password `secret`

##### Identify Search Base

First we will use `ldapsearch` to identify the Distinguished Name (DN) of the parent node(s) for users and groups:

```
$ ldapsearch -x -D "acme\jdoe" -w "secret" -p 389 \
-h ad.acme.com -b "dc=acme,dc=com" -s sub "sAMAccountName=jdoe"
```

This command performs an LDAP search with the search base set to the domain root (`-b "dc=acme,dc=com"`) and a filter targeting the user account (`sAMAccountNam=jdoe`), returning the attributes for said user:

```
# John Doe, Users, acme.com
dn: CN=John Doe,CN=Users,DC=acme,DC=com
objectClass: top
objectClass: person [1]
objectClass: organizationalPerson
objectClass: user
cn: John Doe
sn: jdoe
givenName: Doe
distinguishedName: CN=John Doe,CN=Users,DC=acme,DC=com [5]
(...)
displayName: John Doe
uSNCreated: 12791
memberOf: CN=examplegroup,OU=Groups,DC=acme,DC=com [4]
uSNChanged: 4217521
name: John Doe [2]
(...)
sAMAccountName: jdoe [3]
sAMAccountType: 805306368
userPrincipalName: jdoe@acme.com
(...)
```

Since in this case the user's DN is `CN=John Doe,CN=Users,DC=acme,DC=com` [5], we should configure the **User Search Base** with the parent node DN `CN=Users,DC=acme,DC=com`.

Similarly, based on the DN of the group referenced in the **memberOf** attribute [4], the correct value for the **Group Search Base** would be the parent node of that value, i.e., `OU=Groups,DC=acme,DC=com`.

#### Identify User Schema

The output of the above `ldapsearch` query also allows to determine the correct values to use in the user schema configuration:

- `Object Class`: **person** [1]
- `Username Attribute`: **name** [2]
- `Login Attribute`: **sAMAccountName** [3]
- `User Member Attribute`: **memberOf** [4]



If the AD users in our organization were to authenticate with their UPN (e.g. jdoe@acme.com [mailto:jdoe@acme.com]) instead of the short logon name, then we would have to set the `Login Attribute` to **userPrincipalName** instead.

| | |
|---|---|
| We'll also set the `Search Attribute` parameter to **sAMAccountName | name**. That way users can be added to clusters/projects in the Rancher UI either by entering their username or full name. |

#### Identify Group Schema

Next, we'll query one of the groups associated with this user, in this case `CN=examplegroup,OU=Groups,DC=acme,DC=com`:

```
$ ldapsearch -x -D "acme\jdoe" -w "secret" -p 389 \
-h ad.acme.com -b "ou=groups,dc=acme,dc=com" \
-s sub "CN=examplegroup"
```

This command will inform us on the attributes used for group objects:

```
# examplegroup, Groups, acme.com
dn: CN=examplegroup,OU=Groups,DC=acme,DC=com
objectClass: top
objectClass: group 1
cn: examplegroup
member: CN=John Doe,CN=Users,DC=acme,DC=com 3
distinguishedName: CN=examplegroup,OU=Groups,DC=acme,DC=com 5
instanceType: 4
whenCreated: 20180430233057.0Z
whenChanged: 20180501060714.0Z
uSNCreated: 3355225
uSNChanged: 3361522
name: examplegroup 2
objectGUID:: 7wIhB5KfXEC4j7S0gyzpNw==
objectSid:: AQUAAAAAAAUVAAAA2HMoXUhPN26Ua/qChwQAAA==
sAMAccountName: examplegroup 4
sAMAccountType: 268435457
groupType: 2
objectCategory: CN=Group,CN=Schema,CN=Configuration,DC=acme,DC=com
dSCorePropagationData: 16010101000000.0Z
```

Again, this allows us to determine the correct values to enter in the group schema configuration:

- `Object Class`: **group** [1]
- `Name Attribute`: **name** [2]
- `Group Member Mapping Attribute`: **member** [3]

- `Search Attribute`: **sAMAccountName** [4]

Looking at the value of the **member** attribute, we can see that it contains the DN of the referenced user. This corresponds to the **distinguishedName** attribute in our user object. Accordingly will have to set the value of the `Group Member User Attribute` parameter to this attribute.

In the same way, we can observe that the value in the **memberOf** attribute in the user object corresponds to the **distinguishedName** [5] of the group. We therefore need to set the value for the `Group DN Attribute` parameter to this attribute.

#### Annex: Troubleshooting

If you are experiencing issues while testing the connection to the Active Directory server, first double-check the credentials entered for the service account as well as the search base configuration. You may also inspect the Rancher logs to help pinpointing the problem cause. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

#### Configure Amazon Cognito

If your organization uses Amazon Cognito for user authentication, you can configure Rancher to allow login using Amazon Cognito credentials. The following instructions describe how to configure Rancher to work with Amazon Cognito:

#### Prerequisites

- In Rancher:
  - Amazon Cognito is disabled.



Consult the Amazon Cognito documentation [https://aws.amazon.com/cognito/getting-started/] to configure the user pool.

- In Amazon Cognito:
  - Create a new user pool or use an existing one.
  - In the `App client` settings, set the redirect URL to `https://yourRancherHostURL/verify-auth`. Replace `yourRancherHostURL` with the actual hostname of your Rancher instance (e.g., `https://rancher.example.com/verify-auth`).

# Configuring Amazon Cognito in Rancher

1. In the upper left corner of the Rancher UI, click **> Users & Authentication**.
2. In the left navigation bar, click **Auth Provider**.
3. Select **Amazon Cognito**.
4. Complete the **Configure an Amazon Cognito account** form. For help with filling the form, see the configuration reference.
5. Click **Enable**.

   Rancher redirects you to the Amazon Cognito login page. Enter your Amazon Cognito credentials to validate your Rancher configuration.

You may need to disable your popup blocker to see the Amazon Cognito login page.

**Result:** Rancher is configured to work with your Amazon Cognito using the OIDC protocol. Your users can now sign into Rancher using their Amazon Cognito logins.

User and group search is not supported for Amazon Cognito. When assigning permissions to a Project or Cluster, you must manually enter the UserID generated by Cognito if the user has not yet logged in to Rancher. However, if the user has previously logged in, you can assign permissions using their username or email address.

#### Configuration Reference

| Field | Description |
|---|---|
| Client ID | The Client ID of your Amazon Cognito App Client. |
| Client Secret | The generated Secret of your Amazon Cognito App Client. |
| Issuer | The Issuer URL of your Amazon Cognito App Client. It follows the format `https://cognito-idp.{region}.amazonaws.com/{userPoolId}`, and can be found in the App Client settings page. Rancher uses the Issuer URL to fetch all the required URLs. |

#### OIDC Support for PKCE Extension

Rancher supports the Proof Key for Code Exchange (PKCE) extension (RFC 7636) for OIDC authentication providers. SHA-256 (S256) is the only supported PKCE verification method. To enable this feature, your authentication provider must use PKCE with S256 for authorization requests.

You can enable this feature by selecting **Enable PKCE (S256)** in your authentication provider configuration in Rancher. Enabling S256 PKCE token verification allows you to mitigate authorization code interception attacks during OIDC authentication flows.

#### Configuring OIDC Single Logout (SLO)

Rancher supports the ability to configure OIDC Single Logout (SLO). Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options.

#### Prerequisites

Before configuring OIDC SLO, ensure the following is set up on your IdP:

- **SLO Support**: The **Log Out behavior** configuration section only appears if your OIDC IdP allows for `OIDC SLO`.
- **Post-Logout Redirect URI**: Your Rancher Server URL must be configured as an authorized post-logout redirect URI in your IdP's OIDC client settings. This URL is used by the IdP to redirect a user back to Rancher after a successful external logout.

#### OIDC SLO Configuration

Configure the SLO settings when setting up or editing your OIDC authentication provider.

1. Sign in to Rancher using a standard user or an administrator role.

2. In the top left corner, select ☰ > **Users & Authentication**.
3. In the left navigation menu, select **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

5. If you choose to log out of your IdP, provide an **End Session Endpoint**. Rancher uses this URL to initiate the external logout.

#### How to get the End Session Endpoint

The `end_session_endpoint` is one of the specific URLs published within a standardized JSON object containing the IdP's metadata and is retrieved from the OIDC Discovery URL. To get the `end_session_endpoint` from the OIDC Discovery URL, follow these steps:

1. Obtain the Discovery URL by appending the IdP Issuer URL with the well-known path (`.well-known/openid-configuration`).
2. Send an HTTP `GET` request to the Discovery URL.
3. In the JSON object, look for the key named `end_session_endpoint` and retrieve the URL.

   You can also use a `curl` command to retrieve `end_session_endpoint`:

```
curl -s <ISSUER_URL>/.well-known/openid-configuration | jq
'.end_session_endpoint'
```

#### Troubleshooting

##### You are not redirected to your authentication provider

If you fill out the **Configure an Amazon Cognito account** form and click on **Enable**, and you are not redirected to Amazon Cognito, verify your Amazon Cognito configuration.

#### Configure Microsoft Entra ID

##### Microsoft Entra ID (Microsoft Graph API)

Microsoft Graph API is the required flow for configuring Microsoft Entra ID (formerly Azure AD) authentication in Rancher. The sections below assist new users in configuring Microsoft Entra ID with a new instance and guide existing app owners in migrating to the Microsoft Graph API flow.

##### New User Setup

If you have a Microsoft Entra ID tenant, you can configure Rancher to allow users to log in using their

directory credentials. Configuring Microsoft Entra ID authentication requires setup steps in both the Microsoft Azure portal and Rancher.



- Microsoft Entra ID integration only supports Service Provider–initiated logins.
- Most of this procedure takes place in the Microsoft Azure Portal [https://portal.azure.com/].

#### Microsoft Entra ID Configuration Outline

Configuring Rancher to allow users to authenticate with their Microsoft Entra ID accounts involves multiple procedures. Review the outline below before getting started.



Before you start, open two browser tabs: one for Rancher, and one for the Azure portal. This will make it easier to copy and paste configuration values between the two interfaces.

#### 1. Register Rancher with Azure

Before enabling Microsoft Entra ID within Rancher, you must register Rancher as an application in your tenant.

1. Log in to the Microsoft Azure Portal [https://portal.azure.com/] as an administrative user (Global Administrator or Application Administrator).
2. Search for and select **Microsoft Entra ID** (formerly *Azure Active Directory*).
3. In the left navigation menu, select **App registrations**.
4. Click **+ New registration** and complete the form.
   a. Enter a **Name** (such as `Rancher`).
   b. Under **Supported account types**, select **Accounts in this organizational directory only (Single tenant)**.
   c. In the **Redirect URI** section, ensure **Web** is selected in the platform dropdown menu. Enter your Rancher Server URL appended with the verification path: `<MY_RANCHER_URL>/verify-auth-azure`.

      You can find your exact Redirect URI in Rancher by navigating to the **Microsoft Entra ID** authentication setup page (**Global View > Users & Authentication > Auth Provider > Microsoft Entra ID**).

   d. Click **Register**.



It can take up to five minutes for changes in Microsoft Entra ID to take effect globally.

#### 2. Create a new client secret

Create a client secret that Rancher will use to authenticate with Microsoft Entra ID.

1. From **Microsoft Entra ID > App registrations**, select your newly created Rancher application.
2. In the left navigation menu under **Manage**, click **Certificates & secrets**.
3. Select the **Client secrets** tab, then click **+ New client secret**.

4. Enter a **Description** (such as `Rancher`).
5. Select an expiration window under **Expires**.



Shorter durations are more secure but require rotating the secret more frequently. If the application secret expires, users will not be able to log in to Rancher until the secret is rotated in Azure and updated in Rancher.

6. Click **Add**.
7. Copy the string in the **Value** column immediately.



Copy the secret **Value** now. Once you navigate away from this page or refresh, the secret value will be permanently masked (••••••••••••••••). Do not copy the **Secret ID**.

#### 3. Set Required Permissions for Rancher

Next, set the required Microsoft Graph API permissions for the application registration.



Ensure that you set **Application permissions**, and *not* **Delegated permissions**. Delegated permissions will cause login failures.

1. In the left navigation menu under **Manage**, select **API permissions**.
2. Click **+ Add a permission**.
3. Select **Microsoft Graph**, then select **Application permissions**.
4. Search for and add the following permission: `Directory.Read.All`
5. Click **Add permissions**.
6. Back on the **API permissions** page, click **Grant admin consent for <Tenant Name>**, then click **Yes** to confirm.



Rancher performs user and group lookups via the Graph API and requires permissions to:

- Get a user / List all users
- Get a group / List all groups
- List group memberships for a user

Alternative permission combinations that satisfy these requirements include:

- `Directory.Read.All`
- `User.Read.All` and `GroupMember.Read.All`
- `User.Read.All` and `Group.Read.All`

#### 4. Allow Public Client Flows

To allow authentication from the Rancher CLI, enable public client flows:

1. In the left navigation menu under **Manage**, select **Authentication**.

2. Scroll to **Advanced settings**.
3. Under **Allow public client flows**, select **Yes**.
4. Click **Save**.

#### 5. Copy Microsoft Entra ID Application Data

1. Obtain your **Tenant ID**:
   a. Go to **Microsoft Entra ID > Overview** (or **App registrations** > your application's **Overview**).
   b. Copy the **Directory (tenant) ID**.
2. Obtain your **Application (Client) ID**:
   a. On your app registration's **Overview** page, copy the **Application (client) ID**.
3. Standard and China cloud environments only require entering the **Tenant ID**, **Application (Client) ID**, and **Client Secret Value**.

**For Custom Endpoints:**



Custom Endpoints are not tested or officially supported by Rancher.

To locate custom endpoint URLs:

- From your app registration's **Overview** page, click **Endpoints** in the top toolbar.
- Select the v1 or v2 endpoints matching your deployment needs:
  - **Microsoft Graph API endpoint** (Graph Endpoint)
  - **OAuth 2.0 token endpoint** (Token Endpoint)
  - **OAuth 2.0 authorization endpoint** (Auth Endpoint)

#### 6. Configure Microsoft Entra ID in Rancher

Complete the setup in the Rancher UI:

1. Log into Rancher.
2. In the top left menu, select **> Users & Authentication**.
3. In the left menu, click **Auth Provider**.
4. Click **Microsoft Entra ID**.
5. Complete the form using the values obtained in Step 5.



The account completing this setup will be granted Rancher administrator privileges. Ensure that this level of access is appropriate before proceeding.

**Standard / China Cloud Mapping:**

| Rancher Field | Microsoft Entra ID Value |
|---|---|
| Tenant ID | Directory (tenant) ID |
| Application ID | Application (client) ID |
| Application Secret | Client Secret Value |

| Rancher Field | Microsoft Entra ID Value |
|---|---|
| Endpoint | `https://login.microsoftonline.com/` |

**Custom Endpoint Mapping:**

| Rancher Field | Microsoft Entra ID Value |
|---|---|
| Graph Endpoint | Microsoft Graph API Endpoint |
| Token Endpoint | OAuth 2.0 Token Endpoint |
| Auth Endpoint | OAuth 2.0 Authorization Endpoint |



When entering a custom Graph Endpoint URL, strip the tenant ID suffix from the path (e.g., use <code><a href="https://graph.microsoft.com" class="bare">https://graph.microsoft.com</a></code> instead of <code><a href="https://graph.microsoft.com/<tenant-id&gt" class="bare">https://graph.microsoft.com/<tenant-id&gt</a>;</code>).

6. (Optional) Filter group memberships to reduce log volume and token overhead (Rancher v2.9.0+). See Filtering Users by Microsoft Entra ID Auth Group Memberships for details.
7. Click **Enable**.

**Result:** Microsoft Entra ID authentication is now active.

#### (Optional) Configure Authentication with Multiple Rancher Domains

When managing multiple Rancher domains, additional Redirect URIs must be configured manually in the underlying `azuread` authentication resource in Rancher. By default, authentication requests redirect to the primary Redirect URI registered in Step 1.

#### Migrating from Azure AD Graph API to Microsoft Graph API

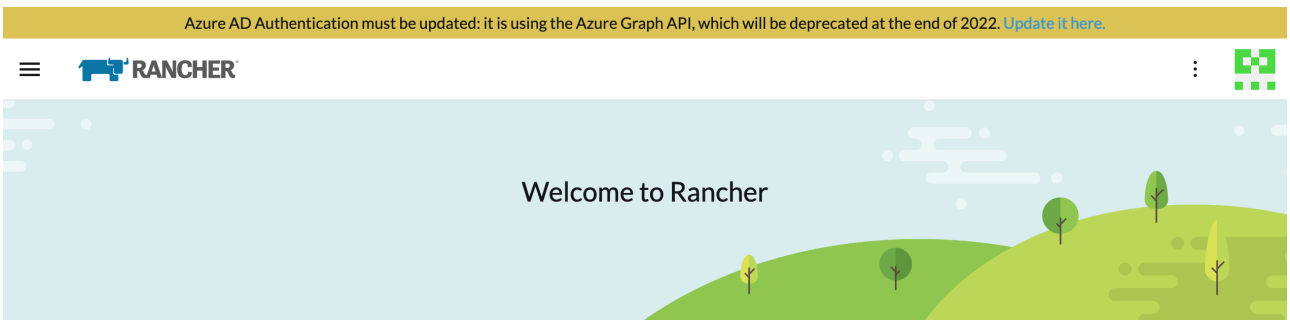
The legacy **Azure AD Graph API** has been retired by Microsoft. Rancher setups using the legacy endpoints must update to **Microsoft Graph API** to prevent authentication failures.

#### Updating Endpoints in the Rancher UI

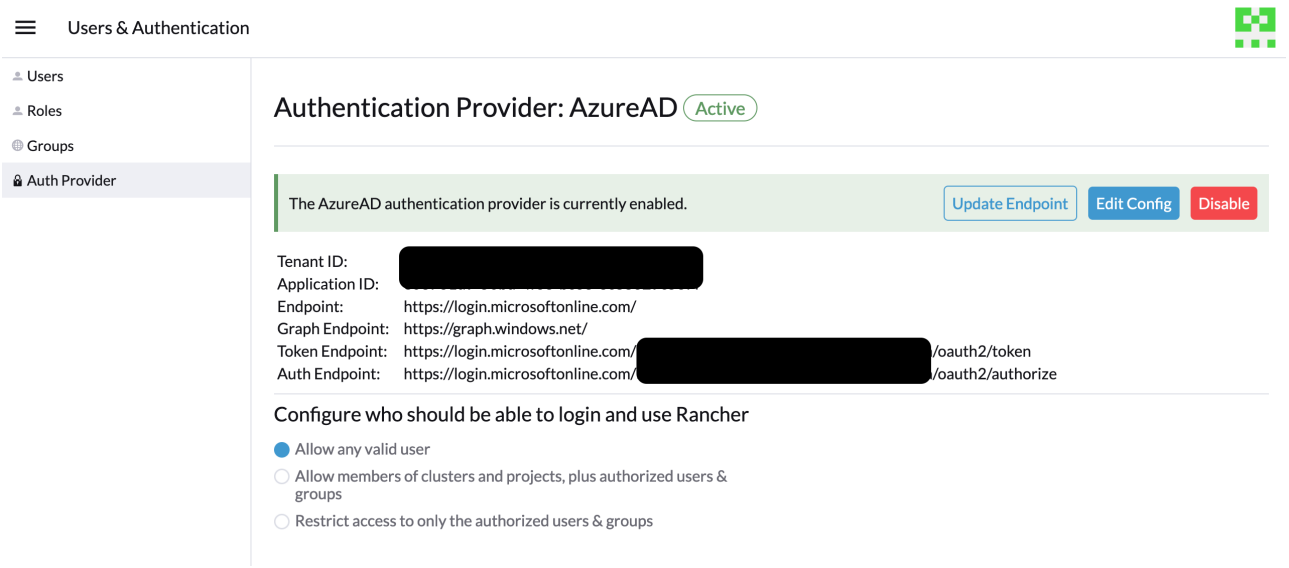


Create a full Rancher backup prior to initiating the endpoint migration.

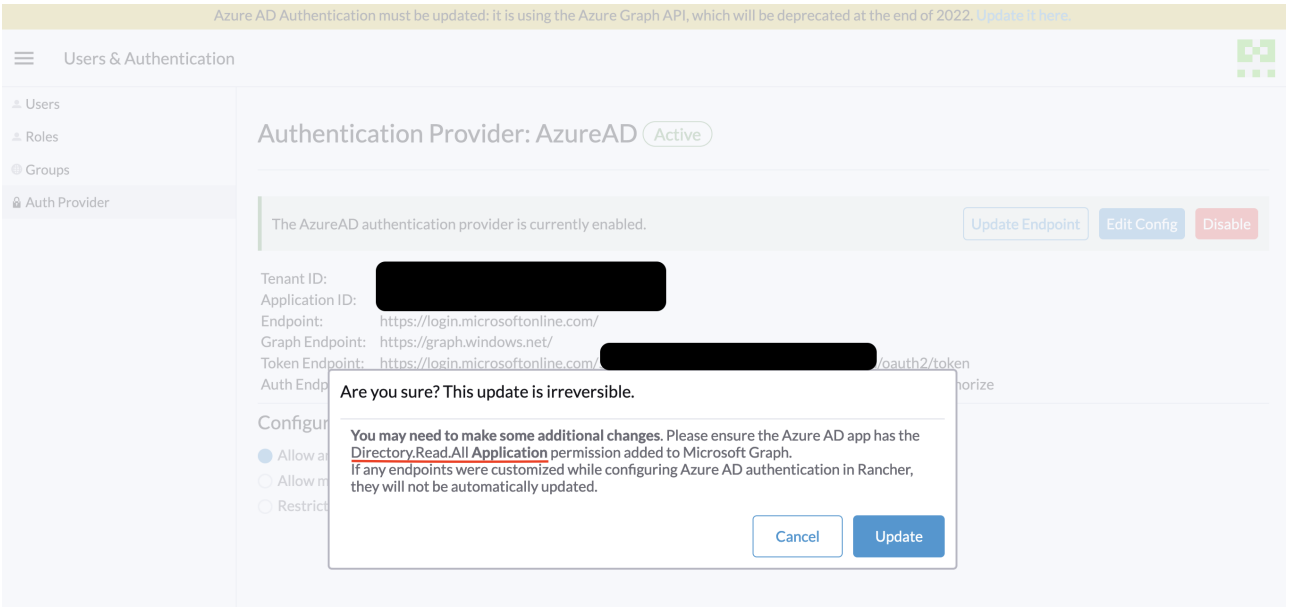
1. Ensure your app registration in Azure has been updated with the required Microsoft Graph permissions.
2. Log into Rancher as an administrator.
3. Click the notification banner prompting an update for Microsoft Entra ID authentication.



4. Click **Update Endpoint**.




5. Confirm the action in the pop-up dialog.




#### Air-Gapped Environments

Air-gapped environments must ensure network access and outbound firewall rules allow access to the new Microsoft Graph endpoints (`https://graph.microsoft.com`).

#### Rolling Back the Migration

If recovery or secret rotation is required post-migration:

1. Restore from backup using the standard Rancher restore procedure.
2. To rotate the Application Secret in Rancher, update the Kubernetes secret `azureadconfig-applicationsecret` in the `cattle-global-data` namespace.

#### Endpoint Reference (Global)

| Field | Legacy Endpoint (Deprecated) | New Endpoint (Microsoft Graph) |
|---|---|---|
| Auth Endpoint | `https://login.microsoftonline.com/{tenantID}/oauth2/authorize` | `https://login.microsoftonline.com/{tenantID}/oauth2/v2.0/authorize` |
| Endpoint | `https://login.microsoftonline.com/` | `https://login.microsoftonline.com/` |
| Graph Endpoint | `https://graph.windows.net/` | `https://graph.microsoft.com` |
| Token Endpoint | `https://login.microsoftonline.com/{tenantID}/oauth2/token` | `https://login.microsoftonline.com/{tenantID}/oauth2/v2.0/token` |

**Endpoint Reference (China)**

| Field | Legacy Endpoint (Deprecated) | New Endpoint (Microsoft Graph) |
|---|---|---|
| Auth Endpoint | `https://login.chinacloudapi.cn/{tenantID}/oauth2/authorize` | `https://login.partner.microsoftonline.cn/{tenantID}/oauth2/v2.0/authorize` |
| Endpoint | `https://login.chinacloudapi.cn/` | `https://login.partner.microsoftonline.cn/` |
| Graph Endpoint | `https://graph.chinacloudapi.cn/` | `https://microsoftgraph.chinacloudapi.cn` |
| Token Endpoint | `https://login.chinacloudapi.cn/{tenantID}/oauth2/token` | `https://login.partner.microsoftonline.cn/{tenantID}/oauth2/v2.0/token` |

**Filtering Users by Microsoft Entra ID Group Memberships**

In Rancher v2.9.0 and later, you can filter group memberships retrieved from Microsoft Entra ID using OData filter expressions.



Filtering out a group membership prevents Rancher from seeing that the user belongs to that group. As a result, users will lose any roles or access rights granted via excluded groups.

1. In Rancher, navigate to ☰ > **Users & Authentication** > **Auth Provider** > **Microsoft Entra ID**.
2. Check **Limit users by group membership**.
3. Enter an OData filter expression [https://learn.microsoft.com/en-us/odata/concepts/queryoptions-overview#filter] into the **Group Membership Filter** field (e.g., `startswith(displayName,'test')`).

**Deprecated Azure AD Graph API**



- The Azure AD Graph API is retired by Microsoft. All Rancher installations authenticating with Microsoft Entra ID must utilize the Microsoft Graph API flow.

**Microsoft Entra ID Roles Claims**

Rancher supports the `roles` claim provided in the Microsoft Entra ID OIDC token, enabling centralized Role-Based Access Control (RBAC) mapping.

By mapping `roles` claims, administrators can:

- Define specific application roles in Microsoft Entra ID.
- Bind Entra ID roles directly to `ProjectRole` or `ClusterRole` resources within Rancher.
- Delegate identity and role management completely to the external identity provider.

For example, consider the following role structure in Microsoft Entra ID:

| Microsoft Entra ID Role Name | Members |
|---|---|
| project-alpha-dev | User A, User C |

When User A logs into Rancher using Microsoft Entra ID, their OIDC token includes a `roles` claim containing `["project-alpha-dev"]`. Rancher processes the token and adds `project-alpha-dev` to User A's internal list of user groups and roles. Because an administrator created a Project Role Binding that maps the Entra ID role `project-alpha-dev` to the Rancher `Dev Member` role for Project Alpha, User A is automatically granted `Dev Member` permissions.

#### Configure FreeIPA

If your organization uses FreeIPA for user authentication, you can configure Rancher to allow your users to login using their FreeIPA credentials.



*Prerequisites:*

- You must have a FreeIPA Server [https://www.freeipa.org/] configured.
- Create a service account in FreeIPA with `read-only` access. Rancher uses this account to verify group membership when a user makes a request using an API key.
- Read External Authentication Configuration and Principal Users.

1. Sign into Rancher using a local user assigned the `administrator` role (i.e., the *local principal*).
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **FreeIPA**.
5. Complete the **Configure an FreeIPA server** form.

   You may need to log in to your domain controller to find the information requested in the form.



*Using TLS?*

If the certificate is self-signed or not from a recognized certificate authority, make sure you provide the complete chain. That chain is needed to verify the server's certificate.



*User Search Base vs. Group Search Base*

Search base allows Rancher to search for users and groups that are in your FreeIPA. These fields are only for search bases and not for search filters.

- If your users and groups are in the same search base, complete only the User Search Base.
- If your groups are in a different search base, you can optionally complete the Group Search Base. This field is dedicated to searching groups, but is not required.

6. If your FreeIPA deviates from the standard AD schema, complete the **Customize Schema** form to match it. Otherwise, skip this step.



*Search Attribute*

The Search Attribute field defaults with three specific values: `uid|sn|givenName`. After FreeIPA is configured, when a user enters text to add users or groups, Rancher automatically queries the FreeIPA server and attempts to match fields by user id, last name, or first name. Rancher specifically searches for users/groups that begin with the text entered in the search field.

The default field value `uid|sn|givenName`, but you can configure this field to a subset of these fields. The pipe (`|`) between the fields separates these fields.

- `uid`: User ID
- `sn`: Last Name
- `givenName`: First Name

With this search attribute, Rancher creates search filters for users and groups, but you *cannot* add your own search filters in this field.

7. Enter your FreeIPA username and password in **Authenticate with FreeIPA** to confirm that Rancher is configured to use FreeIPA authentication.
8. Click **Enable**.

**Result:**

- FreeIPA authentication is configured.
- You are signed into Rancher with your FreeIPA account (i.e., the *external principal*).

#### Configure GitHub

In environments using GitHub, you can configure Rancher to allow sign on using GitHub credentials.



*Prerequisites:*

Read External Authentication Configuration and Principal Users.

1. Sign into Rancher using a local user assigned the `administrator` role (i.e., the *local principal*).
2. In the top left corner, click **☰ > Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **GitHub**.
5. Follow the directions displayed to set up a GitHub Application. Rancher redirects you to GitHub to complete registration.



*What's an Authorization Callback URL?*

The Authorization Callback URL is the URL where users go to begin using your application (i.e. the splash screen).

When you use external authentication, authentication does not actually take place in

your application. Instead, authentication takes place externally (in this case, GitHub). After this external authentication completes successfully, the Authorization Callback URL is the location where the user re-enters your application.

6. From GitHub, copy the **Client ID** and **Client Secret**. Paste them into Rancher.



*Where do I find the Client ID and Client Secret?*

From GitHub, select Settings > Developer Settings > OAuth Apps. The Client ID and Client Secret are displayed prominently.

7. Click **Authenticate with GitHub**.
8. Use the **Site Access** options to configure the scope of user authorization.
   - **Allow any valid Users**

     *Any* GitHub user can access Rancher. We generally discourage use of this setting!

   - **Allow members of Clusters, Projects, plus Authorized Users and Organizations**

     Any GitHub user or group added as a **Cluster Member** or **Project Member** can log in to Rancher. Additionally, any GitHub user or group you add to the **Authorized Users and Organizations** list may log in to Rancher.

   - **Restrict access to only Authorized Users and Organizations**

     Only GitHub users or groups added to the Authorized Users and Organizations can log in to Rancher.
9. Click **Enable**.

**Result:**

- GitHub authentication is configured.
- You are signed into Rancher with your GitHub account (i.e., the *external principal*).

#### Configure GitHub App

In environments using GitHub, you can configure the new GitHub App authentication provider in Rancher, which allows users to authenticate against a GitHub Organization account using a dedicated GitHub App [https://docs.github.com/en/apps/overview]. This new provider runs alongside the existing standard GitHub authentication provider, offering increased security and better management of permissions based on GitHub Organization teams.

#### Prerequisites



The GitHub App authentication provider only works with GitHub Organization accounts [https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts#organization-accounts]. It does not function with individual GitHub User accounts [https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts#user-accounts].

Before configuring the provider in Rancher, you must first create a GitHub App for your organization, generate a client secret for your GitHub App and generate a private key for your GitHub App. Refer to

Registering a GitHub App [https://docs.github.com/en/apps/creating-github-apps/registering-a-github-app/registering-a-github-app] for details.

#### Create GitHub App

1. Open your GitHub organization settings [https://github.com/settings/organizations].
2. To the right of the organization, select **Settings**.
3. In the left sidebar, click **Developer settings** > **GitHub Apps**.
4. Click **New GitHub App**.
5. Fill in the GitHub App configuration form with these values:
   - **GitHub App name**: Anything you like, e.g. `My Rancher`.
   - **Application description**: Optional, can be left blank.
   - **Homepage URL**: `https://localhost:8443`.
   - **Callback URL**: `https://localhost:8443/verify-auth`.
6. Select **Create GitHub App**.

#### Generate a Client Secret

Generate a client secret [https://docs.github.com/en/rest/authentication/authenticating-to-the-rest-api#using-basic-authentication] on the settings page for your app.

1. Go to your GitHub App.
2. Next to **Client Secrets**, select **Generate a new client secret**.

#### Generate a Private Key

Generate a private key [https://docs.github.com/en/enterprise-server/apps/creating-github-apps/authenticating-with-a-github-app/managing-private-keys-for-github-apps#generating-private-keys] on the settings page for your app.

1. Go to your GitHub App.
2. Next to **Private Keys**, click **Generate a private key**.

#### GitHub App Auth Provider Configuration

To set up the GitHub App Auth Provider in Rancher, follow these steps:

1. Navigate to the **Users & Authentication** section in the Rancher UI.
2. Select **Auth Providers**.
3. Select the **GitHub App** tile.
4. Gather and enter the details of your GitHub App into the configuration form fields.

| Field Name | Description |
|---|---|
| **Client ID** (Required) | The client ID of your GitHub App. |
| **Client Secret** (Required) | The client secret of your GitHub App. |
| **GitHub App ID** (Required) | The numeric ID associated with your GitHub App. |

| Field Name | Description |
|---|---|
| **Installation ID** (Optional) | If you want to restrict authentication to a single installation of the App, provide its specific numeric Installation ID. |
| **Private Key** (Required) | The contents of the Private Key file (in PEM format) generated by GitHub for your App. |



A GitHub App can be installed across multiple Organizations, and each installation has a unique Installation ID. If you want to restrict authentication to a single App installation and GitHub Organization, provide the Installation ID during configuration. If you do not provide an Installation ID, the user's permissions are aggregated across all installations.

5. Select **Enable**. Rancher attempts to validate the credentials and, upon success, activates the GitHub App provider.

After it is enabled, users logging in via the GitHub App provider are automatically identified and you can leverage your GitHub Organization's teams and users to configure Role-Based Access Control (RBAC) and to assign permissions to projects and clusters.



Ensure that the users and teams you intend to use for authorization exist within the GitHub organization managed by the App.

- **Users**: Individual GitHub users who are members of the GitHub Organization where the App is installed can log in.
- **Groups**: GitHub Organization teams are mapped to Rancher Groups, allowing you to assign entire teams permissions within Rancher projects and clusters.

#### Configure Google OAuth

If your organization uses G Suite for user authentication, you can configure Rancher to allow your users to log in using their G Suite credentials.

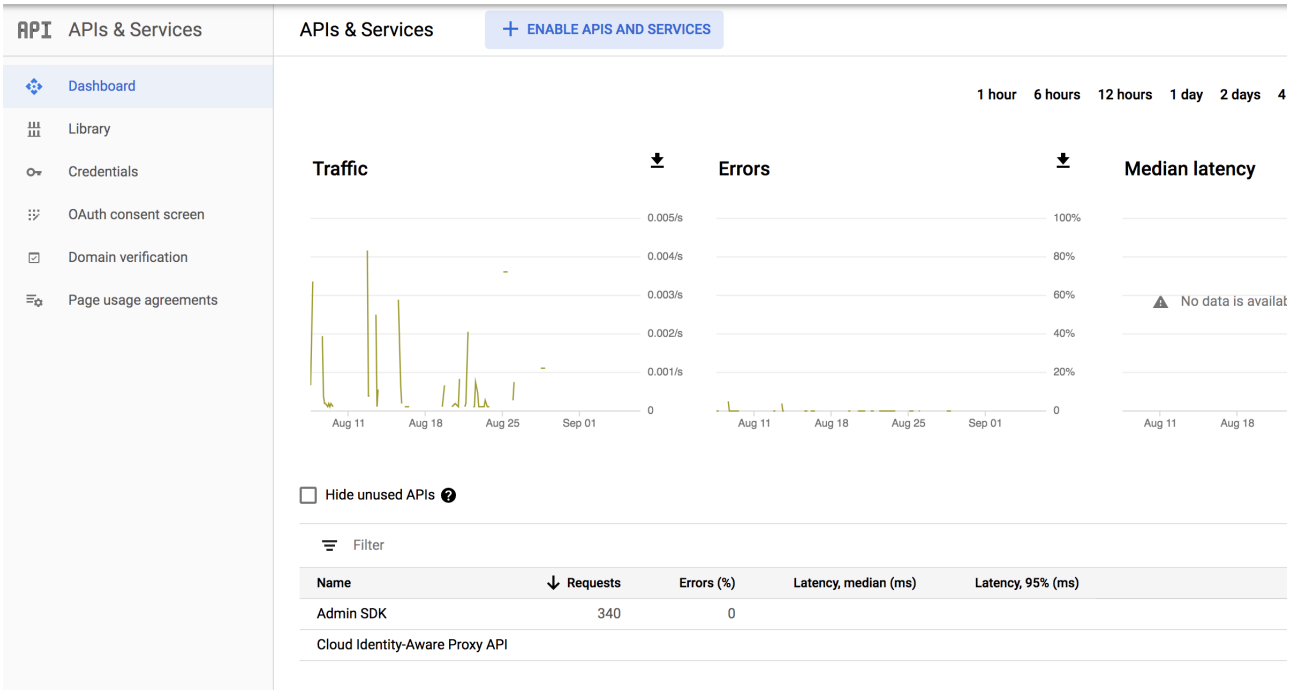
Only admins of the G Suite domain have access to the Admin SDK. Therefore, only G Suite admins can configure Google OAuth for Rancher.

Within Rancher, only administrators or users with the **Manage Authentication** global role can configure authentication.

#### Prerequisites

- You must have a G Suite admin account [https://admin.google.com] configured.
- G Suite requires a top private domain FQDN [https://github.com/google/guava/wiki/InternetDomainNameExplained#public-suffixes-and-private-domains] as an authorized domain. One way to get an FQDN is by creating an A-record in Route53 for your Rancher server. You do not need to update your Rancher Server URL setting with that record, because there could be clusters using that URL.
- You must have the Admin SDK API enabled for your G Suite domain. You can enable it using the steps on this page. [https://support.google.com/a/answer/60757?hl=en]

  After the Admin SDK API is enabled, your G Suite domain's API screen should look like this:



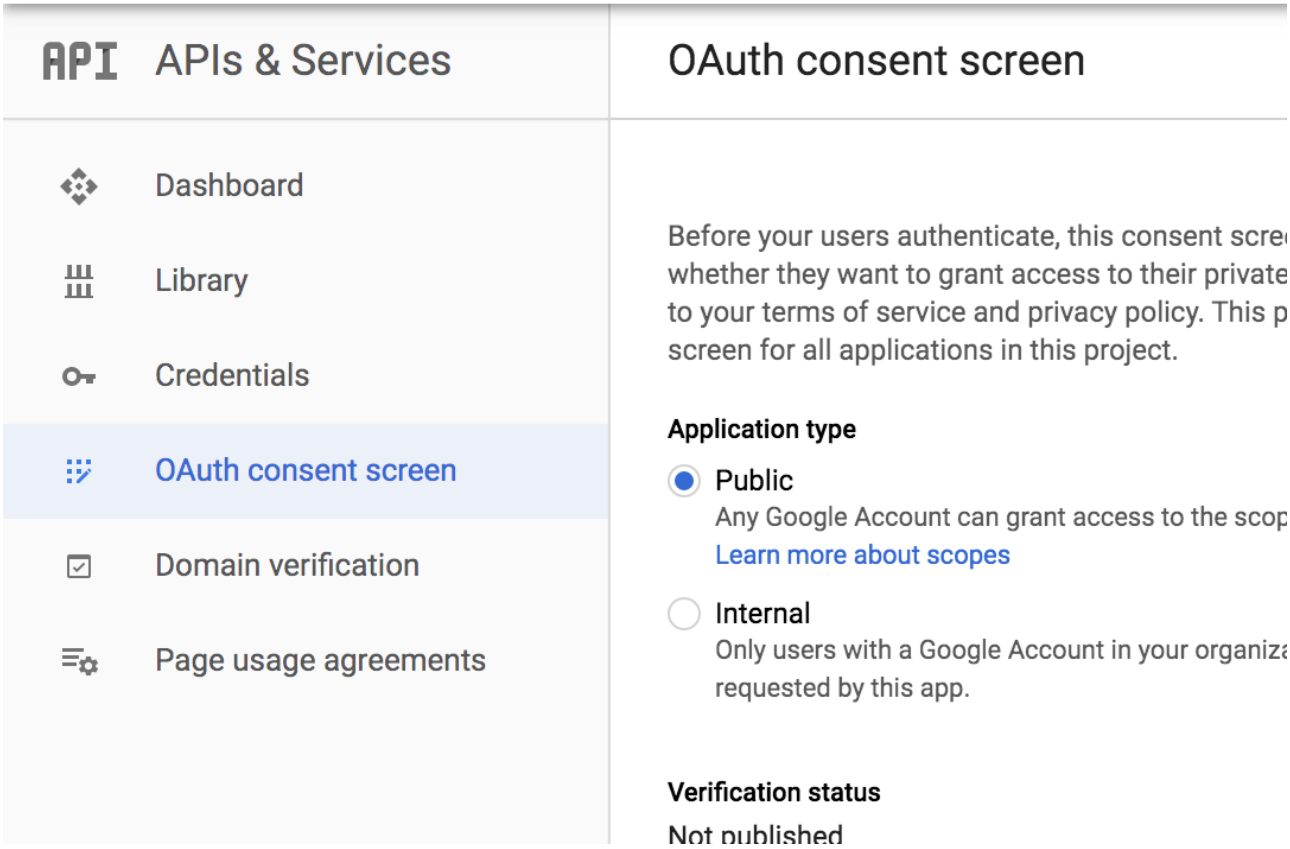

**Setting up G Suite for OAuth with Rancher**

Before you can set up Google OAuth in Rancher, you need to log in to your G Suite account and do the following:

1. Add Rancher as an authorized domain in G Suite
2. Generate OAuth2 credentials for the Rancher server
3. Create service account credentials for the Rancher server
4. Register the service account key as an OAuth Client

**1. Adding Rancher as an Authorized Domain**

1. Click here [https://console.developers.google.com/apis/credentials] to go to credentials page of your Google domain.
2. Select your project and click **OAuth consent screen**.

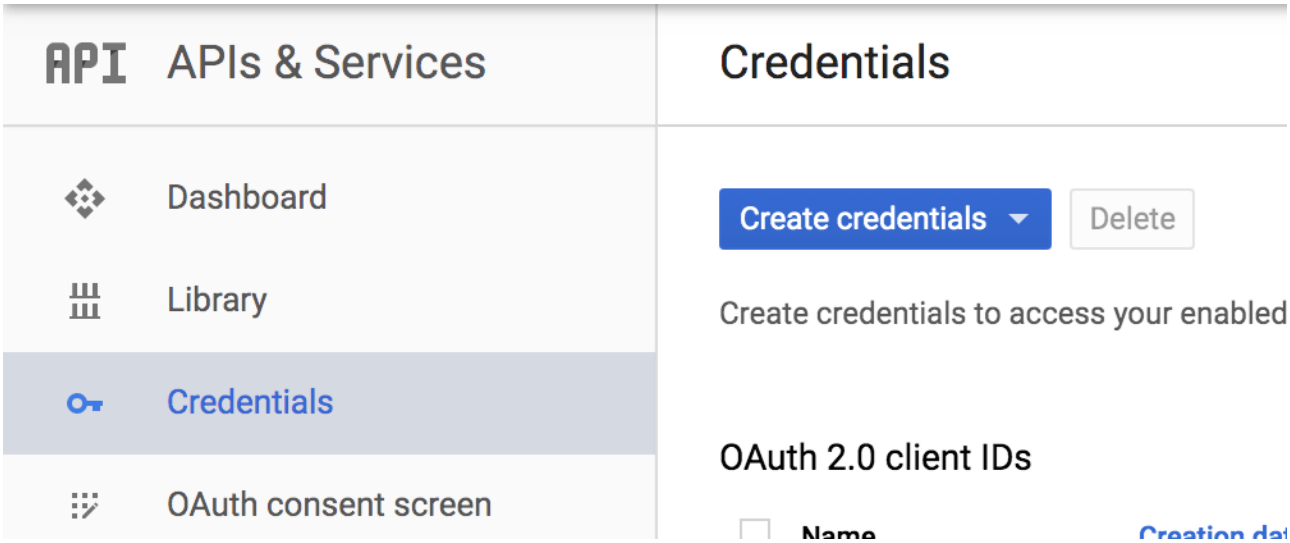



3. Go to **Authorized Domains** and enter the top private domain of your Rancher server URL in the list. The top private domain is the rightmost superdomain. So for example, www.foo.co.uk a top private domain of foo.co.uk. For more information on top-level domains, refer to this article. [https://github.com/google/guava/wiki/InternetDomainNameExplained#public-suffixes-and-private-domains]
4. Go to **Scopes for Google APIs** and make sure **email, profile** and **openid** are enabled.

**Result:** Rancher has been added as an authorized domain for the Admin SDK API.

#### 2. Creating OAuth2 Credentials for the Rancher Server

1. Go to the Google API console, select your project, and go to the credentials page. [https://console.developers.google.com/apis/credentials]




2. On the **Create Credentials** dropdown, select **OAuth client ID**.

3. Click **Web application**.
4. Provide a name.
5. Fill out the **Authorized JavaScript origins** and **Authorized redirect URIs**. Note: The Rancher UI page for setting up Google OAuth (available from the Global view under **Security › Authentication › Google**) provides you the exact links to enter for this step.
   - Under **Authorized JavaScript origins,** enter your Rancher server URL.
   - Under **Authorized redirect URIs,** enter your Rancher server URL appended with the path `verify-auth`. For example, if your URI is `https://rancherServer`, you will enter `https://rancherServer/verify-auth`.
6. Click on **Create**.
7. After the credential is created, you will see a screen with a list of your credentials. Choose the credential you just created, and in that row on rightmost side, click **Download JSON**. Save the file so that you can provide these credentials to Rancher.

**Result:** Your OAuth credentials have been successfully created.

#### 3. Creating Service Account Credentials

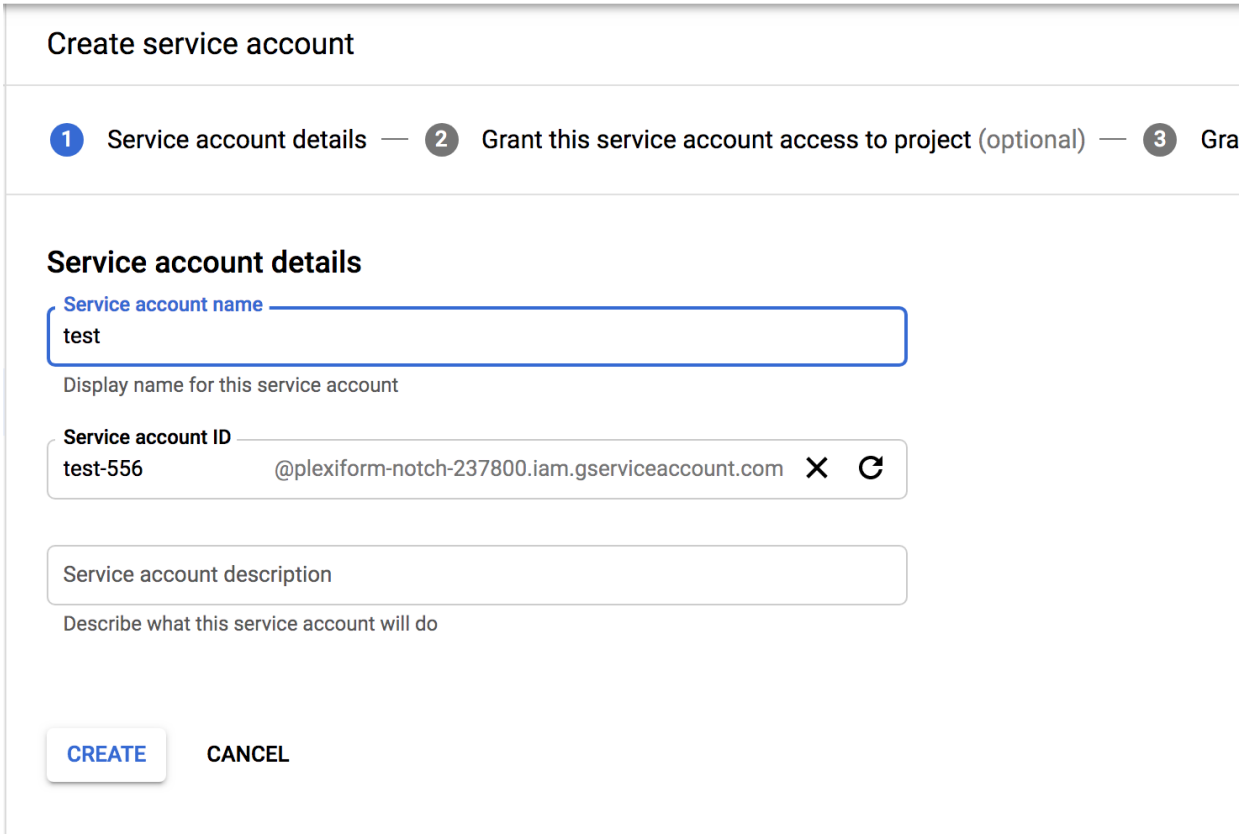
Since the Google Admin SDK is available only to admins, regular users cannot use it to retrieve profiles of other users or their groups. Regular users cannot even retrieve their own groups.

Since Rancher provides group-based membership access, we require the users to be able to get their own groups, and look up other users and groups when needed.

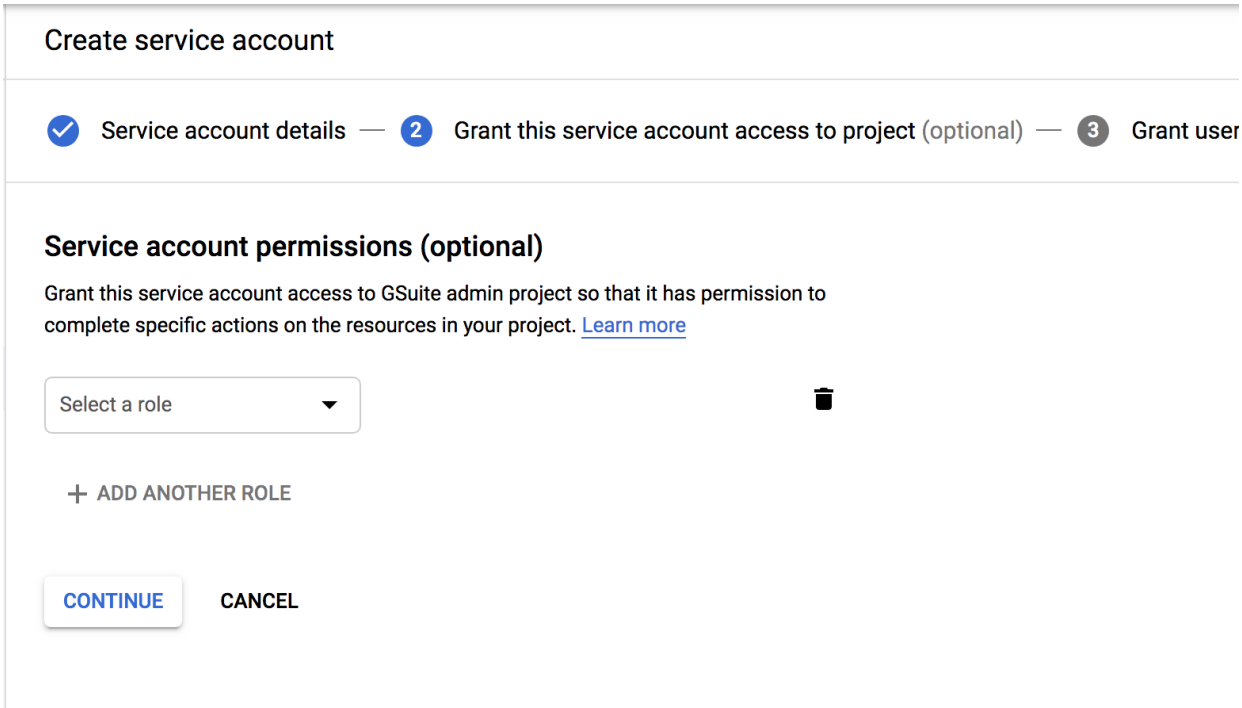
As a workaround to get this capability, G Suite recommends creating a service account and delegating authority of your G Suite domain to that service account.

This section describes how to:

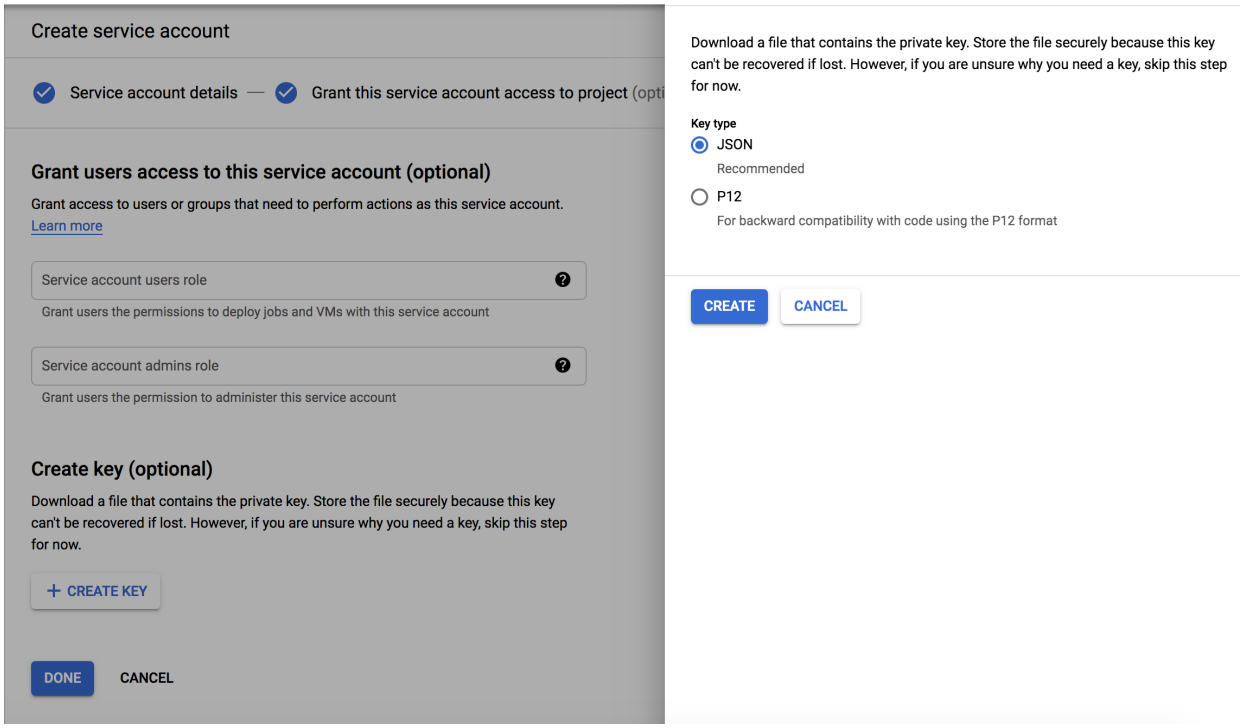
- Create a service account
- Create a key for the service account and download the credentials as JSON
  a. Click here [https://console.developers.google.com/iam-admin/serviceaccounts] and select your project for which you generated OAuth credentials.
  b. Click on **Create Service Account**.
  c. Enter a name and click **Create**.




d. Don't provide any roles on the **Service account permissions** page and click **Continue**




e. Click on **Create Key** and select the JSON option. Download the JSON file and save it so that you can provide it as the service account credentials to Rancher.



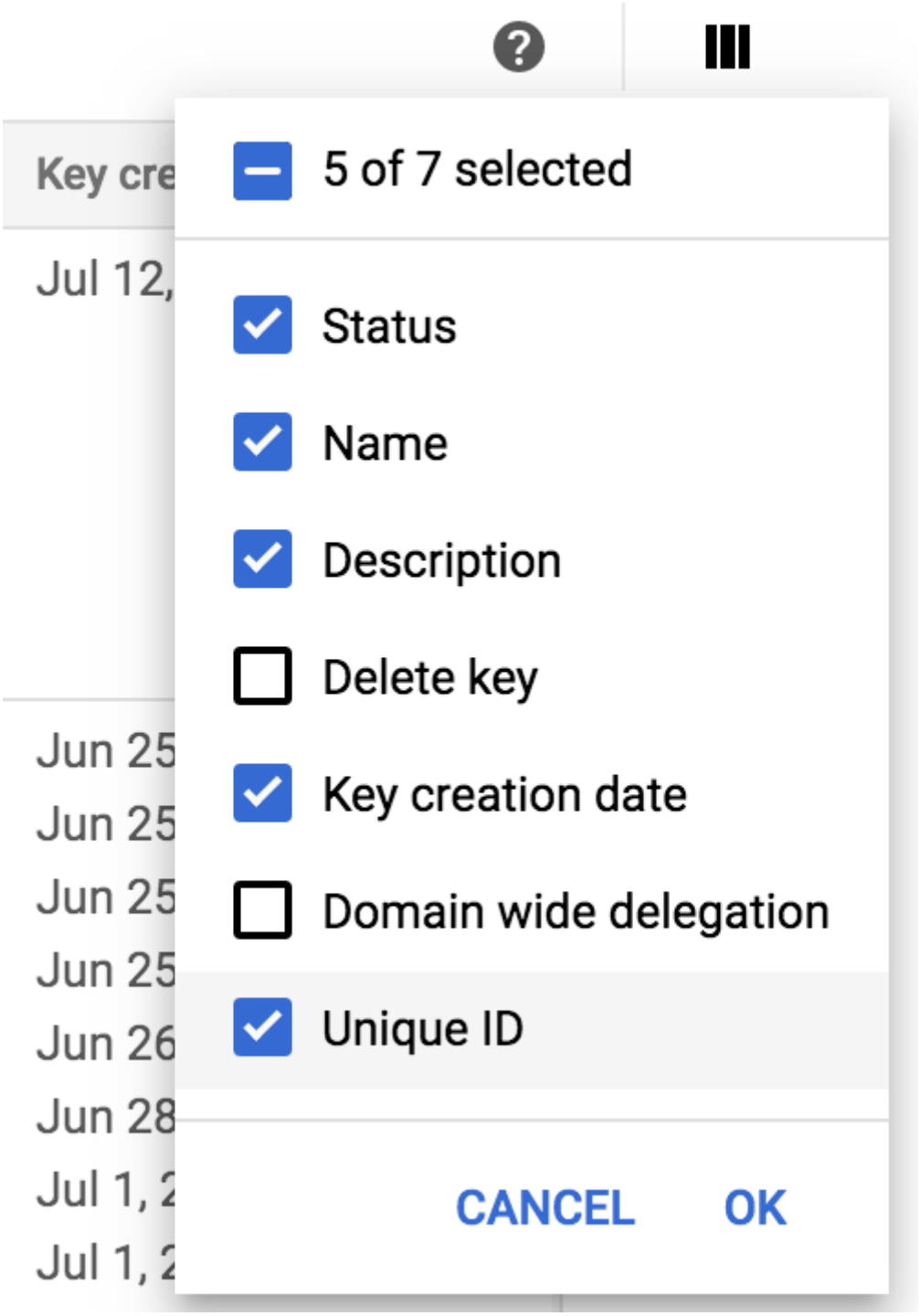

**Result:** Your service account is created.

#### 4. Register the Service Account Key as an OAuth Client

You will need to grant some permissions to the service account you created in the last step. Rancher requires you to grant only read-only permissions for users and groups.

Using the Unique ID of the service account key, register it as an Oauth Client using the following steps:

1. Get the Unique ID of the key you just created. If it's not displayed in the list of keys right next to the one you created, you will have to enable it. To enable it, click **Unique ID** and click **OK**. This will add a **Unique ID** column to the list of service account keys. Save the one listed for the service account you created. NOTE: This is a numeric key, not to be confused with the alphanumeric field **Key ID**.




2. Go to the **Domain-wide Delegation** page. [https://admin.google.com/ac/owl/domainwidedelegation]
3. Add the Unique ID obtained in the previous step in the **Client Name** field.
4. In the **One or More API Scopes** field, add the following scopes:

```
openid,profile,email,https://www.googleapis.com/auth/admin.directory.user.readonly,https://www.googleapis.com/auth/admin.directory.group.readonly
```

5. Click **Authorize**.

**Result:** The service account is registered as an OAuth client in your G Suite account.

#### Configuring Google OAuth in Rancher

1. Sign into Rancher using a local user assigned the administrator role. This user is also called the local principal.
2. In the top left corner, click ☰ > **Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **Google**. The instructions in the UI cover the steps to set up authentication with Google OAuth.
    a. Admin Email: Provide the email of an administrator account from your GSuite setup. In order to perform user and group lookups, google apis require an administrator's email in conjunction with the service account key.
    b. Domain: Provide the domain on which you have configured GSuite. Provide the exact domain and not any aliases.
    c. Nested Group Membership: Check this box to enable nested group memberships. Rancher admins can disable this at any time after configuring auth.
        - **Step One** is about adding Rancher as an authorized domain, which we already covered in this section.
        - For **Step Two,** provide the OAuth credentials JSON that you downloaded after completing this section. You can upload the file or paste the contents into the **OAuth Credentials** field.
        - For **Step Three,** provide the service account credentials JSON that downloaded at the end of this section. The credentials will only work if you successfully registered the service account key as an OAuth client in your G Suite account.
5. Click **Authenticate with Google**.
6. Click **Enable**.

**Result:** Google authentication is successfully configured.

#### Configure Keycloak (OIDC)

If your organization uses Keycloak Identity Provider (IdP) [https://www.keycloak.org] for user authentication, you can configure Rancher to allow your users to log in using their IdP credentials. Rancher supports integration with Keycloak using the OpenID Connect (OIDC) protocol and the SAML protocol. Both implementations are functionally equivalent when used with Rancher. This page describes the process to configure Rancher to work with Keycloak using the OIDC protocol.

If you prefer to use Keycloak with the SAML protocol instead, refer to this page.

If you have an existing configuration using the SAML protocol and want to switch to the OIDC protocol, refer to this section.

#### Prerequisites

- On Rancher, Keycloak (SAML) is disabled.
- You must have a Keycloak IdP Server [https://www.keycloak.org/guides#getting-started] configured.
- Follow the Keycloak documentation [https://www.keycloak.org/docs/latest/server_admin/#proc-creating-oidc-client_server_administration_guide] to create a new OIDC client with the settings below.

| Setting | Value |
|---|---|
| `Client ID` | <client-id> (e.g. `rancher`) |
| `Name` | <client-name> (e.g. `rancher`) |
| `Client type` | `OpenID Connect` |
| `Client authentication` | `ON` |
| `Valid Redirect URI` | `https://yourRancherHostURL/verify-auth` |

- In the new OIDC client, create Mappers [https://www.keycloak.org/docs/latest/server_admin/#_protocol-mappers] to expose the users fields.
  a. In the navigation menu, click **Clients**.
  b. Click the **Clients list** tab.
  c. Find and click the client you created.
  d. Click the **Client scopes** tab.
  e. Find and click the link labeled `<client-name>-dedicated`. For example, if you named your client `rancher`, look for the link named `rancher-dedicated`.
  f. Click the **Mappers** tab.
  g. Click **Configure a new mapper**. If you already have existing mappers configured, click the arrow next to **Add mapper** and select **By configuration**. Repeat this process and create these mappers:
     - From the mappings table, select **Group Membership** and configure a new "Groups Mapper" with the settings below. For settings that are not mentioned, use the default value.

| Setting | Value |
|---|---|
| `Name` | `Groups Mapper` |
| `Mapper Type` | `Group Membership` |
| `Token Claim Name` | `groups` |
| `Full group path` | `OFF` |
| `Add to ID token` | `OFF` |
| `Add to access token` | `OFF` |
| `Add to user info` | `ON` |

     - From the mappings table, select **Audience** and configure a new "Client Audience" with the settings below. For settings that are not mentioned, use the default value.

| Setting | Value |
|---|---|
| `Name` | `Client Audience` |
| `Mapper Type` | `Audience` |
| `Included Client Audience` | <client-name> |
| `Add to ID token` | `OFF` |
| `Add to access token` | `ON` |

     - From the mappings table, select **Group Membership** and configure a new "Groups Path" with the settings below. For settings that are not mentioned, use the default value.

| Setting | Value |
|---|---|
| `Name` | `Group Path` |
| `Mapper Type` | `Group Membership` |
| `Token Claim Name` | `full_group_path` |
| `Full group path` | `ON` |
| `Add to ID token` | `ON` |
| `Add to access token` | `ON` |
| `Add to user info` | `ON` |

- Add the following role mappings to all users or groups that need to query the Keycloak users.

**Users**

a. In the navigation menu, click **Users**.

b. Click the user you want to add role mappings to.

c. Click the **Role mapping** tab.

d. Click **Assign role**.

e. Select the following roles:

- query-users
- query-groups
- view-users

f. Click **Assign**.

**Groups**

a. In the navigation menu, click **Groups**.

b. Click the group you want to add role mappings to.

c. Click the **Role mapping** tab.

d. Click **Assign role**.

e. Select the following roles:

- query-users
- query-groups
- view-users

f. Click **Assign**.

#### Configuring Keycloak in Rancher

1. In the Rancher UI, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Auth Provider**.
3. Select **Keycloak (OIDC)**.
4. Complete the **Configure a Keycloak OIDC account** form. For help with filling the form, see the configuration reference.



When configuring the **Endpoints** section using the **Generate** option, Rancher includes `/auth` as part of the context path in the **Issuer** and **Auth Endpoint** fields, which is only valid for Keycloak 16 or older. You must configure endpoints using the **Specify** option for Keycloak 17 [https://www.keycloak.org/docs/latest/release_notes/index.html#keycloak-17-0-0] and newer, which have migrated to Quarkus [https://www.keycloak.org/migration/migrating-to-quarkus].

5. After you complete the **Configure a Keycloak OIDC account** form, click **Enable**.

   Rancher redirects you to the IdP login page. Enter credentials that authenticate with Keycloak IdP to validate your Rancher Keycloak configuration.



You may need to disable your popup blocker to see the IdP login page.

**Result:** Rancher is configured to work with Keycloak using the OIDC protocol. Your users can now sign in to Rancher using their Keycloak logins.

#### Configuration Reference

| Field | Description |
|---|---|
| Client ID | The `Client ID` of your Keycloak client. |
| Client Secret | The generated `Secret` of your Keycloak client. In the Keycloak console, select **Clients**, select the client you created, select the **Credentials** tab and copy the value of the `Secret` field. |
| Private Key / Certificate | A key/certificate pair to create a secure shell between Rancher and your IdP. Required if HTTPS/SSL is enabled on your Keycloak server. |
| Endpoints | Choose whether to use the generated values for the `Rancher URL`, `Issue`, and `Auth Endpoint` fields or to provide manual overrides if incorrect. |
| Keycloak URL | The URL for your Keycloak server. |
| Keycloak Realm | The name of the realm in which the Keycloak client was created in. |
| Rancher URL | The URL for your Rancher Server. |
| Issuer | The URL of your IdP. |
| Auth Endpoint | The URL where users are redirected to authenticate. |

#### Migrating from SAML to OIDC

This section describes the process to transition from using Rancher with Keycloak (SAML) to Keycloak (OIDC).

1. Reconfigure Keycloak.
   a. Configure a new `OpenID Connect` client according to the Prerequisites. Ensure the same `Valid Redirect URIs` are set.
   b. Configure mappers for the new client according to the Prerequisites.
2. Before configuring Rancher to use Keycloak (OIDC), Keycloak (SAML) must be first disabled.

a. In the Rancher UI, click ☰ > **Users & Authentication**.

b. In the left navigation bar, click **Auth Provider**.

c. Select **Keycloak (SAML)**.

d. Click **Disable**.

3. Follow the steps in Configuring Keycloak in Rancher.



After configuration is completed, Rancher user permissions need to be reapplied as they are not automatically migrated.

#### OIDC Support for PKCE Extension

Rancher supports the Proof Key for Code Exchange (PKCE) extension (RFC 7636) for OIDC authentication providers. SHA-256 (S256) is the only supported PKCE verification method. To enable this feature, your authentication provider must use PKCE with S256 for authorization requests.

You can enable this feature by selecting **Enable PKCE (S256)** in your authentication provider configuration in Rancher. Enabling S256 PKCE token verification allows you to mitigate authorization code interception attacks during OIDC authentication flows.

#### Configuring OIDC Single Logout (SLO)

Rancher supports the ability to configure OIDC Single Logout (SLO). Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options.

#### Prerequisites

Before configuring OIDC SLO, ensure the following is set up on your IdP:

- **SLO Support**: The **Log Out behavior** configuration section only appears if your OIDC IdP allows for `OIDC SLO`.
- **Post-Logout Redirect URI**: Your Rancher Server URL must be configured as an authorized post-logout redirect URI in your IdP's OIDC client settings. This URL is used by the IdP to redirect a user back to Rancher after a successful external logout.

#### OIDC SLO Configuration

Configure the SLO settings when setting up or editing your OIDC authentication provider.

1. Sign in to Rancher using a standard user or an administrator role.
2. In the top left corner, select ☰ > **Users & Authentication**.
3. In the left navigation menu, select **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |

| Setting | Description |
| --- | --- |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

5. If you choose to log out of your IdP, provide an **End Session Endpoint**. Rancher uses this URL to initiate the external logout.

#### How to get the End Session Endpoint

The `end_session_endpoint` is one of the specific URLs published within a standardized JSON object containing the IdP's metadata and is retrieved from the OIDC Discovery URL. To get the `end_session_endpoint` from the OIDC Discovery URL, follow these steps:

1. Obtain the Discovery URL by appending the IdP Issuer URL with the well-known path (`.well-known/openid-configuration`).
2. Send an HTTP `GET` request to the Discovery URL.
3. In the JSON object, look for the key named `end_session_endpoint` and retrieve the URL.

   You can also use a `curl` command to retrieve `end_session_endpoint`:

   ```
   curl -s <ISSUER_URL>/.well-known/openid-configuration | jq
   '.end_session_endpoint'
   ```

#### Annex: Troubleshooting

If you are experiencing issues while testing the connection to the Keycloak server, first double-check the configuration options of your OIDC client. You may also inspect the Rancher logs to help pinpoint what's causing issues. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

All Keycloak related log entries are prepended with either `[generic oidc]` or `[keycloak oidc]`.

#### You are not redirected to Keycloak

When you fill the **Configure a Keycloak OIDC account** form and click **Enable**, you are not redirected to your IdP.

Verify your Keycloak client configuration.

#### The generated `Issuer` and `Auth Endpoint` are incorrect

On the **Configure a Keycloak OIDC account** form, change **Endpoints** to `Specify (advanced)` and override the `Issuer` and `Auth Endpoint` values. To find the values, go to the Keycloak console and select **Realm Settings**, select the **General** tab, and click **OpenID Endpoint Configuration**. The JSON output displays values for `issuer` and `authorization_endpoint`.

#### Keycloak Error: "Invalid grant_type"

In some cases, this error message may be misleading and is caused by setting the `Valid Redirect URI` incorrectly.

#### Unable to See Groups When Assigning Global Roles

If you use a user that is not part of any groups for initial setup, then you cannot search for groups when trying to assign a global role. To resolve this, you can either:

1. Manually edit the `authconfig/keycloakoidc` object to enable group search.
   a. On the Rancher server:

   ```
   kubectl edit authconfigs.management.cattle.io keycloakoidc
   ```

   b. Set `groupSearchEnabled: true`.
   c. Save your changes.
2. Reconfigure your Keycloak OIDC setup using a user that is assigned to at least one group in Keycloak.

#### Configure Keycloak (SAML)

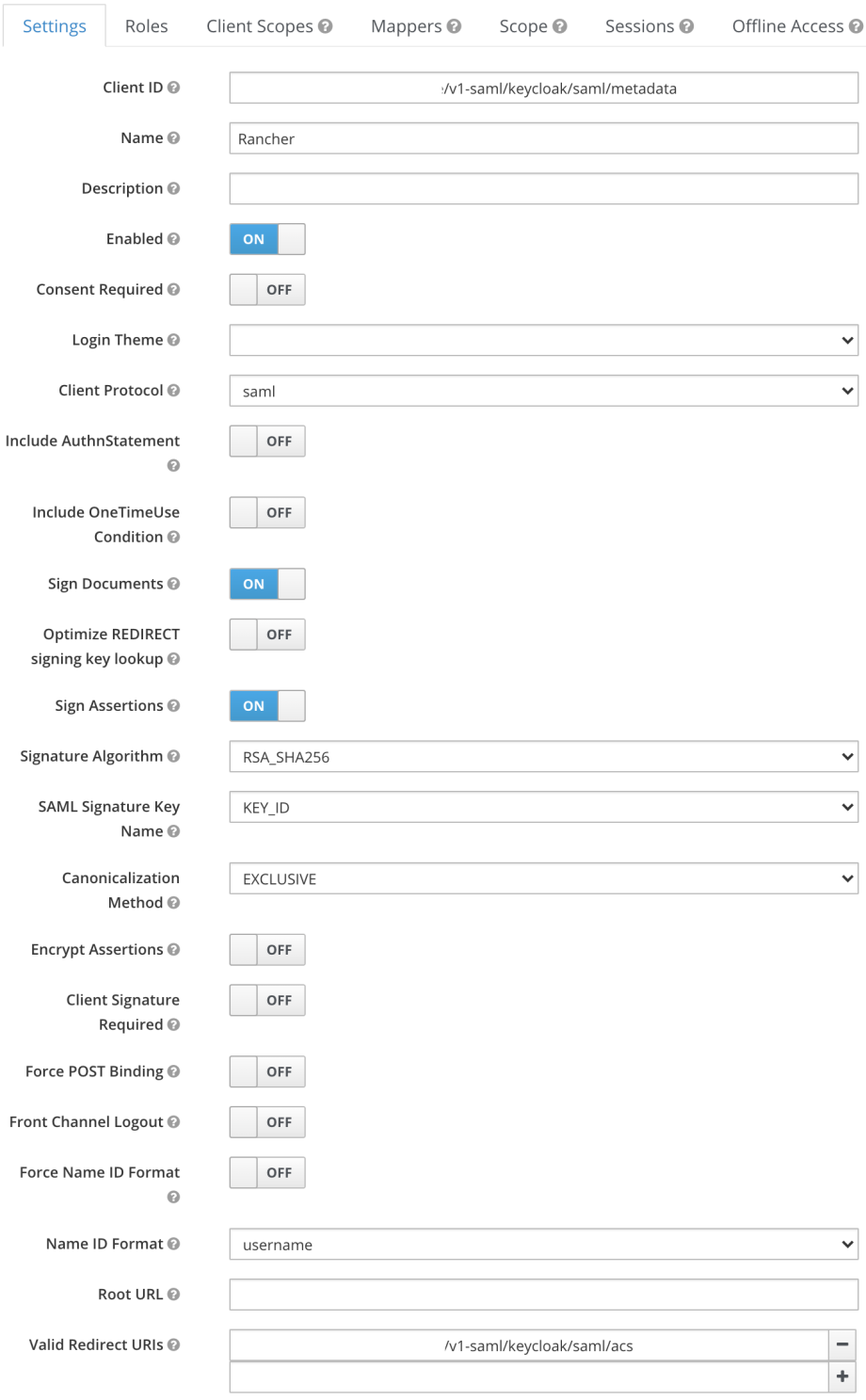
If your organization uses Keycloak Identity Provider (IdP) for user authentication, you can configure Rancher to allow your users to log in using their IdP credentials.

#### Prerequisites

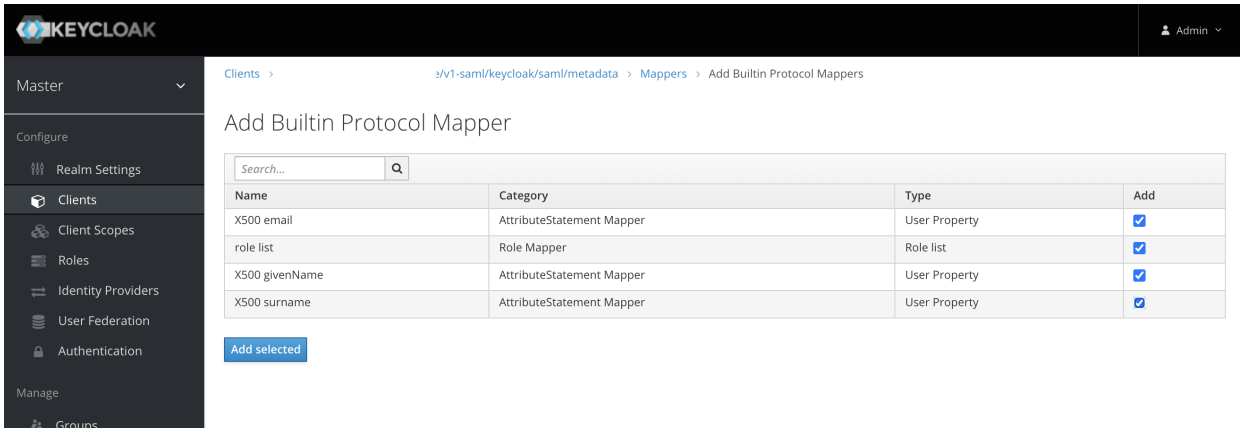
- You must have a Keycloak IdP Server [https://www.keycloak.org/guides#getting-started] configured.
- In Keycloak, create a new SAML client [https://www.keycloak.org/docs/latest/server_admin/#saml-clients], with the settings below. See the Keycloak documentation [https://www.keycloak.org/docs/latest/server_admin/#saml-clients] for help.

| Setting | Value |
|---|---|
| `Sign Documents` | `ON` [1] |
| `Sign Assertions` | `ON` [1] |
| All other `ON/OFF` Settings | `OFF` |
| `Client ID` | Either `https://yourRancherHostURL/v1-saml/keycloak/saml/metadata` or the value configured in the `Entry ID Field` of the Rancher Keycloak configuration[2] |
| `Client Name` | <CLIENT_NAME>(e.g. <code>rancher</code>)</CLIENT_NAME> |
| `Client Protocol` | `SAML` |
| `Valid Redirect URI` | `https://yourRancherHostURL/v1-saml/keycloak/saml/acs` |

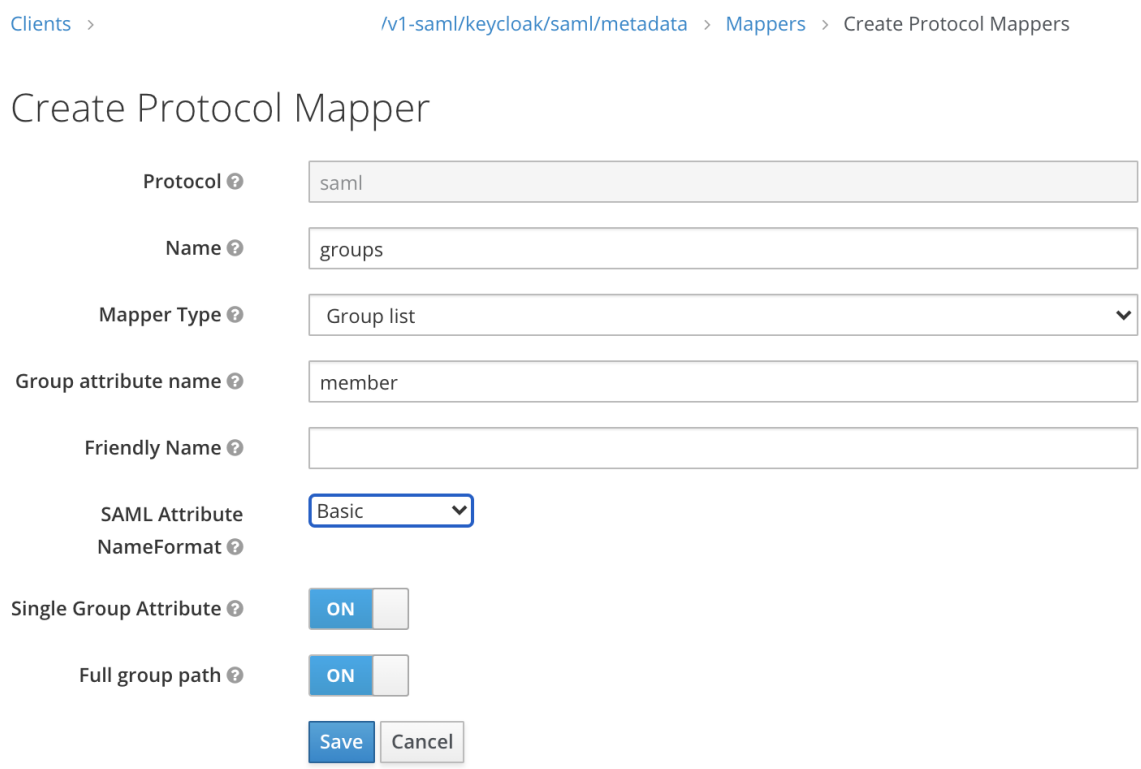
> [1]: Optionally, you can enable either one or both of these settings. [2]: Rancher SAML metadata won't be generated until a SAML provider is configured and saved.




- In the new SAML client, create Mappers to expose the users fields
  - Add all "Builtin Protocol Mappers"




- Create a new "Group list" mapper to map the member attribute to a user's groups




**Getting the IDP Metadata**

**Keycloak 5 and earlier**

To get the IDP metadata, export a `metadata.xml` file from your Keycloak client. From the **Installation** tab, choose the **SAML Metadata IDPSSODescriptor** format option and download your file.

**Keycloak 6-13**

1. From the **Configure** section, click the **Realm Settings** tab.
2. Click the **General** tab.
3. From the **Endpoints** field, click **SAML 2.0 Identity Provider Metadata**.

Verify the IDP metadata contains the following attributes:

```
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
```

Some browsers, such as Firefox, may render/process the document such that the contents appear to have been modified, and some attributes appear to be missing. In this situation, use the raw response data that can be found using your browser.

The following is an example process for Firefox, but will vary slightly for other browsers:

1. Press **F12** to access the developer console.
2. Click the **Network** tab.
3. From the table, click the row containing `descriptor`.
4. From the details pane, click the **Response** tab.
5. Copy the raw response data.

The XML obtained contains `EntitiesDescriptor` as the root element. Rancher expects the root element to be `EntityDescriptor` rather than `EntitiesDescriptor`. So before passing this XML to Rancher, follow these steps to adjust it:

1. Copy all the attributes from `EntitiesDescriptor` to the `EntityDescriptor` that are not present.
2. Remove the `<EntitiesDescriptor>` tag from the beginning.
3. Remove the `</EntitiesDescriptor>` from the end of the xml.

You are left with something similar as the example below:

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" entityID
="https://{KEYCLOAK-URL}/auth/realms/{REALM-NAME}">
....
</EntityDescriptor>
```

**Keycloak 14+**

1. From the **Configure** section, click the **Realm Settings** tab.
2. Click the **General** tab.
3. From the **Endpoints** field, click **SAML 2.0 Identity Provider Metadata**.

Verify the IDP metadata contains the following attributes:

```
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
```

Some browsers, such as Firefox, may render/process the document such that the contents appear to have been modified, and some attributes appear to be missing. In this situation, use the raw response data that can be found using your browser.

The following is an example process for Firefox, but will vary slightly for other browsers:

1. Press **F12** to access the developer console.
2. Click the **Network** tab.
3. From the table, click the row containing `descriptor`.
4. From the details pane, click the **Response** tab.
5. Copy the raw response data.

#### Configuring Keycloak in Rancher

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **Keycloak SAML**.
4. Complete the **Configure Keycloak Account** form. For help with filling the form, see the configuration reference.
5. After you complete the **Configure a Keycloak Account** form, click **Enable**.

   Rancher redirects you to the IdP login page. Enter credentials that authenticate with Keycloak IdP to validate your Rancher Keycloak configuration.

   You may have to disable your popup blocker to see the IdP login page.

**Result:** Rancher is configured to work with Keycloak. Your users can now sign into Rancher using their Keycloak logins.



*SAML Provider Caveats*

- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

#### Configuration Reference

| Field | Description |
|---|---|
| Display Name Field | The attribute that contains the display name of users.<br><br>Example: `givenName` |

| Field | Description |
| --- | --- |
| User Name Field | The attribute that contains the user name/given name.<br><br>Example: `email` |
| UID Field | An attribute that is unique to every user.<br><br>Example: `email` |
| Groups Field | Make entries for managing group memberships.<br><br>Example: `member` |
| Entity ID Field | The ID that needs to be configured as a client ID in the Keycloak client.<br><br>Default: `https://yourRancherHostURL/v1-saml/keycloak/saml/metadata` |
| Rancher API Host | The URL for your Rancher Server. |
| Private Key / Certificate | A key/certificate pair to create a secure shell between Rancher and your IdP. |
| IDP-metadata | The `metadata.xml` file that you exported from your IdP server. |



You can generate a key/certificate pair using an openssl command. For example:

openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout myservice.key -out myservice.cert

#### Annex: Troubleshooting

If you are experiencing issues while testing the connection to the Keycloak server, first double-check the configuration option of your SAML client. You may also inspect the Rancher logs to help pinpointing the problem cause. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

##### You are not redirected to Keycloak

When you click on **Authenticate with Keycloak**, you are not redirected to your IdP.

- Verify your Keycloak client configuration.
- Make sure `Force Post Binding` is set to `OFF`.

##### Forbidden message displayed after IdP login

You are correctly redirected to your IdP login page and you are able to enter your credentials, however you get a `Forbidden` message afterwards.

- Check the Rancher debug log.
- If the log displays `ERROR: either the Response or Assertion must be signed`, make sure either `Sign Documents` or `Sign assertions` is set to `ON` in your Keycloak client.

#### HTTP 502 when trying to access /v1-saml/keycloak/saml/metadata

This is usually due to the metadata not being created until a SAML provider is configured. Try configuring and saving keycloak as your SAML provider and then accessing the metadata.

#### Keycloak Error: "We're sorry, failed to process response"

- Check your Keycloak log.
- If the log displays `failed: org.keycloak.common.VerificationException: Client does not have a public key`, set `Encrypt Assertions` to `OFF` in your Keycloak client.

#### Keycloak Error: "We're sorry, invalid requester"

- Check your Keycloak log.
- If the log displays `request validation failed: org.keycloak.common.VerificationException: SigAlg was null`, set `Client Signature Required` to `OFF` in your Keycloak client.

#### CNI Community Popularity

#### Configuring SAML Single Logout (SLO)

Rancher supports the ability to configure SAML SLO. Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options. The steps below outline configuration from the application GUI:



The **Log Out behavior** configuration section only appears if the SAML authentication provider allows for `SAML SLO`.

1. Sign in to Rancher using a standard user or an administrator role to configure SAML SLO.
2. In the top left corner, click ☰ > **Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

#### SAML and OpenLDAP Group Permissions

When you configure a SAML authentication provider backed by OpenLDAP, the SAML response might return only a subset of the groups that a user belongs to. The exact groups returned depend on the configuration of your external authentication provider.

Rancher assigns user permissions based strictly on the groups provided in the SAML response.



Even if you can search for and view specific OpenLDAP groups in the Rancher UI, you cannot use them to assign permissions if they are missing from the SAML response.

To assign permissions successfully, verify that your SAML authentication provider is configured to return all necessary OpenLDAP groups.

#### Configure Okta (SAML)

If your organization uses Okta Identity Provider (IdP) for user authentication, you can configure Rancher to allow your users to log in using their IdP credentials.



Okta integration only supports Service Provider initiated logins.

#### Prerequisites

In Okta, create a SAML Application with the settings below. See the Okta documentation [https://developer.okta.com/standards/SAML/setting_up_a_saml_application_in_okta] for help.

| Setting | Value |
|---|---|
| `Single Sign on URL` | `https://yourRancherHostURL/v1-saml/okta/saml/acs` |
| `Audience URI (SP Entity ID)` | `https://yourRancherHostURL/v1-saml/okta/saml/metadata` |

#### Configuring Okta in Rancher

You can integrate Okta with Rancher, so that authenticated users can access Rancher resources through their group permissions. Okta returns a SAML assertion that authenticates a user, including which groups a user belongs to.

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **Okta**.
4. Complete the **Configure Okta Account** form. The examples below describe how you can map Okta attributes from attribute statements to fields within Rancher.

| Field | Description |
|---|---|
| Display Name Field | The attribute name from an attribute statement that contains the display name of users. |
| User Name Field | The attribute name from an attribute statement that contains the user name/given name. |
| UID Field | The attribute name from an attribute statement that is unique to every user. |
| Groups Field | The attribute name in a group attribute statement that exposes your groups. |
| Rancher API Host | The URL for your Rancher Server. |
| Private Key / Certificate | A key/certificate pair used for Assertion Encryption. |
| Metadata XML | The `Identity Provider metadata` file that you find in the application `Sign On` section. |



You can generate a key/certificate pair using an openssl command. For example:

```
 openssl req -x509 -sha256 -nodes -days 365 -newkey
rsa:2048 -keyout myservice.key -out myservice.crt
```

5. After you complete the **Configure Okta Account** form, click **Enable**.

   Rancher redirects you to the IdP login page. Enter credentials that authenticate with Okta IdP to validate your Rancher Okta configuration.



If nothing seems to happen, it's likely because your browser blocked the pop-up. Make sure you disable the pop-up blocker for your rancher domain and whitelist it in any other extensions you might utilize.

**Result:** Rancher is configured to work with Okta. Your users can now sign into Rancher using their Okta logins.



*SAML Provider Caveats*

If you configure Okta without OpenLDAP, you won't be able to search for or directly lookup users or groups. This brings several caveats:

- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

#### Okta with OpenLDAP search

You can add an OpenLDAP backend to assist with user and group search. Rancher will display additional users and groups from the OpenLDAP service. This allows assigning permissions to groups that the logged-in user is not already a member of.

#### OpenLDAP Prerequisites

If you use Okta as your IdP, you can set up an LDAP interface [https://help.okta.com/en-us/Content/Topics/Directory/LDAP-interface-main.htm] for Rancher to use. You can also configure an external OpenLDAP server.

You must configure Rancher with a LDAP bind account (aka service account) so that you can search and retrieve LDAP entries for users and groups that should have access. Don't use an administrator account or personal account as an LDAP bind account. Create [https://help.okta.com/en-us/Content/Topics/users-groups-profiles/usgp-add-users.htm] a dedicated account in OpenLDAP, with read-only access to users and groups under the configured searchbase.



*Security Considerations*

> The OpenLDAP service account is used for all searches. Rancher users will see users and groups that the OpenLDAP service account can view, regardless of their individual SAML permissions.

> **Using TLS?**
>
> If the certificate used by the OpenLDAP server is self-signed or from an unrecognized certificate authority, Rancher needs the CA certificate (concatenated with any intermediate certificates) in PEM format. Provide this certificate during the configuration so that Rancher can validate the certificate chain.

#### Configure OpenLDAP in Rancher

Configure the settings for the OpenLDAP server, groups and users. Note that nested group membership isn't available.

> Before you proceed with the configuration, please familiarise yourself with external authentication configuration and principal users.

1. Sign into Rancher using a local user assigned the administrator role (i.e., the *local principal*).
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **Okta** or, if SAML is already configured, **Edit Config**
5. Under **User and Group Search**, check **Configure an OpenLDAP server**

If you experience issues when you test the connection to the OpenLDAP server, ensure that you entered the credentials for the service account and configured the search base correctly. Inspecting the Rancher logs can help pinpoint the root cause. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging for more information.

#### Configuring SAML Single Logout (SLO)

Rancher supports the ability to configure SAML SLO. Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options. The steps below outline configuration from the application GUI:



> The **Log Out behavior** configuration section only appears if the SAML authentication provider allows for `SAML SLO`.

1. Sign in to Rancher using a standard user or an administrator role to configure SAML SLO.
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

#### SAML and OpenLDAP Group Permissions

When you configure a SAML authentication provider backed by OpenLDAP, the SAML response might return only a subset of the groups that a user belongs to. The exact groups returned depend on the configuration of your external authentication provider.

Rancher assigns user permissions based strictly on the groups provided in the SAML response.



Even if you can search for and view specific OpenLDAP groups in the Rancher UI, you cannot use them to assign permissions if they are missing from the SAML response.

To assign permissions successfully, verify that your SAML authentication provider is configured to return all necessary OpenLDAP groups.

#### Configure PingIdentity (SAML)

If your organization uses Ping Identity Provider (IdP) for user authentication, you can configure Rancher to allow your users to log in using their IdP credentials.

> **Prerequisites:**
>
> - You must have a Ping IdP Server [https://www.pingidentity.com/] configured.
> - Following are the Rancher Service Provider URLs needed for configuration: Metadata URL: `https://<rancher-server>/v1-saml/ping/saml/metadata` Assertion Consumer Service (ACS) URL: `https://<rancher-server>/v1-saml/ping/saml/acs` Note that these URLs will not return valid data until the authentication configuration is saved in Rancher.
> - Export a `metadata.xml` file from your IdP Server. For more information, see the PingIdentity documentation [https://documentation.pingidentity.com/pingfederate/pf83/index.shtml#concept_exportingMetadata.html].

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **Ping Identity**.
4. Complete the **Configure a Ping Account** form. Ping IdP lets you specify what data store you want to use. You can either add a database or use an existing ldap server. For example, if you select your Active Directory (AD) server, the examples below describe how you can map AD attributes to fields within

Rancher.

a. **Display Name Field**: Enter the AD attribute that contains the display name of users (example: `displayName`).

b. **User Name Field**: Enter the AD attribute that contains the user name/given name (example: `givenName`).

c. **UID Field**: Enter an AD attribute that is unique to every user (example: `sAMAccountName`, `distinguishedName`).

d. **Groups Field**: Make entries for managing group memberships (example: `memberOf`).

e. **Entity ID Field** (optional): The published, protocol-dependent, unique identifier of your partner. This ID defines your organization as the entity operating the server for SAML 2.0 transactions. This ID may have been obtained out-of-band or via a SAML metadata file.

f. **Rancher API Host**: Enter the URL for your Rancher Server.

g. **Private Key** and **Certificate**: This is a key-certificate pair to create a secure shell between Rancher and your IdP.

   You can generate one using an openssl command. For example:

   ```
   openssl req -x509 -newkey rsa:2048 -keyout myservice.key -out myservice.cert -days 365 -nodes -subj "/CN=myservice.example.com"
   ```

h. **IDP-metadata**: The `metadata.xml` file that you exported from your IdP server [https://documentation.pingidentity.com/pingfederate/pf83/index.shtml#concept_exportingMetadata.html].

5. After you complete the **Configure Ping Account** form, click **Enable**.

   Rancher redirects you to the IdP login page. Enter credentials that authenticate with Ping IdP to validate your Rancher PingIdentity configuration.

   You may have to disable your popup blocker to see the IdP login page.

**Result:** Rancher is configured to work with PingIdentity. Your users can now sign into Rancher using their PingIdentity logins.



*SAML Provider Caveats*

- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. If you have Administrator permissions, you can join a group that you are not a member of.

#### Configuring SAML Single Logout (SLO)

Rancher supports the ability to configure SAML SLO. Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a

prompt asking the user to choose between the previous options. The steps below outline configuration from the application GUI:



The **Log Out behavior** configuration section only appears if the SAML authentication provider allows for `SAML SLO`.

1. Sign in to Rancher using a standard user or an administrator role to configure SAML SLO.
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

#### SAML and OpenLDAP Group Permissions

When you configure a SAML authentication provider backed by OpenLDAP, the SAML response might return only a subset of the groups that a user belongs to. The exact groups returned depend on the configuration of your external authentication provider.

Rancher assigns user permissions based strictly on the groups provided in the SAML response.



Even if you can search for and view specific OpenLDAP groups in the Rancher UI, you cannot use them to assign permissions if they are missing from the SAML response.

To assign permissions successfully, verify that your SAML authentication provider is configured to return all necessary OpenLDAP groups.

#### Configuring Microsoft Active Directory Federation Service (SAML)

If your organization uses Microsoft Active Directory Federation Services (AD FS) for user authentication, you can configure Rancher to allow your users to log in using their AD FS credentials.

#### Prerequisites

You must have Rancher installed.

- Obtain your Rancher Server URL. During AD FS configuration, substitute this URL for the `<RANCHER_SERVER>` placeholder.
- You must have a global administrator account on your Rancher installation.

You must have a Microsoft AD FS Server [https://docs.microsoft.com/en-us/windows-server/identity/active-directory-federation-services] configured.

- Obtain your AD FS Server IP/DNS name. During AD FS configuration, substitute this IP/DNS name for

the <AD_SERVER> placeholder.

- You must have access to add Relying Party Trusts [https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/create-a-relying-party-trust] on your AD FS Server.

**Setup Outline**

Setting up Microsoft AD FS with Rancher Server requires configuring AD FS on your Active Directory server, and configuring Rancher to utilize your AD FS server. The following pages serve as guides for setting up Microsoft AD FS authentication on your Rancher installation.

- 1. Configuring Microsoft AD FS for Rancher
- 2. Configuring Rancher for Microsoft AD FS



*SAML Provider Caveats*

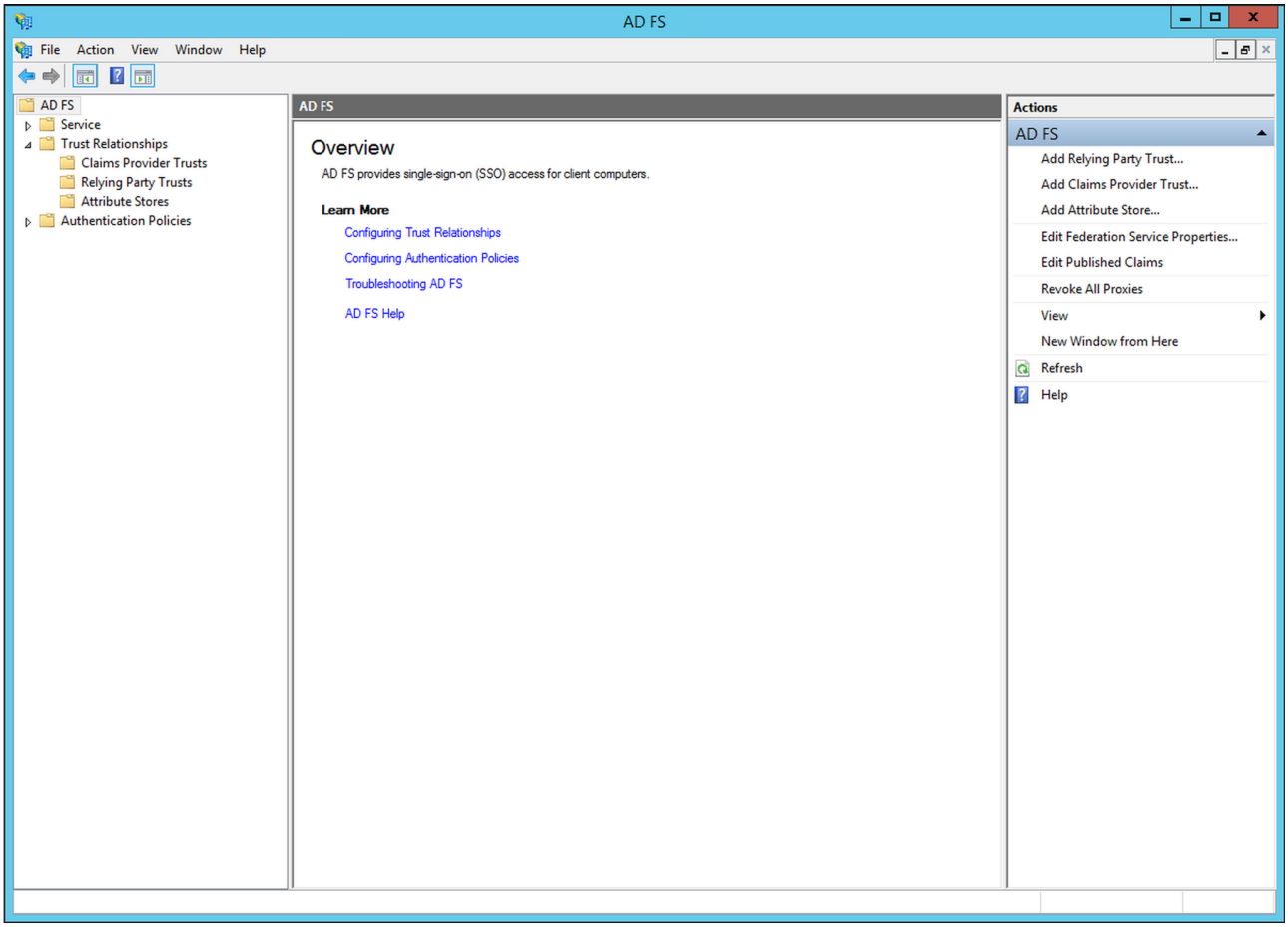
- Users and groups aren't validated when you assign permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. `UID Field`) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

**Next: Configuring Microsoft AD FS for Rancher**

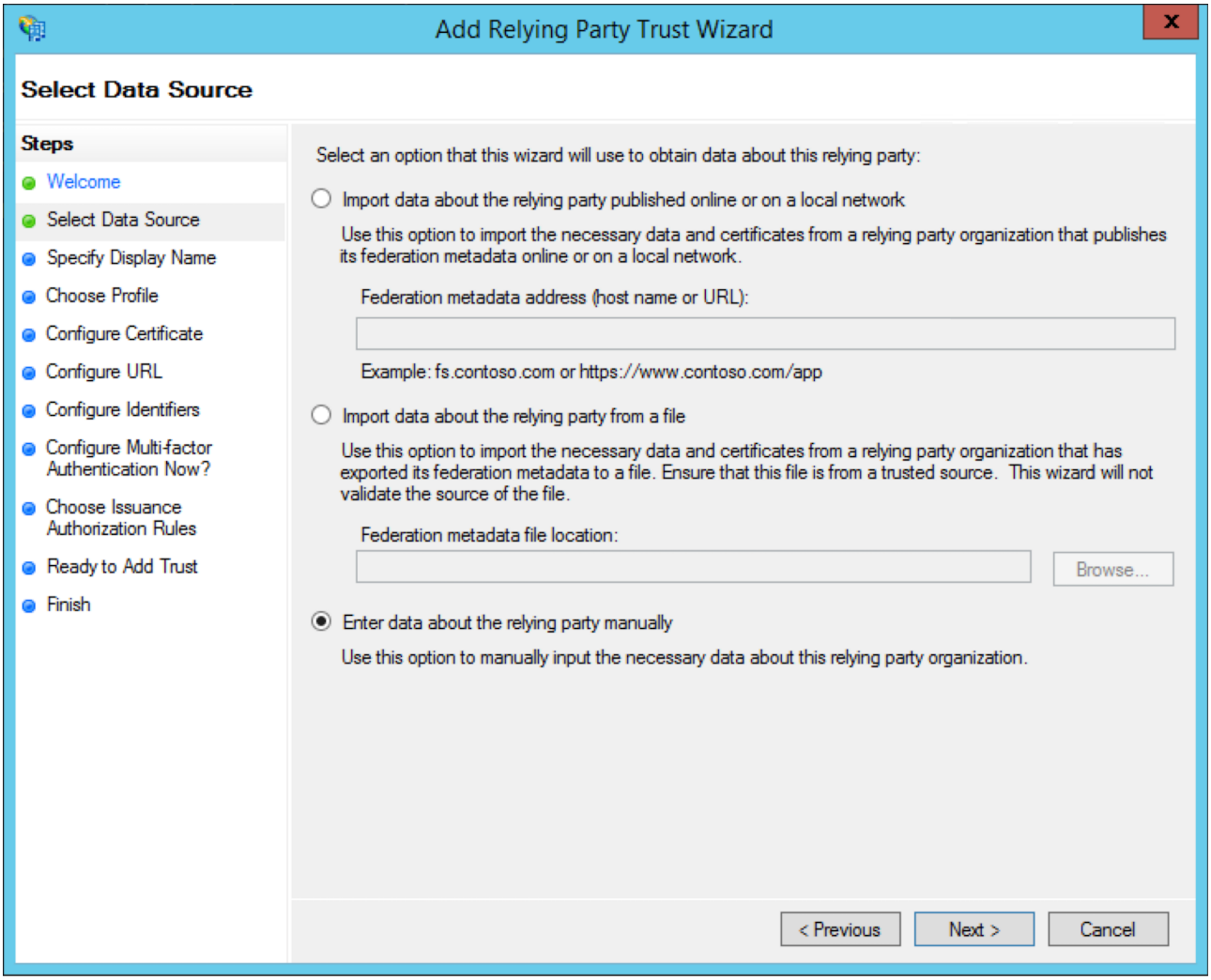
**1. Configuring Microsoft AD FS for SUSE Rancher Prime**

Before you configure Rancher to support Active Directory Federation Service (AD FS), you must add Rancher as a relying party trust [https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/technical-reference/understanding-key-ad-fs-concepts] in AD FS.

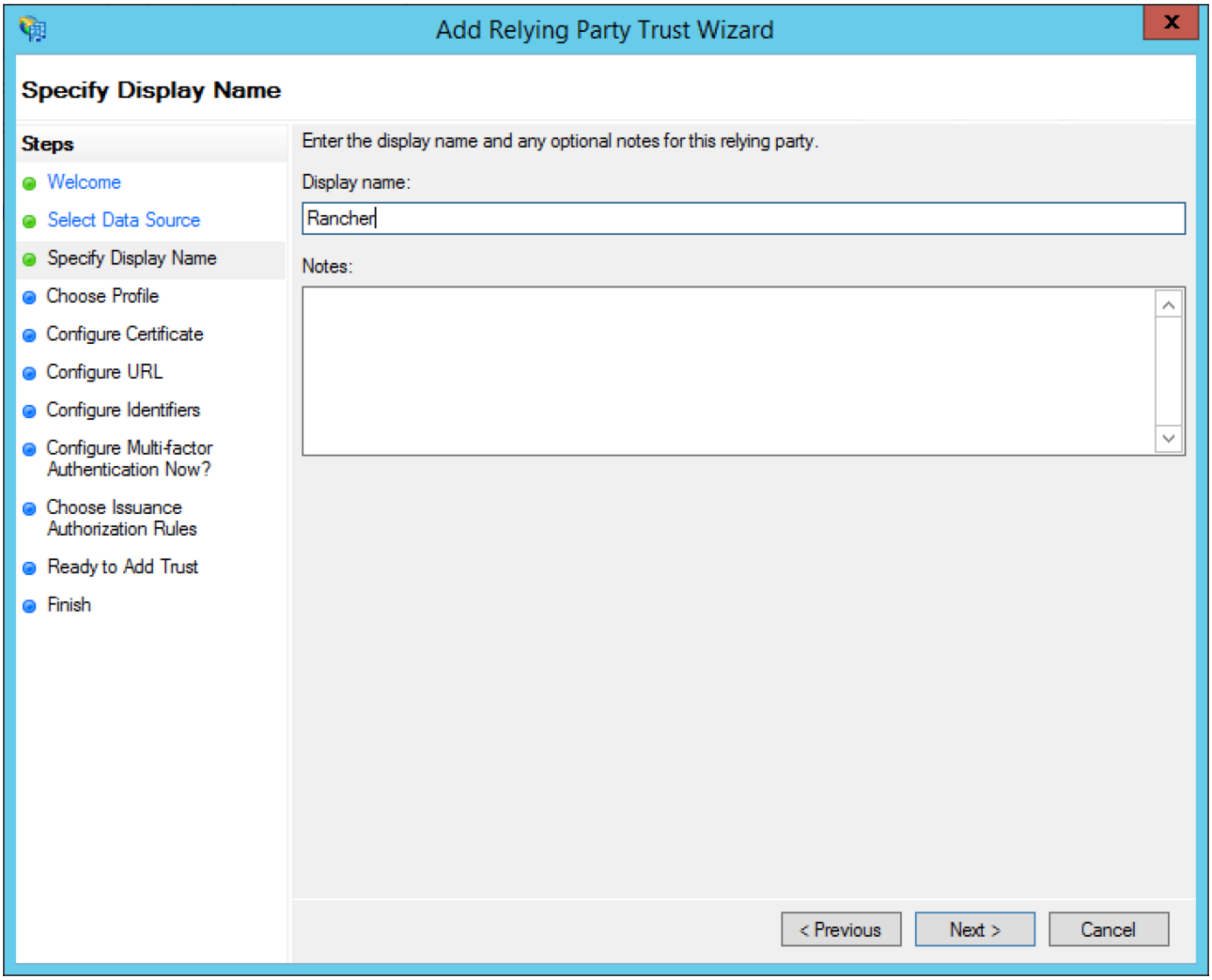
1. Log into your AD server as an administrative user.
2. Open the **AD FS Management** console. Select **Add Relying Party Trust...** from the **Actions** menu and click **Start**.



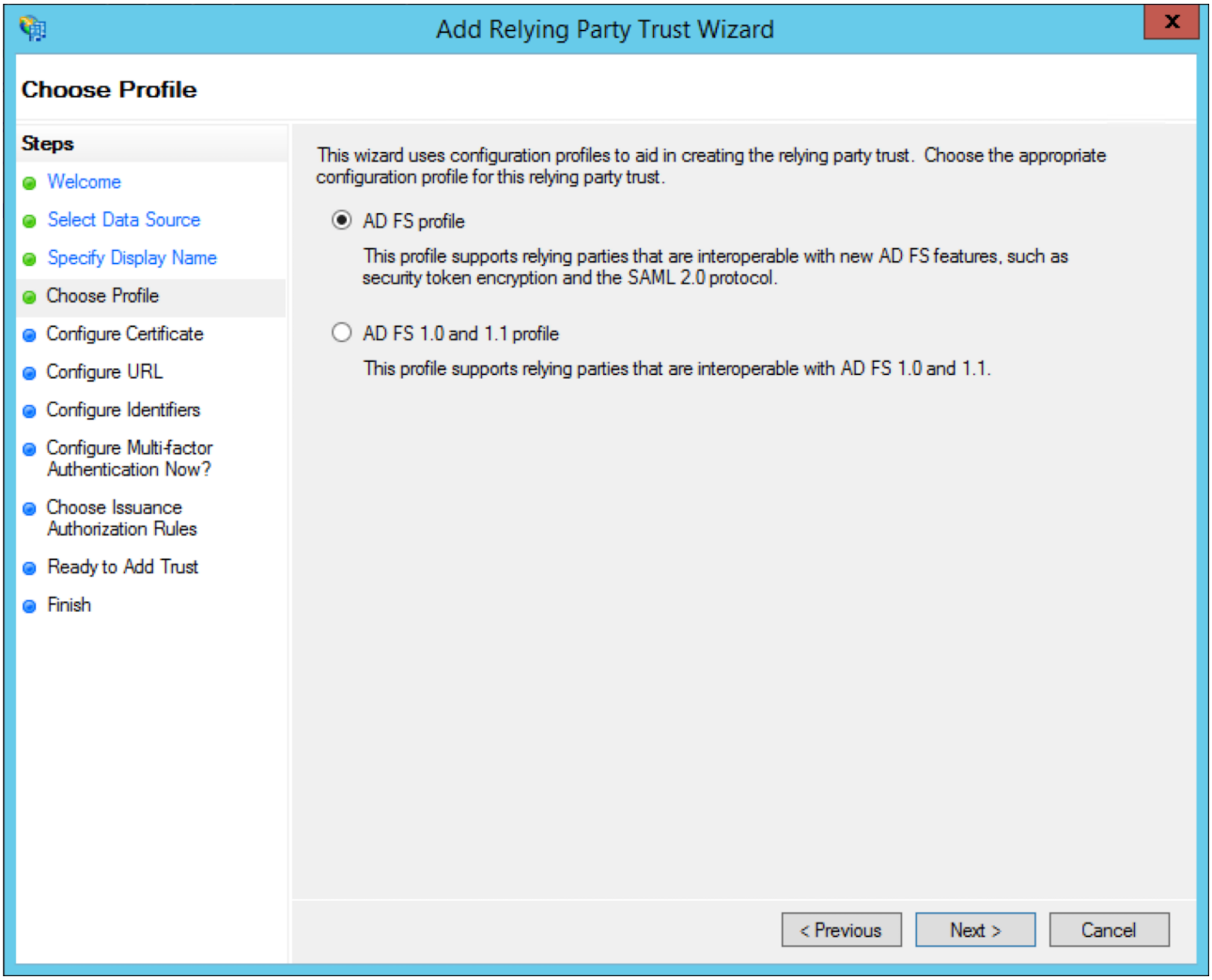

3. Select **Enter data about the relying party manually** as the option for obtaining data about the relying party.



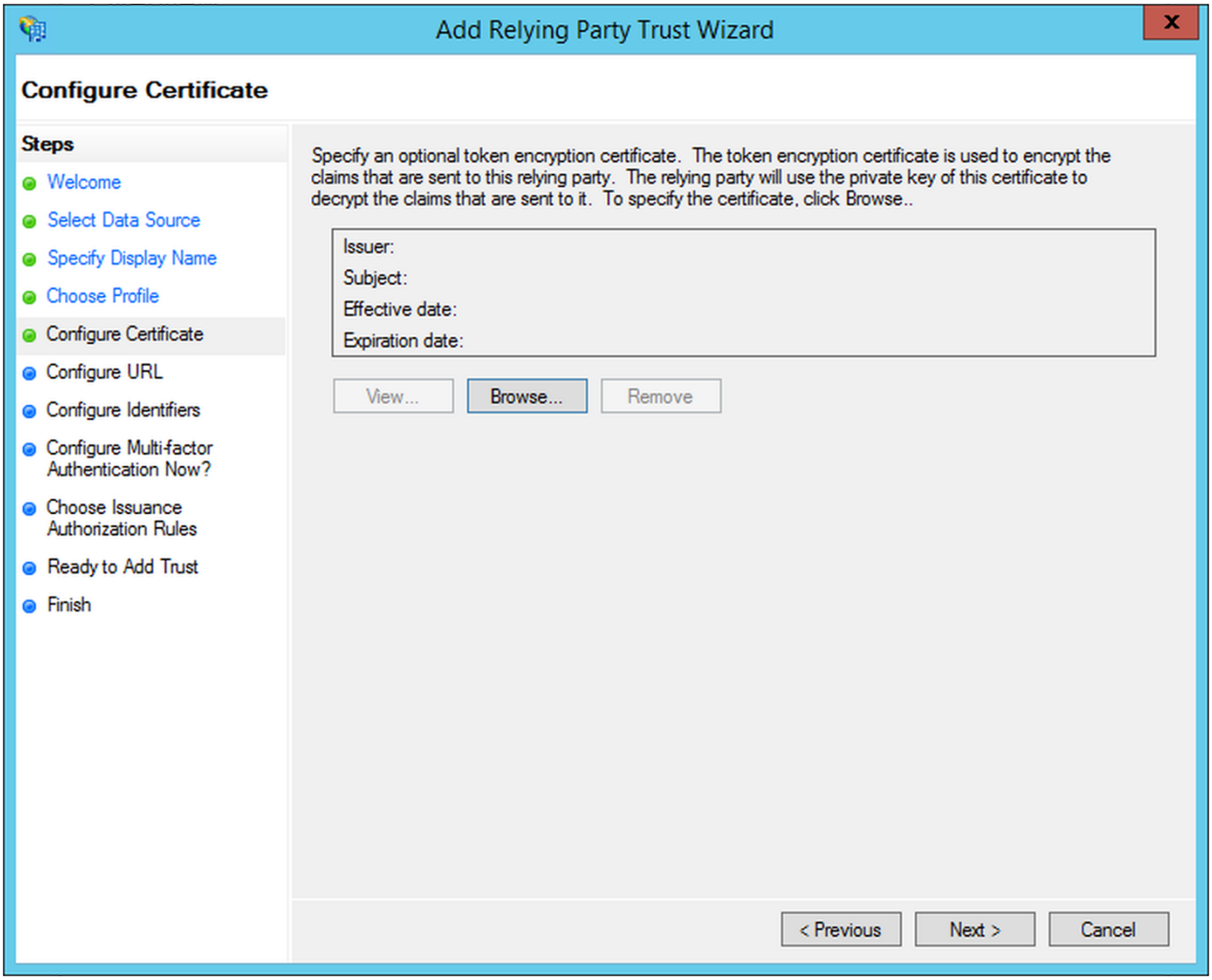

4. Enter your desired **Display name** for your Relying Party Trust. For example, `Rancher`.



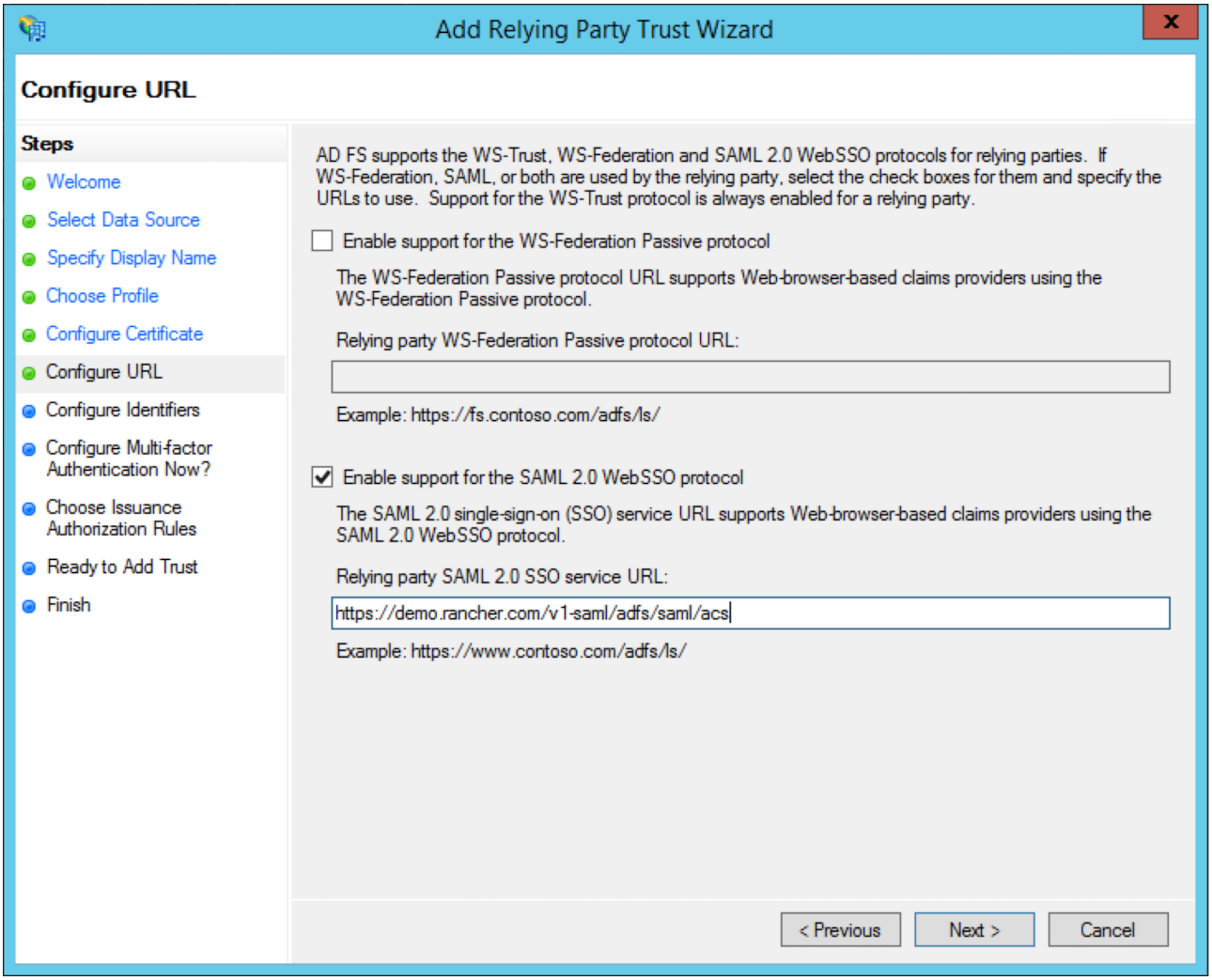

5. Select **AD FS profile** as the configuration profile for your relying party trust.



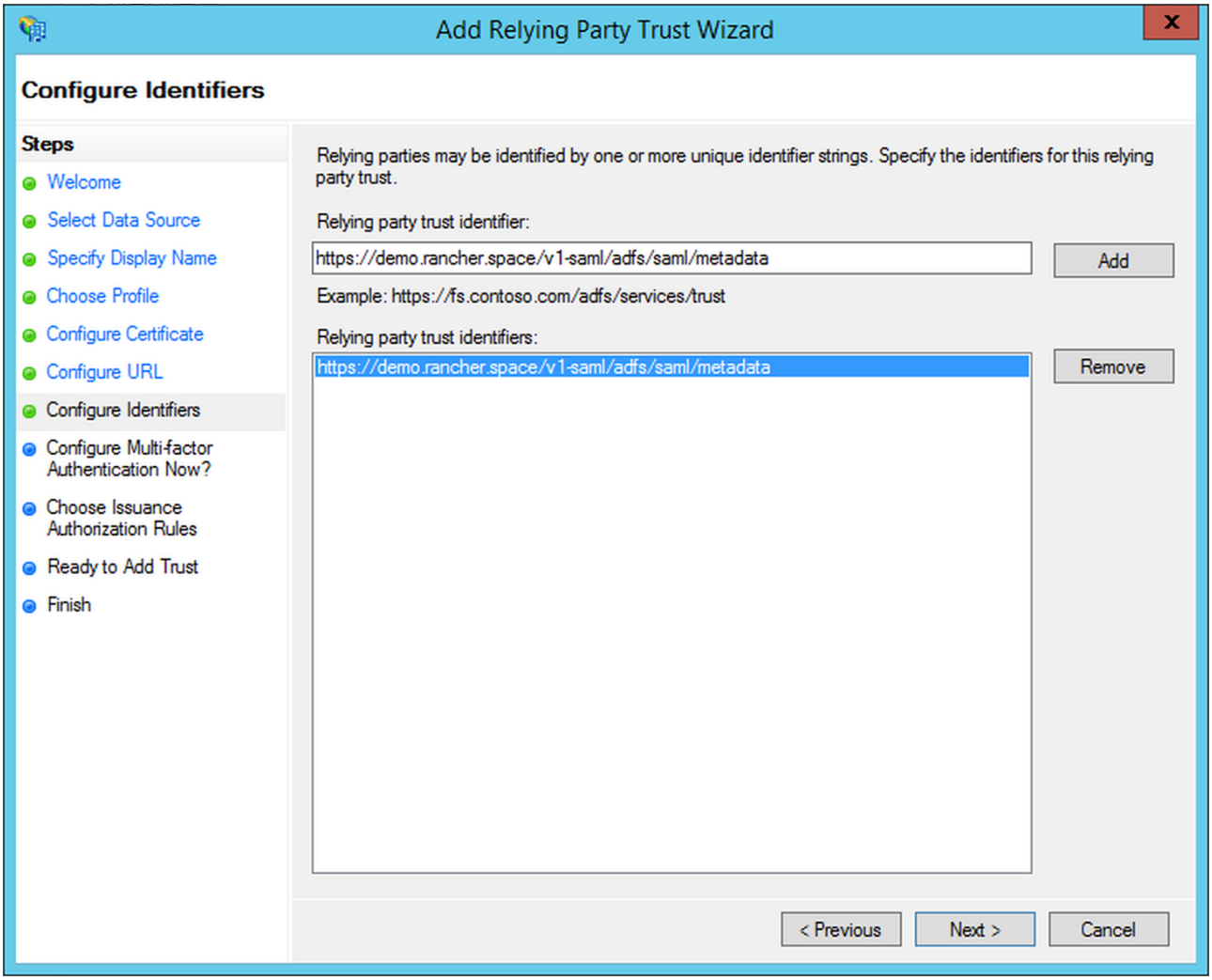

6. Leave the **optional token encryption certificate** empty, as Rancher AD FS will not be using one.



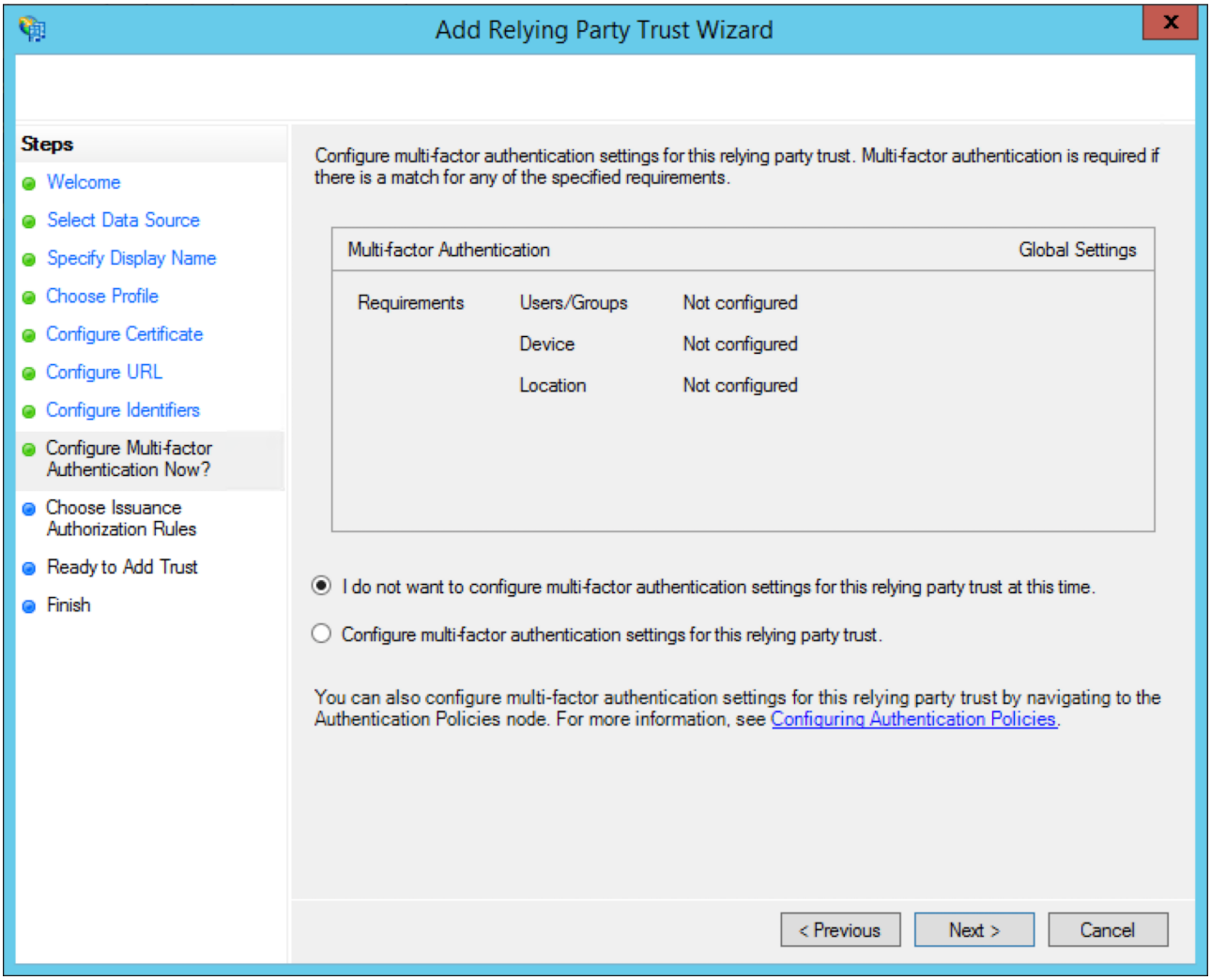

7. Select **Enable support for the SAML 2.0 WebSSO protocol** and enter `https://<rancher-server>/vl-saml/adfs/saml/acs` for the service URL.



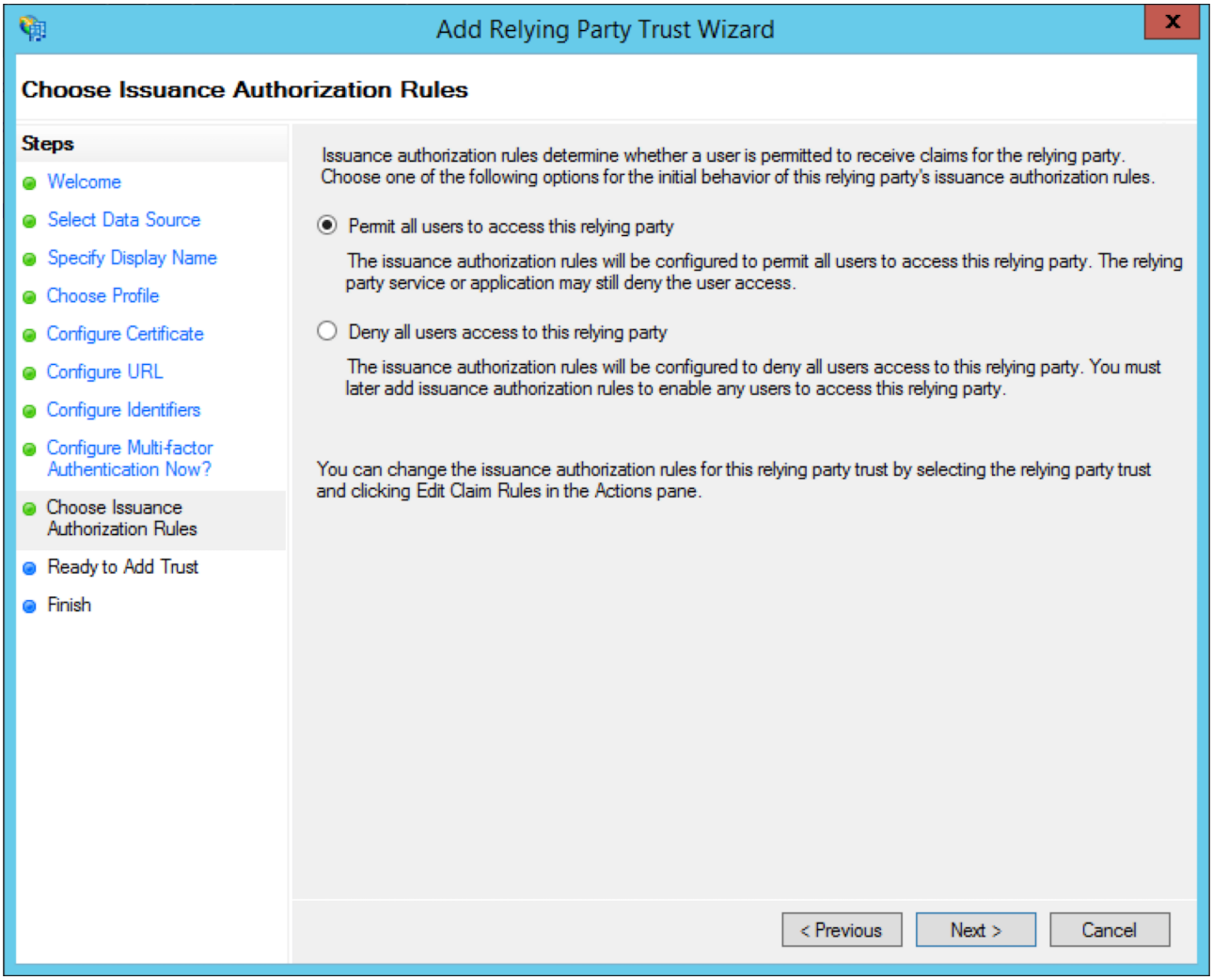

8. Add `https://<rancher-server>/v1-saml/adfs/saml/metadata` as the **Relying party trust identifier**.



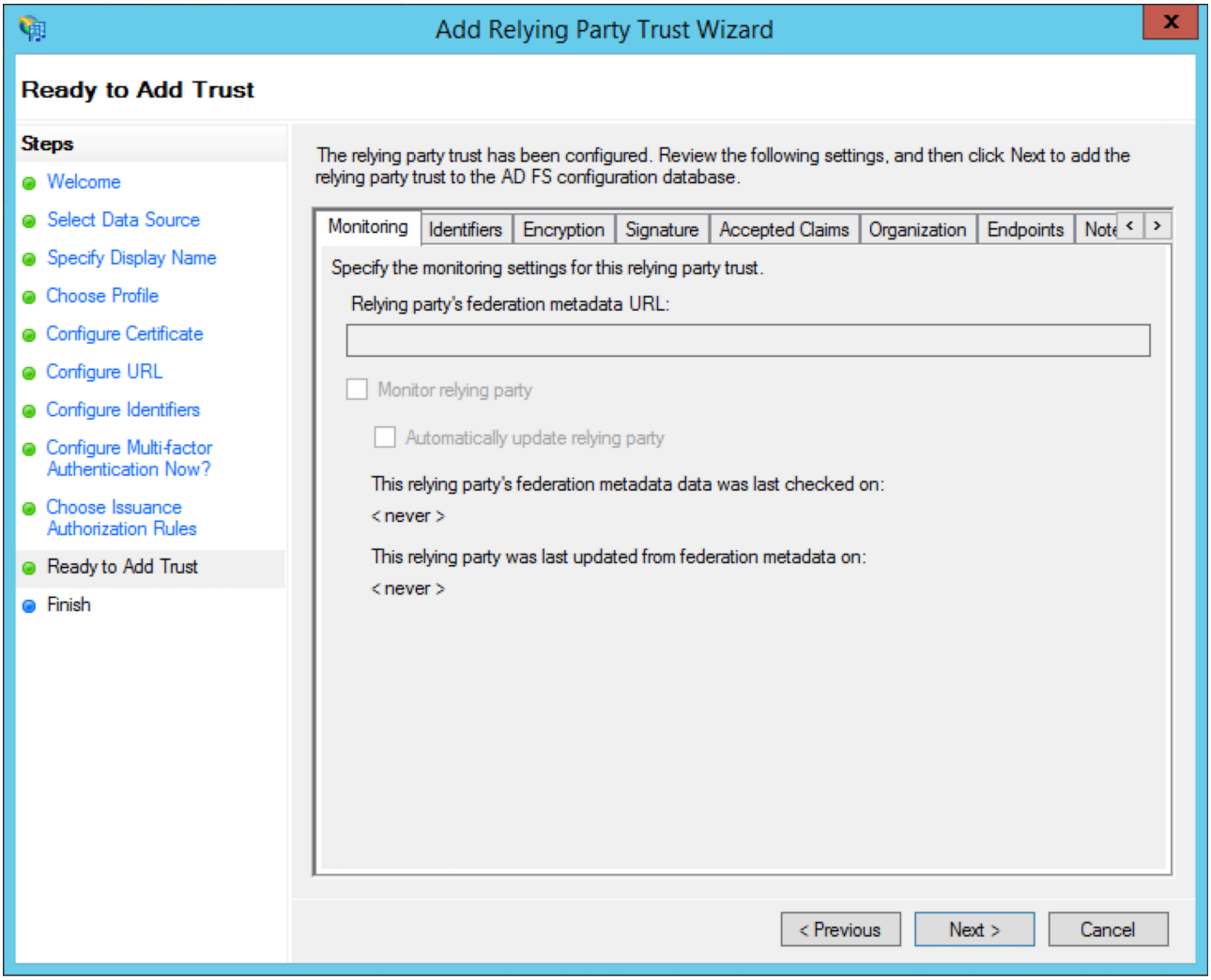

9. This tutorial will not cover multi-factor authentication; please refer to the Microsoft documentation [https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/configure-additional-authentication-methods-for-ad-fs] if you would like to configure multi-factor authentication.



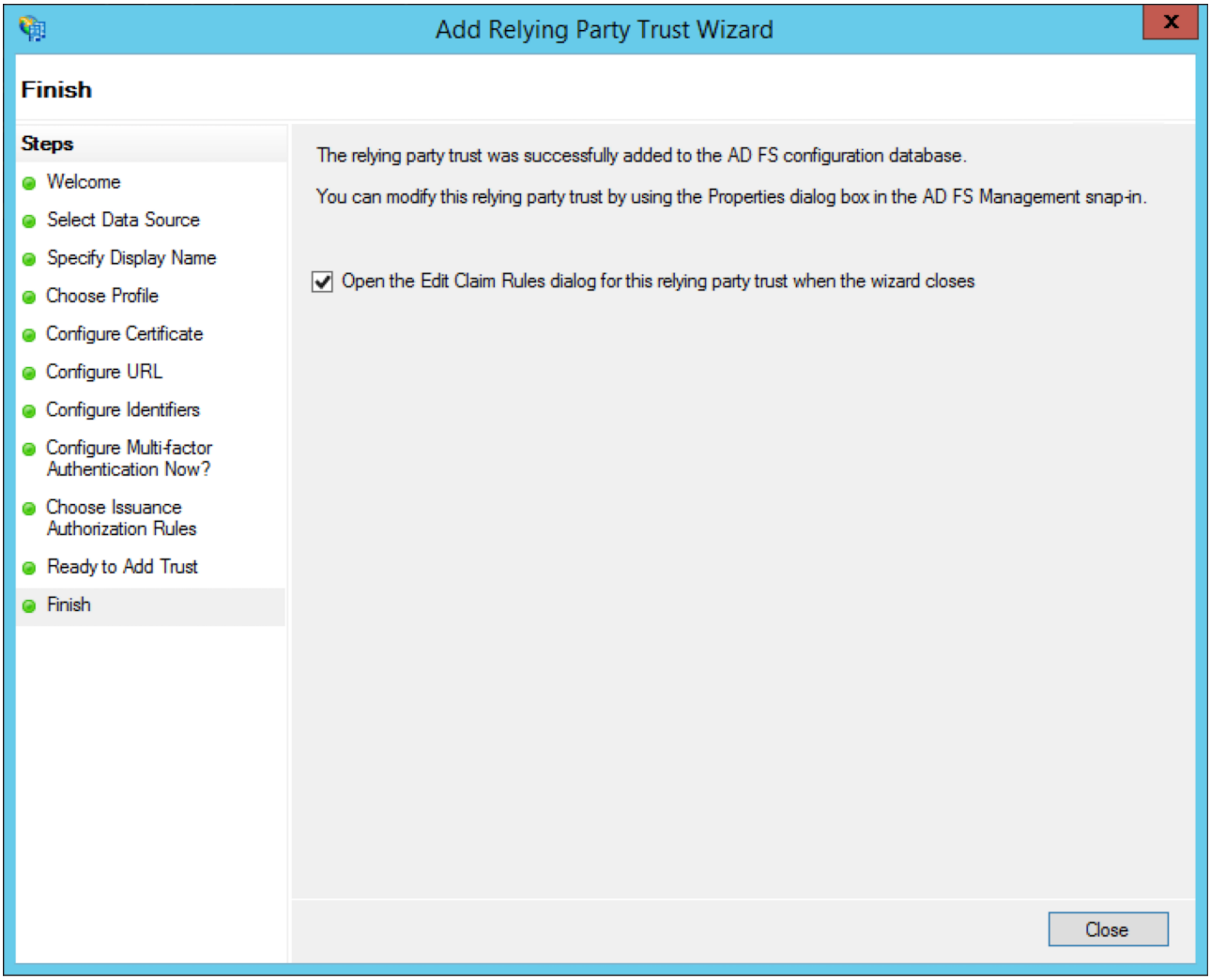

10. From **Choose Issuance Authorization RUles**, you may select either of the options available according to use case. However, for the purposes of this guide, select **Permit all users to access this relying party**.



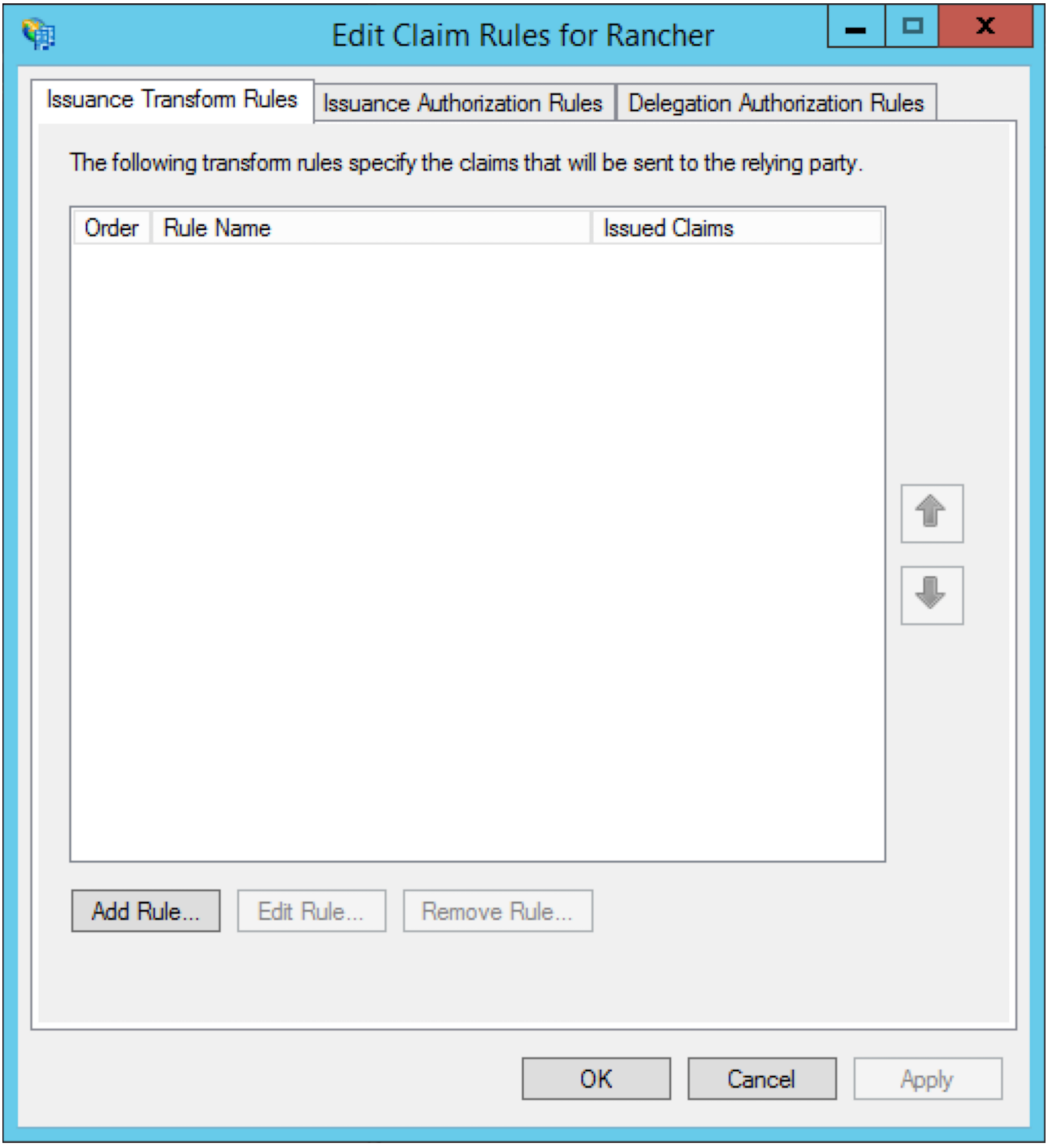

11. After reviewing your settings, select **Next** to add the relying party trust.



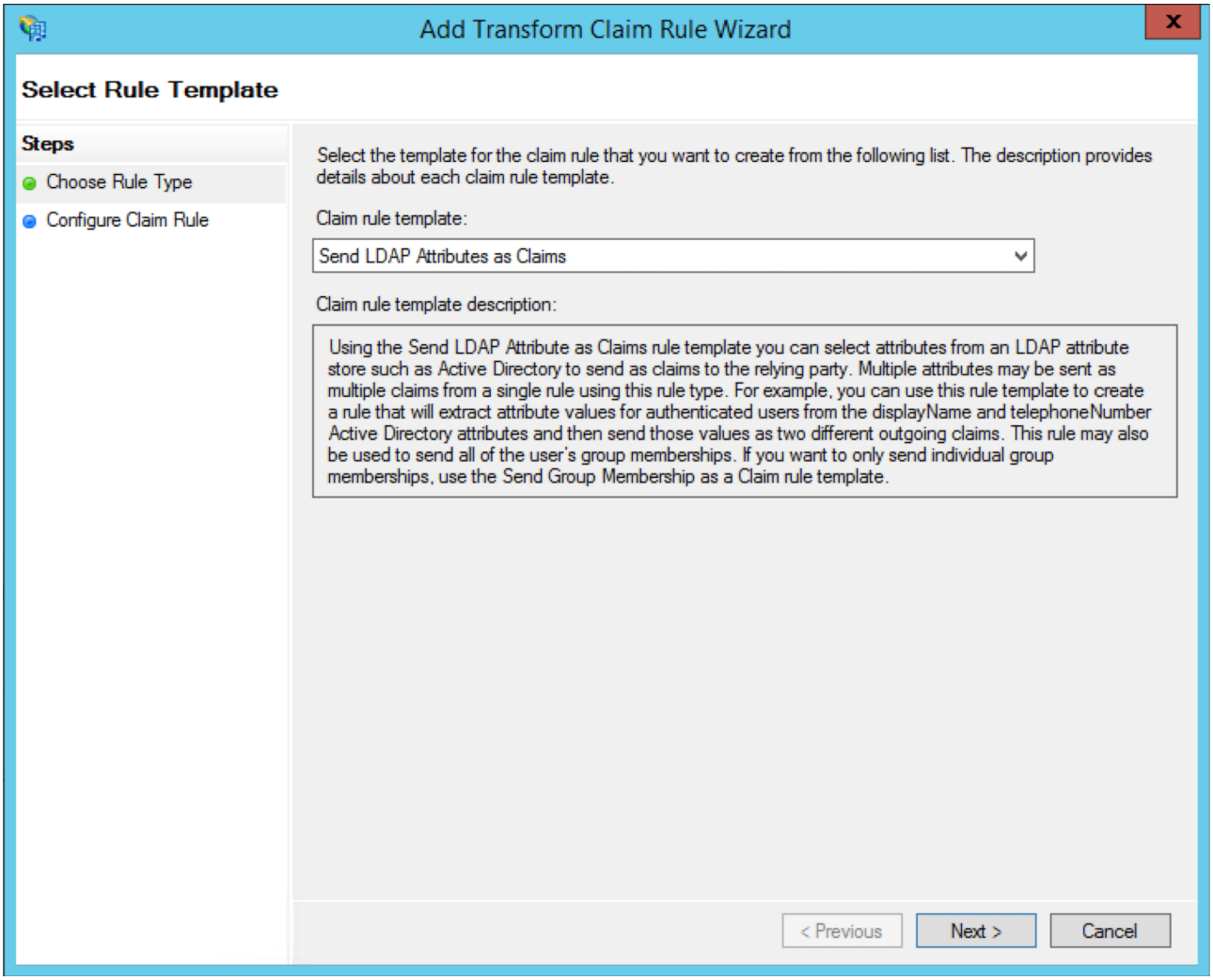

12. Select **Open the Edit Claim Rules...** and click **Close**.




13. On the **Issuance Transform Rules** tab, click **Add Rule...**.

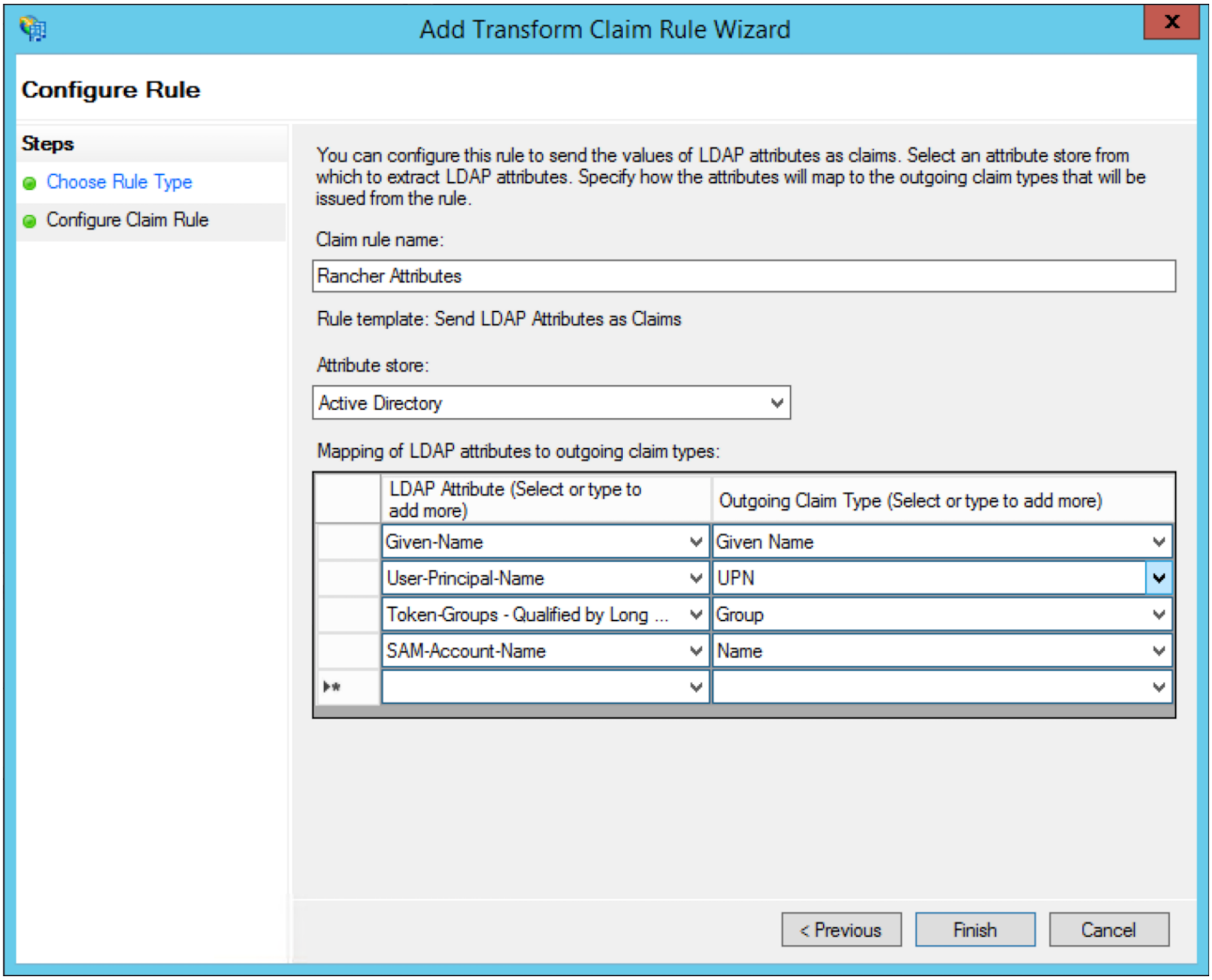



14. Select **Send LDAP Attributes as Claims** as the **Claim rule template**.




15. Set the **Claim rule name** to your desired name (for example, `Rancher Attributes`) and select **Active Directory** as the **Attribute store**. Create the following mapping to reflect the table below:

| LDAP Attribute |
| --- |
| Outgoing Claim Type |
| Given-Name |
| Given Name |
| User-Principal-Name |
| UPN |
| Token-Groups - Qualified by Long Domain Name |
| Group |
| SAM-Account-Name |
| Name |




16. Download the `federationmetadata.xml` from your AD server at:

```
https://<AD_SERVER>/federationmetadata/2007-06/federationmetadata.xml
```

**Result:** You've added Rancher as a relying trust party. Now you can configure Rancher to leverage AD.

**Next: Configuring Rancher for Microsoft AD FS**

#### 2. Configuring SUSE Rancher Prime for Microsoft AD FS

After you complete Configuring Microsoft AD FS for Rancher, enter your Active Directory Federation Service (AD FS) information into Rancher so that AD FS users can authenticate with Rancher.



*Important Notes For Configuring Your ADFS Server:*

- The SAML 2.0 WebSSO Protocol Service URL is: `https://<RANCHER_SERVER>/v1-saml/adfs/saml/acs`
- The Relying Party Trust identifier URL is: `https://<RANCHER_SERVER>/v1-saml/adfs/saml/metadata`
- You must export the `federationmetadata.xml` file from your AD FS server. This can be found at: `https://<AD_SERVER>/federationmetadata/2007-06/federationmetadata.xml`

1. In the top left corner, click ☰ > **Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **ADFS**.
4. Complete the **Configure AD FS Account** form. Microsoft AD FS lets you specify an existing Active Directory (AD) server. The configuration section below describe how you can map AD attributes to fields within Rancher.
5. After you complete the **Configure AD FS Account** form, click **Enable**.

   Rancher redirects you to the AD FS login page. Enter credentials that authenticate with Microsoft AD FS to validate your Rancher AD FS configuration.

   You may have to disable your popup blocker to see the AD FS login page.

**Result:** Rancher is configured to work with MS FS. Your users can now sign into Rancher using their MS FS logins.

#### Configuration

| Field | Description |
|---|---|
| Display Name Field | The AD attribute that contains the display name of users.<br><br>Example: `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name` |
| User Name Field | The AD attribute that contains the user name/given name.<br><br>Example: `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname` |
| UID Field | An AD attribute that is unique to every user.<br><br>Example: `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn` |
| Groups Field | Make entries for managing group memberships.<br><br>Example: `http://schemas.xmlsoap.org/claims/Group` |
| Rancher API Host | The URL for your Rancher Server. |
| Private Key / Certificate | This is a key-certificate pair to create a secure shell between Rancher and your AD FS. Ensure you set the Common Name (CN) to your Rancher Server URL.<br><br>Certificate creation command |
| Metadata XML | The `federationmetadata.xml` file exported from your AD FS server.<br><br>You can find this file at `https://<AD_SERVER>/federationmetadata/2007-06/federationmetadata.xml`. |

#### Example Certificate Creation Command

You can generate a certificate using an openssl command. For example:

```
openssl req -x509 -newkey rsa:2048 -keyout myservice.key -out
myservice.cert -days 365 -nodes -subj "/CN=myservice.example.com"
```

#### Configuring SAML Single Logout (SLO)

Rancher supports the ability to configure SAML SLO. Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options. The steps below outline configuration from the application GUI:



The **Log Out behavior** configuration section only appears if the SAML authentication provider allows for `SAML SLO`.

1. Sign in to Rancher using a standard user or an administrator role to configure SAML SLO.
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

#### SAML and OpenLDAP Group Permissions

When you configure a SAML authentication provider backed by OpenLDAP, the SAML response might return only a subset of the groups that a user belongs to. The exact groups returned depend on the configuration of your external authentication provider.

Rancher assigns user permissions based strictly on the groups provided in the SAML response.



Even if you can search for and view specific OpenLDAP groups in the Rancher UI, you cannot use them to assign permissions if they are missing from the SAML response.

To assign permissions successfully, verify that your SAML authentication provider is configured to return all necessary OpenLDAP groups.

#### Configuring OpenLDAP

If your organization uses LDAP for user authentication, you can configure Rancher to communicate with an OpenLDAP server to authenticate users. This allows Rancher admins to control access to clusters and

projects based on users and groups managed externally in the organisation's central user repository, while allowing end-users to authenticate with their LDAP credentials when logging in to the Rancher UI.

#### Prerequisites

Rancher must be configured with a LDAP bind account (aka service account) to search and retrieve LDAP entries pertaining to users and groups that should have access. It is recommended to not use an administrator account or personal account for this purpose and instead create a dedicated account in OpenLDAP with read-only access to users and groups under the configured search base (see below).

> ### Using TLS?
>
> If the certificate used by the OpenLDAP server is self-signed or not from a recognised certificate authority, make sure have at hand the CA certificate (concatenated with any intermediate certificates) in PEM format. You will have to paste in this certificate during the configuration so that Rancher is able to validate the certificate chain.

#### Configure OpenLDAP in Rancher

Configure the settings for the OpenLDAP server, groups and users. For help filling out each field, refer to the configuration reference.

> Before you proceed with the configuration, please familiarise yourself with the concepts of External Authentication Configuration and Principal Users.

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **OpenLDAP**. Fill out the **Configure an OpenLDAP server** form.
4. Click **Enable**.

#### Test Authentication

Once you have completed the configuration, proceed by testing the connection to the OpenLDAP server. Authentication with OpenLDAP will be enabled implicitly if the test is successful.



The OpenLDAP user pertaining to the credentials entered in this step will be mapped to the local principal account and assigned administrator privileges in Rancher. You should therefore make a conscious decision on which LDAP account you use to perform this step.

1. Enter the **username** and **password** for the OpenLDAP account that should be mapped to the local principal account.
2. Click **Authenticate With OpenLDAP** to test the OpenLDAP connection and finalise the setup.

**Result:**

- OpenLDAP authentication is configured.
- The LDAP user pertaining to the entered credentials is mapped to the local principal (administrative)

account.



You will still be able to login using the locally configured `admin` account and password in case of a disruption of LDAP services.

#### Annex: Troubleshooting

If you are experiencing issues while testing the connection to the OpenLDAP server, first double-check the credentials entered for the service account as well as the search base configuration. You may also inspect the Rancher logs to help pinpointing the problem cause. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

#### OpenLDAP Configuration Reference

For further details on configuring OpenLDAP authentication, refer to the official documentation. [https://www.openldap.org/doc/]

> Before you proceed with the configuration, please familiarize yourself with the concepts of External Authentication Configuration and Principal Users.

#### Background: OpenLDAP Authentication Flow

1. When a user attempts to login with LDAP credentials, Rancher creates an initial bind to the LDAP server using a service account with permissions to search the directory and read user/group attributes.
2. Rancher then searches the directory for the user by using a search filter based on the provided username and configured attribute mappings.
3. Once the user has been found, they are authenticated with another LDAP bind request using the user's DN and provided password.
4. Once authentication succeeded, Rancher then resolves the group memberships both from the membership attribute in the user's object and by performing a group search based on the configured user mapping attribute.

#### OpenLDAP Server Configuration

You will need to enter the address, port, and protocol to connect to your OpenLDAP server. 389 is the standard port for insecure traffic, 636 for TLS traffic.

> **Using TLS?**
>
> If the certificate used by the OpenLDAP server is self-signed or not from a recognized certificate authority, make sure have at hand the CA certificate (concatenated with any intermediate certificates) in PEM format. You will have to paste in this certificate during the configuration so that Rancher is able to validate the certificate chain.

If you are in doubt about the correct values to enter in the user/group Search Base configuration fields, consult your LDAP administrator or refer to the section Identify Search Base and Schema using ldapsearch in the Active Directory authentication documentation.

*Table 6. OpenLDAP Server Parameters*

| Parameter | Description |
|---|---|
| Hostname | Specify the hostname or IP address of the OpenLDAP server |
| Port | Specify the port at which the OpenLDAP server is listening for connections. Unencrypted LDAP normally uses the standard port of 389, while LDAPS uses port 636. |
| TLS | Check this box to enable LDAP over SSL/TLS (commonly known as LDAPS). You will also need to paste in the CA certificate if the server uses a self-signed/enterprise-signed certificate. |
| Server Connection Timeout | The duration in number of seconds that Rancher waits before considering the server unreachable. |
| Service Account Distinguished Name | Enter the Distinguished Name (DN) of the user that should be used to bind, search and retrieve LDAP entries. |
| Service Account Password | The password for the service account. |
| User Search Base | Enter the Distinguished Name of the node in your directory tree from which to start searching for user objects. All users must be descendents of this base DN. For example: "ou=people,dc=acme,dc=com". |
| Group Search Base | If your groups live under a different node than the one configured under `User Search Base` you will need to provide the Distinguished Name here. Otherwise leave this field empty. For example: "ou=groups,dc=acme,dc=com". |

#### User/Group Schema Configuration

If your OpenLDAP directory deviates from the standard OpenLDAP schema, you must complete the **Customize Schema** section to match it.

Note that the attribute mappings configured in this section are used by Rancher to construct search filters and resolve group membership. It is therefore always recommended to verify that the configuration here matches the schema used in your OpenLDAP.

If you are unfamiliar with the user/group schema used in the OpenLDAP server, consult your LDAP administrator or refer to the section Identify Search Base and Schema using ldapsearch in the Active Directory authentication documentation.

#### User Schema Configuration

The table below details the parameters for the user schema configuration.

*Table 7. User Schema Configuration Parameters*

| Parameter | Description |
|---|---|
| Object Class | The name of the object class used for user objects in your domain. If defined, only specify the name of the object class - *don't* include it in an LDAP wrapper such as &(objectClass=xxxx) |
| Username Attribute | The user attribute whose value is suitable as a display name. |
| Login Attribute | The attribute whose value matches the username part of credentials entered by your users when logging in to Rancher. This is typically `uid`. |

| Parameter | Description |
|---|---|
| User Member Attribute | The user attribute containing the Distinguished Name of groups a user is member of. Usually this is one of `memberOf` or `isMemberOf`. |
| Search Attribute | When a user enters text to add users or groups in the UI, Rancher queries the LDAP server and attempts to match users by the attributes provided in this setting. Multiple attributes can be specified by separating them with the pipe ("\|") symbol. |
| User Enabled Attribute | If the schema of your OpenLDAP server supports a user attribute whose value can be evaluated to determine if the account is disabled or locked, enter the name of that attribute. The default OpenLDAP schema does not support this and the field should usually be left empty. |
| Disabled Status Bitmask | This is the value for a disabled/locked user account. The parameter is ignored if `User Enabled Attribute` is empty. |

#### Group Schema Configuration

The table below details the parameters for the group schema configuration.

*Table 8. Group Schema Configuration Parameters*

| Parameter | Description |
|---|---|
| Object Class | The name of the object class used for group entries in your domain. If defined, only specify the name of the object class - *don't* include it in an LDAP wrapper such as &(objectClass=xxxx) |
| Name Attribute | The group attribute whose value is suitable for a display name. |
| Group Member User Attribute | The name of the **user attribute** whose format matches the group members in the `Group Member Mapping Attribute`. |
| Group Member Mapping Attribute | The name of the group attribute containing the members of a group. |
| Search Attribute | Attribute used to construct search filters when adding groups to clusters or projects in the UI. See description of user schema `Search Attribute`. |
| Group DN Attribute | The name of the group attribute whose format matches the values in the user's group membership attribute. See `User Member Attribute`. |
| Nested Group Membership | This settings defines whether Rancher should resolve nested group memberships. Use only if your organization makes use of these nested memberships (ie. you have groups that contain other groups as members). This option is disabled if you are using Shibboleth. |

#### Configuring Shibboleth (SAML)

If your organization uses Shibboleth Identity Provider (IdP) for user authentication, you can configure Rancher to allow your users to log in to Rancher using their Shibboleth credentials.

In this configuration, when Rancher users log in, they will be redirected to the Shibboleth IdP to enter their credentials. After authentication, they will be redirected back to the Rancher UI.

If you also configure OpenLDAP as the back end to Shibboleth, it will return a SAML assertion to Rancher with user attributes that include groups. Then the authenticated user will be able to access resources in Rancher that their groups have permissions for.

> The instructions in this section assume that you understand how Rancher, Shibboleth, and OpenLDAP work together. For a more detailed explanation of how it works, refer to this page.

#### Setting up Shibboleth in Rancher

##### Shibboleth Prerequisites

> - You must have a Shibboleth IdP Server configured.
> - Following are the Rancher Service Provider URLs needed for configuration: Metadata URL: `https://<rancher-server>/v1-saml/shibboleth/saml/metadata` Assertion Consumer Service (ACS) URL: `https://<rancher-server>/v1-saml/shibboleth/saml/acs`
> - Export a `metadata.xml` file from your IdP Server. For more information, see the Shibboleth documentation. [https://wiki.shibboleth.net/confluence/display/SP3/Home]

##### Configure Shibboleth in Rancher

If your organization uses Shibboleth for user authentication, you can configure Rancher to allow your users to log in using their IdP credentials.

1. In the top left corner, click ☰ **> Users & Authentication**.
2. In the left navigation menu, click **Auth Provider**.
3. Click **Shibboleth**.
4. Complete the **Configure Shibboleth Account** form. Shibboleth IdP lets you specify what data store you want to use. You can either add a database or use an existing ldap server. For example, if you select your Active Directory (AD) server, the examples below describe how you can map AD attributes to fields within Rancher.
    a. **Display Name Field**: Enter the AD attribute that contains the display name of users (example: `displayName`).
    b. **User Name Field**: Enter the AD attribute that contains the user name/given name (example: `givenName`).
    c. **UID Field**: Enter an AD attribute that is unique to every user (example: `sAMAccountName`, `distinguishedName`).
    d. **Groups Field**: Make entries for managing group memberships (example: `memberOf`).
    e. **Rancher API Host**: Enter the URL for your Rancher Server.

f. **Private Key** and **Certificate**: This is a key-certificate pair to create a secure shell between Rancher and your IdP.

   You can generate one using an openssl command. For example:

```
openssl req -x509 -newkey rsa:2048 -keyout myservice.key -out myservice.cert -days 365 -nodes -subj "/CN=myservice.example.com"
```

g. **IDP-metadata**: The `metadata.xml` file that you exported from your IdP server.

5. After you complete the **Configure Shibboleth Account** form, click **Enable**.

   Rancher redirects you to the IdP login page. Enter credentials that authenticate with Shibboleth IdP to validate your Rancher Shibboleth configuration.



You may have to disable your popup blocker to see the IdP login page.

**Result:** Rancher is configured to work with Shibboleth. Your users can now sign into Rancher using their Shibboleth logins.



*SAML Provider Caveats*

If you configure Shibboleth without OpenLDAP, the following caveats apply due to the fact that SAML Protocol does not support search or lookup for users or groups.

- There is no validation on users or groups when assigning permissions to them in Rancher.
- When adding users, the exact user IDs (i.e. UID Field) must be entered correctly. As you type the user ID, there will be no search for other user IDs that may match.
- When adding groups, you must select the group from the drop-down that is next to the text box. Rancher assumes that any input from the text box is a user.
- The group drop-down shows only the groups that you are a member of. However, if you have Administrator permissions, you can join a group that you are not a member of.

To enable searching for groups when assigning permissions in Rancher, you will need to configure a back end for the SAML provider that supports groups, such as OpenLDAP.

#### Configuring SAML Single Logout (SLO)

Rancher supports the ability to configure SAML SLO. Options include logging out of the Rancher application only, logging out of Rancher and registered applications tied to the external authentication provider, or a prompt asking the user to choose between the previous options. The steps below outline configuration from the application GUI:



The **Log Out behavior** configuration section only appears if the SAML authentication provider allows for `SAML SLO`.

1. Sign in to Rancher using a standard user or an administrator role to configure SAML SLO.
2. In the top left corner, click ☰ **> Users & Authentication**.

3. In the left navigation menu, click **Auth Provider**.
4. Under the section **Log Out behavior**, choose the appropriate SLO setting as described below:

| Setting | Description |
|---|---|
| Log out of Rancher and not authentication provider | Choosing this option will only logout the Rancher application and not external authentication providers. |
| Log out of Rancher and authentication provider (includes all other applications registered with authentication provider) | Choosing this option will logout Rancher and all external authentication providers along with any registered applications linked to the provider. |
| Allow the user to choose one of the above in an additional log out step | Choosing this option presents users with a choice of logout method as described above. |

#### Setting up OpenLDAP in Rancher

If you also configure OpenLDAP as the back end to Shibboleth, it will return a SAML assertion to Rancher with user attributes that include groups. Then authenticated users will be able to access resources in Rancher that their groups have permissions for.

#### OpenLDAP Prerequisites

Rancher must be configured with a LDAP bind account (aka service account) to search and retrieve LDAP entries pertaining to users and groups that should have access. It is recommended to not use an administrator account or personal account for this purpose and instead create a dedicated account in OpenLDAP with read-only access to users and groups under the configured search base (see below).

> ### Using TLS?
>
> If the certificate used by the OpenLDAP server is self-signed or not from a recognized certificate authority, make sure have at hand the CA certificate (concatenated with any intermediate certificates) in PEM format. You will have to paste in this certificate during the configuration so that Rancher is able to validate the certificate chain.

#### Configure OpenLDAP in Rancher

Configure the settings for the OpenLDAP server, groups and users. For help filling out each field, refer to the configuration reference. Note that nested group membership is not available for Shibboleth.

> Before you proceed with the configuration, please familiarise yourself with the concepts of External Authentication Configuration and Principal Users.

1. Log into the Rancher UI using the initial local `admin` account.
2. In the top left corner, click ☰ **> Users & Authentication**.
3. In the left navigation menu, click **Auth Provider**.
4. Click **Shibboleth** or, if SAML is already configured, **Edit Config**
5. Under **User and Group Search**, check **Configure an OpenLDAP server**

#### Troubleshooting

If you are experiencing issues while testing the connection to the OpenLDAP server, first double-check the credentials entered for the service account as well as the search base configuration. You may also inspect the Rancher logs to help pinpointing the problem cause. Debug logs may contain more detailed information about the error. Please refer to How can I enable debug logging in this documentation.

#### Group Permissions with Shibboleth and OpenLDAP

Because Shibboleth is a SAML provider, it doesn't support searching for groups. While a Shibboleth integration can validate user credentials, it can't be used to assign permissions to groups in Rancher without additional configuration.

One solution to this problem is to configure an OpenLDAP identity provider. With an OpenLDAP back end for Shibboleth, you will be able to search for groups in Rancher and assign them to resources such as clusters, projects, or namespaces from the Rancher UI.

#### Terminology

- **Shibboleth** is a single sign-on log-in system for computer networks and the Internet. It allows people to sign in using just one identity to various systems. It validates user credentials, but does not, on its own, handle group memberships.
- **SAML:** Security Assertion Markup Language, an open standard for exchanging authentication and authorization data between an identity provider and a service provider.
- **OpenLDAP:** a free, open-source implementation of the Lightweight Directory Access Protocol (LDAP). It is used to manage an organization's computers and users. OpenLDAP is useful for Rancher users because it supports groups. In Rancher, it is possible to assign permissions to groups so that they can access resources such as clusters, projects, or namespaces, as long as the groups already exist in the identity provider.
- **IdP or IDP:** An identity provider. OpenLDAP is an example of an identity provider.

#### Adding OpenLDAP Group Permissions to Rancher Resources

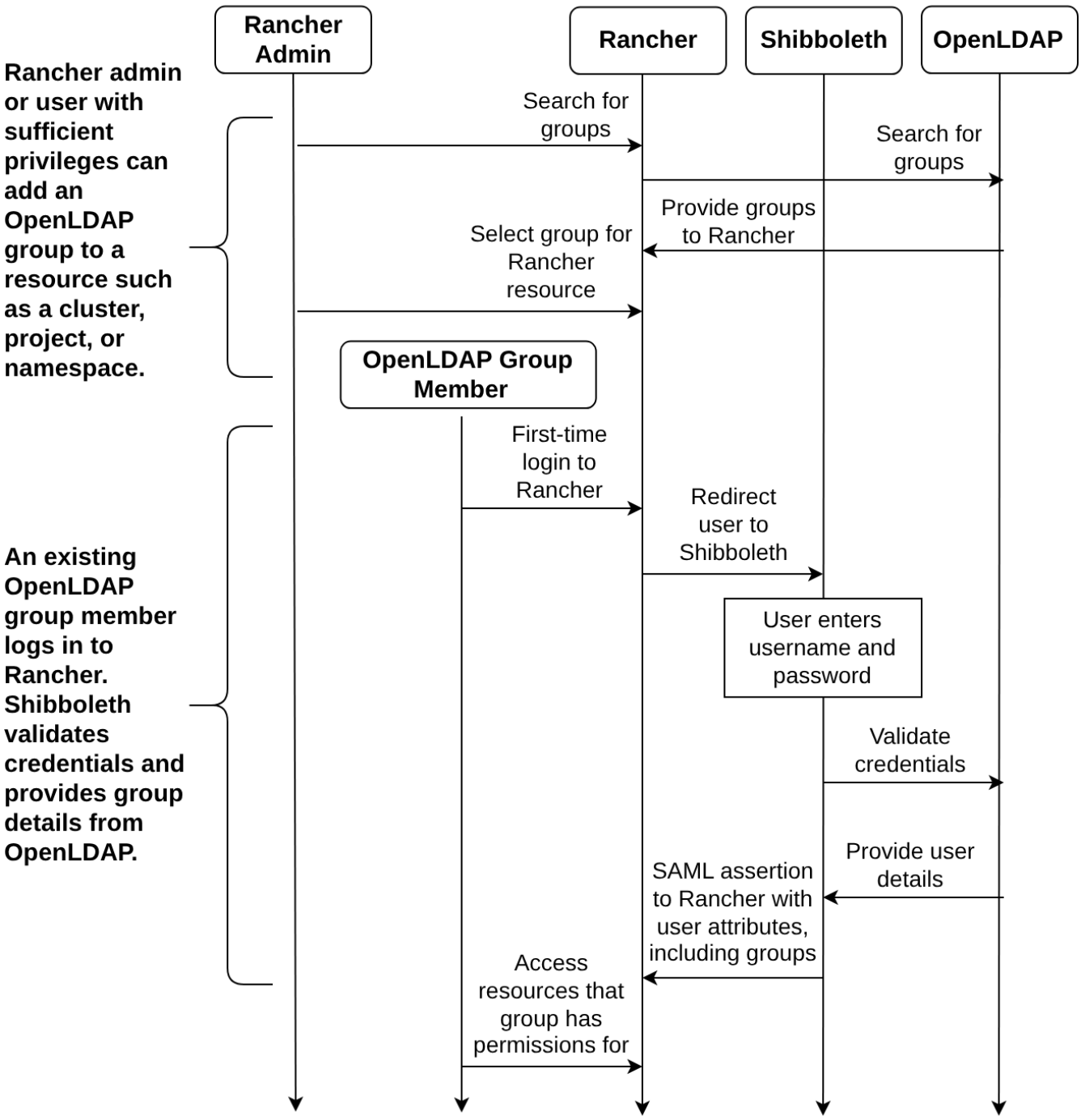
The diagram below illustrates how members of an OpenLDAP group can access resources in Rancher that the group has permissions for.

For example, a cluster owner could add an OpenLDAP group to a cluster so that they have permissions view most cluster level resources and create new projects. Then the OpenLDAP group members will have access to the cluster as soon as they log in to Rancher.

In this scenario, OpenLDAP allows the cluster owner to search for groups when assigning permissions. Without OpenLDAP, the functionality to search for groups would not be supported.

When a member of the OpenLDAP group logs in to Rancher, she is redirected to Shibboleth and enters her username and password.

Shibboleth validates her credentials, and retrieves user attributes from OpenLDAP, including groups. Then Shibboleth sends a SAML assertion to Rancher including the user attributes. Rancher uses the group data so that she can access all of the resources and permissions that her groups have permissions for.




**SAML and OpenLDAP Group Permissions**

When you configure a SAML authentication provider backed by OpenLDAP, the SAML response might return only a subset of the groups that a user belongs to. The exact groups returned depend on the configuration of your external authentication provider.

Rancher assigns user permissions based strictly on the groups provided in the SAML response.



Even if you can search for and view specific OpenLDAP groups in the Rancher UI, you cannot use them to assign permissions if they are missing from the SAML response.

To assign permissions successfully, verify that your SAML authentication provider is configured to return all necessary OpenLDAP groups.

## Managing Role-Based Access Control (RBAC)

Within Rancher, each person authenticates as a *user*, which is a login that grants you access to Rancher. As mentioned in Authentication, users can either be local or external.

After you configure external authentication, the users that display on the **Users** page changes.

- If you are logged in as a local user, only local users display.
- If you are logged in as an external user, both external and local users display.

### Users and Roles

Once the user logs in to Rancher, their *authorization*, or their access rights within the system, is determined by *global permissions*, and *cluster and project roles*.

- Global Permissions:

  Define user authorization outside the scope of any particular cluster.

- Cluster and Project Roles:

  Define user authorization inside the specific cluster or project where they are assigned the role.

Both global permissions and cluster and project roles are implemented on top of Kubernetes RBAC [https://kubernetes.io/docs/reference/access-authn-authz/rbac/]. Therefore, enforcement of permissions and roles is performed by Kubernetes.

### Global Permissions



Users with permissions to modify accounts or resources are considered privileged users, only grant these permissions to trusted users.

*Permissions* are individual access rights that you can assign when selecting a custom permission for a user.

Global Permissions define user authorization outside the scope of any particular cluster. Out-of-the-box, there are four default global permissions: `Administrator`, `Standard User` and `User-base`.

- **Administrator:** These users have full control over the entire Rancher system and all clusters within it.
- **Standard User:** These users can create new clusters and use them. Standard users can also assign other users permissions to their clusters.
- **User-Base:** User-Base users have login-access only.

You cannot update or delete the built-in Global Permissions.

### Global Permission Assignment

Global permissions for local users are assigned differently than users who log in to Rancher using external authentication.

### Global Permissions for New Local Users

When you create a new local user, you assign them a global permission as you complete the **Add User** form.

To see the default permissions for new users,

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. The **Role Templates** page has tabs for roles grouped by scope. Each table lists the roles in that scope. In the **Global** tab, in the **New User Default** column, the permissions given to new users by default are indicated with a checkmark.

You can change the default global permissions to meet your needs.

#### Global Permissions for Users with External Authentication

When a user logs into Rancher using an external authentication provider for the first time, they are automatically assigned the **New User Default** global permissions. By default, Rancher assigns the **Standard User** permission for new users.

To see the default permissions for new users,

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. The **Role Templates** page has tabs for roles grouped by scope. Each table lists the roles in that scope. In the **New User Default** column on each page, the permissions given to new users by default are indicated with a checkmark.

You can change the default permissions to meet your needs.

Permissions can be assigned to an individual user.

You can assign a role to everyone in the group at the same time if the external authentication provider supports groups.

#### Custom Global Permissions

Using custom permissions is convenient for providing users with narrow or specialized access to Rancher.

When a user from an external authentication source signs into Rancher for the first time, they're automatically assigned a set of global permissions (hereafter, permissions). By default, after a user logs in for the first time, they are created as a user and assigned the default `user` permission. The standard `user` permission allows users to login and create clusters.

However, in some organizations, these permissions may extend too much access. Rather than assigning users the default global permissions of `Administrator` or `Standard User`, you can assign them a more restrictive set of custom global permissions.

The default roles, Administrator and Standard User, each come with multiple global permissions built into them. The Administrator role includes all global permissions, while the default user role includes three global permissions: Create Clusters, Use Catalog Templates, and User Base, which is equivalent to the minimum permission to log in to Rancher. In other words, the custom global permissions are modularized so that if you want to change the default user role permissions, you can choose which subset of global permissions are included in the new default user role.

Administrators can enforce custom global permissions in multiple ways:

- Creating custom global roles.
- Changing the default permissions for new users.
- Configuring global permissions for individual users.
- Configuring global permissions for groups.

#### Combining Built-in GlobalRoles

Rancher provides several GlobalRoles which grant granular permissions for certain common use cases. The following table lists each built-in global permission and whether it is included in the default global permissions, `Administrator`, `Standard User` and `User-Base`.

| Custom Global Permission | Administrator | Standard User | User-Base |
|---|---|---|---|
| Create Clusters | ⬔ | ⬔ | |
| Create RKE Templates | ⬔ | ⬔ | |
| Manage Authentication | ⬔ | | |
| Manage Catalogs | ⬔ | | |
| Manage Cluster Drivers | ⬔ | | |
| Manage Node Drivers | ⬔ | | |
| Manage PodSecurityPolicy Templates | ⬔ | | |
| Manage Roles | ⬔ | | |
| Manage Settings | ⬔ | | |
| Manage Users | ⬔ | | |
| Use Catalog Templates | ⬔ | ⬔ | |
| User-Base (Basic log-in access) | ⬔ | ⬔ | |

For details on which Kubernetes resources correspond to each global permission,

1. In the upper left corner, click ⬔ **> Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. If you click the name of an individual role, a table shows all of the operations and resources that are permitted by the role.



- Each permission listed above is comprised of multiple individual permissions not listed in the Rancher UI. For a full list of these permissions and the rules they are comprised of, access through the API at `/v3/globalRoles`.
- When viewing the resources associated with default roles created by Rancher, if there are multiple Kubernetes API resources on one line item, the resource will have `(Custom)` appended to it. These are not custom resources but just an indication that there are multiple Kubernetes API resources as one resource.



The built-in GlobalRole `Manage Users` allows users to create, modify and delete other users within the Rancher environment. While this permission may be necessary for administrative workflows in trusted environments, granting it to non-trusted or lower-privileged users, such as standard users, poses a serious security risk and may result in

privilege escalation.

#### Custom GlobalRoles

You can create custom GlobalRoles to satisfy use cases not directly addressed by built-in GlobalRoles.

Create custom GlobalRoles through the UI or through automation (such as the Rancher Kubernetes API). You can specify the same type of rules as the rules for upstream roles and clusterRoles.

See the list Global Resources for relevant resources.

#### Escalate and Bind verbs

When giving permissions on GlobalRoles, keep in mind that Rancher respects the `escalate` and `bind` verbs, in a similar fashion to Kubernetes [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#restrictions-on-role-creation-or-update].

Both of these verbs, which are given on the GlobalRoles resource, can grant users the permission to bypass Rancher's privilege escalation checks. This potentially allows users to become admins. Since this represents a serious security risk, `bind` and `escalate` should be distributed to users with great caution.

The `escalate` verb allows users to change a GlobalRole and add any permission, even if the users doesn't have the permissions in the current GlobalRole or the new version of the GlobalRole.

The `bind` verb allows users to create a GlobalRoleBinding to the specified GlobalRole, even if they do not have the permissions in the GlobalRole.



The wildcard verb `*` also includes the `bind` and `escalate` verbs. This means that giving `*` on GlobalRoles to a user also gives them both `escalate` and `bind`.

#### Custom GlobalRole Examples

To grant permission to escalate only the `test-gr` GlobalRole:

```
rules:
- apiGroups:
  - 'management.cattle.io'
  resources:
  - 'globalroles'
  resourceNames:
  - 'test-gr'
  verbs:
  - 'escalate'
```

To grant permission to escalate all GlobalRoles:

```
rules:
- apiGroups:
```

```
  - 'management.cattle.io'
  resources:
  - 'globalroles'
  verbs:
  - 'escalate'
```

To grant permission to create bindings (which bypass escalation checks) to only the `test-gr` GlobalRole:

```
rules:
- apiGroups:
  - 'management.cattle.io'
  resources:
  - 'globalroles'
  resourceNames:
  - 'test-gr'
  verbs:
  - 'bind'
- apiGroups:
  - 'management.cattle.io'
  resources:
  - 'globalrolebindings'
  verbs:
  - 'create'
```

Granting `*` permissions (which includes both `escalate` and `bind`):

```
rules:
- apiGroups:
  - 'management.cattle.io'
  resources:
  - 'globalroles'
  verbs:
  - '*'
```

#### GlobalRole Permissions on Downstream Clusters

GlobalRoles can grant one or more RoleTemplates on every downstream cluster through the `inheritedClusterRoles` field. Values in this field must refer to a RoleTemplate which exists and has a `context` of Cluster.

With this field, users gain the specified permissions on all current or future downstream clusters. For example, consider the following GlobalRole:

```
apiVersion: management.cattle.io/v3
```

```
kind: GlobalRole
displayName: All Downstream Owner
metadata:
  name: all-downstream-owner
inheritedClusterRoles:
- cluster-owner
```

Any user with this permission will be a cluster-owner on all downstream clusters. If a new cluster is added, regardless of type, the user will be an owner on that cluster as well.



Using this field on default GlobalRoles may result in users gaining excessive permissions.

#### GlobalRole Permissions in Specific Namespaces

GlobalRoles grant access to resources in specific namespaces using two fields: `namespacedRules` and `inheritedNamespacedRules`.

The `namespacedRules` field provides access to namespaces within the management cluster. For example, when assigned to a user, the following `GlobalRole` allows that user to read secrets in the `my-secrets-namespace` namespace on the management cluster:

```
apiVersion: management.cattle.io/v3
kind: GlobalRole
displayName: Secret Reader
metadata:
  name: secret-reader
namespacedRules:
  my-secrets-namespace:
    - apiGroups:
      - ""
      resources:
      - secrets
      verbs:
      - "get"
      - "list"
      - "watch"
```

The `inheritedNamespacedRules` field applies permissions to namespaces across all downstream clusters. The following example shows a `GlobalRole` that grants full access to the `cattle-monitoring-system` namespace on every cluster:

```
apiVersion: management.cattle.io/v3
kind: GlobalRole
displayName: Edit Monitoring User
metadata:
  name: monitoring-role
```

```
inheritedNamespacedRules:
  cattle-monitoring-system:
    - apiGroups:
      - "*"
      resources:
      - "*"
      verbs:
      - "*"
```

If a cluster does not have the `cattle-monitoring-system` namespace, Rancher grants no permissions. If you later create the namespace on that cluster, Rancher automatically adds permissions for users bound to this `GlobalRole`.

Both fields are maps where keys are namespaces and values are arrays of `PolicyRules`. This structure allows you to specify multiple namespaces and multiple rules per namespace.



These fields alone do not grant direct cluster access. You must also grant users permissions to access the cluster using `inheritedClusterRoles` or `ClusterRoleTemplateBindings`.

#### GlobalRole Permissions for Fleet Workspaces

GlobalRoles can grant access to Fleet workspaces in each downstream cluster with the field `inheritedFleetWorkspacePermissions`. This allows users to deploy Fleet resources in all workspaces except `fleet-local`. The field is made up of two parts:

- `resourceRules` which grants rules to Fleet resources
- `workspaceVerbs` which grants permissions to the cluster-wide `fleetworkspace` resources.

Example:



**Deprecation**: SUSE® Rancher Prime: Continuous Delivery will remove support for the `GitRepoRestriction` resource in a future release. To configure multi-tenant environments, use the `Policy` resource [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/reference/ref-policy.html] instead.

```
apiVersion: management.cattle.io/v3
kind: GlobalRole
metadata:
  name: fleet-permissions
inheritedClusterRoles:
   - cluster-owner
inheritedFleetWorkspacePermissions:
  resourceRules:
    - apiGroups:
        - fleet.cattle.io
      resources:
```

```
        - clusterregistrationtokens
        - gitreporestrictions
        - clusterregistrations
        - clusters
        - gitrepos
        - bundles
        - clustergroups
      verbs:
        - "*"
  workspaceVerbs:
    - get
    - list
    - update
    - create
    - delete
    - patch
    - fleetaddcluster
```

#### Configuring Default Global Permissions

If you want to restrict the default permissions for new users, you can remove the `user` permission as default role and then assign multiple individual permissions as default instead. Conversely, you can also add administrative permissions on top of a set of other standard permissions.



Default roles are only assigned to users added from an external authentication provider. For local users, you must explicitly assign global permissions when adding a user to Rancher. You can customize these global permissions when adding the user.

To change the default global permissions that are assigned to external users upon their first log in, follow these steps:

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Role Templates**. On the **Role Templates** page, make sure the **Global** tab is selected.
3. Find the permissions set that you want to add or remove as a default. Then edit the permission by selecting ☰ **> Edit Config**.
4. If you want to add the permission as a default, Select **Yes: Default role for new users** and then click **Save**. If you want to remove a default permission, edit the permission and select **No**.

**Result:** The default global permissions are configured based on your changes. Permissions assigned to new users display a check in the **New User Default** column.

#### Configuring Global Permissions for Individual Users

To configure permission for a user,

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Users**.

3. Go to the user whose access level you want to change and click ⋮ **> Edit Config**.
4. In the **Global Permissions** and **Built-in** sections, check the boxes for each permission you want the user to have. If you have created roles from the **Role Templates** page, they will appear in the **Custom** section and you can choose from them as well.
5. Click **Save**.

**Result:** The user's global permissions have been updated.

#### Configuring Global Permissions for Groups

If you have a group of individuals that need the same level of access in Rancher, it can save time to assign permissions to the entire group at once, so that the users in the group have the appropriate level of access the first time they sign into Rancher.

After you assign a custom global role to a group, the custom global role will be assigned to a user in the group when they log in to Rancher.

For existing users, the new permissions will take effect when the users log out of Rancher and back in again, or when an administrator refreshes the group memberships.

For new users, the new permissions take effect when the users log in to Rancher for the first time. New users from this group will receive the permissions from the custom global role in addition to the **New User Default** global permissions. By default, the **New User Default** permissions are equivalent to the **Standard User** global role, but the default permissions can be configured.

If a user is removed from the external authentication provider group, they would lose their permissions from the custom global role that was assigned to the group. They would continue to have any remaining roles that were assigned to them, which would typically include the roles marked as **New User Default**. Rancher will remove the permissions that are associated with the group when the user logs out, or when an administrator refreshes group memberships, whichever comes first.



*Prerequisites:*

You can only assign a global role to a group if:

- You have set up an external authentication provider
- The external authentication provider supports user groups
- You have already set up at least one user group with the authentication provider

To assign a custom global role to a group, follow these steps:

1. In the upper left corner, click ⋮ **> Users & Authentication**.
2. In the left navigation bar, click **Groups**.
3. Go to the group you want to assign a custom global role to and click ⋮ **> Edit Config**.
4. In the **Global Permissions, Custom,** and/or **Built-in** sections, select the permissions that the group should have.
5. Click **Create**.

**Result:** The custom global role will take effect when the users in the group log into Rancher.

#### Refreshing Group Memberships

When an administrator updates the global permissions for a group, the changes take effect for individual group members after they log out of Rancher and log in again.

To make the changes take effect immediately, an administrator or cluster owner can refresh group memberships.

An administrator might also want to refresh group memberships if a user is removed from a group in the external authentication service. In that case, the refresh makes Rancher aware that the user was removed from the group.

To refresh group memberships,

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Users**.
3. Click **Refresh Group Memberships**.

**Result:** Any changes to the group members' permissions will take effect.

#### Global Resources

Global Resources are non-namespaced resources provided by Rancher. Users who are cluster-owners/project-owners do not have access to several of these by default. For example principals and roletemplates.

The resources are documented here to support administrators creating and/or modifying custom sets of permissions with finding the minimum set of permissions needed for a particular task in the dashboard.

As an example, to use the cluster/project permissions pages of the dashboard requires permissions on principals to search/display a readable name for users, and on roleTemplates, to see a list of usable roleTemplates as well as to display a readable name for the role.

Please see the list below to determine what permissions you may need when creating a least-privilege user.

| Group | Resource | Purpose |
|---|---|---|
| auditlog.cattle.io | auditpolicies | Specification of log filers, redactions, verbosity |
| catalog.cattle.io | clusterrepos | Helm chart repository location and credentials |
| ext.cattle.io | groupmembershiprefreshrequests | Creation triggers refresh of group membership for specific or all users |
| ext.cattle.io | kubeconfigs | Kubeconfig file for access to remote clusters |
| ext.cattle.io | passwordchangerequests | Creation triggers update of the referenced User's password |
| ext.cattle.io | selfusers | Get delivers User making the request |
| ext.cattle.io | tokens | Raw API key. See Kubeconfig for wrapped API keys |
| management.cattle.io | authconfigs | Configuration of external auth service providers |

| Group | Resource | Purpose |
|---|---|---|
| management.cattle.io | clusters | Remote cluster management |
| management.cattle.io | features | Feature controlling rancher behaviour |
| management.cattle.io | globalrolebindings | Binding of user/group to a global role |
| management.cattle.io | globalroles | Custom role for global permissions (applied local and remote) |
| management.cattle.io | nodedrivers | Configuration of driver to provision clusters with a cloud service provider |
| management.cattle.io | roletemplates | Template for custom roles managing project- or cluster-specific permissions |
| management.cattle.io | settings | Setting controlling Rancher behaviour |
| management.cattle.io | tokens | Raw API key, old style. |
| management.cattle.io | userattributes | Additional information about a managed User |
| management.cattle.io | users | User known to and managed by Rancher |
| telemetry.cattle.io | secretrequests | Request creation of a secret with arbitrary name, in any namespace |

#### Cluster and Project Roles

Cluster and project roles define user authorization inside a cluster or project.

To manage these roles,

1. Click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates** and go to the **Cluster** or **Project/Namespaces** tab.

#### Membership and Role Assignment

The projects and clusters accessible to non-administrative users is determined by *membership*. Membership is a list of users who have access to a specific cluster or project based on the roles they were assigned in that cluster or project. Each cluster and project includes a tab that a user with the appropriate permissions can use to manage membership.

When you create a cluster or project, Rancher automatically assigns you as the `Owner` for it. Users assigned the `Owner` role can assign other users roles in the cluster or project.



Non-administrative users cannot access any existing projects/clusters by default. A user with appropriate permissions (typically the owner) must explicitly assign the project and cluster membership.

#### Cluster Roles

*Cluster roles* are roles that you can assign to users, granting them access to a cluster. There are two primary cluster roles: `Owner` and `Member`.

- **Cluster Owner:**

  These users have full control over the cluster and all resources in it.

- **Cluster Member:**

  These users can view most cluster level resources and create new projects.



When a Cluster Member creates a project, the user is automatically assigned Project Owner privileges. This grants them comprehensive control over the project and its associated resources, including permissions to deploy workloads. Without enforced Pod Security Standards (PSS) and Pod Security Admission (PSA), a Cluster Member is able to execute privileged containers in the cluster.

#### Custom Cluster Roles

Rancher lets you assign *custom cluster roles* to a standard user instead of the typical `Owner` or `Member` roles. These roles can be either a built-in custom cluster role or one defined by a Rancher administrator. They are convenient for defining narrow or specialized access for a standard user within a cluster. See the table below for a list of built-in custom cluster roles.

#### Cluster Role Reference

The following table lists each built-in custom cluster role available and whether that level of access is included in the default cluster-level permissions, `Cluster Owner` and `Cluster Member`.

| Built-in Cluster Role | Owner | Member |
|---|---|---|
| Create Projects | ⬔ | ⬔ |
| Manage Cluster Backups | ⬔ | |
| Manage Cluster Catalogs | ⬔ | |
| Manage Cluster Members | ⬔ | |
| Manage Nodes (see table below) | ⬔ | |
| Manage Storage | ⬔ | |
| View All Projects | ⬔ | |
| View Cluster Catalogs | ⬔ | ⬔ |
| View Cluster Members | ⬔ | ⬔ |
| View Nodes | ⬔ | ⬔ |

#### Manage Nodes Permissions

The following table lists the permissions available for the `Manage Nodes` role in RKE and RKE2.

| Manage Nodes Permissions | RKE | RKE2 |
|---|---|---|
| SSH Access | ⬔ | ⬔ |
| Delete Nodes | ⬔ | ⬔ |
| Scale Clusters Up and Down | ⬔ | * |

**In RKE2, you must have permission to edit a cluster to be able to scale clusters up and down.**

For details on how each cluster role can access Kubernetes resources, you can look them up in the Rancher UI:

1. In the upper left corner, click ⬔ **> Users & Authentication**.

2. In the left navigation bar, click **Role Templates**.
3. Click the **Cluster** tab.
4. Click the name of an individual role. The table shows all of the operations and resources that are permitted by the role.



When viewing the resources associated with default roles created by Rancher, if there are multiple Kubernetes API resources on one line item, the resource will have `(Custom)` appended to it. These are not custom resources but just an indication that there are multiple Kubernetes API resources as one resource.

#### Giving a Custom Cluster Role to a Cluster Member

After an administrator sets up a custom cluster role, cluster owners and admins can then assign those roles to cluster members.

To assign a custom role to a new cluster member, you can use the Rancher UI. To modify the permissions of an existing member, you will need to use the Rancher API view.

To assign the role to a new cluster member,

1. Click ☰ **> Cluster Management**.
2. Go to the cluster where you want to assign a role to a member and click **Explore**.
3. Click menu:Cluster[Cluster Members].
4. Click **Add**.
5. In the **Cluster Permissions** section, choose the custom cluster role that should be assigned to the member.
6. Click **Create**.

**Result:** The member has the assigned role.

To assign any custom role to an existing cluster member,

1. Click ☰ **> Users & Authentication**.
2. Go to the member you want to give the role to. Click the ☰ **> Edit Config**.
3. If you have added custom roles, they will show in the **Custom** section. Choose the role you want to assign to the member.
4. Click **Save**.

**Result:** The member has the assigned role.

#### Project Roles

*Project roles* are roles that can be used to grant users access to a project. There are three primary project roles: `Owner`, `Member`, and `Read Only`.

- **Project Owner:**

  These users have full control over the project and all resources in it.

- **Project Member:**

  These users can manage project-scoped resources like namespaces and workloads, but cannot manage other project members.



  By default, the Rancher role of `project-member` inherits from the `Kubernetes-edit` role, and the `project-owner` role inherits from the `Kubernetes-admin` role. As such, both `project-member` and `project-owner` roles will allow for namespace management, including the ability to create and delete namespaces.

- **Read Only:**

  These users can view everything in the project but cannot create, update, or delete anything.



  Users assigned the `Owner` or `Member` role for a project automatically inherit the `namespace creation` role. However, this role is a Kubernetes ClusterRole [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-and-clusterrole], meaning its scope extends to all projects in the cluster. Therefore, users explicitly assigned the `owner` or `member` role for a project can create namespaces in other projects they're assigned to, even with only the `Read Only` role assigned.

#### Custom Project Roles

Rancher lets you assign *custom project roles* to a standard user instead of the typical `Owner`, `Member`, or `Read Only` roles. These roles can be either a built-in custom project role or one defined by a Rancher administrator. They are convenient for defining narrow or specialized access for a standard user within a project. See the table below for a list of built-in custom project roles.



  *Privilege escalation via wild cards in custom project roles*

  Using wild cards (`*`) in a custom project role can grant unintended cluster-wide administrative access. When rules match non-namespaced resources, such as nodes, persistent volumes, storage classes, API services, or cluster repositories, SUSE Rancher Prime promotes those rules to a `ClusterRole`. A `ProjectRoleTemplateBinding` then grants access outside the project, and combining this with `*` for verbs grants full read and write permissions across the cluster.

  To prevent unintended privilege escalation, **avoid wild cards in custom project roles**. Specify only the required API groups, resources, and verbs, and review the effective permissions before assigning the role.

#### Project Role Reference

The following table lists each built-in custom project role available in Rancher and whether it is also granted by the `Owner`, `Member`, or `Read Only` role.

| Built-in Project Role | Owner | Member | Read Only |
|---|---|---|---|
| Manage Project Members | ⬕ | | |
| Create Namespaces | ⬕ | ⬕ | |
| Manage Config Maps | ⬕ | ⬕ | |
| Manage Ingress | ⬕ | ⬕ | |

| Built-in Project Role | Owner | Member | Read Only |
|---|---|---|---|
| Manage Project Catalogs | ⬕ | | |
| Manage Secrets | ⬕ | ⬕ | |
| Manage Service Accounts | ⬕ | ⬕ | |
| Manage Services | ⬕ | ⬕ | |
| Manage Volumes | ⬕ | ⬕ | |
| Manage Workloads | ⬕ | ⬕ | |
| View Secrets | ⬕ | ⬕ | |
| View Config Maps | ⬕ | ⬕ | ⬕ |
| View Ingress | ⬕ | ⬕ | ⬕ |
| View Project Members | ⬕ | ⬕ | ⬕ |
| View Project Catalogs | ⬕ | ⬕ | ⬕ |
| View Service Accounts | ⬕ | ⬕ | ⬕ |
| View Services | ⬕ | ⬕ | ⬕ |
| View Volumes | ⬕ | ⬕ | ⬕ |
| View Workloads | ⬕ | ⬕ | ⬕ |



- Each project role listed above, including `Owner`, `Member`, and `Read Only`, is comprised of multiple rules granting access to various resources. You can view the roles and their rules on the Global > Security > Roles page.
- When viewing the resources associated with default roles created by Rancher, if there are multiple Kubernetes API resources on one line item, the resource will have `(Custom)` appended to it. These are not custom resources but just an indication that there are multiple Kubernetes API resources as one resource.
- The `Manage Project Members` role allows the project owner to manage any members of the project **and** grant them any project scoped role regardless of their access to the project resources. Be cautious when assigning this role out individually.

#### Defining Custom Roles

As previously mentioned, custom roles can be defined for use at the cluster or project level. The context field defines whether the role will appear on the cluster member page, project member page, or both.

When defining a custom role, you can grant access to specific resources or specify roles from which the custom role should inherit. A custom role can be made up of a combination of specific grants and inherited roles. All grants are additive. This means that defining a narrower grant for a specific resource **will not** override a broader grant defined in a role that the custom role is inheriting from.

#### Default Cluster and Project Roles

By default, when a standard user creates a new cluster or project, they are automatically assigned an ownership role: either cluster owner or project owner. However, in some organizations, these roles may overextend administrative access. In this use case, you can change the default role to something more restrictive, such as a set of individual roles or a custom role.

There are two methods for changing default cluster/project roles:

- **Assign Custom Roles**: Create a custom role for either your cluster or project, and then set the custom role as default.
- **Assign Individual Roles**: Configure multiple cluster/project roles as default for assignment to the creating user.

  For example, instead of assigning a role that inherits other roles (such as `cluster owner`), you can choose a mix of individual roles (such as `manage nodes` and `manage storage`).



- Although you can lock a default role, the system still assigns the role to users who create a cluster/project.
- Only users that create clusters/projects inherit their roles. Users added to the cluster/project membership afterward must be explicitly assigned their roles.

#### Configuring Default Roles for Cluster and Project Creators

You can change the cluster or project role(s) that are automatically assigned to the creating user.

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. Click the **Cluster** or **Project/Namespaces** tab.
4. Find the custom or individual role that you want to use as default. Then edit the role by selecting ☰ > **Edit Config**.
5. In the **Cluster Creator Default** or **Project Creator Default** section, enable the role as the default.
6. Click **Save**.

**Result:** The default roles are configured based on your changes. Roles assigned to cluster/project creators display a check in the **Cluster/Project Creator Default** column.

If you want to remove a default role, edit the permission and select **No** from the default roles option.

#### Cluster Membership Revocation Behavior

When you revoke the cluster membership for a standard user that's explicitly assigned membership to both the cluster *and* a project within the cluster, that standard user loses their cluster roles but retains their project roles. In other words, although you have revoked the user's permissions to access the cluster and its nodes, the standard user can still:

- Access the projects they hold membership in.
- Exercise any individual project roles they are assigned.

If you want to completely revoke a user's access within a cluster, revoke both their cluster and project memberships.

#### External `RoleTemplate` Behavior

In Rancher v2.9.0 and later, external `RoleTemplate` objects can only be created if the backing `ClusterRole` exists in the local cluster or the `ExternalRules` is set in your configuration.

For context, the backing `ClusterRole` holds cluster rules and privileges, and shares the same `metadata.name` used in the `RoleTemplate` in your respective cluster referenced by the

`ClusterRoleTemplateBinding/ProjectRoleTemplateBinding`. Additionally, note that `escalate` permissions on `RoleTemplates` are required to create external `RoleTemplates` with `ExternalRules`.

#### Custom Roles

Within Rancher, *roles* determine what actions a user can make within a cluster or project.

Note that *roles* are different from *permissions*, which determine what clusters and projects you can access.



It is possible for a custom role to enable privilege escalation. For details, see this section.

#### Prerequisites

To complete the tasks on this page, one of the following permissions are required:

- Administrator Global Permissions.
- Custom Global Permissions with the Manage Roles role assigned.

#### Creating A Custom Role

While Rancher comes out-of-the-box with a set of default user roles, you can also create default custom roles to provide users with very specific permissions within Rancher.

The steps to add custom roles differ depending on the version of Rancher.

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. Select a tab to determine the scope of the role you're adding. The tabs are:
   - **Global:** The role is valid for allowing members to manage global scoped resources.
   - **Cluster:** The role is valid for assignment when adding/managing members to clusters.
   - **Project/Namespaces:** The role is valid for assignment when adding/managing members to projects or namespaces.
4. Click **Create Global Role, Create Cluster Role** or **Create Project/Namespaces Role,** depending on the scope.
5. Enter a **Name** for the role.
6. Optional: Choose the **Cluster/Project Creator Default** option to assign this role to a user when they create a new cluster or project. Using this feature, you can expand or restrict the default roles for cluster/project creators.

   > Out of the box, the Cluster Creator Default and the Project Creator Default roles are `Cluster Owner` and `Project Owner` respectively.

7. Use the **Grant Resources** options to assign individual Kubernetes API endpoints [https://kubernetes.io/docs/reference/] to the role.

   > When viewing the resources associated with default roles created by Rancher, if there are multiple Kubernetes API resources on one line item, the resource will have `(Custom)` appended to it. These are not custom resources but just an

> indication that there are multiple Kubernetes API resources as one resource.

> The Resource text field provides a method to search for pre-defined Kubernetes API resources, or enter a custom resource name for the grant. The pre-defined or `(Custom)` resource must be selected from the dropdown, after entering a resource name into this field.

You can also choose the individual cURL methods (`Create`, `Delete`, `Get`, etc.) available for use with each endpoint you assign.

8. Use the **Inherit from** options to assign individual Rancher roles to your custom roles. Note: When a custom role inherits from a parent role, the parent role cannot be deleted until the child role is deleted.
9. Click **Create**.

#### Creating a Custom Role that Inherits from Another Role

If you have a group of individuals that need the same level of access in Rancher, it can save time to create a custom role in which all of the rules from another role, such as the administrator role, are copied into a new role. This allows you to only configure the variations between the existing role and the new role.

The custom role can then be assigned to a user or group so that the role takes effect the first time the user or users sign into Rancher.

To create a custom role based on an existing role,

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. Click the **Cluster** or **Project/Namespaces** tab. Click **Create Cluster Role** or **Create Project/Namespaces Role** depending on the scope. Note: Only cluster roles and project/namespace roles can inherit from another role.
4. Enter a name for the role.
5. In the **Inherit From** tab, select the role(s) that the custom role will inherit permissions from.
6. In the **Grant Resources** tab, select the Kubernetes resource operations that will be enabled for users with the custom role.

> The Resource text field provides a method to search for pre-defined Kubernetes API resources, or enter a custom resource name for the grant. The pre-defined or `(Custom)` resource must be selected from the dropdown, after entering a resource name into this field.

7. Optional: Assign the role as default.
8. Click **Create**.

#### Deleting a Custom Role

When deleting a custom role, all global role bindings with this custom role are deleted.

If a user is only assigned one custom role, and the role is deleted, the user would lose access to Rancher. For the user to regain access, an administrator would need to edit the user and apply new global permissions.

Custom roles can be deleted, but built-in roles cannot be deleted.

To delete a custom role,

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. Go to the custom global role that should be deleted and click ☰ **(...) > Delete**.
4. Click **Delete**.

#### Assigning a Custom Role to a Group

If you have a group of individuals that need the same level of access in Rancher, it can save time to create a custom role. When the role is assigned to a group, the users in the group have the appropriate level of access the first time they sign into Rancher.

When a user in the group logs in, they get the built-in Standard User global role by default. They will also get the permissions assigned to their groups.

If a user is removed from the external authentication provider group, they would lose their permissions from the custom role that was assigned to the group. They would continue to have their individual Standard User role.



*Prerequisites:*

You can only assign a global role to a group if:

- You have set up an external authentication provider
- The external authentication provider supports user groups
- You have already set up at least one user group with the authentication provider

To assign a custom role to a group, follow these steps:

1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation bar, click **Groups**.
3. Go to the existing group that will be assigned the custom role and click ☰ > **Edit Config**.
4. If you have created roles, they will show in the **Custom** section. Choose any custom role that will be assigned to the group.
5. Optional: In the **Global Permissions** or **Built-in** sections, select any additional permissions that the group should have.
6. Click **Save.**.

**Result:** The custom role will take effect when the users in the group log into Rancher.

#### Privilege Escalation

The `Configure Catalogs` custom permission is powerful and should be used with caution. When an admin assigns the `Configure Catalogs` permission to a standard user, it could result in privilege

escalation in which the user could give themselves admin access to Rancher provisioned clusters. Anyone with this permission should be considered equivalent to an admin.

#### Locked Roles

You can set roles to a status of `locked`. Locking roles prevent them from being assigned to users in the future.

Locked roles:

- Cannot be assigned to users that don't already have it assigned.
- Are not listed in the **Member Roles** drop-down when you are adding a user to a cluster or project.
- Do not affect users assigned the role before you lock the role. These users retain access that the role provides.

  **Example:** let's say your organization creates an internal policy that users assigned to a cluster are prohibited from creating new projects. It's your job to enforce this policy.

  To enforce it, before you add new users to the cluster, you should lock the following roles: `Cluster Owner`, `Cluster Member`, and `Create Projects`. Then you could create a new custom role that includes the same permissions as a **Cluster Member**, except the ability to create projects. Then, you use this new custom role when adding users to a cluster.

Roles can be locked by the following users:

- Any user assigned the `Administrator` global permission.
- Any user assigned the `Custom Users` permission, along with the `Manage Roles` role.

#### Locking/Unlocking Roles

If you want to prevent a role from being assigned to users, you can set it to a status of `locked`.

You can lock roles in two contexts:

- When you're adding a custom role.
- When you editing an existing role (see below).

Cluster roles and project/namespace roles can be locked, but global roles cannot.

1. In the upper left corner, click ☰ **> Users & Authentication**.
2. In the left navigation bar, click **Role Templates**.
3. Go to the **Cluster** tab or the **Project/Namespaces** tab.
4. From the role that you want to lock (or unlock), select ☰ **> Edit Config**.
5. From the **Locked** option, choose the **Yes** or **No** radio button. Then click **Save**.

### Enabling User Retention

In Rancher v2.8.5 and later, you can enable user retention to automatically disable or delete inactive user accounts after a configurable time period.

The user retention feature is off by default.

#### Enabling User Retention with kubectl

To enable user retention, you must set `user-retention-cron`. You must also set at least one of `disable-inactive-user-after` or `delete-inactive-user-after`. You can use `kubectl edit setting <name-of-setting>` to open your editor of choice and set these values.

#### Configuring Rancher to Delete Users, Disable Users, or Combine Operations

Rancher uses two global user retention settings to determine if and when users are disabled or deleted after a certain period of inactivity. Disabled accounts must be re-enabled before users can log in again. If an account is deleted without being disabled, users may be able to log in through external authentication and the deleted account will be recreated.

The global settings, `disable-inactive-user-after` and `delete-inactive-user-after`, do not block one another from running.

For example, you can set both operations to run. If you give `disable-inactive-user-after` a shorter duration than `delete-inactive-user-after`, the user retention process disables inactive accounts before deleting them.

You can also edit some user retention settings on a specific user's `UserAttribute`. Setting these values overrides the global settings. See User-specific User Retention Overrides for more details.

#### Required User Retention Settings

The following are global settings:

- `user-retention-cron`: Describes how often the user retention process runs. The value is a cron expression (for example, `0 * * * *` for every hour).
- `disable-inactive-user-after`: The amount of time that a user account can be inactive before the process disables an account. Disabling an account forces the user to request that an administrator re-enable the account before they can log in to use it. Values are expressed in time.Duration units [https://pkg.go.dev/time#ParseDuration] (for example, `720h` for 720 hours or 30 days). The value must be greater than `auth-user-session-ttl-minutes`, which is `16h` by default. If the value is not set, set to the empty string, or is equal to 0, the process does not disable any inactive accounts.
- `delete-inactive-user-after`: The amount of time that a user account can be inactive before the process deletes the account. Values are expressed in time.Duration units (for example, `720h` for 720 hours or 30 days). The value must be greater than `auth-user-session-ttl-minutes`, which is `16h` by default. The value should be greater than `336h` (14 days), otherwise it is rejected by the Rancher webhook. If you need the value to be lower than 14 days, you can bypass the webhook. If the value is not set, set to the empty string, or is equal to 0, the process does not delete any inactive accounts.

#### Optional User Retention Settings

The following are global settings:

- `user-retention-dry-run`: If set to `true`, the user retention process runs without actually deleting or disabling any user accounts. This can help test user retention behavior before allowing the process to disable or delete user accounts in a production environment.
- `user-last-login-default`: If a user does not have `UserAttribute.LastLogin` set on their account, this setting is used instead. The value is expressed as an RFC 3339 date-time [https://datatracker.ietf.org/doc/html/rfc3339#section-5.6] truncated to the last second; for example, `2023-`

`03-01T00:00:00Z`. If the value is set to the empty string or is equal to 0, this setting is not used.

#### User-specific User Retention Overrides

The following are user-specific overrides to the global settings for special cases. These settings are applied by editing the `UserAttribute` associated with a given account:

```
kubectl edit userattribute <user-name>
```

- `disableAfter`: The user-specific override for `disable-inactive-user-after`. The value is expressed in time.Duration units [https://pkg.go.dev/time#ParseDuration] and truncated to the second. If the value is set to `0s` then the account won't be subject to disabling.
- `deleteAfter`: The user-specific override for `delete-inactive-user-after`. The value is expressed in time.Duration units [https://pkg.go.dev/time#ParseDuration] and truncated to the second. If the value is set to `0s` then the account won't be subject to deletion.

#### Viewing User Retention Settings in the Rancher UI

You can see which user retention settings are applied to which users.

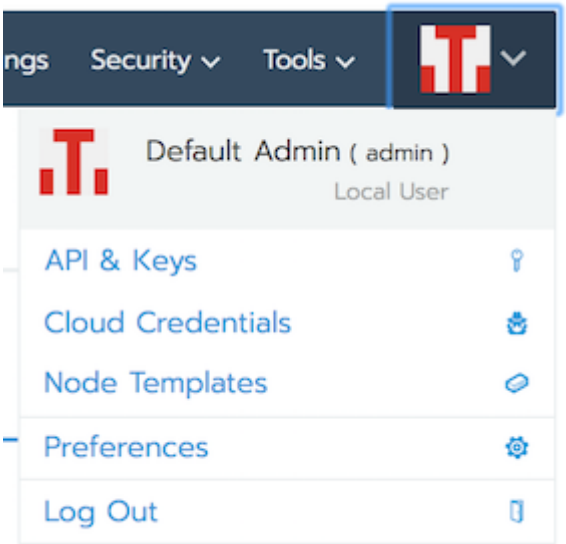
1. In the upper left corner, click ☰ > **Users & Authentication**.
2. In the left navigation menu, select **Users**.

The **Disable After** and **Delete After** columns for each user account indicate how long the account can be inactive before it is disabled or deleted from Rancher. There is also a **Last Login** column roughly indicating when the account was last active.

The same information is available if you click a user's name in the **Users** table and select the **Detail** tab.

## User Settings

Within Rancher, each user has a number of settings associated with their login: personal preferences, API keys, etc. You can configure these settings by choosing from the **User Settings** menu. You can open this menu by clicking your avatar, located within the main menu.



The available user settings are:

- API & Keys: If you want to interact with Rancher programmatically, you need an API key. Follow the directions in this section to obtain a key.
- Cloud Credentials: Manage cloud credentials used by machine pools to provision nodes for clusters.
- Preferences: Sets superficial preferences for the Rancher UI.
- Log Out: Ends your user session.

### API Keys



*Deprecation of v3 API tokens*

Rancher v2.13 introduced a new token resource in the `ext.cattle.io` API group to serve as the public API for tokens, which was only accessible through `kubectl`.

Starting with Rancher v2.14.1:

- Legacy v3 API tokens (`tokens.management.cattle.io`) are being phased out and scheduled for removal in a future release. If you currently rely on legacy tokens for automation or API access, transition to the new `tokens.ext.cattle.io` tokens to ensure continued compatibility.
- The Rancher dashboard provides basic support for `tokens.ext.cattle.io` (create, view or list, and delete).

#### API Keys and User Authentication

If you want to access your Rancher clusters, projects, or other objects using external applications, you can do so using the Rancher API. However, before your application can access the API, you must provide the app with a key used to authenticate with Rancher. You can obtain a key using the Rancher UI.

An API key is also required for using Rancher CLI.

API Keys are composed of four components:

- **Endpoint:** This is the IP address and path that other applications use to send requests to the Rancher API.
- **Access Key:** The token's username.
- **Secret Key:** The token's password. For applications that prompt you for two different strings for API authentication, you usually enter the two keys together.
- **Bearer Token:** The token username and password concatenated together. Use this string for applications that prompt you for one authentication string.



Users may opt to enable token hashing.

#### Creating an API Key

1. Select **User Avatar** › **Account & API Keys** from upper right corner.
2. Click **Create API Key**.
3. **Optional:** Enter a description for the API key and select an expiration period or a scope. We recommend

setting an expiration date.

The API key won't be valid after expiration. Shorter expiration periods are more secure.

Expiration period is bound by `v3/settings/auth-token-max-ttl-minutes`. If it exceeds the max-ttl, the API key is created with max-ttl as the expiration period.

A scope limits the API key so that it only works against the Kubernetes API of the specified cluster. If the cluster is configured with an Authorized Cluster Endpoint, you are able to use a scoped token directly against the cluster's API without proxying through the Rancher server. See Authorized Cluster Endpoints for more information.

4. Click **Create**.

   **Step Result:** Your API Key is created. Your API **Endpoint**, **Access Key**, **Secret Key**, and **Bearer Token** are displayed.

   Use the **Bearer Token** to authenticate with Rancher CLI.

5. Copy the information displayed to a secure location. This information is only displayed once, so if you lose your key, you'll have to make a new one.

#### What's Next?

- Enter your API key information into the application that sends requests to the Rancher API.
- Learn more about the Rancher endpoints and parameters by selecting **View in API** for an object in the Rancher UI.
- API keys are used for API calls and Rancher CLI.

#### Deleting API Keys

If you need to revoke an API key, delete it. You should delete API keys:

- That may have been compromised.
- That have expired.

To delete an API, select the stale key and click **Delete**.

### User Preferences

You can set preferences to personalize your Rancher experience. To change preference settings:

1. Click on your user avatar in the upper right corner.
2. Click **Preferences**.

#### Language

Choose your language for the Rancher UI. Options include:

- English
- ▯▯▯▯ (Simplified Chinese)

#### Theme

Choose your background color for the Rancher UI. If you choose **Auto**, the background color changes from light to dark at 6 PM, and then changes back at 6 AM.

#### Login Landing Page

Choose the page displayed after logging in. Options include:

- The home page.
- The area you last visited.
- A specific cluster of your choosing.

#### Display Settings

Choose how certain information is displayed:

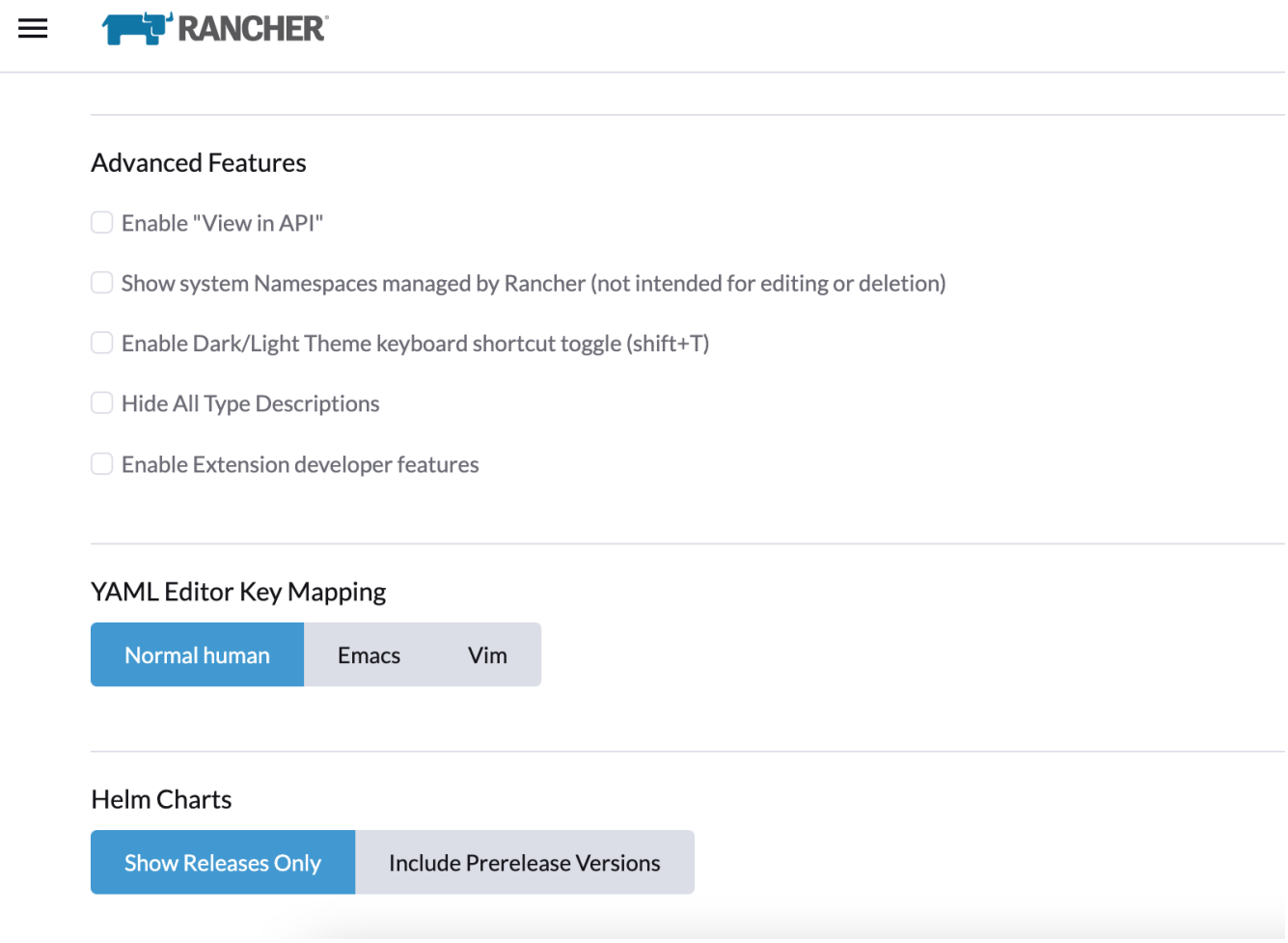
- Date format
- Time format
- Table rows per page
- Number of clusters to show in side menu

#### Confirmation Setting

Choose whether to ask for confirmation when scaling down node pools.

#### Advanced Features

- Enable "View in API".
- Show system Namespaces managed by Rancher (not intended for editing or deletion).
- Enable Dark/Light Theme keyboard shortcut toggle (shift+T).
- Hide All Type Descriptions.
- Enable Extension developer features.




#### YAML Editor Key Mapping

- Normal human (Sublime)
- Emacs
- Vim

#### Helm Charts

Choose whether to display released Helm charts only or to include prerelease versions as well. A version is considered to be a prerelease if it adheres to the specification [https://semver.org/#spec-item-9] defined by Semantic Versioning 2.0.0 [https://semver.org/]. For example, a Helm chart with a version of `0.1.3-dev.12ab4f` requires `Include Prerelease Versions` to be selected in order to be displayed.

### Managing Cloud Credentials

The creation or association of cloud credentials are part of the cluster creation process, the information below provides guidance on managing credentials in Rancher.

Cloud credentials are bound to their creator's user profile. They **cannot** be shared between non-admin users. However, admins can view and manage the cloud credentials of other users.

#### Creating a Cloud Credential

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.

3. Click **Create**.
4. Click a cloud credential type. The values of this dropdown is based on the `active` node drivers in Rancher.
5. Enter a name for the cloud credential.
6. Based on the selected cloud credential type, enter the required values to authenticate with the infrastructure provider.
7. Click **Create**.

**Result:** The cloud credential is created.

#### Updating a Cloud Credential

When access credentials are changed or compromised, updating a cloud credential allows you to rotate those credentials while keeping the same node template.

1. Click ☰ **> Cluster Management**.
2. Click **Cloud Credentials**.
3. Choose the cloud credential you want to edit and click the ☰ **> Edit Config**.
4. Update the credential information and click **Save**.

**Result:** The cloud credential is updated with the new access credentials.

#### Deleting a Cloud Credential

1. Click ☰ **> Cluster Management**.
2. Click **Cloud Credentials**.
3. You can either individually delete a cloud credential or bulk delete.
   - To individually delete one, choose the cloud credential you want to edit and click the ☰ **> Delete**.
   - To bulk delete cloud credentials, select one or more cloud credentials from the list. Click **Delete**.
4. Confirm that you want to delete these cloud credentials.

# Global Configuration

## Custom Branding

### Prerequisite

You will need to have at least cluster member permissions.

### Brand Configuration

To configure the brand settings,

1. Click ☰ **> Global settings**.
2. Click **Branding**.

#### Private Label Company Name

This option replaces "Rancher" with the value you provide in most places. Files that need to have Rancher in the name, such as `rancher-compose.yml`, won't be changed.

#### Support Links

Use a url address to send new "File an Issue" reports instead of sending users to the GitHub issues page. Optionally show Rancher community support links.

#### Logo

Upload light and dark logos to replace the Rancher logo in the top-level navigation header.

#### Primary Color

You can override the primary color used throughout the UI with a custom color of your choice.

#### Fixed Banners

Display a custom fixed banner in the header, footer, or both.

To configure banner settings,

1. Click ☰ **> Global settings**.
2. Click **Banners**.

### Custom Navigation Links

In this section, you'll learn how to configure the links in the left navigation bar of the **Cluster Dashboard**. To get to the cluster dashboard,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want custom navigation links and click **Explore**.

It can be useful to add a link for quick access to services installed on a cluster. For example, you could add a link to the Kiali UI for clusters with Istio installed, or you could add a link to the Grafana UI for clusters with Rancher monitoring installed.

The custom links don't affect who has access to each service.

Links can be created at the top level and multiple links can be grouped together.

#### Adding a Custom Navigation Link



*Prerequisite:*

You will need to have at least cluster member or project member permissions.

1. Click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you would like to add custom navigation links and click **Explore**.
3. In the top navigation menu, click ☰ **(Resource Search)**.

4. Type **Nav** and click **Nav Links**.
5. Click **Create from YAML**.
6. The simplest way to create a navigation link is to add these fields:

```
name: linkname
toURL: https://example.com
```

   For more details on setting up links, including optional fields, see Link Configuration.

7. Click **Create**.

#### Link Configuration

**`name`**

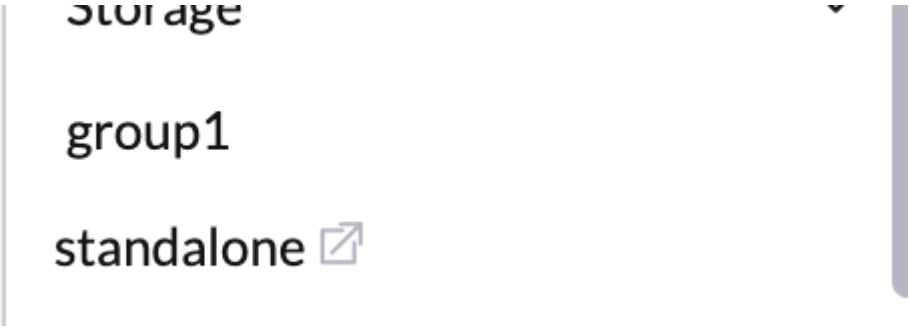
Display name for the link. Required.

**`group`**

Name of a group of links that expands when clicked.

Optional. If not provided, the link appears standalone.

Groups are displayed separately from standalone links, as shown below:




**`iconSrc`**

Icon source in base64 format.

Below is an example of the Grafana logo in base64 format:

```

lEQVR4Aey9d5xkZZn3zb/P+3mffZ9nDcDAzHTuqs49PZEhCBhBJcl0nO7pHKbD9PR07p5AWl
EEZM2ioph3dXGNqLgCxhVBRIYRmNy5cjqnTlWdqu/7ue5zTk+DiNIsCn4suOacrjrxvq/fle
/7PofX+ycDZNT/pIAUaUUmaRBKp8FMQ1L9qL6S4+VYUikIeuBrn+LppiuJNG/BvHYd7HbhbX
LBZ/dB8AkwFokCHsAH6Kb8IxTHJIyBHwgDMTBT6h7yWAmblL3sxyEjfxkYxDDQMa0nWV0vyE
2slwZ5JtUO8v5JUhgk0EkRw5TnUg9sgJm03lsezkyTIU0C2VrNJU2WsdtTPVQyAmYU4mGIhD
ESaULAcSG5RjwJEQ8YsxgEOZoOcSxjvXsyKM8nL24QJ4UmF5TnlL7IWLure/G/3lnn/PVu9S
rdaQVAp03/CCAZAYjNSLIVBrD/VMwSnsf4/B38ofWthFu3krhhPZmaLJZuyEY7vJPIV++AwE
nImJwJ66qTFcMLSswkGWKkiKqtMIICwB890zL/2lwoHGLYnCIMtMqP3Md5N7mMDRDTBkhKAU
```

```
RAqNBs3TOdspjWERgrmkYuJbQMEPUeKdAEIBrIuSIKEiZ+B0ShADz7W/Tjv8TMLOIjybxcJw
NJOUgAkjHQSFoAkedMWn2g7rXKV/9rnfZ3DRCRjgoN0ik2A0nDLgMkbYD3aU7dPcRT7Zfhad
mEVldA/IZlRN6TRahxO8f37CTyxU/B4pySvOlYHNM0sZhQRKgwTRrRWALOZc0lWksoY30n0l
PkuMBCPYCcKn/Ic632Iy/ivNvy+4k2EOAamDbJ/rIKFYEhx74AWPIWcpRhv8dyu5mQTEEkmU
YTjSsaMBWDiA9mjsF/foWTH76Z33zqEL6jD5IgRkTeR64valfOQQSL6My0Ap/c32qvlNJgq3
39v8Z5fzcAkcayBKrFkNIBCiDOD/Lj8jG2iZWOwvGHefxQNY+3beHk7grCDS7067JhRwn6dZ
X4d72L3zVei/6Vz0HYB/EQJINKawRTQXVP+UfYQEjdRphITBlTNIrFqBopIsqcsZnaQYww0i
v5yAlXkuJ+eRJREStJvrMFhnO8A5S09ewCDoMkCQUw67KxOMQMCBkpkok4JIIQnYGf3k/s7m
lO7N3Fw7VX8J2Ot3Pi/rvBXMLMJKxbx+UR5L4CEJ0IKYUbeV0xAUXDZVSrvZIGeHXPff0DRD
GE9PRZPhGec8jhBWcr30uXCWEG4Xf/wW+H38ajXVUcaS7GX+dGvyYX6jeQvL6MZOllzFx7Mc
8lXkPyM3eC/xlIz5LJzBIz/bbUtUyKZaksUtPUIS06wzK/LGluM6jwrVg9wkCvECDOe5llE2
kL5w2drdU+Ths5bSBbMacsVMtGtKDFug5+5Q00Iw2JFOhhmD0C3/goS6MlHKvfiqfxMo7t3M
Lv2i7F97nDoJ+BpG45IXLysgYRgIhvJPeRu4QVibZ7LX/+rgDiNLTT58IADjM4rPI8HyXpgY
c+yXODF3G0ZyPPtZSxUFeM/9p8MrUVJK4rIX5NMfEdmljauZlj7Vfj/exNcPoRSB2HxIICiH
I+Hb4U00mYSWkP4RAhHTKiUexggfCEPJDiUOepX/5W3tN5R9m3PpZZJ6bdWbJ+kWPkto5lJy
aVxbBnpYtcT35XwFDXt8Ee8cJ//wj9X6c40fF2TtZU4qspJV5bidawCU/HxWgfHoTwccuhl4
srE0saRnwQwwaIZQKa+BH6B0CcfnvVttIBK8jmFId5zjKEmA+WhJTIkeoYYxY+t5/FtmJ8ze
UEWzdypqmM07VFhGpK0W9wYVyXhl5dQLxnG/NdF/NE00V4PzgAv/0OLB5RbyWKQD2CML5wVl
KMbIlmCSgkapQEkcLCNPJ72mJAsdXl+Vb7cRh+mcnlQvKl0IomUfs2mOT28rwCaiErgiW+hX
WOaBSnzRSw4/Mw/wR87zN4xht55vqNzNxQQXj3VoyWzRjX5ZPcWUigrozozU0QeMbSNAnIyA
0EcaQRElN8EPWn0hoSDxSSRnntfl73GkTMAsvXsDnCYZAXMERc2dei2i0HVnWMdpro4etYuv
58orUujLatLLZsZKapkqXaYqINRZi7XWQ63ASacwh2lhPtvZwjdVs4M94ETz4M8ajFjI5TLl
sJLavwsu0GCA84JpX4uEAAVHBYGHalH3lVuZaQxXgvAIh86QDFDqu+ECDSIstS3AGWnCdmUn
wRjv4Y/XMHON5lOSerSwjv2kCqdRta8lZiDZWwqwjq8onWFKIfrIPQs5CIKl/ekJvZDyagFJ
JbWKFuiQFLePwfJtZq+//PnieNLc64lUuwTYoXMITwZowMgbSu3EIjE8NMiKMdhmd/zlzrRj
Jl2UQb3IQaKojUbiRUW8VSQynzjQV4mtfjbz6fSNN5hBvXsrjbzXzTZjzlV/Bs0/Xw0A9g7q
Ry3E3DRzARUJpKni0ZSljpEUkcCEBsQR3BYIEIC2mxw+WBV/dx3v950TP5UshpBxskqURG+c
vCjtImwqyyDYZ9pDPiMKfR4hHQY5aJdPIIPPgljrS/nZndW/E0lRJodBHY5SbYUEq4biOx2g
oil6+DliLCO/PwDL0HfvdD5X9JFNwXP+vjyL2UMJDnUs8kRpzkjv4BkNXl/l9wlmiOhHL4RI
baDrA0vs0UwifSMVEkuSWJsyTxRACMIKSi8Nj3WWyphLrl6PWFaPVlGDs3ou2swldXpgCy0L
oWT+t5RJreqEASaMpRDGLs2E6w+u2c7mkgcM/t8IdHID2PSZAQUaJmSrkAypgXXrClaTIj5k
ZcRXXiKlH4ygAibXA2Yme3wUqAJDMWWDJgGEmiWgzDFLlhCRcjHkWP+kgaPkgHQUyqIz8l+f
HDzLa/i0DdVrTaUmLlLsINBTZIignXVRCpK8W3cx3Jdjehnfl4970bHnsA0rpi/QWxLqUf7S
iZ2pd/BBPio0kQQyVO/4LO/hsd8ro2sYQxRPYJSJSZJYwhHSBkWoJUABJRqTLRM0m08KKlPY
wgiR99kZOdlRi7lpOuKSS90wJIrLqK2M4yZTaEGy0N4ml9M96W89Hqs0lVu0nt2Ii+YztPv8
XF8ZZ3EPrkAfj9dyF50gaJhscwEP5U0twGrZlKkEhKwDNkZ7gV27yC7ndUxYtfwkymMZMp5L
7KF0pJNj2OmYiR0MMkRWsIoyY8MPsYia/fyZmxGzjdsInAjjKM6nISlaUKJOH6UgK7LAo2FB
Pc5WKh9kJiYoJWF7I08G5SP7sfMrryLuZsYCjZJYlFp39EWKhIn4TBVy8gXvyN/2e/fd0DRE
oY4sqvsEo8lHQSCSV8Z5c8RDJJxAeRUKvmm4FEGDQ/Sl+6k2NdG4glZWPWFJLZWUZi5wbiOy
pJXF9GfIcbrT6HYNNaPC0X4G2+AF0BxEVqZ7k6NvRuN96dGzjR+VZOva+LyMNfgOizYrwQJk
IkYlqmhcMcSZGccTJmCF0XJ/UVAESuuRytOssY8rWjSGVfl2MYsbDKyYh0z4SXyIQWIB60hI
U2D7/6FjN37uVo56XMNpQSqXdjlLpIVBdjlJSilWwgUrtBaQ4xscTc8u7O40z9hYTa3UR2uv
D2vpPIA/dBJooXmLOfQwkIx7SSl3X6RwIY/wDI2Y77n94TDWKVMCQtP0Q6QRpfJJRsJXyfgW
haoCFaJk5SACKM4ZnlxIemOd2+mWhTnvJB4nUlmDsqMK8vVwDRdxYr0yLSkEekIYfwrhyCjX
kqmajVFlvMc0M++k43vh2lnKzfxqmpRrRvfRwWHlPVW7G0D40YyaSBGRVJLV66xSWmZPIV96
```

```
yyZRQSnIy9JRDkK7m6NIESlCIYMgkykv0W3yvug+giaB7LnDrxKlLfvYeT040cqdvEUm0JdF
RhtpbhR84mXF9MuL4crWYjevVG9JoKYhIKbypgsTmH0w3rCLcWkbjBTbD9SrxfuxvSfnyYLD
mvpQIXKzSpoFceVJlX/9AgTjP9j2/FsYyRVKQcdWlr4QrHIzQhYUIsbSq7Py2Z8/CC0h4cO8
rRw3uZb9mKvstFZJdoixISO8tJ3lCuTKxItUjSYiVNtTqXAotyVBtd6ju9lgWNLpI7skheV4
BevYHFhks4PnA9S/feQup3D4B+0o7WxIjrYUxdEogWs4j584o+DkCk3kku9YJIlTjikZROUs
IGKfEvvJDyg+EB3wkSj3yD2dt7OTnwdk7UbySwq4pE0wYiNYUEavOJNJfiF5OqoVw55HpNOU
ICEKVBmgo4U7+OsBR2Xu8itvsyFu65GVIegsRUpkNlhiRNRVvY7+3UellZ9FfYBq+oAf/8ya
9rE0sAItlZAclLAUTLpIkmJfQoUZoASNLrsV/x9Gg3vqaLSNcWkagtQq8rUY6nxPT9tWXKUQ
/WVygJKlJUnFIHMFKSEqvPg+4K0rU5pG7IgdpSkjdUMn/NBmbarmLuln7MR/4NPE8qkMTQCK
SlhtcSnlIoqyTpn++nFz9CACKaUsiWyqI9RD5ICkLyDhF0YmYYQl+CmJhVi7BwlNPf+zK/Pt
BGtPsSwlefy+w7zlP+RazrYhYbN3K6upTFunIFDvE35H0jDQVKMKg2qCtV2mWxJksFOHivi2
TdxczdNQqJWaKZABKtIxO2KhZshlwe09JuVtWxFaJ/8dd7LXz7ugaINLb4FpZ/YYd5RZ07XJ
K0SoESaZOY5icj9dcSvZLy7Ie/y+mBBsINW0nXlJKsKVEAEfvallCCTyI0dWUogNRV2FEbkZ
6lSoKKjR6pd5FoLMLcVUSqzkXyhgL09+ShS/a9fjvB7qs5PlJP8vufhugzKjEmxp4wsDyiVI
orS8PGibP/oowhP4oIdrbOvi2V5VfHtLJyDhLhk5yPbmmN2Ax4j5L6lbc58pGDPDTcyFNd78
SoLoTq9dBUjL67grn6ck5Ul3GmpgJPQxXB+nLCDW4VwXLMSzG7YrVWW/h2ZqHX5cFlhcTrt+
G5bS/EThNP+YjKmwowklEwJWJllZo4Cl4EnHqdF33hl8aXr2uAqCa0TQtpaKvCVuobjGVTS3
zAuCHJvCCYAZICEM2DcecowV0X2aZDKbo4ohLOFC2xTGJarKA66xjruGIideUqbyI5AUvTyP
luxBwzatzqev66TXj73g2fvRWOPQpxj2KaUlqIZ2NhO5ttMbcw+LJ5btsjmYyE44QEHBKIcL
ZytDU2Q5KOkmqRrELELsBPZOKQDkPKC8YMnPkl+r/fydHhnfy+YRvzjdtJtl+GUVNMosaFVu
dW7y2ACDRUqPdR4Kh3tKZoEREK8t5WG4mppTfkE63OVu8c7ryMk/vq4fhvVeInYJorhJVIAy
vULhUN8p6veXQAr2+ACCrEvLATcAlV8xNUA4WUHo+DRDWTCRnLEADTgxxDeAbz5naSNWXLwB
Cmf7kkibJw7VYCddvwN2zE2liKv9GlnHkV7arJg8ZKaLmUeP8NxO+cgF9+T4VUpZJlMRlTCb
uzDvXZ6FPaCYvKOzpaQrbCWYoEKOJbpfASZyEVYTERJGRKUFukdRiklmzpaXjk6yzePcLxve
9ltmkbkboK0tUlZKrd6LVnGd5h/L90K8IgUZdPpCbb8km6L2W2fycc+ZXKhQTkOaV/RGWIv5
XRVd9IxbAChy3cXhu64sWf4u8OIHGkFF23OiYOetQkrQoHxbySMYE6zD5FdLzOju8XK5NJpO
HLJqVdJPy5Udnq3sZiPLsL8O7OIdyYrULEyVo3qcbNeOsv4Wjj2zl2yyCZn/4nhE6qkXppCS
RImZZpkRPkUmkLqWdaaTeJ8xJNk4mYZMIp0mHNMhkTS6rM3Kpt8oG5CN4j8Mwj+L54JzOHez
jW/HbmqrcSq5YQtovUNdkYl65Hr7WlggpGSEDiLycBiFGbpwAigsHTtZ2Z/hvglz9WWjKsRk
39AyAvDr2/xrciXYWBVmiQuCqjtgFiQDRkmSLL5obklZ98mKW+d2PsLEHCtS8bGDaYRONIIk
2rLVcOqwzTXWjJY6ElB//ubKVJQjuyiVaLlN7CfMOlHG27iuMHOkh842Nw9BcQ84IeAc2maB
hiMuYiCJEgEm3j2afh6FNw5En4/eOkn3iUzOO/hsd+Bk88BI//CJ78L3juF3Div+G3P8T4z3
vwfewQj7VdzTONV6roWrLxIlXGn7khH65bR2ZHNhKJezmgWHmsACReY2kQ365Cljq3MrPnGv
jJf6j8SlTU4PM0iDUGRHyjf2iQvyZARHur6Ig4wFGVEFQdk4BYWPS8HCCOokhXP+Eff4XZji
uIVxe9IoCIeZLaKTa85bB7mwqYb8ljtilHgUQ0iUR+gtflEb2uGKPhIoJNlzHbdDmefTtI3L
4HfnUf/Pwe+OHd8B+3wudGMT7cQ/C23XhursV7Yw3ew9V4DlzP4tSlLIy/h7nRq5nd/w7m9l
lFaLgeT88OllquI9xRjdm/G/qbibVdz2LNFczVXsJ8/RYWaivwlrgxGlywuwh25UNltkoGrm
T6l7MvANGr84hU5+JpzGexfSMz3e+Cb34GdC9xGaIrALESMsrekuG/UtHwD4D8DQAiDS9DPs
VZd0I6CTFLRJJJNEUAEjnJwlc+yJmWi/4HAOLG3OkiVS0h3wKVPJtvKWCmLQ/Zirnlr8kjur
OA5AlukteXkthZgV5bRbhhM77GSuYb3cwJsHa7WGh2sdTsxtPqwt/qVhRocRNsLiDYlE+gMQ
9/Qw6++vV469bhq87FaN5GtHoLwasriFlVBddeDDVXQtlboe5StOrNxHZVqfox784cwrXrye
zOh0YpmblQDTF+OaBYeawARNuZvwyQhfYK5trfCl+4EyLzmKrU3/ZDRE5lZEShjLBcAZC/Bp
+8gnu8/n0Q24GV9heAWMM4ZRyG3TGSaLYdRFI+mP8tpz48ohKEEmmSTrZCti9/K5EfAUequk
AlEcUOX2h2M9viZr65GE9TMUvX55KQ7HRtKZnrCkm8NwdzRyHpWpcybxJdm9G6thDr3EqkfS
PRlirCzZWEdlcQ3F2Kt86Ft64An02B2gKVxJNEnqIGF776QjUSMlBfonI2oZoygjfIwK9CNf
```

```
hL8jvplgqSzUWEG7Lwla5hqfZNeOrWqPCtFcaVUO7LI2m72I48wjtzWGrMYb6tlPnWizE/cR
iC4mNJlMQ2g6WfVFhXQs82QETtv8Y/r2+ASOPakRALICKh7LCW9YX6XUq9JcRI0gfHH+H0+9
rxNm9RodhXChAJkUrNkoAssEtAUcrC7nI8jRX4dlUQra9QtUzmDS7YUQA78mBnNunq9aTq85
h/73pmr8ln/rpCFm9w4a0uIVgrSbhyIgllBGqKCFbLAC6poC2y/SUr6y8l+r7GXPxt+QQ6ix
R5W4tZairEU5uPrzqPeEOxer5kvYtUc5Eqq5mvXcOJujczs+tCAo0FLxsYDpCk7aLXWwBZbM
pisa2IxeatpP5lBHzPWZM7SD8ICRgUIOz+Wf77tY2Qlz9A7Ia2+kEST45TbneMjHKNZ0gbGs
Tm4dff4Ll9V+HdVUFkR54aKBWuc7EakkFWMaGaIsKlJWpUnb+uEl99FYHaKjWuRKJGcSlf2V
GiRt4ZNdnEa9cRql9DtH6dun+0RiqHy4jWlqhtrKYEi4rUtbWaIixyEa92Ea8pVNtoXT6epi
zmWtcxl5JlUy4Lu3PxNeYTqi8kVluIVmNRTLROg5yTylxLLvPNuYhzHWhwrYqkzVKitXbmEe
l2M9ucQ6BlI6cHr4VnfmbNpyUdY/uIFhRWRFVe29hQT/d3ARDBiAMQ9VYKNHZ9kgKIOIoJ0B
bgh59kpu9ygollRHfmvyKAWKCywBGqKyNUW6lAEarZpPyC2M5NaIoq0apLFLOG6yVnsA7f7j
X4G9cpYMZqyrBIgCH7Jep4BYrqEhscAhKXTWcB4m/MwrN7Hb6mdch+qEEKDHMRMGil+QpMAh
ABijyvgMG3y4WnsVCRf5XgkOvI9TKNlerawW4XpxsvVLVbZ/a8Ax7/nnLUpWOk9spRIhZabC
H2D4C8+i3gaGoLILYaV2aX3QkSaZTaCylBic3BFw7i6d5MrEkYNl+Vi0id0epIRtcVWyPspC
SjbgOxmo3Eajaj79yKXr3ZcpKlTFyNpXCrEvHFlmzm29YiWzlfEncWSb2X7MvWrvla3p7NaI
s5p0LT9S60+jwStdnKbEPqwa4VyoNrClQBYXqHW5mSco5VmWuFtSUCJyFu+V7GdayG5Fx2bl
ZmnL87n5O7zlNaa777CvjplyC2ZJm4diBL+mi5ImA5+/nq88grucPrXoM42kPgIPtn292ydS
URJ9l09aPvGJHbWwm3lxFvLlYRHLHjpZJ3NRRqLCLYWKK0kfgL4m9odVUqShWv2YReu4loXZ
UayhtoKsHb7GapJZ+FtlwW2rNYaslV58t50Xo53yJlrYYSIg0lROltZFcRFpl9luguqX/KJl
mXT6Y6H3bkg+Q4ri+EGwphhxuzxi7ErC9SlcrisCdqyzCrylTwQMbdhxrdqyI5l8aNGLtK8H
bmcqrxXPRdBYR6r4AHP2dVDcuQAztXKHlk9Y/s2QLslXDvX+Hcvz+ALKsSVd+gZsuURLoCyP
HH8Ey8B6PFhdkiYz0KiO0qItq4Ogo3FRFqKlYUbSwmtqsUvaGUeH05upSIN5QTbiwluLsYX0
sRvhYXntYCfKl5ioLNecQaXarcXkruHbK+KyDWWIDeKMesoKZcYjbJOJbw7gKCzS5l7aVWF/
PtFkmoWL6X90vWF6mK5UxNMUIpSXDWW88qzx3aXbQqiu0qJlVTqTSIpyub2eY3k2krIrnvbf
DtjlrjTaQWzh4DvwwQmfFFjc957YPkdQ0QR3sIJqSplQax2l6VlMiUm6rOT6KKaQj88gGWBq
+ElnwyLUVEanPQmtzEmopXRZHdxUR2uxXJdYTijdY2luQm3Owm0FqIt80if2uh+jvcXIjWlE
+8KR+tSfbPUmx3IbHd+RYJgGyKNOcRacl5PjXnWfdoKWapvZjZzmJOdbs52eNmpsvNggyFbZ
H3c5PY5casd5NqKCbRUEx8V4kyM+Udws2rIzFT9etLlgGy0HYedBeT2Xcl+hduBnlWFVeenW
LIcRb/AZC/gvKzAOGAYxkgdhRRaq5UqbeNIhFaz/3nF/ANXKoAQoubUHWWYh6HyVezFYYWZt
d259lkMXGoNQd/Ww6ejhyWOmWbR6AlTzF+QrLYdYVkJMrU5CYkz/I8soAk4LIoX4Vy/Wl5rK
RQSz7xxmJl4lgMbwFWQCFg9LQXstRhkewLUOU3Aa7zrrK/WlJ+3LVFGAlleLuzWGw/HzpdJH
su4rRUCcwfeXGAqBGVMsJTek0lSP4q/LKam7yGNYg03AtImH0l2X9KM4udq5xASzjZVq8M2L
F/MBIcue9u/AOXQ0chtOYS3HkuWnMBEdEmq6Boi4toSwFaS766jlxL9iOt+QTb8vFl5CKmh5
C3M5tgezZ6cy5mYz7UCxWo84JtBTyfXATank++dhe+9gLObgsIt7owmmQ8SolNRSTFZNttPV
egvQBvRwGLXfksdMlWzrfuFWktUM8uzy/7kRbXi9AL26WESMsKai7FU+0ilFaqAOJrPRfack
k2l3F85Fr43Q9UTZb0i5hZ0keqS6WPVg7DVUEV6UyJBzsduAI49kzyZ7veCuf/NQD2NweIem
m7ASwbyW5FaShFYh8JOWMhbFTY2DH0FKFYnEBaJmawf9OTEIlDPIVKEkpnhDw8dfdBwmPXIq
FW/7X/C63h/6J3FBDrKH4F5LbPLUVrq0Br20CsfQORjgrCnaUEu9wEevII9uQQ6c5C78wilZ
YDLXkgJlPTeQR3WxRqXkOweR3B3VmEmnMJiXllPlu0qwShSFcZEdnvLkXvKiXVWUK6vRizrY
hUq4u4Ddjobjk/l+DuHIIt2YRa8wi35RNtK0BvdxHvcBPvLCTTX0Zqj7UfFaEhxyvKJbS7AL
2tlKZy9LYK9NYNyxTs3MCZvnJO9RcR6F6P2boGGtbArlx8HZt5Yu+74blHIHpazSujy8yWUu
ErGJCpkJYk9C6DWVIQS9jDcmWAlcxbJgdYCRQVJl4xQbhUTJiq5k7mPZaLvXqfvzlAnBj5Wf
FgN4wARIAhWmAFQBylImLJljwpE8JpmXTZBoiIKl2mNMkQlepYmazg+GMcOdSJv/cdBGtz0R
r/mVjLG9A684h0Fq+Kwp3FBLtKCXQJEEoJd5YT6SgnllFqkwBPAOQmlulC63Ap5jTaC0i0uZ
BttD2HyDIJE+cqRhZmDreKlrA0RailUDndgd2FquZLEnzeXXnonUXqurF2l2J+0QYWMCxmlz
```

```
rc6nfZxtoLcY6LthUSEQ3TkoO3JQtfS5YCUrgljlhrAUabmlR7MYlWtyKjrRi9vRit3Xq3SE
cp/p5iTvWXcKbfRbgzF7NlPTQK5aKllnFqz0Wc/FAXxq+/BimZ4ySKaWhktKQFDBlyvDLEJd
EUGXkY9lujEJVQtPIoAgM5lNIvUq4SVmR98/cMEOHpleCQ/WU9LA6FTUqb2ClZOcYGhxxtJF
LEjJTS2nYLKmUkQzplmeTTnIOf/RvHB65Fb96OUZ2P2VeAv20Nka4CIp3uVZGAwtu9AU9PBd
49bvx7cgj3XECs+3z0rgswOrIsBmupING8WRUW6ilb0No2Kk0T7thAcM8lBHovw997Ef6erf
h6qvBlleHvKsLfUUi8vlxRYk8ZiT0VJLorMbo2kOisIt6xAbl3I9G+DUR6qwjvqSDUU0mop5
xgdwXB7jKWWouQ6Ja3vQRfRzGBrnLle3jPBoL9Vcz3livy9pap4/WOEhKtxdaUq40F0JxHuj
WHVJul/cI96/H3rmexfy3evvWEOwrR2oqItZURtSncXkKoo4iFnjKe6d3MiffVE/7OR+HMb0
Am7ktrpE2ZsClDlATNnsPMshIMezoima9LhuxafS7yUMAhW+vj8IYwwtH2mK0AACAASURBVK
v3+ZtrkBf6FC8KDtU0KxpiBaDiCQMhmR9agUa2KUvaxJUenwf9d6S+cjNL3ZdD6zaodcPQRu
bblqsSiWhXEashMXVC3WUEe0oI7ikk2JtFqG8N0T3nE+9eQ7IzC7OlALO5HHP3ZhK7t6GlbC
fcthl/5za8ndsI9V9OsO8tCij+nm0Eujfh79qAv70Uf0cx3uZ8fM35BHbnK5NHImDK92kW/8
ClGFGAtJIsX8XyN8I9xaykUHcRDgX2lLDQVcRiVyn+7lJlthldZUpzpFvdpFsKSLflYrZnk+
jMQutZR6j3Qnz9F7K0d4l633SzG5rE9NpIoHMjiz0bWNhTgWdPsSU0ess5lV7BM3vfhudTY/
D7H0JKlqAKq8k2ZhI6c8kUkUzGsrClRF40iZhYQmI9KCjZUUpHOFqqxP7y7xkg8m4OwysOFy
DI2zsSwop2yOwXzmErtzK6Wf5W/5j2HAEx0FOaWhIMjsLv72Pp5huItGyAzougqRRz3lZm2w
sJdYttvzqA6J2FinnS7VkkOnPRugqXr6VlWXZ9ol2kbw6pVrH93Yjd7unaxkzPW5jv2U5kTx
nx7gLiXUWWuSQmTGs5sdZKoi2WTyO+jfgClrlUgN6Vi96VTbxrHVrHhWida5ZJNNdKCrW8iX
Drm4m0nUus43xlXLz7QoyetaR6sqC3CPaUwZ5i6Cki2eNSzxPtzifcnUeoJ5/Qntw/okhPrv
Jj2LUBdm0m0bJBaadZ8UkGyjk9UM7inlLlR9FZgtnmxtNejveDTfDYVyEzq2Zc8WCySEZNES
TTMln9qEqwQeYSUHPAyJgFMaDtyQGFPeRPoRVy89WAyd9egyyDwgGGAwirLFqcMGuCaqfe6v
lbOdoBiClDVMW3k/GrMsWPcZzQo5/lzGc6ebarglBjIXRuhLYK4n2bWOouI9xTSqy7eFUU73
JBe7aiTHuBkrzJ9kqMjg3EOyuJdxUrHyfWlaUcdHHU/TluPD3lLPRsxttdgdaejdl+gTJhku
35JDrcGJ2l6nytu4pY9xYi3VsI7dlCsG8zgf6N+Pduwje4Ed9QFWe6rNzH6Z4SzuwpZqa3jJ
neEoRRZ/tKWdi7gYW9FcwPVDLXX6a+l9/P7CllsdtNqrecdFcRiY5Coml5hNoLCHQVEuotIt
xfQrC3yNrfU0S0pwi9u0iBOdlZRLqtlExzJWZLFVpHJYEeuUclswMVLPaVEepxk27NgqYLQb
Y9JSwObufY+xsJPHQvxI+rhT/DagWqJFEzoeYPVh0qnSrLRygUSKeqjrU6X76WiMzfP0DssJ
74Fyu0howXsIBhgUN8CecIZyu2qFAsmSSeThNPmWoOWjVBmRGGM0/AL77KsX9t46n9F3O6w0
203QWdpdBdTqijhEhvJdGeYtXx0vkvl2KKWYpJdhSTaSuGllKbykm3laN3lePtKWa+z83sgI
v5/jy8vVmEe9aS6DxfASO9exlmSzbptnzMrnxSfS4Sg0XEhyuIjFUxt7eS0/u3cmb87cwcvJ
75WltZ+mA/3rtGmf/INOZ370V/4F7iP7gP44efJ/ngl0j9+IuY//UV0j/5Mvz062qrP/B5fP
/xUc588Q6euecmnvzIAf5wRz8nJt/J3P4tnOyt4GRnKTMdxfh6KtH3biWz/2IFVKOjklS79U
7yXplWi8SJj3bnEugtZKm3ksU9W/D2bCXSWYnZng9ta6BnHVrdPyEROnPsYkKjb+NYl8WEpn
fCN2+HM/9F2jyhpmoVq0F8SSlh8b70tQjHsyCRCejsUYoCDvHa/74liAMQx5yytgIQmQjO+k
/A8XyAOOCQBtRUOtCadleNGpRZA7lHSXzn4xw9UM+xfVfyXFsZsaHNJPZWqmiS0Vem6qBifW
WEbcn4csEhx4v2CfVsINwtGqOcVHupKrVQ5RYSnu0uxttTynxfKbP9xcz3F+DvzVJOvNnxz5
jt5xHtKFQOdaC3iuC+LQRGLsY/fQWLN76D2Vuvgf84AN+9HX5yLzz2HXjuUVg6DiEZHSnjlk
PWeHZdVqKVCNCfIRn7LueF/OA/BY/+O/z4bmJfmWLu7g5OHtzByeGrmN/7Nvx9VxDu3I7esZ
VU20YyrZXQWm4JgdZiUh0u5YcsDmSz2F+EV4IEXRXEO0rItOVC6zqMhjcqX0zrkQBBKaHujS
T7LoaeiljqU7gKlOkn/0OJGfUNEWJpEnAgGDaWqBKcGCFciWaac/Q6Mx9JoygzIdXw7iyrv
k3NrEEIHYo9wU+h0gOAUYsqaslti2QpNXU/VpcxzBkjiWZORD8yTAZAUZiFk79HP2rt3D64E
4W979DRYciPZuI9FQR2VNBuLeEUL+bUH8h4b5CIrluIr3Fq6JwbxmhgYsI7NtOaGgL4X0VBA
eK8PVk42lfylLzGrQ9Vpg30laIlllEvLcUY28xWl8eS3luToxdzLGb3sviv7ahf3EafvBJ+M
234blfw+yzEPFDLAq6zPUlQ4nFiXVyRLIvs+OtklKylqBMYClzZy2A/ll4+mFS3/8c/k8eYP
HWDuZGriW4753oA5eS7tsCPRXQ6YaOPOjMJT5USnS/lF/lYu5dT7L3QrSutfhac5nfXajMQ6
```

```
lrE2ZHBbQXQlshtBdidhYjkbRj+7ZjfGUcnvsRhE+QNmJqfi9ZWkfm+lKznCnr24CUzOIijr
vARozrV//zGgGIo0EsA8oBh4BCZpaV5Q3ipoFuRLEmfLYYJGnGmUv4lYqzSgL9/rtEPjPK7O
RVeIYuITl2GWLHG50Wad0bCPWV4B9w49+bq8gCyWoBUkJo72Z8e7fgFZ9gpJLAeAWBURf+gW
x8XRdi9rthsBIGt2Ds3YZv8BIWxt6B933VLHlsD8YPP0ryv78Gxx4C/x9AZlrXRDNEIBg5G/
wXvlXW+FieVtQywxWLCAOthjIQ0ZJoyYwValeTLAgYF8H3DMz9Gn79NTJfO8zcjdfybO9G5n
sr0PZvwhypwtxXQbK3HGOPi8SeCzH6ziW+9lwi+9bhHShmsbeKQM+laBlvISPRw5YKCyAdOR
h7soj05ZIa2Yxn5C2EPjEIR36gZoGUyR7mE7AohpcpARd5P5lXQISpRjIZQktbK7682lOXvg
YAskIa2supiea0oCLbNEY6QVQLocl0OCr0JwkliWzEVJ4jkzoGj99P7GNDLAxchlcy2AMlxP
aVou8pJdlVCe0bSXdUofWWEtjrYnF/LovD2QT3uoj2Fa2aRFJG9pSgDVcR2l+JZ28xnoECtJ
EiOLABvS9XZdElJzAz9nY8HxtSfgKnnrKWbwuKiSTTg9raQGL/GV2tgmUtj+CYmxZWBCDin0
oCWvalrVb7EQmsEttSki5zc6mtRAXl6iLDpdjwGOiPw6lvk3j4Q8x+dg9P3nwlj49cwrHeKl
Lid7WXkt5bTniomDP7cjk5uJ6FoXxi+ysxujeT7LiEZNtbSXRcjtazmUi/i+DQm4kOvgm6l6
vk5cz+Kln8zCippx8EM6T632OkCMQzhGPidwhARAoY6EkfYdOrVl5XczKvtgH+gvP+tgBRkQ
p7PT+pwzkbrX0+QJKamhldqVgjjBlcsObXTcv6GsfxPPYFTt3ejm/gbZh9WzD7iwkN5OIbyE
brc5HsLoH2ckWJnlIFioXhXIReKUCMgVJCnYUsteYQ6HRhDFVhjm9BG97AQn8R0ZveQuTO64
l/aQR+/jmYewIkiGAzpRIEUhKehFjcJBo3iCY04qZGAs2eBURmApGAhQWI/8mtU8Yhgb9oKk
M4lSSc0YgisSUfCRaAU8AxQDTcb0g8803+8O338+Rd7cr8MnsvJdS3nfl+GW5bxtxgEYGhIu
JD5eji8HdvJdJzGYHeS/AObMSzz4V3+MlE972BVOMboK8cfeRijo69g2c/PUbm2YfU9EwChl
QiSTQSJ67ZdXWyTHVGpuQOqxls/r5LTQQgCiRnnS0llVZoEGEGNQuGWndcJK3M0O6B6DzEjh
H4lSc59slWZgcuJdWzFfZUoHVmsbh3HeEDLqKDOaR686FD7GY3qT3FRAfceIfcLOl3E9orf6
+OIgOl+PaUEeqtwOgpJdNdCj3lsHczxtTb8PzLTvj5p+H4g5A4LXljtU6JP5ViMawzGwgRND
QFCNlMIsuzyfuKNWGaGSsq5xTwKfPCWbZM5hqWBXGseiWnGVe3XVn45wRJ4qRFi2WiJNJBYk
kfkZRHMWWcKBphvJlFwv6nmP/mR/D+6xBnRq9lrutSol2byfRvxezfRKyvAt9gBYv7y5gbKW
VuxM3ScD6+/XmE9uUR788j05YNXW6SQ5vxTl7Oczdew4l79pF54n5IzEBKSoV04ppOMmXlwt
IkleJ7ii9e5TDW3l6DOAARBWIXLQqTiGRVklJpFjHD7BIEWedCZicJncH87bd47Jb3MDu+lV
TfRugsJ9GWh6f9As4MrcN/MJ/ovrWk+9ZCVxZ05ZDszVcACQ6KJikj8goBstTlVgDR+qvQ+j
ajDl0Od7TC9z8Kz8oquIvq2WViackfz6QzLKUFKhLUzBCKB4kmIyQyoiXOJkOV7y02j+NbSO
RGrbEh64s4AJGrnHVWpSkdLfyXbOXiGbWIj5irTjLOWqJNli6MabJMnMwSLzPoy8TYsGgmmU
0nmFeGWIq0/wQceRi+fQ/6HYOE9l5NqGWLKtgUcza4rxDvcBaLYxewNHoBvuFlBPflERkoUw
CS+jTJvxgD5SQnt7M0spWjo5exdO8gPHW/NfF2Okg6k0A3TeIpazHSjCwrZ8gqodY7vlr//u
0B4jDACwCiwCHSlJnNXCIYRggML/hP4v3ljzh272HmhreSEQ3R6YIOF+E9LuYHCzgxms2Z4Q
uI7lsD/edC77nQvwZtMJfwYAnC0JIslKSj9hatirS9LpjcgD5UxPzQJvx3NsGDn4b5p8CIKZ
QLvrUURAzQhcdVT6ZVLZKmiZ0vDJ5W852bmRSptEhHK/n5ohpBhIhNZ5ljZSO+jH2ZyC0WIS
Mr28rAfclkv4gfGE0k8IdlASA7ky2HxSGyFLAEl0j5yAI8+RDpL9lB+OYmYvsvJj3ottp+4J
/IDP4TqcE3EB9cp9o6vHeTigDOdJYQGChH3ltKRLL2XbkEhysJvO89nP54Jzz9HYgdVwBWII
2nSciO4DlqC5BXCx0yu7u8qwVCJyehQgbqO8dRtjpVDnT8BZHo9okrStXVdV7498t+eOlgub
5M/iYmlTT+HERmQJ8Bz+8JPvh5fvPhcZ44cAPcciUMCkByoN+NNlLG3LCL40NZzA+vUwDJDJ
yLUHLvGqL7cgnvKyLeV0Wytwqjv4j4gMvqtMESooNFimKDLrS9BcQHLBIQRQdLCA+WEdpnkX
9ISr3LiNlVAz+5C+Z+CvE5VWQnTq/Id6nkliWRRQEobSDOpvRwRhxhXUlwicwlUjpGUkNPxR
FzSyI5MuRLcmEOOf3hbJf7ZYXHptruL/lbhI+Ej6XUPGaQihsqfK4nU8iiQ/KEihclepZIkl
RIF7SnIJqBeFqZPoYCV8oSCoFZePS7ZD49RnjyCuJ788kMnAf9blJgSQ2st9p7oJLA3q3M79
uOZ/92fH3l+NrWo7WdR2ZvLsbERmYPvgX/v43CyR9BJqiESzSWsCbhcBpFAdriWcWTTjvLlu
FRtSti56yGtnj+zzPnOdIIEuZLRHzWRMpGkIweJRQHX8Jad0KTq0lvSLw9IYVkMjm0tVCNDG
lVK4DZW8GQREaU8ycxBzt8n5Sl+WxGERvbuaSEcWUFc7VMmJQTqHU8vJCYA/0EBJ6EmZ/CEl
+HH3yIlOeG8XygnpMT7+TkwEai+0tIDuaTHCwkPlRIdL+L0IiL4GghoZF8tOFc4vuzFen7c4
```

```
kOW8doQyUqY53oySLZl4e+v5zo+EZi4xuIjZYQGxJgZMFQLum9uUT6iljqK2F+oJyF4U0EDm
xn7rZrmPv6nYR++6AFYDH/TA0jFkXTk6TsUv5lIbTcYbYQULkfQ6U6rVUUrVYQplT9smIrES
vhCetazvmylLSMo7BzI85WKjdtyiSdlWlF4NjHiaZQJHMi2ReWrf2gspF7Oc8hW/lbfSQULL
NgJERD6giYxPSSY6TfVWjNG4Hjz8ETD+O9o5fZ4Svw7CmFsUoV2ZP2jPesx9hfgXffpfgmrs
I79lY8feWkutfAwAUwlEVsbw6nRyvQ75+CpafsnE8aM2pYkb+krpZ6ULP5y4vI/QUrK7MG0l
RKP69MPC+/qvlSf3pzjrycNL6KMZuy4IosmSXrOFg2pyzMIq6gxdEyCCmolvjLxCIktJgFBF
uxpFMmYhsSF3tCA0lUtzSmbkkXZxZzZ/bymAf96E+JHvkRoce/jf8XX2PpwXtY+vYdeP79MI
Evj3Lqrnrmb78e7/veReCmywkf2kZ0spLoWCmx0SKL4YXpV0H6UCHJvhxM0RLDZUTHNxCcqC
I4WkpoqIDI3vUw7ibRn6Wknj6xhfmRbfxh/3ZCH2uFR7+szDlJWjqfRCJBLBZD0zQljZ3vX3
wrvWnZ/zJ+XiJVKxnTAYnDoMK4Vu/rloMumeWUQSadIiMqSqSVSB9RVbL5clslzUTQ2bafLX
UdPpNnEZKntLFjD2Kzw+xqQRxL0zjaRiaxFPmZ8SVhdhGO/Abzm59k7uYmTvZvwz9YRmaiBE
bzVHLRNlBFYPhyAiOXEhnaBMMlMFkEY9kk+y9Q/ey9/Rq0733YWutEVqtKGKTj0ioylWzMjm
bZAHEALw8u+zZABB4yq7z4U897nxfvmOVvzxEAyMupj7Lzg6QTIaJxHa9usBCPs2RohBMBjK
QPM+0VllI9WIogaWbI8BzwDKSPgnEEor+DwOPgeRROPgJP/wh+9Z8gSwR/+cNkPn4Lidsnid
/aS+TA9cSm30ls+hlEp68gOnUZkaltxKY2E5ssJzRWQHgsh/D4WiJj5xMZfxORsf9DZOz/Iz
L6RqIj2asChwBK2+9SmeDEUBnJkSLi48X4xzewOFbJ4nAxS4NuVQ+VGK2AkQLM0VICtlxF9L
4x+Ol3rYnolJMr89IliMfjpFIpMlK6ncmofadpX3wrjCzhSyHLbBXmdCT4SuZUHa0Uhy3BJR
8kmfCMwMoKdr7YViw6iY4JU8jvcm2HrCSb46BLyMAyQZbVuxxonWidZONOvhKRICR5KusJFC
dawJRdeXjh4XgKluYxHrqfUx8Z4pnxK5kbLCYxlAWj64n2riXen0+mJ59kr5vwyDYiBy4iPO
rC07de8cLMyEV4b3sP/PgOCB6l2wuloSWJLIsiSTDheRpkBQqsuj4Bk5AVKZRXUzLkxTtm+d
tzfAYE4qBpcUjIEgGiQXyoiZ5lXTtZjEXq95NnVHUskT9A6Aj4nwLvf5M59XUyz34W44kPE3
zkfSx8e5JTX+jj+CdaOHlXPSduuZaZm65h6dAlBKbeTXT83ehjV5EYfhfsv4K0lB60rrOofS
l0rCXTfSFmz/kkxLEeW09mYh2pqTUkp8/DOPBm9ANvInbwjcQOnE90NIfYiGtlNOzGGKtSlB
grQRsrwj9RwcJ4BXOj5cwNl+Ef30RkfBOJySrSH7gavvMvajkzlctIWhJbgBGJRBRAlltWBJ
iYpC/lUWgQ6blCggtzvZBWMqpjPigTl2JKh4dfbCsOdixpoKVSyq8xMk4pqMXcsqSoQUwZuS
Jd5Vbq/nIxYXKH5L5iItt8L5aHkPxsiWo7BJ0W/8qW5vJ4coK0kyyc+uzPCX39Vv5w8F0cE5
9utABz34UwKIGUtUqTL+zfyqmxS5gbrWRufwmBqYsIT28jNllB5K6rMR7+FETFz4OIWPzqGa
wckWJ4555yXxsBFkAERJJ0PeuHOL+/VBedE08lEScrbUSs9fOiJyzJ/8y34NHPwoN3wPdvg/
tvIfXlQ0TvGWPpQwPMvL+ThVvqWRi9FN/wJvwjm/GNbVQMFZjYiH9qE/6pjYQObyZwuArfwU
oWD5QyNlnEmYkCTo/msTCUA6NlsK8Y9hXBYBH0F5Lqzlcx8uhArpLy0RE3oVE3ofEiAlPF+K
dL8B8sJ3SgVEmaVwKQlHgF8fEqgpOb8E5swDfhxjPhZmZqA3OTGwlOVOKd3Er0I+3w8/vAd8
RaalySe7bzqurCEgkr4ma3tmiUaFTc9D/zkU50ACGdqrhcfAMxfSSDbPsJDvfLMSIsbV9Pgg
GJjJVDEfloaRFHq5hKeyjZmZHsQep5f8tKHWFSyh6I2KFcuc3zJPELASIxhhf4RkozpXTSMl
ow4YOMU0Wlq+XvTPFX0mIO+sHzFNGHP8cfPt7PsQNXkh7KhvlvhulcMjeWEzq4lYWJS5mfuB
zPgSuUsIrfUoV5sBD/SCGhuxvgN99S682L5SPPIs8szaIwYfu/0qwWCXjEz3PIEir2j3+mc+
Ac9VKiNdIRCBwn9N9f59i9+znz/muIve8KEoc2Yx7YTHzqEiITbyE49jb8IlcSGroMfe8WGK
6CoVIYKoEhqTsqJL0vj8S+XLShLEJDa/HvX4t/bC2+qfV4D67Hd2M2gZtzid5UQGK8EEaKyI
yWYI6VKDMmNVZOcqxSRTK00SoiYxsJj2wiNLoF/8hWvCNb8I5sVcCMjJagjbpXTfGxMiLj5X
gnqlicLMc/6cY37cJ7eAPewluZP7Ad874BePJ7EJoDQypKlz0Ba3KOdFqZVIq3bPPKMbNeqg
fOduJyb2LNdCdOthTnOSTOtQUMcYSFKcR6ERNHysPjCYmAxe0QsZSKOBGbF9864WQtIwCxAj
GOLyq3UcyzUhLb9xZQOiFm59klEKFAmohjGCFMQxYp8trl60FSpoDFHs8hyU0zCAkv0T/8kv
l/uw3/eBXpobXo0wXoNleg37gR7/gmTo9eyszEJSyNFhCbyiE5uZ7wcA7+qYuJfX4M89nHl4
fjOs8ijynkyBILOKIlrECIlWK2KeYIJTn5JT7npPynSIdnIBmA8Ck8D3+e37x/J8+OFmMeLo
DRN8LIG0mOZhMdLyI0uYHQpER7yjAmpPAvm8DeLCKDOWj78zFGXSQnXKQni8hMuUmM5dqUTW
IsG2M0i8ToOoyRtcRGlxM56CZ4qJjAwRKC08X4J4vwjxcTGCkjMFKhIkcS3ZDx0+GBzeiD2z
```

```
GGLiOl/0qSIxersG5srJDVUGSsSDnkgZES/KMFeMfy8EwXEjjsJnmzm+htF6F9dRKe+xnEJa
MLft3qbmlXMyVh2hSmaSpSOZuXaOwX/mRlrGPDixx27CdhfWdfutkyfWTPAYcDEHVN5b+IDy
N5ColWSeTKIJOMk9AiJPUoqXjMmjBBQrLyezqpQC3Xs3wJxyKRu8i9rXs6v8tWgUcBR7SaDV
47IJYwIW5CTGaXkSy8MsAkT7IEyNqQHtLxBTJSjStMLNp39jm0f59Ev+sqZqc3sTRRijblUn
6nRK/OjJah7V9LYug8jLE8olOleCY3MHvrlcS+eYelarDSThaopT3l6eV9HLJmQHkBQBzwy/
bPAURlZePyItKwUcK/e4AnPtzOsclyuNUF+/8X7P9/SIydS2wyj/DBUsKHK4gfLiFxuJTgeA
H+cReBCTehCTeRiSKi4y70CTfx8UIFltR4AamxfMyRPFJDOST3ZZMaXI+2fz3esRwWpnJZmi
pQzOmbLiI4XUJ0qhxtagPahPgIm0iObiYlshVz+CIYvkRRemSbKgpcDTjknPC4G+9IMZ4RFw
HJ8I5egH86i9BNbqLv38T83deC/7dqzUDhKbF5fRlL4lpdIZ2dUqaVOOcrASKgEd/kpT/SQl
bnOSsvWdEWcTytDLbjBJ/VWfYKthkJKftAPw3h58D3NMz9Fk78itQfHiH++x8TfeKHJJ9+CP
OZn6rvle/+oxA7AfHTYMxb0UUpo3cYXkwhO6sugBVj7awTLuFKu6hSwsiqzN72n+xKY9FEEv
iRtXZlFf4JQnqRdPy0BZC0rFNoB4bk9aNPwX99kOMf2MGxye0EDlWg3lRM+MZStEMu0vvXwP
CFpCbdyg9ZHC9jZnIz0Q/VkfrG+6wlLdSzW0EOAbGAYzmqpt7BgYsEQ+zkohz4lwBE5alslS
0Pju7H+8h9PHf7e5mfKCA2/Ab0oX8iPHIBwSkXnuly5ibK8E+Xod20geBUGcGpCkWhyQqEIp
NliqITJQjFbNLHixGSaJFFbvSJfPTJvGWKT+SzTOOFygQTMywxJtrITVJotFiR/ClAjE0Wro
qik26lrbxKjWeRPHgh8Yl/JvIvJfDgYdCetMpaZGJ4exlykaSW0HEkvLTyS38EPLpujWER08
v5SA4jri2RSPqUByDgEEhJ5wqjiQcTjmtkpP5MCjNTpyH0Gzj2Tfjlx0g/cCOBTzQS/WgN4b
t34L/rOjwfeA/ztl3F7C3v4Mwtb+fY9GWcOHwFc7e+E9+dl6J9tJbUZ3bDfR3w5X2qnJ2nH4
KFp60aNwmjSoLWtH3S9JIlmMk4qRKlGVl70IgSNNJEpTRAluUlksrmk5GAQUAMLClVl9IaaT
elaKdjZknEzp6pRDGoVEZEn4Hf34/vi8OceP+78N6yidShtaT2/2+Sg2/CHF5HeiKPuPieN2
7Ff+M2jOkNykcJf+cTVhY/EcaMSvGqNf3TYjyCpoSM5YNYsEla95YucwDidMaf2J4jLyDk5D
vEZDCO/YLFL45wfHoL4YlczInlJCfy0cZF/ZUpM0tCsGJmCRhCU2VEJmxgONtlkJQRnRCywL
JyG5sQoLjPgmA8n4RNFkjyiE+cBY/Yos+jyTxik/kvCQ5R2S9F/uFcQuM5hCbWEphah3nXBn
igD05/H+ILymwRwSFttBwOtxX4yymWE5CIMy+RLdEulo8i/kJSMVw4GkI3pHpXACg9KEwXAP
8JMid+jv6rL+K7/zBzn2rmlO3v5PTN2lg4UI55axXcVAaHy0kdLCI5VUR8sgB9rIDYmGSkXe
pv0ebaeN7y97GRPPxjpfhueQ/zd7TgvWeU2Pl3wi++Bsd/CsGnWewSOQAAIABJREFURahl4j
ikjoM5Axlh+6gan+PLZPALMGI6BMOkQgaiMBleEpDMZ2TuEqvd5L2UllUmoJPIdELWsrLvKT
Voau7rh/j9wUtYGvg/MPZPcFMuifGlaENr0MfzCB3cRHi6HCZySB7exKm7u+Hx74MmGnEJUr
IMuEwimCaUljCEFbZQ7SlWkoDT0RxnZdWfgAeoRKG8lLyIkPrI6Lxf3Mfs+68ndnAzTBXDaC
EM5sCgC0bKYX8pZl8+KQmPThYRn7C22qSzLVLfy29/mkrQJsrRJiptKle5j9hkKdHpEsIHil
RFbuigi9DBAgKH8vEfzrUpm8ChXGLTBS8JgJcChzBScjKb6NhavBPZ+O66nPSDN8HSTyB5go
zhUlBQhpI0pmpQm3ntBJ/V2k7D/fFWolkCCgGIbJl9OVKYRq4dk7XSRTWpS8cgeAaO/hfmw/
eyeO8w3o8l47ntXSxOb2RJqgRGckiM55GeyleCy5i0BIs+nqtsdW0sh/hoLrLlcCncWAIHiz
EPujGnXDjHByeLmb3xLZw4fCUz0xfhnd6Cfsul8OHr4EsD8K33wZmfQeiYmi5GTC0pwg8SJY
IMWJIKWystYbstjnqVOfsIJayRJSJYxMhRIJE2VOZZlBrbk05jZsScFDEUJn3mlyx8dRrPwU
loI2tI35hN4uA64pNr0cfFhy0iNV4E09nKxD86fSVz98mIxAdAgVnK8wXESfyGGHp2vylgiI
8mZDvqf9xdf/TNOValpl37lcpYGVnJpi88hv6Nmwh94NlKnaWHlsPgBTAsodkC2F+golWm+B
c2JScKEDImz5IwoZA2ZZEw9DJNFdmAqLSy4lPlyvcIT5cSXgGQ8IECwgfzCR3KJXQo26blhA
9m/lmA6NNu/hQZ0wVwcxHBiRw8H3gLPHirFeJOLZJJBkiZcdWxwruKeUX6qQiRdKZjqyvD9I
8alvlC8iOiORz/RLYCFkVSniP952BOxpk/8wsSD3wC/zl9LP3LlfimNiFCJzNZCFP5MJUH03
lwoAAOFpAczyE+kWMx/VQeyelCktP5pA64SB0oIDFVsPy3hErTh4rIHHYr0Ji3lBG4uRLfje
VEDxRgjGeTHlkLYtIe2IL/pqsI3TuM/zsfRzvykLUwJ34y+EixoMaLCOOr9hHGl3yORN5kds
SkTsaQ3IPl+/OOU+UqkuiU8K+pJsEU40hpaIl0Lf0GfvAvRD70LuZGlpO8OZ/MbRXKWtCGs2
HKBYfEesjDe/PFPHvLlYTvPwCen4J5EtIe9ERImaeqH6R9paHl+eTeapIQK8Ln9NOf2p6jXk
YShAkdMx5VJM46yVk48QCeL/Vx5raLmBk+j8jUGzBvXoN2+Hy8h9YSuTGflHTYuE0TeTCRp+
```

```
xFczJP/SYdJIwoYTztQAGxgxZFDhUgFDvgPksHC9EOCBWgH8jDmM4jMZlDaipLkTmlHqH0lF
pF5lSWurZljnPu87f6QRcvRfFDbvzvu4zQNw/D4qNKikXSCbwJqYsS7rWljlOEqVryLwfICx
teTCtHm6hybcllBL2YR39J8PufZunTQyze9l6CUxswhBluzIcDa2HiPBg9F/avIbN/HYmRXK
UlBBzxqWzVVsmDeaQOFZK+sZDMTUVkbnKp743pHLWV4+JTuep4fVLOuwB97P8lMfW/SRw4n8
RNBRg3V2Hcsp34TZeh3XQ5wUOXMX/wck697z0sfWEvyUfvBd8vgTNqftyQDNuleU/5LhLmjS
9ZVdcyMExK9O2aNAGSEgiqUSypEIj6kJiXaFJxaSxTSIYzHIHffZngnVeRvHUzocMb8U4XEz
6YS+qmfNKHRBMWwPu3MH9oI8++/53Ef/JBMJ6EzDwyGjMptWdyTUGpyDYFEvlHqgYkAGFVD7
ywjlb+fQ4pAYc4ZGEy8TAyDkBFtERCmrPEfvkZTtlTy7ED2YRuPI/UbRcQuvHNzNy4Dv9tJY
qRxYESElA4JMBJKHo+QFaCRMAifys6mId2MG8ZGNKpFjhyMCdzSCvKgoksGLdIvhPwvRRA/n
/e3gO6ruu689b62swksSlZYhFI9F4JgBSpFpdxquM4TuI4zngmseOZzKRMkkkmK7YlkUTvAE
H0QoIA2FQs23KXYjuO4xa5SLIkq8vsRH29l9+3/vvcB0KyLUfkzGCtg/vKfe/du/f+73b22U
cA+GlD2biLLXcQvffvXNOAxJLNtCrQlEYT482tMq2z4Tl4BJZ2FISMq5tp+hMfK+4QMF7xlw
zAcl8i8+URLs7+Oc93/ibnD9yB/0CTaX8+tgP+7kb4nze4499tJfP3BaQ+VkHi7hpi+2sJfq
yUwN2FBO8uIbS/mMiBcqItZUQPVhBrLScukLRVEGuptOc65t6PHyyEljeTPfgm4i03E2wpZf
lgLWsHGlm/R9a8nszBRuJ3KdOnJQTlvNT5qyyf+Esy/zIHF75pzTKcKlHZiwJ8zXOsQUxLE/
TcZZfMs/KsiYDiwJIhkgkSS0eJe3OiTq8nHG3TF+HRY8RnPsgLd+3j/D2NRLtqibeVEd9fDa
3lcLDMUvLP3r2Lp0beR+KJU5A556JGscbhwf2gQKLrsdIUxSeql35t/llnlWWZhMuZJxLmOM
jUKS4x5q/8kMgX+swFibdUQlsJyXuKWLm7hrXWZvwHawgdrLQROVCJRnT/lRG7pwo3aoypYu
yVUUXi7ioS98ikv2rodRubz68jcXcdybvcMCG5p+YlLUT8YA0/bSy338nlEx+DJ78IMZXSLB
GO+Sl7pOktBZqiw4YLYb6ryl7JZ47napdeIfU//kRxh4oXFY/oT2BZWVkh9NSXiB95L/RWE7
pnB/67dhBuKSPRXUeso45AazXLd5Wwur+SYGsdkY5dxDp2EWmrJXSgytzOtfllFospPlNMpq
PvYBm+gxV2zL2v5/6WcgKtVQTbKgmllxDqrGWltYLltjKbvFlrKcR/sIDAgZ2EDuwkcqCQ5b
/dRuxAFfTshZ7biO/fbe5w4uBt+A69h8Sj04TOPALhZ7ll7EpbuwIAzY3YxKJueqOCO7cAK2
OrK6XJlYQhHlGdX2xjnkeTj6aok0skHr2XlZkPstZlp5tMPFhF9GAlmdZy+Nib4WAeq+3lfL
/jLTx2/G8Jq/5PYLVCtE0+3gZABA7FKWGviu3HeZZ75Tp/yNlQOulKTsRCmbtlFeXqiUozH/
88q0f+C6udd5Bsryd9sIrwXXVu7uNgDb7WSnwtVfhaagwwAk3wYJUxUYCJSfhfIfBlJO5qMG
F/NUB07mZQRQWAe2Sp6ojsbyB6T8OmY50BUlozdLDcgdPOd6DRb4dbq4m0ViNwJw6WE28pJ9
xaZ77rM92/R+Kxhl2aUKXqcR/B0DqBVNqyesvqGu/R44olcbZapDFxlzR4SsjccM+i67H9ST
CSKbLWmCFmWRbOP84LDx/l8dEPsL5/O9xlHRz4t2Rb30yybQfR9lLCHbUEOxpZ72hmtXMPa+
lNJgTrrVWst5ThaynCBLqjkvWOatbaq/ClubHeVk5uBDqr0PB3VNr7662VaKylCBiVnG2t5X
xHLcudVQS7y4hl7iTevolYyxYi+7eSaCsmcHc+vr8vspQ8Bl3GjL8vx/+3pTx5cB/Pz3+I0L
cnYPnrNuchDS3ayL5KX28EyoYT5ZVkclRZmySuiT4VXmqZg60ld+ltNbSOJNLus5EL8JljrE
7+By7cs4fQQSmJeoKysAe2kr3rBuhvYqnnTr7R8hYufL7TpY7Da+4idCGObRvunmZpsqb+Xm
XVc3zzjtcJDLoZVw9qmXd3aznG68sTARKPf5zHR97PSx23EOtugP0lcFcByZYqp9F0we27CH
gj2F5PpK3GzHuypYy0xsEKktLoLbtsDiXS2kS8tda5AXIFNkYN8dYfH5pA0vkbx7Yqou3FRD
oKCXYWE2qvInGwieSBZvt8qKOCpdZSIr0lJNuLyLTuINmxgzP3lPPs4T8h/O0vuDb7tt5bvr
KIdWXWWs82GGwEExLcMACozGIl5NJQXgmIKKgl58FU0vbCSMdyRVspiKzC9+7n8tQfcv5gFY
GOmwn2bCXYcxOhrulEOgWOYosFRKf0gVrSB+pJHqwl0eJoGekoJtSZT7B7B76efNa7Kljvqm
G9qwrfxqgg0PmTxuZzquwzq9lVrHZX4OsuJdhVSLRjB4n2PJLteaTbdpBsyzeQyElztN9FtK
XZSkKiLUlmXaQYz/XcwYXjf0zyiaOQesqcVFkE0UmeZUpC5imSZEa9zlQR7pSznaS5l2zcQC
WrLbmU6PmlnkWPkkvwlKcIHvvPLB/cQ/hgLQwlEW8vhJ5yaKmlxESwpZhLfXtIffEeiDwJkR
DZtRgRrb7wem35jLFayeb3MiSvQsWmp7ZgShdyBSBrriGARfwuKWGBlvJTXPiHQzwl9E4utd
VDWxns30HmYKkJtsy+QCGA+Dp2EeioNYGNtpcZgdMtpQgoidYqoq3lhFqbCbUlGbiiEvSrGe
llRDrzCXXtINBVTLCjinjLKwHi76/h/P6bCR7YQranEH9bHi9338rlz47DyiWX1dhEkNflUA
wPul0qxcdg0jWyE4OV4wqq4Zssh+q3tCXyN+9nfeqP8Xc2Q+cW6L/BgLHeU4yvu5hgZymR9j
```

```
KibRUOJClVZqlNsYh2baXEPYUQ6nRKQdbBlllLoKPGhmggxaDvsdFRahZJVkm8yA29F+oo2w
BSsLOMiM7tKCTRnm/AEDiSbYXud9sqTOFJCframllv32v8ol0ZpUJz5852N/PS7Hu4/PB+ki
9/BmIvuSJFT6/YOvtEllZPJtTAIhs0bFyxzq4qV+DQkECvqdOKiCsARV6C7x8jevQPWWm5Bd
+BagLt5QQPFpFRVq+lCva/mfWPXk94+l3wvWMQugShhLUzlQLnFa30FGpllmJa3++h9qcwfs
OC2Ps2kSJz57Wdt2rVlNXykF6Di9/g/MJf8lLb7SQ6a6GlwPLUsg7S/rIYwXYxS2C5ApB4Wy
mJVoHDnScLIOsRadlFtK3m6sBhlqPMNJ40b7CjglC7rEu95celcQW6lFA9F+/eynrrVoLdJb
zcUs3K/B/Dy/9sltHSfj+FOP+ql0XfpDCgYj0tNHMzuaYBVfekVObqM2T+aZqzh3+f8wf3EO
+phwFljPJItlQ469DqrlfCHegsM42+3llqRl+XE2RZSCmieIvucZdZy2irrPEuu29peN23rL
pAJeXFwWJQMH5AI3/T2GHvyVLLXdH3ihfio7JZOX4Zz9pUSLiZv/WeEqwnoc8drLCM5lJrOc
8dqOCp7lu5cOrPiH57DvzPeVbatTbS3jnmlJAgquSQVUCahnbklky7uWAu/ltVWbspa7li6m
TztLlbvpk/4kd3NxHpamTlYzutRCjTWQWtO4jdvZPVrn28NPZBOPeoldEIDz6tolelSQ4gWu
JqrsBP5/RluhxZHPsTsy0lJofNXWhErT9jEYfg+EUyXz/GhfEPsNzZRORgMdnWEmgpIdVWRq
K9nGhHtfnPCgDDHdVEOyqJdZSTaK90o63aiBpra8BGey2x9uqrGx3l5jPHOguIttdZABdrq7
PvT7bWWnYj2VlJrLcUf28BL7SW8MLgu+BbRyH2sosHbG4jR4DXd8zRWRv4JBPSd45msWQIdX
20tjzrPyTw5RFeGHgnZ/dXE+2uhqF6/N3VXPhoIamWBrIHGulowtZRTaizEn93OWu9Zazllr
PWU4m/q5ZgZwOh9mairbegIDlxcB+Jlt0kWptJtjSSbGkglVJHVu6H5joOVMKBcpsz4ECJc4
v3F4GNAns9c7Bh4zs2f0/C6FhHzPhT6/FVvNXluRHsrMXfXkegrY5Ie4XxOXCwiMsHyjjXfR
svj70P/5cOk9I2bEnFGBmLm0MpBeqebHoAkQCbrFrHlpAp+JxLJKoaLa29rEpTnod/WWB56k
Osdr2VlYNVrHeWEuoqIbq/kExbPdGefTyx/y2saflI+Ix9uRZ+ZjX772UlfxY49LPXuWpHl4
qzqzKnW0hxacxQJkI8mySrzmJqjLz8FMmHB3l54Je4IC0lgBwsgBb5+CUeSCoJd3pDgOmsNA
LHBBIDiKuWzT2+FoAkOneS6CgiltpMrHU3UWNoJQJIpqXatEl2oJKLPSU83Xcrvi8M2T2QuA
gpr0jz9eFi42wxlNyBTMra0rhIUAyIeKnOF7n8xT7OzPwel9oriXfsINtdQLgtn4utNSx37S
Hc3kiitYGcQG4+CuzR9gbCNhoJdjTj79iNr/MWfB37CLbfYvctcCRa3RBIHFAcWDIH67DRUm
vAEXjcqCZ7sI7sgWYyBlT7tM+GyjcSLfre3UTbmu367Pc7aj2elhPtLCPaWUKoqxxfbyPB3i
YHkLt3krprulksxSyXWhv5Yfcvc+nBj8JLD7vFd7hl7FpLY4pZRPQeO4B4JfGeJZa7agAhSS
q86rqvyN3yPQ/fu48LYx/kcvcdrPcXEuzdYUWvvv3NZHpvY7XvLTx39I9IvPAZNx8jFNr+h2
tuojYHyg2O/viD6lwkL6Z6pkRXrce6IOKEslpxBiGVsvojbrHQil/iwvE/4+Xe20mq4rclHl
oLoa2YdHspSdPsFUS6qgh3ldiIdNYQ66wh3lFFsqOCdJsb+ny8Q69fxeisINlZaL+XatlNsn
UPkc46YplVpNrqoKXGViKGW4t5oaeWix//b7CkAsSwtlxWMx6yllf3J3qHbLbdBfZJdSuRo6
lKhMuPE/nnKX44/C6WDu0hlqfrvJHowZsIthbh67mF4ODb8HfvJty5ylK48fZ6Um3lZFtUHq
IcvzumW3eRbNM5TQS6drPevZvVnj2s9zQR76izCcF0Ww2p9hq772R7nWUb9X3x9l02YulNvH
ok25rg4G44sAe07ufgXlIttxJvvZVI+62EO/YS7thDuLOJcFed8TPWVU68q5h4Vz6R7mJLgv
h76o23sbsLSH30ZtifDy3FxPYXs9JWw/Pdt/PywodJPn7CGuhp8ZTcHO2zmrMcOtqfqollRW
wSz4FDHR8jiaRrqCehlnyqgjd+iew3Frk4+X6ebdnGxfYtpjwCrXeQ6GgmPngrz/btZfWLH4
Nz33OVy+qpFl8hop3Jchm23G//hONlLpz0lkjqKjWsVMXlq0PeSjMtLU5Z0VAYwi8T+5djnJ
n9AKHOJmgthjYBpJBsezGpjmLineVEuioIddcQ6qqzISJLeAUgAcmBqcJe0+uvd8QNIKWk2i
twQtREpLPeQCIhsaCtQynREi7M/hbZpxe9+EqqX9ZDSdxrAYgaFsQ2Mi7GOJWOX36O5FcmuD
j2XlZ7dxEfLCbRl0eo4yYC7fnE+htI9N9GsGsfa9l7We/eY3TUHEe6VcBW4FvhuUflcLAcWi
tMGYmGEkh9r7+nhnRHPrRth/Y8aN9JuqNwE/2dggp0y6XbZUOg0ljrabLfNDAeEBh3kVWCo2
03sfY9BoxQp65rt5l3BSClxLsKSXXuJNmVz/KBbfjaipECTHU0uMLJllrMpTuQB4MV+A7czP
P7izk/9l54TIHzC+ojxLpfcyE5C5GTfEmpBND8IZsLUTtWvysYNqsTVYlXDlnhs6S+Ps6PRu
/kTEcJgf47ifW9zVnezkp8PWVcPHwnwS90w+XHbFWjlqRoN6t/JUAkIF6K0wq6nN3R78tPVM
QvXzDnM2ZDPrde/fJ3OHf/3Sz33EmqvYp0e7mBQwBJdxSTVEamq9wAEuiuQ0wSUGRVBJBsuw
OTgPR6gbH5fFkk+3lp0LY608ahrl2mWdNtVXCokXDvHmKf/xisP+Z8YNMUihYlHbgRgf0E/f
HaL4lGIrI/rZy+Z3mjQVJfe4DlsT9iuaWRbG859OWRHMonNlxOeKCBgK6vvYlE5257LOENde
```

```
WArRKRGgeSlgpXaqJyk7ZSUh2lxEXX7lJCPToWQsdWaL8B2m+0x9mOPBNeaXk7r7uCQE+VgU
mA2jzCXVVk25QidQAUvaRYBFRZDSm/QJfGrldYkGRnManOfJJdO8kOlRnoV/aXEm5vJttzJ3
TeCm2l0FHkZeu2kurbwaWeal6eei/+r4zD0nPWTUV0Ew0tHthoLqHn8mpUzRGlOrbcBIR6GK
tNlarP7XNKKCWeg0eHeHn4VzjX2kS4u5FYdx3J7nIYKuXiR29ibeJd8Ox9toZGa22UZRQ4za
l7DTZfl7swd3ruYldWk+kiNPRlTpSEbK+kIPoMkfkP4ZPpb62GgXrSncpW7STdXUqsq5RQdx
UCiLSejuHuCmNytiMfjVhHGeHO8qsaSgCkuhqtvit2oIB0Z4XNxaylNhLurCfZowRAJckj74
FnPmlBuXzai+pkbskI3ZX5k69Bop/+lughJ03lSM56aFLlq5wb/nNSg78BPU3QkU+m62bCfS
Ws99ey3ttMuKuZTFsDaNXigTxC7TuMVom+GpJ9TcS7mwm2NrC+v8YeJ7oaSHdVkeosJNN+M5
n2G8h0/Dy0/zy03QBtN5Jpu4lMxzZSHZrr2WlaPiZXqLuEaGcR4bYdBFu2ET6wlcjBrTYRmG
rdBll59p2a95C7muqrRNcR7Kpktb3ceGZ866rZcLGk/OQl6PxYdz7hnmIDeKBzL+H2O4ml3U
paZSDyKtol0/3/km7/OeL9hVzuv4VzY79H5stjth+I5E9S5/jgLIr60qXkQtkut5I397oBwn
NyVAakoaJEsqrdet7iy5WhO4n3lZEarLbro7MADlWS6Skhcuo/WlcU/d6lgN+aWPxsgEg+7C
x9TDiVCLl8tK039laKCekK6LM57CnlpqzCPw4RmX4vSwoauxtJ99aQ6Sol0llMqrfMABHoqW
G9VyCpI9RdgRgnl0AjllFyVeAwUHVUk+7Za5mWRPsO0t3FRLt3E+y5lWDfLgK9Dazl7IaH/h
YuPm7VnBJoFUSL2BllM67hT59e9poMWhXrxSfJPjRIeOB3Sbfshu46aC8i0VlCoK+a9b4GfL
3NRDsbnYZtKyPdmUei52biPQWEZRm6aljvrGe5vZHLrY3Ehn+RcP8tbib8wA4C+28klno9dL
8RBncQ6akk0NNowbK/bzf+3lvw9dlCoG8vgf49BHsaCPfUkuiuIdldTqazyEBLWx60bSF04I
lEu24i2b+DlEAB0d4CAp078XUU4tOkZLfc5ArjolzmWKcUXLl5AbJmod5iAr0CUj0BuWQdil
/2uoqL9lLo2k6m5Q2k2n+BdO92/B2lXGrfRWL29+Grolbek0jEyCTlp7j2RLYNQyJFRk0Kve
pbiagstv50lNmSVFecouW/wIUfwmf/gtWeUtZbtsOhCtBE4oGtZFpuIDhxJ0vfOkI4eMHm8T
W/8rMBol8zJSqAyCf3AiRVYOY60lglpkCTm573Yhah/OI34B86OT/8W5xr20usrxH6q8n2FM
FgGdGeMoK9lfj6am3ocbSnhHRXgQlZGRH+qob83r59pHoarYgy3plHoq+ZyNAd+AaaOdfTxN
LEb9rkkmlKk4bljNv5QgBJZLwNQD3Cv96DeLLkbZlH/DzZR4aJDP0WdN9maxfoqUPaP6R4oW
+XgTba00CyS8CpIdtZSaq/iPhgAdGBIhM2X5c3M97TZMK+2r3bQO6XsA80ERxsIDBYjb+vlI
t9jVw++iHOHP0fNs4e+2vOzf0lF4/9FctH/4Klo39CYOYPCYz9Dv6+X8anyb2DSgTUgaohhh
oI9exgvXc7Po2ePAI9eYR68wn3FRDtKzZeiYcacVn/rkobic5qIt3VxtPlvnoCvfVEuutJdN
aT6qgllVlNQuf2VeDvyDfQxfpLCHcVsX73zcRbS4iOvxOe/gSsv+S2VlPK3eaO0taFUzsebH
g43ky7BFp0l3DuWYqQukSav5uBHz3I+vHfZFUKoDMP5NJ/JI9Y+82sDNTy4vT7SDz9oMm5Xw
2wfwbTr7MzDEa6misWxC5MwNmwMLn3BZKUazOpb9e65pcfZuXev+OZjrey0tVEqq+ajPzjoR
LiPUWEe8vw9TliCiCRnjJS3QU24tllRLsrr2pEumuJ9ewhOdBMrCePcOcW4v0lRIf3cXloN8
8PvoXApz8CF9R0IYlaNkW9mEoaKPKvXzfzE8mo2zf3QAHlS//A2ux/InigzhV03pVHtqeGUE
+TaXRpdwEl0VlFuqvSwKGj6p/8A+VEhmoIDlYT7K8k0FdBeLCO8OEm/MO3sHJoHxcHbuPs0N
u5OPluAif/mNRn/ob0l/tg+XFYfdEbL8CqxnOw8jQsfx9e/BI8fi98aZDU6f9JaPwD+Pp/g/
WOt7PWuY+zHRWc7ytldagS36EKAkNlhAdLLLGQGiol0V3gjRLEK4HE8auacHc9a32NrPU3EO
ytJdpdQaqrhHRnCQkpvZ5KQkNNLHXXsNRZRWigllRfubl0tLyJTF8Nl+Y/TPrpT0HqsqvgCF
22tqZG282pYC3BSIdMLC3es5WCScLErIellbLY9N2L8MMp0vO/xtpHt0KPJlQriQ/UstZfy/
nOBlKf/4htnaFCSc86/ET+6kUvBtFDllXDlZyYSXHoNYBonkSi4LLSKhE25Jnd0z5350k9us
jzY3/AGWVAuivIdBdAXz7JXmmiUoL9VajsQ8dIXzmJ3kKSPYXEesuJyqpczeipMdMeHdxFsD
uP9Y43E+0vwD9UxdnhvTw/+3vw9AMQPKPmUWSEDrOGyj5lbYbW7uOnkudnveFlW8IvEvxcK0
tDbzd3KttyI3Tnk+gpJ9DfjG/AuTvR3l0ke6vMH870FhHvreJSSx3LvXsJDu4lNCRNXMx698
34+rfhGypgbayWlZk7WT/lBwQ/9zGS356H5/4Zln8EIc0uez6HbmTzkIqlXsra3FPNHc65FP
czD8PXjhH+ZA/rp/6Wx9ru4OXBOlmbejuh6bcQOiw3rdw0vcAhPmZ6dprCS/YUb/BL9W3Bvg
azjP7+BuOfeJru2Umydyex3mLjdXBkn9VGXehsZqWnkWg4+/l8AAAgAElEQVR/FdnefOi5AQ
YK+VF7E2v3/xU892mIvuB6Msd9lg4sF/ua/pYFSav6llkPFxXHrdG3XLJkWP2lBR25Pefg0U
GCw2/D39ZoMd/64K2s9zeR6CojPfsr8MRJWlhl2bLXYPNlOQTphzcPez2XSVCmwIYKykT5K2
```

```
ZOKTerkln6Lv7PHOTy8C+ZH20A6dlOujfP5gCkGX0DNfgHqgjla3a7mERvsZngWF8VVzOisk
rdtUSG6vFlF+Lr2kp0cAvLvTfzo7HbWfrsRyHwDKgSQK37NVQ5l3VrloNZrlr0NQj0mm8py6
LdWZ/7JC9O/h5Lw7eSHa0glv4GGMojlZdPuL/WQKKYINTXbPeZ7Csklb/d/P6llhJLYEQE8v
4qVjuLudRZyNJABcvje0k++B/gKx+BZ4+D//uuMZsllfO692hNkDwMrU60NqNXeCO8RCPy5d
WxRn66lv6sQfSitwfhN4h8vp3lU/+VsyPv5kzf2ljpv5XwQJNZg2iLXJSd0JlHtnsn6Z4C45
sUnK4l2F9DpLeOWE+V8TLRt5NY/3aiA3kEB4rxidejt7M2eDuXuveylNnEuiaQ5V733ES2P8
+U6bnOvayc+jN4+XOg5hCJgDU9F0A0TOIsw5rT+HpFHoliFK2CxeKVVDrrMlsCidrePjrKub
63s963lyo/ZMEZqiMy2Ezsof8Boe+49SuvwWRvHkT9ZFl3buvMbh/QBag8YNXrqKElkl4DM9
PC7hJXVEem89We9NkHSN37QWKDzdBbAN0qENxOoreA4EC5EUxEElhkVQSSWF/FVYFDgBJAZJ
UiI40OeAMFxIffyMXuX+D8kXfAD+bNLEuRpiVU2nBFBYQJVUzJiniW8DUI9JpvqT5t7Z9JfP
Yj/HDoHVwYv43wdA3BwZvIDLwJDm0l3V9gbl9gYB/rg/vwDzUQGSoiPaQg+9+Q7f6/oP/nYX
Arkd4iVnsaWBv5VaKn/pLUI0Nw6VFYeR4CWqUXcQrSEw8THGUUlfFEbYDMRVYFhGuCLQ6aK6
ljNkMsk7BCwbSWxKpTSuwiBB+3lXvh+z/CxdH3szr4a0SGfpFkT70VKtK5Ezq3G0hkSTThGe
ovxT9QaYou2lWMMkWyHAKGf2gn64cKWD0kt62KwKFmwkP7iA7eRrB3N6tdtaz0VLA2WEZwsN
DiVK0/We65lewX7rGl3lK4ygpq2wjZhNwwS2LzI86TMWWtxtsJFc+rQthlu9RoxXVb+RHhz/
xPwlNvs0VW9NWTGHkLl/puJTX7a/C9Qbdy9jWY/GMA2VjgYggV0QWQNWclcl3APTdFDFjSmg
kz7apYfQ6+0odv8teJDtaS7M0j03ezaVIBISyNMqhjGbF+N+KyJgOVXM0xMlhNeLiO2FgjoU
NVxEeKyYy9mbNdN/Hi8T+CpW9btaYW7jjbHCezvmSbZ+bW4hvRN/Vr3WxF7blcHOatRReznG
CKMVqWPI9v+le5MHQna6O3EhirIDqyg0D3z8Ooywzp/oKDjQQGmwgYQIpJHfoFGLoO+q9zQB
naYe9dGn03a/ffBd/9NFx+CdS+VJtYumpw85py+UYrJldvrOQFSC07PuGaiGpBqbSvRElDQM
lp5Ki2l84kScelBuYyRF6Ey9+Fby5wee5veLHrnSy330aqvxm6S6Bnh4lM3w5iA4UWo4QGHf
/oLoLuQpL9+YSH8vENF7J6uIjVw2X4hlx5vWJEhu8ge2gfkcEmUxKrY3VGKwZ2wGCxxSO+sV
8n/vVJp2yzKfwh3cUVi3gFIKKAZ02sYl3MmtWFBRPNk0QduKx+68JDxD7xYWJDt8LoHQRGf8
lcPsYa4BMfgOQLr/BMc/x3nlWK69zPu5jDXUAOTnpNFyEYeBOJVz7t+YLq46Qafq+Vik5dfp
7ww/2cHXsHofE6ksMFMFwMh+rJ9lcbIeP9+SQGK0gMVJE4VEnsUOVVHquJDCgoryYyWkh2ug
RmKjnbu5tL37qXpLZgyAm4lTXrfpRk8NYh6FblunZa8jSVYdlr2qbHJlXm2kZtn75lEq77i0
qlg0+yNPMrpMbrjPkM7CEzWGpZqdBgCb7BcvyHqu0Y6S+0iblwa4mluePDNxDr+39gVtm+Yl
a6dnFp6vdJf/soqD9yKkTCr7okL7ZQutMrstNLEnaR2/iXm3U2fgm+Wrh65aO5r9h8NF4bcV
yhpTbPsS0rLrlA5GunWZn7clb63krmUJNNtikFnBgoIDZURHyohPShMjhcTXKwjsRgnfEgNF
JKcLTERvhwqfE00l8HfQ3QI6+ikdSAOzc4Uk5wtIjo0M0k+7ZDbyGx/jpWFn8bnjlmnktOHj
PaPiOrqUFHDyW7tMAvpcV8Of6+KkRw9xeG2LPw0oNcuv8veHbg7Vwc3kd4uBr63kx48hYuP3
GvWZ7lRIalQNqslegkmU/FVgSQq/9TQB+3jSmTLqsl+QtFyTzlCEsP/gXnx24lMFxKWqa0Tw
vsy8n2FZAa2ElS5d6DHkCGy22WWTPNr2sckqtWTWSwnPj4DtKTO2Gshosjv8nFpx4l4CmZjX
yluR6qLXOiZcLn5dklVnpVR7lfArlZCiX2JI3am5tLXLJmz0JT2LJ3genbYLQYBhpNAOgrJS
H/e6iapUNlrIzsYn2onFh/Hpkel3ZUGjw+mkf40E0wVE5q8DbiJz4M35kDdT5Mhm22WNfvmO
UlAxTzKC70wKzrdYJwdTzUd+s7AllYkfepG9YXqhXt9+8n+sm/4/LAW0iN7iNzuIZY906ivf
kWZ6WGSllu3070UB3hQw2EDlcTGiknNFJCeKSE2HAJqcFyS/nTWwc9DeiYGagmcajczgmNFh
AbzSc5tMMSOqn+Utan9xH+yt/C0jddBa8tB9e+NRc9d9/F4bpMgcSuVyDRCxvE0Au6mSiZ6H
nr65V85jTPzP8hL/TfYpk6+m9k7VAVz365i0T2slUeBsIZ2yrFyuGzKbJJ37UCRH58VN2RHC
NFcU3uBF+C70xzfvY9LA3X45f57Soi2VNKpr/EtGxSbtZguRHrdYFiM4g8gMgnjo/l20gdqi
J0/EPEL7lo/aZMw9iEjoRIvvmrAKL3tDegJyy6BblfapBpANF0rZUFrZJkBbc7t9ImPlLfmM
U3uQ9Gi2BAmrIe+svRvfkPlbA8Usf6WL2lT02w+opgaBccbiA6VoJvpIzlQ7eROPmf4bE5iD
lj20/okgJagOix2dSkFfFpnY4uxpvI3SwTV4MRTQLr67zl4i6DKfdFTRdegDMPE/9cK6Hp9x
HpaoYele5UkBwsZKW/gKWhYqLDlabUoodL0ZDl0BBPk0OVBohsfy30lZoHkRrMnV9CdKSAzH
gR6ZECkoMlluZeGmnkR3O/SeCbYxBTwzq/xZGZ5IoJbA4I4pOWnOeebxw9OrhmDHGS2hE5ex
```

```
7ijxH6p27OzvwmlwcaifQVsDJUxeOL7ye7/AVXaRyJWB4jJYUopyiTuVaAgPabCKXFMKFWKk
8r6Fbh4jeJfKGFy7O/wuWhSlY6dxLuKSQ9VGNDLlZ8qIzkcAXxwlc5hquI9FZZajQxXkBoZI
elSnnkHtsJS6tdnYvi9Y8li/cqgJjr5dYnyJNyNNdzJ4TZHECiasK8ip+EbdiiJg+XH/gYvo
m9ZEeKwbRlFQxUkRysxne4hpXRWoLTjYQOK4ldQLa3FA7vIjVcz9qhCi6O30Lw/j+F7y+4Hr
WpVeu4KBHVzyp2MJAat+ROKJvo/G9dp4RkQ2leLUD0dfZF+iX9oiaLFXsG3A6za98j85VDBM
d/l8TQbWQPNRDpL7YYI3K03uK+9HA+qeFikgLGSBkx8XO4iuShGlLG7yrSQxoVHr/L7DzFjA
IH46UWQ64NlbI03MgLg7dx4fR/gR/e56p/CZNKh60DpQXmuHUlWrPuGCYllxvuJUcb9R9Tlx
7F0IqzHib7T2lcnvkNLvXWszLSzGMje0l8tx2C33W7pwVdllVVIgmAl+hiOQaqeEyz7M5HVD
cJBZbL8NKX8D30V1yafSsX+vIJDOSTGd5FZqjBxSCHSkkOlxE/fJXDA4jqnGITBQRGCvGP3Q
6PTdtKPtMwopRUssBiluHVAHGvb86USPu4lpVeYki3lnAdyjUxZfe59Glemv0g/pE9pA8JII
UwUAoDNSQlnzFax/p4DdHZBmLjtbZoS6U3HK6lDNDZwTrOzv82PPMA+J5llkPdTuKwnAGVUQ
okuq5XWJBNbtalAiQHsmgsQcKv5MXKxv7rcl8StlOvD0JPw2NHuXzs/ZzpaSR4uIHUzC0kJ6
tJH94OwzeRHd5pdDCFNlxDXMmT4ToDilOApcQPbx5lBqjE4E7S45WsjdZxabiB9dE9LA8lsz
KhXmV/CuvfMIumWbqgJnvNgiYIxZIEtPmoaTSl7jUEkisAEX2Up0yoF0D8AiSfg0ufI/bFuz
k3/m5e7N/NC4criXzyt+Hl+yB0zpltTR0pJLOJQmPAlf0zAstMW38usVN7QwRA2wpr5ZbM2+
MnWP/EBzk/Usn6cB6ZkV2khxotQBcxEyPllzAqbeIpKndtogDfZDnB+d+Cs582S2baN+eOGu
2037cAolvPaU73MAcQ90xkFUu89I+2G7ZOWZeJmZZdgR/cy5nx38F/uJnkIQXaO2CokKyClu
FGgnKtJqoJTVSRnKwl0VdtFQbpkRqWBqo4N/F2/A/fDckX0W5VyZTbilzrps8m4Ly3gaqzIC
Lw/3oLovtTuiIUD5MIr0My6LZlSyZtbkEC5hcvZUlSzxF+bJYfHf+PnBt5C6HRvTBZC4e3wM
jlcPhmGC4iPVxpFjIx3EB0pJ7oaDWRsTIi40VEx4qIjxWRGCkznqcOlxMfLCIxVsHSeB3nDz
cQHNlDamw38ZFaLkrZPTEBseeNF/6E5ntjqOmDwBHJlUIYODyQbEpQiHaK2BJqdapFbNotLf
gkvPhpgp89wA8G38HFsRpWj94GT6gJ9osbKT9tuKZK9mu2IMKBJqiypu/E3iiJeNQaNlou2v
c80X/q5vyRJtbH8yzjkz7UQHywhuRIHYmRSuKj5Vcl9FnNzCoTFh3PJ3Cknuin/gR834JUzK
7LLIdTJUbkDYAYup2PoocCiAmjnlga2+u6pzcCApTmGi7YnkwkzhD+UgerU79O4NAuEkPFZI
a3wKE80oPSms0EJxrwTlWxcqiA5FglGZVZDNSRGG/g0kgTq/d9EF76lKVmtVgzqOFl3VjKuh
ovPdflGJsNIF6udlPnFfe+g/Xr/S9L6U+qC7oSxk77EtdKJm9vMzlMaxdbpTUElDMknrufyx
//c/xj/x6mmmBkC4y+EQ5vM4BkDlUj/uYAEh6rJDhRQnCygOBkvgPJaAmpw5WkDlfbMThcwe
XxKi6NljgXeawRFKMNlrL2iT+Giw/bfiYCSMTbFTkmoklvmdVQsC4o5LIyjpfGOq0jyRFJez
qGzkLoOXj+YZYf+ihnJ27n/NSd8MQshF6EkLrVO6Wv/MwlAySkCgDjjITIbylUlZL7c/6BOp
p/7xTnjt/B6tQ2UhPSMLVunmS0kfglAiQ2ICJXG0D8c3uIPHIXxJ8lKyba/WsAousXMe0+vM
+oTseCVslS+5VilHU8RlpBY+QFlj/xF0SPvIPAUJ2lPpPDbyZleItpz9jIbkKTuwhMV3OpZ5
tpSlkWWc7k9G4uz7yV4MMHIKGgXLu0OsuuBgXy/kU6m7fIXVCuosEL0MVvfUYjx/vXCw6dLl
Bo+zaXtXOFBrZE3wiSJOpXax53LYpMNB2npt688HFin/pzVoabLDGSHruJzIjc5zIyh2o3Aa
SW4EQ5vqkifNP5BKbyCU8UEM8BZLiWzESzxWMrY2WsjpVZZTCHqmC6kMxoAWfHb4UfztoqwF
A8ixbwqVWpFzo6gOi6cgDxKjlyPF2JqHWQRyvpAKln0vp47Wf/wuf5wcRv89z078Czn4DIef
ArRPBCh/8VABEBjY8miRIqEdxlRmVZzJu5+AOWHnwna7PbiI8VkBlzmiI62kRsrIbYePlVjc
RoFYw2GUAyR8pZOnorPLvgUoK2/NW7OF2gN/t/xYJ4cYn3+gZAdK4pUlebZmWllni/DFllRC
Kw8n3WTn+A6HgzsdFdJEcryIxvsZEYrSEyloR/0lkQuRdyKZK9FWRHmgjP7GP9vt+HMw9BTF
sKXJkMywm96OcEXxci5lsazfnZnqXLzYM42l8NPLzPbNDFeRf6Xl2Hl7qzZuY5QKrmyWbr0y
+Q/eExzpl+PxfEg+NlZKdrLHhnpJHMcAMRrSuZqCM0WY5/upjl2QJ8MwWEpgqJjZeSHqkwDy
Iyupvg+C5CU8XEJgqRC8pwBRzeCqNbCY5Xc2bud2H9Sbt/TYMabHVRqo4wLSiKeRbEAOLmgX
Qfup/cJKnOMsKqIkTLrtWX4IXPwsuftyXSBC+5LKzR2GUyr9mC6EeNSTltbZNsTgvq4izRvv
QjVj7lbtaObUXZJgEkeGgXIk70WgAyVgEjDUZUAWR54S2e2xIgql5KuYvTtW0IgtwFuSqvBI
i0qLkZ3mdkPdTcOKNcu24ivQTZFcgmrEo2ePp3iY9XkhitMx86PbGF5MQWYmMVhMZ34RNAJm
tsXiAzVkFmsJzsaCOBI3cQ+NyfweVvubZDpvlEqNyfLjYXdG4Gh3MfROsc40XfawGIZEyfly
```

```
lvFiRl8DSZakLnJThU7xTXHmuWRVsmtfYtVr8zzMr9v4lvdg/B8SZSU3tgtJn0oVo3l3GoiN
BkqQeQQtZnigkKCALIaDnJkRpCY7cYQCQX6fE8A05mRHzdbgBJjxWyMvlWkt9dtNWAiiClqY
EtqLK9F0Q3j2YbINHzV96XQKI72pBV3ZdcrrWXwH8GlIVRnek9vidto4fAtbpYLh0qAhuVxU
OvRFkmWV6rcXN9hfXPvIfgwjbSE8UwXk9guN6IGpuoJj5RcVUjOV5pAMmMlpIWQE7/GsmLX7
OLyajn0WaAiB4mCDmAvLKuTEG5neEdBBYtEEvZPuhx0iq3kfOTUYDwfSKn30VqopDkWC26jt
TkzSSnthKbLCM0VYt/ahcBjaEimKgETZiONeA79g4y39TWYS9dabG0YSFUfKdgfNOw9xw4JN
C6vBxANhhu4vD6/+m7xCMNPXb/5SpruCSFZNC0bjpDMqJ9Gl3NVlZ7eYQeI/Hdfl4+8uucGb
2T5Pw7SE40kzpUQXa0DH/fNqJTJQRmSvDNluGbrSA0XWE0yoyVkRqrJDTRbO5oZmIHjG/DvS
6LXEh2fAeM7SQy2cT5U/8NVtRVft2lLY2onsxZCtHFgcRztSxod5edo9VmK2LnC0O54SlPRl
vJtFqnrhPj8rUBxE2oOR92g3Oe3Akg8qWNm/4lAp97D+GF7WQnq2G8Gd9IHb6pOqKTlVcFDo
FKgpk9XG8WKX2kEt8nf5ek7wemDVllibeQRoTwgKvSGGvRLSJ66lcHWQsTEBlEZ7OEWmvu2h
9pPljLM4m6y08Qu/dXyU7m2TXEx6tJTu4gPrWN6HQhgekKA0hoopFAfxHYPZeQGq9hfUFrox
dd5xO7rlyGKpel2lw57SYFxUn9bE7bO0Y6cXbCYRLyuv9JeLwCDvt+A6atu5CFdb+nCnJHpj
jp+JrbxXajFiFkmwld/Mzf8dzkulmefgexid3IYjJWSPzQNuKTRYSnyvBPV9sITVUSmxAAik
iNlxGdqCc6UUt2PB/GbyYlXkR8ooz4RIkpICYKyEzV8OKht8D3pyFzxs2Qp5MkE25qM4dho8
iGS6pXHY2kSGQhddQ92/2IcJ5M5N7XOe48RVtrpFi6doDEbPGit6xOv+RZkZxmMuD4zhL6zD
uJzefB2G4Yu4Pl8SrWjlQSmS4nPnVlIzVRZQBhvJaUtNMX/iOpxHmb6jKOW5m3R5GfBBBHQy
PYhkshCnrpdVcjKosjpyxiwaxpnaUnSdz3yzC5xZIOiYkaElNFxGfyCB/JI3CkGP9Mk9X6hA
cqYKoSJnaSmq5m7eR74dIjZj0sRjNGea7UjlkOd4GbgaHHV0AhhF39Xw4gpsjEOE3yqu5J5b
CebtEuUcpxpQmTZplMap2stxGnlWQkLpF+/iFevPeveGb47QSndsOUXKRtMJVvHkNsqsomTA
PTTYSn64hPVpKdKLKhpElistImCxFoJouIyw3zZILpMpit57JKWj75XyDwLbd5qZKLXqcTiZ
0DcS5m87J9KhfxQOKxlcTxCv3c5/SegJGznTG0DEJrT66xlEQWRGJj+5YKGDmYZlxgJLNmV7
T6FIGH3k78iCaVfhFGf5nl6Qp8CyWEZ0qvASCVZIbrYKLBABL/yp+g7rjatsCIEPfUn2dBHC
FkQVSyobUhTric0HnolsEDSMx2kEgQsSSn6lhuMeHK0yTueztMv5nURDnxyRoSk6XEZnYSnN
uCf26nA8jEPhKH60ECMLGV7JEqC+7xPWrmXfS5wlhdjOcimJZxVkOv/jhzJQhe4G6IvTqQOD
dSFlKiIffOS9/oorziA4UdcjWdIxYkk9GcTRSlEHVLYiWlZln/+gTPTP8WgZk9MFsGozfC7E
4k9FIgKgwMTu4jPNVIfLKK7EQBTOTZMauYZLzKKZupAmIzBUSmK4lNlpCcqoIj9eZl+KZ/EZ
5ZsK6Y4q+WO0sRS+w8Vjq6mKLROldcRd2SznG8lkA4emfScVKag9K0ne5JsZadp+RJ+NosiH
4i7NUt2VoRXamGl39VP2SLlC79M74H9hGd2goDvwoj78Z/pBL/qXzCs8XXBBAFhEzuMoBkvv
6X5hKpKMQAoopbj9l64bUB4gmJBxB9TABxu0gkCXo70Cq2Y/kZDyBvIjVZagCRFRRjg3M34p
/bbgAJTtxOdnw3jJfD2BthoYrl+z4EgadsT8Ll7BXmOsY5Qdel63lOs+X0ziuEwFuC6kovrg
4gBrLkZbfcleusLtdYbtcGeFXMqeBVrVWTMdsXPpnN2PqSUEaWRWCKwvkv4//8X+Of20tmpg
CmboLJmwwgqnaOjN+K6BGe3G0AscLSyS0wsQ3GC0iMNxKbaDQaxmbzDCCRqTqbcE3NlMNMoZ
Xs8MW/h7Xv2e9e2HStopX9SXFYSY7WyXilaxZPuSSMc8oEeDFaMxlK3/vNauo2teWH9iYRra
UYrimLJeK4qlepm9QVjnp5Zll0JrgK5z6P73QzkfGtMPjvYfRdBI9UET5ZRHS2hMR0xVUNaR
dNFqand6G6oPQ3/952JpIFMYFTJZ4eSOF6LpbsnWnMTVksCaRbn+y5il54ogU4mqXQ5JQ2rz
QmCCArT+N78L2EjpQRmakmOlNNcrqE5MwO4nM3EDl6M6HpZqITt8FMswlJcuwGEieauPTgn0
LwaaKRDD5lc93kG4spulwNPdbviY25c/SaZdqkgZQ8sGXQYrbe0Ul6msG0gycvGy/pwjedY0
pUM2JaA67da3WPrqGF6KfH9jXWKTIO4bDtcaLYTtfn9nF3v2yZvfgZeHKeS/O/juIM5gvIjr
4BpnaQnKogOtVEdHIviYm9JCbrSczuID0rEG2HiWKSE7uITzYRnykheiTfaBs8Us3KWBXRiW
qYyLe6r8SJ34LlLxKN+FHxT27OSBRw9/cqgJgVcTQSONxw0xHO/njrnTIpuw3lIVGJl+PDNR
YrOoaI6B7hRXRxlcOKbVGWUlnGAmtzjXCsGCb3kZ64g9hMA7HZSpKzVab9FUO83pE8UmHfEZ
lrwH/irfDUIeuWKIFywuTRzNMIrqAy559urmLdJFtecJoTTgmC7iORWkcVpbZ4LPUCgX8Z4f
uTv8ZLU00EjlTAfBEcuxlmbyQ5sZ3YaAPp0SY4vMOC+fhiHWcW38aTn/rvEH8SUhHXpdSzwB
7ZbNmAkVPVCd6lKMMmbaYNZ3Rebuge3TZwcicEGC9lvQkLDmy50hmvXkkf3PgSCZRgqHPcy3
```

```
qmx/Zn37mJxxuAE82SxBKCklZpxmzb7PVHOjl/5JdJzVeTmLmR+PxNxOa3E58rJjldb/Fndn
IvoYVCQovbSR0phekqmKqxuZTkbA2RuQr8i4WsHy/ENltCbKqSzFgJmdlaLszW4//a39iWlN
puIncbjre6MVl9LprwvAJzEVlBbe7WdZZS+y5oFoCuKI/NeuaaLIgxRBpG6BPh9E+/5S2VlL
wnqQvwgzn8RwWQQjIzDTabnJzSsYbUNQAkfrSC8NFSfAsNrJ78JXhyzNYY54TCBEgaT4v/jL
EyDXK7LEllBUSbCtxyBLxCeImJAljtuXfR7X+XfpHVJz/Jl49+mOen9xI6Wgzz2+HYFpjZ7v
zqsToYb4DRfDJT+QQWannplNt47KE/Bf8/u7XQlkNlgqlrlhANbWwIonxht5+eSz++2qpIGw
og7p7sS/RFHr/d/UiQ9D06Ry6Tlh97aJCMXPWfmO0t9xXvEwkyT36OCyc/RPBYI/GFbYSO30
R48SaSx7aTnVFscgeM324AWDu5ndhcmQHDgvHpCpIz9YTnalg7kc/ayTz8R4tIzAggFaRmar
g4V8rKlz4A4ccM2LonKQ9XO6fr+UlDVBCdHaB067KQOopUr/V3bQCRiJmZlyo9XaT3i0Z/IV
ZFbpccQObqYD6P9Gwt8dldJGaqScyWkzxSdtUjfrTMABKY9wDylAjax0TXIZJoCByvBZDNbo
l47ATKASr3Hc790szrZZArkT5D4uzX+N6J/86FiWYS09vJLNxIdnEr6aNlpGeqbGUjM7WglY
bTVQSPbmflvjqee+B9ZF7+uPWINV7mfkSCugFUWQyXTFBCQWlpCYGbzPQUkWcldK9OezpQXL
lx933ufrQOxgmSA5IHEr2pm77qP3lBiHD4svsa9d4JvET4iy2sCiAndxJazCO8kGcAYaYURm
83gAQWClk/uZXYXCnpmQr0ngDkAFJrlmP9xA6CR4vJHK0iOl5OarqSpWNFnP3UO0mc+bR5C8
o+ilqiwZWkb26idTNxHUAcPa7oh59l+9cOEAt0AsvFCHwAACAASURBVF4a0HNQDSDy9TU7uW
TNE/xzlbC4hdTRSuJHaokdKSF2tIjUXBmpo+VXNRJz5Q4gC3WsnnobPHXYNSfecE2cl6FGFF
csiLp7iICe9hCFPOHMvZYja+5oK9ekb5JrrtgtcxF8T/PiQ/fgn2qG6ZuIL9xA9MR2YvMVpI
7Vw9EaOFoP07eQmakjOPMGIvflc+H0W4g92g8KjqXCcppckm4AcRZDOzA5cAgoEgBPljlgGB
A8cus67R5ymkFKVOd54NHnlbJWulpAM8DbZJpnZtypV/FfQpnAHlhxIFWAHPHBE/dy+fg+Qi
d3ElwsIbJQQlru50wJjNlmbnZkoYDAyW3E54tMqXCkBFVDJI7WETlWR+B4CYHjRUSPlsJRxS
BlpKdKWDtWxEsnbmftu4fdQigVx+aUYY5GdvOeW5h77FVK5EiU+8z/ZoCIE8p7aFLFZ8Q3Th
pAVFK2Cl4M4jtWCYs3GCBiR6uIzu0kOpdH4liJA4mA8jpHYq7MCBicr2P91C/C04OWI7/iu3
vh0StcLKUynWCIWBuC5Qle7rM5cNhRfXWk7lMh0kG5WW59gf8bo6Rnb4GZm2wSdP3kTgKnik
icqAbFXEf2wuxe0keqic38HKkTb8Z/ooLgwx+G4DNYxzNxSmT0LkYa0c055DSj6zbzyuvU5K
JLY+r6DPy6fu8ynel00uLcCvUOUBCudkculnHrXTwUXQU0ch+JRuOEoxGi6q6iOjWlVlp5nN
VP/QbLxwsILFYSPVZO5uhOmCmCsVtgcg/RhZ2ET2whPr+T9FGBxAEl7gEkuFBOcLEMeQkGkK
kyMlPFBOcKeWmunkuP/A3EvmcWLCfsuv0NgdeDVw97yVkS0cVInruRn3K8RgsizrrOJwnW3E
o7/arlkj2AJNetSZdvvhxOXI+EOjJXRfjYzYTnt5GcLyZ9rPyqRvJYuQeQWnynboOne627R0
7IjWCeT29CJMmR9XAGJCeTzq/yAGJafJMF0mSe+i0ZktIxsjGf279Cu+I+/0k4/jY4VkDweA
HLp3fiv/dm4qdK4KiAcxvZI3tIzZaRnXsTyfl/R+z4Vvwf/yV45iFYWbkCDuOsxF0mxeXvjc
E5wc8hVs+lqY3usjLuE3aJuuHceXbzkhH53SmbrwqT0tq8jbkrJSDsZ3+KcPxrXg4EEpYWjW
T8ZLNhL6l0kdS3/orLp+vwHa8nNl8FitNmS2GiCaZ3kVjIJ3piK8mFPNLz+aTnC0nOl5I4Vk
d0vp7IQhXRhQrzOCw+mSknOllkivXisUouffJ9cP5+U845gOjWc5jYeJB7IXf0LInTjD9bQf
wvAEiALGvE8RkjctpL5lwzkWgfusfvZW2+0gASPlZOeK6K0PzNhBa2kFhwhBFxXvc4Vk7sSC
nh+Wr8p/bC0xluUcyGL38l4HXC4Pnem+JZ08yirATqp4FEk2WpjCtcTGsWUdYkAr7vwIPvg5
O7CJ0sY+X0DgL3XU/slGKtW0hO3E5qtpH4bDGczCMy8/+ROnG9CU30kTY4q/0qvASHSaMuwA
uk9VDXJR2UG7lrtDe0TZlK9zyA6PP6jM7JBV5Zl/uXACkolc3JDfUEU4pepl/lnzqzelNNcc
FO/bb0Q9EAnJth+aFfZvXELgeQI+UwXQGTdTBTRXp+J4njW0kubietGGUhn8R8KbH5GgNIdL
6GxLEqMrOVruTkaDnM5BM7utOs0tp9byf5WDdwecPF+tcCxBFKZ2uIaD/97xoBoh/QlHzAup
uIEcZMS/OK8VrXHDKA+I5Vw/EbiM9VWpYiOJ9HaH7btQFEBDlSQuRYNcFTzfB0K6TPmxrZUB
geWFyFqjfP8WqAmEBtAoho5lkefY/ejicypHNd9bJpsmqynDxD8rN/Dff/EqHTlayf2kbw3j
cSO7mV+Gw9sfFbSM40EZ0qgvsLWZ/8v0mduhHfXBmB0x+A5z/t9loxRomWHjhela7dDN7cfc
lF0CdeARAxW7PhueGVWogvSnlLdnNDDpyjyWsLyE8XHUejXN9oS0OnI07mlJwv+lVWHv5Dlk
7tIrpQDrMllsolkBwpJDOfRlJJjQUHElmU2EIpUVmO+ToDSvJYFdmjFdYBhvlSOLKTxNGdxE
+UEzrZSOSrfw7ZM0YH0ws/yYJ4vNywKAYIj9b/+wGiX48Ty4RYT/qNWQbnhJo56JKV+QmR+f
```

```
YCsZO7SR65gfBMuQVhMd3kwg4SC8XmZsnVev2jlFyzlIk6/MdryHzzv9uiJk3+iiAJW5TuzK
6yOJbl8LSxCb64bzNfV5ILRj9PCnPCuPlohNbn9GJqDZa/QuJz/5WlkzUkPr6D0OIbCc/dQP
ZkDSzuIT27l+RcvblWodNvImiZnRpiR29nae79pH74WQhdtM4dmpHOxCOkUgnDhG4jlFaori
DbyZ6oamlgZay9+Sa954ot5ZopFrgyg6wO9loguBJO4Yu7NKeyYQnNxMuGaH/FTMZGNpulph
GvCYrNb+bcVytHcXMlYa3I04XFniHx/S7OLNYRO14Es4lWxY2s6dFtJI9uIXnsJtKLWwwosc
WdRI4XElmsMJAkjtWQnquCuTKYK7KJR+YLyS6UkjleRvJEGZfu/RVY/66xIpFME405axzTfS
Zc4bW9uZmBUj5KUGyY5ddWENdoQUzCiGcSloI+Z0HEnlAsSkp9YONr8Ogs8ePNJKevJzpTQX
xxF4lTNQTnCzyAyM0SQF7vsZjssQoyJ2os+El/408grdlrnQCr2lMEEjDcDPoVCyJBs4kiES
znkohWr9I4r6Cthwu9Zn8qsfA/D49PEPjEW/EvbiN5agepE3nEFgvJnqgnfWS3ASR0Mg/fvV
tZOlVOaLEGZitIHLmdcw/8BclnP2kd/qzDeWydYGgdXzyCL6vlrqou04y+A4mu29DjTa/ruQ
O/nCclTLyOJBbHZAhFde+OJFpfZDeogFrdlJVlzAgkVw8Q0ctau3qxTiBqUmA9gONPHuPsqS
ZiJ/JhugkmG+FoPtn5LaSOaW7kJjILW8yKRI/nEzpeTOh4KZHFcmQ9MnPlFt9xbAcs7iS7WA
iLVbBYAcdvZv30bbD0Vbu7dCpBQp0WBPtEBtWpWkX3qxmoC/4/CRAZCi29EFnECAFTfRuCwS
Dx8GXX3PnRw8QXmkhMXE9itpTMyWbS9zYRPFpCbKHYAjbFIjKzr+coUDFfSWaxitDJEmJf/U
9uua2shGbPdbTyAjkUShp4KterYVRcYieKkhqmXTwUGKmvIOLVdDaQ6PxYABUfJr/63liaqy
B5qhTuKyZ8bCeZE5WmBcXs4IlSVk+Vs3KqjpAYfGQrHMnn/HQTa5/9Y/jBGCz/o9sLnCTBDJ
yNWR5QuUDWSVviN6kYSF0y5CuZPCgF7FkOA0cOIK57pLXv9DS9aVNtkJk8A2vfIqG9XdIqQL
xKgIg80jkxZ3k00+JPu6JOKZ3Uil/k4ifuIHpyB8zsghlNFheYlUgJJPNbPIBsJ7pYSPBEsQ
lZEiVgsnPFsLATjm8nfSKPlPFi0ovlMC93fQvhU03wo4ecxUyGScmdV0ugZBrNwUrvbeZbjq
WeuvCsiGmNK2+96tElWxBrTOilUS2TIqaF427f8PQyxB6Df2kltdBAavx6kmqleV8z3L+X4N
Ey4oslJBcLrmqkFmR6nUaJnCok8MhvQ+AJd/+qp/EQq5lm5W9+HCAyyR5IcsccRV9FKD3Nvb
X5aCDMLMOzRli7/zfwK7g8WWL3FZ8vILOQj64zdLyW9ZMNrJlqIHq8DI69CY69gfjCDpYWGr
h477sIf+0gXPoyiG4qlMzmaqOyrKvlaTpMUn2elFZJpRlS3RvaMGdBZEW0j4abuNVOssJTSt
ZUiQVtDbD+T0S+P8xL2l8ktXLlABHwFJtph9e0RSH4iFoiQG4gS4+z8vl3EjldADPVcHQXmY
UiEioxOb7NgMLCNjKLO4ieKDElFzpVQPREAZn5Emc9FreRPbGFxGmXHUwu7IGFBlh8s3khsa
fnIbPutHJc27ElkTVRvJhbcGhu6Sb+/R8FiMmV/HlZEYFDWw2oc7hptYtw+eMkv/6nZBYl2X
M96aN58EADPHALIU0aCiDH869qpBYLYEGFcaXETulk9TO/CivfdvcvcJg/YoltrC73lQDR5J
mbgZX4vwoor3j+E9DiEdw8CpWB+n5A6ls9nJ+/Df9CGdn76okc2UbmxPWkjm8lutBAaGGvAc
VcjoU3wOK/IXP852w+QDHM2ifehf/Lf0/iBydh5QnH9HQuThCYZadlOhRnBLxqlZyalA2LAa
73sJ7p7KglGFQ2HHYZvsSj8OQQlz7l+zzzwIcgffaaAGLOfjwqBFpdmCJRVQObcQudIfyP/4
nwac2GF8HxalKL5cidSp/aTnrxBgwgSvkeLyNwqoTA6RlET95MZqEA5nfAyTeTPn0Dkfu3E7
q3jMTx22H+Fli8kezJKgL/Mux6XqmJtxoWqrIjE7el9IpBJQI5UdBjBxbRUu6F3vnfaUGkJf
T9qkpVoy0pMRWvZdVHVcszn2XlsTYCj7wbTpbB9PVkj94I95fDx/cQPVZN4ngpqRMFVzXSx5
lPmjlWQvz0Di4++Fay5796RUHYvcv9CFi7HgtkJTWeiyWAKA5xhHMpUWdRRLhXj80AcoDR7U
tDxzQ5Jql+8R9Z+9yHCdx3KzxYR3zuTWRP/QLpklvMgibmbyGlWEXqxE7Sp64nffrnSJl+I4
lTNxK7txj/6VlcPvGLXLrv9wl/tRWefdAqf4mqe7sAoXuR6GkyUW6jHOlNfN6It5TCzcHFuR
pypUg9C+cWWP38B7h8qonAI38AqeevHiD6cbl8aWl4rmVVMes8qcXJcnYkC4lv/iXhU+Wkj2
6HUyU2iRo9XkH6vjxTHizkWWwROlFJ8HQ5wfvyiNy7jezxnbCoz7yJ5HlvIPjAVgL3VxLVvN
P87bBwo8nU0lfarMsMGVUmqLGGytxjqLFXPOEqEF7NSSlKDVeaYkLykzXgtXYlMZ8jnbGsog
xHPCIOnQdeIBl8maUXP8sTX/gAlx/aC6eLYe4GMnM/D/fmw8ebiS/UGkASJwq4mmEAOV4Nx4
qI33szZ+69lcTL/+BkW0DYAIhm+VUOc6UOX6DQXI3iEEdAN8MqojnC6VUv5bWhbXKkdmARQM
ICiAJmE9RlePE42X/4ANmPF8Dpf+uAcCKPrCzdXA3MVZJdLCB83434HthC7BPVhE5W2ERj/H
QhiVOFRBaKiJ9qJv3J98BjnfDsKbj4bQircYRL2arzoqINSxbqXnVpnjHR0lxaN6wgOp0iE3
yJlI8+ZZWwZxabid5fAI99ANJPXjlAFIOpEljzHxpErOOIluPo+lTgmf3ufoKnakgeeyM8kE
/8RD2h4/VkHlCM9iYEEHkAkRNVBE6XE7h/O5F7t5A9ngeL2+D0z5P4+C/g/8QWfA9UEznxKz
```

```
D/Vpi/CY6Xcf5zH3HN4DLnXZeS/5+99wCvK7vuey+HBexEryQI9jKcrhLLlhRFUYrlOLFjRY
mduDyX5ljOs2PHzy+2FSdOHMe27Eh2JCeO7TiSZoYVBAESIFhBdICoLGCvYEXH7fee8sv33/
scEBwNZ4ag9OaT7MtvcV/ce8655+y9/qvttdfWPimKEmn9TkZCxGrSQC4GvrICFyFI3hUgQc
/qgUKNoFZkR/3x3p/5zp6nxFgJEa2nMWkG3g3w+0kPH+Rq62/Q9+bL3Nu7GjQgXyvA+4vl8G
YF7H2FzNd2PBNAHGkQOcL/ew3ZXWUMv/4iXN4T2OjGQw8yYZXwJ6n7iOctQEJwSAmGAJEm0b
O9Ez0CiNZsG7lulJBWil2G/t9hatcLUFNBamcpmTdWY8D8lXXwlxvwv7aO6d2rGd9bSaz6BS
bf2GzylvzqCqguwH9zqZWuezbh7PsY3pGfgP4/hLuHIXnZaESbfBgkmYWKZbbVYMZPs3j3IN
5L+vrr3G/+Ja7v/hRX/mIb6VrllS+aiolzdtJNBFAMoHX00hvTJuImD2pC/aF0mL4vMfn6R0
h+dSXUFJHauY2YUnH2VuK/XmCjUl/dgNUgG4jvKiels8Q45Xy9BH9XLpm9uUxXlzGldwupNz
6K/7W/AV8thTe2cefAL8Jkn82yVgV4E8lThzikJLUDs0qjKWGouZ/3Cg6dGlFKulmPrQ7VQ4
V8IR4wyzoVU5ddp/yfADPqGMMWWjgfOMNG2GoBzg0YP0i29xe5Vf0Sk3VFZGqWwe4iUDRj52
rYuQHvzZdxRbu24O5eP2ealk9TvRn2V5KQ3d/4y5A6C9nbSqNDm6rIbZVEleOFzyAnUo8a4j
lsTY+GBz2xtUfpfHlbijKNKe6uPsjEYLIbzvxH4oe/l7Fdm4ntrSSzbzXe3rW4O9eTfXMLqV
3Pk9i9nWTlRpLVlST3lhDfk0ty7zKylctwDuTilJYQ2l/JWPU27u77EMOlf5/Rpp8ideY/wJ
3/CeMHIN4N0SsQGzPBEeOiyOSTuZHthZt/SLL9hxmv/zAThl7l9u5XuPSlDxNvlsKtertPn3
2cp/9fnRbyCbIcVJrVztZLYpsgwnAz0T0/QqZ6De6BxWRqVxPbWYH7xhp4vQp2rjf84O7ahL
N7A+7uKkPsqsLQ7tW4elbj7Kkis2cD7s7n8XZugzcrcXa+yoM9Pws3DgfFHMaIE2XKi5NUiX
YDYPGt5sBsJFOC0pimM/f+zo8dUYRnBiB62BAkRgJoMYlleuxcQdghYg2hNGvX6MjgNQDRhu
7tOKd/k9G672GqutyCozoHdufDLgFkA7z5PO7OF3F37kAdMleAqENjr6/F27ce9pfh7NxItu
5HYKIR3CtkmWQKz6h7yRITZVOniYLGasp37qQnfasu0nUFQJXFlDal/TAJY2lw7U94eOQHGa
3/AGP7q5jcW0Jq7xq8vZtgz0uw+wXYtwl33xqye8rJ7Com9WYuyTdXkti5koTs75oy4nvLGN
tZxv3Xl3D/jY2M73uFaP3HSDV+H+7Rn8A98StkW36fdMefEe38KhMtX2Hi5K8TO/rPyTZ+GK
++Cu9gCfGaNdza9SpXq/8JsbN/CpnLgTnypCd8D58bgOjBh4F7xoQROyjEapK0bvUxtednSO
5fS6ZuIdmDhcR2FeMKHPJLJSx3rZ8TJXe9xujuH4aLX7fzX0wHAtEnrXQgSSyFMs2CMIX5Y0
E0U3wdKAMB5Rlej4d5jRgN5gTCuQFdaDaZC4ZXDxlCnSHtcRku/leGD34vt9/YTqx6E44xr4
pgT67VIm9uw3vzRZxdz+Ps3mKkhrNnPXOi3RvI7N5GRtJmf4H5reje74NbO8G5Y6I+kvCKqs
xokFByCCDmed+hd97tq/BainKmPDzVtVXXCDlmYdZNkre+yoPun+PSgRe5truA2IEiOFQJ+7
QKcTl8bQN8bRN8XZNfWyxpnuSrG+z3e7fDni2wax3+m3Luy3Dk4O4rxzlQydiulYzs38BI/Y
uMHHmVh0df4cHhrYzUljCxbxmxNyL4e5ea4yd27uBuzQ8ycfr38cdslXSL6Hd70Hf4Xs9rQs
lWg0hzPAKIA7d6mNr706RqlpOpW0TmUCHx3SV4byg/7dkAkt69g5E9fwuv9z9C9Kzpc5Ut0+
/LjEopcBIKRFum26xR8iXcQ0Vg+PnJzxeZYX7zRpe2mmHGHwnBYa4RRnoeuSjupBbxqlTMLb
zLf8ndY/+Y4dpXmTrwMun9L+Lu3o6/ewPsLYU95fi7tuLu3kF2zlYy+zbi7t0wd9oj0+pVUm
+sxd+/HKe6kLGd3wln/hhSDw2zChgCiRk0PUPIwCbi8+SOeU/f6Fq6sCjj4afTpNJZEloKar
aukFq5Rfx+Dfd7/jW3Gz/JWNlWsjUVsEtOZoExFYxW3bUZdj0POl+ANxTnfwG+qkVmz5tiD3
x9E7y5DnZXwp4y2FcKlSVk9xeTqCliqjaPsbqlTNTnEDu8AKdxHhyeD/ueg+oiMw5Te/8uif
bfgtFmM4v+XqI479oPBiD6TzPzY2/RIA7c7mZq708agGTrlpA5VPw4QHYpJL5hTpTZu5XJ/a
8wffJHYVg75Gr9kd3xV0ZVLJMwOLAYkJYL54qCWgUCybsCRAfp+YxtoAtYBycEiMn7MULRLu
SxSxxtRXdjxEdVCmIYRvfzoOknuLhnB6NlW3GPvIpb+xLerpfwd76Av3uz6QR37zoyletIVW
80lN0ngKybGwkgNR+0E0v7c0jvX8bIzhdItfwaTN6dCecKJNZJD8KiQce8S9+8K2/Y5w9imh
KdXoqEN8GoO8KIM82Yp0rAQmIC4lfg+k7iLf83o9UvMbVzMfHdEZyGlWQO5ZOpKzLmlHyV7M
4qk2vkfG0L3td3wNdfwn/dkv5Of3ULib/cQPx/rYc9r8Gul3B2bia+s5LpXcXE9xXg7s+D/d
Lcq6H6Faj/AWj7DbhVD9mHZngl0uqKZ3qpE0lHWic9lCAS3CbfY7iV6L4fJlOzCad2BdlDZS
R2l+FrScCbmwxPeHvWMRcSH8Xrt/Cw9ruh5/dg+oKZblAQS+WKNNFqxl43Ewh/4ldbnjf3+C
5MEDGDrPNNjdhAe+hPhbilLmKmGohmozWfoOWfds2RERfJcXjQSKbvl7hT/z3c2rve2NvZui
0QAuTNl/B3voK7ZyvZ6jWkasqJ799gHNTMMwFkI96+l40GcaoXk6pZzETlZkYbfgLuXLRYN4
```

```
l0EixKDle0JeAKo0H0xzO8dLrsN8kVk+ikvpEdPMUUUSbNAiXbl7avJuBBA5z5fxk/9ho39y
5j5GABD2vzeLg/l4f78pjYV0S8uoJMzQao3YovU2zvVti7w5AmIOWoOm9uxlM0aOfL8OYOM7
ckXyZRXczk3gLGdxUz+vpaxt/4AG79j0L/H8Gdk6AFXlm7F59qQH3zACJPzFaqFytagCThbh
PR6h8xAHHrVpE9VE5sT5kJVjwrQLL7lpKuX8+DvVtIHFZ29Fdh/ApMJMzWcvIPZT2ofJMBiH
wSw7wWIIGcfEcGiFgnRikLIhtINxGeIIIoiaDQWMasGNTKQaVsWDVmBvlhO5z+PA8PfZwHNZ
uYrNlIbH8VmeqlULcdf8/zsOtV/F0fxNnzItmaNaRqy4gfWE+iZiPZ/etxqqvmSBvJ7txKdv
c6MvuWkqhZSrxhHXdqvheuttjZKj2AAYPmQSatsDMC4R375Tl+6eE7SbxsAleFkM3MvECiPl
L8TJVGAuFlR8ok8ZHpxbn9RUZO/xhjJ/8eE43fxcO6l7i/bxOjlWuJla3Bb6yE42tIV6/A2b
fMkLdvJX5loBkOlMEBTaSthFlL8WqeI1kfYeJohAfHl3P/2HpuH/448bZfhbO74e5lul+3wK
x5PUWkg4Vj7/Fh3/Yw48eZb3Rhm4elrBYjiBTRu9dAouYzFiClBTjlFcT3lpvADLs2wt7l+P
uq5kTim/SBDUzv20h0/wtwSnuulMDEPTN3qTuysA3sYIFDKiUYFv35LgqEiPHwTY2lIIw7S2
uE6tKGxkIG088Gxyrdu//3SDR8L8NvbuDhntWk6jbil2+GgxvgwEb8vZsMSLzdL+PsfYF0TR
WJ2jXEa6sMZfdX4eyvtAB52rZ6vdUee6pI7c0nXrOC9LEl3K7+EFzYA9H4jEulaTWl/tkZlP
fQM2/LDo9/aFfraalQnKnsJAnHrs03PyqzypTCCUJd6rZQQWu2lxtAD9z8czj728Ta/hX3Gz
7D7eq/yZ3qVxk9sI3pQ+tIHV5Dur6EVF0+yZpckjX5ZA8U49aVwcElUF2Jt7+CWG0po42rGW
nZzlj/p5i69C9JXP0CRLshqRBwIE7VamCVGiSz/BlfhsH0nzFd5BrbyxspqjX892tIHPh+sg
c249YWBQApMwAxvocBSGUAkKdrFblM7Nlorh3bnUviwA4499swNWhuQjETC5AgvPgNAJGn8s
5WRMSIEhMKCyWtlUIWHCrXrIJpGVw/Y2qymlTd7Ah+8gZMH+b+wU+QqNlMsuZF0vt34NSsw6
2pxN+/FqrXQfVGQl7NetyazWRqXiBl4HmSB6tIHlqDW7sW74Co6mlbt2atud7bt2vN6jlqN+
HUrCVzsIL08RLuHtjCaOMvwd0ztockQJgmyR0eRm+RNqkINrbwLPyh/g75TuVJVWjBqi3lJg
WaI5BYxsWzCtrwpz42zGX2c7wNYwNw6yjZ83/B9On/xFjL5xg98c+4l/ApHjR8jJGGv8FYww
cZa3iNcVH9Bxk/9DlEG/4xsaM/Q6Ll10gP/BHOlWq7tcLUdUjIaQ2KgolHdE+iQIOa33+WDt
ClpJ3N9SQ0fbPxpX7CAuQBXPwTotWfNP4R9atNQCG2rwRqt8C+DbB/PdSshZqqp2/3byaz60
VczYMdWkaqeinjlR+Bil82m+RIu92Pxon6Lqr270xmbPqBKS3rEc3Kong3gISdFhhkOkErD2
xVjRgJN246wfrwStWYAO8q7tgBJgd/gYn6DWTqCnD2b8WpfhF3/6YAHAJIJdSsgQMVeLUVZG
vXk6l5idSBl0geXEvqUBlu7ZoAGCFQHm8Ftnei7G5l9nrky6TqqsicLOZhXQXjDZ+BK/tgUq
vbNGJavz3CpDOMlk+LO73A/Jorj6jrHgHEbjFtfkyTqmLIwC80bWDs6yfDuROda8ZH32mFT3
oSErfsjrfRdpg+Qvbqn+Fc+zL+tS/CtS/Atd+Ha79r6eofwd39JgeMBwMwph2SMvam9NshKA
JA6Oc0vtKidjZZUcm5v3SuwZveCCkz/l7wXM4wXPg9YjUfNRVdLEAKie8vhroNVoDOAEMgeU
ravxktmzA+Wl0OlD5H4kAVbudPwugpM5OvqKI2/jGRaPGBOt8YQFlSbuI9AEQjpo40QFJ+ih
xxrfO9h8MYWSWhqbNlQ0tAph7AZD0T5/8frhl+nujhlbgHF+PuX4e7bwf+vu34lQJJJf6BEq
grwj9YhFe3Gq92A86BHWRrt5M9WIlzsNx+iNxyWgAAIABJREFUXleJ9wRyDqzmSeTWrMZTuL
NmrVGlybqNOKdKGT9cyOjBD+FlfR7GtB2b/CsFGSZJ8ZCoP4JWnRkp9wwcolOtpp2xWiy36Q
sxvbhnFpMaHAS40TgJINNmpl+bwqiCvNXWnjbpEYWJgEpTl6bJToEzYZfpOqopO4HvTpFlEm
ZDmWzaMRntM4drvPSjgeYSKFRjK8U0CcZNalcizg0keszw8ULNFMpbA5jsFTjz6yRrP0hS0b
RDFcRri+xc0EFN7lbBgSr82so5ETVKen0Z9m2DmuVwKIfsoVwmGl4mPvAbJsUGdlZIVzyuLc
udJI6XfU8r8iNGus0ARPlKIzgMGzLLddQLAocy47Sr6EQP8b7Pc+Poa9xtzCdleAkcXAZyGv
fLnHoeb/8WozG8ukL8Q3kBQKrwajcFtM4C4z0ARBrmSSStRE2ZAWNszwYSBylAoo25jNRUMn
XkH8GNg5AYMdJEjnMKpSOMkxBoxDx6vmd46XQTCg9MDRPnEEOK9JkJENif0s/ZteTKIrabKk
SJMUXMGIBy6xUWNqFJreEIuU9t+F4caC9kri0MKJCmIVKuqOZfdJ7Bls4xzyetoXI/KvIQI8
UkCSMs5DPpgnN76dLKhlYxa7PMOdAgujlT8Do5iN/zOdIHXzKpNn5dGfHaAkPGfzqwZk7ACA
HFgXVwQDlvyu3Kx9+/FL9+KWMlhYwc/bv4V/8bZIZnKQBFZWM4fswsNdbYvNsrYkSg6SOpW0
VfBJAHRnuYkpZCnVFNHkzfhPN/wp2Df4fhA4VkmwvIHsrHqyvGq8vHqy3Fq9mIe2Ajbl0x7s
Fc/Pp8/EPlJmTJgWlQVwEHpWLLg/dr8A8+mdzaCp5EBiClmiRcS6x6PbH6KtInS4keWcF47U
pGal+CPjltKlOpNAPVhRKDxNEaZpM79W499I7fiwMlOZTNaplY2lYRIpn+0i6PAhzKeQu9dH
```

```
W4zpVWGwm09YiJss3YR8pk0DWFlpCkyeV5aofXjLaHU4DACxxRa9EZ5jSolY3rBFWdUXBC46
qJvOngnuzl3wOPPLEHZK4pwqm0DpNmMwMQDy8zZdKO0m0/TKZ+G87+CqgtJXEw35B3sAxqy2
2gQcGGuVDdWji0CQ5uxtldQWpXHv6hXBIHcxmv28B002fhYYOZ95GNZarSe6rTr37ROAWq9Y
lPqFlulaPGvLILi6yJpTmDQLJIa5g9z4fh0k4mjv4YD/dvJ3E4F5oEDqnIdVZLHMpDD+4cKs
OpV0ivALe+HP+gkL4NDmyGulI4mA9lAUgOVrwjQIxpJvPsbUiShAPrze/L6Y83VBI/WkyscQ
Wx+kVMlFUwffKfw/B+8GQ6iv0yaJcpMYZjp7vfoXve7StltFIVVSvLmlSKlivmod+ypL4cty
WQjFEVLJZWv2vQXG3lNULWH8fxdJlgIZRSVVRqXNmgIam0p8iohwwyxaQTQs2T9rWKzsNxHL
OXeNrXdpcqxyRgKEQvsMjcnLm5Z9KgFiBBxUbFdmcAksVT5GzsKNMnvx+ncaPxQ2Vyi3mThl
aZ3DAOCiASmHMFSAXUr4H6dXj7NpLctcYIaw4Xkj5YxKiiWmd/C8aVOTBNwneY9jX6Fhi++P
pdJETESixZw7bOklSmBtYkmylOrml8YsMwXMl0288b2z7VsA4aC6FOam0HXs2LxodW6nNxGu
eTPTKfbEMe2UOrceu24tdug9qNIMRLatQWQ22eBcrBclxFs55AzsHVPIkETmoEuklkG9cSPl
LKlKE84oeX4xxdQOpIIQ/qP0ri7Bcg8cDIApkk0pMaXLNB57t00DtDRKiQJpBmUB+K9F5kxH
3wffDeJNAFzKnTgnXT0jTqcwk0Mbg2qElnYmTMfoCPrqfUEOEqBJ/RFvoj9CNVFE9FGHx5M3
ESZvmrZmVmfFOrNnQ74a3q/Dm+LEDSpPwk2cCn0z2ZxVyJh/j3DzF65JM4R6qMteDVFBpwpB
pW4R4shEMSls8AEAMwCdtSOLQDr+5F3AProaES6guJ7i8n3vSPyF79Y9zUJZR4orCvFLIBhg
SQlQ5P7IGIzaGRhLOTPDpZp5nl3Or4lBjc3k+i++eZOPZxEkd34BxbS7ZhJXHlPx3YbgDilF
XhlOeRPTLPkgBysAq3bpsh/+BajMSQ3lBbasBlNIkBSEUAkG9sLTjKA5A83kqrULsODm3AOl
ZF8shqJmuKiB/Kh5OL8JpWcL9uPZMdPwsT/QblinzLsJEv4Dlm2AfqVoB5K5lONGp2piPDQ+
wH4jJBT9pEqyHU6u9gSwIdPDv5UlwUAEQ8rbQIs2RZZpmksOeazUMdXwaU9vASqWio/tnx0T
jJjDNRGXn8Kd9qGhNREUDSZgY5hK05XvehC+iPbwJALJ95qJBENp0x2knmi5ZQuLHbcGcfIw
0fgSMSjJX4e0vJlJWQqi8kVZ9PtrHImN/+IQuSp20NPxlYhX+gAI6+CMc/aMx7o5GOFuDUrW
KiYQuJgZ/FGzthxkQjpa4y5rUZUvlnlwBJAIVkgaP9QdwL4N03kkn9LItapLlnTOdPNOIP/h
zRxhlm0so7+SLJk5t5cHIZd48uJN24Gk8MWl+JllCGd7jAkt7rs0PrLdULlastqUPqywKqwG
uowKtfPYe2DBoKoKEEt76KbP0WMod2kGlYT/bYQjInIsRPFDJxXFt3/RkkboIW9gdCdzKrFG
hxipg5ULe2v6ykNe8FHKlSda3pUYMfKylDDa3P9b06LaTg+BBJT2hnO/LhIRZ04Y2Eg/Z4oq
h+X8eb/3SoOXn2OXYBmO5CNHP8o0NmfWh/8an/l2+q+0yKm2z7MaLcI+3fhvRlHrT/LtOHP2
yiVlRvsNGmA9vM+ERPFDB9MpdsY4kZd/GG+OBpWuor4FCZvf6hqhleM583LsM9ugSvpYzx4x
/GH/oSxIcgOYyTiptKPNFE0G3BpkrihTAlySwBIU3ELJFlY6YCRMyz4FDI0Umrds8AXPk8tH
836YNL4NBiOLKZ5PGtPGjK5cGp5WSOluI3rJk7HV6NPldqLIYjK/AbC/AaNuIdegHn0Es4DR
txji0ne2IBiWPLiB/fQbrj5+DmPltWxlUMHKKeCglI6UokBAAJOT/g70c5ajJlJH5tp4aHGS
Z9as76DjlBYAtki8cUSYZJowW3Kit0nJHWf0288RU4pIRJpZVshwPbcQ5vYPJUIZPNeZZ/5j
j+NJRDQymmrV+L3yBSmk4x/tHleMeW4p0sJN30Aqn2H4crfwqxQcjGSacgqjVVZjwVvbVmaT
j/p4CKQBIxYy7fL6mCW5YFHH8YEtUw/pskWz9K9nAZlETMRIxlqDYYcyZlfAlO4zOAQw9jOq
cc//AcSB0RAqR+qwWIwCGtlliKc6SYeH0BiRObud/wYaZO/yKkZGrFSCWSJnPK5mfJeQ0UxG
MiV3+IA2ymqhWXdnLNdux3CKPP9TFkqpj6COqTSZL+/UCI3CF2/Q8Ybf00ySNbLUBqloGobi
lOYznRk8VMNRWSlYCdy9gfLofDxXA4DznlHLIpTv7hUrwjefhH8gwPZA9vxml+jQdHNnO/9R
+ZbauldimbcEwdcw28xlK+t5YxK/iufwqLi8x6EBN5kWUQ6uJUL/7Ir+Fe/hST9ZW4tQVQux
Rqta5Ys+Mb4dB6aNxgESsmD5n9ads5do7pVAOQPPzGQryGDTgNmwlgDUAaluEcXke8roTEkU
ruH67iYcvfgzt/AdnL4GgpsawMJRfKlLKZy+E8hkWMNIY0jAASgkQdJdH5ly8xlqpqJh2bnJ
lOaxJTxfmGGOn/UUabt5I5oiiTQvqiUuM8e0cKiZ8oJnailOyROQjGkGdkQTSugsY8oz28hn
V4jeV44gsB5fBa3MPbcI5v537jSm4c3Ub6whdtpRhV59dgy4IOMtY1stYHFlgUrPKIRLlPQk
vtLYxAD/mwlvTQDzPduo2E0tYP7oD656FuG+zfDvuVyr4eDlaCNMiRCvy5tjr3SNmcicYyOF
KCf6QE92gJ7pEluI3r8Ru24TdsIXmwlOThIlKtFUy2v8h474/C2C6zTUJoVdn5ChsMmqlAbM
0s63Q+8s6kUWRu6ci/2kDRVECcaSY9lXlUt2gtfBRGD/Kg5UNMnVqFe7QIDhdBQyE05MORlf
jHckkfLzOJpe4zjL/GnSOFcCQXjq3APlqIf3ijGXsatsDh9XBsDenGQmInyhk9uZV77Z+G+l
```

```
8CX0JSmylZB0lNCBALEssPkYdcYJrbJlpialqNNeFc+vdEOz7CZGMp2cMb4ejLcOQDcPAFqN
0OhzbbUJom/BpD5g4Z/Wnb8Py5tBVwuMoAlD2+EufECpxjZThHlhvJIYBogU5aczany0h2r+
P28RdJX/lVu5FmQuvtbUc8Aomteh7+bftPjrK80TCEIa2iCNU3IV/82lgRaeJliklGmcTsMS
QOmxyAi/+esRNVJE8shKO50JgPjSuNv2gY+ViuHaejFbhH5zLu4TkCSBEcXYl/IgfxgNu4Eb
fhRfz6HdCwEY7bbA+/ZS2pjm3cPL6O6NA/hWSdWQVrXIxZ4XaBwwQFA20SGeE8ca7hch9nop
n4lV9lsvtjTDVVkjieS+p4Hl5TKeljVcTql5I6vBZOrYNTRbhHlsOxYjhaCkfL59jq3DnSkU
o4vBWOrMM7uRS3aRHuiSLco2vxjmyynzeU4B/Nh5Z8Uq0FPDhRznjH98LlP4eoainZ2sICgt
3yTHudaDsHKlFCoEhaKnxpTS2p578GiGbRJ5hmTCaqOEuK9eGfE2/7hNEQcpQ5mmfp+HI4sR
KOFbwNz8xx/MV75nqr8E6swjlRgHusEq9RAvwFOLwJGpfhHlsEp8rJNK/l4ckiRjtewb/xKy
YbnawmZAMtEqxAFEBCikS5RUZrE9whssP/i8nuTzF2tJh4YwE0lxA/lkPs5BLGGldy9+AqJo
8V4Xeuga5csicW4R2f9cAhWN5rOldghOeFAGncYDuoaRFOkzRJEd7RlSBqLIKThWQOLyJ+dA
mpljU8PPYSMc2NxNpmOkeTXjaLQLPOsRllGwJFALKhP4FExQCCGXGZpn9FX7LTJ0iaYnFW7I
7BvX/DpMyeYyU2BG8kfAHeiZW4J1fhHRMYVsORUjgSCte5AqQIjq2C46vwjhfhHi+ldGwl3p
EN0LgWrYOnKRfv6ApSRleRbClnpKmKye6PwvgfgDKOBRAZA8EiwVCLCCSRTDZqioulbuxntO
VzJJtfhpMlODUqCFBEtG4BY4efY+T4Ukab8phoKSTenofTtRg6l+El5eKfKMY/XvK2LSdKzf
XettV3z0LHK+HIFji6BY5X4jcV4LUuNERTnr22BuJoARxfin8yl8zJKmInX2Sy6XsYPqnatN
fMwiY/JSfclkBS/ELLUdVRIoHEmlryOYI5Eb0VfYcDJJlM4rrGw5gRA0plUbE59Uk438h0FG
7W4PR+F7GGCN6BFXB0HWgW/dhqOzanivCPr4Nj66zFIRAdfxYeKIETeXCyAP9EFd7JKmPteK
fyMON/UsK7HATKk0vhlEqyJzcRP7WFyY4S7rV/GGInbE2xVAIv7ZqJWo23lOG0rlysWILszS
7GOn6V0caPkT2sybw8qF8LTdugZQ20l5LtLCPRUUa8o5hE6yrSzYtwTi0OAFJowXHiGluBjZ
OS4m/X6rOA3glIOubtvj8hLbEBjm6CYxvwmkpw2ubjtEfwmxcbyYGcRCPFVhnV7h3bSPbEVu
LN6xhp+Qjc/e8Q7wHtXJuJ4SY8m2wYzLiGGkSmlmETASLAyaPZtxne+Y57klUF+VkvgSOVSp
HJZMxkm9GsWpE60YV76TdJdlbhHo9Ag6JLSknaEoxNqVmK4J3YYAEipjlWBCeeYfwNXxSYa4
QA0XIHp2UlfstCaM6xADy22gLpRAH+8U2kTlUR75rHWOdquP3FIMviAXhR9LwSjgKIwB9h/C
7x7v/O2PHvMrOOnFA4Ls+mghypgJPr4dQGaN0E7evw2yugvQRayuCUkFlkpIPfNJdW54RUEr
x/760BzbEqK6mOS3qU4LTm4LRF8Noi0LwQjss5FEgqbMqDgHSykmzLSqbaS7nb8Qmc2/8Zkn
2QiJuisr7WUwWJd9bcUgasXU6qyKARnValfMdrkFnYMFojnU4jraLWJHuqH5K38O/9IdG+v0
28qQhOSVrLzKqAxu2g9KSmSpymciPlpe2NVDcAeYbxF0CkgY6X4mvsT5WSPbXaZJmLB/zWCD
TJTy4H45NughOr8ZpXku6MMNWZS7Tvx+HOl8HrM9u5+dkJ4umMqYov6yHC6HmiLb/JyPHV0L
MYWjT5kgtHFkDjAlAlioPFpOvLSB8pIXN8hbH3OVYGRyugqdR2hjpkTlQMp+ZI+u3ja+DYWj
hRgd9cjNu6Cqd9MW5HBNrmQdMSOFYIjZq3kaapNOrXbltAqnMxd04UMNbld+HG/4DoDSs2BB
Azy6plG9q9V6T3wfJSM2nyV8PECgGi7dlkakmDhCaWieIlJ0lKefLSZxlrryB6LA9OFUCr+K
ECjm6DYy/gN63DbS6DpjI4KbOq2FLTHMdePKPxPlFh6VQxbnMxmZZSMq35OO0L8NvnQctiOK
mMi+ct6bdbF+F2Rkh0LmOi9WOkz/0bmHzdlghQqkx6ykasDUCifcT6/hl3W9cy0TmPZPNyaF
kCp8VgEWhZhH8qzyDUbcrHPbUQ9+RCM/egORBOlUFLwZOpOR/eieYEqgCM6tzjZdb8kt3ZnI
fTIhVbGHRQBP/UAjiuUKMiGpIgCj4shrbncDqeI9a8gPGmtaS7fgzu7oLkHelnYBas+07UOO
42/UCTR3Yy0ZhYf0V8kLCwtVqB5LGXKqmPt8Dl32SyZzvjrYtJyKIwTFgIHRqbdXBiI37TGi
PANEacWgUnCi3jPuv4Gx9WwKvAbS4l07aCdMcy0vKTW/OgdbH9vaObQdSkiOYi/LbnSLevIN
Ymf+Rv4lxWTed9wEV8Z8xsnaDxjpC+Cnd3M33lJ5m88Alig99Dov9lollFjJzIIdO5AbdtB2
7LDvzW9fgdi6H9OWjRepBiKykkLZ5E7wQefddcNHc6Jd9GpOvkGpC6LavJtpbjti/Ha8+xdu
jJXDhaaemkgCSALIT2+dC6ALelkHTLh/H6fgZG3gDnnNlsJjl536TFS3M8CveGVpU+lX0upH
znvkKN8VZH3Txx5oZZJ+/2fpqp9nyi7QtJN5eQPVlmwdBRbiKhSHM0yyzPh9YlVgCfyoemkr
mPvfjmlPwXgbAcmlbjtZSQ6VxKunM5mbYysqll0Cq+yIWT5ZbEJ6lLoXUlbls+XncZE80bmO
z6Xrj/e+B2KlZtl9QYJl3FAlLjuNNncSaPQ3QPzv0vMXH+3zLe86+Y6vgcsbafJtn8z0m3fJ
xMlwoy3REyHRGy0jDtK6BVKvUJpE55Ij0jQKSZZO/KzhRYTpXjtpTbzmkrwW0rwD+l0n5/cj
```

```
mcXAUGIOqkJQFp0EqMepY0yV74LEx+GZIDZKeUOvFoD0aBRGkpZnvrmXV8Asp37is0q6RBwp
c+m5qaInG/iWj3D+C0VxqJnWpbbswbmTlO8xoj0WlbDm3q65XQovcSSvOtQNOYPZOAVPBGjC
8AFuCl5ZLpyCfZWUC6XQCRvlEZgFPAXGjvwQjmcgMoTq8glZ7PWPsHmTr3L2F8flCIW7V9Ia
IlIlpPnFbRQVMoSTVWr4FzEbyLMNIM9w7BzTfg8m+RufgPiJl7kYnezYx3bCTd+jJOy6s4LS
/itm7DbduI37YaX458Rz62g4TYgNRJLeqs3MD0eitIZLIV4bWtxGtbjteabx7EPuhqa8pJKr
QKeJIOS+CUACJptBq3tYxsW4mRHnrvyR4WSKQlTikmLpNPYNY9rIQuResK8ZoWku5YQqy3it
jQ98OD/2krhodhLNXaDXK3lOCoORNVC58Jos947gJMSKEdFrLWW9vw+0fM99YjvvFvHTv7+s
ERb7F+vvG8Wcfp2JCMFgwnPYPU/mBdhCZHwyWllgOTxozhJm8yMdzC5MUv8PDoFtzWZdBZQK
Ytj3RrLm5HCdmWSpInCvDalwSaejm0CiALLZnxl9iHJLBIy4iK8FpFBUbKu225hvnFC9acl7
EKEgXgMD7PcmMlCCDpjiIyhgfK8Zqr7LHmd2X5iBdK4FQVXrP4NA+vp5ypzud50PV9pG79Pt
BrsibcdErp7rM6y3SyOkFBrqQtja9dUclS0HG0nx3eADgdkGmDlAmmu3+HVOevkez4HIn2Hy
ba/kmibS8Saysn0bYMvzcHTs+DLmkbRZbmwcn5cCwHjkj654LUrbRBSwlKCXAUVj49n2xPhG
RbDpm2cpyWl8k0bSPVtIx0i663AnpLwNiZYvhi/NZSQl5bMV6bfe+3hWZgqOH0d/CZAhIKNh
zXdZZC53y8vgWk+0pJnP4E453K/r0KsYytjKCZI8OeWr6kPKQkqewkjjONp6WyBjBxfD+K40
Uxc0yGmR9lAAdsOhMrDhfoaKJy9lA89kf4hTHnwvFREFKMHVxKuAmPelIb4nF2kCGtPdqDfd
qVm5SeJp0cIZoeYcofNxOBYyZbz8wMmN3DmK4hdeHfMNr0UTJtimzm4bcJDCEVojHw2wrx2l
fhy8poywtoFbSJgvFoXmV8RxMRba7Ab60w45/pLCTVvYrk6aUkexaTOr2ITPdSnI5VZmy9lg
34LeusCdW2xJjTbscSAxIDKAnWlgBIBnShQC2E5lJoroDmdbintnGvbjX3jlYy2vMhRs78E+
LDvwsZlTEYeRuAPKFzTeKjQp8qjZX2SKdc/MQUZJWuMQSJdhivgTtfxr/6a2Qv/ATZc/+QB0
3PM9a0jqlTFSSay0g3y4kuNSFlMTRdq/E6y030IXGqmERTKanWPLI9C/AGFpLpWoXftQE6Po
Db+hLptkJSbYtwupbC6VVmYASCuRD6fWO/llrV27nQBid6l+L2rCfd9Sm48QV4cBQS9yFjCi
qZ6Q8rQlwyaKGQdiHRbubSKAk8L2FLkqrqvUkPDpg3QIedtbflqbRG3G4oGWiRsP9DZp7dmu
/0gUAilfaWRV76Ssc8FkV4y0rJ8HohSJJBKSEJQbN92SiOPxqUBkrywFHdSNcsozXrPKL7cK
/8IomeD5HUOLYXBmNQ8NStAY20T+uKwDKwgi3bUUyqq4Dk6RUGHMmeBaR6FpDpXozTucIAT/
NzApPAkOlYOousJnHb5WMoB68YTpdCt3itFDpk2ZSYuT3alkLXazjN20l2ljPds5oH3duZOP
9ZmPg6+L08vj/II/H2De9kXphCG9peN+uS0QaJaS3JUpEBbUg/aXdllTpqLdNNaOuvq7ao2d
lquPUVvKufJzX0fxEd/D7G+j7CWM8Oome2MN27jrG29Yw2lTHZstSEX+nLh4FKaFday0Y4vR
m6NuO0bTKZuYnOJWROL8KXGddeNDdqk6ottoMjqdaeCxlF0FmBl72WdO9Wov0fxLn+4zD9v8
A7axfOyMIxeTvSIirZM2FaUw5ADKjvQ69erfg5OEcdqzRqFQ5QqoYFR3hAwMxvZeKQmdXqOu
HfBhCBRjG5YdIooU04u9Uxb0M6XlpR92h8qmEy3CLBXVOlSleQTDC/qe2nJ0+QvfILTHa/xl
R7LunORfid6rP8OZLOlXmWAxJOXTn4nctxO3NxO/Mtda00n9G5HDpWgpjemNgyllYabZPqqC
DVWUqqU8DKM0Il073MaJx0lxIe0VLrQ3etItOVZzQe/VXQVwI9EdLdEaY7lzLR/Tzu5Z+H+J
+/O0AEjJBCgGgPakfFBVzHbHWVcFLGoTECVuMQMofpYY2N6uA8hPQgJI5A9HX86S/jTf4W0Z
s/TvTqP2bq/A8x2fdJpk9vJdFdQbZD4FhrHTlFyLoK8bvKcNu3km7fQrxbEmaZ6UTaCwOAzK
HVtY26z8NeZzV0VOJlVZDpLWaqbwWT59YRv/YDMPoFSPWCmEUZJlqVSNqYIZOoBI9vhIhhKP
VByKN6L0YTQwZvw69kYs0s6Q2lTQgQtSEYZrch0+ozklEcACNYjz5jIpu5YJliYUJIYDrPmN
CPcmm0Pj/JQ6a4yzSjpsCBWQKh8Uw8hFvVpPp/gWjnh4h2FpLuysE5LYZeOUdw5Btw6Xy/ax
m+wBEAxABOQk8CrK3IRkhbQodeJpL8xxzr/LeXmCCB07Eap6MCp7MEp6sQpyufbHcusZYc4g
o/ty0h2b6UlELAinLJMjGgWwYdEdxmmfMR3IEI8Z4iJrv+Jt7Fn353gMzUGnjrxlOqwOFbAS
TrW6TiZSFpdlUzZZ/FgEemmeeEeymo5pO83jH8dDskD8LUXnjwP+Dar+Ke/Qzprg+Sbi0j0z
wfpyVCtjWC07aEbMdqMt2bSfZUkTxdjttVbBxEOiX5C3jqVuAyWiiUgrpeMV5XAU7PCtKDOU
wN5jAyWMjEhe8iO/xvYbIBEhNGEIj39cyBR2Aqk5jl66GgCJk5ZPqA58OPH/Mhws42TC/zaR
bjP/Ze380qHxSCT79hflzw0x3J5HvkGlng6MZ0QmCO6TIKZHqPjtaZGipzmPacvPtnuAN/n2
RrCamWZZaxThfA6aDPu/JgDmT6uLsUp6sct7PcmNq0r4Y2BWPWQnOQOd6kQMoaE8olGl9zKc
2Knsq3CfwZ+ZptpfbclrXQKv8kIJlSum57CX5HIZ78mI4VZsrCO6G5sgiJpgjxtgjZMxGifU
```

```
sZ73wJZ+gH3h0g0haimbELNEooCbWb6Zg/kwQeBD89s/zdluNQ7SbPTN0nHEhqP/UEJGKgTU
gdlXkyVUDugXcJEi0w8jrc/E9w9V8YU8fpXkCyPUKiPUK6vYRM90bSfevJ9K3F7Q6BIXA8JW
lQuldaEsA6iq0W6cjF6lxKtlt+0HxSg88xPfgco70FjPZ+F6kLKox9GOIjNgYc+glGSfhknC
zxRIppVZcPlnTObil3iptnuQvSBobpW/ulAAAgAElEQVTBw0iSmFwSX+wqCqW/PhdApHmCc2
bODZjecLaAIAqON5GpxwMBOl3Va6TgtUZflT5mLC79XvYc2cu/Ded/EPrXQvcC6BDlQFe+te
3nAIwQTJ7ModMFZE4X48gX7VDe3xpoXWNzAAWQpg3QpOUVa6BZ5rAm/wSMFdAus2uJpfaV0C
5zW2NYZq/TVgkdG6FjA7Svx2+rxGkuI3WqhGRTEfGTinZuxu3cQra3nNRgKfGBKsb7XyNl6S
dg/D+/O0A0UaS4dzijOns2VR0cDqM6Vqu8tVN2nAfEuE2Cm6S4Tdps7jhhlLhklzHZxAeu3W
dc+lQYZlbF0hRvFnomJmCyBy7+PAy9QmZgAfFuAaSMbNdG3L5S/L58/O58mCudXgH986FP6l
3Om6RMkYm6eJ0RXEXe+nLw+paT6VtJ7HQhk53rifV+DIZ+Dq5+Be6dhrG7kBizRd/MRpEJpr
wkDzOquWsljPJ6JArUZ9Zxl2KtoPxpaD4ZgIQH6Q9ljCgAkzk3+D44ljr5ykLW/Iwt56/fMB
QIs3AjJNO+xWrzFVTw0wYoRvkId6lpstNv4D78Wcb7N+D0F0LfGji9Gjplii4BZXP3LA8ETC
50Pz35PcvIDkQMub0r8but9raCShpB0UUBpgI6ZGYvg57n8PsieL05NkLasxB65sPpoO3VeN
oxpW8x9OVCfwl+/zr8/m3Q9xp+/3dB/8dxej8DF34Xd+jLcPO/woMvkb3lX0nf/EtTYhfn3n
sHSAgSI/Vm/acCcyILHMFFal2p45pPuYNnViuqHcFjHOkWVQT0Xc9YWcbq0vyC4QchJli4p+
IYIknqG79AamgLsf4ckt0FuKfL8HuKoEca4OkHJjxHA8Tgc/j98w3QvM5S/I4AIB3zcZXL02
slTLYzD6e7FG9gLe7AJjI9zxNt+YTdj+LqmzB2DNyztqIHWnSllXYm99HssiugCCTCgl0LHd
QoDRjdcrTtWBvynV3uZ5Yror4K+svCR9saq2qkStbY2YoQVo8BIzxvFkCM/2PGSoWwo5CagO
lrZO/tYfziP+Nezxrigwtx+iK2n00EqBDkd3QrfC9aMecx8HtW4PblkOlfhNe7BL9nCfSEll
WrlKcV+KdzcXsKSfWWEOtbwlR/FRP9m5ns20ayZysp0entJHu2E+/ZQaz3RaZ7X2Gq50Okzv
0d0kM/gHv5x+DmL8Dd34SRL8DEV2ByP4xfIHvnEkQvgn/Njp+CTupETRTO4vU5vBUyZMTOIv
09Q2FUJTQVQps4XLoqyRdIOyP9BLBAaEolhYffbyZ55WeZPqsiYItwJSVkTyoPTCZWz6q5Ue
9y/L7F+P05RlPQuxxkW0ubGCqC7lXQLSCWQPdq6KrE75FEkgRbSeJUFdnOT8LQ5+D+VyDeaj
elN7UxrPYI/TOjZV3XhMiduEs2ljUVFNPOFBlPJULF6KpwKVJwVaVoRCqXak2gpI8xV5Ne0J
pAlN0YT6asjpMuMUVrfG2HbPfsMAWmFa7lM2Qczd/ESKfDQndX7bzWw68Qv/gjTPR+gMmuzc
S7V5s5oWz/KpyBHDzzzHm2j06vhNPqrzn2vcasu5xsy3YTmfT7lsPZCJyP4A3KUY4w2RkhdS
FC8sJSYhe3Ebv4aaYv/Sumr/4Hpm7+B6K3fpvYtf9C4voXyd7+U3iwEyZk+nZCesguqXVGzF
4hpjq+CnVoBaFK+GjORwWFJbFMirYV7D7jQSEP64c9I0AEs3BCUVPy+kFbKcIgcLbpICfG2M
QCi25GIAm26TXhTlUWETPYpEBpFqPydfhkGu/OHuKX/qlxmJNKpOyQk7jeDpak/BzJ7l2FGR
xjaklNSzqWQKfsl0roEkgEkIKABMjlMBCxlCtToxS37lWcMz+Ie+XX4OHXINkO/kl8Rs0Kxa
SXJOlmTdlps/4o0BxZL0nWlNClGyGEhctsETNVXFHFS8loPwqKGbAEGkmgC0ldpe8eeR2BSa
cKHdkkmeQkWWkJV7pNY3AP6IP4Trw7nydl/h+QHqjC6lsJfUUwUIHTU066v4jk2RxSZxeSHV
iOl7fKAuNZwBEAhK4PQOfL+D0FeAPzcM5FSJ2PED0zj9GBFSRvvED85qeJ3/4lUvf/B9mxo/
iJPvDOABfsgjd3GNwR+lwh0wcaYCaiqh0lAtktv8vUT9beQmHcyFS/j5tKxyr7E/Zx5Bsmld
46yfSOf+suQi9Erf7WyIcOplUP8jmkKozvEah4fWTUh0BkTpF5oOrrsrJkuz8weljoigZX02
M4t/6EyfNVTPVG8HsXwflSAwwx+JxI4JA2kLlm7NWF0KsJSGmRSujURFJpICVzQJqrV8eEtN
isrNQMb6p7Gan+IjLnN+Fc+wT+7Z/CHf7PkD4JqSugOSP5zDJJg6Wd5tlCFaruMCnlvgmfy9
FPOtrfS7P2tuC2ClUn3/Jv0plmyomZIngJ3xZTNf2tPhUJWWHgylTxiEL6ISRuQ6KD2J3PE7
v+A8TOvkqmrwp6ltgMhe4VZl5C/ldqYBWxMwXEzq4ieXYx2QEJBYEkz7TSpHMhc43+EhhcA4
MbcPrKiPYuY2KwgOjlV0ne+imIlsPEdZiSbyrhK6tDGlalq4JAhfrQc02l+2wmTiodJZGeJp
qaCjYLst0gDn0raQysHBf/2cqKj8xVlR4lvRj25tO2spPtTHCYKhEqDdMGdm94E2pnkwFJeE
JQh0gAiZJkmltEGTaAMbeoJxs5TfL6DzNltojU4AI4vwA0WP3LoF8O9lO2koS9hdBTDL35oL
/NZ7n2MzMLq+/0uX7nORiYB/0LoHeFOTbVtYBk7zwy/RGcs/Pwzi3COZtHZnAj8b6Pw93fgI
f/G6ZaIH0zcOSleUvWlODlZDNJBejBxcxhZwVDoWohUspWfEibqD69FnBpjYpMMnljB9lEGn
WNUEmbCIFCh5q3eQDuZUi2wsguYtf+iPGhX+Th4HczMbiGeO9yXK0HklaQ+dSlyKTe+P0ryA
```

```
zkkzhTSvxsMalzS3DOLLL9HfbXXNv+JXjdETi3DC5WkT2zgdHTVYye+QjZe78M6Xq7kWKgIh
XpVoFvdVnMc4m5jonAKZ9Q2M9mfLPSMZNNkHaiyHQN657ZfrLWibpodldbGaUIn0o8CYA2am
grKz4DQGzKxONMbwAgtRXSrJvRoiO7tkKFEe6ZTTWN3hNIBKYA4eoPYVll/RNmB4yslYKxEb
MpZOrazzBxMY/xM2LK56B/SQCOubSLoE+OpsKYMtnKLEgGnoPBSHBtgUHREIFQQFF0ZAX+wD
K8C4twLi4ge24h7uCSQJLmQW8xfs8asnlbyJ79CM7lz+Ld/XUY/7pNy3HugqJ2GqnZYi3U7e
FnihQnbAVDL+0bU0m5X9o2IasJSwFJnR5eR+CwHWjb2ChM9cCDl+H2b5C69hmmL36IyaFlTA
+VkDxbQuZMAU7vIrPKLq3VmFpslrsAzhZBr8zHMpz+YgMUZyAXd2AZDEg4LXjUP6ZvlD9PQQ
OLyZ6O4A5GyAxFSFwoIXrxk2Tu/DokjoP/wFTzlopeJ4UpF6qa0Yp6ahMgU+ZUyHgrxxsmnF
ENto8MCtRZYehbroFsrVC6BBZN+L2ZdFXp0W8FQGZpCvl+KOGsWpR6lIYuD5BDZMwsHSB//y
ljLR4xm904U2aDeFIqqHoPRvYyeePT3DuziszZlTCwdI60BAYWghzEnnLoqYRe2d5LrcMop3
FwCd5gLt5AkSEGVsLgUhhcjndmCdmhRTiXFsOlfLhQCufLYKDISGLNEDunl5DoXs5EVyljpl
9kcvAfkrr6y3D/yzal2u8B9xw4lyGj4toPQflRiRTE3EfgERNkpU60h7N8Nyl7jEImjukXTS
5pdZ9209I+h8mzkFQm9p/DzX9n0vhj/a8xlVNMtH8B6XMRGFK4VH2QB2dXwJnnyPZFSPdE8P
ol2eXbCSRltu0rAFH/SnMsZ2z/mD5RvzwtDS6BcxHivRHGTkeIDr0AD38dsifBVdjcagWlna
eUca6ijYGbq02BTJhTfaF+MTRr7/JQwDwGFplockQUPAoA8tj3gVkq4ATHRd5G/s9i7/DsJ7
UhGnU3eh8au0Jq8LL6y3L/W9+H0k+XD7SIjb9oNyYb0lfJRy+TxpeZoH23tVdf7AqJe/uYvv
xLJBXbVkfPhfqXW80gp3Qgx4R8zaCfmR8AYyXeQB7uQBHuQImlwQLcMyvxzi7EPbuQZJ9MkG
I4VwEXVsOFcjhTjNujXKXlMFiCll9CuqeIeHehSWNI9pfinFtL8uJ2Jm/8LcZv/xDT93+S9O
j/B9NfhMROSJ+ATD84l2xCaHYEU9hPrcwlkXcrAFcbJPeTnfxjEg9+hanhzzJ57buZurKVya
FyEufzyJ5bhX9mFQzkQ48icuUm58wsk+4shDOFcGkV/pUcspeso5wxgQgFLpbbYEWnzivARP
sG50PQTwwoaDEHOpODfz5CvC+HidOvkLn4CzDRaKteah2OapYZ5ah6AFpeoK0lFPtXHyjJUo
EGmUWWFMOTESVBrKkEV6onVABv5T39Lf4LNUjY6jOT8mN5O9Aguo2QW5+2lUlIr4tC2M4Cy+
z8ovAmlM7GXPi5btpcR4FRXctmz5pGE4jZuO2ldJLM5D0y9w8xlvd9JAfXkRkswT2z3A7aGZ
lHOVYTDIpJl+KdEc0C0oDMseXQK6koiRh5pDXOKtS43IJDgAgoa9o88zvuufkGIJwXcFSlcR
VRrSfpzCGl+ZUzBTBURqYnD29A70vgSjFcWW4kt8yK6b4Ik5dzGL2Sz/iVtUxce4mpqx8nev
UHiV/5KWKXfxnvzn/Bu/0l3Nt/gnvrT8ne/jMyt//MhDWzw3/M9LXfYPL6zzNx7UcYu/JJRi
5t4eHFlYwMqdxAhNGBCNGBiIkOcUFhlFwYkLbcAF3roXuNiQhmOpeQ6FtI9sIS3GuL8a9H4F
LE9ouWLGi+qUuRwlzokwZdaISE6degj42Qmq3Nle/qx8GVVgsP6rdX2n7vk+ZaTrwvzwQI0h
c/B7e/DmMXIWHNJpM5bgAi0lxblN0K9pdXloXdnFOTnOH+KWFrLRWBJvSPQ/4OzKgQLGLlt5
LhQZkzOkfFq8lLR+kllbnhNz+ll8L7CZdOryp2a0On/23uQd7YzN3Hn4fPokBXPgAw8Qe/j
funP8h7g++SOJSMdmhYGCv5sKFIvzBfOM0Z8+tIHt+Kd65pXBGJlng0A/KtJBJtQDOzTetf2
4R/tml+GeXPUbezN9Lsccstt+fWYEvmvl+lnmPfW7Ps78lD//8PJyh+WSHFpAdWkxmaBmZ83
lkzhaTOVNBZnAdqcGNpAY3kxrcSmpgu6HkYNhuJXlmM/Gz60meW0PyfCmpoVVkhhaTvTCf7I
UIzoUI3lDESGr7fIvs8+q5B6VRVoHpg2Wmb7xzi3HOL8I7vwDf9IeOXwyDmlQNyPTXYnO8NJ
P6lj+72PbhwCIQnVEfLsYZWk7qfB6pc6U4Z8vgrLSVwsjSTCXcaf0oYxd/Bab2WG2onXylD6
MsoWCLd+tiaApAQQlpEjnRllcfTarKyVbQSK2lGR6a4WvDYI/+C3krbB99M4OcZ5wHeeyK78
Mfk2TSzTy88l8YufpZoldfJHrWJpxxYQWcL8Y7U0j2bKFhnNTQMpzzGkgNYuC3nBWTv090Tq
CcZ+xw2eJzITGxf07MPFeyjDyXPpDAyJ7LI3s23wgII2gGF4EijGdlP4vInFtC4twK4mfzyZ
4rsZpUWqx/vqmlNn7mXzI9/HVIdmH3qlG6keVPhaslLilANBckTaKoXbBNYCBfv5WM920OEE
mSGzjJg2Qe/g7pW59lerCS6e75NtJyvgD3bBHOuRLSQ/lkLyzHHVpow8NnckB0LgfOvl+ke3
nOzB7PrdW5CnfrOnOlZ3j2czJp7YSiWbszKA2x/LE+VUg4O5hDdnAhvo6/WAgX84l2muqvxB
/5HfyYIlZK9VAoOmXTxGUkBJFNa5/IDxE47D6Nssol+L/Vr29zgMjXUSRMdmsjjH2BzOVPm5
VhSUWmhlbinCsykit7vojs0ErcC2IoSevn4OxCOCfGegYmeaZz9du6HzH6XCiYC5ozOJ7x2c
/Jj5A2CEAyWIF7Lh/3/DLcoRz8oQX4Z+bBWaslvbPPkT6zgtT5CjJXXiB54zPgVQPnbA6UP2
4mVJlklgaRAh9aGiPc0MZuSRGA4/8HhHybA0TLSpQHFofMXRvWfPgfyVz4W8QHC3AuLsW5UI
```

```
BzvtSQO7QKLs7DOqvz4Px8OL8ILryPJI32rHRhIcyZnuHZBUwBQNGsc4vh/AqcoWIyFwpxLy
6Ciza3yrSXrB80PbiUiaEPkb3972HyEDAUVBEZA2cSJxkjGU+YZd2O6xt/QuB45HAHfu3b+g
3ffH3ybQ0Q9VEyyBq3AbQHkKiD4Z8mfqGM9JX5ZC6uMoMmkLhDeXbQNHBy5gUQMef7CpBFAU
Dm2M4ZGCGongEgQ4H/pPki9eeF+TgXVpG5mGcBoiiY+vpqBK5F8C+tJHbhg8Su/zKMd9mJUi
3ZVl6eF8XNREmnEqb2bzqbQbvnWsfbAsR6JIFtFcaDvsVa5NseIJkw3V7uiJYxpgfgweeJXq
kgcTlC5tJynKFC3POleOdnA2TeI8n9zEwWMtvTtrOYc2ixBerTtu/bvetZA2AYcFgweJciuJ
fmmwwDo60FjqsREzb2br8Gd34fJs7alBpZyCYY5Zg96l0nY0ubellcX4ZVOF0QtoH20DlyQt
T+NUCerDbVN4pwKK0lE7VFOUwh5YdfYPL6ahK3IqQvLzKawztXDAKIBtVoEJk2YsqnZepv9v
HSHIvnRkbzfbPv52muJ60R9Kf69DGah3dpHt6lCPGLS5m8uJXM8Ocg3mUHLVwVocltOeOhoD
M8H4ZsBQyFtNTOWkH5lwB5MihmfxMCRGsjTDxQc5XOBKS/xtjtSkavRshcXYB3cSWcL7STZB
pEqf4LS/AvrMC/mIN/acH7RIvs7l9cgj8n0r0vep/uXX22CG4sw72cg2f8uQVwcQlcXoh/KU
LsfITJC0VMX/t7ZLQvYKzTluCSz+hlTI0CZTYrSjV7rtgu5HoLOKQuNOAijXdIsxniW/D+29
zEUjzcIe27NpVewsZVbtJeRu+s5+HlCOnr8+CSIlrSHsvh0jy4PM/Yw97lVfiXF5sBlaC+L3
Rpsb2HS0ufukU5YJdz3p/7vrwQ70oO8cs5pK6thGv5cKUALmqCdjnOhRyiQ2vI3v8XpB9+he
zEOUgZO9gWkvAn8bLKywusgEDGWQyEk8WB5ngrOEKQqP0Wv77NAaKYuGZYlbvl4GeVyKe850
OM3N3G6M0QIDKllsIFMVMErszDubyK7JU8vCuL8a/Mf59oocl9EqMJqE/bKm/Kv6JrPAvN/d
mzl+YzdTlCbHgp7s0KuFJhtfKFUvwrH8e9qYrpbZAew/fs/IWiUWaxljuFrw2LvgEgdgnFjG
llEfNIY4RFML7FwAgv/20NEClnVTPMKn6ueirelF0qljnF6L0PMHYrggaRSznWllCI93IE/6
qK2q8kbQAiJps7kzzrud5VK4kFjqemq88CDJ37bM+tvo3fWcz0neVMX80jeqGY5MXlcOP7YO
SLEDsNWWUdWjNK/qIAYUBiFnBoCfAjDTLznQlJfqPfEVhZOk6LpRT+tcgJ2fmb336bA8TD99
K4qupoikQoylMiqZ+Je59i4lYOzjXZxEpHjljtEURU0tdXkL6+yiTmce053he6ugD/2iLcOZ
LO5apowbPRHJ9ffZu6XcbEjXxGrkcYvllK+s6/gIk3ID5uIlWBbylRpjWAqqWmajZmnLQc4z
GAyDmXnSxnUl58wP86XuCYmVlXzq6WI2vlvQb8W/f69gaIOtBRRpvWWA/bNdYSKpnrTN39DN
O3i3CursLY6gLIFRuPd29ESNlYQuLmCsOc7ws4rj2Hb2gB7vUFFiRzaN8vcKjP3KsrSF5ea9
Lqo3eex5n6p5B5E3zt92gDUI7WcHguGT9lSNEqJSEaVaLIVYAVtdb8CjPDNQEcHKcvtRRmJj
dLK/VjM6bltw4ezlzV5Ft5a+/h2gKIBI4q0JlU6LvBwDwgevcnid3egHelCC4vseBQTP5GBO
9WhMRwDonbS3FvLAQ58u8LzYfr8/FuiBbOoV0Il56Rruse5vj8V3Nxr+yAG/8ERv4AkklmrY
akuoYl5vlmWWzKyZrl4mZfu0BDSFPITJL8F+m9lgLNFMkzi5ZmAWSWtlFJpYTZyl6ZvZKI37
rXt78Gyfi2IIJZTKNUaYFmgqm7v8T07VfIXiuxTBTM5gog7q0IyeH5hrybFjT6/BGJYcQ4Ak
9Ilpm5MW/WcW85R98F5zli+Hl4TzxH54fnBNc357/H999wXd3bbGafHwBoCVxdakkmp55p9r
la+xHchwVreB2dn2OvoWMM5cDlHNRv2WtFMPxD8FD+Rp9d3aiSs6aMaYqEaoR4abP2/tEiJI
FAzrnWlatVnlWoPULzSuljDsfbahBpkb82sd5ROEj2hLULg8VVEiiph7jRL5Gd+H4SwyW4dy
JwOwK3xAg5cHMZ7vB8nLsR0Hez6XYE//ZC/FvLcG8tnyH97d/KgdvPwXDE0h0da8m9GcG58R
zpazmkruaQuLqI+JUFZG8vIDs8D2c4gnMngjdsSeeZ+7n5HNycPze6FcG/E8G9Zlu9N/cWXt
s8rwCRC0ZQlMDlfNBz6BkeO34+7u0cnGHRQtzh5+zz3iiAaysMOPxrEdwruTg38snciTA9vB
7YbXaHxZ0GN2N3AJgpMiGtELzMG9lMoc7Qd/Z7e0w4saHv9X7Wef+nvXMNte266vg65+xzzj
7Pe+7Nuff03pt3GlobjGlME7W9IpVQCTVSqpZUKkZMiSik2pp+CKgfzAcpVBFbNdoiFB/UQC
BSIaCxRaGlEUKjlUaTe85+v9+P9V4/+c+5lr773ua2+ZDmZF+yYbDW2WudNedce/zHGHOOMc
fQDenqlT2lpSNMVpP5e7P/eQ2Pi6lBjLpo2fyFenN6twKIlyAYfgGv+wCj8gFBykAGJIe7cL
hHUlw2zDVj9pTpzffFVcssxQ3C4gaRoXUiAUf/V7IMaRhMIBOVVgxDhUeb+IfbeEebuEdr+K
Vlw0xBeQ4gApWYV9pLxlekZSjM0woUrqDisgG/gB6lNGN68/xlOBRAduHidRYcRlskxRU79j
mA6DuBIyht4JUFEgsYLp6Bl09YgBw6hC/t4R+ewa+t06v+KPCNWVraLD3t/PEl5NVjedSCA0
RBbG0SJfkUQLIZn9cmGn+JsP8rDCoHeLWUoSVZ5wEiUBxtw+EpC5qjbZLChmEgaZ2M6SSZDS
ikPTImlbmeVlmGSg7KalDKQ3ELSlvEpTxROWe0lJ5zuXRfsYx7tGolwByDZ22a9tUHUUm0bL
SeQJqRwBsW9ggLu/Z6phHNuASOdduOjuqrKAW3GY/uS7/XM/W8oLx+OUAO94y3XCBOjhyiQw
```

```
mNMwTlc4waHzTlyuYBceX5sXDla9jowgNEkaAJw0srI9IickxNvkwyfJhh5Rx+PZXyxSU4Og
FHJ43Ej8s5OHwLXLzJHgsnoZg3ly4zu8pLICqtXKJiDor6O2OyHJRWobgOZdEqSWWZuOIQV+
fMOPWhoBCNbShsEpdXCGvWTAqrVx6XiCpLhFUdVwirK5cdo3KeqLBvSIBUW7N+q39qR6Q2pS
l0vWr7Y8ZeEqgtaOLSGmF5E7+Sx6+uEVbWDMiTw004WjX3xUUHXlTZJ27eRdB+xGQ3VMK7q9
FryKvH8qhrACADU/4sVLmRTIOoypVqjkw/ybBy/rsBUjhBXF5CTBEfnic+vIm4cNL8PQNCYQ
cO9ylwdCxsg5hODJWZJjqXmXSZqSTQrBmQJBULkkQAEXOKStI8qlDYgKNdKJwkKl5HXLiOqH
iSuHiCqLRDLO90OY8YWeBQfzXXENiiWnrUuUBS2jb3azy2jysWFCnzZyDIwGPGXs4ZcGZjEc
AMQKpr+FVpPgFkjeRIgmDJ9F3j8CoOXv0MtD8Cg88DxauCQ6BZ9M+CA0TqYmQAIoetqvVaM0
t5tP4VvN9nVLmZoGElp2XOXSjuoh87ruaIi6dJCgck5S0SmUoymaQBCqfg6ACOzkJhH4oblr
kFDgMQSdVV4osOiZaOj7R8rIntBklph6SyTVzbJK6vkEiDZSSwGI0iIOVtO4XTcHTatqN2i3
tQ3IGS2pQ2kgknU+6SFphpA2mfWo64sk5S3kill9y92f9Iu5W27DjLW8QpeMlzypqPbRFVdg
hqqwSlHFFV4F4l4zL3qJ26w7Tu4LbOQ+dRGP9zmsl/0WFw9f4vPEC0dq4sF0EYm4TEBiDBAN
xvgv8ZhtVb8RsOSU2MLem9C5LQ6Q8elWQGrVqGETBKYsydOQYVYFKmFrPpXPMMAefoPMHhJk
EhTlDcJSjvElUPiOpvIWyeIWrv4TVX8VoOQdtS3HKImw5JMwWNeZ6kdEoFmWnSMK90FHAln0
iPpRxx3TECQMxrxqL5lsYqyjSXQCYAHp0nKZwnKe4bQOkdWNBdDSA5Qk329ZyWQ9R0GDUdJu
3boPtn4B7aBGxX56+Fv7LgALGhCcqDpGTP0iIGICai97/A/3OGldssQIzUFhMKAFsWIA2HSF
RPtUl5awYgM58QIPR/lzGbJvGnSApniUoHTKs7TGu7TGvXM629E7f2Ptz6B/DqP4vfuJ9h6f
0My+9lXHkX49oPMandilu/Aa9+lrB+ykr8kky3V0tWc2VzHb+2wbSxjltfNeOUttSYYgkFM/
fSsvSOAXRydKMBiLRmXNkgrl7SNJdpkPqylSBVhlCTeD2n5eA3Hfoth3H73baisetjizIuPA
6uOoCFB4h2nYmU7yJK4nQeotpuSjL2JOPq7YQtB2RmVVahvA2VzRlAdMlI9NoqSXUTypv2Pg
FDjNFO/7eYJzw8QVg4RVw5CXrZgg4AAAtoSURBVOlNktEq3bbDoJ/HG78L/Acg/AxET0P0gu
lDUAXvJZh+E8ZfgcET0H2cpPVJks4DJK0fgc7l0DpH3NwnrO/hl7bwqmu4lWW8umx+B6/x3T
St5Um6dzOu30W3dJpBZQWv48DQgZEDfS3drpKUdqFyGqpnoLZPXN4lKGziyjko8NeXobpDUj
tBlFgjauZIGjk7fr23ngNdh2HNodtycMf3wvg/bKzVnMviqly2wBcWHCByKcndZGM7zTZNpT
hX2Luv3KlfYVK7IwWImECTZwEkTyIp23TMtVAmT32JpLYClZRpxDiipv7OQ/mAuHQ9fuU00/
oWk7ZDv+vQ7p+mN76bIHwQ+CPgqxBXLfMoM2ZGZiuEijMewvQ5m5zZ+xuYPgbex8F9CNwHYf
IxkvFHiEcfJhrdz7DxfobNCwxb72XY+rHLqf4zMHiUuPfbBIMP4vVvZ9o9ybTtGLNO4Pclud
cSdG0DGinVNohKm4SFHMjUayxBfZu4sUvcXCdurUBTpPELaBlAlui0tnHHD8BEmeJTjb3AAP
h+XV98gJgANoFEOkSe3DSpq+KzkmcZld5NIKmqH7y6DtUtqK0bE0TAkLQUGUbINI00R00T9h
2oHED5RpLqzfiNc0xaJ+h3lul2HarNk/T6v8Zw9DmI/xH4lg2YNHn6bbIVw0RZaQOFsmq/ih
IuKwI5Ua5ZBVkeXkEXbTkwlQTzXgTvf9KjzkWqN/J/4L4I4+fBfxaSv4LwEdz+BbrlM3SqDo
Oqw7TmmFU8CQOjDVtL0MpBbQ2qcyCob84BRPdYs8ocew5xZ4lBbYtu++lMp58CL4l7elODfD
+MHd/lzDeolV3l3FNulkRhpAohVV3j+BvG/Ai7S9CUFMlDbRsa6ySadIrqW0SSrGIgMYWOAk
hVK0L7UD5PVLkBt3aGcWufQfeAweAWBoMfptv7RfzpU/iT75AombRSYipUNTHlXAh8peyPbc
klPyJRhgl5+udJ4eAaQLZMHdnVOFVDMqtyuq65laK/rzjOIgdUU8/klnoK3MeYdO6jU7uJTj
XHpOnga5wyFWe0DNIS+jsDQiNP3Ngmaq0TtZcu3avrPYewk6PfOE2v89P43mchTOv4Lf5K7v
dk4IXWIBlAxDeKHlX5MlOlT3HRRv0/z6R+DlFvGdrrlsSo70Azbxggbmsyu2fImFIZQBrSNn
tQvc7Y5V4jz6jt0Ovs0u/dyWT0q0STJ0imXzfl7xJhMiviog2NgUK8lSAwxo083GiKH3oEQW
TvU6oi5eFWOq8AVB5bKf7NMbTl5ZXuX3Xm9RyVZlZMplFEc0cpJBNBYMKYhKgaRP8O7ufwex
9llLwdt3OCsOsQSYvOAJKedx3Ivm+tkbS2iNt54s6y/V7XDUBWCDp5hs23Mug+SBT+A9A3+z
NM+9+TxRb74jUEEC33XgmQbzNt/ARxfwU6eWhuQmPbAqTjkHQEkG2o7loTbAYQzTs2jV0uDe
O39ph0rmc6uIA3fgS8v4egOB93Zwq8yKpTni5PlY5izYtk9qkYp6JOxzYlv8K2Vc5R6wjTtE
55Visv3RCUKhS7IGeUi4LzsgC9YC7LoA0X97VXQgGwZhVPCRFeAO8JGD5E3LsLBmegl7NMn4
0xM7kEGoGklSNpbbwyQDrrBJ0NRq07mPQfI4m+ZnZwqp/XuALJsrsvJsqNBpltotEOs4HdU6
BfTltEoheZNt9HPJREzENrywKktW5XZnoOSVvmRg5kml8pYVs5osZJovZ7oKeUNV8A7zkIW2
```

```
memjTwVB2RRSdGjxLCWKtq6o9KgGXFgurEtEzcWKLiLaaSZEzsTYi9MYk/MWSrsGpvvW/J7I
tIH64G5klGZTRmjGtSH6kPlnyTSvoOeE9C52EY/DgMdu2YBRBjQq5BffWSpmivQnuDpLNB0l
0xqlZaudJ7STqb+J0dxq27mfY/C8l/ktAzuzmUSfla/iy+BplJ3WzDjSoHmdJUEB/Rb96HPx
RzbEB7DVoyrlZIeg7RwCHpWk0ibWKo6xB3HWNSuK0bGFcvEDQ/AcOnIaxdmj+oqJFKHqkSke
FKa+jZdTUFUSqZRB8vruFTMRSmZecS7Z8XghPXFoFRIRjZW7Fr6l5okm9SqgagIhmKo8lIdp
USganZOGIYdEyZumy/t9EiMi8lB4vK0Pw76PwOcfcCQec0QTNlLMr7LoAIBKIUIFq+pnNJgF
jBsYHbyzPs3sN08NeQaGGhk9by07iv3c9CA0RAMOE+EmLSGjqa8lppYQma+NGniZO7jEc7C/
kwXmzXIZSvYOLA2B4jl8FzHSbuHsPBXXRbvwxodep/LcOKKWecqMYyu0bnGYlZdG73OlhOln
0Z6SGXwDQ7NzPwbD/qFUcB/pXIlAHQ8oTmX5bsttW0CfVV5L4Mk7/EG/wS7vBtuONlomkOvH
W8wSphb90KjqY07J4xtYKuJubWl8JwnUnPYTB8D/CMWYyIgxrE3VQ4vAmQN+wbmAEkE+RGoq
cOQwZMkj8k4l6C3qaVlNmklF3DFUCCJZg6uAP5Ndbo925hNLmPIPhd4lCZxwvWrMmePzO89Y
VE9fFJUJk3mulYcNgkBjo3XbQYtQARSLxvEbt/QeB+FHf6TsajLYajJcbjNYLhBnRWQaZnW9
p225il8UDaJQ+DTaaDJUbjnySOv27BqoDQWJrw+Mb/ejDlYmuQbJIoZsgY+DKAjHD5Y+B+ku
Ge9TDL6dVeIh6fYNhdI/KW8CcO49Eqo94dTFufhsHTELycMoCYQMtOWWNiCGkDgUNHXTiejw
WIACESWNJIm6yrpm/qZ5rlJX4egi8RTn7DLFO3uuuM/FWCqQMCgzzm/TUQYCZ5GG9D6xR0zu
IODphMfoFQBUfN+5apl6YsOZ7hvy6tLjRAZHWIKYz5MbNabLoY4ltgyjT+PAk/TzLcTwGi5c
wNgv4NdJtnGfZPMRnfSOz/FISPgvcv4CoaODVPDP8LfQKD3OFyjeuo744PHJY7sjiCOXCkcz
KZWppIJ5RT56Wck/JdfBvcv8VlH2Ls3s3QO4fnrs8BxCEZrRBNNmC8B63T0LqFYHAn/vS3CI
KLlm+TCST9CNfw55oAiPmNMvEp7lqWewmPSfBF4vhjBL2zYPwh20TlA/zunXRq9zDs30sw+T
gEXwSesxNQlS9XbXIpjxkOBAyFr6RfqtHjxkfaB5mZOrUaRWDRUvDYACSmSmKqwk7AV7HxLg
T/DclTwOOMph/And5KMsgZEzToOfgDB3+cIx5uQ/skNG8j6X+I2PtT/KBpMi3JspL/xzT8Jk
DemG9ATJFVHJrZF2ZZNPVOyzoPv0wS/Tpe+3oSrWIl94kb74DRzzHuChh/AsE/QfCSRYRQEc
uZEZo0v+a5BggyVQQOldhOwaHvdX5cH7UtzWnUqLqiRGpKhdMzWUOSNHtIGA8J/AmhOzXLyo
QaR9XsJ/emjxO5H4bRLdBfwu87TAcOk5FDMNIqV564+TYYPAzBk3hB2zgtzWt+EyDH9cu/un
YlMTVFNdktLFpsPIYWgQzTyAR5BpJH8btvNev5tG+G3gWIPwHTL0DUtNaTvG3hBCLtblfohk
gJ6dKJuJ4vEijm6dVl9Qdzl/ohJpWWMlpEDskyCSUSswwbGLdJpKGFIUE0IIx7JhvlDPj+C2
ZbANMPwehGgvEy3sRhqlU+lyHpO8Stm2D0KYifwY2qxtgMtYqctvuDGdwb46kLbmLNAyQtMG
EcaRlA9JL/Dfg9wt7boasVmnfA+D7gD4BnLf9rSiFGk3POVDxSphTFVknKyqxKJ+kCXSaxjl
t7aGjqg6ZGM0kuR+nLJLyUAiSeld2QqIgZEtEhSnzj+Y+MMlUhKl8F/zfBvYPE3cL3HfzQIQ
wdkoEins/B+DHga0ySkpmeyRUjl8yxatDXAUP/Dzulf7mPSc8LAAAAAElFTkSuQmCC
```


#### annotation

A Kubernetes annotation for the NavLink custom resource.

#### label

A Kubernetes label for the NavLink custom resource.

#### sideLabel

Label that appears in the left navigation bar

#### target

Sets the target property of the link's anchor tag (<a>), which (depending on browsers) determines if it opens in a new window or in an existing tab.

The default value is `_self`, which opens the link on the current tab. To open the link in a new window or tab, set the target to `_blank`.

For more information about the target property, see this page. [https://www.w3schools.com/tags/att_a_target.asp]

#### `toService`

Has five fields that are constructed to create a URL like the following: `https://<rancher_server_url>/k8s/clusters/<CLUSTER>/k8s/namespace/<NAMESPACE>/service/<SCHEME>:<NAME>:<PORT>/proxy/<PATH>`

For example, a link to a monitoring service can be set up as follows:

- name: `rancher-monitoring-grafana`
- namespace: `cattle-monitoring-system`
- path: `proxy/?orgId=1`
- port: `"80"`
- scheme: `http`

It is required to provide either the `toService` directive or the `toURL` directive.

#### `toUrl`

Can be any link, even to links outside of the cluster.

It is required to provide either the `toService` directive or the `toURL` directive.

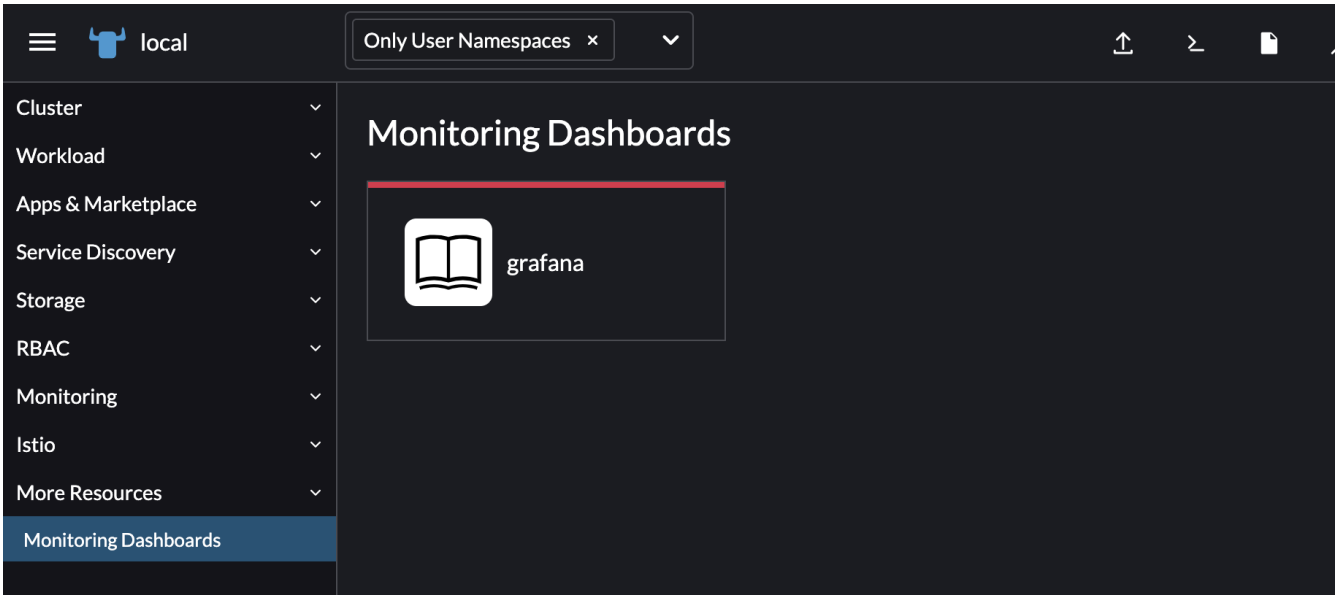
### Link Examples

#### Example of Link with `toUrl`

This example NavLink YAML shows an example of configuring a NavLink to a Grafana dashboard:

```
apiVersion: ui.cattle.io/v1
kind: NavLink
metadata:
  name: grafana
spec:
  group: "Monitoring Dashboards"
  toURL: https://<RANCHER_SERVER_URL>/api/v1/namespaces/cattle-monitoring-system/services/http:rancher-monitoring-grafana:80/proxy/?orgId=1
```

Adding the above YAML results in a link to Grafana being created, as shown in the following screenshot:




**Example of Link with `toService`**

This example YAML shows an example of `toService` used for the link target:

```
apiVersion: ui.cattle.io/v1
kind: NavLink
metadata:
  annotations:
    key: annotation
    labels:
      key: label
  name: navlinkname
spec:
  description: This is a description field # Optional.
  group: "group1" # Optional. If not provided, the links appear
standalone.
  iconSrc: [icon source string is clipped for
brevity]
  label: This is a label # Optional.
  sideLabel: A side label. # Optional.
  target: _blank #Optional. _blank opens the link in a new tab or
window.
  toService: # toService or #toUrl needs to be provided.
    name: rancher-monitoring-grafana
    namespace: cattle-monitoring-system
    path: proxy/?orgId=1
    port: "80"
    scheme: http
```

Adding the `toService` parameters above results in a link to Grafana being created, as shown in the following screenshot:




## Notifications

The Rancher dashboard delivers dynamic notifications and system alerts to ensure that you stay informed about the latest updates, lifecycle events, and statuses relevant to your deployment.

### Dynamic Content

Rancher fetches dynamic content, including notifications about new releases, End of Maintenance (EOM) or End of Life (EOL) statuses, and other important announcements.

Dynamic content is delivered to users through three areas in the Rancher dashboard.

- **Home page notification banner**: A dynamic banner displayed at the top of the home page for high-priority announcements.
- **Home page section below the Links**: A dedicated area on the home page, located beneath the **Links** section, that displays relevant updates and resources.
- **Notification center**: A centralized hub for all notifications, including system alerts, resources and announcements.

To interact with these notifications, click **Learn More** on the listed resource. You can also dismiss the notification.

### What is the Notification Center?

The notification center acts as a central repository for both system-generated alerts and dynamic announcements. You can access it by clicking the bell icon in the top right corner of the Rancher dashboard.

Notifications are categorized by severity and type:

- **Error** indicates a high-severity issue.
- **Warning** indicates a medium-severity concern.
- **Information** provides general updates of lower severity.
- **Task** shows that a process or action is currently in progress.
- **Success** confirms that a process or action has completed successfully.

For example, you might receive a notification when a new version of Rancher is available, or a direct link to the Rancher Release Notes.

You can easily browse through your notifications using the up and down arrow keys. Certain notifications may also include an associated action, allowing you to respond immediately.

## Configuring a Global Default Private Registry



This page describes how to configure a global default private registry from the Rancher UI, after Rancher is already installed.

For instructions on how to set up a private registry during Rancher installation, refer to the air-gapped installation guide.

A private registry provides a consistent, centralized source of truth for container images across your clusters. You can use a private registry to share custom images within your organization or to avoid rate limits when pulling from public registries.

There are two main ways to set up a private registry in Rancher:

- Set up a global system default registry in the **Settings** tab in the global view.
- Set up a cluster-level system default private registry in **Advanced Options** when provisioning or updating a downstream cluster.

Use the global system default registry in air-gapped environments or other setups where all clusters share the same default registry host. Rancher uses this registry to pull core images required for system components and configures downstream clusters to do the same by default.

Use a cluster-level default registry when a downstream cluster needs a different registry than the global default or requires different credentials for the same registry.

#### Set a Private Registry with No Credentials as the Default Registry

1. Log into Rancher and configure the default administrator password.
2. Select ☰ > **Global Settings**.
3. Go to `system-default-registry` and choose ☰ > **Edit Setting**.
4. Enter your registry's hostname and port (e.g. `registry.yourdomain.com:port`). Do not prefix the text with `http://` or `https://`.

**Result:** Rancher pulls system images from your private registry.

#### Set an Authenticated Private Registry as the Default Registry

If your registry requires authentication (such as `.dockerconfigjson`), perform the following steps:

1. Log in to Rancher and configure the default administrator password.
2. Create a new registry secret in the `cattle-system` namespace using `kubectl`, Terraform, or the Rancher UI.
3. If you create the secret outside the Rancher UI, ensure that backup labels are set
4. Select ☰ > **Global Settings**.
5. Find `system-default-registry-pull-secrets` and select ☰ > **Edit Setting**.
6. Enter the name of the registry pull secret in the `cattle-system` namespace created in step 2 (such as `my-reg-cred`).
7. Find `system-default-registry` and select ☰ > **Edit Setting**.

8. Enter your registry hostname and port (such as `registry.yourdomain.com:port`). Do not prefix the text with `http://` or `https://`.

**Result:** Rancher propagates the pull secret into the required Rancher-managed namespaces in the local cluster and configures Rancher-managed workloads to use it. Rancher automatically configures new and existing downstream clusters to point to the registry. All Rancher system images across all clusters then pull from your private registry.



For details on how Rancher works with authenticated default registries, see Authenticated Private Registries.



Reconfiguring registry details globally can trigger the redeployment of system components in the local cluster and downstream clusters that do not define cluster-level registry configurations. This can temporarily prevent access to downstream clusters through the Rancher UI.

#### Configure a Cluster Level Private Registry

In addition to global registry configurations, Rancher supports cluster-scoped registry configurations. Specify registry details in **Advanced Options** when creating or updating a cluster.



Reconfiguring registry details on an existing cluster can trigger the redeployment of system components in that cluster. This can temporarily prevent access to the cluster through the Rancher UI.

##### Using a Cluster Level Private Registry with Provisioned Clusters

1. Select ☰ > **Cluster Management**.
2. On the **Clusters*** page, click **Create**.
3. Choose a cluster type.
4. In **Cluster Configuration**, select the **Registries** tab.
5. Select **Enable cluster-scoped container registry for Rancher system container images**.
6. Enter the registry hostname.
7. **Optional:** Under **Authentication**, select **Create an HTTP Basic Auth Secret** and enter the credentials.
8. Click **Create**.

**Result:** The new cluster pulls images from the private registry.

Follow this same process when updating an existing cluster configuration.

##### Namespaced Private Registry with SUSE® Rancher Prime: RKE2 Downstream Clusters

Most private registries should work, by default, with RKE2 downstream clusters.

However, you'll need to do some additional steps if you're trying to set a namespaced private registry whose URL is formatted like this: `website/subdomain:portnumber`.

1. Select ☰ > **Cluster Management**.
2. Find the RKE2 cluster in the list and click ☰ >**Edit Config**.

3. From the **Cluster config** menu, select **Registries**.
4. In the **Registries** pane, select the **Configure advanced containerd mirroring and registry authentication options** option.
5. In the text fields under **Mirrors**, enter the **Registry Hostname** and **Mirror Endpoints**.
6. Click **Save**.
7. Repeat as necessary for each downstream RKE2 cluster.

#### Configure a Private Registry with Credentials when Creating a Cluster

1. Select ☰ **> Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Choose a cluster type.
4. In the **Cluster Configuration** go to the **Registries** tab.
5. Check the box next to **Enable cluster scoped container registry for Rancher system container images**.
6. Enter the registry hostname.
7. Under **Authentication** select **Create a HTTP Basic Auth Secret** and fill in the credential fields.
8. Click **Create**.

**Result:** The new cluster pulls images from the private registry. You can change this setting later by editing the cluster configuration.

#### Configure a Private Registry with Credentials When Importing a Cluster

1. Select ☰ **> Cluster Management**.
2. On the **Clusters** page, click **Import Existing**.
3. Choose a cluster type.
4. Select the **Registries** tab.
5. Select **Enable cluster-scoped container registry for Rancher system container images**.
6. Enter the registry hostname.
7. **Optional:** Select an image pull secret to authenticate with the registry.
8. Click **Create**.

#### Working with Private Registry Credentials

When working with private registries, it is important to ensure that any secrets created for these registries are properly backed up. When you add a private registry credential secret through the Rancher GUI and select **Create a HTTP Basic Auth Secret**, the secret is included in backup operations using Rancher Backups.

However, if you create a credential secret outside of the Rancher GUI, such as by using kubectl or Terraform, you must add the `fleet.cattle.io/managed=true` label to indicate that the secret should be included in backups created by Rancher Backups.

For example, if you have a custom private registry named "my-private-registry" and create a secret called "my-reg-creds" for it, apply the `fleet.cattle.io/managed=true` label to this secret. This ensures that your backup process captures the secret, providing easy restoration if needed.

By following this guidance, you can ensure that all of your private registry credentials are backed up and easily accessible in the event of a restore or migration.

## Authenticated Private Registries



Changing global registry settings can redeploy downstream cluster components, such as the `cattle-cluster-agent`. Evaluate this feature and its impact on existing resources before updating production environments.



This page explains how authenticated private registries behave and are configured in local and downstream clusters. For instructions on setting up a private registry, see **configuring a global default private registry**.

Starting in Rancher v2.15.0, you can configure authenticated private registries at the global or cluster level. This configuration is useful when your environment requires authenticated access to an internal registry, meets specific compliance requirements, or pulls images from credential-protected public registries.

You can use an authenticated registry in Rancher in two ways:

- **Global-level configuration**: Set image pull secrets globally by configuring the `system-default-registry` and `system-default-registry-pull-secrets` settings. This configures both the local cluster and, by default, all downstream clusters to pull from this registry.
- **Cluster-level configuration**: Set a private registry for specific downstream clusters. When provisioning or importing a cluster in the Rancher UI, configure credentials in the **Advanced Options** or **Registries** section, depending on the cluster type.



Partial registry inheritance is not supported. A cluster that defines any cluster-level registry configuration does not inherit any global settings. It must define its own registry hostname and pull secrets independently.

### Registry Behavior by Cluster Type

Because Rancher's control varies across downstream environments, how it applies registry credentials depends on the cluster type. Understanding this behavior helps you plan deployments and troubleshoot existing configurations.

#### Provisioned Clusters (RKE2/K3s)

For provisioned RKE2 and K3s clusters, Rancher writes registry credentials directly into the container runtime (`containerd`) configuration on each node. Because authentication occurs at the node level, all workloads automatically inherit registry authentication. You do not need to define image pull secrets on individual pods or deployments.

If you configure multiple pull secrets globally using the `system-default-registry-pull-secrets` setting, Rancher uses only the first secret matching the registry host when generating the `containerd` configuration. Additional secrets are not sent downstream.

#### Hosted (AKS, EKS, GKE, etc.), Imported Clusters and the Management Cluster

For all other cluster types, Rancher cannot directly configure the underlying container runtime. Instead, the management cluster propagates Kubernetes image pull secrets into Rancher-managed namespaces. Rancher

propagates these secrets into namespaces in the default `system` project, where Rancher-managed components use them.

Rancher manages the full lifecycle of these propagated secrets and reverts manual changes during cluster reconciliation. Always modify registry configurations at the cluster or global level using the Rancher UI or by editing the relevant `clusters.management.cattle.io` object with `kubectl`.



When manually defining registry credentials on `clusters.management.cattle.io` objects, create those credentials in the `fleet-default` namespace. Rancher supports only secret types of `basic-auth`, `rke-auth`, or `.dockerconfigjson`.

Because Rancher cannot configure the container runtime for these clusters, user workloads do not automatically receive registry authentication. You must create and configure pull secrets independently when deploying workloads.

### Considerations when using Authenticated Registries

#### Rancher Managed Charts and Pull Secret Injection

When you configure an imported or hosted cluster (such as AKS, EKS, or GKE) to use an authenticated registry, Rancher automatically injects configured pull secrets into the Helm values of charts from the built-in `rancher-charts` repository. This allows these charts to pull images from the registry without additional configuration.

The injected credentials depend on the cluster configuration. If the cluster inherits global registry settings, Rancher uses global pull secrets. If the cluster uses a cluster-level override, Rancher uses those cluster-specific secrets. Rancher skips secret injection if you manually specify pull secrets during installation or select the **skip pull secrets** option.



Pull secret injection applies only to charts installed from the built-in `rancher-charts` repository. It does not apply to custom repositories and cannot be customized.



Unlike pull secrets propagated into the `system` namespace, chart-specific pull secrets update only during chart upgrades.



Charts from `rancher-charts` can be installed into public namespaces, placing the injected pull secrets in those namespaces. Unlike system namespaces, public namespaces might lack RBAC policies that restrict secret access. Review your cluster's RBAC configuration to ensure credentials remain secure.

#### Updating the Global Configuration

Updating `system-default-registry` or `system-default-registry-pull-secrets` affects all downstream clusters inheriting the global configuration. This includes redeploying Rancher-managed workloads, such as `cattle-cluster-agent`, across those clusters.

If registry credentials or hostnames change frequently, configure registry settings at the cluster level rather than globally. This limits configuration changes to specific clusters and prevents unintended redeployments across your downstream environments.

#### Use Read Only Registry Credentials

Registry credentials used with Rancher must follow the principle of least privilege:

- Grant **read-only** (pull) access. Rancher and deployed charts do not require write access.
- Scope credentials to only the images or repositories the cluster needs to access.
- Avoid reusing credentials across environments with different trust boundaries.

Limiting credential scope reduces the security impact if an unauthorized user or workload accesses a secret.

## About Provisioning Drivers



In Rancher v2.11 Ember-based UI plugins for cluster and node drivers were deprecated. They have been removed as of v2.15. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see:

- Custom Hosted Provider (Cluster Drivers) UI [https://extensions.rancher.io/extensions/provisioning/hosted-provider/overview]
- Custom Node Drivers UI [https://extensions.rancher.io/extensions/provisioning/node-driver/overview]

Drivers in Rancher allow you to manage which providers can be used to deploy hosted Kubernetes clusters or nodes in an infrastructure provider to allow Rancher to deploy and manage Kubernetes.

#### Rancher Drivers

With Rancher drivers, you can enable/disable existing built-in drivers that are packaged in Rancher. Alternatively, you can add your own driver if Rancher has not yet implemented it.

There are two types of drivers within Rancher:

- Cluster Drivers
- Node Drivers

#### Cluster Drivers



In Rancher v2.11 Ember-based UI plugins for cluster and node drivers were deprecated. They have been removed as of v2.15. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see Custom Hosted Provider (Cluster Drivers) UI [https://extensions.rancher.io/extensions/provisioning/hosted-provider/overview].

Cluster drivers are used to provision hosted Kubernetes clusters, such as GKE, EKS, AKS, etc.. The availability of which cluster driver to display when creating a cluster is defined based on the cluster driver's status. Only `active` cluster drivers will be displayed as an option for creating clusters for hosted Kubernetes clusters. By default, Rancher is packaged with several existing cluster drivers, but you can also create custom cluster drivers to add to Rancher.

By default, Rancher has activated several hosted Kubernetes cloud providers including:

- Amazon EKS

- Google GKE
- Azure AKS

#### Node Drivers



In Rancher v2.11 Ember-based UI plugins for cluster and node drivers were deprecated. They have been removed as of v2.15. Migrate your plugins to the UI Extensions Framework [https://extensions.rancher.io/]. For details, see Custom Node Drivers UI [https://extensions.rancher.io/extensions/provisioning/node-driver/overview].

Node drivers are used to provision hosts, which Rancher uses to launch and manage Kubernetes clusters. A node driver is the same as a Docker Machine driver [https://github.com/docker/docs/blob/vnext-engine/machine/drivers/index.md]. The availability of which node driver to display when creating node templates is defined based on the node driver's status. Only `active` node drivers will be displayed as an option for creating node templates. By default, Rancher is packaged with many existing Docker Machine drivers, but you can also create custom node drivers to add to Rancher.

If there are specific node drivers that you don't want to show to your users, you would need to de-activate these node drivers.

Rancher supports several major cloud providers, but by default, these node drivers are active and available for deployment:

- Amazon EC2
- Azure
- Digital Ocean
- vSphere

There are several other node drivers that are disabled by default, but are packaged in Rancher:

- SUSE Virtualization, available as of Rancher v2.6.1
- Google GCE, available as of Rancher v2.12.0

#### Cluster Drivers

Cluster drivers are used to create clusters in a hosted Kubernetes provider, such as Google GKE. The availability of which cluster driver to display when creating clusters is defined by the cluster driver's status. Only `active` cluster drivers will be displayed as an option for creating clusters. By default, Rancher is packaged with several existing cloud provider cluster drivers, but you can also add custom cluster drivers to Rancher.

If there are specific cluster drivers that you do not want to show your users, you may deactivate those cluster drivers within Rancher and they will not appear as an option for cluster creation.

##### Managing Cluster Drivers



*Prerequisites:*

To create, edit, or delete cluster drivers, you need *one* of the following permissions:

- Administrator Global Permissions

- Custom Global Permissions with the Manage Cluster Drivers role assigned.

#### Activating/Deactivating Cluster Drivers

By default, Rancher only activates drivers for the most popular cloud providers, Google GKE, Amazon EKS and Azure AKS. If you want to show or hide any node driver, you can change its status.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the left navigation menu, click **Drivers**.
3. On the **Cluster Drivers** tab, select the driver that you wish to activate or deactivate and click ☰ > **Activate** or ☰ > **Deactivate**.

#### Adding Custom Cluster Drivers

If you want to use a cluster driver that Rancher doesn't support out-of-the-box, you can add the provider's driver in order to start using them to create *hosted* kubernetes clusters.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the left navigation menu, click **Drivers**.
3. On the **Cluster Drivers** tab, click **Add Cluster Driver**.
4. Complete the **Add Cluster Driver** form. Then click **Create**.

### Node Drivers

A node driver is the same as a Docker Machine driver [https://github.com/docker/docs/blob/vnext-engine/machine/drivers/index.md]. Node drivers are used to provision hosts, which Rancher uses to launch and manage Kubernetes clusters. By default, Rancher is packaged with many node drivers, but you can also create and add custom node drivers to Rancher.

Only `Active` node drivers are displayed in the Rancher UI when you create node templates. If there are specific node drivers that you don't want to show your users, you must deactivate these node drivers.

#### Managing Node Drivers



*Prerequisites:*

To create, edit, or delete drivers, you need *one* of the following permissions:

- Administrator Global Permissions
- Custom Global Permissions with the Manage Node Drivers role assigned.

#### Activating/Deactivating Node Drivers

By default, Rancher only activates drivers for the most popular cloud providers, such as Amazon EC2, Azure, DigitalOcean, Linode and vSphere. If you want to show or hide any node driver, you can change its status.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the left navigation menu, click **Drivers**.
3. On the **Node Drivers** tab, select the driver that you wish to activate or deactivate and click ☰ > **Activate** or ☰ > **Deactivate**.

You can lose access to clusters after deactivating a node driver.

Deactivating a node driver doesn't just affect its visibility in the Rancher UI. When you deactivate or delete a node driver, any nodes deployed with that driver become inaccessible.



For example, if you deactivate a vSphere node driver to hide it in the UI, and you have a vSphere cluster that was deployed with that driver, the initial node in the cluster will fail, and the entire cluster will become inaccessible. Attempts to delete the vSphere nodes will fail, with nodes stuck in an extended `Removing` state.

Before you deactivate a node driver, make sure that it has no associated clusters. One way to check is to see if the respective platform for a driver is listed among your clusters:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Select **Clusters**.
3. Check the **Provider** column of the table for instances of the node driver you are deactivating.

#### Adding Custom Node Drivers

If you want to use a node driver that Rancher doesn't support out-of-the-box, you can add that provider's driver in order to start using them to create node templates and eventually node pools for your Kubernetes cluster.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the left navigation menu, click **Drivers**.
3. On **Node Drivers** tab, click **Add Node Driver**.
4. Complete the **Add Node Driver** form. Then click **Create**.

#### Developing Your Own Node Drivers

Node drivers are implemented with Rancher Machine [https://github.com/rancher/machine], a fork of Docker Machine [https://github.com/docker/machine]. Docker Machine is no longer under active development.

Refer to the original Docker Machine documentation [https://github.com/docker/docs/blob/vnext-engine/machine/overview.md] for details on how to develop your own node drivers.

## UI Server-Side Pagination

Server-Side Pagination (SSP) is a Rancher feature to provide significant performance improvements across the UI for resources with high counts, restricting the amount of resources browser fetches and stores in memory.

The SSP feature flag `ui-sql-cache` is enabled by default and can no longer be disabled through the Rancher UI.

### Disk Space



It is crucial that you review the available disk space on your nodes and plan accordingly

> before upgrading to Rancher v2.12.0 and later to avoid potential disk pressure and pod eviction issues.

The SSP relies on a caching mechanism that introduces a new requirement for ephemeral disk space on your cluster nodes. This cache, an internal SQLite database, is stored within the container's file system. This affects the nodes running the **Rancher server pods** (`rancher` in the `cattle-system` namespace on the local cluster) and the nodes running the **Rancher agent pods** (`cattle-cluster-agent` in the `cattle-system` namespace on all downstream clusters).

The amount of disk space required is dynamic and depends on the quantity and size of Kubernetes resources visualized in the UI. As a guideline, the cache may consume approximately **twice the size of the raw Kubernetes objects** it stores.

For example, internal tests showed that caching 5000 ConfigMaps, totaling 50 MB, consumed 81 MB of disk space. For a conservative, high-level estimate, you can plan for the available disk space on each relevant node to be at least **twice the size of your etcd snapshot**. For most production environments, ensuring a few extra gigabytes of storage are available on the relevant nodes is a safe starting point.

Please note this space counts against ephemeral storage [https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#setting-requests-and-limits-for-local-ephemeral-storage] requests and limits you might have set for your Rancher container via the `resource` value in the Helm chart. Make sure those settings provide for abundant available space.

If you see the error `database or disk is full (13)` in the pod logs, this is a symptom that more space needs to be allocated.

SQLite-backed caching persists copies of any cached Kubernetes objects to disk. See Encrypting SQLite-backed Caching if this is a security concern.

#### Encrypting SQLite-backed Caches

UI server-side pagination persists copies of any cached Kubernetes objects to disk. If you are concerned about the safety of this data, you can encrypt all objects before they are persisted to disk, by setting the environment variable `CATTLE_ENCRYPT_CACHE_ALL` to `true` in `rancher` pods in the upstream cluster and `cattle-cluster-agent` pods in the downstream clusters.

Secrets and security Tokens are always encrypted regardless of the above setting.

#### Known Limitations of UI Server-Side Pagination

This release improves the performance of most pages used to view, create or edit resources within the `local` or downstream clusters i.e. the Cluster Explorer view. However, RBAC related resources and areas outside the Cluster Explorer are not yet covered by this feature.

Additionally, the following limitations are present when the feature is enabled. These mainly revolve around different sort or filter behaviors in affected lists:

- Resources in lists are automatically updated, however, not instantaneously.
- All lists that utilize Server-Side Pagination:
  - `State` column sort and filter features work on the resources `metadata.state.name` field instead of one deduced locally by the UI.
  - Updates are shown every 5 seconds, rather than instantly.

- Cluster Explorer:
  - Project/Namespace filter does not support filtering namespaces by all namespaces not in a project via the `Not in a Project` entry.
  - `Cluster` group -→ `Nodes` page
    - The following columns are not sortable or filterable: `Roles`, `External/Internal IP`, `CPU`, `RAM` (logic to determine their value is calculated in the browser)
  - `Workloads` list:
    - The `Workloads` list, which showed multiple different resource types has been replaced with an overview page.
      - Support for Server-Side Pagination backed lists using multiple resources is not currently possible.
  - `Workloads` group / `Pod` List
    - `Images` is not sortable (sorting on an array).
  - `Service Discovery` group / `Ingresses`
    - `Default` is not sortable/filterable (logic to determine their value is calculated in the browser).
  - `Storage` group / `ConfigMaps`
    - `Data` is not sortable/filterable (logic to determine their value is calculated in the browser).
  - `Storage` group / `Secrets`
    - `Data` is not sortable/filterable (logic to determine their value is calculated in the browser).

# Backup, Restore, and Disaster Recovery

In this section, you'll learn how to create backups of Rancher, how to restore Rancher from backup, and how to migrate Rancher to a new Kubernetes cluster.

The `rancher-backup` operator is used to backup and restore Rancher on any Kubernetes cluster. This application is a Helm chart, and it can be deployed through the Rancher **Apps** page, or by using the Helm CLI. The `rancher-backup` Helm chart is here. [https://github.com/rancher/charts/tree/release-v2.6/charts/rancher-backup]

The backup-restore operator needs to be installed in the local cluster, and only backs up the Rancher app. The backup and restore operations are performed only in the local Kubernetes cluster.

## Backup and Restore for Rancher installed with Docker

For Rancher installed with Docker, refer to xref:[this page] to perform backups and xref:[this page] to perform restores.

## How Backups and Restores Work

The `rancher-backup` operator introduces three custom resources: Backups, Restores, and ResourceSets. The following cluster-scoped custom resource definitions are added to the cluster:

- `backups.resources.cattle.io`
- `resourcesets.resources.cattle.io`

- `restores.resources.cattle.io`

The ResourceSet defines which Kubernetes resources need to be backed up. The ResourceSet is not available to be configured in the Rancher UI because the values required to back up Rancher are predefined. This ResourceSet should not be modified.

When a Backup custom resource is created, the `rancher-backup` operator calls the `kube-apiserver` to get the resources in the ResourceSet that the Backup custom resource refers to.

The operator then creates the backup file in the .tar.gz format and stores it in the location configured in the Backup resource.

When a Restore custom resource is created, the operator accesses the backup .tar.gz file specified by the Restore, and restores the application from that file.

The Backup and Restore custom resources can be created in the Rancher UI, or by using `kubectl apply`.



Refer here for help on restoring an existing backup file into a v1.22 cluster in Rancher v2.6.3.

## Installing the rancher-backup Operator

The `rancher-backup` operator can be installed from the Rancher UI, or with the Helm CLI. In both cases, the `rancher-backup` Helm chart is installed on the Kubernetes cluster running the Rancher server. It is a cluster-admin only feature and available only for the **local** cluster. (*If you do not see `rancher-backup` in the Rancher UI, you may have selected the wrong cluster.*)



There is a known issue in Fleet that occurs after performing a restoration using the backup-restore-operator: Secrets used for clientSecretName and helmSecretName are not included in Fleet gitrepos. Refer SUSE® Rancher Prime: Continuous Delivery Troubleshooting for a workaround.

**Installing rancher-backup with the Rancher UI**

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the `local` cluster and click **Explore**.
3. In the left navigation bar, **Apps › Charts**.
4. Click **Rancher Backups**.
5. Click **Install**.
6. Optional: Configure the default storage location. For help, refer to the configuration section.
7. Click **Install**.

**Result:** The `rancher-backup` operator is installed.

From the **Cluster Dashboard,** you can see the `rancher-backup` operator listed under **Deployments**.

To configure the backup app in Rancher, go to the left navigation menu and click **Rancher Backups**.

#### RBAC

Only the rancher admins and the local cluster's cluster-owner can:

- Install the Chart
- See the navigation links for Backup and Restore CRDs
- Perform a backup or restore by creating a Backup CR and Restore CR respectively
- List backups/restores performed so far

## Backing up Rancher

A backup is performed by creating a Backup custom resource. For a tutorial, refer to this page.

## Restoring Rancher

A restore is performed by creating a Restore custom resource. For a tutorial, refer to this page.

#### Rollback Rancher

Follow the instructions from the Rollbacks page to roll back Rancher to a previously supported version from a backup instance after upgrading.

## Migrating Rancher to a New Cluster

A migration is performed by following these steps.

## Default Storage Location Configuration

Configure a default storage location for your backups. There are various configuration options, such as specifying an S3-compatible object store as the location for individual backups or choosing an existing `StorageClass` during installation of the `backup-restore-operator` Helm chart. You also have the option to override the configured storage location with each backup, but are limited to using an S3-compatible or Minio object store.

For information on configuring these options, refer to this page.

#### Example YAML File: Rancher Backup Helm Chart

The example values.yaml file can be used to configure the `backup-restore-operator` when the Helm CLI is used to install it.

## Backup Restore Usage Guide

The Rancher Backups chart is our solution for disaster recovery and migration. This chart allows you to take backups of your Kubernetes resources and save them to a variety of persistent storage locations.

This chart is a very simple tool which has its hands in many different areas of the Rancher ecosystem. As a result, edge cases have arisen which lead to undocumented functionality. The purpose of this document is to highlight the proper and defined usage for Rancher Backups, as well as discussing some of these edge cases we've run into.

### Functionality Overview

#### Backup

The operator collects all the resources captured by the resourceSet in the chart as in-memory unstructured objects. After the resources have been collected, a compressed tar file of the resources are saved as a collection of manifests in JSON and then uploaded to a user-defined object store. This backup can be put on a repeating schedule and can also be encrypted. This encryption option is important since some of the resources are sensitive and the values are stored in plaintext without encryption.

See the Backup Configuration documentation for more information about the options, including encryption, to configure a backup.



As noted in the Backing up Rancher documentation, you must manually save the encryption configuration file contents since the operator will **not** back it up.

#### Restore

There are two main restore scenarios, restoring to a cluster with Rancher running and restoring to a fresh cluster. You can only restore to a cluster with Rancher running if it's the same cluster the backup was taken from and the `prune` option is enabled during the restore. A restore has similar inputs as a backup. It requires a backup filename, the encryptionConfigSecret name, and the storage location.

Resources are restored in this order:

1. Custom Resource Definitions (CRDs)
2. Cluster-scoped resources
3. Namespaced resources

See the Restore Configuration documentation for more information about the options to configure a restore.

#### Resource Sets

The resourceSet determines which resources the backup-restore-operator collects in a backup. It's a set of ResourceSelectors, which define the selection requirements using key/value matching, regular expression matching, or the Kubernetes client labelSelector.

These are the different fields available for a resourceSelector:

- apiVersion
- excludeKinds
- excludeResourceNameRegexp
- kinds
- kindsRegexp
- labelSelectors
- namespaceRegexp
- namespaces
- resourceNameRegexp

- resourceNames

The Rancher Backups chart contains a default resourceSet [https://github.com/rancher/backup-restore-operator/tree/release/v3.0/charts/rancher-backup/files/default-resourceset-contents], which is a combination of YAML files that are appended to one large resourceSet when the chart is installed. The file order does not matter. The resourceSets may differ between versions.



If you wish to make edits to the resourceSet please edit it **before** installing the chart.

### Proper Usage

This section outlines the guidelines for the proper usage of the Rancher Backups chart according to its use case.

#### All Cases

- Rancher Backups must be installed on the local cluster.
  - NOTE: Rancher Backups does not handle any cluster other than the one it is installed on to. It may restore cluster resources that are on the local cluster but will not communicate with the downstream clusters or back them up.
- The Rancher version being restored to must match the Rancher version from backup.
- The Kubernetes version should be considered since you may be restoring outdated resources (resources deprecated from the version of Kubernetes you are restoring to).

#### Backups

- Some user generated resources will not be backed up unless they can are captured by the default resourceSet or the resourceSet was altered to capture them.
  - We provide a label `resources.cattle.io/backup:true` that when added to a secret in any namespace, will result in it being backed up.
- Backups are non-mutating
- Backups are only of the local cluster

#### Restores

- A restore refers to restoring a backup to the same cluster it was taken from. This can be with Rancher installed (**prune must be enabled**) or with it not installed (no special instructions).
- One thing to note when restoring is that you may find yourself needing to "wipe" the cluster of any Rancher resources. This can be done by deploying the Rancher cleanup script [https://github.com/rancher/rancher-cleanup] as a job to the cluster. This allows you to install Rancher Backups again and restore to a completely fresh cluster.
  - Make sure to use kubectl to deploy the scripts.

#### Migrations

Migration have some more nuance since we are restoring to a different cluster. These are a few things to remember that are commonly missed or forgotten.

- The Rancher domain must be the same when migrating. This means your previous cluster's domain name

must now point to the new cluster.

- Rancher should **not** be running already on the cluster you are migrating to. This can cause many issues with Rancher backups and certain Rancher services as well.
- Install the **same** version of Rancher from the backup **after** the backup has been restored.
- If you choose to provision the new cluster on a different Kubernetes version know that this can cause a wide variety of unsupported behaviors because the Kubernetes API available may be different from the one you took a backup from. This can lead to deprecated resources being restored which will cause issues.
- You should **not** perform any upgrades during a migration.

### Edge Cases and Improper Usage

Below are some examples of some **incorrect** uses or expectations of Rancher Backups.

#### Upgrades

- Using Rancher backups for upgrading Rancher versions is not a valid use case. The recommended procedure is to take a backup of the current version, then upgrade your Rancher instance using these instructions, and then taking **another** backup after the upgrade is complete. This way if the upgrade fails you have a backup to restore to, and the second backup will be valid to restore to the upgraded Rancher version.
- Using Rancher backups for upgrading Kubernetes versions is not a valid use case either. Because the Kubernetes API and available resources are tied to the version, upgrading using backup restore can lead to issues with misaligned sets of resources which may be deprecated, unsupported, or updated. How to upgrade your cluster version will depend on how it was provisioned however the same format as above is recommended (backup, upgrade, backup).

#### ResourceSet

- Because of evolving resources and services from various teams, developers should take note if new resources need to be added to or removed from the default resourceSet.
- Rancher backups only backs up what is captured by the default resourceSets (unless edited). We have added a specific label for user created secrets that will back up a secret of any name and namespace that has said label (see Proper Usage on Backups).

#### Downstream Clusters

- Rancher Backups **only** backs up Kubernetes resources on the local cluster. This means downstream clusters are **not** touched or backed up other than their presence in resources in the local cluster. The updating and communication of downstream clusters falls upon the rancher-agent and rancher-webhook.

#### Restoring Deleted Resources

- Some resources have external results produced, such as provisioning a downstream cluster. Deleting a downstream cluster and restoring the cluster resource on the local cluster does **not** cause Rancher to reprovision said cluster. Depending on the resource, restoring may not fully bring back the resource to an available state.
- The corner case of "restoring a deleted cluster" is **not** a supported feature. When it comes to downstream clusters, whether provisioned or imported, deleting them causes a series of cleanup tasks which doesn't

allow the user to restore the deleted clusters. Provisioned clusters will have their nodes and Rancher-related provisioning resources destroyed, and imported clusters will likely have their Rancher agents and other resources/services related to registration with a local cluster destroyed.



Trying to delete and restore a downstream cluster can lead to a variety of issues with Rancher, Rancher Backups, rancher-webhook, Fleet, and more. It is not recommended to do this.

#### SUSE® Rancher Prime: Continuous Delivery, SUSE Virtualization, and Other Services

Other services, which are backed up by Rancher Backups, often change and evolve. As this happens, their resources and backup needs may change as well. Some resources may not need to be backed up and some may not be backed up at all. It is important for teams to consider this in their development process and assess whether their related resourceSets are correctly capturing the proper set of resources for their services to be restored correctly.

### Monitoring Backups and Restores

Rancher offers out-of-the box monitoring features for the Backup Operator. They are disabled by default but can be easily enabled when deploying the operator Helm Chart.

#### Metrics

Metrics can be enabled by setting `monitoring.metrics.enabled: true` and `monitoring.serviceMonitor.enabled: true` in the Helm Chart values. When enabled, the Operator exports the following metrics. Note that **rancher-monitoring** needs to be previously installed for the metrics to be properly exported.

| Metric Name | Description |
|---|---|
| `rancher_backup_info` | Details on a specific Rancher Backup CR (labels: `name`, `status`, `resourceSetName`, `retentionCount`, `backupType`, `filename`, `storageLocation`, `nextSnapshot`, `lastSnapshot`). Type **GaugeVec**. |
| `rancher_backup_count` | Number of existing Rancher Backup CRs. Type **Gauge**. |
| `rancher_backups_attempted_total` | Total number of Rancher Backups processed by the Operator (labels: `name`). Type **CounterVec**. |
| `rancher_backups_failed_total` | Total number of failed Rancher Backups processed by this operator (labels: `name`). Type **CounterVec**. |
| `rancher_backup_duration_seconds` | Duration of each backup processed by the Operator in seconds (labels: `name`). Type **HistogramVec**, buckets can be customized by the user. |
| `rancher_backup_last_processed_timestamp_seconds` | Unix time of when the last Backup was processed (in seconds) (labels: `name`). Type **GaugeVec**. |
| `rancher_restore_info` | Details on a specific Rancher Restore CR (labels: `name`, `status`, `fileName`, `prune`, `storageLocation`, `restoreTime`). Type **GaugeVec**. |
| `rancher_restore_count` | Number of existing Rancher Restore CRs. Type **Gauge**. |

#### Alerting

Only one alert is provided by default, 'BackupFailed', which warns users when a Backup fails to be processed

by the Operator. It can be enabled by setting `monitoring.prometheusRules.defaultAlert.enabled: true`.

Users can also deploy their own alerting rules by setting `monitoring.prometheusRules.customRules.enabled: true` and defining them under `monitoring.prometheusRules.customRules.rules`.

#### Dashboards

Rancher also provides Grafana dashboards to help monitor the Backup Operator health. These, however, can only be deployed by the **rancher-monitoring** Helm Chart. To do so `rancherBackupMonitoring.dashboards.enabled: true` needs to be set.

#### Conclusion

Rancher Backups is a very useful tool, however it is somewhat limited in its scope and intended purposes. In order to avoid possible difficulties, it is important to follow the specific procedures described to ensure the proper operation of the chart.

## Backing up SUSE Rancher Prime

In this section, you'll learn how to back up Rancher running on any Kubernetes cluster.

The backup-restore operator needs to be installed in the local cluster, and only backs up the Rancher app. The backup and restore operations are performed only in the local Kubernetes cluster.

Note that the rancher-backup operator version 2.x.x is for Rancher v2.6.x.



When restoring a backup into a new Rancher setup, the version of the new setup should be the same as the one where the backup is made. The Kubernetes version should also be considered when restoring a backup, since the supported apiVersion in the cluster and in the backup file could be different.

#### Prerequisites

The Rancher version must be v2.5.0 and up.

Refer here for help on restoring an existing backup file into a v1.22 cluster in Rancher v2.6.3.

#### 1. Install the Rancher Backup operator

The backup storage location is an operator-level setting, so it needs to be configured when the Rancher Backups application is installed or upgraded.

Backups are created as .tar.gz files. These files can be pushed to S3 or Minio, or they can be stored in a persistent volume.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the `local` cluster and click **Explore**. The `local` cluster runs the Rancher server.
3. Click **Apps** › **Charts**.

4. Click **Rancher Backups**.
5. Click **Install**.
6. Configure the default storage location. For help, refer to the storage configuration section.
7. Click **Install**.



There is a known issue in Fleet that occurs after performing a restoration using the backup-restore-operator: Secrets used for clientSecretName and helmSecretName are not included in Fleet gitrepos. Refer here for a workaround.

**2. Perform a Backup**

To perform a backup, a custom resource of type Backup must be created.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the `local` cluster and click **Explore**.
3. In the left navigation bar, click **Rancher Backups** › **Backups**.
4. Click **Create**.
5. Create the Backup with the form, or with the YAML editor.
6. For configuring the Backup details using the form, click **Create** and refer to the configuration reference and to the examples.
7. For using the YAML editor, we can click **Create** › **Create from YAML**. Enter the Backup YAML. This example Backup custom resource would create encrypted recurring backups in S3. The app uses the `credentialSecretNamespace` value to determine where to look for the S3 backup secret:

```yaml
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: s3-recurring-backup
spec:
  storageLocation:
    s3:
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: rancher-backups
      folder: rancher
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
  schedule: "@every 1h"
  retentionCount: 10
```



When creating the Backup resource using YAML editor, the `resourceSetName` must

be set to `rancher-resource-set-full` or `rancher-resource-set-basic`.

For help configuring the Backup, refer to the configuration reference and to the examples.



The `rancher-backup` operator doesn't save the EncryptionConfiguration file. The contents of the EncryptionConfiguration file must be saved when an encrypted backup is created, and the same file must be used when restoring from this backup.

8. Click **Create**.

**Result:** The backup file is created in the storage location configured in the Backup custom resource. The name of this file is used when performing a restore.

#### Backup Configuration

The **Backup: Create** page lets you configure a schedule, enable encryption and specify the storage location for your backups.

You must first install the `rancher-backup` operator. After you do so, you can access the **Backup: Create** page:

##### Schedule

Select the first option to perform a one-time backup, or select the second option to schedule recurring backups. Selecting **Recurring Backups** lets you configure following two fields:

- **Schedule**: This field accepts
  - Standard cron expressions [https://en.wikipedia.org/wiki/Cron], such as `"0 * * * *"`
  - Descriptors, such as `"@midnight"` or `"@every 1h30m"`
- **Retention Count**: This value specifies how many backup files must be retained. If files exceed the given retentionCount, the oldest files will be deleted. The default value is 10.

| YAML Directive Name | Description |
|---|---|
| `schedule` | Provide the cron string for scheduling recurring backups. |
| `retentionCount` | Provide the number of backup files to be retained. |

##### ResourceSet

While you can create your own ResourceSets to back up custom applications, two ResourceSets specifically for backing up Rancher are officially maintained and created by default by the `rancher-backup` operator. We refer to these as `rancher-resource-set-basic` and `rancher-resource-set-full`. The difference between them is whether they include secrets or not in the backups.

`rancher-resource-set-basic` does not include any secrets in the backup files to safeguard confidential information. You are responsible for recording any secrets and redeploying them safely.

`rancher-resource-set-full` includes all essential secrets in the backup files to ensure Rancher continues running smoothly after a restore or migration. To avoid storing sensitive information in plain text, we strongly advise you to enable encryption with a strong key.

| YAML Directive Name | Description |
| --- | --- |
| `resourceSetName` | Provide the name of the ResourceSet to define which resources will be included in this backup. |



Use `rancher-resource-set-full` in production environments.

Starting in Rancher 2.13, Kubernetes secrets serve as the backing store for Public API tokens. As `rancher-resource-set-basic` excludes Kubernetes secrets, this resource set does not save or restore public API tokens. Because Rancher 2.15 updates Kubeconfig resources to rely on public API tokens, `rancher-resource-set-basic` is no longer suitable for production environments.

#### Encryption

The rancher-backup gathers resources by making calls to the kube-apiserver. Objects returned by apiserver are decrypted, so even if encryption at rest [https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/] is enabled, even the encrypted objects gathered by the backup will be in plaintext.

To avoid storing them in plaintext, you can use the same `EncryptionConfiguration` file that was used for at rest encryption, to encrypt certain resources in your backup.



When encrypting objects in the backup you must save the `EncryptionConfiguration` file for future use, because it won't be saved by the rancher-backup operator.

For example, when migrating Rancher to a new cluster the file is used to re-create the secret in the new cluster.

The operator consumes the `EncryptionConfiguration` as a Kubernetes Secret in the `cattle-resources-system` namespace under the key named `encryption-provider-config.yaml` in the secret data.

For the `EncryptionConfiguration`, you can use the sample file provided in the Kubernetes documentation. [https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/#understanding-the-encryption-at-rest-configuration]

To ensure the correct key is used in the secret, the encryption configuration file must be named `encryption-provider-config.yaml`. The below command uses the `--from-file` flag to create the secret with the correct key name.

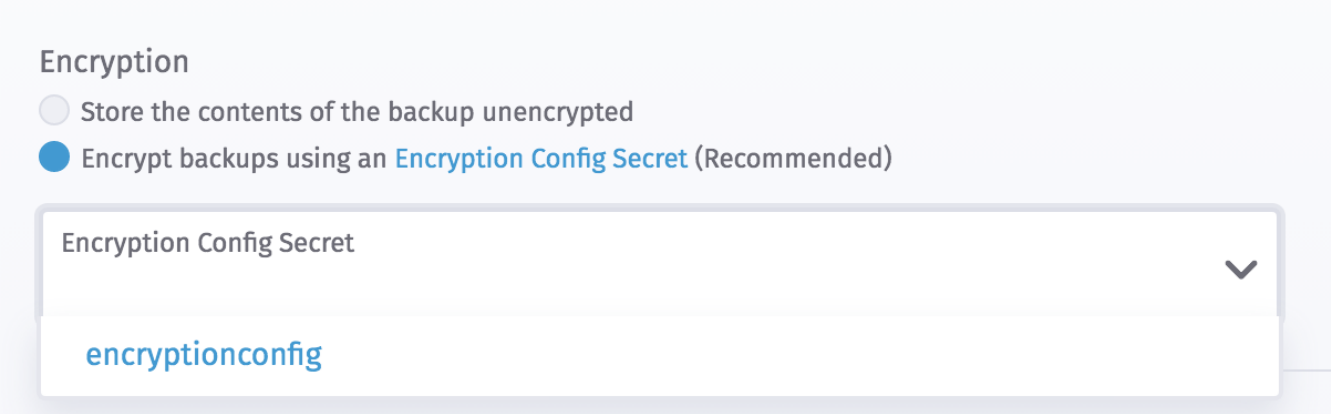
Save the `EncryptionConfiguration` in a file called `encryption-provider-config.yaml` and run this command:

```
kubectl create secret generic encryptionconfig \
  --from-file=./encryption-provider-config.yaml \
  -n cattle-resources-system
```

This will ensure that the secret contains a key named `encryption-provider-config.yaml`, and the operator will use this key to get the encryption configuration.

The `Encryption Config Secret` dropdown will filter out and list only those Secrets that have this exact

key.




In the example command above, the name `encryptionconfig` can be changed to anything.

| YAML Directive Name | Description |
|---|---|
| `encryptionConfigSecretName` | Provide the name of the Secret from `cattle-resources-system` namespace, that contains the encryption config file. |

#### Storage Location

If the StorageLocation is specified in the Backup, the operator will retrieve the backup location from that particular S3 bucket. If not specified, the operator will try to find this file in the default operator-level S3 store, and in the operator-level PVC store. The default storage location is configured during the deployment of the `rancher-backup` operator.

Selecting the first option stores this backup in the storage location configured while installing the rancher-backup chart. The second option lets you configure a different S3 compatible storage provider for storing the backup.

#### S3



If you use an S3 backup target, make sure that every cluster has its own bucket or folder. Rancher populates snapshot information from any available snapshot listed in the S3 bucket or folder configured for that cluster.

The S3 storage location contains the following configuration fields:

1. **Credential Secret** (optional): If you need an AWS access key or secret key to access an S3 bucket, create a secret using your credentials, with keys and directives named `accessKey` and `secretKey`. The secret can be in any namespace. An example secret is here. This directive is unnecessary if the nodes running your operator are in EC2 and assigned IAM permissions to access S3. The **Credential Secret** dropdown lists secrets in all namespaces.
2. **Bucket Name**: The name of the S3 bucket where backup files will be stored.
3. **Region** (optional): The AWS region [https://aws.amazon.com/about-aws/global-infrastructure/regions_az/] where the S3 bucket is located. This field isn't needed for configuring MinIO.
4. **Folder** (optional): The name of the folder in the S3 bucket where backup files will be stored. Nested folders (e.g., `rancher/clusterl`) are not supported. If this field is left empty, the default behavior is to store the backup files in the root folder of the S3 bucket.
5. **Endpoint**: The endpoint [https://docs.aws.amazon.com/general/latest/gr/s3.html] that is used to access S3 in

the region of your bucket.

6. **Endpoint CA** (optional): This should be the Base64 encoded CA cert. For an example, refer to the example S3 compatible configuration.
7. **Skip TLS Verifications** (optional): Set to true if you are not using TLS.

#### YAML Directive Fields

| YAML Directive Name | Description | Required |
|---|---|---|
| `credentialSecretName` | If you need an AWS access key or secret key to access an S3 bucket, create a secret using your credentials, with keys and directives named `accessKey` and `secretKey`. The secret can be in any namespace. An example secret is here. This directive is unnecessary if the nodes running your operator are in EC2 and assigned IAM permissions to access S3. | |
| `credentialSecretNamespace` | The namespace of the secret containing the credentials to access S3. This directive is unnecessary if the nodes running your operator are in EC2 and set up with IAM permissions that allow them to access S3, as described in this section. | |
| `bucketName` | The name of the S3 bucket where backup files will be stored. | ⬔ |
| `folder` | The name of the folder in the S3 bucket where backup files will be stored. Nested folders (e.g., `rancher/cluster1`) are not supported. If this field is left empty, the default behavior is to store the backup files in the root folder of the S3 bucket. | |
| `region` | The AWS region [https://aws.amazon.com/about-aws/global-infrastructure/regions_az/] where the S3 bucket is located. | ⬔ |
| `endpoint` | The endpoint [https://docs.aws.amazon.com/general/latest/gr/s3.html] that is used to access S3 in the region of your bucket. | ⬔ |
| `endpointCA` | This should be the Base64 encoded CA cert. For an example, refer to the example S3 compatible configuration. | |
| `insecureTLSSkipVerify` | Set to true if you are not using TLS. | |

#### Example S3 Storage Configuration

```
s3:
  credentialSecretName: s3-creds
  credentialSecretNamespace: default
  bucketName: rancher-backups
```

```
  folder: rancher
  region: us-west-2
  endpoint: s3.us-west-2.amazonaws.com
```

**Example MinIO Configuration**

```
s3:
  credentialSecretName: minio-creds
  bucketName: rancherbackups
  endpoint: minio.35.202.130.254.xip.io
  endpointCA:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURHakNDQWdLZ0F3SUJBZ0lKQUtpWFZp
NEpBb0J5TUEwR0NTcUdTSWIzRFFFQkN3VUFNQkl4RURBT0JnTlYKQkFNTUIzUmxjMlF0WTJF
d0hoY05NakF3T0RNd0lUZ3lOVFE1V2hjTklqQXhNREklTVRneU5UUTVXakFTTVJBdwpEZlE
VlFRRERBZDBaWE4wTFdOaElJSUJJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBUThBTUlJQkNn
S0NBUUVBCjA4dnV3Q2Y0SEhtR2Q2azVNTmozRW5NOG00T2RpS3czSGszdlNlOUlXQkwyVzY5
WDZxenBhN2I2M3U2L05mMnkKSnZWNDVqeXplRFB6bFJycjlpbEpWaVZlNFNqWlFjdG9jWmFC
aVNsL0xDbEFDdkFaUlYvKzN0TFVTZSs1ZDY0QQpWcUhDQlZObU5xM3E3aVY0TE1aSVpRc3N6
K0FxaUlSd0pOMVVKQTZ6V0tUc2Yzc3ByQ0J2dWxJWmZsVXVETVAyCnRCTCt6cXZEc0pDdWlh
NEEvU2JNT29tVmM2WnNtTGkwMjdub3dGRld3MnRpSkM5d0xMREl4NnJoVHQ4a3VvVHYKQXJp
UjB4WktiRU45LlUzb0l1eUVKbHZyck9YS2ZuUDUwbk8ycGNaQnZCb3pUTStYZnRvQ1d5UnhK
UmI5cFNTRApKQjlmUEFtLzNZcFpMMGRKY2sxRlh3SURBUUFCbzNNd2NUQWRCZ05WSFE0RUZn
UVU5NHU4WXlMdmE2MTJnTlpyCm44QnlFQ2NucVFjdlFnWURWUjBqQkRzd09ZQVU5NHU4WXlM
dmE2MTJnTlpybjhCeUVDY25xUWVoRnFRVUlCSXgKRURBT0JnTlZCQU1NQjNSbGMzUXRZMkdD
QlFDb2wxWXVDUUtBY2pBTUJnTlZIUklFQlRBREFRSC9NQTBHQlNxRwpTSWIzRFFFQkN3VUFB
NElCQVFERlJRZlRtdzdVNXRQRHA5Q2psOXlLRW9Vd2pYWWM2UlAwdmlGSHpubXJ3dUVLCjFr
TkVJNzhBTUwlMEpuS29CY0ljVDNEeGQ3TGdIbTNCRE5mVVh2anArNnZqaXhJYXR2UWhsSFNV
aWIyZjJsSTkKVEMxNzVyNCtROFkzelclRlFXSDdLK08vY3pJTGh5ei93aHRDUlFkQ29lSldX
ZkFiby8wd0VSejZzNkhkVFJzNwpHcWlGNWZtWGp6S0lOcTBjMHRyZ0xtalNKdlhwSnU0ZnNG
OEcyZUh4b2pOKzdJQlFuSkg5cGRIRVpUQUtOL2ppCnIvem04RlZtdlkvdTBndEZneWVQYlZW
bXBqRm03Y0ZOSkc4Y2ZYd0QzcEFwVjhVOGNocTZGeFBHTkVvWFZnclMKYlVRMklaU0RJdlFF
Y3FvSzFKSGdCUWw2RXBaUVpWMWlDRklrdFBwSQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0t
```

**Example credentialSecret**

```
apiVersion: vl
kind: Secret
metadata:
  name: creds
type: Opaque
data:
  accessKey: <base64-encoded access key>
  secretKey: <base64-encoded secret key>
```



To avoid encoding issues, the `credentialSecret` can be created with the below command, updating the values for `accessKey` and `secretKey`.

```
kubectl create secret generic s3-creds \
  --from-literal=accessKey=<access key> \
  --from-literal=secretKey=<secret key>
```

**IAM Permissions for EC2 Nodes to Access S3**

There are two ways to set up the `rancher-backup` operator to use S3 as the backup storage location.

One way is to configure the `credentialSecretName` in the Backup custom resource, which refers to AWS credentials that have access to S3.

If the cluster nodes are in Amazon EC2, the S3 access can also be set up by assigning IAM permissions to the EC2 nodes so that they can access S3.

To allow a node to access S3, follow the instructions in the AWS documentation [https://aws.amazon.com/premiumsupport/knowledge-center/ec2-instance-access-s3-bucket/] to create an IAM role for EC2. When you add a custom policy to the role, add the following permissions, and replace the `Resource` with your bucket name:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
     "Resource": [
        "arn:aws:s3:::rancher-backups"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
         "arn:aws:s3:::rancher-backups/*"
      ]
    }
  ]
```

```
}
```

After the role is created, and you have attached the corresponding instance profile to your EC2 instance(s), the `credentialSecretName` directive can be left empty in the Backup custom resource.

#### Examples

For example Backup custom resources, refer to this page.

### Backup Storage Location Configuration

Configure a storage location where all backups are saved by default. You will have the option to override this with each backup, but will be limited to using an S3-compatible object store.

Only one storage location can be configured at the operator level.

#### Storage Location Configuration

##### No Default Storage Location

You can choose to not have any operator-level storage location configured. If you select this option, you must configure an S3-compatible object store as the storage location for each individual backup.

##### S3-compatible Object Store

| Parameter | Description |
|---|---|
| Credential Secret | Choose the credentials for S3 from your secrets in Rancher. Example. |
| Bucket Name | Enter the name of the S3 bucket [https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html] where the backups will be stored. Default: `rancherbackups`. |
| Region | The AWS region [https://aws.amazon.com/about-aws/global-infrastructure/regions_az/] where the S3 bucket is located. |
| Folder | The folder in the S3 bucket [https://docs.aws.amazon.com/AmazonS3/latest/user-guide/using-folders.html] where the backups will be stored. If this field is left empty, the default behavior is to store the backup files in the root folder of the S3 bucket. |
| Endpoint | The S3 endpoint [https://docs.aws.amazon.com/general/latest/gr/s3.html] For example, `s3.us-west-2.amazonaws.com`. |
| Endpoint CA | The CA cert used to for the S3 endpoint. Default: base64 encoded CA cert |
| insecureTLSSkipVerify | Set to true if you are not using TLS. |

##### Existing StorageClass

Installing the `rancher-backup` chart by selecting the StorageClass option will create a Persistent Volume Claim (PVC), and Kubernetes will in turn dynamically provision a Persistent Volume (PV) where all the backups will be saved by default.

For information about creating storage classes refer to this section.



It is highly recommended to use a StorageClass with a reclaim policy of "Retain".

Otherwise if the PVC created by the `rancher-backup` chart gets deleted (either during app upgrade, or accidentally), the PV will get deleted too, which means all backups saved in it will get deleted. If no such StorageClass is available, after the PV is provisioned, make sure to edit its reclaim policy and set it to "Retain" before storing backups in it.

#### Existing Persistent Volume

Select an existing Persistent Volume (PV) that will be used to store your backups. For information about creating PersistentVolumes in Rancher, refer to this section.



It is highly recommended to use a Persistent Volume with a reclaim policy of "Retain". Otherwise if the PVC created by the `rancher-backup` chart gets deleted (either during app upgrade, or accidentally), the PV will get deleted too, which means all backups saved in it will get deleted.

#### Example YAML File: Rancher Backup Helm Chart

The documented `values.yaml` file that can be used to configure the `backup-restore-operator` when the Helm CLI is used can be found in the backup-restore-operator repository. [https://github.com/rancher/backup-restore-operator/blob/master/charts/rancher-backup/values.yaml]

For more information about `values.yaml` files and configuring Helm charts during installation, refer to the Helm documentation. [https://helm.sh/docs/intro/using_helm/#customizing-the-chart-before-installing]

## Restoring SUSE Rancher Prime

This page outlines how to perform a restore with Rancher.

Follow the instructions from this page when:

- The running Rancher instance is at the same version as when the backup was made.
- The upstream (local) cluster is the same as where the backup was made.



While restoring Rancher on the same setup, the operator will scale down the Rancher deployment when restore starts, and it will scale back up the deployment once restore completes. Rancher will be unavailable during the restore.



*Warning for Fleet users*

You must consider how to handle Kubernetes API server availability and paused GitRepos when restoring a Rancher setup that is running Fleet workloads. See the SUSE® Rancher Prime: Continuous Delivery documentation [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-users/gitrepo-add.html#_backing_up_and_restoring] for more details.



- Follow those steps to migrate Rancher.
- If you need to restore Rancher to a previous version after an upgrade, see the rollback documentation.

#### Create the Restore Custom Resource

A restore is performed by creating a Restore custom resource.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the `local` cluster and click **Explore**. The `local` cluster runs the Rancher server.
3. In the left navigation bar, click **Rancher Backups** › **Restores**.
4. Click **Create**.
5. Create the Restore with the form, or with YAML. For creating the Restore resource using form, refer to the configuration reference and to the examples.
6. For using the YAML editor, we can click **Create** › **Create from YAML**. Enter the Restore YAML.

   ```
   apiVersion: resources.cattle.io/v1
   kind: Restore
   metadata:
     name: restore-migration
   spec:
     backupFilename: backup-b0450532-cee1-4aa1-a881-f5f48a007b1c-2020-
   09-15T07-27-09Z.tar.gz
     encryptionConfigSecretName: encryptionconfig
     storageLocation:
       s3:
         credentialSecretName: s3-creds
         credentialSecretNamespace: default
         bucketName: rancher-backups
         folder: rancher
         region: us-west-2
         endpoint: s3.us-west-2.amazonaws.com
   ```

   For help configuring the Restore, refer to the configuration reference and to the examples.

7. Click **Create**.

**Result:** The rancher-operator scales down the rancher deployment during restore, and scales it back up once the restore completes. The resources are restored in this order:

1. Custom Resource Definitions (CRDs)
2. Cluster-scoped resources
3. Namespaced resources

#### Logs

To check how the restore is progressing, you can check the logs of the operator. Run this command to follow the logs:

```
kubectl logs -n cattle-resources-system -l app.kubernetes.io/name
=rancher-backup -f
```

### Cleanup

If you created the restore resource with kubectl, remove the resource to prevent a naming conflict with future restores.

### Known Issues

In some cases, after restoring the backup, Rancher logs will show errors similar to the following:

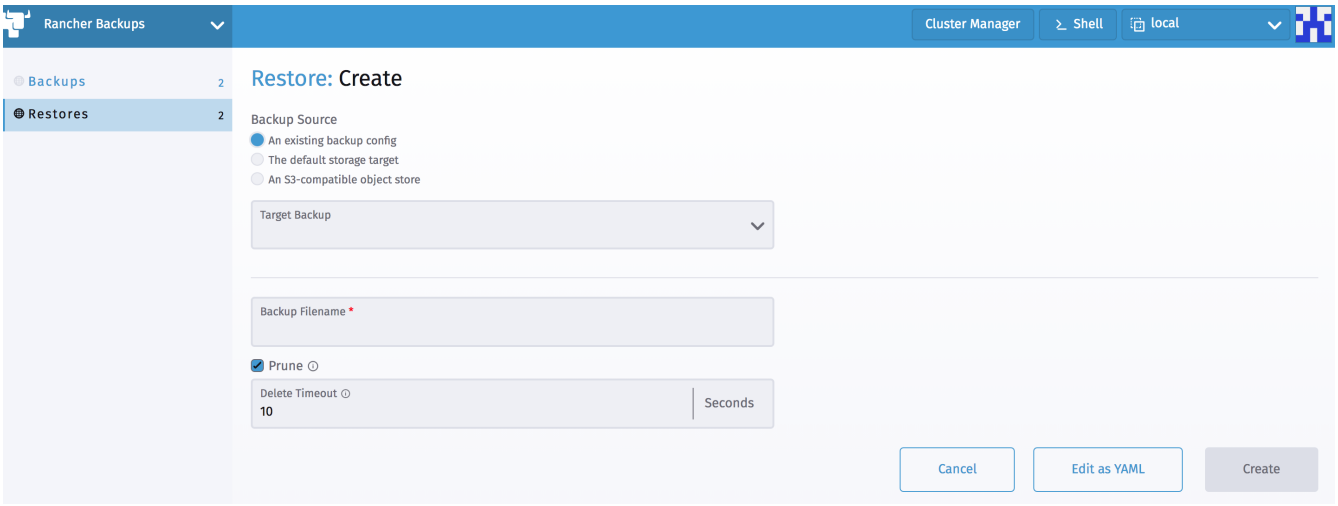
```
2021/10/05 21:30:45 [ERROR] error syncing 'c-89d82/m-4067aa68dd78':
handler rke-worker-upgrader: clusters.management.cattle.io "c-89d82" not
found, requeuing
```

This happens because one of the resources that was just restored has finalizers, but the related resources have been deleted so the handler cannot find it.

To eliminate the errors, we need to find and delete the resource that causes the error. See more information here [https://github.com/rancher/rancher/issues/35050#issuecomment-937968556]

### Restore Configuration

The Restore Create page lets you provide details of the backup to restore from

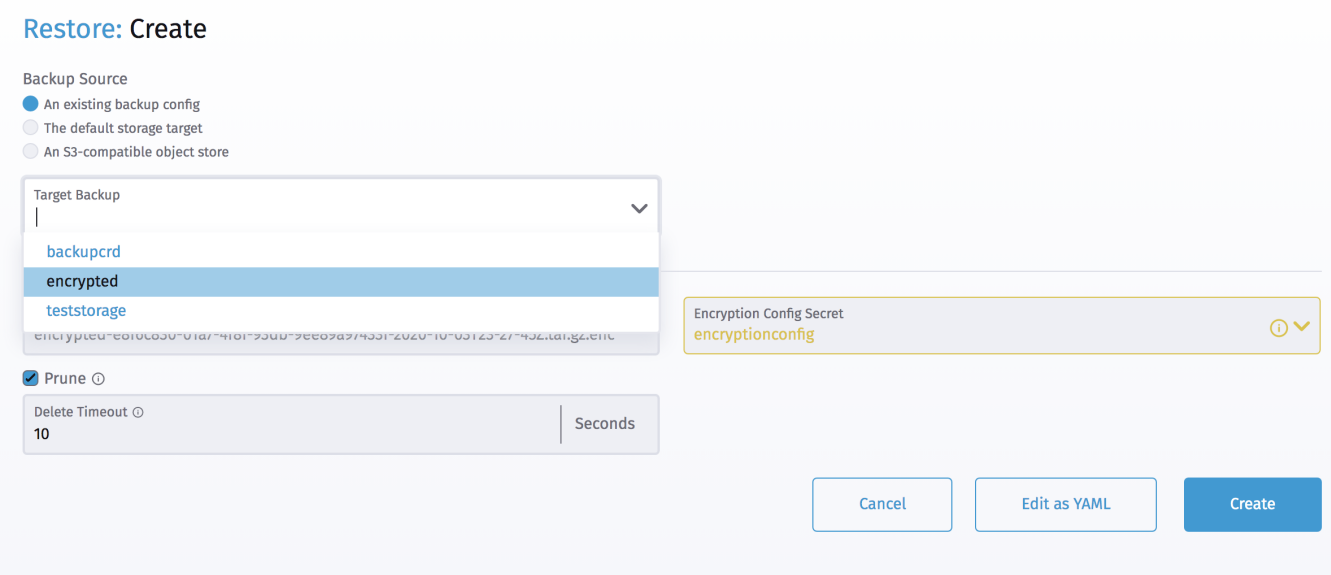



#### Backup Source

Provide details of the backup file and its storage location, which the operator will then use to perform the restore. Select from the following options to provide these details

##### An existing backup config

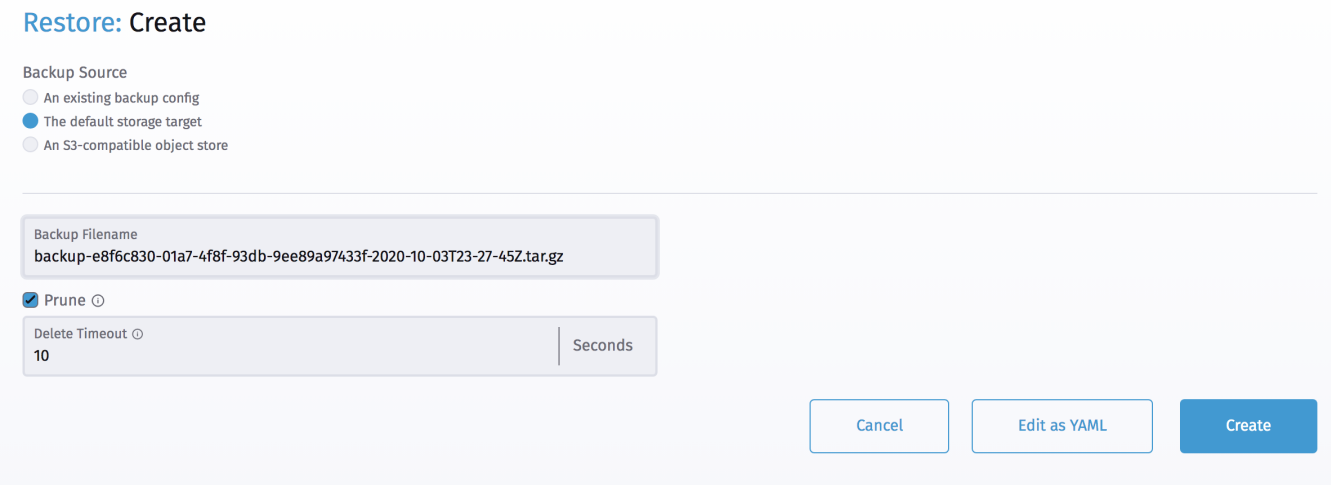
Selecting this option will populate the **Target Backup** dropdown with the Backups available in this cluster. Select the Backup from the dropdown, and that will fill out the **Backup Filename** field for you, and will also pass the backup source information from the selected Backup to the operator.




If the Backup custom resource does not exist in the cluster, you need to get the exact filename and provide the backup source details with the default storage target or an S3-compatible object store.

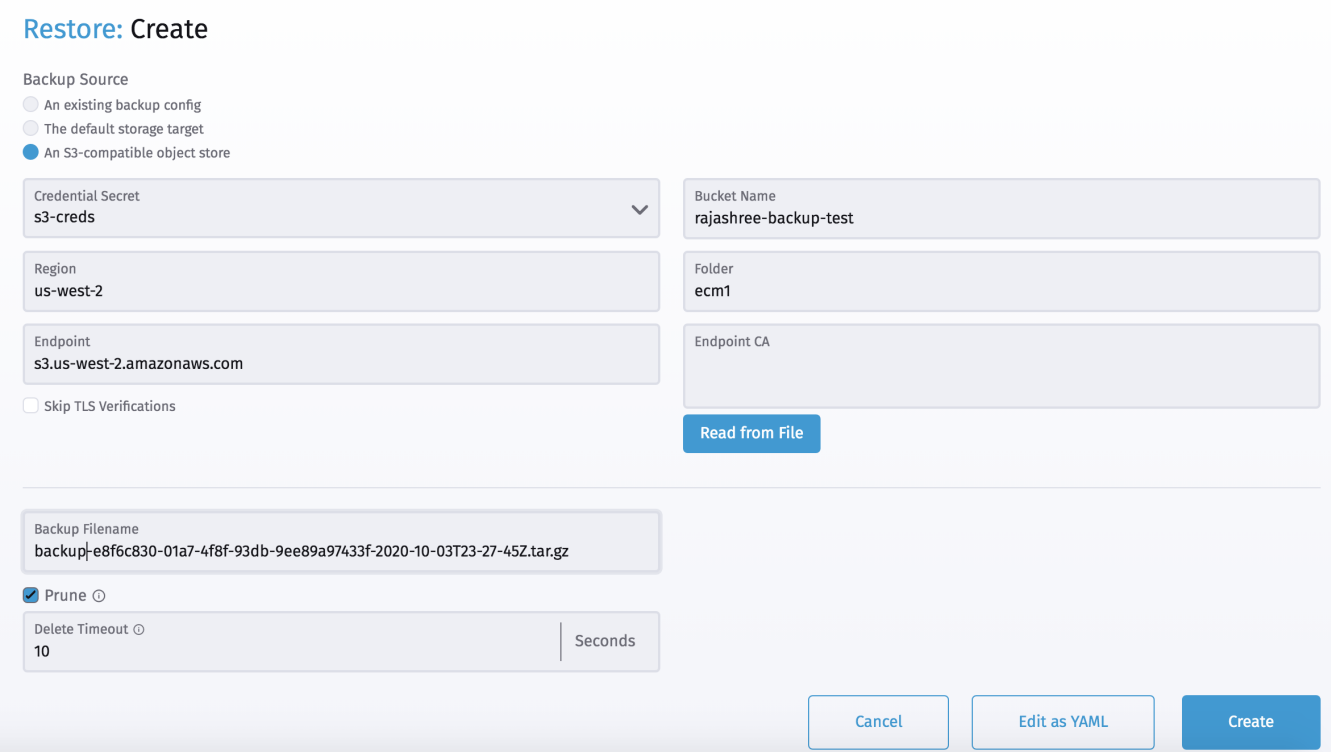
#### The default storage target

Select this option if you are restoring from a backup file that exists in the default storage location configured at the operator-level. The operator-level configuration is the storage location that was configured when the `rancher-backup` operator was installed or upgraded. Provide the exact filename in the **Backup Filename** field.




#### An S3-compatible object store

Select this option if no default storage location is configured at the operator-level, OR if the backup file exists in a different S3 bucket than the one configured as the default storage location. Provide the exact filename in the **Backup Filename** field. Refer to this section for exact steps on getting the backup filename from s3. Fill in all the details for the S3 compatible object store. Its fields are exactly same as ones for the `backup.StorageLocation` configuration in the Backup custom resource.




**Encryption**

If the backup was created with encryption enabled, its file will have `.enc` suffix. Choosing such a Backup, or providing a backup filename with `.enc` suffix will display another dropdown named **Encryption Config Secret**.




The Secret selected from this dropdown must have the same contents as the one used for the Backup custom resource while performing the backup. If the encryption configuration doesn't match, the restore will fail

The `Encryption Config Secret` dropdown will filter out and list only those Secrets that have this exact key

| YAML Directive Name | Description |
|---|---|
| `encryptionConfigSecretName` | Provide the name of the Secret from `cattle-resources-system` namespace, that contains the encryption config file. |



This field should only be set if the backup was created with encryption enabled. Providing the incorrect encryption config will cause the restore to fail.

**Prune During Restore**

- **Prune**: In order to fully restore Rancher from a backup, and to go back to the exact state it was at when the backup was performed, we need to delete any additional resources that were created by Rancher after the backup was taken. The operator does so if the **Prune** flag is enabled. Prune is enabled by default and it is recommended to keep it enabled.

- **Delete Timeout**: This is the amount of time the operator will wait while deleting a resource before editing the resource to remove finalizers and attempt deletion again.

| YAML Directive Name | Description |
|---|---|
| `prune` | Delete the resources managed by Rancher that are not present in the backup (Recommended). |
| `deleteTimeoutSeconds` | Amount of time the operator will wait while deleting a resource before editing the resource to remove finalizers and attempt deletion again. |

#### Getting the Backup Filename from S3

This is the name of the backup file that the `rancher-backup` operator will use to perform the restore.

To obtain this file name from S3, go to your S3 bucket (and folder if it was specified while performing backup).

Copy the filename and store it in your Restore custom resource. So assuming the name of your backup file is `backupfile`,

- If your bucket name is `s3bucket` and no folder was specified, then the `backupFilename` to use will be `backupfile`.
- If your bucket name is `s3bucket` and the base folder is`s3folder`, the `backupFilename` to use is only `backupfile` .
- If there is a subfolder inside `s3Folder` called `s3sub`, and that has your backup file, then the `backupFilename` to use is `s3sub/backupfile`.

| YAML Directive Name | Description |
|---|---|
| `backupFilename` | This is the name of the backup file that the `rancher-backup` operator will use to perform the restore. |

## Migrating SUSE Rancher Prime to a New Cluster

If you are migrating Rancher to a new Kubernetes cluster, you don't need to install Rancher on the new cluster first. If Rancher is restored to a new cluster with Rancher already installed, it can cause problems.

#### Prerequisites

These instructions assume that you have created a backup and already installed a new Kubernetes cluster where Rancher will be deployed. The backup is specific to the Rancher application and can only migrate the Rancher application.



You must use the same hostname that was set as the server URL in the original cluster. If you don't, downstream clusters will show as unavailable in the cluster management page of the UI, and you won't be able to click inside the cluster or on the cluster's **Explore** button.

Rancher version must be v2.5.0 and up

Rancher can be installed on any Kubernetes cluster, including hosted Kubernetes clusters such as Amazon EKS clusters. For help installing Kubernetes, refer to the documentation of the Kubernetes distribution. A

Rancher-created Kubernetes distributions such as, but not limited to, RKE [https://rke.docs.rancher.com/installation] or K3s [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/installation.html] may also be used.

Since Rancher can be installed on any Kubernetes cluster, you can use this backup and restore method to migrate Rancher from one Kubernetes cluster to any other Kubernetes cluster. This method *only* migrates Rancher-related resources and won't affect other applications on the cluster. Refer to the support matrix [https://www.suse.com/lifecycle/] to identify which Kubernetes cluster types and versions are supported for your Rancher version.

#### 1. Install the rancher-backup Helm chart

Install the `rancher-backup chart` [https://github.com/rancher/backup-restore-operator/tags]:

1. Add the Helm repository:

```
helm repo add rancher-charts https://charts.rancher.io
helm repo update
```

2. Set a `CHART_VERSION` variable, selecting a `rancher-backup` chart version compatible with your version of Rancher. See the support matrix [https://www.suse.com/suse-rancher/support-matrix/all-supported-versions], within the **Rancher Apps / Cluster Tools** section, to see which `rancher-backup` versions are supported:

```
CHART_VERSION=<chart-version>
```

3. Install the charts:

```
helm install rancher-backup-crd rancher-charts/rancher-backup-crd -n
cattle-resources-system --create-namespace --version $CHART_VERSION
helm install rancher-backup rancher-charts/rancher-backup -n cattle-
resources-system --version $CHART_VERSION
```



The above assumes an environment with outbound connectivity to Docker Hub.

For an **air-gapped environment**, use the following Helm values to pull the `backup-restore-operator` and `kubectl` images from your private registry when you install the rancher-backup Helm chart.

```
--set image.repository=<registry>/rancher/backup-
restore-operator --set global.kubectl.repository
=<registry>/rancher/kubectl
```

If the host running the Helm commands is **also air-gapped and cannot reach charts.rancher.io**, download the charts on a non-air-gapped host and then install them from local files on the air-gapped host.

On a non-air-gapped host:

```
helm repo add rancher-charts https://charts.rancher.io
helm repo update
helm fetch rancher-charts/rancher-backup-crd --version
$CHART_VERSION
helm fetch rancher-charts/rancher-backup --version
$CHART_VERSION
```

Copy the `rancher-backup-crd-<chart-version>.tgz` and `rancher-backup-<chart-version>.tgz` files to the air-gapped host, then use them to install the charts:

```
helm install rancher-backup-crd ./rancher-backup-crd-
<chart-version>.tgz -n cattle-resources-system --create
-namespace
helm install rancher-backup ./rancher-backup-<chart-
version>.tgz -n cattle-resources-system --set
image.repository=<registry>/rancher/backup-restore-
operator --set global.kubectl.repository
=<registry>/rancher/kubectl
```

**2. Restore from backup using a Restore custom resource**

1. When using S3 object storage as the backup source for a restore that requires credentials, create a `Secret` object in this cluster to add the S3 credentials. The secret data must have two keys - `accessKey`, and `secretKey`, that contain the S3 credentials.

   The secret can be created in any namespace, this example uses the default namespace.

   ```
   kubectl create secret generic s3-creds \
     --from-literal=accessKey=<access key> \
     --from-literal=secretKey=<secret key>
   ```

   Add your access key and secret key as values for `accessKey` and `secretKey` in the command above.

2. Create a `Restore` object:

   During a migration, `prune` must be set to `false`. See the example below:

   ```
   # restore-migration.yaml
   apiVersion: resources.cattle.io/v1
   kind: Restore
   ```

```
metadata:
  name: restore-migration
spec:
  backupFilename: backup-b0450532-cee1-4aa1-a881-f5f48a007b1c-2020-
09-15T07-27-09Z.tar.gz
  // highlight-next-line
  prune: false
  // highlight-next-line
  encryptionConfigSecretName: encryptionconfig
  storageLocation:
    s3:
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: backup-test
      folder: ecml
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
```



The field `encryptionConfigSecretName` should be used only if your backup was created with encryption enabled.

If this applies, provide the name of the `Secret` object containing the encryption config file. If you only have the encryption config file, but don't have the secret created in this cluster, use the following steps to create the secret:

a. Create an encryption configuration file

b. The command below uses a file named `encryption-provider-config.yaml`, with the `--from-file` flag. Run the below once the `EncryptionConfiguration` is saved in a file called `encryption-provider-config.yaml`:

```
kubectl create secret generic encryptionconfig \
  --from-file=./encryption-provider-config.yaml \
  -n cattle-resources-system
```

3. Apply the manifest, and monitor the Restore status:

   a. Apply the `Restore` object resource:

```
kubectl apply -f restore-migration.yaml
```

   b. Watch the Restore status:

```
kubectl get restore
```

c. Watch the restoration logs:

```
kubectl logs -n cattle-resources-system --tail 100 -f -l
app.kubernetes.io/instance=rancher-backup
```

d. Once the Restore resource has the status `Completed`, you can continue the cert-manager and Rancher installation.



When migrating Rancher between any two different Kubernetes distributions (e.g. from K3s to RKE2), the object representing the local cluster has to be modified to allow Rancher to detect the new distribution. After the restoration is completed, and **before** bringing up Rancher on the new cluster, edit the local cluster object:

```
kubectl edit clusters.management.cattle.io local
```

1. Change the value of `status.driver` to `imported`.
2. Remove `status.provider`.
3. Remove the entire `status.version` map.
4. Remove the label with the key `provider.cattle.io` in `metadata.labels`.
5. Remove the annotation with the key `management.cattle.io/current-cluster-controllers-version` in `metadata.annotations`.
6. Remove the entire `spec.rke2Config` or `spec.k3sConfig` map, if present.
7. Save the changes.

Note that removing `spec.rke2Config` or `spec.k3sConfig` will erase your distribution-specific upgrade configuration for the local cluster. It can be reconfigured if the new distribution is configurable for the local cluster.

#### 3. Install cert-manager

Follow the steps to install cert-manager in the documentation about installing cert-manager on Kubernetes.

#### 4. Bring up Rancher with Helm

Use the same version of Helm to install Rancher, that was used on the first cluster.

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=<same hostname as the server URL from the first Rancher
server> \
  --version x.y.z
```



If the original Rancher environment is running, you can collect the current values with a

kubeconfig for the original environment:

```
helm get values rancher -n cattle-system -o yaml > rancher-values.yaml
```

These values can be reused using the `rancher-values.yaml` file. Be sure to switch the kubeconfig to the new Rancher environment.

```
helm install rancher rancher-prime/rancher -n cattle-system -f rancher-values.yaml --version x.y.z
```

**5. Redirect Traffic to the New Cluster**

After migration completes, update your DNS records and any load balancers, so that traffic is routed correctly to the migrated cluster. Remember that you must use the same hostname that was set as the server URL in the original cluster.

Full instructions on how to redirect traffic to the migrated cluster differ based on your specific environment. Refer to your hosting provider's documentation for more details.

**6. Scale Down the Original Rancher Instance**

After redirecting traffic to the new Rancher environment, **scale the original Rancher instance to 0 replicas** so it no longer contacts your managed clusters.

Leaving the old server up can cause agents to keep contacting the original `server-url`, which often leaves clusters stuck in **Updating** in the new environment.

```
kubectl scale deployment rancher -n cattle-system --replicas=0
```



If clusters do not automatically reconnect to the new environment after you have redirected traffic, for example if there is a delay in scaling down the original Rancher instance, you can also restart the `cattle-cluster-agent` pods on each cluster connected to your Rancher environment.

```
kubectl rollout restart deployment cattle-cluster-agent -n cattle-system
```

This triggers a rolling restart of the agent so it re-establishes the connection to the new Rancher environment.

## Backup and Restore Examples

This section contains examples of Backup and Restore custom resources.

The default backup storage location is configured when the `rancher-backup` operator is installed or

upgraded.

Encrypted backups can only be restored if the Restore custom resource uses the same encryption configuration secret that was used to create the backup.

### Backup

This section contains example Backup custom resources.



Refer to the backup config reference page for more information on configuring the options below.

#### Backup in the Default Location with Encryption

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: default-location-encrypted-backup
spec:
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
```

#### Recurring Backup in the Default Location

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: default-location-recurring-backup
spec:
  resourceSetName: rancher-resource-set-basic
  schedule: "@every 1h"
  retentionCount: 10
```

#### Encrypted Recurring Backup in the Default Location

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: default-enc-recurring-backup
spec:
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
  schedule: "@every 1h"
  retentionCount: 3
```

#### Encrypted Backup in Minio

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: minio-backup
spec:
  storageLocation:
    s3:
      credentialSecretName: minio-creds
      credentialSecretNamespace: default
      bucketName: rancherbackups
      endpoint: minio.xip.io
      endpointCA: <base64-encoded-cert>
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
```

#### Backup in S3 Using AWS Credential Secret

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: s3-backup
spec:
  storageLocation:
    s3:
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: rancher-backups
      folder: ecm1
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
```

#### Recurring Backup in S3 Using AWS Credential Secret

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: s3-recurring-backup
spec:
  storageLocation:
    s3:
```

```
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: rancher-backups
      folder: ecml
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
  schedule: "@every 1h"
  retentionCount: 10
```

#### Backup from EC2 Nodes with IAM Permission to Access S3

This example shows that the AWS credential secret does not have to be provided to create a backup if the nodes running `rancher-backup` have these permissions for access to S3.

```
apiVersion: resources.cattle.io/v1
kind: Backup
metadata:
  name: s3-iam-backup
spec:
  storageLocation:
    s3:
      bucketName: rancher-backups
      folder: ecml
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
  resourceSetName: rancher-resource-set-full
  encryptionConfigSecretName: encryptionconfig
```

### Restore

This section contains example Restore custom resources.



Refer to the restore config reference page for more information on configuring the options below.

#### Restore Using the Default Backup File Location

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-default
spec:
  backupFilename: default-location-recurring-backup-752ecd87-d958-4d20-
```

```
8350-072f8d090045-2020-09-26T12-29-54-07-00.tar.gz
#  encryptionConfigSecretName: test-encryptionconfig
```

**Restore for Rancher Migration**

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-migration
spec:
  backupFilename: backup-b0450532-cee1-4aa1-a881-f5f48a007b1c-2020-09-
15T07-27-09Z.tar.gz
  prune: false
  storageLocation:
    s3:
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: rancher-backups
      folder: ecm1
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
```

**Restore from Encrypted Backup**

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-encrypted
spec:
  backupFilename: default-test-s3-def-backup-c583d8f2-6daf-4648-8ead-
ed826c59147l-2020-08-24T20-47-05Z.tar.gz
  encryptionConfigSecretName: encryptionconfig
```

**Restore an Encrypted Backup from Minio**

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-minio
spec:
  backupFilename: default-minio-backup-demo-aa5c04b7-4dba-4c48-9ac4-
ab79168l2eaa-2020-08-30T13-18-17-07-00.tar.gz
  storageLocation:
```

```
    s3:
      credentialSecretName: minio-creds
      credentialSecretNamespace: default
      bucketName: rancherbackups
      endpoint: minio.xip.io
      endpointCA: <base64-encoded-cert>
  encryptionConfigSecretName: test-encryptionconfig
```

**Restore from Backup Using an AWS Credential Secret to Access S3**

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-s3-demo
spec:
  backupFilename: test-s3-recurring-backup-752ecd87-d958-4d20-8350-
072f8d090045-2020-09-26T12-49-34-07-00.tar.gz.enc
  storageLocation:
    s3:
      credentialSecretName: s3-creds
      credentialSecretNamespace: default
      bucketName: rancher-backups
      folder: ecm1
      region: us-west-2
      endpoint: s3.us-west-2.amazonaws.com
  encryptionConfigSecretName: test-encryptionconfig
```

**Restore from EC2 Nodes with IAM Permissions to Access S3**

This example shows that the AWS credential secret does not have to be provided to restore from backup if the nodes running `rancher-backup` have these permissions for access to S3.

```
apiVersion: resources.cattle.io/v1
kind: Restore
metadata:
  name: restore-s3-demo
spec:
  backupFilename: default-test-s3-recurring-backup-84bf8dd8-0ef3-4240-
8ad1-fc7ec308e216-2020-08-24T10#52#44-07#00.tar.gz
  storageLocation:
    s3:
      bucketName: rajashree-backup-test
      folder: ecm1
      region: us-west-2
```

```
    endpoint: s3.us-west-2.amazonaws.com
  encryptionConfigSecretName: test-encryptionconfig
```

**Example EncryptionConfiguration**

The snippets below demonstrate two different types of secrets and their relevance with respect to Backup and Restore of custom resources. Creating the secret can be done with the following command:

```
kubectl create secret generic example-encryptionconfig \
  --from-file=./encryption-provider-config.yaml \
  -n cattle-resources-system
```

The first example is that of a secret used to encrypt the backup files. The backup operator will read the contents of the **encryption-provider-config.yaml** key, which contains the definition of an EncryptionConfiguration resource encoded as Base64.

The second example is that of the Kubernetes EncryptionConfiguration resource itself, being the plain-text form of the Base64-encoded content from the first example. This resource is also used to encrypt secrets when stored in etcd and general cases of Encryption at Rest in Kubernetes. More information on that can be found in the upstream documentation [https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/]. The keys defined in this resource are essential for accessing the contents of encrypted Backups, particularly during the Restore process. For that reason, these must be kept secure, confidential and easily retrievable.

```
apiVersion: v1
data:
  encryption-provider-config.yaml:
YXBpVmVyc2lvbjogYXBpc2VydmVyLmNvbmZpZy5rOHMuaW8vdjEKa2luZDogRW5jcnlwdGlv
bkNvbmZpZ3VyYXRpb24KcmVzb3VyY2VzOgogIC0gcmVzb3VyY2VzOgogICAgICAtIHNlY3Jl
dHMKICAgIHByb3ZpZGVyczoKICAgICAgLSBhZXNnY206CiAgICAgICAgICBrZXlzOgogICAg
ICAgICAgICAtIG5hbWU6IGtleTEKICAgICAgICAgICAgICBzZWNyZXQ6IGMyVmpjbVYwSUds
eklITmxZMlZ5WlE9PQogICAgICAgICAgICAtIG5hbWU6IGtleTIKICAgICAgICAgICAgICBz
ZWNyZXQ6IGRHaHBjeUJwY3lCdllYTnpkMjl5WkE9PQogICAgICAtIGFlc2NiYzoKICAgICAg
ICAgIGtleXM6CiAgICAgICAgICAgIC0gbmFtZToga2V5MQogICAgICAgICAgICAgIHNlY3Jl
dDogYzJWamNtVjBJR2x6SUhObFkzVnlaUT09CiAgICAgICAgICAgIC0gbmFtZToga2V5Mgog
ICAgICAgICAgICAgIHNlY3JldDogZEdocGN5QnBjeUJ3WVhOemQyOXlaQT09CiAgICAgIC0g
c2VjcmV0Ym94OgogICAgICAgICAga2V5czoKICAgICAgICAgICAgLSBuYW1lOiBrZXkxCiAg
ICAgICAgICAgICAgc2VjcmV0OiBZV0pqWkdWbVoyaHBhbXRzYlc1dmNIRnljMlIxZG5kNGVY
b3hNak0wTlRZPQ==
kind: Secret
metadata:
  name: example-encryptionconfig
  namespace: cattle-resources-system
type: Opaque
```

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
  - resources:
      - secrets
    providers:
      - aesgcm:
          keys:
            - name: key1
              secret: c2VjcmV0IGlzIHNlY3VyZQ==
            - name: key2
              secret: dGhpcyBpcyBwYXNzd29yZA==
      - aescbc:
          keys:
            - name: key1
              secret: c2VjcmV0IGlzIHNlY3VyZQ==
            - name: key2
              secret: dGhpcyBpcyBwYXNzd29yZA==
      - secretbox:
          keys:
            - name: key1
              secret: YWJjZGVmZ2hpamtsbW5vcHFyc3R1dnd4eXoxMjM0NTY=
```

# CLI

## kubectl Utility

### kubectl

Interact with Rancher using kubectl.

#### kubectl Utility

Install the `kubectl` utility. See install kubectl [https://kubernetes.io/docs/tasks/tools/install-kubectl/].

Configure kubectl by visiting your cluster in the Rancher Web UI, clicking on `Kubeconfig`, copying contents, and putting them into your `~/.kube/config` file.

Run `kubectl cluster-info` or `kubectl get pods` successfully.

#### Authentication with kubectl and kubeconfig Tokens with TTL

*Requirements*

If admins have kubeconfig token generation turned off, the kubeconfig file requires the Rancher CLI to be present in your PATH when you run `kubectl`. Otherwise, you'll see an error like: `Unable to connect to the server: getting credentials: exec: exec: "rancher": executable file not`

`found in $PATH`.

This feature enables kubectl to authenticate with the Rancher server and get a new kubeconfig token when required. The following auth providers are currently supported:

1. Local
2. Active Directory (LDAP only)
3. FreeIPA
4. OpenLDAP
5. SAML providers: Ping, Okta, ADFS, Keycloak, Shibboleth
6. Microsoft Entra ID

When you first run kubectl, for example, `kubectl get pods`, you are prompted to pick an auth provider and log in with the Rancher server. The kubeconfig token is cached in the path where you run kubectl under `./.cache/token`. This token is valid until it expires, or gets deleted from the Rancher server. Upon expiration, you must log in with the Rancher server again to run the `kubectl get pods` command.

## SUSE Rancher Prime CLI

The Rancher CLI (Command Line Interface) is a unified tool that you can use to interact with Rancher. With this tool, you can operate Rancher using a command line rather than the GUI.

### Download Rancher CLI

The binary can be downloaded directly from the UI.

1. In the upper left corner, click ☰.
2. At the bottom of the navigation sidebar menu, click **About**.
3. Under the **CLI Downloads section**, there are links to download the binaries for Windows, Mac, and Linux. You can also check the releases page for our CLI [https://github.com/rancher/cli/releases] for direct downloads of the binary.

### Requirements

After you download the Rancher CLI, you need to make a few configurations. Rancher CLI requires:

- Your Rancher Server URL, which is used to connect to Rancher Server.
- An API Bearer Token, which is used to authenticate with Rancher. For more information about obtaining a Bearer Token, see Creating an API Key.

### CLI Authentication

Before you can use Rancher CLI to control your Rancher Server, you must authenticate using an API Bearer Token. Log in using the following command (replace `<BEARER_TOKEN>` and `<SERVER_URL>` with your information):

```
$ ./rancher login https://<SERVER_URL> --token <BEARER_TOKEN>
```

If Rancher Server uses a self-signed certificate, Rancher CLI prompts you to continue with the connection.

#### Project Selection

Before you can perform any commands, you must select a Rancher project to perform those commands against. To select a project to work on, use the command `./rancher context switch`. When you enter this command, a list of available projects displays. Enter a number to choose your project.

**Example: `./rancher context switch` Output**

```
User:rancher-cli-directory user$ ./rancher context switch
NUMBER    CLUSTER NAME   PROJECT ID              PROJECT NAME
1         cluster-2      c-7q96s:p-h4tmb         project-2
2         cluster-2      c-7q96s:project-j6z6d   Default
3         cluster-1      c-lchzv:p-xbpdt         project-1
4         cluster-1      c-lchzv:project-s2mch   Default
Select a Project:
```

After you enter a number, the console displays a message that you've changed projects.

```
INFO[0005] Setting new context to project project-1
INFO[0005] Saving config to /Users/markbishop/.ranchcli2.json
```

Ensure you can run `rancher kubectl get pods` successfully.

#### Commands

The following commands are available for use in Rancher CLI.

| Command | Result |
|---|---|
| `clusters, [cluster]` | Performs operations on your clusters. |
| `context` | Switches between Rancher projects. For an example, see Project Selection. |
| `inspect [OPTIONS] [RESOURCEID RESOURCENAME]` | Displays details about Kubernetes resources [https://kubernetes.io/docs/reference/kubectl/cheatsheet/#resource-types] or Rancher resources (i.e.: projects and workloads). Specify resources by name or ID. |
| `kubectl` | Runs kubectl commands [https://kubernetes.io/docs/reference/kubectl/overview/#operations]. |
| `login, [l]` | Logs into a Rancher Server. For an example, see CLI Authentication. |
| `machines, [machine]` | Performs operations on machines. |
| `namespaces, [namespace]` | Performs operations on namespaces. |
| `nodes, [node]` | Performs operations on nodes. |
| `projects, [project]` | Performs operations on projects. |
| `ps` | Displays workloads in a project. |

| Command | Result |
|---|---|
| `server` | Performs operations for the server. |
| `settings, [setting]` | Shows the current settings for your Rancher Server. |
| `ssh` | Connects to one of your cluster nodes using the SSH protocol. |
| `up` | Applies compose config. |
| `token` | Authenticates and generates new kubeconfig token. |
| `help, [h]` | Shows a list of commands or help for one command. |

#### Rancher CLI Help

Once logged into Rancher Server using the CLI, enter `./rancher --help` for a list of commands.

All commands accept the `--help` flag, which documents each command's usage.

#### Limitations

The Rancher CLI **cannot** be used to install dashboard apps or Rancher feature charts.

## Enabling Experimental Features

Rancher includes some features that are experimental and disabled by default. You might want to enable these features, for example, if you decide that the benefits of using an unsupported storage type outweighs the risk of using an untested feature. Feature flags were introduced to allow you to try these features that are not enabled by default.

The features can be enabled in three ways:

- Enable features when starting Rancher. When installing Rancher with a CLI, you can use a feature flag to enable a feature by default.
- Enable features from the Rancher UI by going to the **Settings** page.
- Enable features with the Rancher API after installing Rancher.

Each feature has two values:

- A default value, which can be configured with a flag or environment variable from the command line
- A set value, which can be configured with the Rancher API or UI

If no value has been set, Rancher uses the default value.

Because the API sets the actual value and the command line sets the default value, that means that if you enable or disable a feature with the API or UI, it will override any value set with the command line.

For example, if you install Rancher, then set a feature flag to true with the Rancher API, then upgrade Rancher with a command that sets the feature flag to false, the default value will still be false, but the feature will still be enabled because it was set with the Rancher API. If you then deleted the set value (true) with the Rancher API, setting it to NULL, the default value (false) would take effect. See the feature flags page for more information.

## Enabling Features when Starting Rancher

When you install Rancher, enable the feature you want with a feature flag. The command is different depending on whether you are installing Rancher on a single node or if you are doing a Kubernetes Installation of Rancher.

### Enabling Features for Kubernetes Installs



Values set from the Rancher API will override the value passed in through the command line.

When installing Rancher with a Helm chart, use the `--set` option. In the below example, two features are enabled by passing the feature flag names in a comma separated list:

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  --set hostname=rancher.my.org \
  --set 'extraEnv[0].name=CATTLE_FEATURES'
  --set 'extraEnv[0].value=<FEATURE-FLAG-NAME-1>=true\,<FEATURE-FLAG-
NAME-2>=true'
```

### Enabling Features for Air Gap Installs

To perform an air gap installation of Rancher, add a Helm chart repository and download a Helm chart, then install Rancher with Helm.

When you install the Helm chart, you should pass in feature flag names in a comma separated list, as in the following example:

```
helm install rancher ./rancher-<VERSION>.tgz \
  --namespace cattle-system
  --set hostname=<RANCHER.YOURDOMAIN.COM> \
  --set image.registry=<REGISTRY.YOURDOMAIN.COM:PORT> \
  --set ingress.tls.source=secret \
  --set systemDefaultRegistry=<REGISTRY.YOURDOMAIN.COM:PORT> \ # Set a
default private registry to be used in Rancher
  --set useBundledSystemChart=true # Use the packaged Rancher system
charts
  --set 'extraEnv[0].name=CATTLE_FEATURES'
  --set 'extraEnv[0].value=<FEATURE-FLAG-NAME-1>=true\,<FEATURE-FLAG-
NAME-2>=true'
```

## Enabling Features with the Rancher UI

1. In the upper left corner, click ☰ > **Global Settings**.
2. Click **Feature Flags**.

3. To enable a feature, go to the disabled feature you want to enable and click ⋮ > **Activate**.

**Result:** The feature is enabled.

#### Disabling Features with the Rancher UI

1. In the upper left corner, click ☰ > **Global Settings**.
2. Click **Feature Flags**. You will see a list of experimental features.
3. To disable a feature, go to the enabled feature you want to disable and click ⋮ > **Deactivate**.

**Result:** The feature is disabled.

### Enabling Features with the Rancher API

1. Go to `<RANCHER-SERVER-URL>/v3/features`.
2. In the `data` section, you will see an array containing all of the features that can be turned on with feature flags. The name of the feature is in the `id` field. Click the name of the feature you want to enable.
3. In the upper left corner of the screen, under **Operations,** click **Edit**.
4. In the **Value** drop-down menu, click **True**.
5. Click **Show Request**.
6. Click **Send Request**.
7. Click **Close**.

**Result:** The feature is enabled.

#### Disabling Features with the Rancher API

1. Go to `<RANCHER-SERVER-URL>/v3/features`.
2. In the `data` section, you will see an array containing all of the features that can be turned on with feature flags. The name of the feature is in the `id` field. Click the name of the feature you want to enable.
3. In the upper left corner of the screen, under **Operations,** click **Edit**.
4. In the **Value** drop-down menu, click **False**.
5. Click **Show Request**.
6. Click **Send Request**.
7. Click **Close**.

**Result:** The feature is disabled.

### Continuous Delivery

Continuous Delivery with Fleet comes preinstalled in Rancher and can't be fully disabled. However, the Fleet feature for GitOps continuous delivery may be disabled using the `continuous-delivery` feature flag.

To enable or disable this feature, refer to the instructions on the main page about enabling experimental features.

| Environment Variable Key | Default Value | Description |
|---|---|---|
| `continuous-delivery` | `true` | This flag disables the GitOps continuous delivery feature of Fleet. |

If Fleet was disabled in Rancher v2.5.x, it will become enabled if Rancher is upgraded to v2.6.x. Only the continuous delivery part of Fleet can be disabled. When `continuous-delivery` is disabled, the `gitjob` deployment is no longer deployed into the Rancher server's local cluster, and `continuous-delivery` is not shown in the Rancher UI.

### RoleTemplate Aggregation



RoleTemplate aggregation is an experimental feature in v2.13 that changes the RBAC architecture used for RoleTemplates, ClusterRoleTemplateBindings and ProjectRoleTemplateBindings. **It is not supported for production environments**. Breaking changes may occur between v2.13 and v2.14.

RoleTemplate aggregation implements RoleTemplates, ClusterRoleTemplateBindings and ProjectRoleTemplateBindings using the Kubernetes feature Aggregated ClusterRoles [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#aggregated-clusterroles]. The new architecture results in a net reduction in RBAC objects (Roles, RoleBindings, ClusterRoles and ClusterRoleBindings) both in the Rancher cluster and the downstream clusters.

For more information on how the feature can improve scalability and performance, please see the Rancher Blog post [https://www.suse.com/c/rancher_blog/fewer-bindings-more-power-ranchers-rbac-boost-for-enhanced-performance-and-scalability/].

| Environment Variable Key | Default Value | Description |
|---|---|---|
| `aggregated-roletemplates` | `false` | [Beta] Make RoleTemplates use aggregation for generated RBAC roles. |

The value of this feature flag is locked on installation, which shows up in the UI as a lock symbol beside the feature flag. That means the feature can only be set on the first ever installation of Rancher. After that, attempts to modify the value is denied.

### UI for Istio Virtual Services and Destination Rules

This feature enables a UI that lets you create, read, update and delete virtual services and destination rules, which are traffic management features of Istio.

> **Prerequisite:** Turning on this feature does not enable Istio. A cluster administrator needs to enable Istio for the cluster in order to use the feature.

To enable or disable this feature, refer to the instructions on the main page about enabling experimental features.

| Environment Variable Key | Default Value | Status | Available as of |
|---|---|---|---|
| `istio-virtual-service-ui` | `false` | Experimental | v2.3.0 |
| `istio-virtual-service-ui` | `true` | GA | v2.3.2 |

#### About this Feature

A central advantage of Istio's traffic management features is that they allow dynamic request routing, which is useful for canary deployments, blue/green deployments, or A/B testing.

When enabled, this feature turns on a page that lets you configure some traffic management features of Istio using the Rancher UI. Without this feature, you need to use `kubectl` to manage traffic with Istio.

The feature enables two UI tabs: one tab for **Virtual Services** and another for **Destination Rules**.

- **Virtual services** intercept and direct traffic to your Kubernetes services, allowing you to direct percentages of traffic from a request to different services. You can use them to define a set of routing rules to apply when a host is addressed. For details, refer to the Istio documentation. [https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/]
- **Destination rules** serve as the single source of truth about which service versions are available to receive traffic from virtual services. You can use these resources to define policies that apply to traffic that is intended for a service after routing has occurred. For details, refer to the Istio documentation. [https://istio.io/docs/reference/config/networking/v1alpha3/destination-rule]

To see these tabs,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where Istio is installed and click **Explore**.
3. In the left navigation bar, click **Istio**.
4. You will see tabs for **Kiali** and **Jaeger**. From the left navigation bar, you can view and configure **Virtual Services** and **Destination Rules**.

### Allowing Unsupported Storage Drivers

This feature allows you to use types for storage providers and provisioners that are not enabled by default.

To enable or disable this feature, refer to the instructions on the main page about enabling experimental features.

| Environment Variable Key | Default Value | Description |
|---|---|---|
| `unsupported-storage-drivers` | `false` | This feature enables types for storage providers and provisioners that are not enabled by default. |

#### Types for Persistent Volume Plugins that are Enabled by Default

Below is a list of storage types for persistent volume plugins that are enabled by default. When enabling this feature flag, any persistent volume plugins that are not on this list are considered experimental and unsupported:

| Name | Plugin |
|---|---|
| Amazon EBS Disk | `aws-ebs` |
| AzureFile | `azure-file` |
| AzureDisk | `azure-disk` |
| Google Persistent Disk | `gce-pd` |

| Name | Plugin |
| --- | --- |
| VMware vSphere Volume | `vsphere-volume` |
| Local | `local` |
| Network File System | `nfs` |
| hostPath | `host-path` |

### Types for StorageClass that are Enabled by Default

Below is a list of storage types for a StorageClass that are enabled by default. When enabling this feature flag, any persistent volume plugins that are not on this list are considered experimental and unsupported:

| Name | Plugin |
| --- | --- |
| Amazon EBS Disk | `aws-ebs` |
| AzureFile | `azure-file` |
| AzureDisk | `azure-disk` |
| Google Persistent Disk | `gce-pd` |
| VMware vSphere Volume | `vsphere-volume` |
| Local | `local` |

# Cluster Deployment

Rancher simplifies the creation of clusters by allowing you to create them through the Rancher UI rather than more complex alternatives. Rancher provides multiple options for launching a cluster. Use the option that best fits your use case.

This section assumes a basic familiarity with Docker and Kubernetes. For a brief explanation of how Kubernetes components work together, refer to the concepts page.

For a conceptual overview of how the Rancher server provisions clusters and what tools it uses to provision them, refer to the architecture page.

## Cluster Management Capabilities by Cluster Type

The options and settings available for an existing imported cluster change based on the method that you used to provision it.

The following table summarizes the options and settings available for Rancher launched or existing imported cluster types:

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Using kubectl and a kubeconfig file to Access a Cluster | ⬔ | ⬔ | ⬔ | ⬔ |
| Managing Cluster Members | ⬔ | ⬔ | ⬔ | ⬔ |
| Editing and Upgrading Clusters | ⬔ | ⬔ | ⬔ | ⬔[2] |
| Managing Nodes | ⬔ | ⬔ | ⬔ | ⬔[3] |
| Managing Persistent Volumes and Storage Classes | ⬔ | ⬔ | ⬔ | ⬔ |
| Managing Projects, Namespaces and Workloads | ⬔ | ⬔ | ⬔ | ⬔ |
| Using App Catalogs | ⬔ | ⬔ | ⬔ | ⬔ |
| Configuring Tools (Alerts, Notifiers, Monitoring, Logging, Istio) | ⬔ | ⬔ | ⬔ | ⬔ |
| Running Security Scans | ⬔ | ⬔ | ⬔ | ⬔ |
| Ability to rotate certificates | ⬔ | ⬔ | | |
| Ability to backup and restore Rancher-launched clusters | ⬔ | ⬔ | | ⬔[4] |

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Cleaning Kubernetes components when clusters are no longer reachable from Rancher | ⬔ | | | |

1. Registered EKS, GKE and AKS clusters have the same options available as EKS, GKE and AKS clusters created from the Rancher UI. The difference is that when a registered cluster is deleted from the Rancher UI, it is not destroyed.
2. Cluster configuration options can't be edited for registered clusters, except for K3s and RKE2 clusters.
3. For registered cluster nodes, the Rancher UI exposes the ability to cordon, drain, and edit the node.
4. For registered clusters using etcd as a control plane, snapshots must be taken manually outside of the Rancher UI to use for backup and recovery.

# Setting up Clusters in a Hosted Kubernetes Provider

In this scenario, Rancher does not provision Kubernetes because it is installed by providers such as Google Kubernetes Engine (GKE), Amazon Elastic Container Service for Kubernetes, or Azure Kubernetes Service.

If you use a Kubernetes provider such as Google GKE, Rancher integrates with its cloud APIs, allowing you to create and manage role-based access control for the hosted cluster from the Rancher UI.

For more information, refer to the section on hosted Kubernetes clusters.

# Launching Kubernetes with Rancher

Rancher uses RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html] or K3s [https://documentation.suse.com/cloudnative/k3s/latest/en/introduction.html] as a library when provisioning Kubernetes on your own nodes. RKE2 is Rancher's own lightweight Kubernetes installer.

In RKE2 clusters, Rancher manages the deployment of Kubernetes. These clusters can be deployed on any bare metal server, cloud provider, or virtualization platform.

If you already have a node that you want to add to an RKE2 cluster, you can add it to the cluster by running a Rancher RKE2 agent container on it.

For more information, refer to Launching Kubernetes with Rancher.

### Launching Kubernetes Using SUSE® Rancher Prime: Cluster API

SUSE® Rancher Prime: Cluster API integrates Cluster API (CAPI) capabilities into SUSE Rancher Prime, enabling Kubernetes-native cluster lifecycle management. By leveraging custom resources and `ClusterClasses`, you can automate and standardize cluster provisioning across infrastructure providers directly within Rancher.

For detailed steps on configuring your CAPI environment and provisioning a cluster, refer to the Create & Import Your First Cluster [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/tutorials/first-cluster.html] tutorial.

### Launching Kubernetes and Provisioning Nodes in an Infrastructure Provider

Rancher can dynamically provision nodes in infrastructure providers such as Amazon EC2, DigitalOcean, Azure, or vSphere, then install Kubernetes on them.

One benefit of using nodes hosted by an infrastructure provider is that if a node loses connectivity with the cluster, Rancher can automatically replace it, thus maintaining the expected cluster configuration.

For more information, refer to Launching Kubernetes on New Nodes in an Infrastructure Provider

### Launching Kubernetes on Existing Custom Nodes

When setting up this type of cluster, Rancher installs Kubernetes on existing custom nodes, which creates a custom cluster.

You can bring any nodes you want to Rancher and use them to create a cluster.

These nodes include on-prem bare metal servers, cloud-hosted virtual machines, or on-prem virtual machines.

## Registering Existing Clusters

The cluster registration feature replaces the feature to import clusters.

Registering EKS clusters now provides additional benefits. For the most part, registered EKS clusters and EKS clusters created in Rancher are treated the same way in the Rancher UI, except for deletion.

When you delete an EKS cluster that was created in Rancher, the cluster is destroyed. When you delete an EKS cluster that was registered in Rancher, it is disconnected from the Rancher server, but it still exists and you can still access it in the same way you did before it was registered in Rancher.

For more information, refer to Registering Existing Clusters.

## Programmatically Creating Clusters

The most common way to programmatically deploy Kubernetes clusters through Rancher is by using the Rancher2 Terraform provider. Refer to the documentation for creating clusters with Terraform [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster].

EKS, GKE, and AKS clusters can be created or imported with Terraform.

## Node Requirements for SUSE Rancher Prime Managed Clusters

This page describes the requirements for the Rancher managed Kubernetes clusters where your apps and services will be installed. These downstream clusters should be separate from the three-node cluster running Rancher.



If Rancher is installed on a high-availability Kubernetes cluster, the Rancher server three-node cluster and downstream clusters have different requirements. For Rancher installation requirements, refer to the node requirements in the installation section.

## Operating Systems and Container Runtime Requirements

To provision downstream Kubernetes clusters with Rancher, all host nodes must meet specific operating system, CPU architecture, and container runtime requirements.

### Operating Systems and CPU Architectures

Control plane and etcd nodes in downstream Kubernetes clusters managed by Rancher must run Linux. Worker nodes may run either Linux or Windows Server.

- **Linux Nodes**: Required for `etcd` and `controlplane` nodes. Optional for `worker` nodes.
  - **Supported Architectures**: `x86_64`, `ARM64`, or mixed-architecture clusters.
- **Windows Server Nodes**: Supported for `worker` nodes only (see Windows Server Worker Nodes).
  - **Supported Architectures**: `x86_64` only.

For a complete list of OS versions tested and validated with this release, see the Support Maintenance Terms [https://rancher.com/support-maintenance-terms/].



ARM64 and mixed-architecture support applies exclusively to Linux nodes. Windows Server worker nodes do not support ARM64.

### Container Runtime Requirements

Each host node requires a Container Runtime Interface (CRI)-compliant runtime to run cluster components and workloads. The required runtime depends on your Kubernetes engine:

- **RKE2 & K3s Clusters**: Runtimes (typically `containerd`) come pre-packaged with the installation binaries. No prior runtime installation is required on the host.
- **Hosted / Registered Clusters (EKS, GKE, AKS, etc.)**: Use the cloud provider's default CRI-compliant runtime (such as `containerd` or `CRI-O`).

### Oracle Linux and RHEL Derived Linux Nodes

Some distributions of Linux derived from RHEL, including Oracle Linux, may have default firewall rules that block communication with Helm. We recommend disabling firewalld. For Kubernetes 1.19, firewalld must be turned off.



In RHEL 8.4, two extra services are included on the NetworkManager: `nm-cloud-setup.service` and `nm-cloud-setup.timer`. These services add a routing table that interferes with the CNI plugin's configuration. If these services are enabled, you must disable them using the command below, and then reboot the node to restore connectivity:

```
   systemctl disable nm-cloud-setup.service nm-cloud-
setup.timer
   reboot
```

#### SUSE Linux Nodes

SUSE Linux may have a firewall that blocks all ports by default. In that situation, follow these steps to open the ports needed for adding a host to a custom cluster.

#### Flatcar Container Linux Nodes

When Launching Kubernetes with Rancher using Flatcar Container Linux nodes, it is required to use the following configuration in the Cluster Config File.

**Canal**

```
rancher_kubernetes_engine_config:
  network:
    plugin: canal
    options:
      canal_flex_volume_plugin_dir: /opt/kubernetes/kubelet-
plugins/volume/exec/nodeagent~uds
      flannel_backend_type: vxlan

  services:
    kube-controller:
      extra_args:
        flex-volume-plugin-dir: /opt/kubernetes/kubelet-
plugins/volume/exec/
```

**Calico**

```
rancher_kubernetes_engine_config:
  network:
    plugin: calico
    options:
      calico_flex_volume_plugin_dir: /opt/kubernetes/kubelet-
plugins/volume/exec/nodeagent~uds
      flannel_backend_type: vxlan

  services:
    kube-controller:
      extra_args:
        flex-volume-plugin-dir: /opt/kubernetes/kubelet-
plugins/volume/exec/
```

It is also required to enable the Docker service, you can enable the Docker service using the following command:

```
systemctl enable docker.service
```

The Docker service is enabled automatically when using Node Drivers.

#### Windows Nodes

Use Windows server nodes for worker nodes only. For more information, see Configuring Custom Clusters for Windows.

### Hardware Requirements

The hardware requirements for nodes with the `worker` role mostly depend on your workloads. The minimum to run the Kubernetes node components is 1 CPU (core) and 1GB of memory.

Regarding CPU and memory, it is recommended that the different planes of Kubernetes clusters (etcd, controlplane, and workers) should be hosted on different nodes so that they can scale separately from each other.

For hardware recommendations for large Kubernetes clusters, refer to the official Kubernetes documentation on building large clusters. [https://kubernetes.io/docs/setup/best-practices/cluster-large/]

For hardware recommendations for etcd clusters in production, refer to the official etcd documentation. [https://etcd.io/docs/v3.5/op-guide/hardware/]

### Networking Requirements

For a production cluster, we recommend that you restrict traffic by opening only the ports defined in the port requirements below.

The ports required to be open are different depending on how the user cluster is launched. Each of the sections below list the ports that need to be opened for different cluster creation options.

For a breakdown of the port requirements for etcd nodes, controlplane nodes, and worker nodes in a Kubernetes cluster, refer to the port requirements for the Rancher Kubernetes Engine. [https://rancher.com/docs/rke/latest/en/os/#ports]

Details on which ports are used in each situation are found under Downstream Cluster Port Requirements.

#### IPv6 Address Requirements

Rancher supports clusters configured with IPv4-only, IPv6-only, or dual-stack networking.

You must provision each node with at least one valid IPv4 address, one IPv6 address, or both, according to the cluster networking configuration.

For IPv6-only environments, ensure you correctly configure the operating system and that the `/etc/hosts` file includes a valid localhost entry, for example:

```
::1       localhost
```



You should never register a node with the same hostname or IP address as an existing node. Doing so causes RKE to prevent the node from joining, and provisioning to hang. This can occur for both node driver and custom clusters. If a node must reuse a hostname or IP of an existing node, you must set the `hostname_override` RKE option [https://rke.docs.rancher.com/config-options/nodes#overriding-the-hostname] before registering the node, so that it can join correctly.

### Optional: Security Considerations

If you want to provision a Kubernetes cluster that is compliant with the CIS (Center for Internet Security) Kubernetes Benchmark, we recommend to following our hardening guide to configure your nodes before installing Kubernetes.

For more information on the hardening guide and details on which version of the guide corresponds to your Rancher and Kubernetes versions, refer to the security section.

## Checklist for Production-Ready Clusters

In this section, we recommend best practices for creating the production-ready Kubernetes clusters that will run your apps and services.

For a list of requirements for your cluster, including the requirements for OS/Docker, hardware, and networking, refer to the section on node requirements.

This is a shortlist of best practices that we strongly recommend for all production clusters.

For a full list of all the best practices that we recommend, refer to the best practices section.

### Node Requirements

- Make sure your nodes fulfill all of the node requirements, including the port requirements.

### Back up etcd

- Enable etcd snapshots. Verify that snapshots are being created, and run a disaster recovery scenario to verify the snapshots are valid. etcd is the location where the state of your cluster is stored, and losing etcd data means losing your cluster. Make sure you configure recurring snapshots of etcd for your cluster(s), and make sure the snapshots are stored externally (off the node) as well.

### Cluster Architecture

- Nodes should have one of the following role configurations:
  - `etcd`
  - `controlplane`
  - `etcd` and `controlplane`
  - `worker` (the `worker` role should not be used or added on nodes with the `etcd` or `controlplane` role)
- Have at least three nodes with the role `etcd` to survive losing one node. Increase this count for higher node fault toleration, and spread them across (availability) zones to provide even better fault tolerance.

- Assign two or more nodes the `controlplane` role for master component high availability.
- Assign two or more nodes the `worker` role for workload rescheduling upon node failure.

For more information on what each role is used for, refer to the section on roles for nodes in Kubernetes.

For more information about the number of nodes for each Kubernetes role, refer to the section on recommended architecture.

### Logging and Monitoring

- Configure alerts/notifiers for Kubernetes components (System Service).
- Configure logging for cluster analysis and post-mortems.

### Reliability

- Perform load tests on your cluster to verify that its hardware can support your workloads.

### Networking

- Minimize network latency. Rancher recommends minimizing latency between the etcd nodes. The default setting for `heartbeat-interval` is `500`, and the default setting for `election-timeout` is `5000`. These settings for etcd tuning [https://etcd.io/docs/v3.5/tuning/] allow etcd to run in most networks (except really high latency networks).
- Cluster nodes should be located within a single region. Most cloud providers provide multiple availability zones within a region, which can be used to create higher availability for your cluster. Using multiple availability zones is fine for nodes with any role. If you are using Kubernetes Cloud Provider resources, consult the documentation for any restrictions (i.e. zone storage restrictions).

### Recommended Cluster Architecture

There are three roles that can be assigned to nodes: `etcd`, `controlplane` and `worker`.

#### Separating Worker Nodes from Nodes with Other Roles

When designing your cluster(s), you have two options:

- Use dedicated nodes for each role. This ensures resource availability for the components needed for the specified role. It also strictly isolates network traffic between each of the roles according to the port requirements.
- Assign the `etcd` and `controlplane` roles to the same nodes. These nodes must meet the hardware requirements for both roles.

In either case, the `worker` role should not be used or added to nodes with the `etcd` or `controlplane` role.

Therefore, each node should have one of the following role configurations:

- `etcd`
- `controlplane`
- Both `etcd` and `controlplane`
- `worker`

#### Recommended Number of Nodes with Each Role

The cluster should have:

- At least three nodes with the role `etcd` to survive losing one node. Increase this count for higher node fault toleration, and spread them across (availability) zones to provide even better fault tolerance.
- At least two nodes with the role `controlplane` for master component high availability.
- At least two nodes with the role `worker` for workload rescheduling upon node failure.

For more information on what each role is used for, refer to the section on roles for nodes in Kubernetes.

##### Number of Controlplane Nodes

Adding more than one node with the `controlplane` role makes every master component highly available.

##### Number of etcd Nodes

The number of nodes that you can lose at once while maintaining cluster availability is determined by the number of nodes assigned the `etcd` role. For a cluster with n members, the minimum is (n/2)+1. Therefore, we recommend creating an `etcd` node in 3 different availability zones within a region to survive the loss of one availability zone. If you use only two zones, you can only survive the loss of the zone where you don't lose the majority of nodes.

| Nodes with `etcd` role | Majority | Failure Tolerance |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 2 | 0 |
| 3 | 2 | **1** |
| 4 | 3 | 1 |
| 5 | 3 | **2** |
| 6 | 4 | 2 |
| 7 | 4 | **3** |
| 8 | 5 | 3 |
| 9 | 5 | **4** |

References:

- Official etcd documentation on optimal etcd cluster size [https://etcd.io/docs/v3.5/faq/#what-is-failure-tolerance]
- Official Kubernetes documentation on operating etcd clusters for Kubernetes [https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/]

##### Number of Worker Nodes

Adding more than one node with the `worker` role will make sure your workloads can be rescheduled if a node fails.

##### Why Production Requirements are Different for the Rancher Cluster and the Clusters Running Your Applications

You may have noticed that our Kubernetes Install instructions do not meet our definition of a production-ready cluster, as there are no dedicated nodes for the `worker` role. However, for your Rancher installation,

this three node cluster is valid, because:

- It allows one `etcd` node failure.
- It maintains multiple instances of the master components by having multiple `controlplane` nodes.
- No other workloads than Rancher itself should be created on this cluster.

**References**

- Kubernetes: Master Components [https://kubernetes.io/docs/concepts/overview/components/#master-components]

## Roles for Nodes in Kubernetes

This section describes the roles for etcd nodes, controlplane nodes, and worker nodes in Kubernetes, and how the roles work together in a cluster.

This diagram is applicable to Kubernetes clusters launched with Rancher using RKE..

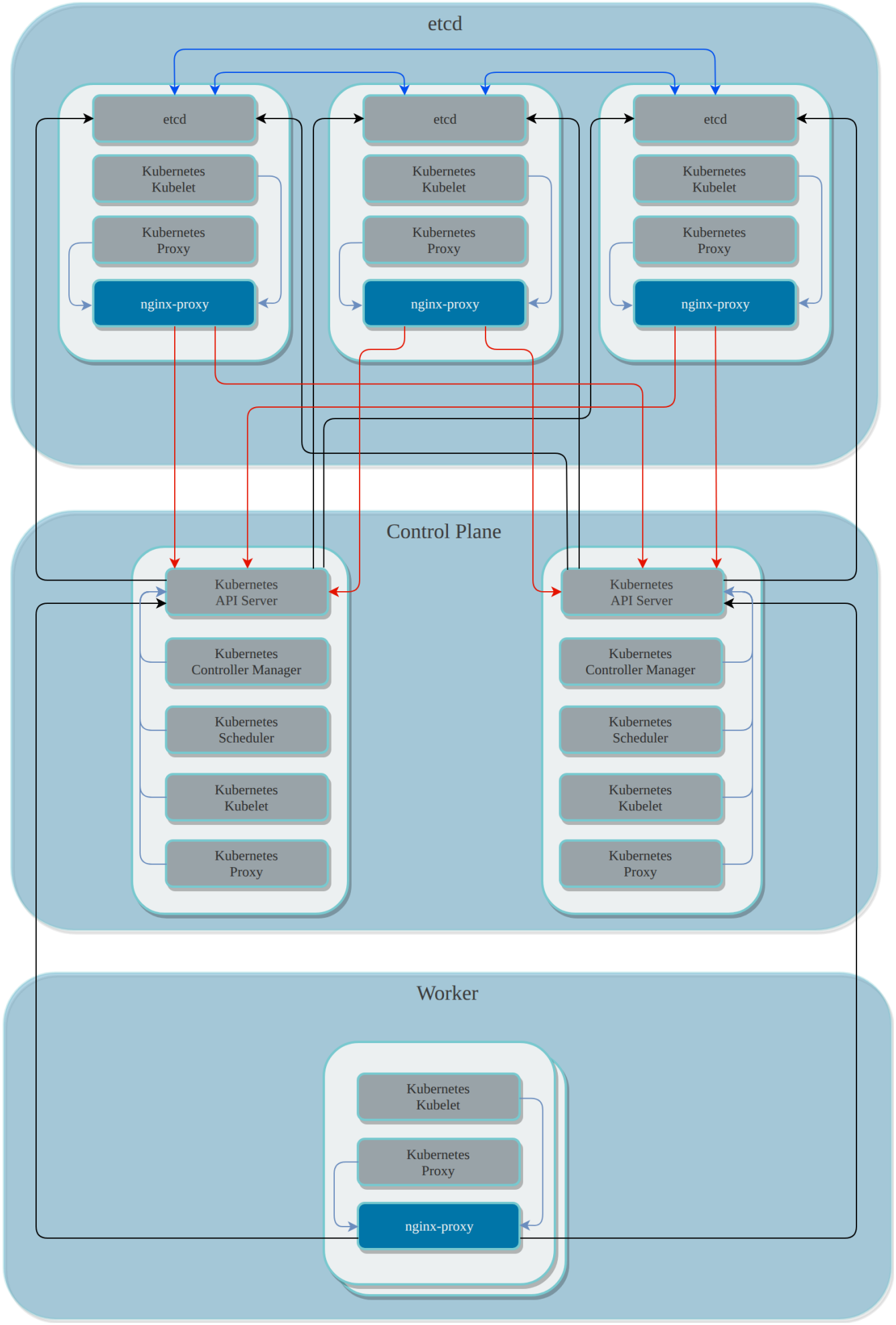



*Figure 10. Lines show the traffic flow between components. Colors are used purely for visual aid*

#### etcd

Nodes with the `etcd` role run etcd, which is a consistent and highly available key value store used as Kubernetes' backing store for all cluster data. etcd replicates the data to each node.



Nodes with the `etcd` role are shown as `Unschedulable` in the UI, meaning no pods will be scheduled to these nodes by default.

#### controlplane

Nodes with the `controlplane` role run the Kubernetes master components (excluding `etcd`, as it's a separate role). See Kubernetes: Master Components [https://kubernetes.io/docs/concepts/overview/components/#master-components] for a detailed list of components.



Nodes with the `controlplane` role are shown as `Unschedulable` in the UI, meaning no pods will be scheduled to these nodes by default.

##### kube-apiserver

The Kubernetes API server (`kube-apiserver`) scales horizontally. Each node with the role `controlplane` will be added to the NGINX proxy on the nodes with components that need to access the Kubernetes API server. This means that if a node becomes unreachable, the local NGINX proxy on the node will forward the request to another Kubernetes API server in the list.

##### kube-controller-manager

The Kubernetes controller manager uses leader election using an endpoint in Kubernetes. One instance of the `kube-controller-manager` will create an entry in the Kubernetes endpoints and updates that entry in a configured interval. Other instances will see an active leader and wait for that entry to expire (for example, when a node is unresponsive).

##### kube-scheduler

The Kubernetes scheduler uses leader election using an endpoint in Kubernetes. One instance of the `kube-scheduler` will create an entry in the Kubernetes endpoints and updates that entry in a configured interval. Other instances will see an active leader and wait for that entry to expire (for example, when a node is unresponsive).

#### worker

Nodes with the `worker` role run the Kubernetes node components. See Kubernetes: Node Components [https://kubernetes.io/docs/concepts/overview/components/#node-components] for a detailed list of components.

#### References

- Kubernetes: Node Components [https://kubernetes.io/docs/concepts/overview/components/#node-components]

## Tips for Setting Up Containers

Running well-built containers can greatly impact the overall performance and security of your environment.

Below are a few tips for setting up your containers.

For a more detailed discussion of security for containers, you can also refer to Rancher's Guide to Container Security. [https://rancher.com/complete-guide-container-security]

#### Use a Common Container OS

When possible, you should try to standardize on a common container base OS.

Smaller distributions such as Alpine and BusyBox reduce container image size and generally have a smaller attack/vulnerability surface.

Popular distributions such as Ubuntu, Fedora, and CentOS are more field-tested and offer more functionality.

#### Start with a FROM scratch container

If your microservice is a standalone static binary, you should use a FROM scratch container.

The FROM scratch container is an official Docker image [https://hub.docker.com/_/scratch] that is empty so that you can use it to design minimal images.

This will have the smallest attack surface and smallest image size.

#### Run Container Processes as Unprivileged

When possible, use a non-privileged user when running processes within your container. While container runtimes provide isolation, vulnerabilities and attacks are still possible. Inadvertent or accidental host mounts can also be impacted if the container is running as root. For details on configuring a security context for a pod or container, refer to the Kubernetes docs [https://kubernetes.io/docs/tasks/configure-pod-container/security-context/].

#### Define Resource Limits

Apply CPU and memory limits to your pods. This can help manage the resources on your worker nodes and avoid a malfunctioning microservice from impacting other microservices.

In standard Kubernetes, you can set resource limits on the namespace level. In Rancher, you can set resource limits on the project level and they will propagate to all the namespaces within the project. For details, refer to the Rancher docs.

When setting resource quotas, if you set anything related to CPU or Memory (i.e. limits or reservations) on a project or namespace, all containers will require a respective CPU or Memory field set during creation. To avoid setting these limits on each and every container during workload creation, a default container resource limit can be specified on the namespace.

The Kubernetes docs have more information on how resource limits can be set at the container level [https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#resource-requests-and-limits-of-pod-and-container] and the namespace level.

#### Define Resource Requirements

You should apply CPU and memory requirements to your pods. This is crucial for informing the scheduler which type of compute node your pod needs to be placed on, and ensuring it does not over-provision that node. In Kubernetes, you can set a resource requirement by defining `resources.requests` in the

resource requests field in a pod's container spec. For details, refer to the Kubernetes docs [https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#resource-requests-and-limits-of-pod-and-container].



If you set a resource limit for the namespace that the pod is deployed in, and the container doesn't have a specific resource request, the pod will not be allowed to start. To avoid setting these fields on each and every container during workload creation, a default container resource limit can be specified on the namespace.

It is recommended to define resource requirements on the container level because otherwise, the scheduler makes assumptions that will likely not be helpful to your application when the cluster experiences load.

#### Liveness and Readiness Probes

Set up liveness and readiness probes for your container. Unless your container completely crashes, Kubernetes will not know it's unhealthy unless you create an endpoint or mechanism that can report container status. Alternatively, make sure your container halts and crashes if unhealthy.

The Kubernetes docs show how to configure liveness and readiness probes for containers. [https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/]

## Setting up Clusters from Hosted Kubernetes Providers

In this scenario, Rancher does not provision Kubernetes because it is installed by providers such as Google Kubernetes Engine (GKE), Amazon Elastic Container Service for Kubernetes, or Azure Kubernetes Service.

If you use a Kubernetes provider such as Google GKE, Rancher integrates with its cloud APIs, allowing you to create and manage role-based access control for the hosted cluster from the Rancher UI.

In this use case, Rancher sends a request to a hosted provider using the provider's API. The provider then provisions and hosts the cluster for you. When the cluster finishes building, you can manage it from the Rancher UI along with clusters you've provisioned that are hosted on-prem or in an infrastructure provider.

Rancher supports the following Kubernetes providers:

- Google GKE (Google Kubernetes Engine) [https://cloud.google.com/kubernetes-engine/]
- Amazon EKS (Amazon Elastic Container Service for Kubernetes) [https://aws.amazon.com/eks/]
- Microsoft AKS (Azure Kubernetes Service) [https://azure.microsoft.com/en-us/services/kubernetes-service/]

### Hosted Kubernetes Provider Authentication

When using Rancher to create a cluster hosted by a provider, you are prompted for authentication information. This information is required to access the provider's API. For more information on how to obtain this information, see the following procedures:

- Creating a GKE Cluster
- Creating an EKS Cluster
- Creating an AKS Cluster

## Creating an AKS Cluster

You can use Rancher to create a cluster hosted in Microsoft Azure Kubernetes Service (AKS).

### Prerequisites in Microsoft Azure



Deploying to AKS will incur charges.

To interact with Azure APIs, an AKS cluster requires an Microsoft Entra ID service principal. The service principal is needed to dynamically create and manage other Azure resources, and it provides credentials for your cluster to communicate with AKS. For more information about the service principal, refer to the AKS documentation [https://docs.microsoft.com/en-us/azure/aks/kubernetes-service-principal].

Before creating the service principal, you need to obtain the following information from the Microsoft Azure Portal [https://portal.azure.com]:

- Subscription ID
- Client ID (also known as app ID)
- Client secret

The below sections describe how to set up these prerequisites using either the Azure command line tool or the Azure portal.

#### Setting Up the Service Principal with the Azure Command Line Tool

You must assign roles to the service principal so that it has communication privileges with the AKS API. It also needs access to create and list virtual networks.

In the following example, the command creates the service principal and gives it the Contributor role. The Contributor role can manage anything on AKS but cannot give access to others. Note that you must provide `scopes` a full path to at least one Azure resource:

```
az ad sp create-for-rbac --role Contributor --scopes
/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>
```

The result should show information about the new service principal:

```
{
  "appId": "xxxx--xxx",
  "displayName": "<service-principal-name>",
  "name": "http://<service-principal-name>",
  "password": "<secret>",
  "tenant": "<tenant-name>"
}
```

The following creates a Resource Group [https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-cli] to contain your Azure resources:

```
az group create --location <azure-location-name> --resource-group
<resource-group-name>
```

#### Setting Up the Service Principal from the Azure Portal

Follow these instructions to set up a service principal and give it role-based access from the Azure Portal.

1. Go to the Microsoft Azure Portal home page [https://portal.azure.com].
2. Click **Microsoft Entra ID**.
3. Click **App registrations**.
4. Click **New registration**.
5. Enter a name for your service principal (application).
6. (Optional) Choose which accounts can use the service principal.
7. Click **Register**. You should now see the name of your service principal under **Microsoft Entra ID › App registrations**.
8. Click the name of your service principal. Take note of the Application ID (also called app ID or client ID) so that you can use it when provisioning your AKS cluster. Then click **Certificates & secrets**.
9. Click **New client secret**.
10. Enter a short description, pick an expiration time, and click **Add**. Take note of the client secret so that you can use it when provisioning the AKS cluster.

**Result:** You have created a service principal and you should be able to see it in your Azure Portal listed in the **Microsoft Entra ID** section under **App registrations > All Applications**. You still need to give the service principal access to AKS.

Follow these instructions to give role-based access to your service principal.

1. Click **All Services** in the left navigation bar. Then click **Subscriptions**.
2. Click the name of the subscription that you want to associate with your Kubernetes cluster. Take note of the subscription ID so that you can use it when provisioning your AKS cluster.
3. Click **Access Control (IAM)**.
4. On the **Add role assignment** tab, click **Add**.
5. On the **Role** tab, select a role that will have access to AKS. For example, you can use the **Contributor** role, which has permission to manage everything except for giving access to other users.
6. On the **Members tab**, select **User, group, or service principal** to assign the selected role to one or more Microsoft Entra users, groups, or service principals (applications).
7. In the **Select** field, select the name of your service principal and click **Save**.

**Result:** Your service principal now has access to AKS.

#### Create the AKS Cloud Credentials

1. In the Rancher UI, click ☰ **> Cluster Management**.
2. Click **Cloud Credentials**.

3. Click **Create**.
4. Click **Azure**.
5. Fill out the form. For help with filling out the form, see the configuration reference.
6. Click **Create**.

#### Create the AKS Cluster

Use Rancher to set up and configure your Kubernetes cluster.

1. Click ☰ > **Cluster Management**.
2. In the **Clusters** section, click **Create**.
3. Click **Azure AKS**.
4. Fill out the form. For help with filling out the form, see the configuration reference.
5. Click **Create**.

**Result:** Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

#### Configure Role-based Access Control

When provisioning an AKS cluster in the Rancher UI, RBAC is not configurable because it is required to be enabled.

RBAC is required for AKS clusters that are registered or imported into Rancher.

##### Setting Up the Role Assignment to Service Principal with the Azure Command Line Tool

Assign the Rancher AKSv2 role to the service principal with the Azure Command Line Tool:

```
az role assignment create \
--assignee <client-id> \
--scope "/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>" \
--role "Rancher AKSv2"
```

#### AKS Cluster Configuration Reference

For more information about how to configure AKS clusters from the Rancher UI, see the configuration reference.

#### Private Clusters

Typically, AKS worker nodes do not get public IPs, regardless of whether the cluster is private. In a private cluster, the control plane does not have a public endpoint.

Rancher can connect to a private AKS cluster in one of two ways.

The first way to ensure that Rancher is running on the same NAT [https://docs.microsoft.com/en-us/azure/virtual-

network/nat-overview] as the AKS nodes.

The second way is to run a command to register the cluster with Rancher. Once the cluster is provisioned, you can run the displayed command anywhere you can connect to the cluster's Kubernetes API. This command is displayed in a pop-up when you provision an AKS cluster with a private API endpoint enabled.



Please be aware that when registering an existing AKS cluster, the cluster might take some time, possibly hours, to appear in the `Cluster To register` dropdown list. This outcome will be based on region.

For more information about connecting to an AKS private cluster, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/private-clusters#options-for-connecting-to-the-private-cluster]

**Setting Up the Minimum Permission Role with the Azure Command Line Tool**

1. Create the Minimum Rancher AKSv2 Permission Role by running this command:

```
cat >> rancher-azure.json << EOF

{
    "Name": "Rancher AKSv2",
    "IsCustom": true,
    "Description": "Everything needed by Rancher AKSv2 operator",
    "Actions": [
        "Microsoft.Compute/disks/delete",
        "Microsoft.Compute/disks/read",
        "Microsoft.Compute/disks/write",
        "Microsoft.Compute/diskEncryptionSets/read",
        "Microsoft.Compute/locations/DiskOperations/read",
        "Microsoft.Compute/locations/vmSizes/read",
        "Microsoft.Compute/locations/operations/read",
        "Microsoft.Compute/proximityPlacementGroups/write",
        "Microsoft.Compute/snapshots/delete",
        "Microsoft.Compute/snapshots/read",
        "Microsoft.Compute/snapshots/write",

"Microsoft.Compute/virtualMachineScaleSets/manualUpgrade/action",
        "Microsoft.Compute/virtualMachineScaleSets/delete",
        "Microsoft.Compute/virtualMachineScaleSets/read",

"Microsoft.Compute/virtualMachineScaleSets/virtualMachines/networkInterf
aces/read",

"Microsoft.Compute/virtualMachineScaleSets/virtualMachines/networkInterf
aces/ipconfigurations/publicipaddresses/read",

"Microsoft.Compute/virtualMachineScaleSets/virtualmachines/instanceView/
```

```
read",

"Microsoft.Compute/virtualMachineScaleSets/virtualMachines/read",

"Microsoft.Compute/virtualMachineScaleSets/virtualMachines/write",
          "Microsoft.Compute/virtualMachineScaleSets/write",
          "Microsoft.Compute/virtualMachines/read",
          "Microsoft.Compute/virtualMachines/write",
          "Microsoft.ContainerService/managedClusters/read",
          "Microsoft.ContainerService/managedClusters/write",
          "Microsoft.ContainerService/managedClusters/delete",

"Microsoft.ContainerService/managedClusters/accessProfiles/listCredentia
l/action",
          "Microsoft.ContainerService/managedClusters/agentPools/read",
          "Microsoft.ContainerService/managedClusters/agentPools/write",

"Microsoft.ContainerService/managedClusters/agentPools/delete",

"Microsoft.ManagedIdentity/userAssignedIdentities/assign/action",
          "Microsoft.Network/applicationGateways/read",
          "Microsoft.Network/applicationGateways/write",
          "Microsoft.Network/loadBalancers/write",

"Microsoft.Network/loadBalancers/backendAddressPools/join/action",
          "Microsoft.Network/loadBalancers/delete",
          "Microsoft.Network/loadBalancers/read",
          "Microsoft.Network/networkInterfaces/join/action",
          "Microsoft.Network/networkInterfaces/read",
          "Microsoft.Network/networkInterfaces/write",
          "Microsoft.Network/networkSecurityGroups/read",
          "Microsoft.Network/networkSecurityGroups/write",
          "Microsoft.Network/publicIPAddresses/delete",
          "Microsoft.Network/publicIPAddresses/join/action",
          "Microsoft.Network/publicIPAddresses/read",
          "Microsoft.Network/publicIPAddresses/write",
          "Microsoft.Network/publicIPPrefixes/join/action",
          "Microsoft.Network/privatednszones/*",
          "Microsoft.Network/routeTables/read",
          "Microsoft.Network/routeTables/routes/delete",
          "Microsoft.Network/routeTables/routes/read",
          "Microsoft.Network/routeTables/routes/write",
          "Microsoft.Network/routeTables/write",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/join/action",
```

```
            "Microsoft.Network/virtualNetworks/subnets/read",
            "Microsoft.Network/virtualNetworks/joinLoadBalancer/action",
            "Microsoft.OperationalInsights/workspaces/sharedkeys/read",
            "Microsoft.OperationalInsights/workspaces/read",
            "Microsoft.OperationsManagement/solutions/write",
            "Microsoft.OperationsManagement/solutions/read",
            "Microsoft.Resources/subscriptions/resourcegroups/read",
            "Microsoft.Resources/subscriptions/resourcegroups/write",
            "Microsoft.Storage/operations/read",
            "Microsoft.Storage/storageAccounts/listKeys/action",
            "Microsoft.Storage/storageAccounts/delete",
            "Microsoft.Storage/storageAccounts/read",
            "Microsoft.Storage/storageAccounts/write"
        ],
        "NotActions": [],
        "DataActions": [],
        "NotDataActions": [],
        "AssignableScopes": [
            "/subscriptions/SUBSCRIPTION_ID"
        ]
    }
    EOF
```

1. Apply the Rancher AKSv2 Role:

```
az role definition create --role-definition rancher-azure.json
```

1. Verify if the Rancher AKSv2 Role was created:

```
az role definition list | grep "Rancher AKSv2"
```

#### Syncing

The AKS provisioner can synchronize the state of an AKS cluster between Rancher and the provider. For an in-depth technical explanation of how this works, see Syncing.

For information on configuring the refresh interval, see this section.

#### Programmatically Creating AKS Clusters

The most common way to programmatically deploy AKS clusters through Rancher is by using the Rancher2 Terraform provider. The documentation for creating clusters with Terraform is here [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster].

### AKS Cluster Configuration Reference

#### Role-based Access Control

When provisioning an AKS cluster in the Rancher UI, RBAC cannot be disabled. If role-based access control is disabled for the cluster in AKS, the cluster cannot be registered or imported into Rancher. In practice, this means that local accounts must be enabled in order to register an AKS cluster.

Rancher can configure member roles for AKS clusters in the same way as any other cluster. For more information, see the section on role-based access control.

#### Cloud Credentials



The configuration information in this section assumes you have already set up a service principal for Rancher. For step-by-step instructions for how to set up the service principal, see this section.

#### Subscription ID

To get the subscription ID, click **All Services** in the left navigation bar. Then click **Subscriptions**. Go to the name of the subscription that you want to associate with your Kubernetes cluster and copy the **Subscription ID**.

#### Client ID

To get the Application (Client) ID:

1. Sign in to the Azure Portal [https://portal.azure.com].
2. Search for and select **Microsoft Entra ID** (formerly *Azure Active Directory*).
3. In the left navigation menu, select **App registrations**.
4. Click the name of your application.
5. On the **Overview** page, locate and copy the value listed under **Application (client) ID**.

#### Client Secret

You can't retrieve the client secret value after it is created, so if you don't already have a client secret value, you will need to create a new client secret.

To get a new client secret:

1. Sign in to the Azure Portal [https://portal.azure.com].
2. Search for and select **Microsoft Entra ID** (formerly *Azure Active Directory*).
3. In the left navigation menu, select **App registrations**.
4. Click the name of your application.
5. Under the **Manage** section in the left menu, select **Certificates & secrets**.
6. Select the **Client secrets** tab, then click **+ New client secret**.
7. Provide a description, choose an expiration period, and click **Add**.
8. Copy the **Value** of the new client secret.



Copy the secret **Value** immediately after creation. It will only be displayed once and will be permanently masked (••••••••••••••••) after you navigate away or refresh the page. Do not copy the **Secret ID**.

#### Environment

Microsoft provides multiple clouds [https://docs.microsoft.com/en-us/cli/azure/cloud?view=azure-cli-latest] for compliance with regional laws, which are available for your use:

- AzurePublicCloud
- AzureGermanCloud
- AzureChinaCloud
- AzureUSGovernmentCloud

#### Account Access

In this section you will need to select an existing Azure cloud credential or create a new one.

For help configuring your Azure cloud credential, see this section.

#### Cluster Location

Configure the cluster and node location. For more information on availability zones for AKS, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/availability-zones]

The high availability locations include multiple availability zones.

#### Cluster Options

#### Kubernetes Version

The available Kubernetes versions are dynamically fetched from the Azure API.

#### Cluster Resource Group

A resource group is a container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Generally, add resources that share the same lifecycle to the same resource group so you can easily deploy, update, and delete them as a group.

Use an existing resource group or enter a resource group name and one will be created for you.

Using a resource group containing an existing AKS cluster will create a new resource group. Azure AKS only allows one AKS cluster per resource group.

For information on managing resource groups, see the Azure documentation. [https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal]

#### Linux Admin Username

The username used to create an SSH connection to the Linux nodes.

The default username for AKS nodes is `azureuser`.

#### SSH Public Key

The key used to create an SSH connection to the Linux nodes.

#### Tags

Cluster tags can be useful if your organization uses tags as a way to organize resources across multiple Azure services. These tags don't apply to resources within the cluster.

#### Networking Options

#### LoadBalancer SKU

Azure load balancers support both standard and basic SKUs (stock keeping units).

For a comparison of standard and basic load balancers, see the official Azure documentation. [https://docs.microsoft.com/en-us/azure/load-balancer/skus#skus] Microsoft recommends the Standard load balancer.

The Standard load balancer is required if you have selected one or more availability zones, or if you have more than one node pool.

#### Network Policy

All pods in an AKS cluster can send and receive traffic without limitations, by default. To improve security, you can define rules that control the flow of traffic. The Network Policy feature in Kubernetes lets you define rules for ingress and egress traffic between pods in a cluster.

Azure provides two ways to implement network policy. You choose a network policy option when you create an AKS cluster. The policy option can't be changed after the cluster is created:

- Azure's own implementation, called Azure Network Policies. The Azure network policy requires the Azure CNI.
- Calico Network Policies, an open-source network and network security solution founded by Tigera [https://www.tigera.io/].

You can also choose to have no network policy.

For more information about the differences between Azure and Calico network policies and their capabilities, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/use-network-policies#differences-between-azure-and-calico-policies-and-their-capabilities]

#### DNS Prefix

Enter a unique DNS prefix for your cluster's Kubernetes API server FQDN.

#### Network Plugin

There are two network plugins: kubenet and Azure CNI.

The kubenet [https://kubernetes.io/docs/concepts/cluster-administration/network-plugins/#kubenet] Kubernetes plugin is the default configuration for AKS cluster creation. When kubenet is used, each node in the cluster

receives a routable IP address. The pods use NAT to communicate with other resources outside the AKS cluster. This approach reduces the number of IP addresses you need to reserve in your network space for pods to use.

With the Azure CNI (advanced) networking plugin, pods get full virtual network connectivity and can be directly reached via their private IP address from connected networks. This plugin requires more IP address space.

For more information on the differences between kubenet and Azure CNI, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/concepts-network#compare-network-models]

#### HTTP Application Routing

When enabled, the HTTP application routing add-on makes it easier to access applications deployed to the AKS cluster. It deploys two components: a Kubernetes Ingress controller [https://kubernetes.io/docs/concepts/services-networking/ingress/] and an External-DNS [https://github.com/kubernetes-incubator/external-dns] controller.

For more information, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/http-application-routing]

#### Set Authorized IP Ranges

You can secure access to the Kubernetes API server using authorized IP address ranges. [https://docs.microsoft.com/en-us/azure/aks/api-server-authorized-ip-ranges#overview-of-api-server-authorized-ip-ranges]

The Kubernetes API server exposes the Kubernetes API. This component provides the interaction for management tools, such as kubectl. AKS provides a single-tenant cluster control plane with a dedicated API server. By default, the API server is assigned a public IP address, and you should control access to it using Kubernetes-based or Azure-based RBAC.

To secure access to the otherwise publicly accessible AKS control plane and API server, you can enable and use authorized IP ranges. These authorized IP ranges only allow defined IP address ranges to communicate with the API server.

However, even if you use authorized IP address ranges, you should still use Kubernetes RBAC or Azure RBAC to authorize users and the actions they request.

#### Container Monitoring

Container monitoring gives you performance visibility by collecting memory and processor metrics from controllers, nodes, and containers that are available in Kubernetes through the Metrics API. Container logs are also collected. After you enable monitoring, metrics and logs are automatically collected for you through a containerized version of the Log Analytics agent for Linux. Metrics are written to the metrics store and log data is written to the logs store associated with your Log Analytics [https://docs.microsoft.com/en-us/azure/azure-monitor/logs/log-query-overview] workspace.

#### Log Analytics Workspace Resource Group

The resource group [https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview#resource-groups] containing the Log Analytics Workspace. You must create at least one workspace to use Azure Monitor Logs.

#### Log Analytics Workspace Name

Data collected by Azure Monitor Logs is stored in one or more Log Analytics workspaces. [https://docs.microsoft.com/en-us/azure/azure-monitor/logs/design-logs-deployment] The workspace defines the geographic location of the data, access rights defining which users can access data, and configuration settings such as the pricing tier and data retention.

You must create at least one workspace to use Azure Monitor Logs. A single workspace may be suffxicient for all of your monitoring data, or may choose to create multiple workspaces depending on your requirements. For example, you might have one workspace for your production data and another for testing.

For more information about Azure Monitor Logs, see the Azure documentation. [https://docs.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs]

#### Support Private Kubernetes Service

Typically, AKS worker nodes do not get public IPs, regardless of whether the cluster is private. In a private cluster, the control plane does not have a public endpoint.

Rancher can connect to a private AKS cluster in one of two ways.

The first way to ensure that Rancher is running on the same NAT [https://docs.microsoft.com/en-us/azure/virtual-network/nat-overview] as the AKS nodes.

The second way is to run a command to register the cluster with Rancher. Once the cluster is provisioned, you can run the displayed command anywhere you can connect to the cluster's Kubernetes API. This command is displayed in a pop-up when you provision an AKS cluster with a private API endpoint enabled.



Please be aware that when registering an existing AKS cluster, the cluster might take some time, possibly hours, to appear in the `Cluster To register` dropdown list. This outcome will be based on region.

For more information about connecting to an AKS private cluster, see the AKS documentation. [https://docs.microsoft.com/en-us/azure/aks/private-clusters#options-for-connecting-to-the-private-cluster]

#### Node Pools

#### Mode

The Azure interface allows users to specify whether a Primary Node Pool relies on either `system` (normally used for control planes) or `user` (what is most typically needed for Rancher).

For Primary Node Pools, you can specify Mode, OS, Count and Size.

System node pools always require running nodes, so they cannot be scaled below one node. At least one system node pool is required.

For subsequent node pools, the Rancher UI forces the default of User. User node pools allow you to scale to zero nodes. User node pools don't run any part of the Kubernetes controlplane.

AKS doesn't expose the nodes that run the Kubernetes controlplane components.

#### Availability Zones

Availability zones [https://docs.microsoft.com/en-us/azure/availability-zones/az-overview] are unique physical locations within a region. Each zone is made up of one or more data centers equipped with independent power, cooling, and networking.

Not all regions have support for availability zones. For a list of Azure regions with availability zones, see the Azure documentation. [https://docs.microsoft.com/en-us/azure/availability-zones/az-region#azure-regions-with-availability-zones]

#### VM Size

Choose a size for each VM in the node pool. For details about each VM size, see this page. [https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/]

#### OS Disk Type

The nodes in the node pool can have either managed or ephemeral disks.

Ephemeral OS disks [https://docs.microsoft.com/en-us/azure/virtual-machines/ephemeral-os-disks] are created on the local virtual machine storage and not saved to the remote Azure Storage. Ephemeral OS disks work well for stateless workloads, where applications are tolerant of individual VM failures, but are more affected by VM deployment time or reimaging the individual VM instances. With Ephemeral OS disk, you get lower read/write latency to the OS disk and faster VM reimage.

Azure managed disks [https://docs.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview] are block-level storage volumes that are managed by Azure and used with Azure Virtual Machines. Managed disks are designed for 99.999% availability. Managed disks achieve this by providing you with three replicas of your data, allowing for high durability.

#### OS Disk Size

The size in GB for the disk for each node.

#### Node Count

The number of nodes in the node pool. The maximum number of nodes may be limited by your Azure subscription. [https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits]

#### Max Pods Per Node

The maximum number of pods per node defaults to 110 with a maximum of 250.

#### Enable Auto Scaling

When auto scaling is enabled, you will need to enter a minimum and maximum node count.

When Auto Scaling is enabled, you can't manually scale the node pool. The scale is controlled by the AKS autoscaler.

## Creating an EKS Cluster

Amazon EKS provides a managed control plane for your Kubernetes cluster. Amazon EKS runs the

Kubernetes control plane instances across multiple Availability Zones to ensure high availability. Rancher provides an intuitive user interface for managing and deploying the Kubernetes clusters you run in Amazon EKS. With this guide, you will use Rancher to quickly and easily launch an Amazon EKS Kubernetes cluster in your AWS account. For more information on Amazon EKS, see this documentation [https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html].

### Prerequisites in Amazon Web Services



Deploying to Amazon AWS will incur charges. For more information, refer to the EKS pricing page [https://aws.amazon.com/eks/pricing/].

To set up a cluster on EKS, you will need to set up an Amazon VPC (Virtual Private Cloud). You will also need to make sure that the account you will be using to create the EKS cluster has the appropriate permissions. For details, refer to the official guide on Amazon EKS Prerequisites [https://docs.aws.amazon.com/eks/latest/userguide/getting-started-console.html#eks-prereqs].

#### Amazon VPC

An Amazon VPC is required to launch the EKS cluster. The VPC enables you to launch AWS resources into a virtual network that you've defined. You can set one up yourself and provide it during cluster creation in Rancher. If you do not provide one during creation, Rancher will create one. For more information, refer to the Tutorial: Creating a VPC with Public and Private Subnets for Your Amazon EKS Cluster [https://docs.aws.amazon.com/eks/latest/userguide/create-public-private-vpc.html].

#### IAM Policies

Rancher needs access to your AWS account in order to provision and administer your Kubernetes clusters in Amazon EKS. You'll need to create a user for Rancher in your AWS account and define what that user can access.

1. Create a user with programmatic access by following the steps here [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html].
2. Next, create an IAM policy that defines what this user has access to in your AWS account. It's important to only grant this user minimal access within your account. The minimum permissions required for an EKS cluster are listed here. Follow the steps here [https://docs.aws.amazon.com/eks/latest/userguide/EKS_IAM_user_policies.html] to create an IAM policy and attach it to your user.
3. Finally, follow the steps here [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey] to create an access key and secret key for this user.



It's important to regularly rotate your access and secret keys. See this documentation [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#rotating_access_keys_console] for more information.

For more detailed information on IAM policies for EKS, refer to the official documentation on Amazon EKS IAM Policies, Roles, and Permissions [https://docs.aws.amazon.com/eks/latest/userguide/IAM_policies.html].

### Create the EKS Cluster

Use Rancher to set up and configure your Kubernetes cluster.

1. Click ☰ > **Cluster Management**.

2. On the **Clusters** page, click **Create**.
3. Choose **Amazon EKS**.
4. Enter a **Cluster Name**.
5. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
6. Fill out the rest of the form. For help, refer to the configuration reference.
7. Click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

#### EKS Cluster Configuration Reference

For the full list of EKS cluster configuration options, see this page.

#### Architecture

The figure below illustrates the high-level architecture of Rancher 2.x. The figure depicts a Rancher Server installation that manages two Kubernetes clusters: one created by RKE and another created by EKS.

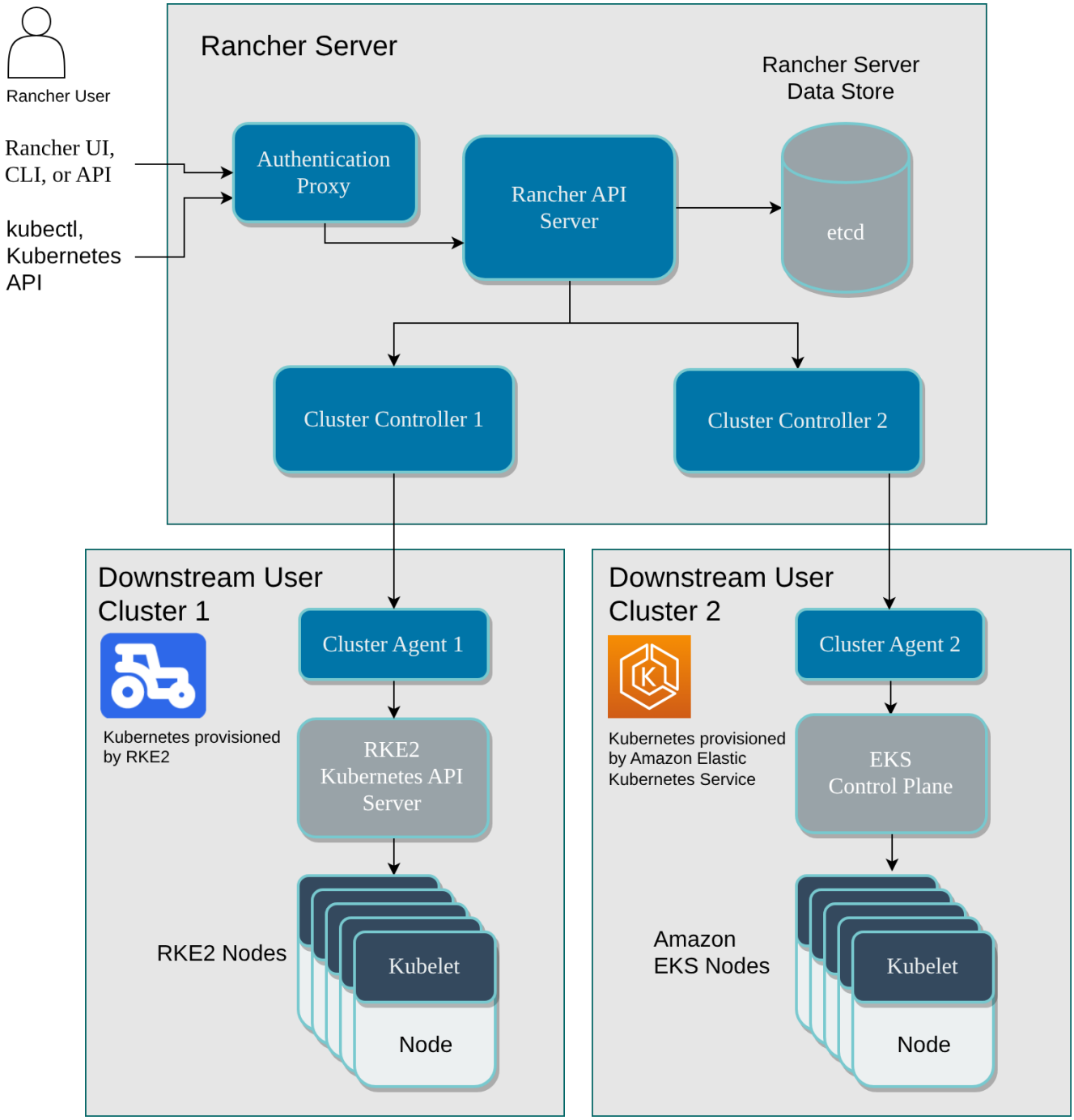



*Figure 11. Managing Kubernetes Clusters through Rancher's Authentication Proxy*

### AWS Service Events

To find information on any AWS Service events, please see this page [https://status.aws.amazon.com/].

### Security and Compliance

By default only the IAM user or role that created a cluster has access to it. Attempting to access the cluster with any other user or role without additional configuration will lead to an error. In Rancher, this means using a credential that maps to a user or role that was not used to create the cluster will cause an unauthorized error. For example, an EKSCtl cluster will not register in Rancher unless the credentials used to register the cluster match the role or user used by EKSCtl. Additional users and roles can be authorized to access a cluster by being added to the aws-auth configmap in the kube-system namespace. For a more in-depth explanation and detailed instructions, please see this documentation [https://aws.amazon.com/premiumsupport/knowledge-center/amazon-eks-cluster-access/].

For more information on security and compliance with your Amazon EKS Kubernetes cluster, please see this documentation [https://docs.aws.amazon.com/eks/latest/userguide/shared-responsibilty.html].

### Tutorial

This tutorial [https://aws.amazon.com/blogs/opensource/managing-eks-clusters-rancher/] on the AWS Open Source Blog will walk you through how to set up an EKS cluster with Rancher, deploy a publicly accessible app to test the cluster, and deploy a sample project to track real-time geospatial data using a combination of other open-source software such as Grafana and InfluxDB.

### Minimum EKS Permissions

These are the minimum set of permissions necessary to access the full functionality of Rancher's EKS driver. These permissions allow Rancher to create the Service Role and Virtual Private Cloud (VPC) resources on the users' behalf, if necessary.



In EKS v1.23 and above, you must use the out-of-tree drivers for EBS-backed volumes. You need specific permissions to enable this add-on.

| Resource | Description |
|---|---|
| EBS CSI Driver add-on | Provides permissions that allow Kubernetes to interact with EBS and configure the cluster to enable the add-on (required for EKS v1.23 and above). Rancher can install the add-on with the following EBS CSI Driver addon Permissions. |

Resource targeting uses * as the ARN of many of the resources created cannot be known before creating the EKS cluster in Rancher.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EC2Permissions",
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateKeyPair",
        "ec2:CreateLaunchTemplate",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:DeleteKeyPair",
        "ec2:DeleteLaunchTemplate",
        "ec2:DeleteLaunchTemplateVersions",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteTags",
        "ec2:DescribeAccountAttributes",
```

```
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeImages",
        "ec2:DescribeInternetGateways",
        "ec2:DescribeInstanceTypes",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeRegions",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeTags",
        "ec2:DescribeVpcs",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:RunInstances"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudFormationPermissions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:CreateStackSet",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackResources",
        "cloudformation:ListStacks"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IAMPermissions",
      "Effect": "Allow",
      "Action": [
        "iam:AddRoleToInstanceProfile",
        "iam:AttachRolePolicy",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:CreateServiceLinkedRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRole",
```

```
        "iam:DetachRolePolicy",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:ListAttachedRolePolicies",
        "iam:ListInstanceProfiles",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRoles",
        "iam:ListRoleTags",
        "iam:PassRole",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:TagRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "KMSPermissions",
      "Effect": "Allow",
      "Action": "kms:ListKeys",
      "Resource": "*"
    },
    {
      "Sid": "EKSPermissions",
      "Effect": "Allow",
      "Action": [
        "eks:CreateCluster",
        "eks:CreateFargateProfile",
        "eks:CreateNodegroup",
        "eks:DeleteCluster",
        "eks:DeleteFargateProfile",
        "eks:DeleteNodegroup",
        "eks:DescribeAddon",
        "eks:DescribeCluster",
        "eks:DescribeFargateProfile",
        "eks:DescribeNodegroup",
        "eks:DescribeUpdate",
        "eks:ListClusters",
        "eks:ListFargateProfiles",
        "eks:ListNodegroups",
        "eks:ListTagsForResource",
        "eks:ListUpdates",
        "eks:TagResource",
        "eks:UntagResource",
        "eks:UpdateClusterConfig",
        "eks:UpdateClusterVersion",
        "eks:UpdateNodegroupConfig",
```

```
        "eks:UpdateNodegroupVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "VPCPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:AssociateRouteTable",
        "ec2:AttachInternetGateway",
        "ec2:CreateInternetGateway",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateVpc",
        "ec2:DeleteInternetGateway",
        "ec2:DeleteRoute",
        "ec2:DeleteRouteTable",
        "ec2:DeleteSubnet",
        "ec2:DeleteTags",
        "ec2:DeleteVpc",
        "ec2:DescribeVpcs",
        "ec2:DetachInternetGateway",
        "ec2:DisassociateRouteTable",
        "ec2:ModifySubnetAttribute",
        "ec2:ModifyVpcAttribute",
        "ec2:ReplaceRoute"
      ],
      "Resource": "*"
    }
  ]
}
```

When you create an EKS cluster, Rancher creates a service role with the following trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Effect": "Allow",
```

```
      "Sid": ""
    }
  ]
}
```

This role also has two role policy attachments with the following policies' ARNs:

```
arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
arn:aws:iam::aws:policy/AmazonEKSServicePolicy
```

#### EBS CSI Driver addon Permissions

The following are the required permissions for installing the Amazon EBS CSI Driver add-on.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:AssociateIdentityProviderConfig",
        "eks:CreateAddon",
        "eks:DeleteAddon",
        "eks:DescribeAddonConfiguration",
        "eks:DescribeAddonVersions",
        "eks:DescribeIdentityProviderConfig",
        "eks:ListAddons",
        "eks:ListIdentityProviderConfigs",
        "eks:UpdateAddon",
        "iam:CreateOpenIDConnectProvider",
        "iam:ListOpenIDConnectProviders",
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": "*"
    }
  ]
}
```

#### IPv6 Dual-Stack Networking Permissions

The following are the additional permissions required for provisioning an **IPv6** EKS cluster from Rancher.



These permissions are only necessary when Rancher is **creating** a new IPv6 cluster. During new cluster creation, Rancher needs these permissions to generate the dual-stack VPC, create the IAM OIDC provider, and assign an inline policy

(`RancherManaged_AllowIPv6ForCNI`) to the Node Instance Role.

If you are registering (importing) an existing EKS cluster that already has IPv6 configured, these additional permissions are **not required**.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AssignIpv6Addresses",
        "ec2:UnassignIpv6Addresses",
        "ec2:AssignPrivateIpAddresses",
        "ec2:UnassignPrivateIpAddresses",
        "ec2:AssociateVpcCidrBlock",
        "ec2:DisassociateVpcCidrBlock",
        "iam:PutRolePolicy",
        "iam:CreateOpenIDConnectProvider",
        "iam:ListOpenIDConnectProviders",
        "eks:AssociateIdentityProviderConfig",
        "eks:DescribeIdentityProviderConfig",
        "eks:ListIdentityProviderConfigs",
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": "*"
    }
  ]
}
```

#### Syncing

The EKS provisioner can synchronize the state of an EKS cluster between Rancher and the provider. For an in-depth technical explanation of how this works, see Syncing.

For information on configuring the refresh interval, refer to this section.

#### Troubleshooting

If your changes were overwritten, it could be due to the way the cluster data is synced with EKS. Changes shouldn't be made to the cluster from another source, such as in the EKS console, and in Rancher within a five-minute span. For information on how this works and how to configure the refresh interval, refer to Syncing.

If an unauthorized error is returned while attempting to modify or register the cluster and the cluster was not created with the role or user that your credentials belong to, refer to Security and Compliance.

For any issues or troubleshooting details for your Amazon EKS Kubernetes cluster, please see this

documentation [https://docs.aws.amazon.com/eks/latest/userguide/troubleshooting.html].

### Programmatically Creating EKS Clusters

The most common way to programmatically deploy EKS clusters through Rancher is by using the Rancher2 Terraform provider. The documentation for creating clusters with Terraform is here. [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster]

### EKS Cluster Configuration Reference

#### Account Access

Complete each drop-down and field using the information obtained for your IAM policy.

| Setting | Description |
|---|---|
| Region | From the drop-down choose the geographical region in which to build your cluster. |
| Cloud Credentials | Select the cloud credentials that you created for your IAM policy. For more information on creating cloud credentials in Rancher, refer to this page. |

#### Service Role

Choose a service role [https://docs.aws.amazon.com/IAM/latest/UserGuide/using-service-linked-roles.html].

| Service Role | Description |
|---|---|
| Standard: Rancher generated service role | If you choose this role, Rancher automatically adds a service role for use with the cluster. |
| Custom: Choose from your existing service roles | If you choose this role, Rancher lets you choose from service roles that you're already created within AWS. For more information on creating a custom service role in AWS, see the Amazon documentation [https://docs.aws.amazon.com/IAM/latest/UserGuide/using-service-linked-roles.html#create-service-linked-role]. |

#### Secrets Encryption

Optional: To encrypt secrets, select or enter a key created in AWS Key Management Service (KMS) [https://docs.aws.amazon.com/kms/latest/developerguide/overview.html]

#### API Server Endpoint Access

Configuring Public/Private API access is an advanced use case. For details, refer to the EKS cluster endpoint access control documentation. [https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html]

#### Private-only API Endpoints

If you enable private and disable public API endpoint access when creating a cluster, then there is an extra step you must take in order for Rancher to connect to the cluster successfully. In this case, a pop-up will be displayed with a command that you will run on the cluster to register it with Rancher. Once the cluster is provisioned, you can run the displayed command anywhere you can connect to the cluster's Kubernetes API.

There are two ways to avoid this extra manual step:

- You can create the cluster with both private and public API endpoint access on cluster creation. You can disable public access after the cluster is created and in an active state and Rancher will continue to communicate with the EKS cluster.
- You can ensure that Rancher shares a subnet with the EKS cluster. Then security groups can be used to enable Rancher to communicate with the cluster's API endpoint. In this case, the command to register the cluster is not needed, and Rancher will be able to communicate with your cluster. For more information on configuring security groups, refer to the security groups documentation [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html].

**Public Access Endpoints**

Optionally limit access to the public endpoint via explicit CIDR blocks.

If you limit access to specific CIDR blocks, then it is recommended that you also enable the private access to avoid losing network communication to the cluster.

One of the following is required to enable private access:

- Rancher's IP must be part of an allowed CIDR block
- Private access should be enabled, and Rancher must share a subnet with the cluster and have network access to the cluster, which can be configured with a security group

For more information about public and private access to the cluster endpoint, refer to the Amazon EKS documentation. [https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html]

**IPv6 / Dual-stack Networking**

Rancher supports provisioning and managing Amazon EKS clusters with IPv6 network routing. When IPv6 is enabled, Kubernetes pods and services are assigned IPv6 addresses. However, the underlying EC2 worker nodes operate in a **dual-stack mode** (receiving both IPv4 and IPv6 addresses). This dual-stack architecture ensures that the nodes can still communicate with the AWS API and the Kubernetes control plane over IPv4, while your application workloads communicate over IPv6.

To provision a dual-stack cluster from the Rancher UI:

1. During cluster creation, expand the **Networking** section.
2. Under **IP Family**, select the **IPv6** radio button.



Changing the IP Family clears any current subnet selections you have made in the form.

3. Under **VPCs and Subnets**, choose either to **Create a new vpc and subnet automatically** or **Select from existing subnets**.



*Warning for Custom VPCs*

If you choose to select existing subnets, you **must** select Dual-Stack subnets. The selected subnets must have both an IPv4 CIDR block and an IPv6 CIDR block. IPv6-only subnets are not supported.

4. Complete the remainder of the cluster configuration (Node Groups, etc.).
5. Click **Create**.



The IP Family setting is **immutable**. After an EKS cluster is created, you cannot convert an IPv4 cluster to IPv6, nor can you convert an IPv6 cluster to IPv4.

#### Importing Existing IPv6 Clusters

Rancher fully supports registering (importing) existing Amazon EKS clusters that were configured with IPv6 networking outside of Rancher (e.g., via Terraform or the AWS Console).

When you register an existing EKS cluster:

1. Follow the standard cluster registration process (**Cluster Management > Add Cluster > Generic**).
2. Rancher queries the AWS EKS API to read the cluster's upstream state.
3. Rancher automatically detects if the cluster was provisioned with IPv6.

When configuring an IPv6 cluster, several automated behaviors and strict requirements apply:

- **Service CIDR:** AWS automatically assigns the Service CIDR from a fixed IPv6 range (`fd00::/108`). You cannot customize the Service CIDR for an IPv6 cluster.
- **OIDC Provider (IRSA) Requirement:** In an IPv6 cluster, the Amazon VPC CNI plugin strictly requires IAM permissions to assign IPv6 prefixes to Elastic Network Interfaces (ENIs). This authentication is handled via IAM Roles for Service Accounts (IRSA). Therefore, **an IAM OIDC provider is mandatory and is automatically enabled by Rancher** when provisioning an IPv6 cluster. Without it, pods fail to acquire IP addresses.

#### EKSClusterConfig Reference Example (IPv6)

If you are programmatically deploying EKS clusters using the `eksclusterconfigs.eks.cattle.io` Custom Resource, you can enable IPv6 by setting the `ipFamily` field to `ipv6`.

Below is an example of a minimal `EKSClusterConfig` configured for IPv6. Notice that the `ipFamily` is set, and standard IPv4 fields like `serviceCidr` are omitted because AWS manages them automatically in IPv6 mode.

```
apiVersion: eks.cattle.io/v1
kind: EKSClusterConfig
metadata:
  name: my-ipv6-cluster
  namespace: cluster-fleet-default
spec:
  amazonCredentialSecret: cattle-global-data/my-aws-credentials
  displayName: my-ipv6-cluster
  region: us-west-2
  imported: false
  kubernetesVersion: "1.33"
  ipFamily: "ipv6" # Enables Dual-Stack IPv6 networking. Triggers
automatic OIDC creation.
  nodeGroups:
    - nodegroupName: initial-nodegroup
```

```
      desiredSize: 2
      maxSize: 3
      minSize: 1
      instanceType: t3.medium
      diskSize: 20
      requestSpotInstances: false
      version: "1.33"
  privateAccess: false
  publicAccess: true
  secretsEncryption: false
```

#### Subnet

| Option | Description |
|---|---|
| Standard: Rancher generated VPC and Subnet | While provisioning your cluster, Rancher generates a new VPC with 3 public subnets. |
| Custom: Choose from your existing VPC and Subnets | While provisioning your cluster, Rancher configures your Control Plane and nodes to use a VPC and Subnet that you've already created in AWS [https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html]. |

For more information, refer to the AWS documentation for Cluster VPC Considerations [https://docs.aws.amazon.com/eks/latest/userguide/network_reqs.html]. Follow one of the sets of instructions below based on your selection from the previous step.



*Warning for Custom VPCs*

If you have selected IPv6 as your IP Family and are bringing a custom VPC, you **must** select Dual-Stack subnets. The selected subnets must have **both an IPv4 CIDR block and an IPv6 CIDR block**. "IPv6-only" subnets are not supported by EKS; EC2 worker nodes still require an IPv4 address to join the cluster and communicate with AWS services.

- What Is Amazon VPC? [https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html]
- VPCs and Subnets [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html]

#### Security Group

Amazon Documentation:

- Cluster Security Group Considerations [https://docs.aws.amazon.com/eks/latest/userguide/sec-group-reqs.html]
- Security Groups for Your VPC [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html]
- Create a Security Group [https://docs.aws.amazon.com/vpc/latest/userguide/getting-started-ipv4.html#getting-started-create-security-group]

#### Logging

Configure control plane logs to send to Amazon CloudWatch. You are charged the standard CloudWatch Logs data ingestion and storage costs for any logs sent to CloudWatch Logs from your clusters.

Each log type corresponds to a component of the Kubernetes control plane. To learn more about these

components, see Kubernetes Components [https://kubernetes.io/docs/concepts/overview/components/] in the Kubernetes documentation.

For more information on EKS control plane logging, refer to the official documentation. [https://docs.aws.amazon.com/eks/latest/userguide/control-plane-logs.html]

#### Managed Node Groups

Amazon EKS managed node groups automate the provisioning and lifecycle management of nodes (Amazon EC2 instances) for Amazon EKS Kubernetes clusters.

For more information about how node groups work and how they are configured, refer to the EKS documentation. [https://docs.aws.amazon.com/eks/latest/userguide/managed-node-groups.html]

#### User-provided Launch Templates

You can provide your own launch template ID and version to configure the EC2 instances in a node group. If you provide the launch template, none of the template settings will be configurable from Rancher. You must set all of the required options listed below in your launch template.

Also, if you provide the launch template, you can only update the template version, not the template ID. To use a new template ID, create a new managed node group.

| Option | Description | Required/Optional |
|---|---|---|
| Instance Type | Choose the hardware specs [https://aws.amazon.com/ec2/instance-types/] for the instance you're provisioning. | Required |
| Image ID | Specify a custom AMI for the nodes. Custom AMIs used with EKS must be configured properly [https://aws.amazon.com/premiumsupport/knowledge-center/eks-custom-linux-ami/] | Optional |
| Node Volume Size | The launch template must specify an EBS volume with the desired size | Required |
| SSH Key | A key to be added to the instances to provide SSH access to the nodes | Optional |
| User Data | Cloud init script in MIME multi-part format [https://docs.aws.amazon.com/eks/latest/userguide/launch-templates.html#launch-template-user-data] | Optional |
| Instance Resource Tags | Tag each EC2 instance and its volumes in the node group | Optional |



You can't directly update a node group to a newer Kubernetes version if the node group was created from a custom launch template. You must create a new launch template with the proper Kubernetes version, and associate the node group with the new template.

#### Rancher-managed Launch Templates

If you do not specify a launch template, then you will be able to configure the above options in the Rancher UI and all of them can be updated after creation. In order to take advantage of all of these options, Rancher will create and manage a launch template for you. Each cluster in Rancher will have one Rancher-managed

launch template and each managed node group that does not have a specified launch template will have one version of the managed launch template. The name of this launch template will have the prefix "rancher-managed-lt-" followed by the display name of the cluster. In addition, the Rancher-managed launch template will be tagged with the key "rancher-managed-template" and value "do-not-modify-or-delete" to help identify it as Rancher-managed. It is important that this launch template and its versions not be modified, deleted, or used with any other clusters or managed node groups. Doing so could result in your node groups being "degraded" and needing to be destroyed and recreated.

#### Custom AMIs

If you specify a custom AMI, whether in a launch template or in Rancher, then the image must be configured properly [https://aws.amazon.com/premiumsupport/knowledge-center/eks-custom-linux-ami/] and you must provide user data to bootstrap the node [https://docs.aws.amazon.com/eks/latest/userguide/launch-templates.html#launch-template-custom-ami]. This is considered an advanced use case and understanding the requirements is imperative.

If you specify a launch template that does not contain a custom AMI, then Amazon will use the EKS-optimized AMI [https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-ami.html] for the Kubernetes version and selected region. You can also select a GPU enabled instance [https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-ami.html#gpu-ami] for workloads that would benefit from it.



The GPU enabled instance setting in Rancher is ignored if a custom AMI is provided, either in the dropdown or in a launch template.

#### Spot instances

Spot instances are now supported by EKS [https://docs.aws.amazon.com/eks/latest/userguide/managed-node-groups.html#managed-node-group-capacity-types-spot]. If a launch template is specified, Amazon recommends that the template not provide an instance type. Instead, Amazon recommends providing multiple instance types. If the "Request Spot Instances" checkbox is enabled for a node group, then you will have the opportunity to provide multiple instance types.



Any selection you made in the instance type dropdown will be ignored in this situation and you must specify at least one instance type to the "Spot Instance Types" section. Furthermore, a launch template used with EKS cannot request spot instances. Requesting spot instances must be part of the EKS configuration.

#### Node Group Settings

The following settings are also configurable. All of these except for the "Node Group Name" are editable after the node group is created.

| Option | Description |
|---|---|
| Node Group Name | The name of the node group. |
| Desired ASG Size | The desired number of instances. |
| Maximum ASG Size | The maximum number of instances. This setting won't take effect until the Cluster Autoscaler [https://docs.aws.amazon.com/eks/latest/userguide/cluster-autoscaler.html] is installed. |
| Minimum ASG Size | The minimum number of instances. This setting won't take effect until the Cluster Autoscaler [https://docs.aws.amazon.com/eks/latest/userguide/cluster-autoscaler.html] is installed. |

| Option | Description |
|---|---|
| Labels | Kubernetes labels applied to the nodes in the managed node group. |
| Tags | These are tags for the managed node group and do not propagate to any of the associated resources. |

#### Self-managed Amazon Linux Nodes

You can register an EKS cluster containing self-managed Amazon Linux nodes. You must configure this type of cluster according to the instructions in the official AWS documentation for launching self-managed Amazon Linux nodes [https://docs.aws.amazon.com/eks/latest/userguide/launch-workers.html]. EKS clusters containing self-managed Amazon Linux nodes are usually operated by the Karpenter [https://karpenter.sh/docs/] project. After you provision an EKS cluster containing self-managed Amazon Linux nodes, register the cluster so it can be managed by Rancher. However, the nodes won't be visible in the Rancher UI.

#### IAM Roles for Service Accounts

An Applications Deployment running on an EKS cluster can make requests to AWS services via IAM permissions. These applications must sign their requests with AWS credentials. IAM roles for service accounts manage these credentials using an AWS OIDC endpoint. Rather than distributing AWS credentials to containers or relying on an EC2 instance's role, you can link an IAM role to a Kubernetes service account [https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts.html] and configure your Pods to use this account.



Linking to an IAM role is not supported for Rancher pods in an EKS cluster.

To enable IAM roles for service accounts:

1. Create an IAM OIDC provider for your cluster [https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html]
2. Configure a Kubernetes service account to assume an IAM role [https://docs.aws.amazon.com/eks/latest/userguide/associate-service-account-role.html]
3. Configure Pods to use a Kubernetes service account [https://docs.aws.amazon.com/eks/latest/userguide/pod-configuration.html]
4. Use a supported AWS SDK [https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts-minimum-sdk.html]

#### Configuring the Refresh Interval

The `eks-refresh-cron` setting is deprecated. It has been migrated to the `eks-refresh` setting, which is an integer representing seconds.

The default value is 300 seconds.

The syncing interval can be changed by running `kubectl edit setting eks-refresh`.

If the `eks-refresh-cron` setting was previously set, the migration will happen automatically.

The shorter the refresh window, the less likely any race conditions will occur, but it does increase the likelihood of encountering request limits that may be in place for AWS APIs.

## Creating a GKE Cluster

### Prerequisites

Some setup in Google Kubernetes Engine is required.

#### Service Account Token

Create a service account using Google Kubernetes Engine [https://console.cloud.google.com/projectselector/iam-admin/serviceaccounts]. GKE uses this account to operate your cluster. Creating this account also generates a private key used for authentication.

The service account requires the following roles:

- **Compute Viewer:** `roles/compute.viewer`
- **Project Viewer:** `roles/viewer`
- **Kubernetes Engine Admin:** `roles/container.admin`
- **Service Account User:** `roles/iam.serviceAccountUser`

Google Documentation: Creating and Enabling Service Accounts [https://cloud.google.com/compute/docs/access/create-enable-service-accounts-for-instances]

For help obtaining a private key for your service account, refer to the Google cloud documentation here. [https://cloud.google.com/iam/docs/creating-managing-service-account-keys#creating_service_account_keys] You will need to save the key in JSON format.

#### Google Project ID

Your cluster will need to be part of a Google Project.

To create a new project, refer to the Google cloud documentation here. [https://cloud.google.com/resource-manager/docs/creating-managing-projects#creating_a_project]

To get the project ID of an existing project, refer to the Google cloud documentation here. [https://cloud.google.com/resource-manager/docs/creating-managing-projects#identifying_projects]

### Provisioning a GKE Cluster



Deploying to GKE will incur charges.

#### 1. Create a Cloud Credential

1. Click ☰ > **Cluster Management**.
2. In the left navigation bar, click **Cloud Credentials**.
3. Click **Create**.
4. Enter a name for your Google cloud credentials.
5. In the **Service Account** text box, paste your service account private key JSON, or upload the JSON file.
6. Click **Create**.

**Result:** You have created credentials that Rancher will use to provision the new GKE cluster.

#### 2. Create the GKE Cluster

Use Rancher to set up and configure your Kubernetes cluster. To successfully create a GKE cluster with Rancher, your GKE must be in Standard mode. GKE has two modes of operation when creating a Kubernetes cluster, Autopilot and Standard mode. The cluster configuration for Autopilot mode has restrictions on editing the kube-system namespace. However, Rancher needs to create resources in the kube-system namespace during installation. As a result, you will not be able to create a GKE cluster in Autopilot mode. For more information about the difference between GKE Autopilot mode and Standard mode, visit Compare GKE Autopilot and Standard. [https://cloud.google.com/kubernetes-engine/docs/resources/autopilot-standard-feature-comparison]

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **Google GKE**.
4. Enter a **Cluster Name**.
5. Optional: Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
6. Optional: Add Kubernetes labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] or annotations [https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations/] to the cluster.
7. Enter your Google project ID and your Google cloud credentials.
8. Fill out the rest of the form. For help, refer to the GKE cluster configuration reference.
9. Click **Create**.

**Result:** You have successfully deployed a GKE cluster.

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

#### Private Clusters

Private GKE clusters are supported. Note: This advanced setup can require more steps during the cluster provisioning process. For details, see this section.

#### Configuration Reference

For details on configuring GKE clusters in Rancher, see this page.

#### Updating Kubernetes Version

The Kubernetes version of a cluster can be upgraded to any version available in the region or zone for the GKE cluster. Upgrading the master Kubernetes version does not automatically upgrade worker nodes. Nodes can be upgraded independently.



GKE has removed basic authentication in 1.19+. In order to upgrade a cluster to 1.19+,

basic authentication must be disabled in the Google Cloud. Otherwise, an error will appear in Rancher when an upgrade to 1.19+ is attempted. You can follow the Google documentation [https://cloud.google.com/kubernetes-engine/docs/how-to/api-server-authentication#disabling_authentication_with_a_static_password]. After this, the Kubernetes version can be updated to 1.19+ via Rancher.

### Syncing

The GKE provisioner can synchronize the state of a GKE cluster between Rancher and the provider. For an in-depth technical explanation of how this works, see Syncing.

For information on configuring the refresh interval, see this section.

### Programmatically Creating GKE Clusters

The most common way to programmatically deploy GKE clusters through Rancher is by using the Rancher2 Terraform provider. The documentation for creating clusters with Terraform is here. [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster]

### GKE Cluster Configuration Reference

#### Cluster Location

| Value | Description |
|---|---|
| Location Type | Zonal or Regional. With GKE, you can create a cluster tailored to the availability requirements of your workload and your budget. By default, a cluster's nodes run in a single compute zone. When multiple zones are selected, the cluster's nodes will span multiple compute zones, while the controlplane is located in a single zone. Regional clusters increase the availability of the controlplane as well. For help choosing the type of cluster availability, refer to these docs. [https://cloud.google.com/kubernetes-engine/docs/best-practices/scalability#choosing_a_regional_or_zonal_control_plane] |
| Zone | Each region in Compute engine contains a number of zones. For more information about available regions and zones, refer to these docs. [https://cloud.google.com/compute/docs/regions-zones#available] |
| Additional Zones | For zonal clusters, you can select additional zones to create a multi-zone cluster. [https://cloud.google.com/kubernetes-engine/docs/concepts/types-of-clusters#multi-zonal_clusters] |
| Region | For regional clusters, [https://cloud.google.com/kubernetes-engine/docs/concepts/types-of-clusters#regional_clusters] you can select a region. For more information about available regions and zones, refer to this section [https://cloud.google.com/compute/docs/regions-zones#available]. The first part of each zone name is the name of the region. |

#### Cluster Options

#### Kubernetes Version

*Mutable: yes*

For more information on GKE Kubernetes versions, refer to these docs. [https://cloud.google.com/kubernetes-engine/versioning]

#### Container Address Range

*Mutable: no*

The IP address range for pods in the cluster. Must be a valid CIDR range, e.g. 10.42.0.0/16. If not specified, a random range is automatically chosen from 10.0.0.0/8 and will exclude ranges already allocated to VMs, other clusters, or routes. Automatically chosen ranges may conflict with reserved IP addresses, dynamic routes, or routes within VPCs peering with the cluster.

#### Network

*Mutable: no*

The Compute Engine Network that the cluster connects to. Routes and firewalls will be created using this network. If using Shared VPCs [https://cloud.google.com/vpc/docs/shared-vpc], the VPC networks that are shared to your project will appear here. will be available to select in this field. For more information, refer to this page [https://cloud.google.com/vpc/docs/vpc#vpc_networks_and_subnets].

#### Node Subnet / Subnet

*Mutable: no*

The Compute Engine subnetwork that the cluster connects to. This subnetwork must belong to the network specified in the **Network** field. Select an existing subnetwork, or select "Auto Create Subnetwork" to have one automatically created. If not using an existing network, **Subnetwork Name** is required to generate one. If using Shared VPCs [https://cloud.google.com/vpc/docs/shared-vpc], the VPC subnets that are shared to your project will appear here. If using a Shared VPC network, you cannot select "Auto Create Subnetwork". For more information, refer to this page. [https://cloud.google.com/vpc/docs/vpc#vpc_networks_and_subnets]

#### Subnetwork Name

*Mutable: no*

Automatically create a subnetwork with the provided name. Required if "Auto Create Subnetwork" is selected for **Node Subnet** or **Subnet**. For more information on subnetworks, refer to this page. [https://cloud.google.com/vpc/docs/vpc#vpc_networks_and_subnets]

#### Ip Aliases

*Mutable: no*

Enable alias IPs [https://cloud.google.com/vpc/docs/alias-ip]. This enables VPC-native traffic routing. Required if using Shared VPCs [https://cloud.google.com/vpc/docs/shared-vpc].

#### Network Policy

*Mutable: yes*

Enable network policy enforcement on the cluster. A network policy defines the level of communication that can occur between pods and services in the cluster. For more information, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/how-to/network-policy]

#### Project Network Isolation

*Mutable: yes*

Choose whether to enable or disable inter-project communication.

#### Imported Clusters

For imported clusters, Project Network Isolation (PNI) requires Kubernetes Network Policy to be enabled on the cluster beforehand. For clusters created by Rancher, Rancher enables Kubernetes Network Policy automatically.

1. In GKE, enable Network Policy at the cluster level. Refer to the official GKE guide [https://cloud.google.com/kubernetes-engine/docs/how-to/network-policy] for instructions.
2. After enabling Network Policy, import the cluster into Rancher and enable PNI for project-level isolation.

#### Node Ipv4 CIDR Block

*Mutable: no*

The IP address range of the instance IPs in this cluster. Can be set if "Auto Create Subnetwork" is selected for **Node Subnet** or **Subnet**. Must be a valid CIDR range, e.g. 10.96.0.0/14. For more information on how to determine the IP address range, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips#cluster_sizing]

#### Cluster Secondary Range Name

*Mutable: no*

The name of an existing secondary range for Pod IP addresses. If selected, **Cluster Pod Address Range** will automatically be populated. Required if using a Shared VPC network.

#### Cluster Pod Address Range

*Mutable: no*

The IP address range assigned to pods in the cluster. Must be a valid CIDR range, e.g. 10.96.0.0/11. If not provided, will be created automatically. Must be provided if using a Shared VPC network. For more information on how to determine the IP address range for your pods, refer to this section. [https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips#cluster_sizing_secondary_range_pods]

#### Services Secondary Range Name

*Mutable: no*

The name of an existing secondary range for service IP addresses. If selected, **Service Address Range** will be automatically populated. Required if using a Shared VPC network.

#### Service Address Range

*Mutable: no*

The address range assigned to the services in the cluster. Must be a valid CIDR range, e.g. 10.94.0.0/18. If not provided, will be created automatically. Must be provided if using a Shared VPC network. For more information on how to determine the IP address range for your services, refer to this section.

[https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips#cluster_sizing_secondary_range_svcs]

#### Private Cluster

*Mutable: no*



Private clusters require additional planning and configuration outside of Rancher. Refer to the private cluster guide.

Assign nodes only internal IP addresses. Private cluster nodes cannot access the public internet unless additional networking steps are taken in GCP.

#### Enable Private Endpoint



Private clusters require additional planning and configuration outside of Rancher. Refer to the private cluster guide.

*Mutable: no*

Locks down external access to the control plane endpoint. Only available if **Private Cluster** is also selected. If selected, and if Rancher does not have direct access to the Virtual Private Cloud network the cluster is running in, Rancher will provide a registration command to run on the cluster to enable Rancher to connect to it.

#### Master IPV4 CIDR Block

*Mutable: no*

The IP range for the control plane VPC.

#### Master Authorized Network

*Mutable: yes*

Enable control plane authorized networks to block untrusted non-GCP source IPs from accessing the Kubernetes master through HTTPS. If selected, additional authorized networks may be added. If the cluster is created with a public endpoint, this option is useful for locking down access to the public endpoint to only certain networks, such as the network where your Rancher service is running. If the cluster only has a private endpoint, this setting is required.

#### Additional Options

#### Cluster Addons

Additional Kubernetes cluster components. For more information, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/reference/rest/v1/projects.locations.clusters#Cluster.AddonsConfig]

#### Horizontal Pod Autoscaling

*Mutable: yes*

The Horizontal Pod Autoscaler changes the shape of your Kubernetes workload by automatically increasing or decreasing the number of Pods in response to the workload's CPU or memory consumption, or in response to custom metrics reported from within Kubernetes or external metrics from sources outside of

your cluster. For more information, see this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/horizontalpodautoscaler]

#### HTTP (L7) Load Balancing

*Mutable: yes*

HTTP (L7) Load Balancing distributes HTTP and HTTPS traffic to backends hosted on GKE. For more information, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/tutorials/http-balancer]

#### Network Policy Config (master only)

*Mutable: yes*

Configuration for NetworkPolicy. This only tracks whether the addon is enabled or not on the master, it does not track whether network policy is enabled for the nodes.

#### Cluster Features (Alpha Features)

*Mutable: no*

Turns on all Kubernetes alpha API groups and features for the cluster. When enabled, the cluster cannot be upgraded and will be deleted automatically after 30 days. Alpha clusters are not recommended for production use as they are not covered by the GKE SLA. For more information, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/alpha-clusters]

#### Logging Service

*Mutable: yes*

The logging service the cluster uses to write logs. Use either Cloud Logging [https://cloud.google.com/logging] or no logging service in which case no logs are exported from the cluster.

#### Monitoring Service

*Mutable: yes*

The monitoring service the cluster uses to write metrics. Use either Cloud Monitoring [https://cloud.google.com/monitoring] or monitoring service in which case no metrics are exported from the cluster.

#### Maintenance Window

*Mutable: yes*

Set the start time for a 4 hour maintenance window. The time is specified in the UTC time zone using the HH:MM format. For more information, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/maintenance-windows-and-exclusions]

#### Node Pools

In this section, enter details describing the configuration of each node in the node pool.

#### Kubernetes Version

*Mutable: yes*

The Kubernetes version for each node in the node pool. For more information on GKE Kubernetes versions, refer to these docs. [https://cloud.google.com/kubernetes-engine/versioning]

#### Image Type

*Mutable: yes*

The node operating system image. For more information for the node image options that GKE offers for each OS, refer to this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/node-images#available_node_images]



The default option is "Container-Optimized OS with Docker". The read-only filesystem on GCP's Container-Optimized OS is not compatible with the legacy logging implementation in Rancher. If you need to use the legacy logging feature, select "Ubuntu with Docker" or "Ubuntu with Containerd". The current logging feature is compatible with the Container-Optimized OS image.



If selecting "Windows Long Term Service Channel" or "Windows Semi-Annual Channel" for the node pool image type, you must also add at least one Container-Optimized OS or Ubuntu node pool.

#### Machine Type

*Mutable: no*

The virtualized hardware resources available to node instances. For more information on Google Cloud machine types, refer to this page. [https://cloud.google.com/compute/docs/machine-types#machine_types]

#### Root Disk Type

*Mutable: no*

Standard persistent disks are backed by standard hard disk drives (HDD), while SSD persistent disks are backed by solid state drives (SSD). For more information, refer to this section. [https://cloud.google.com/compute/docs/disks]

#### Local SSD Disks

*Mutable: no*

Configure each node's local SSD disk storage in GB. Local SSDs are physically attached to the server that hosts your VM instance. Local SSDs have higher throughput and lower latency than standard persistent disks or SSD persistent disks. The data that you store on a local SSD persists only until the instance is stopped or deleted. For more information, see this section. [https://cloud.google.com/compute/docs/disks#localssds]

#### Preemptible nodes (beta)

*Mutable: no*

Preemptible nodes, also called preemptible VMs, are Compute Engine VM instances that last a maximum of 24 hours in general, and provide no availability guarantees. For more information, see this page. [https://cloud.google.com/kubernetes-engine/docs/how-to/preemptible-vms]

#### Taints

*Mutable: no*

When you apply a taint to a node, only Pods that tolerate the taint are allowed to run on the node. In a GKE cluster, you can apply a taint to a node pool, which applies the taint to all nodes in the pool.

#### Node Labels

*Mutable: no*

You can apply labels to the node pool, which applies the labels to all nodes in the pool.

Invalid labels can prevent upgrades or can prevent Rancher from starting. For details on label syntax requirements, see the Kubernetes documentation. [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set]

#### Network Tags

*Mutable: no*

You can add network tags to the node pool to make firewall rules and routes between subnets. Tags will apply to all nodes in the pool.

For details on tag syntax and requirements, see the Kubernetes documentation [https://cloud.google.com/vpc/docs/add-remove-network-tags].

#### Group Details

In this section, enter details describing the node pool.

#### Name

*Mutable: no*

Enter a name for the node pool.

#### Initial Node Count

*Mutable: yes*

Integer for the starting number of nodes in the node pool.

#### Max Pod Per Node

*Mutable: no*

GKE has a hard limit of 110 Pods per node. For more information on the Kubernetes limits, see this section. [https://cloud.google.com/kubernetes-engine/docs/best-practices/scalability#dimension_limits]

#### Autoscaling

*Mutable: yes*

Node pool autoscaling dynamically creates or deletes nodes based on the demands of your workload. For more information, see this page. [https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-autoscaler]

#### Auto Repair

*Mutable: yes*

GKE's node auto-repair feature helps you keep the nodes in your cluster in a healthy, running state. When enabled, GKE makes periodic checks on the health state of each node in your cluster. If a node fails consecutive health checks over an extended time period, GKE initiates a repair process for that node. For more information, see the section on auto-repairing nodes. [https://cloud.google.com/kubernetes-engine/docs/how-to/node-auto-repair]

#### Auto Upgrade

*Mutable: yes*

When enabled, the auto-upgrade feature keeps the nodes in your cluster up-to-date with the cluster control plane (master) version when your control plane is updated on your behalf. [https://cloud.google.com/kubernetes-engine/upgrades#automatic_cp_upgrades] For more information about auto-upgrading nodes, see this page. [https://cloud.google.com/kubernetes-engine/docs/how-to/node-auto-upgrades]

#### Access Scopes

*Mutable: no*

Access scopes are the legacy method of specifying permissions for your nodes.

- **Allow default access:** The default access for new clusters is the Compute Engine default service account. [https://cloud.google.com/compute/docs/access/service-accounts?hl=en_US#default_service_account]
- **Allow full access to all Cloud APIs:** Generally, you can just set the cloud-platform access scope to allow full access to all Cloud APIs, then grant the service account only relevant IAM roles. The combination of access scopes granted to the virtual machine instance and the IAM roles granted to the service account determines the amount of access the service account has for that instance.
- **Set access for each API:** Alternatively, you can choose to set specific scopes that permit access to the particular API methods that the service will call.

For more information, see the section about enabling service accounts for a VM. [https://cloud.google.com/compute/docs/access/create-enable-service-accounts-for-instances]

#### Configuring the Refresh Interval

The refresh interval can be configured through the setting "gke-refresh", which is an integer representing seconds.

The default value is 300 seconds.

The syncing interval can be changed by running `kubectl edit setting gke-refresh`.

The shorter the refresh window, the less likely any race conditions will occur, but it does increase the likelihood of encountering request limits that may be in place for GCP APIs.

### Private Clusters

In GKE, private clusters [https://cloud.google.com/kubernetes-engine/docs/concepts/private-cluster-concept] are clusters whose nodes are isolated from inbound and outbound traffic by assigning them internal IP

addresses only. Private clusters in GKE have the option of exposing the control plane endpoint as a publicly accessible address or as a private address. This is different from other Kubernetes providers, which may refer to clusters with private control plane endpoints as "private clusters" but still allow traffic to and from nodes. You may want to create a cluster with private nodes, with or without a public control plane endpoint, depending on your organization's networking and security requirements. A GKE cluster provisioned from Rancher can use isolated nodes by selecting "Private Cluster" in the Cluster Options (under "Show advanced options"). The control plane endpoint can optionally be made private by selecting "Enable Private Endpoint".

#### Private Nodes

Because the nodes in a private cluster only have internal IP addresses, they will not be able to install the cluster agent and Rancher will not be able to fully manage the cluster. This can be overcome in a few ways.

#### Cloud NAT



Cloud NAT will incur charges [https://cloud.google.com/nat/pricing].

If restricting outgoing internet access is not a concern for your organization, use Google's Cloud NAT [https://cloud.google.com/nat/docs/using-nat] service to allow nodes in the private network to access the internet, enabling them to download the required images from Docker Hub and contact the Rancher management server. This is the simplest solution.

#### Private Registry



This scenario is not officially supported, but is described for cases in which using the Cloud NAT service is not sufficient.

If restricting both incoming and outgoing traffic to nodes is a requirement, follow the air-gapped installation instructions to set up a private container image registry on the VPC where the cluster is going to be, allowing the cluster nodes to access and download the images they need to run the cluster agent. If the control plane endpoint is also private, Rancher will need direct access to it.

#### Private Control Plane Endpoint

If the cluster has a public endpoint exposed, Rancher will be able to reach the cluster, and no additional steps need to be taken. However, if the cluster has no public endpoint, then considerations must be made to ensure Rancher can access the cluster.

#### Cloud NAT



Cloud NAT will incur charges [https://cloud.google.com/nat/pricing].

As above, if restricting outgoing internet access to the nodes is not a concern, then Google's Cloud NAT [https://cloud.google.com/nat/docs/using-nat] service can be used to allow the nodes to access the internet. While the cluster is provisioning, Rancher will provide a registration command to run on the cluster. Download the kubeconfig [https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl] for the new cluster and run the provided kubectl command on the cluster. Gaining access to the cluster in order to run this command can be done by creating a temporary node or using an existing node in the VPC, or by logging on to or creating an SSH tunnel through one of the cluster nodes.

#### Direct Access

If the Rancher server is run on the same VPC as the cluster's control plane, it will have direct access to the control plane's private endpoint. The cluster nodes will need to have access to a private registry to download images as described above.

You can also use services from Google such as Cloud VPN [https://cloud.google.com/network-connectivity/docs/vpn/concepts/overview] or Cloud Interconnect VLAN [https://cloud.google.com/network-connectivity/docs/interconnect] to facilitate connectivity between your organization's network and your Google VPC.

## Syncing Hosted Clusters

Syncing allows Rancher to update cluster values so that they're up to date with the corresponding cluster object hosted in AKS, EKS or GKE. This enables sources other than Rancher to own a hosted cluster's state.



You may accidentally overwrite the state from one source if you simultaneously process an update from another source. This may also occur if you process an update from one source within 5 minutes of finishing an update from another source.

#### How It Works

There are two fields on the Rancher Cluster object that must be understood to understand how syncing works:

1. The config object for the cluster, located on the Spec of the Cluster:
   - For AKS, the field is called AKSConfig
   - For EKS, the field is called EKSConfig
   - For GKE, the field is called GKEConfig
2. The UpstreamSpec object
   - For AKS, this is located on the AKSStatus field on the Status of the Cluster.
   - For EKS, this is located on the EKSStatus field on the Status of the Cluster.
   - For GKE, this is located on the GKEStatus field on the Status of the Cluster.

The struct types that define these objects can be found in their corresponding operator projects:

- aks-operator [https://github.com/rancher/aks-operator/blob/master/pkg/apis/aks.cattle.io/v1/types.go]
- eks-operator [https://github.com/rancher/eks-operator/blob/master/pkg/apis/eks.cattle.io/v1/types.go]
- gke-operator [https://github.com/rancher/gke-operator/blob/master/pkg/apis/gke.cattle.io/v1/types.go]

All fields are nillable, except for the following: the cluster name, the location (region or zone), Imported, and the cloud credential reference.

The AKSConfig, EKSConfig or GKEConfig represents the desired state. Nil values are ignored. Fields that are non-nil in the config object can be thought of as managed. When a cluster is created in Rancher, all fields are non-nil and therefore managed. When a pre-existing cluster is registered in Rancher all nillable fields are set to nil and aren't managed. Those fields become managed once their value has been changed by Rancher.

UpstreamSpec represents the cluster as it is in the hosted Kubernetes provider. It's refreshed every 5 minutes. After the UpstreamSpec is refreshed, Rancher checks if the cluster has an update in progress. If it's

currently updating, nothing further is done. If it is not currently updating, any managed fields on AKSConfig, EKSConfig or GKEConfig are overwritten with their corresponding value from the recently updated UpstreamSpec.



When you import a cluster from a cloud provider into Rancher, UpstreamSpec represents the cluster state and Config is empty. If you then update the imported cluster through the Rancher UI, both UpstreamSpec and Config become non-null. Any further updates to the cluster should be applied through Rancher. This is because there is no safe way to determine if changes originating from UpstreamSpec represent the desired state or just a mismatch with Config. If you update the imported cluster through the cloud provider console after you apply any updates through the Rancher UI, the controller will deploy a rollback and the content of Config will be considered the desired state.

The effective desired state can be thought of as the UpstreamSpec, plus all non-nil fields in the AKSConfig, EKSConfig or GKEConfig. This is what is displayed in the UI.

If Rancher and another source attempt to update a cluster at the same time, or within 5 minutes of an update finishing, any managed fields are likely to get caught in a race condition. To use EKS as an example, a cluster may have PrivateAccess as a managed field. If PrivateAccess is false and then enabled in EKS console at `11:01`, and tags are updated from Rancher before `11:05`, then the value is likely to be overwritten. This can also occur if tags are updated while the cluster is still processing the update. The issue described in this example shouldn't occur if the cluster is registered and the PrivateAccess fields are nil.

# Setting up Cloud Providers

A *cloud provider* is a module in Kubernetes that provides an interface for managing nodes, load balancers, and networking routes.

When a cloud provider is set up in Rancher, the Rancher server can automatically provision new nodes, load balancers or persistent storage devices when launching Kubernetes definitions, if the cloud provider you're using supports such automation.

Your cluster will not provision correctly if you configure a cloud provider cluster of nodes that do not meet the prerequisites.

By default, the **Cloud Provider** option is set to None.

The following cloud providers can be enabled:

- Amazon
- Azure
- GCE (Google Compute Engine)
- vSphere

## Setting up the Amazon Cloud Provider

For details on enabling the Amazon cloud provider, refer to this page.

### Setting up the Azure Cloud Provider

For details on enabling the Azure cloud provider, refer to this page.

### Setting up the GCE Cloud Provider

For details on enabling the Google Compute Engine cloud provider, refer to this page.

### Setting up the VMware vSphere Cloud Provider

For details on enabling the vSphere cloud provider, refer to in-tree vSphere config and out-of-tree vSphere config.

### Setting up a Custom Cloud Provider

The `Custom` cloud provider is available if you want to configure any Kubernetes cloud provider.

For the custom cloud provider option, you can refer to the RKE docs [https://rancher.com/docs/rke/latest/en/config-options/cloud-providers/] on how to edit the yaml file for your specific cloud provider. There are specific cloud providers that have more detailed configuration:

- vSphere [https://rke.docs.rancher.com/config-options/cloud-providers/vsphere]
- OpenStack [https://rancher.com/docs/rke/latest/en/config-options/cloud-providers/openstack/]

### Setting up the Amazon Cloud Provider

> In Kubernetes 1.27 and later, you must use an out-of-tree AWS cloud provider. In-tree cloud providers have been deprecated. The Amazon cloud provider has been removed completely, and won't work after an upgrade to Kubernetes 1.27. The steps listed below are still required to set up an Amazon cloud provider. You can set up an out-of-tree cloud provider after creating an IAM role and configuring the ClusterID.
>
> You can also migrate from an in-tree to an out-of-tree AWS cloud provider on Kubernetes 1.26 and earlier. All existing clusters must migrate prior to upgrading to v1.27 in order to stay functional.
>
> Starting with Kubernetes 1.23, you must deactivate the `CSIMigrationAWS` feature gate to use the in-tree AWS cloud provider. You can do this by setting `feature-gates=CSIMigrationAWS=false` as an additional argument for the cluster's Kubelet, Controller Manager, API Server and Scheduler in the advanced cluster configuration.

When you use Amazon as a cloud provider, you can leverage the following capabilities:

- **Load Balancers:** Launch an AWS Elastic Load Balancer (ELB) when you select `Layer-4 Load Balancer` in **Port Mapping** or when you launch a `Service` with `type: LoadBalancer`.
- **Persistent Volumes**: Use AWS Elastic Block Stores (EBS) for persistent volumes.

See the cloud-provider-aws README [https://kubernetes.github.io/cloud-provider-aws/] for more information about the Amazon cloud provider.

To set up the Amazon cloud provider,

1. Create an IAM role and attach to the instances
2. Configure the ClusterID

**1. Create an IAM Role and attach to the instances**

All nodes added to the cluster must be able to interact with EC2 so that they can create and remove resources. You can enable this interaction by using an IAM role attached to the instance. See Amazon documentation: Creating an IAM Role [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html#create-iam-role] how to create an IAM role. There are two example policies:

- The first policy is for the nodes with the `controlplane` role. These nodes have to be able to create/remove EC2 resources. The following IAM policy is an example, please remove any unneeded permissions for your use case.
- The second policy is for the nodes with the `etcd` or `worker` role. These nodes only have to be able to retrieve information from EC2.

While creating an Amazon EC2 cluster, you must fill in the **IAM Instance Profile Name** (not ARN) of the created IAM role when creating the **Node Template**.

While creating a Custom cluster, you must manually attach the IAM role to the instance(s).

IAM Policy for nodes with the `controlplane` role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVolumes",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:ModifyInstanceAttribute",
        "ec2:ModifyVolume",
        "ec2:AttachVolume",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateRoute",
```

```
        "ec2:DeleteRoute",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteVolume",
        "ec2:DetachVolume",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DescribeVpcs",
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:AttachLoadBalancerToSubnets",
        "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
        "elasticloadbalancing:CreateLoadBalancer",
        "elasticloadbalancing:CreateLoadBalancerPolicy",
        "elasticloadbalancing:CreateLoadBalancerListeners",
        "elasticloadbalancing:ConfigureHealthCheck",
        "elasticloadbalancing:DeleteLoadBalancer",
        "elasticloadbalancing:DeleteLoadBalancerListeners",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DetachLoadBalancerFromSubnets",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:ModifyLoadBalancerAttributes",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
        "elasticloadbalancing:AddTags",
        "elasticloadbalancing:CreateListener",
        "elasticloadbalancing:CreateTargetGroup",
        "elasticloadbalancing:DeleteListener",
        "elasticloadbalancing:DeleteTargetGroup",
        "elasticloadbalancing:DescribeListeners",
        "elasticloadbalancing:DescribeLoadBalancerPolicies",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:ModifyListener",
        "elasticloadbalancing:ModifyTargetGroup",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
        "iam:CreateServiceLinkedRole",
        "kms:DescribeKey"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

IAM policy for nodes with the `etcd` or `worker` role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeRegions",
        "ec2:DescribeAvailabilityZones",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

**2. Configure the ClusterID**

The following resources need to tagged with a `ClusterID`:

- **Nodes**: All hosts added in Rancher.
- **Subnet**: The subnet used for your cluster.
- **Security Group**: The security group used for your cluster.



Do not tag multiple security groups. Tagging multiple groups generates an error when creating an Elastic Load Balancer (ELB).

When you create an Amazon EC2 Cluster, the `ClusterID` is automatically configured for the created nodes. Other resources still need to be manually tagged.

Use the following tag:

**Key** = `kubernetes.io/cluster/<cluster-id>` **Value** = `owned`

Setting the value of the tag to `owned` tells the cluster that all resources with this tag are owned and managed by this cluster.

If you share resources between clusters, you can change the tag to:

**Key** = `kubernetes.io/cluster/<cluster-id>` **Value** = `shared`.

The string value, `<cluster-id>`, is the Kubernetes cluster's ID.



Do not tag a resource with multiple owned or shared tags.

#### Using Amazon Elastic Container Registry (ECR)

The kubelet component has the ability to automatically obtain ECR credentials, when the IAM profile mentioned in Create an IAM Role and attach to the instances is attached to the instance(s). When using a Kubernetes version older than v1.15.0, the Amazon cloud provider needs be configured in the cluster. Starting with Kubernetes version v1.15.0, the kubelet can obtain ECR credentials without having the Amazon cloud provider configured in the cluster.

#### Using the Out-of-Tree AWS Cloud Provider

**RKE2**

1. Node name conventions and other prerequisites [https://cloud-provider-aws.sigs.k8s.io/prerequisites/] must be followed for the cloud provider to find the instance correctly.
2. Rancher managed RKE2/K3s clusters don't support configuring `providerID`. However, the engine will set the node name correctly if the following configuration is set on the provisioning cluster object:

   ```
   spec:
     rkeConfig:
       machineGlobalConfig:
         cloud-provider-name: aws
   ```

   This option will be passed to the configuration of the various Kubernetes components that run on the node, and must be overridden per component to prevent the in-tree provider from running unintentionally:

   **Override on Etcd:**

   ```
   spec:
     rkeConfig:
       machineSelectorConfig:
         - config:
             kubelet-arg:
               - cloud-provider=external
           machineLabelSelector:
             matchExpressions:
               - key: rke.cattle.io/etcd-role
                 operator: In
                 values:
                   - 'true'
   ```

**Override on Control Plane:**

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
        disable-cloud-controller: true
        kube-apiserver-arg:
          - cloud-provider=external
        kube-controller-manager-arg:
          - cloud-provider=external
        kubelet-arg:
          - cloud-provider=external
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/control-plane-role
              operator: In
              values:
                - 'true'
```

**Override on Worker:**

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          kubelet-arg:
            - cloud-provider=external
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/worker-role
              operator: In
              values:
                - 'true'
```

3. Select `Amazon` if relying on the above mechanism to set the provider ID. Otherwise, select **External (out-of-tree)** cloud provider, which sets `--cloud-provider=external` for Kubernetes components.
4. Specify the `aws-cloud-controller-manager` Helm chart as an additional manifest to install:

```
spec:
  rkeConfig:
    additionalManifest: |-
```

```
      apiVersion: helm.cattle.io/v1
      kind: HelmChart
      metadata:
        name: aws-cloud-controller-manager
        namespace: kube-system
      spec:
        chart: aws-cloud-controller-manager
        repo: https://kubernetes.github.io/cloud-provider-aws
        targetNamespace: kube-system
        bootstrap: true
        valuesContent: |-
          hostNetworking: true
          nodeSelector:
            node-role.kubernetes.io/control-plane: "true"
          args:
            - --configure-cloud-routes=false
            - --v=5
            - --cloud-provider=aws
```

**RKE**

1. Node name conventions and other prerequisites [https://cloud-provider-aws.sigs.k8s.io/prerequisites/] must be followed so that the cloud provider can find the instance. Rancher provisioned clusters don't support configuring `providerID`.



If you use IP-based naming, the nodes must be named after the instance followed by the regional domain name (`ip-xxx-xxx-xxx-xxx.ec2.<region>.internal`). If you have a custom domain name set in the DHCP options, you must set `--hostname-override` on `kube-proxy` and `kubelet` to match this naming convention.

To meet node naming conventions, Rancher allows setting `useInstanceMetadataHostname` when the `External Amazon` cloud provider is selected. Enabling `useInstanceMetadataHostname` will query ec2 metadata service and set `/hostname` as `hostname-override` for `kubelet` and `kube-proxy`:

```
rancher_kubernetes_engine_config:
  cloud_provider:
    name: external-aws
    useInstanceMetadataHostname: true
```

You must not enable `useInstanceMetadataHostname` when setting custom values for `hostname-override` for custom clusters. When you create a custom cluster, add `--node-name` to the `docker run` node registration command to set `hostname-override` — for example, `"$(hostname -f)"`. This can be done manually or by using **Show Advanced Options** in the Rancher UI to add **Node Name**.

2. Select the cloud provider.

   Selecting **External Amazon (out-of-tree)** sets `--cloud-provider=external` and enables `useInstanceMetadataHostname`. As mentioned in step 1, enabling `useInstanceMetadataHostname` will query the EC2 metadata service and set `http://169.254.169.254/latest/meta-data/hostname` as `hostname-override` for `kubelet` and `kube-proxy`.

   You must disable `useInstanceMetadataHostname` when setting a custom node name for custom clusters via `node-name`.

   ```
   rancher_kubernetes_engine_config:
     cloud_provider:
       name: external-aws
       useInstanceMetadataHostname: true/false
   ```

   Existing clusters that use an **External** cloud provider will set `--cloud-provider=external` for Kubernetes components but won't set the node name.

3. Install the AWS cloud controller manager after the cluster finishes provisioning. Note that the cluster isn't successfully provisioned and nodes are still in an `uninitialized` state until you deploy the cloud controller manager. This can be done manually, or via Helm charts in UI.

   Refer to the official AWS upstream documentation for the cloud controller manager [https://kubernetes.github.io/cloud-provider-aws].

#### Helm Chart Installation from CLI

**RKE2**

Official upstream docs for Helm chart installation [https://github.com/kubernetes/cloud-provider-aws/tree/master/charts/aws-cloud-controller-manager] can be found on GitHub.

1. Add the Helm repository:

   ```
   helm repo add aws-cloud-controller-manager
   https://kubernetes.github.io/cloud-provider-aws
   helm repo update
   ```

2. Create a `values.yaml` file with the following contents to override the default `values.yaml`:

   ```
   # values.yaml
   hostNetworking: true
   tolerations:
     - effect: NoSchedule
       key: node.cloudprovider.kubernetes.io/uninitialized
   ```

```
    value: 'true'
  - effect: NoSchedule
    value: 'true'
    key: node-role.kubernetes.io/control-plane
nodeSelector:
  node-role.kubernetes.io/control-plane: 'true'
args:
  - --configure-cloud-routes=false
  - --use-service-account-credentials=true
  - --v=2
  - --cloud-provider=aws
clusterRoleRules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
      - update
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - '*'
  - apiGroups:
      - ""
    resources:
      - nodes/status
    verbs:
      - patch
  - apiGroups:
      - ""
    resources:
      - services
    verbs:
      - list
      - patch
      - update
      - watch
  - apiGroups:
      - ""
    resources:
      - services/status
```

```
    verbs:
      - list
      - patch
      - update
      - watch
  - apiGroups:
     - ''
    resources:
      - serviceaccounts
    verbs:
    - create
    - get
  - apiGroups:
      - ""
    resources:
      - persistentvolumes
    verbs:
      - get
      - list
      - update
      - watch
  - apiGroups:
      - ""
    resources:
      - endpoints
    verbs:
      - create
      - get
      - list
      - watch
      - update
  - apiGroups:
      - coordination.k8s.io
    resources:
      - leases
    verbs:
      - create
      - get
      - list
      - watch
      - update
  - apiGroups:
      - ""
    resources:
      - serviceaccounts/token
```

```
    verbs:
      - create
```

3. Install the Helm chart:

```
helm upgrade --install aws-cloud-controller-manager aws-cloud-
controller-manager/aws-cloud-controller-manager --values
values.yaml
```

Verify that the Helm chart installed successfully:

```
helm status -n kube-system aws-cloud-controller-manager
```

4. (Optional) Verify that the cloud controller manager update succeeded:

```
kubectl rollout status daemonset -n kube-system aws-cloud-
controller-manager
```

**RKE**

Official upstream docs for Helm chart installation [https://github.com/kubernetes/cloud-provider-aws/tree/master/charts/aws-cloud-controller-manager] can be found on GitHub.

1. Add the Helm repository:

```
helm repo add aws-cloud-controller-manager
https://kubernetes.github.io/cloud-provider-aws
helm repo update
```

2. Create a `values.yaml` file with the following contents, to override the default `values.yaml`:

```
# values.yaml
hostNetworking: true
tolerations:
  - effect: NoSchedule
    key: node.cloudprovider.kubernetes.io/uninitialized
    value: 'true'
  - effect: NoSchedule
    value: 'true'
    key: node-role.kubernetes.io/controlplane
nodeSelector:
  node-role.kubernetes.io/controlplane: 'true'
args:
```

```
  - --configure-cloud-routes=false
  - --use-service-account-credentials=true
  - --v=2
  - --cloud-provider=aws
clusterRoleRules:
  - apiGroups:
      - ""
    resources:
      - events
    verbs:
      - create
      - patch
      - update
  - apiGroups:
      - ""
    resources:
      - nodes
    verbs:
      - '*'
  - apiGroups:
      - ""
    resources:
      - nodes/status
    verbs:
      - patch
  - apiGroups:
      - ""
    resources:
      - services
    verbs:
      - list
      - patch
      - update
      - watch
  - apiGroups:
      - ""
    resources:
      - services/status
    verbs:
      - list
      - patch
      - update
      - watch
  - apiGroups:
     - ''
```

```
    resources:
      - serviceaccounts
    verbs:
    - create
    - get
  - apiGroups:
      - ""
    resources:
      - persistentvolumes
    verbs:
      - get
      - list
      - update
      - watch
  - apiGroups:
      - ""
    resources:
      - endpoints
    verbs:
      - create
      - get
      - list
      - watch
      - update
  - apiGroups:
      - coordination.k8s.io
    resources:
      - leases
    verbs:
      - create
      - get
      - list
      - watch
      - update
  - apiGroups:
      - ""
    resources:
      - serviceaccounts/token
    verbs:
      - create
```

3. Install the Helm chart:

```
helm upgrade --install aws-cloud-controller-manager -n kube-
system aws-cloud-controller-manager/aws-cloud-controller-
```

```
manager --values values.yaml
```

Verify that the Helm chart installed successfully:

```
helm status -n kube-system aws-cloud-controller-manager
```

4. If present, edit the Daemonset to remove the default node selector `node-role.kubernetes.io/control-plane: ""`:

```
kubectl edit daemonset aws-cloud-controller-manager -n kube-system
```

5. (Optional) Verify that the cloud controller manager update succeeded:

```
kubectl rollout status daemonset -n kube-system aws-cloud-controller-manager
```

#### Helm Chart Installation from UI

**RKE2**

1. Click ☰, then select the name of the cluster from the left navigation.
2. Select **Apps** > **Repositories**.
3. Click the **Create** button.
4. Enter `https://kubernetes.github.io/cloud-provider-aws` in the **Index URL** field.
5. Select **Apps** > **Charts** from the left navigation and install **aws-cloud-controller-manager**.
6. Select the namespace, `kube-system`, and enable **Customize Helm options before install**.
7. Add the following container arguments:

```
- '--use-service-account-credentials=true'
- '--configure-cloud-routes=false'
```

8. Add `get` to `verbs` for `serviceaccounts` resources in `clusterRoleRules`. This allows the cloud controller manager to get service accounts upon startup.

```
- apiGroups:
    - ''
  resources:
    - serviceaccounts
  verbs:
    - create
```

```
      - get
```

9. Rancher-provisioned RKE2 nodes are tainted `node-role.kubernetes.io/control-plane`. Update tolerations and the nodeSelector:

```
tolerations:
  - effect: NoSchedule
    key: node.cloudprovider.kubernetes.io/uninitialized
    value: 'true'
  - effect: NoSchedule
    value: 'true'
    key: node-role.kubernetes.io/control-plane
```

```
nodeSelector:
  node-role.kubernetes.io/control-plane: 'true'
```



There's currently a known issue [https://github.com/rancher/dashboard/issues/9249] where nodeSelector can't be updated from the Rancher UI. Continue installing the chart and then edit the Daemonset manually to set the `nodeSelector`:

+

```
nodeSelector:
  node-role.kubernetes.io/control-plane: 'true'
```

10. Install the chart and confirm that the Daemonset `aws-cloud-controller-manager` is running. Verify `aws-cloud-controller-manager` pods are running in target namespace (`kube-system` unless modified in step 6).

**RKE**

1. Click ☰, then select the name of the cluster from the left navigation.
2. Select **Apps** > **Repositories**.
3. Click the **Create** button.
4. Enter `https://kubernetes.github.io/cloud-provider-aws` in the **Index URL** field.
5. Select **Apps** > **Charts** from the left navigation and install **aws-cloud-controller-manager**.
6. Select the namespace, `kube-system`, and enable **Customize Helm options before install**.
7. Add the following container arguments:

```
  - '--use-service-account-credentials=true'
  - '--configure-cloud-routes=false'
```

8. Add `get` to `verbs` for `serviceaccounts` resources in `clusterRoleRules`. This allows the cloud controller manager to get service accounts upon startup:

```
- apiGroups:
    - ''
  resources:
    - serviceaccounts
  verbs:
    - create
    - get
```

9. Rancher-provisioned RKE nodes are tainted `node-role.kubernetes.io/controlplane`. Update tolerations and the nodeSelector:

```
tolerations:
  - effect: NoSchedule
    key: node.cloudprovider.kubernetes.io/uninitialized
    value: 'true'
  - effect: NoSchedule
    value: 'true'
    key: node-role.kubernetes.io/controlplane
```

```
nodeSelector:
  node-role.kubernetes.io/controlplane: 'true'
```



There's currently a known issue [https://github.com/rancher/dashboard/issues/9249] where `nodeSelector` can't be updated from the Rancher UI. Continue installing the chart and then Daemonset manually to set the `nodeSelector`:

+

```
nodeSelector:
  node-role.kubernetes.io/controlplane: 'true'
```

10. Install the chart and confirm that the Daemonset `aws-cloud-controller-manager` deploys successfully:

```
kubectl rollout status deployment -n kube-system aws-cloud-
controller-manager
```

## Setting up the Azure Cloud Provider

In Kubernetes 1.30 and later, you must use an out-of-tree Azure cloud provider. The Azure cloud provider has been removed completely [https://github.com/kubernetes/kubernetes/pull/122857], and won't work after an upgrade to Kubernetes 1.30. The steps listed below are still required to set up an Azure cloud provider. You can set up an out-of-tree cloud provider after completing the prerequisites for Azure.

You can also migrate from an in-tree to an out-of-tree Azure cloud provider on Kubernetes 1.29 and earlier. All existing clusters must migrate prior to upgrading to v1.30 in order to stay functional.



Starting with Kubernetes 1.29, in-tree cloud providers have been disabled. You must set `DisableCloudProviders` and `DisableKubeletCloudCredentialProviders` feature gates to `false` to use the in-tree Azure cloud provider. You can do this by setting `feature-gates=DisableCloudProviders=false,DisableKubeletCloudCredentialProviders=false` as an additional argument for the cluster's Kubelet, Controller Manager, and API Server in the advanced cluster configuration. Setting `DisableKubeletCloudCredentialProviders=false` in the Kubelet, Controller Manager, and API Server arguments enables in-tree functionality for authenticating to Azure container registries for image pull credentials. See the Kubernetes 1.29: Mandala release notes [https://kubernetes.io/blog/2023/12/13/kubernetes-v1-29-release/#in-tree-cloud-provider-integration-removal] and this blog post [https://kubernetes.io/blog/2023/12/14/cloud-provider-integration-changes/#continue-using-in-tree-provider-code] for more details.

Starting with Kubernetes version 1.26, in-tree persistent volume types `kubernetes.io/azure-disk` and `kubernetes.io/azure-file` are deprecated and will no longer be supported. For new clusters, install the CSI drivers, or migrate to the corresponding CSI drivers `disk.csi.azure.com` and `file.csi.azure.com` by following the upstream migration documentation [https://learn.microsoft.com/en-us/azure/aks/csi-migrate-in-tree-volumes].

When using the `Azure` cloud provider, you can leverage the following capabilities:

- **Load Balancers:** Launches an Azure Load Balancer within a specific Network Security Group.
- **Persistent Volumes:** Supports using Azure Blob disks and Azure Managed Disks with standard and premium storage accounts.
- **Network Storage:** Support Azure Files via CIFS mounts.

The following account types are not supported for Azure Subscriptions:

- Single tenant accounts (i.e. accounts with no subscriptions).
- Multi-subscription accounts.

#### Prerequisites for RKE and SUSE® Rancher Prime: RKE2

To set up the Azure cloud provider for both RKE and RKE2, the following credentials need to be configured:

1. Set up the Azure Tenant ID
2. Set up the Azure Client ID and Azure Client Secret

3. Configure App Registration Permissions
4. Set up Azure Network Security Group Name

#### 1. Set up the Azure Tenant ID

Visit the Azure portal [https://portal.azure.com], sign in, and go to **Microsoft Entra ID** (formerly *Azure Active Directory*). On the **Overview** page (or under **Properties**), copy the value listed as **Tenant ID**.

If you want to use the Azure CLI, you can run the command `az account show` to get the information.

#### 2. Set up the Azure Client ID and Azure Client Secret

Visit the Azure portal [https://portal.azure.com], sign in, and follow the steps below to create an **App Registration** and obtain the corresponding **Azure Client ID** (`aadClientId`) and **Azure Client Secret** (`aadClientSecret`).

##### Register the Application

1. Search for and select **Microsoft Entra ID** (formerly *Azure Active Directory*).
2. In the left navigation menu, select **App registrations**.
3. Select **+ New registration**.
4. Enter a **Name** for your application.
5. Under **Supported account types**, choose who can use the application (the default, *Accounts in this organizational directory only*, is typical for internal integrations).
6. Select **Register**.

Once registered, you will be directed to the application's **Overview** page. Copy the value listed under **Application (client) ID, this is your *Azure Client ID** (`aadClientId`).

##### Generate the Azure Client Secret

1. In the left navigation menu under **Manage**, select **Certificates & secrets**.
2. Select the **Client secrets** tab and click **+ New client secret**.
3. Enter a **Description**, select an expiration period, and click **Add**.
4. Copy the string in the **Value** column—this is your **Azure Client Secret** (`aadClientSecret`).



Copy the secret **Value** immediately after creation. It will only be displayed once and will be permanently masked (`••••••••••••••••••`) after you navigate away or refresh the page. Do not copy the **Secret ID**.

#### 3. Configure App Registration Permissions

The last thing you will need to do, is assign the appropriate permissions to your App registration.

1. Go to **More services**, search for **Subscriptions** and open it.
2. Open **Access control (IAM)**.
3. Select **Add**.
4. For **Role**, select `Contributor`.

5. For **Select**, select your created App registration name.
6. Select **Save**.

#### 4. Set up Azure Network Security Group Name

A custom Azure Network Security Group (securityGroupName) is needed to allow Azure Load Balancers to work.

If you provision hosts using Rancher Machine Azure driver, you will need to edit them manually to assign them to this Network Security Group.

You should already assign custom hosts to this Network Security Group during provisioning.

Only hosts expected to be load balancer back ends need to be in this group.

### SUSE® Rancher Prime: RKE2 Cluster Set-up in Rancher



This section is valid only for creating clusters with the in-tree cloud provider.

1. Choose "Azure" from the Cloud Provider drop-down in the Cluster Configuration section.
2. Supply the Cloud Provider Configuration. Note that Rancher automatically creates a new Network Security Group, Resource Group, Availability Set, Subnet, and Virtual Network. If you already have some or all of these created, you must specify them before creating the cluster.

   - Click **Show Advanced** to view or edit these automatically generated names. Your Cloud Provider Configuration **must** match the fields in the **Machine Pools** section. If you have multiple pools, they must all use the same Resource Group, Availability Set, Subnet, Virtual Network, and Network Security Group.
   - An example is provided below. Modify it as needed.

   *Example Cloud Provider Config*

```
{
    "cloud":"AzurePublicCloud",
    "tenantId": "YOUR TENANTID HERE",
    "aadClientId": "YOUR AADCLIENTID HERE",
    "aadClientSecret": "YOUR AADCLIENTSECRET HERE",
    "subscriptionId": "YOUR SUBSCRIPTIONID HERE",
    "resourceGroup": "docker-machine",
    "location": "westus",
    "subnetName": "docker-machine",
    "securityGroupName": "rancher-managed-KA4jV9V2",
    "securityGroupResourceGroup": "docker-machine",
    "vnetName": "docker-machine-vnet",
    "vnetResourceGroup": "docker-machine",
    "primaryAvailabilitySetName": "docker-machine",
    "routeTableResourceGroup": "docker-machine",
    "cloudProviderBackoff": false,
    "useManagedIdentityExtension": false,
```

```
    "useInstanceMetadata": true
}
```

3. Under the menu:Cluster Configuration[Advanced] section, click **Add** under **Additional Controller Manager Args** and add this flag: `--configure-cloud-routes=false`
4. Click **Create** to submit the form and create the cluster.

#### Cloud Provider Configuration

Rancher automatically creates a new Network Security Group, Resource Group, Availability Set, Subnet, and Virtual Network. If you already have some or all of these created, you will need to specify them before creating the cluster. You can check **RKE1 Node Templates** or **RKE2 Machine Pools** to view or edit these automatically generated names.

**Refer to the full list of configuration options in the upstream docs [https://cloud-provider-azure.sigs.k8s.io/install/configs/].**



1. `useInstanceMetadata` must be set to `true` for the cloud provider to correctly configure `providerID`.
2. `excludeMasterFromStandardLB` must be set to `false` if you need to add nodes labeled `node-role.kubernetes.io/master` to the backend of the Azure Load Balancer (ALB).
3. `loadBalancerSku` can be set to `basic` or `standard`. Basic SKU will be deprecated in September 2025. Refer to the Azure upstream docs [https://learn.microsoft.com/en-us/azure/virtual-network/ip-services/public-ip-basic-upgrade-guidance#basic-sku-vs-standard-sku] for more information.

Azure supports reading the cloud config from Kubernetes secrets. The secret is a serialized version of the azure.json file. When the secret is changed, the cloud controller manager reconstructs itself without restarting the pod. It is recommended for the Helm chart to read the Cloud Provider Config from the secret.

Note that the chart reads the Cloud Provider Config from a given secret name in the `kube-system` namespace. Since Azure reads Kubernetes secrets, RBAC also needs to be configured. An example secret for the Cloud Provider Config is shown below. Modify it as needed and create the secret.

```
# azure-cloud-config.yaml
apiVersion: v1
kind: Secret
metadata:
  name: azure-cloud-config
  namespace: kube-system
type: Opaque
stringData:
  cloud-config: |-
    {
      "cloud": "AzurePublicCloud",
      "tenantId": "<tenant-id>",
```

```
    "subscriptionId": "<subscription-id>",
    "aadClientId": "<client-id>",
    "aadClientSecret": "<client-secret>",
    "resourceGroup": "docker-machine",
    "location": "westus",
    "subnetName": "docker-machine",
    "securityGroupName": "rancher-managed-kqmtsjgJ",
    "securityGroupResourceGroup": "docker-machine",
    "vnetName": "docker-machine-vnet",
    "vnetResourceGroup": "docker-machine",
    "primaryAvailabilitySetName": "docker-machine",
    "routeTableResourceGroup": "docker-machine",
    "cloudProviderBackoff": false,
    "useManagedIdentityExtension": false,
    "useInstanceMetadata": true,
    "loadBalancerSku": "standard",
    "excludeMasterFromStandardLB": false,
}
```

**Using the Out-of-tree Azure Cloud Provider**

**RKE2**

1. Select **External** from the **Cloud Provider** drop-down in the **Cluster Configuration** section.
2. Under menu:Cluster Configuration[Advanced], click **Add** under **Additional Controller Manager Args** and add this flag: `--configure-cloud-routes=false`.
3. Prepare the Cloud Provider Configuration to set it in the next step. Note that Rancher automatically creates a new Network Security Group, Resource Group, Availability Set, Subnet, and Virtual Network. If you already have some or all of these created, you must specify them before creating the cluster.

   Click **Show Advanced** to view or edit these automatically generated names. Your Cloud Provider Configuration **must** match the fields in the **Machine Pools** section. If you have multiple pools, they must all use the same Resource Group, Availability Set, Subnet, Virtual Network, and Network Security Group.

4. Under **Cluster Configuration > Add-on Config**, add the cloud controller manager manifest shown below into **Additional Manifest**.

   Note that this chart reads the Cloud Provider Config from the secret in the `kube-system` namespace. An example secret for the Cloud Provider Config is shown below; modify it as needed. Refer to the full list of configuration options in the upstream docs [https://cloud-provider-azure.sigs.k8s.io/install/configs/].

   Alternatively, you can also install the cloud controller manager using the Helm CLI.

   ```
   apiVersion: helm.cattle.io/v1
   ```

```
kind: HelmChart
metadata:
  name: azure-cloud-controller-manager
  namespace: kube-system
spec:
  chart: cloud-provider-azure
  repo: https://raw.githubusercontent.com/kubernetes-
sigs/cloud-provider-azure/master/helm/repo
  targetNamespace: kube-system
  bootstrap: true
  valuesContent: |-
    infra:
      clusterName: <cluster-name>
    cloudControllerManager:
      cloudConfigSecretName: azure-cloud-config
      cloudConfig: null
      clusterCIDR: null
      enableDynamicReloading: 'true'
      nodeSelector:
        node-role.kubernetes.io/control-plane: 'true'
      allocateNodeCidrs: 'false'
      hostNetworking: true
      caCertDir: /etc/ssl
      configureCloudRoutes: 'false'
      enabled: true
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/master
        - effect: NoSchedule
          key: node-role.kubernetes.io/control-plane
          value: 'true'
        - effect: NoSchedule
          key: node.cloudprovider.kubernetes.io/uninitialized
          value: 'true'
---
apiVersion: v1
kind: Secret
metadata:
  name: azure-cloud-config
  namespace: kube-system
type: Opaque
stringData:
  cloud-config: |-
    {
      "cloud": "AzurePublicCloud",
```

```
      "tenantId": "<tenant-id>",
      "subscriptionId": "<subscription-id>",
      "aadClientId": "<client-id>",
      "aadClientSecret": "<client-secret>",
      "resourceGroup": "docker-machine",
      "location": "westus",
      "subnetName": "docker-machine",
      "securityGroupName": "rancher-managed-kqmtsjgJ",
      "securityGroupResourceGroup": "docker-machine",
      "vnetName": "docker-machine-vnet",
      "vnetResourceGroup": "docker-machine",
      "primaryAvailabilitySetName": "docker-machine",
      "routeTableResourceGroup": "docker-machine",
      "cloudProviderBackoff": false,
      "useManagedIdentityExtension": false,
      "useInstanceMetadata": true,
      "loadBalancerSku": "standard",
      "excludeMasterFromStandardLB": false,
    }
```

5. Click **Create** to submit the form and create the cluster.

**RKE1**

1. Choose **External** from the **Cloud Provider** drop-down in the **Cluster Options** section. This sets `--cloud-provider=external` for Kubernetes components.
2. Install the `cloud-provider-azure` chart after the cluster finishes provisioning. Note that the cluster is not successfully provisioned and nodes are still in an `uninitialized` state until you deploy the cloud controller manager. This can be done manually using CLI, or via Helm charts in UI.

Refer to the official Azure upstream documentation [https://cloud-provider-azure.sigs.k8s.io/install/azure-ccm/] for more details on deploying the Cloud Controller Manager.

#### Helm Chart Installation from CLI

Official upstream docs for Helm chart installation [https://github.com/kubernetes-sigs/cloud-provider-azure/tree/master/helm/cloud-provider-azure] can be found on Github.

1. Create a `azure-cloud-config` secret with the required cloud provider config.

```
kubectl apply -f azure-cloud-config.yaml
```

2. Add the Helm repository:

```
helm repo add azure-cloud-controller-manager
```

```
https://raw.githubusercontent.com/kubernetes-sigs/cloud-provider-azure/master/helm/repo
helm repo update
```

3. Create a `values.yaml` file with the following contents to override the default `values.yaml`:

**RKE2**

```yaml
# values.yaml
infra:
  clusterName: <cluster-name>
cloudControllerManager:
  cloudConfigSecretName: azure-cloud-config
  cloudConfig: null
  clusterCIDR: null
  enableDynamicReloading: 'true'
  configureCloudRoutes: 'false'
  allocateNodeCidrs: 'false'
  caCertDir: /etc/ssl
  enabled: true
  replicas: 1
  hostNetworking: true
  nodeSelector:
    node-role.kubernetes.io/control-plane: 'true'
  tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/master
    - effect: NoSchedule
      key: node-role.kubernetes.io/control-plane
      value: 'true'
    - effect: NoSchedule
      key: node.cloudprovider.kubernetes.io/uninitialized
      value: 'true'
```

**RKE**

```yaml
# values.yaml
cloudControllerManager:
  cloudConfigSecretName: azure-cloud-config
  cloudConfig: null
  clusterCIDR: null
  enableDynamicReloading: 'true'
  configureCloudRoutes: 'false'
  allocateNodeCidrs: 'false'
  caCertDir: /etc/ssl
```

```
    enabled: true
    replicas: 1
    hostNetworking: true
    nodeSelector:
      node-role.kubernetes.io/controlplane: 'true'
      node-role.kubernetes.io/control-plane: null
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/controlplane
        value: 'true'
      - effect: NoSchedule
        key: node.cloudprovider.kubernetes.io/uninitialized
        value: 'true'
  infra:
    clusterName: <cluster-name>
```

4. Install the Helm chart:

```
helm upgrade --install cloud-provider-azure azure-cloud-controller-manager/cloud-provider-azure -n kube-system --values values.yaml
```

Verify that the Helm chart installed successfully:

```
helm status cloud-provider-azure -n kube-system
```

5. (Optional) Verify that the cloud controller manager update succeeded:

```
kubectl rollout status deployment -n kube-system cloud-controller-manager
kubectl rollout status daemonset -n kube-system cloud-node-manager
```

6. The cloud provider is responsible for setting the ProviderID of the node. Check if all nodes are initialized with the ProviderID:

```
kubectl describe nodes | grep "ProviderID"
```

#### Helm Chart Installation from UI

1. Click ☰, then select the name of the cluster from the left navigation.
2. Select **Apps** > **Repositories**.
3. Click the **Create** button.

4. Enter `https://raw.githubusercontent.com/kubernetes-sigs/cloud-provider-azure/master/helm/repo` in the **Index URL** field.
5. Select **Apps** > **Charts** from the left navigation and install **cloud-provider-azure** chart.
6. Select the namespace, `kube-system`, and enable **Customize Helm options before install**.
7. Replace `cloudConfig: /etc/kubernetes/azure.json` to read from the Cloud Config Secret and enable dynamic reloading:

```
cloudConfigSecretName: azure-cloud-config
enableDynamicReloading: 'true'
```

8. Update the following fields as required:

```
allocateNodeCidrs: 'false'
configureCloudRoutes: 'false'
clusterCIDR: null
```

**RKE2**

9. Rancher-provisioned RKE2 nodes have the selector `node-role.kubernetes.io/control-plane` set to `true`. Update the nodeSelector:

```
nodeSelector:
  node-role.kubernetes.io/control-plane: 'true'
```

**RKE**

9. Rancher-provisioned RKE nodes are tainted `node-role.kubernetes.io/controlplane`. Update tolerations and the nodeSelector:

```
tolerations:
  - effect: NoSchedule
    key: node.cloudprovider.kubernetes.io/uninitialized
    value: 'true'
  - effect: NoSchedule
    value: 'true'
    key: node-role.kubernetes.io/controlplane
```

```
nodeSelector:
  node-role.kubernetes.io/controlplane: 'true'
```

10. Install the chart and confirm that the cloud controller and cloud node manager deployed successfully:

```
kubectl rollout status deployment -n kube-system cloud-controller-
manager
kubectl rollout status daemonset -n kube-system cloud-node-manager
```

11. The cloud provider is responsible for setting the ProviderID of the node. Check if all nodes are initialized with the ProviderID:

```
kubectl describe nodes | grep "ProviderID"
```

#### Installing CSI Drivers

Install Azure Disk CSI driver [https://github.com/kubernetes-sigs/azuredisk-csi-driver] or Azure File CSI Driver [https://github.com/kubernetes-sigs/azurefile-csi-driver] to access Azure Disk [https://azure.microsoft.com/en-us/services/storage/disks/] or Azure File [https://azure.microsoft.com/en-us/services/storage/disks/] volumes respectively.

The steps to install the Azure Disk CSI driver are shown below. You can install the Azure File CSI Driver in a similar manner by following the helm installation documentation [https://github.com/kubernetes-sigs/azurefile-csi-driver/blob/master/charts/README.md].



Clusters must be provisioned using `Managed Disk` to use Azure Disk. You can configure this when creating **RKE1 Node Templates** or **RKE2 Machine Pools*.

Official upstream docs for Helm chart installation [https://github.com/kubernetes-sigs/azuredisk-csi-driver/blob/master/charts/README.md] can be found on Github.

1. Add and update the helm repository:

```
helm repo add azuredisk-csi-driver
https://raw.githubusercontent.com/kubernetes-sigs/azuredisk-csi-
driver/master/charts
helm repo update azuredisk-csi-driver
```

2. Install the chart as shown below, updating the --version argument as needed. Refer to the full list of latest chart configurations in the upstream docs [https://github.com/kubernetes-sigs/azuredisk-csi-driver/blob/master/charts/README.md#latest-chart-configuration].

```
helm install azuredisk-csi-driver azuredisk-csi-driver/azuredisk-csi-
driver --namespace kube-system --version v1.30.1 --set
controller.cloudConfigSecretName=azure-cloud-config --set
controller.cloudConfigSecretNamespace=kube-system --set
controller.runOnControlPlane=true
```

3. (Optional) Verify that the azuredisk-csi-driver installation succeeded:

```
kubectl --namespace=kube-system get pods --selector
="app.kubernetes.io/name=azuredisk-csi-driver" --watch
```

4. Provision an example Storage Class:

```
cat <<EOF | kubectl create -f -
kind: StorageClass
apiVersion: storage.k8s.io/vl
metadata:
  name: standard
provisioner: kubernetes.io/azure-disk
parameters:
  storageaccounttype: Standard_LRS
  kind: Managed
EOF
```

Verify that the storage class has been provisioned:

```
kubectl get storageclasses
```

5. Create a PersistentVolumeClaim:

```
cat <<EOF | kubectl create -f -
kind: PersistentVolumeClaim
apiVersion: vl
metadata:
  name: azure-disk-pvc
spec:
  storageClassName: standard
  accessModes:
 ** ReadWriteOnce
  resources:
 requests:
storage: 5Gi
EOF
```

Verify that the PersistentVolumeClaim and PersistentVolume have been created:

```
kubectl get persistentvolumeclaim
kubectl get persistentvolume
```

6. Attach the new Azure Disk:

You can now mount the Kubernetes PersistentVolume into a Kubernetes Pod. The disk can be consumed by any Kubernetes object type, including a Deployment, DaemonSet, or StatefulSet. However, the following example simply mounts the PersistentVolume into a standalone Pod.

```
cat <<EOF | kubectl create -f -
kind: Pod
apiVersion: vl
metadata:
  name: mypod-dynamic-azuredisk
spec:
  containers:
    - name: mypod
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: storage
  volumes:
    - name: storage
      persistentVolumeClaim:
        claimName: azure-disk-pvc
EOF
```

### Setting Up an In-tree VMware vSphere Cloud Provider

To set up the in-tree VMware vSphere cloud provider, follow these steps while creating the vSphere cluster in Rancher:

1. Set **Cloud Provider** option to `Custom` or `Custom (In-Tree)`.
2. Click on **Edit as YAML**
3. Insert the following structure to the pre-populated cluster YAML. This structure must be placed under `rancher_kubernetes_engine_config`. Note that the `name` *must* be set to `vsphere`.

```
rancher_kubernetes_engine_config:
  cloud_provider:
      name: vsphere
      vsphereCloudProvider:
          [Insert provider configuration]
```

Rancher uses RKE (the Rancher Kubernetes Engine) to provision Kubernetes clusters. Refer to the vSphere configuration reference in the RKE documentation [https://rancher.com/docs/rke/latest/en/config-options/cloud-providers/vsphere/config-reference/] for details about the properties of the `vsphereCloudProvider` directive.

## Setting Up an Out-of-tree VMware vSphere Cloud Provider

Kubernetes is moving away from maintaining cloud providers in-tree. VMware vSphere has an out-of-tree cloud provider that can be used by installing the vSphere cloud provider and cloud storage plugins.

This page covers how to install the Cloud Provider Interface (CPI) and Cloud Storage Interface (CSI) plugins after setting up a cluster.

### Prerequisites

The vSphere versions supported:

- 6.7u3
- 7.0u1 or higher.

The Kubernetes version must be 1.19 or higher.

Using the vSphere out-of-tree cloud provider requires Linux nodes and is not supported on Windows.

### Installation

The Cloud Provider Interface (CPI) should be installed first before installing the Cloud Storage Interface (CSI).

#### 1. Create a VMware vSphere cluster

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **VMware vSphere** or **Custom**.
4. On the **Basics** tab in the **Cluster Configuration** section, set the **Cloud Provider** to **vSphere**.
5. In the **Add-On Config** tab, the vSphere Cloud Provider (CPI) and Storage Provider (CSI) options.
6. Finish creating your cluster.

#### 2. Install the CPI plugin

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where the vSphere CPI plugin will be installed and click **Explore**.
3. Click **Apps › Charts**.
4. Click **vSphere CPI**.
5. Fill out the required vCenter details.
6. vSphere CPI initializes all nodes with ProviderID which is needed by the vSphere CSI driver. Check if all nodes are initialized with the ProviderID before installing CSI driver with the following command:

```
kubectl describe nodes | grep "ProviderID"
```

#### 3. Installing the CSI plugin

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where the vSphere CSI plugin will be installed and click **Explore**.
3. Click **Apps › Charts**.
4. Click **vSphere CSI**.
5. Click **Install**.
6. Fill out the required vCenter details. On the **Features** tab, set **Enable CSI Migration** to **false**.
7. On the **Storage** tab, fill out the details for the StorageClass. This chart creates a StorageClass with the `csi.vsphere.vmware.com` as the provisioner.
8. Click **Install**.

#### Using the CSI driver for provisioning volumes

The CSI chart by default creates a storageClass.

If that option was not selected while launching the chart, create a storageClass with the `csi.vsphere.vmware.com` as the provisioner.

All volumes provisioned using this StorageClass will get provisioned by the CSI driver.

## Setting up the Google Compute Engine Cloud Provider

In this section, you'll learn how to enable the Google Compute Engine (GCE) cloud provider for custom clusters in Rancher. A custom cluster is one in which Rancher installs Kubernetes on existing nodes.

The official Kubernetes documentation for the GCE cloud provider is here. [https://github.com/kubernetes/website/blob/release-1.18/content/en/docs/concepts/cluster-administration/cloud-providers.md#gce]



*Prerequisites:*

The service account of `Identity and API` access on GCE needs the `Computer Admin` permission.

If you are using Calico,

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the custom cluster and click *☰ > *Edit YAML*. Enter the following configuration:

```
rancher_kubernetes_engine_config:
  cloud_provider:
    name: gce
    customCloudProvider: |-
      [Global]
      project-id=<your project ID, optional>
      network-name=<your network, optional if using default network>
      subnetwork-name=<your subnetwork of the above network,
```

```
optional if using default network>
       node-instance-prefix=<your instance group name/your instance
name specific prefix, required>
       node-tags=<your network tags, must patch one or some tags,
required>
   network:
     options:
       calico_cloud_provider: "gce"
     plugin: "calico"
```

If you are using Canal or Flannel,

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the custom cluster and click *⋮ > *Edit YAML*. Enter the following configuration:

```
 rancher_kubernetes_engine_config:
   cloud_provider:
     name: gce
     customCloudProvider: |-
       [Global]
       project-id=<your project ID, optional>
       network-name=<your network, optional if using default network>
       subnetwork-name=<your subnetwork of the above network,
optional if using default network>
       node-instance-prefix=<your instance group name/your instance
name specific prefix, required>
       node-tags=<your network tags, must patch one or some tags,
required>
   services:
     kube_controller:
       extra_args:
         configure-cloud-routes: true # we need to allow the cloud
provider configure the routes for the hosts
```

### Migrating Amazon In-tree to Out-of-tree

Kubernetes is moving away from maintaining cloud providers in-tree. In Kubernetes v1.27 and later, the in-tree cloud providers have been removed. The Rancher UI allows you to upgrade to Kubernetes v1.27 when you migrate from an in-tree to out-of-tree provider.

However, if you're performing a manual migration, existing clusters must upgrade to Kubernetes v1.27 after you migrate in order to remain functional.

To migrate from the in-tree cloud provider to the out-of-tree AWS cloud provider, you must stop the existing cluster's kube controller manager and install the AWS cloud controller manager. There are many ways to do

this. Refer to the official AWS documentation on the external cloud controller manager [https://cloud-provider-aws.sigs.k8s.io/getting_started/] for details.

If it's acceptable to have some downtime during migration, follow the instructions to set up an external cloud provider. These instructions outline how to configure the out-of-tree cloud provider for a newly provisioned cluster. During set up, there will be some downtime, as there is a time gap between when the old cloud provider stops running and when the new cloud provider starts to run.

If your setup can't tolerate any control plane downtime, you must enable leader migration. This facilitates a smooth transition from the controllers in the kube controller manager to their counterparts in the cloud controller manager. Refer to the official AWS documentation on Using leader migration [https://cloud-provider-aws.sigs.k8s.io/getting_started/] for more details.



The Kubernetes cloud controller migration documentation [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#before-you-begin] states that it's possible to migrate with the same Kubernetes version, but assumes that the migration is part of a Kubernetes upgrade. Refer to the Kubernetes documentation on migrating to use the cloud controller manager [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/] to see if you need to customize your setup before migrating. Confirm your migration configuration values [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#default-configuration]. If your cloud provider provides an implementation of the Node IPAM controller, you also need to migrate the IPAM controller [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#node-ipam-controller-migration].

**RKE2**

1. Update the cluster config to enable leader migration:

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          kube-controller-manager-arg:
            - enable-leader-migration
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/control-plane-role
              operator: In
              values:
                - 'true'
```

Note that the cloud provider is still `aws` at this step:

```
spec:
  rkeConfig:
    machineGlobalConfig:
```

```
      cloud-provider-name: aws
```

2. Cordon control plane nodes so that AWS cloud controller pods run on nodes only after upgrading to the external cloud provider:

```
kubectl cordon -l "node-role.kubernetes.io/control-plane=true"
```

3. To install the AWS cloud controller manager with leader migration enabled, follow Steps 1-3 for deploying the cloud controller manager chart. From Kubernetes 1.22 onwards, the kube-controller-manager will utilize a default configuration which will satisfy the controller-to-manager migration. Update container args of the `aws-cloud-controller-manager` under `spec.rkeConfig.additionalManifest` to enable leader migration:

```
- '--enable-leader-migration=true'
```

4. Install the chart and confirm that the Daemonset `aws-cloud-controller-manager` successfully deployed:

```
kubectl rollout status daemonset -n kube-system aws-cloud-
controller-manager
```

5. Update the provisioning cluster to change the cloud provider and remove leader migration args from the kube controller. If upgrading the Kubernetes version, set the Kubernetes version as well in the `spec.kubernetesVersion` section of the cluster YAML file



Only remove `cloud-provider-name: aws` if not relying on the rke2 supervisor to correctly set the providerID.

Remove `enable-leader-migration` if you don't want it enabled in your cluster:

```
spec:
  rkeConfig:
    machineGlobalConfig:
      cloud-provider-name: external
```

Remove `enable-leader-migration` from:

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          kube-controller-manager-arg:
            - enable-leader-migration
```

```
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/control-plane-role
              operator: In
              values:
                - 'true'
```



You can also disable leader migration after the upgrade, as leader migration is no longer required due to only one cloud-controller-manager and can be removed. Upgrade the chart and remove the following section from the container arguments:

```
- --enable-leader-migration=true
```

Verify the cloud controller manager update was successfully rolled out with the following command:

```
kubectl rollout status daemonset -n kube-system aws-cloud-
controller-manager
```

6. The cloud provider is responsible for setting the ProviderID of the node. Check if all nodes are initialized with the ProviderID:

```
kubectl describe nodes | grep "ProviderID"
```

**RKE**

1. Update the cluster config to enable leader migration in `cluster.yml`:

```
services:
  kube-controller:
    extra_args:
      enable-leader-migration: "true"
```

Note that the cloud provider is still `aws` at this step:

```
cloud_provider:
  name: aws
```

2. Cordon the control plane nodes, so that AWS cloud controller pods run on nodes only after upgrading to the external cloud provider:

```
kubectl cordon -l "node-role.kubernetes.io/controlplane=true"
```

3. To install the AWS cloud controller manager, you must enable leader migration and follow the same steps as when installing AWS on a new cluster. To enable leader migration, add the following to the container arguments in step 7 while following the steps to install the chart:

   ```
   - '--enable-leader-migration=true'
   ```

4. Confirm that the chart is installed but that the new pods aren't running yet due to cordoned controlplane nodes. After updating the cluster in the next step, RKE will upgrade and uncordon each node, and schedule `aws-controller-manager` pods.

5. Update `cluster.yml` to change the cloud provider and remove the leader migration arguments from the kube-controller.

   Selecting **External Amazon (out-of-tree)** sets `--cloud-provider=external` and lets you enable `useInstanceMetadataHostname`. You must enable `useInstanceMetadataHostname` for node-driver clusters and for custom clusters if not you don't provide a custom node name via `--node-name`. Enabling `useInstanceMetadataHostname` will query ec2 metadata service and set `/hostname` as `hostname-override` for `kubelet` and `kube-proxy`:

   ```
   rancher_kubernetes_engine_config:
     cloud_provider:
       name: external-aws
       useInstanceMetadataHostname: true/false
   ```

   Remove `enable-leader-migration` if you don't want it enabled in your cluster:

   ```
     services:
       kube-controller:
         extra_args:
           enable-leader-migration: "true"
   ```

   You can also disable leader migration after you finish the migration. Upgrade the chart and remove the following section from the container arguments:

   ```
   - --enable-leader-migration=true
   ```

6. If you're upgrading the cluster's Kubernetes version, set the Kubernetes version as well.

7. Update the cluster. The `aws-cloud-controller-manager` pods should now be running.

### Migrating Azure In-tree to Out-of-tree

Kubernetes is moving away from maintaining cloud providers in-tree.

Starting with Kubernetes 1.29, in-tree cloud providers have been disabled. You must disable `DisableCloudProviders` and `DisableKubeletCloudCredentialProvider` to use the in-tree Azure cloud provider or migrate from in-tree cloud provider to out-of-tree provider. You can disable the required feature gates by setting `feature-gates=DisableCloudProviders=false` as an additional argument for the cluster's Kubelet, Controller Manager, and API Server in the advanced cluster configuration. Additionally, set `DisableKubeletCloudCredentialProvider=false` in the Kubelet's arguments to enable in-tree functionality for authenticating to Azure container registries for image pull credentials. See upstream docs [https://github.com/kubernetes/kubernetes/pull/117503] for more details.

In Kubernetes v1.30 and later, the in-tree cloud providers have been removed. Rancher allows you to upgrade to Kubernetes v1.30 when you migrate from an in-tree to out-of-tree provider.

To migrate from the in-tree cloud provider to the out-of-tree Azure cloud provider, you must stop the existing cluster's kube controller manager and install the Azure cloud controller manager.

If it's acceptable to have some downtime during migration, follow the instructions to set up an external cloud provider. These instructions outline how to configure the out-of-tree cloud provider for a newly provisioned cluster. During set up, there will be some downtime, as there is a time gap between when the old cloud provider stops running and when the new cloud provider starts to run.

If your setup can't tolerate any control plane downtime, you must enable leader migration. This facilitates a smooth transition from the controllers in the kube controller manager to their counterparts in the cloud controller manager.



The Kubernetes cloud controller migration documentation [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#before-you-begin] states that it's possible to migrate with the same Kubernetes version, but assumes that the migration is part of a Kubernetes upgrade. Refer to the Kubernetes documentation on migrating to use the cloud controller manager [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/] to see if you need to customize your setup before migrating. Confirm your migration configuration values [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#default-configuration]. If your cloud provider provides an implementation of the Node IPAM controller, you also need to migrate the IPAM controller [https://kubernetes.io/docs/tasks/administer-cluster/controller-manager-leader-migration/#node-ipam-controller-migration].

Starting with Kubernetes v1.26, in-tree persistent volume types `kubernetes.io/azure-disk` and `kubernetes.io/azure-file` are deprecated and no longer supported. There are no plans to remove these drivers following their deprecation, however you should migrate to the corresponding CSI drivers, `disk.csi.azure.com` and `file.csi.azure.com`. To review the migration options for your storage classes and upgrade your cluster to use Azure Disks and Azure Files CSI drivers, see Migrate from in-tree to CSI drivers [https://learn.microsoft.com/en-us/azure/aks/csi-migrate-in-tree-volumes].

**RKE2**

1. Update the cluster config to enable leader migration:

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          kube-controller-manager-arg:
            - enable-leader-migration
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/control-plane-role
              operator: In
              values:
                - 'true'
```

Note that the cloud provider is still `azure` at this step:

```
spec:
  rkeConfig:
    machineGlobalConfig:
      cloud-provider-name: azure
```

2. Cordon control plane nodes so that Azure cloud controller pods run on nodes only after upgrading to the external cloud provider:

```
kubectl cordon -l "node-role.kubernetes.io/control-plane=true"
```

3. To deploy the Azure cloud controller manager, use any of the available options:

   - UI: Follow steps 1-10 of Helm chart installation from UI to install the cloud controller manager chart.
   - CLI: Follow steps 1-4 of Helm chart installation from CLI.
   - Update the cluster's additional manifest: Follow steps 2-3 to install the cloud controller manager chart.

   Confirm that the chart is installed but that the new pods aren't running yet due to cordoned controlplane nodes.

4. To enable leader migration, add `--enable-leader-migration` to the container arguments of `cloud-controller-manager`:

```
kubectl -n kube-system patch deployment cloud-controller-
manager \
--type=json \
-p='[{"op": "add", "path":
"/spec/template/spec/containers/0/args/-", "value": "--enable-
```

```
leader-migration"}]'
```

5. Update the provisioning cluster to change the cloud provider and remove leader migration args from the kube controller manager. If upgrading the Kubernetes version, set the Kubernetes version as well in the `spec.kubernetesVersion` section of the cluster YAML file.

```
spec:
  rkeConfig:
    machineGlobalConfig:
      cloud-provider-name: external
```

   Remove `enable-leader-migration` from the kube controller manager:

```
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          kube-controller-manager-arg:
            - enable-leader-migration
        machineLabelSelector:
          matchExpressions:
            - key: rke.cattle.io/control-plane-role
              operator: In
              values:
                - 'true'
```

6. Uncordon control plane nodes so that Azure cloud controller pods now run on nodes:

```
kubectl uncordon -l "node-role.kubernetes.io/control-
plane=true"
```

7. Update the cluster. The `cloud-controller-manager` pods should now be running.

```
kubectl rollout status deployment -n kube-system cloud-
controller-manager
kubectl rollout status daemonset -n kube-system cloud-node-
manager
```

8. The cloud provider is responsible for setting the ProviderID of the node. Check if all nodes are initialized with the ProviderID:

```
kubectl describe nodes | grep "ProviderID"
```

9. (Optional) You can also disable leader migration after the upgrade, as leader migration is not required with only one cloud-controller-manager. Update the `cloud-controller-manager` deployment to remove leader migration from the container arguments:

```
- --enable-leader-migration=true
```

**RKE**

1. Update the cluster config to enable leader migration in `cluster.yml`:

```
services:
  kube-controller:
    extra_args:
      enable-leader-migration: "true"
```

   Note that the cloud provider is still `azure` at this step:

```
cloud_provider:
  name: azure
```

2. Cordon the control plane nodes, so that Azure cloud controller pods run on nodes only after upgrading to the external cloud provider:

```
kubectl cordon -l "node-role.kubernetes.io/controlplane=true"
```

3. To install the Azure cloud controller manager, follow the same steps as when installing Azure cloud provider on a new cluster:
   - UI: Follow steps 1-10 of Helm chart installation from UI to install the cloud controller manager chart.
   - CLI: Follow steps 1-4 of Helm chart installation from CLI to install the cloud controller manager chart.
4. Confirm that the chart is installed but that the new pods aren't running yet due to cordoned controlplane nodes. After updating the cluster in the next step, RKE will upgrade and uncordon each node, and schedule `cloud-controller-manager` pods.
5. To enable leader migration, add `--enable-leader-migration` to the container arguments of `cloud-controller-manager`:

```
kubectl -n kube-system patch deployment cloud-controller-
manager \
--type=json \
-p='[{"op": "add", "path":
"/spec/template/spec/containers/0/args/-", "value": "--enable-
leader-migration"}]'
```

6. Update `cluster.yml` to change the cloud provider to `external` and remove the leader migration arguments from the kube-controller.

```
rancher_kubernetes_engine_config:
  cloud_provider:
    name: external
```

Remove `enable-leader-migration` if you don't want it enabled in your cluster:

```
services:
  kube-controller:
    extra_args:
      enable-leader-migration: "true"
```

7. If you're upgrading the cluster's Kubernetes version, set the Kubernetes version as well.
8. Update the cluster. The `cloud-controller-manager` pods should now be running.

```
kubectl rollout status deployment -n kube-system cloud-
controller-manager
kubectl rollout status daemonset -n kube-system cloud-node-
manager
```

9. The cloud provider is responsible for setting the ProviderID of the node. Verify that all nodes are initialized with the ProviderID:

```
kubectl describe nodes | grep "ProviderID"
```

10. (Optional) You can also disable leader migration after the upgrade, as leader migration is not required with only one cloud-controller-manager. Update the `cloud-controller-manager` deployment to remove leader migration from the container arguments:

```
- --enable-leader-migration=true
```

## Migrating VMware vSphere In-tree to Out-of-tree

Kubernetes is moving away from maintaining cloud providers in-tree. VMware vSphere has an out-of-tree cloud provider that can be used by installing the vSphere cloud provider and cloud storage plugins.

This page covers how to migrate from the in-tree vSphere cloud provider to out-of-tree, and manage the existing VMs post migration.

It follows the steps provided in the official vSphere migration documentation [https://vsphere-csi-driver.sigs.k8s.io/features/vsphere_csi_migration.html] and provides the steps to be performed in Rancher.

#### Cloud-config Format Limitation

Existing volumes that were provisioned using the following cloud-config format won't get migrated due to an existing bug in vSphere Cloud Storage Interface (CSI).

If the cloud-config has this format for datastore and resource pool path, vsphere CSI driver cannot recognize it:

```
default-datastore: </datacenter>/datastore/<default-datastore-name>
resourcepool-path: "</datacenter>/host/<cluster-
name>/Resources/<resource-pool-name>"
```

Volumes provisioned with the in-tree provider using the following format will get migrated correctly:

```
default-datastore: <default-datastore-name>
resourcepool-path: "<cluster-name>/Resources/<resource-pool-name>"
```

Upstream bug: https://github.com/kubernetes-sigs/vsphere-csi-driver/issues/628

Rancher issue tracking this bug: https://github.com/rancher/rancher/issues/31105

#### Prerequisites

- vSphere CSI Migration requires vSphere 7.0u1. In order to be able to manage existing in-tree vSphere volumes, upgrade vSphere to 7.0u1.
- The Kubernetes version must be 1.19 or higher.

#### Migration

##### 1. Install the CPI plugin

Before installing CPI, we need to taint all nodes with `node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule`.

This can be done by running the following commands:

```
curl -O https://raw.githubusercontent.com/rancher/helm3-
charts/56b622f519728378abeddfe95074f1b87ab73b1e/charts/vsphere-
cpi/taints.sh
```

Or:

```
wget https://raw.githubusercontent.com/rancher/helm3-
charts/56b622f519728378abeddfe95074f1b87ab73b1e/charts/vsphere-
cpi/taints.sh
chmod +x taints.sh
./taints.sh <path to kubeconfig if running the command outside the
```

```
cluster>
```

Once all nodes are tainted by the running the script, launch the Helm vSphere CPI chart.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where the vSphere CPI chart will be installed and click **Explore**.
3. Click **Apps › Charts**.
4. Click **vSphere CPI**.
5. Click **Install**.
6. Fill out the required vCenter details and click **Install**.

vSphere CPI initializes all nodes with ProviderID, which is needed by the vSphere CSI driver.

Check if all nodes are initialized with the ProviderID with the following command:

```
kubectl describe nodes | grep "ProviderID"
```

#### 2. Install the CSI driver

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where the vSphere CSI chart will be installed and click **Explore**.
3. Click **Apps › Charts**.
4. Click **vSphere CSI**.
5. Click **Install**.
6. Fill out the required vCenter details and click **Install**.
7. Check **Customize Helm options before install** and click **Next**.
8. On the **Features** tab, check **Enable CSI Migration**.
9. Optionally, go to the **Storage** tab and set up a datastore. This chart creates a StorageClass with the `csi.vsphere.vmware.com` as the provisioner. You can provide the URL of the datastore to be used for CSI volume provisioning while creating this StorageClass. The datastore URL can be found in the vSphere client by selecting the datastore and going to the Summary tab. Fill out the details for the StorageClass.
10. Click **Install**.

#### 3. Edit the cluster to enable CSI migration feature flags

1. While editing the cluster, if the Kubernetes version is less than 1.19, select Kubernetes version 1.19 or higher from the **Kubernetes Version** dropdown.
2. For enabling feature flags, click on "Edit as YAML", and add the following under kube-controller and kubelet:

    ```
    extra_args:
    ```

```
      feature-gates: "CSIMigration=true,CSIMigrationvSphere=true"
```

#### 4. Drain worker nodes

Worker nodes must be drained during the upgrade before changing the kubelet and kube-controller-manager args.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you will drain worker nodes and click ☰ > **Edit Config**.
3. In the **Advanced Options** section, set the field **Maximum Worker Nodes Unavailable** to `1`.
4. To drain the nodes during upgrade, select **Drain Nodes** › **Yes**.
5. Set **Force** and **Delete Local Data** to **true**.
6. Click **Save** to upgrade the cluster.

# Launching Kubernetes with SUSE Rancher Prime

Rancher allows you to launch a Kubernetes cluster on different types of nodes, including bare-metal servers, on-premise virtual machines, and virtual machines from an infrastructure provider. When deploying Kubernetes to these nodes, Rancher gives you the option to use RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html] or K3s [https://documentation.suse.com/cloudnative/k3s].

Rancher can install Kubernetes on existing nodes, or it can dynamically provision nodes in an infrastructure provider and install Kubernetes on them.

Rancher can also create pools of machines. One benefit of installing Kubernetes on machine pools hosted by an infrastructure provider is that if a node loses connectivity with the cluster, Rancher can automatically create another node to join the cluster to ensure that the count of the machine pool is as expected.

## SUSE® Rancher Prime: RKE2

Rancher v2.6 introduced provisioning for RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html] clusters directly from the Rancher UI. RKE2, also known as RKE Government, is a fully conformant Kubernetes distribution that focuses on security and compliance within the U.S. Federal Government sector. In Rancher v.2.6.4 and earlier, RKE2 provisioning was in tech preview.

Note that in Rancher v2.6.5, RKE2 provisioning became GA.

#### Requirements

If you use RKE2 to set up a cluster, your nodes must meet the requirements [https://documentation.suse.com/cloudnative/rke2/latest/en/install/requirements.html] for nodes in downstream user clusters.

#### Launching Kubernetes on New Nodes in an Infrastructure Provider

RKE2 provisioning is built on top of a new provisioning framework that leverages the upstream Cluster API (CAPI) [https://github.com/kubernetes-sigs/cluster-api] project. With this new provisioning framework, you can:

- Provision RKE2 clusters onto any provider for which Rancher has a node driver.
- Fully configure RKE2 clusters within Rancher.

- Choose CNI options Calico, Cilium, and Multus in addition to Canal.

When you make changes to your cluster configuration in RKE2, this may result in nodes reprovisioning. This is controlled by CAPI controllers and not by Rancher itself. Note that for etcd nodes, the same behavior does not apply.

An example configuration change that may cause the described behavior is when editing a cluster and enabling drain before delete. In this scenario, the existing control plane nodes and worker are deleted and new nodes are created.

RKE2 provisioning also includes installing RKE2 on clusters with Windows nodes.

Windows features for RKE2 include:

- Windows supports the vSphere node driver
- Calico and Flannel CNI for Windows RKE2 custom clusters
- Project Network Isolation (PNI) for Calico
- Windows Containers with RKE2 powered by containerd
- Provisioning of Windows RKE2 clusters through Terraform
- Provisioning of Windows RKE2 custom clusters directly from the Rancher UI

Windows Support for RKE2 Custom Clusters requires choosing Calico as the CNI.

#### Launching Kubernetes on Existing Custom Nodes

RKE2 provisioning also allows you to install custom clusters on previously provisioned VMs or bare-metal nodes.

If you want to reuse a node from a previous custom cluster, clean the node before using it in a cluster again. If you reuse a node that hasn't been cleaned, cluster provisioning may fail.

#### Programmatically Creating SUSE® Rancher Prime: RKE2 Clusters

The most common way to programmatically deploy RKE2 clusters through Rancher is by using the Rancher2 Terraform provider [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster_v2].

# SUSE Rancher Prime Agents

There are two different agent resources deployed on Rancher managed clusters:

- cattle-cluster-agent
- rancher-system-agent

For a conceptual overview of how the Rancher server provisions clusters and communicates with them, refer to the architecture.

## cattle-cluster-agent

The `cattle-cluster-agent` is used to connect to the Kubernetes API of Rancher Launched Kubernetes [] clusters. The `cattle-cluster-agent` is deployed using a Deployment resource.

### rancher-system-agent

The `rancher-system-agent` is a daemon used to manage nodes in a Rancher provisioned RKE2/K3s Kubernetes cluster when performing cluster lifecycle operations. Examples of cluster operations include upgrading the Kubernetes version and creating/restoring etcd snapshots. The `rancher-system-agent` is designed to apply plans to the Rancher system and can support both local and remote plans.

## Requests

The `cattle-cluster-agent` pod does not define the default CPU and memory request values. As a baseline, we recommend setting the CPU request at `50m` and memory request at `100Mi`. However, it is important that you assess your use case appropriately and that you allocate the correct resources to your cluster for your needs.

To configure request values through the UI for RKE2/K3s clusters:

1. When you create or edit an existing cluster, go to the **Cluster Configuration**.
2. Select the **Cluster Agent** subsection.
3. Configure your request values using the **CPU Reservation** and **Memory Reservation** fields as needed.

If you prefer to configure via YAML, add the following snippet to your RKE2/K3s configuration file:

```
spec:
  clusterAgentDeploymentCustomization:
    overrideResourceRequirements:
      requests:
        cpu: 50m
        memory: 100Mi
```

## Scheduling Rules

The `cattle-cluster-agent` uses either a fixed set of tolerations, or dynamically-added tolerations based on taints applied to the control plane nodes. This structure allows Taint based Evictions [https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/#taint-based-evictions] to work properly for `cattle-cluster-agent`.

If control plane nodes are present in the cluster, the default tolerations will be replaced with tolerations matching the taints on the control plane nodes. The default set of tolerations are described below.

| Component | nodeAffinity nodeSelectorTerms | nodeSelector | Tolerations |
|---|---|---|---|
| `cattle-cluster-agent` | `beta.kubernetes.io/os:NotIn:windows` | none | **Note:** These are the default tolerations, and will be replaced by tolerations matching taints applied to controlplane nodes.<br><br>`effect:NoSchedule`<br>`key:node-role.kubernetes.io/controlplane`<br>`value:true`<br><br>`effect:NoSchedule`<br>`key:node-role.kubernetes.io/control-plane`<br>`operator:Exists`<br><br>`effect:NoSchedule`<br>`key:node-role.kubernetes.io/master`<br>`operator:Exists` |

The `cattle-cluster-agent` Deployment has preferred scheduling rules using `preferredDuringSchedulingIgnoredDuringExecution`, favoring to be scheduled on nodes with the `controlplane` node. When there are no controlplane nodes visible in the cluster (this is usually the case when using Clusters from Hosted Kubernetes Providers), you can add the label `cattle.io/cluster-agent=true` on a node to prefer scheduling the `cattle-cluster-agent` pod to that node.

See Kubernetes: Assigning Pods to Nodes [https://kubernetes.io/docs/concepts/configuration/assign-pod-node/] to find more information about scheduling rules.

The `preferredDuringSchedulingIgnoredDuringExecution` configuration is shown in the table below:

| Weight | Expression |
|---|---|
| 100 | `node-role.kubernetes.io/controlplane:In:"true"` |
| 100 | `node-role.kubernetes.io/control-plane:In:"true"` |
| 100 | `node-role.kubernetes.io/master:In:"true"` |
| 1 | `cattle.io/cluster-agent:In:"true"` |

# IPv4/IPv6 Dual-stack

Kubernetes supports IPv4-only, IPv6-only, and dual-stack networking configurations. For more details, refer to the official Kubernetes documentation [https://kubernetes.io/docs/concepts/services-networking/dual-stack/].

## Installing Rancher on IPv6-Only or Dual-Stack Clusters

Rancher can run on clusters using:

- IPv4-only
- IPv6-only
- Dual-stack (IPv4 + IPv6)

When you install Rancher on an **IPv6-only cluster**, it can communicate externally **only over IPv6**. This means it can provision:

- IPv6-only clusters
- Dual-stack clusters *(IPv4-only downstream clusters are not possible in this case)*

When you install Rancher on a **dual-stack cluster**, it can communicate over both IPv4 and IPv6, and can therefore provision:

- IPv4-only clusters
- IPv6-only clusters
- Dual-stack clusters

For installation steps, see the guide: **Installing and Upgrading Rancher**.

#### Requirement for the Rancher Server URL

When provisioning IPv6-only downstream clusters, the **Rancher Server URL must be reachable over IPv6** because downstream nodes connect back to the Rancher server using IPv6.

## Provisioning IPv6-Only or Dual-Stack Clusters

You can provision RKE2 and K3s **Node driver** (machine pools) or **Custom cluster** (existing hosts) clusters using IPv4-only, IPv6-only, or dual-stack networking.

#### Network Configuration

To enable IPv6-only or dual-stack networking, you must configure:

- Cluster CIDR
- Service CIDR
- Stack Preference

Configuration references:

- K3s Cluster Configuration Reference
- RKE2 Cluster Configuration Reference

#### Support for Windows

Kubernetes on Windows:

| Feature | Support Status |
|---|---|
| IPv6-only clusters | Not supported |
| Dual-stack clusters | Supported |
| Services | Limited to a single IP famil |

For more information, see the Kubernetes Documentation [https://kubernetes.io/docs/concepts/services-networking/dual-stack/#windows-support].

K3s does **not** support Windows (FAQ [https://documentation.suse.com/cloudnative/k3s/latest/en/faq.html#_does_k3s_support_window])

RKE2 supports Windows, but requires using either `Calico` or `Flannel` as the CNI. Note that Windows installations of RKE2 do not support dual-stack clusters using BGP. For more details, see RKE2 Network Options [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html].

### Provisioning Node Driver Clusters

Rancher currently supports assigning IPv6 addresses in **node driver** clusters with:

- Amazon EC2
- DigitalOcean

Support for additional providers will be introduced in future releases.



*DigitalOcean Limitation*

Creating an **IPv6-only cluster** using the DigitalOcean node driver is currently **not supported**. For more details, please see rancher/rancher#52523 [https://github.com/rancher/rancher/issues/52523#issuecomment-3457803572].

#### Infrastructure Requirements

Cluster nodes must meet the requirements listed in the Node Requirements for Rancher Managed Clusters.

Machine pool configuration guides:

- Amazon EC2
- DigitalOcean

### Provisioning Custom Clusters

To provision on your own nodes, follow the instructions in Provision Kubernetes on Existing Nodes.



- **Node Public IP** and **Node Private IP** fields accept IPv4, IPv6, or both (comma-separated). > Example: `10.0.0.5,2001:db8::1`
- In **IPv6-only** and **dual-stack** clusters, specify the node's **IPv6 address** as the **Private IP**.

#### Infrastructure Requirements

Infrastructure requirements are the same as above for node-driver clusters.

### Limitations



Refer to the IPv4/IPv6 Dual-Stack Prime documentation [https://scc.suse.com/rancher-docs/rancherprime/latest/en/suse-rancher-prime/https://scc.suse.com/rancher-docs/rancherprime/latest/en/suse-rancher-prime/dual-stack.html] for additional limitations regarding Prime-only IPv6 URL support.

#### GitHub.com

GitHub.com does **not** support IPv6. As a result:

- Any application repositories ( `ClusterRepo.catalog.cattle.io/v1` CR) hosted on GitHub.com are **not reachable** from IPv6-only clusters.
- Similarly, any **non-builtin node drivers** hosted on GitHub.com are also **not accessible** in IPv6-only environments.

## Enabling Cluster Agent Scheduling Customization

The `cattle-cluster-agent` allows enabling automatic deployment of a Priority Class and Pod Disruption Budget.

When this feature is enabled, all newly provisioned Node Driver, Custom, and Imported RKE2 and K3s clusters will automatically deploy a Priority Class and Pod Disruption Budget during the provisioning process. Existing clusters can be gradually updated with this new behavior using the Rancher UI or by setting a specific annotation on cluster objects.

This feature is disabled by default.

### Enabling Cluster Agent Scheduling Customization



Enabling or disabling this feature only impacts new clusters. Existing downstream clusters will not be automatically updated. See *Updating Existing Clusters*.

1. In the upper left corner, click ☰ **> Global Settings**
2. Select **Feature Flags**
3. Find the `cluster-agent-scheduling-customization` feature and click ☰ **> Activate**

### Configuring the Global Settings

You can customize the default Priority Class (PC) and Pod Disruption Budget (PDB) by updating the `cluster-agent-default-priority-class` and `cluster-agent-default-pod-disruption-budget` global settings in the Rancher UI. Note that both the Priority Class and Pod Disruption Budget have configuration restrictions:

- The `Value` set for the default PC cannot be less than negative `1` billion, or greater than `1` billion.
- The `PreemptionPolicy` set for the PC must be equal to `PreemptLowerPriority` or `Never`.
- You cannot configure the PDB `minAvailable` and `maxUnavailable` fields to both have a non-zero value.

- The PDB `minAvailable` must either be a non-negative whole number integer, or a non-negative whole number percent (e.g. `1` or `100%`).
- The PDB `maxUnavailable` must either be a non-negative whole number integer, or a non-negative whole number percent (e.g. `1` or `100%`).

## Updating Existing Clusters



When this feature is disabled, you cannot modify the cluster agent scheduling customization fields for existing clusters. However, you can always remove the configuration, regardless of the feature's status.

After enabling this feature, you can configure scheduling customization for existing clusters in two ways:

- **Using the Rancher UI**
  - Edit the desired cluster and navigate to the **Cluster Agent** tab within the **Cluster Configuration** section.
  - Enable the `Prevent Rancher cluster agent pod eviction` checkbox.
    - The necessary fields on the associated `clusters.provisioning.cattle.io` or `clusters.management.cattle.io` object will be automatically configured using the values set in the global settings.
  - Save the cluster.
- **Using an annotation**
  - The `provisioning.cattle.io/enable-scheduling-customization` annotation can be used to update clusters without requiring the use of the Rancher UI. This annotation will be automatically removed from the cluster after the Priority Class and Pod Disruption Budget are configured.
    - The value of this annotation can be either `true` or `false`, to add or remove scheduling customization automatically.
    - For Node Driver Provisioned and Custom clusters, apply this annotation on the associated `clusters.provisioning.cattle.io` object.
    - For Imported clusters, apply the annotation on the associated `clusters.management.cattle.io` object.

## Applying Updated Global Settings

In order to prevent unexpected changes in scheduler behavior, Rancher does not update existing downstream clusters when the `cluster-agent-default-priority-class` and `cluster-agent-default-pod-disruption-budget` global settings are changed. There are two ways to update existing clusters to use the most recent global settings:

- **Using the Rancher UI**
  - When configuring a cluster, an additional checkbox will be shown in the **Cluster Agent** tab within the **Cluster Configuration** section. Checking the `Apply global settings for Priority Class and Pod Disruption Budget` checkbox will automatically update the Priority Class and Pod Disruption Budget to match the global settings once the cluster is saved.
- **Adjusting the cluster yaml**

- You may manually adjust the relevant fields in the cluster object using `kubectl` or the Rancher UI 'Edit As Yaml' feature. Scheduling customization can be found in the `spec.ClusterAgentDeploymentCustomization.SchedulingCustomization` section of the cluster object.
- Alternatively, the `provisioning.cattle.io/enable-scheduling-customization` annotation can be used to remove and re-add the updated scheduling customization fields set on a specific cluster.

### Downstream Objects

When this feature is enabled for a given cluster, two downstream resources will be automatically created by Rancher:

- A Pod Disruption Budget will be automatically created in the `cattle-system` namespace, named `cattle-cluster-agent-pod-disruption-budget`.
- A Priority Class will be automatically created, named `cattle-cluster-agent-priority-class`.

These objects are maintained by Rancher and must not be modified or deleted. The Rancher server will automatically update these objects to match the configuration set on the Cluster object and remove them when they are no longer needed.

#### RBAC considerations

Before enabling this feature on a downstream cluster, cluster administrators should assess their current RBAC configuration to prevent common access to the `cattle-cluster-agent-priority-class`. In cases where external users have access to a cluster, such as when offering clusters as a service, it is recommended to limit access to the `cattle-cluster-agent-priority-class` object to prevent changes or deletion.

Similar considerations do not need to be made for the `cattle-cluster-agent-pod-disruption-budget` object, as Pod Disruption Budgets are namespaced objects. Rancher will create the `cattle-cluster-agent-pod-disruption-budget` in the privileged `cattle-system` namespace.

## Launching Kubernetes on New Nodes in an Infrastructure Provider

When you create an RKE2 cluster using a machine config in Rancher, each resulting node pool is shown in a new **Machine Pools** tab. You can see the machine pools by doing the following:

1. Click ☰ **> Cluster Management**.
2. Click the name of the RKE2 cluster.

### SUSE® Rancher Prime: RKE2 Clusters

Rancher v2.6 introduces provisioning for RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html] clusters directly from the Rancher UI. RKE2, also known as RKE Government, is a fully conformant Kubernetes distribution that focuses on security and compliance within the U.S. Federal Government sector.



For RKE2 cluster templates, please refer to this page for additional information.

#### Node Roles

The RKE2 CLI exposes two roles, `server` and `agent`, which represent the Kubernetes node-roles `etcd` + `controlplane` and `worker` respectively. With RKE2 integration in Rancher v2.6, RKE2 node pools can represent more fine-grained role assignments such that `etcd` and `controlplane` roles can be represented.

The same functionality of using `etcd`, `controlplane` and `worker` nodes is possible in the RKE2 CLI by using flags and node tainting to control where workloads and the Kubernetes master were scheduled. The reason those roles were not implemented as first-class roles in the RKE2 CLI is that RKE2 is conceptualized as a set of raw building blocks that are best leveraged through an orchestration system such as Rancher.

In our recommended cluster architecture, we outline how many nodes of each role clusters should have:

- At least three nodes with the role etcd to survive losing one node
- At least two nodes with the role controlplane for master component high availability
- At least two nodes with the role worker for workload rescheduling upon node failure

### Creating a DigitalOcean Cluster

In this section, you'll learn how to deploy an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in DigitalOcean.

First, you set up your DigitalOcean cloud credentials in Rancher.

Then you create a DigitalOcean cluster in Rancher, and when configuring the new cluster, you define machine pools for it. Each machine pool has a Kubernetes role of etcd, controlplane, or worker. Rancher installs Kubernetes on the new nodes, and it sets up each node with the Kubernetes role defined by the machine pool.

#### Create your cloud credentials

If you already have a set of cloud credentials to use, skip this section.

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.
3. Click **Create**.
4. Click **DigitalOcean**.
5. Enter your Digital Ocean credentials.
6. Click **Create**.

#### Create your cluster

Use Rancher to create a Kubernetes cluster in DigitalOcean.

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Toggle the switch to **RKE2/K3s**.

4. Click **DigitalOcean**.
5. Select a **Cloud Credential**, if more than one exists. Otherwise, it's preselected.
6. Enter a **Cluster Name**.
7. Create a machine pool for each Kubernetes role. Refer to the best practices for recommendations on role assignments and counts.
    a. For each machine pool, define the machine configuration. Refer to the DigitalOcean machine configuration reference for information on configuration options.
8. Use the **Cluster Configuration** to choose the version of Kubernetes to install, what network provider to use and if you want to enable project network isolation. For help configuring the cluster, refer to the RKE2 and K3s cluster configuration references.
9. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
10. Click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.

### DigitalOcean Machine Configuration

For more details about DigitalOcean, Droplets, refer to the official documentation [https://docs.digitalocean.com/products/compute/].

#### Region

Configure the region [https://docs.digitalocean.com/glossary/region/] where Droplets are created.

#### Size

Configure the size [https://docs.digitalocean.com/products/droplets/resources/choose-plan/] of Droplets.

#### OS Image

Configure the operating system image [https://docs.digitalocean.com/products/images/] Droplets are created from.

#### Monitoring

Enable the DigitalOcean agent for additional monitoring [https://docs.digitalocean.com/products/monitoring/].

#### IPv6

Enable IPv6 for Droplets.

For more information, refer to the Digital Ocean IPv6 documentation [https://docs.digitalocean.com/products/networking/ipv6].

#### Private Networking

Enable private networking for Droplets.

#### Droplet Tags

Apply a tag (label) to a Droplet. Tags may only contain letters, numbers, colons, dashes, and underscores. For example, `my_server`.

### DigitalOcean Node Template Configuration

Account access information is stored as a cloud credential. Cloud credentials are stored as Kubernetes secrets. Multiple node templates can use the same cloud credential. You can use an existing cloud credential or create a new one.

#### Droplet Options

The **Droplet Options** provision your cluster's geographical region and specifications.

#### Docker Daemon

If you use Docker, the Docker daemon [https://docs.docker.com/engine/docker-overview/#the-docker-daemon] configuration options include:

- **Labels:** For information on labels, refer to the Docker object label documentation. [https://docs.docker.com/config/labels-custom-metadata/]
- **Docker Engine Install URL:** Determines what Docker version will be installed on the instance.
- **Registry mirrors:** Docker Registry mirror to be used by the Docker daemon
- **Other advanced options:** Refer to the Docker daemon option reference [https://docs.docker.com/engine/reference/commandline/dockerd/].

## Creating an Amazon EC2 Cluster

In this section, you'll learn how to deploy an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in Amazon EC2.

First, you set up your EC2 cloud credentials in Rancher.

Then you create an EC2 cluster in Rancher, and when configuring the new cluster, you define machine pools for it. Each machine pool has a Kubernetes role of etcd, controlplane, or worker. Rancher installs Kubernetes on the new nodes, and it sets up each node with the Kubernetes role defined by the machine pool.

### Prerequisites

- **AWS EC2 Access Key and Secret Key** that will be used to create the instances. See Amazon Documentation: Creating Access Keys [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey] how to create an Access Key and Secret Key.
- **IAM Policy created** to add to the user of the Access Key And Secret Key. See Amazon Documentation: Creating IAM Policies (Console) [https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html#access_policies_create-start] how to create an IAM policy. See our three example JSON policies below:
  - Example IAM Policy
  - Example IAM Policy with PassRole (needed if you want to use Kubernetes Cloud Provider or want to pass an IAM Profile to an instance)
  - Example IAM Policy to allow encrypted EBS volumes
- **IAM Policy added as Permission** to the user. See Amazon Documentation: Adding Permissions to a User (Console) [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console] how to attach it to a user.
- **IPv4-only or IPv6-only or dual-stack subnet and/or VPC** where nodes can be provisioned and assigned IPv4 and/or IPv6 addresses. See Amazon Documentation: IPv6 support for your VPC [https://docs.aws.amazon.com/vpc/latest/userguide/vpc-migrate-ipv6.html].

### Creating an EC2 Cluster

The steps to create a cluster differ based on your Rancher version.

#### Create your cloud credentials

If you already have a set of cloud credentials to use, skip this section.

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.
3. Click **Create**.
4. Click **Amazon**.
5. Enter a name for the cloud credential.
6. In the **Default Region** field, select the AWS region where your cluster nodes will be located.
7. Enter your AWS EC2 **Access Key** and **Secret Key**.
8. Click **Create**.

**Result:** You have created the cloud credentials that are used to provision nodes in your cluster.

#### Create your cluster

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Toggle the switch to **RKE2/K3s**.
4. Click **Amazon EC2**.
5. Select a **Cloud Credential**, if more than one exists. Otherwise, it's preselected.
6. Enter a **Cluster Name**.
7. Create a machine pool for each Kubernetes role. Refer to the best practices for recommendations on role assignments and counts.
    a. For each machine pool, define the machine configuration. Refer to the EC2 machine configuration reference for information on configuration options.
8. Use the **Cluster Configuration** to choose the version of Kubernetes that to install, what network provider to use and if you want to enable project network isolation. For help configuring the cluster, refer to the RKE2 and K3s cluster configuration references.
9. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
10. Click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

#### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.

### IAM Policies

#### Example IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:Describe*",
                "ec2:ImportKeyPair",
                "ec2:CreateKeyPair",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:DeleteKeyPair",
                "ec2:ModifyInstanceMetadataOptions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:REGION::image/ami-*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:instance/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:placement-group/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:volume/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:subnet/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:key-pair/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:network-interface/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:security-group/*"
            ]
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "ec2:RebootInstances",
                "ec2:TerminateInstances",
```

```
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:instance/*"
        }
    ]
}
```

**Example IAM Policy with PassRole**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:Describe*",
                "ec2:ImportKeyPair",
                "ec2:CreateKeyPair",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:DeleteKeyPair",
                "ec2:ModifyInstanceMetadataOptions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:REGION::image/ami-*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:instance/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:placement-group/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:volume/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:subnet/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:key-pair/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:network-interface/*",
                "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:security-group/*",
                "arn:aws:iam::AWS_ACCOUNT_ID:role/YOUR_ROLE_NAME"
```

```
            ]
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "ec2:RebootInstances",
                "ec2:TerminateInstances",
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:instance/*"
        }
    ]
}
```

**Example IAM Policy to allow encrypted EBS volumes**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Encrypt",
        "kms:DescribeKey",
        "kms:CreateGrant",
        "ec2:DetachVolume",
        "ec2:AttachVolume",
        "ec2:DeleteSnapshot",
        "ec2:DeleteTags",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:DeleteVolume",
        "ec2:CreateSnapshot"
      ],
      "Resource": [
        "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:volume/*",
        "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:instance/*",
        "arn:aws:ec2:REGION:AWS_ACCOUNT_ID:snapshot/*",
        "arn:aws:kms:REGION:AWS_ACCOUNT_ID:key/KMS_KEY_ID"
      ]
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeTags",
        "ec2:DescribeVolumes",
        "ec2:DescribeSnapshots"
      ],
      "Resource": "*"
    }
  ]
}
```

### EC2 Machine Configuration Reference

For more details about EC2 nodes, refer to the official documentation for the EC2 Management Console [https://aws.amazon.com/ec2].

#### Basic Options

##### Region

The geographical region [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html] in which to build your cluster.

##### Zone

The zone [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-availability-zones], an isolated location within a region to build your cluster

##### Instance Type

The instance type [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html], which determines the hardware characteristics, used to provision your cluster.

##### Root Disk Size

Configure the size (in GB) for your root device [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/RootDeviceStorage.html].

##### VPC/Subnet

The VPC [https://docs.aws.amazon.com/vpc/latest/userguide/configure-your-vpc.html] or specific subnet [https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html], an IP range in your VPC, to add your resources to.

##### IAM Instance Profile Name

The name of the instance profile [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html] used to pass an IAM role to an EC2 instance.

#### Advanced Options

##### AMI ID

The Amazon Machine Image [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html] used for the nodes in your cluster.

##### SSH Username for AMI

The username for connecting to your launched instances. Refer to here [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connection-prereqs.html] for the default usernames to selected AMIs. For AMIs not listed, check with the AMI provider.

##### Security Group

Choose the default security group or configure a security group.

Please refer to Amazon EC2 security group when using Node Driver to see what rules are created in the `rancher-nodes` Security Group.

##### EBS Root Volume Type

The EBS volume type [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html] to use for the root device.

##### Encrypt EBS Volume

Enable Amazon EBS Encryption [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html].

##### Request Spot Instance

Enable option to request spot instances [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-requests.html] and specify the maximum instance price per hour you're willing to pay.

##### Use only private address

Enable option on use only private addresses [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html].

##### EBS-Optimized Instance

Use an EBS-optimized instance [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html].

##### Allow access to EC2 metadata

Enable access to EC2 metadata [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html].

##### Use tokens for metadata

Use Instance Metadata Service Version 2 (IMDSv2) [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.html], a token-based method to access metadata.

#### Add Tag

Add metadata using tags [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html] to categorize resources.

#### IPv6 Address Count

Specify how many IPv6 addresses to assign to the instance's network interface.

#### IPv6 Address Only

Enable this option if the instance should use IPv6 exclusively. IPv6-only VPCs or subnets require this. When enabled, the instance has IPv6 as its sole address, and the IPv6 Address Count must be greater than zero.

#### HTTP Protocol IPv6

Enable or disable IPv6 endpoints for the instance metadata service.

#### Enable Primary IPv6

Enable this option to designate the first assigned IPv6 address as the primary address. This ensures a consistent, non-changing IPv6 address for the instance. It does not control whether IPv6 addresses are assigned.

### EC2 Node Template Configuration

For more details about EC2, nodes, refer to the official documentation for the EC2 Management Console [https://aws.amazon.com/ec2].

#### Region

In the **Region** field, select the same region that you used when creating your cloud credentials.

#### Cloud Credentials

Your AWS account access information, stored in a cloud credential.

See Amazon Documentation: Creating Access Keys [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey] how to create an Access Key and Secret Key.

See Amazon Documentation: Creating IAM Policies (Console) [https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html#access_policies_create-start] how to create an IAM policy.

See Amazon Documentation: Adding Permissions to a User (Console) [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console] how to attach an IAM

See our three example JSON policies:

- Example IAM Policy
- Example IAM Policy with PassRole (needed if you want to use Kubernetes Cloud Provider or want to pass an IAM Profile to an instance)
- Example IAM Policy to allow encrypted EBS volumes policy to an user.

#### Authenticate & Configure Nodes

Choose an availability zone and network settings for your cluster.

#### Security Group

Choose the default security group or configure a security group.

Please refer to Amazon EC2 security group when using Node Driver to see what rules are created in the `rancher-nodes` Security Group.

If you provide your own security group for an EC2 instance, please note that Rancher will not modify it. As such, you will be responsible for ensuring that your security group is set to allow the necessary ports for Rancher to provision the instance. For more information on controlling inbound and outbound traffic to EC2 instances with security groups, refer here [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html#WorkingWithSecurityGroups].

#### Instance Options

Configure the instances that will be created. Make sure you configure the correct **SSH User** for the configured AMI. It is possible that a selected region does not support the default instance type. In this scenario you must select an instance type that does exist, otherwise an error will occur stating the requested configuration is not supported.

If you need to pass an **IAM Instance Profile Name** (not ARN), for example, when you want to use a Kubernetes Cloud Provider, you will need an additional permission in your policy. See Example IAM policy with PassRole for an example policy.

#### Engine Options

In the **Engine Options** section of the node template, you can configure the container daemon. You may want to specify the container version or a container image registry mirror.

## Creating an Azure Cluster

In this section, you'll learn how to deploy an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in Azure through Rancher.

First, you set up your Azure cloud credentials in Rancher.

Then you create an Azure cluster in Rancher, and when configuring the new cluster, you define machine pools for it. Each machine pool has a Kubernetes role of etcd, controlplane, or worker. Rancher installs Kubernetes on the new nodes, and it sets up each node with the Kubernetes role defined by the machine pool.



When the Rancher RKE2/K3s cluster is running in Azure and has an Azure load balancer in front, the outbound flow fails. The workaround for this problem is as follows:

- Terminate the SSL/TLS on the internal load balancer
- Use the L7 load balancer

For more information, refer to the documentation on Azure load balancer limitations [https://docs.microsoft.com/en-us/azure/load-balancer/components#_limitations].

For more information on configuring the Kubernetes cluster that Rancher installs on the Azure nodes, refer to the RKE2 and K3s cluster configuration references.

For more information on configuring Azure machines, refer to the Azure machine configuration reference.

- Preparation in Azure
- Creating an Azure Cluster

#### Preparation in Azure

Before a cluster can be deployed, we must configure our Azure subscription to allow the manipulation of its resources by a third party, such as Rancher.

To do this, we will first create a new Azure **service principal (SP)** in Azure **Active Directory (AD)**, which, in Azure, is an application user who has permission to manage Azure resources.

The following is a template `az cli` script that you have to run for creating an service principal, where you have to enter your SP name, role, and scope:

```
az ad sp create-for-rbac \
  --name="<Rancher ServicePrincipal name>" \
  --role="Contributor" \
  --scopes="/subscriptions/<subscription Id>"
```

The creation of this service principal returns the **application ID**, also called the **client ID**, and the **client secret**. This information is used when you create your cloud credentials.

#### Creating an Azure Cluster

##### Create your cloud credentials

If you already have a set of cloud credentials to use, skip this section.

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.
3. Click **Create**.
4. Click **Azure**.
5. Enter your Azure credentials.
6. Click **Create**.

**Result:** You have created the cloud credentials that are used to provision nodes in your cluster.

##### Create your cluster

Use Rancher to create a Kubernetes cluster in Azure.

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.

3. Toggle the switch to **RKE2/K3s**.
4. Click **Azure**.
5. Select a **Cloud Credential**, if more than one exists. Otherwise, it's preselected.
6. Enter a **Cluster Name**.
7. Create a machine pool for each Kubernetes role. Refer to the best practices for recommendations on role assignments and counts.
    a. For each machine pool, define the machine configuration. Refer to the Azure machine configuration reference for information on configuration options.
8. Use the **Cluster Configuration** to choose the version of Kubernetes to install, what network provider to use and if you want to enable project network isolation. For help configuring the cluster, refer to the RKE2 and K3s cluster configuration references.
9. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
10. Click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

#### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.

### Azure Machine Configuration

For more information about Azure, refer to the official Azure documentation. [https://docs.microsoft.com/en-us/azure/?product=featured]

#### Basic Options

#### Environment

Microsoft provides multiple clouds [https://docs.microsoft.com/en-us/cli/azure/cloud?view=azure-cli-latest] for compliance with regional laws, which are available for your use:

- AzurePublicCloud
- AzureGermanCloud
- AzureChinaCloud
- AzureUSGovernmentCloud

#### Location

Configure the cluster and node location [https://docs.microsoft.com/en-us/azure/virtual-machines/regions].

#### Resource Group

A resource group is a container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Generally, add resources that share the same lifecycle to the same resource group so you can easily deploy, update, and delete them as a group.

Use an existing resource group or enter a resource group name and one will be created for you.

For information on managing resource groups, see the Azure documentation. [https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal]

#### Availability Set (unmanaged)

Name or ID of an existing availability set [https://docs.microsoft.com/en-us/azure/virtual-machines/availability-set-overview] to add the VM to.

#### Image

The name of the operating system image provided as an ARM resource identifier. Requires using managed disk.

#### VM Size

Choose a size for each VM in the node pool. For details about each VM size, see this page. [https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/]

#### Advanced Options

#### Fault Domain Count

Fault domains define the group of virtual machines that share a common power source and network switch. If the availability set has already been created, the fault domain count will be ignored.

For more information on fault domains, see refer here [https://docs.microsoft.com/en-us/azure/virtual-machines/availability-set-overview#how-do-availability-sets-work].

#### Update Domain Count

Update domains indicate groups of virtual machines and underlying physical hardware that can be rebooted at the same time. If the availability set has already been created, the update domain count will be ignored.

For more information on update domains, see refer here [https://docs.microsoft.com/en-us/azure/virtual-machines/availability-set-overview#how-do-availability-sets-work].

#### Purchase Plan

Some VM images in the Azure Marketplace require a plan. If applicable, select a purchase plan, formatted as `publisher:product:plan`, to use with your chosen image.

#### Subnet

The name of the subnet when creating a new VNet or referencing an existing one.

Default: `docker-machine`

#### Subnet Prefix

The subnet IP address prefix to use when creating a new VNet in CIDR format.

Default: `192.168.0.0/16`

#### Virtual Network

The virtual network [https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview] to use or create if one does not exist. Formatted as `[resourcegroup:]name`.

#### Public IP Options

#### No Public IP

Do not allocate a public IP address.

#### Static Public IP

Allocate a static public IP address.

#### Use Private IP

Use a static private IP address.

#### Private IP Address

Configure a static private IP address to use.

#### Network Security Group

The network security group [https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview] to use. All nodes using this template will use the supplied network security group. If no network security group is provided, a new one will be created for each node.

#### DNS Label

A unique DNS name label for the public IP address.

#### Storage Type

The storage account [https://docs.microsoft.com/en-us/azure/storage/common/storage-account-overview] type to use with your VMs. Options include Standard LRS, Standard ZRS, Standard GRS, Standard RAGRS, and Premium LRS.

#### Use Managed Disks

Azure managed disks [https://docs.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview] are block-level storage volumes that are managed by Azure and used with Azure Virtual Machines. Managed disks are designed for 99.999% availability. Managed disks achieve this by providing you with three replicas of your data, allowing for high durability.

#### Managed Disk Size

The size in GB for the disk for each node.

#### SSH Username

The username used to create an SSH connection to your nodes.

#### Open Port

Opens inbound traffic on specified ports. When using an existing Network Security Group, Open Ports are ignored.

Default: `2379/tcp, 2380/tcp, 6443/tcp, 9796/tcp, 10250/tcp, 10251/tcp, 10252/tcp, 10256/tcp` and `8472/udp, 4789/udp`

### Azure Node Template Configuration

For more information about Azure, refer to the official Azure documentation. [https://docs.microsoft.com/en-us/azure/?product=featured]

Account access information is stored as a cloud credential. Cloud credentials are stored as Kubernetes secrets. Multiple node templates can use the same cloud credential. You can use an existing cloud credential or create a new one.

- **Placement** sets the geographical region where your cluster is hosted and other location metadata.
- **Network** configures the networking used in your cluster.
- **Instance** customizes your VM configuration.



If using a VNet in a different Resource Group than the VMs, the VNet name should be prefixed with the Resource Group name. For example, `<resource group>:<vnet>`.

If you use Docker, the Docker daemon [https://docs.docker.com/engine/docker-overview/#the-docker-daemon] configuration options include:

- **Labels:** For information on labels, refer to the Docker object label documentation.

[https://docs.docker.com/config/labels-custom-metadata/].

- **Docker Engine Install URL:** Determines what Docker version will be installed on the instance.



If you're provisioning Red Hat Enterprise Linux (RHEL) or CentOS nodes, leave the **Docker Install URL** field as the default value, or select **none**. This will bypass a check for Docker installation, as Docker is already installed on these node types.

If you set **Docker Install URL** to a value other than the default or **none**, you might see an error message such as the following: `Error creating machine: RHEL ssh command error: command: sudo -E yum install -y curl err: exit status 1 output: Updating Subscription Management repositories.`

- **Registry mirrors:** Docker Registry mirror to be used by the Docker daemon.
- **Other advanced options:** Refer to the Docker daemon option reference [https://docs.docker.com/engine/reference/commandline/dockerd/].

## Creating a Google Compute Engine cluster

In this section, you'll learn how to use Rancher to provision an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster on the Google Cloud Platform (GCP) using Google Compute Engine (GCE) through Rancher.

First, you will enable the GCE node driver in the Rancher UI. Then, you follow the steps to create a GCP service account with the necessary permissions, and generate a JSON key file. This key file will be used to create a cloud credential in Rancher.

Then, you will create a GCE cluster in Rancher, and when configuring the cluster, you will define machine pools for it. Each machine pool will have a Kubernetes role of etcd, controlplane, or worker. Rancher will install RKE2 onto the new nodes, and it will set up each node with the Kubernetes role defined by the machine pool.

1. Enable the GCE node driver
2. Create a cloud credential
3. Create a cluster using the cloud credential
4. GCE Best Practices

### Prerequisites

1. A valid Google Cloud Platform account and project.
2. A GCP Service Account JSON key file. The service account associated with this key must have the following IAM roles:
   a. **Compute Admin**
   b. **Service Account User**
   c. **Viewer**
3. A VPC Network to provision VMs within.

Refer to the GCP documentation [https://cloud.google.com/iam/docs/service-account-overview] on creating and

managing service account keys for more details.

**Enable the GCE node driver**

The GCE node driver is not enabled by default in Rancher. You must enable it before you can provision GCE clusters, or work with GCE specific CRDs.

1. Click ☰ > **Cluster Management**.
2. On the left hand side, click **Drivers**.
3. Open the **Node Drivers** tab.
4. Find the **Google GCE** driver and select ☰ > **Activate**.

**Create a cloud credential**

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.
3. Click **Create**.
4. Click **Google**.
5. Enter your GCP Service Account JSON key file.
6. Click **Create**.

**Result:** You have created the cloud credentials that will be used to provision nodes in your cluster. You can reuse these credentials in other clusters. Depending on the permissions granted to the service account, this credential may also be used for GKE clusters.

**Create a cluster using the cloud credential**

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **Google GCE**.
4. Select a **Cloud Credential** and provide the GCP project to create the VM in.
5. Enter a **Cluster Name**.
6. Create a machine pool for each Kubernetes role. Refer to the [best practices](use-new-nodes-in-an-infra-provider.md#node-roles) for recommendations on role assignments and counts.
    a. For each machine pool, define the machine configuration. Refer to the Google GCE machine configuration reference for information on configuration options.
7. Use the **Cluster Configuration** to choose the version of Kubernetes that will be installed, what network provider will be used and if you want to enable project network isolation. For help configuring the cluster, refer to the RKE2 and K3s cluster configuration references.
8. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
9. Click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

### GCE Best Practices

#### External Firewall Rules, Open Ports, and ACE

If the cluster being provisioned will utilize the Authorized Cluster Endpoint (ACE) feature, controlplane nodes must expose port 6443. This port is not exposed in the default machine pool configuration to prevent it from being exposed across all cluster nodes, and to reduce the number of firewall rules created by Rancher.

In order for ACE to work as expected, you must specify this port in the Rancher UI when configuring the controlplane machine pool by enabling the `Expose external ports` checkbox, under the `Show Advanced` section of the machine pool configuration UI. Alternatively, you may manually create a custom firewall rule in GCP and provide the related network tag in the controlplane machine-pool configuration.

#### Internal Firewall Rules

Rancher will automatically create a firewall rule and network tag to facilitate communication between cluster nodes internally within the specified VPC network. This rule will contain the minimum number of ports required to create an RKE2/K3s cluster.

If you need to extend the number of ports exposed internally between cluster nodes, a new firewall rule should be manually created, and the associated network tag assigned to the relevant machine pools. If desired, the automatic creation of the internal firewall rule can be disabled for each given machine pool when creating or updating the cluster.

#### Cross Network Deployments

While it is possible to deploy different machine pools into different VPC networks, the internal firewall rule created by Rancher does not support this configuration by default. To create machine pools in different networks, additional firewall rules to facilitate communication between nodes in different networks must be manually created.

### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher,

you can still access the cluster.

### GCE Machine Configuration

For more information about Google Cloud Platform (GCP) and the Google Compute Engine (GCE), refer to the official GCP documentation [https://cloud.google.com/docs].

#### Basic Options

##### Zone

The GCP Region and Zone that the VM will be deployed to. For example, `us-east1-b`.

##### Machine Image Project

The image project that the desired image families belong to.

##### Machine Image Family

The image family that the desired machine operating system belongs to.

##### Machine Image

The operating system that will be installed onto the VM.

##### Disk Type

The type of the disk attached to the VM. The available types may differ between regions.

##### Disk Size

The size of the disk attached to the VM, in Gigabytes.

##### Machine Type

The type of VM that will be deployed. Machine types determine the number of resources (vCPU, RAM, etc.) allocated for each node.

##### Network

The VPC network that the VM will be created in. This value cannot be changed once the machine pool has been provisioned.

##### Subnet

The VPC subnetwork that the VM will be created in. This value cannot be changed once the machine pool has been provisioned.

##### Username

A custom username set as the default user of the GCE VM.

##### External Address

The desired external IP address for the GCE VM.

#### Scopes

A list of OAuth2 scopes which allow the VM to access other GCP APIs.

#### Allow Internal Communication

By default, a VPC firewall rule is automatically created to expose a fixed set of ports within the VPC to facilitate communication between cluster nodes. This behavior can be disabled on a per machine pool basis, when clicking the `Show Advanced` option and disabling the `Allow Internal Communication` checkbox.

#### Expose External ports

A list of ports to be opened *externally* to the wider internet. Open ports are defined at the machine pool level. Enabling this option will result in the automatic creation of a VPC firewall rule. This rule will be automatically deleted when the cluster or machine pool is deleted.

#### Network Tags

Tags is a list of *network tags*, which can be used to associate preexisting Firewall Rules with all VMs within a machine pool.

#### Labels

A comma-separated list of custom labels to be attached to all VMs within a given machine pool. Unlike Tags, Labels do not influence networking behavior and only serve to organize cloud resources.

#### Advanced Options

When creating clusters via the Rancher UI some options are automatically configured for you. However, when creating machine config objects manually, you must ensure you properly configure the below fields.

#### external-firewall-rule-prefix

A prefix that will be used when creating the firewall rule to expose ports publicly. Ideally, this should be a concatenation the machine pool name and the cluster name. This field must be set if the machine pool is configured to expose ports publicly, otherwise it can be omitted.

#### internal-firewall-rule-prefix

A prefix that will be used when creating the internal firewall rule which allows for communication between nodes within the cluster. If this field is omitted, no internal firewall rule will be created.

## Creating a Nutanix AOS Cluster

Nutanix Acropolis Operating System [https://www.nutanix.com/products/acropolis] (Nutanix AOS) is an operating system for the Nutanix hyper-converged infrastructure platform. AOS comes with a built-in hypervisor called Acropolis Hypervisor [https://www.nutanix.com/products/ahv], or AHV. By using Rancher with Nutanix AOS (AHV), you can bring cloud operations on-premises.

Rancher can provision nodes in AOS (AHV) and install Kubernetes on them. When creating a Kubernetes cluster in AOS, Rancher first provisions the specified number of virtual machines by communicating with the Prism Central API. Then it installs Kubernetes on top of the VMs.

A Nutanix cluster may consist of multiple groups of VMs with distinct properties, such as the amount of memory or the number of vCPUs. This grouping allows for fine-grained control over the sizing of nodes for each Kubernetes role.

### Creating a Nutanix Cluster

Refer to the Provisioning Kubernetes Clusters in Nutanix AOS to learn how to use Rancher to install an RKE2 [https://documentation.suse.com/cloudnative/rke2/] / K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in Nutanix AOS.

### Provisioning Kubernetes Clusters in Nutanix AOS

To use Rancher to install an RKE2/K3s Kubernetes cluster in Nutanix AOS (AHV) refer to the Nutanix documentation [https://portal.nutanix.com/page/documents/solutions/details?targetId=BP-2103-Rancher-SUSE-Nutanix:new-rke2-or-k3s-clusters-deployment.html].

### Nutanix Node Template Configuration

#### Account Access

| Parameter | Required | Description | Default |
|---|---|---|---|
| Management Endpoint | ◪ | Hostname/IP address of Prism Central | |
| Username | ◪ | Username of the Prism Central user | |
| Password | ◪ | Password of the Prism Central user | |
| Allow insecure communication | | Set to true to allow insecure SSL communication to Prism Central | False |

#### Scheduling

Choose what Nutanix cluster the virtual machine will be scheduled to.

| Parameter | Required | Description |
|---|---|---|
| Cluster | ◪ | Name of the Nutanix cluster where the VM should be deployed (case sensitive) |

#### Instance Options

In the **Instance Options** section, configure the number of vCPUs, memory, and disk size for the VMs created by this template.

| Parameter | Required | Description | Default |
|---|---|---|---|
| CPUs | | Number of vCPUs allocated to the VM (cores) | 2 |
| Memory | | Amount of RAM allocated to the VM (MB) | 2 GB |

| Parameter | Required | Description | Default |
|---|---|---|---|
| Template Image | ⬕ | Name of the Disk Image template to clone as the VM's primary disk (must support cloud-init) | |
| VM Disk Size | | New size of the VM's primary disk (in GiB) | |
| Additional Disk Size | | Size of an additional disk to add to the VM (in GiB) | |
| Storage Container | | Storage container *UUID* in which to provision an additional disk | |
| Cloud Config YAML | | Cloud-init to provide to the VM (will be patched with Rancher root user) | |
| Network | ⬕ | Name(s) of the network(s) to attach to the VM | |
| VM Categories | | Name(s) of any categories to be applied to the VM | |

The VM may use any modern Linux operating system that is configured with support for cloud-init [https://cloudinit.readthedocs.io/en/latest/] using the Config Drive v2 datasource [https://cloudinit.readthedocs.io/en/latest/reference/datasources/configdrive.html].

#### Networks

The node template allows a VM to be provisioned with multiple networks. In the **Network** field, you can click **Add** to add any networks available to you in AOS.

#### VM Categories

A category is a grouping of entities into a key value pair. Typically, VMs are assigned to a category based on some criteria. Policies can then be tied to those entities that are assigned (grouped by) a specific category value.

#### cloud-init

Cloud-init [https://cloudinit.readthedocs.io/en/latest/] allows you to initialize your nodes by applying configuration on the first boot. This may involve things such as creating users or authorizing SSH keys.

To make use of cloud-init initialization, paste a cloud config using valid YAML syntax into the **Cloud Config YAML** field. Refer to the cloud-init documentation [https://cloudinit.readthedocs.io/en/latest/topics/examples.html] for a commented set of examples of supported cloud config directives.

Note that cloud-init based network configuration is not recommended and only supported via user data `runcmd` rather than by NoCloud or other network configuration datasources.

Nutanix IP Address Management (IPAM) or another DHCP service is recommended.

#### Engine Options

In the **Engine Options** section of the node template, you can configure the container daemon. You may want to specify the container version or a container image registry mirror.



If you're provisioning Red Hat Enterprise Linux (RHEL) or CentOS nodes, leave the **Docker Install URL** field as the default value, or select **none**. This will bypass a check for Docker installation, as Docker is already installed on these node types.

If you set **Docker Install URL** to a value other than the default or **none**, you might see an error message such as the following: `Error creating machine: RHEL ssh command error: command: sudo -E yum install -y curl err: exit status 1 output: Updating Subscription Management repositories.`

## Creating a VMware vSphere Cluster

import YouTube from '@site/src/components/YouTube'

By using Rancher with VMware vSphere, you can bring cloud operations on-premises.

Rancher can provision nodes in vSphere and install Kubernetes on them. When creating a Kubernetes cluster in vSphere, Rancher first provisions the specified number of virtual machines by communicating with the vCenter API. Then it installs Kubernetes on top of them.

A vSphere cluster may consist of multiple groups of VMs with distinct properties, such as the amount of memory or the number of vCPUs. This grouping allows for fine-grained control over the sizing of nodes for each Kubernetes role.

### Creating a VMware vSphere Cluster

In this section, you'll learn how to use Rancher to install an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in vSphere.

### Provisioning Storage

For an example of how to provision storage in vSphere using Rancher, refer to this section. In order to dynamically provision storage in vSphere, the vSphere provider must be enabled. Refer to in-tree vSphere config and out-of-tree vSphere config.

### Enabling the VMware vSphere Cloud Provider

When a cloud provider is set up in Rancher, the Rancher server can automatically provision new infrastructure for the cluster, including new nodes or persistent storage devices.

For details on enabling the vSphere cloud provider, refer to in-tree vSphere config and out-of-tree vSphere config.

### Best Practices for SUSE Rancher Prime Managed VMware vSphere Clusters

This guide outlines a reference architecture for provisioning downstream Rancher clusters in a vSphere environment, in addition to standard vSphere best practices as documented by VMware.

- 1. VM Considerations
- 2. Network Considerations
- 3. Storage Considerations
- 4. Backups and Disaster Recovery

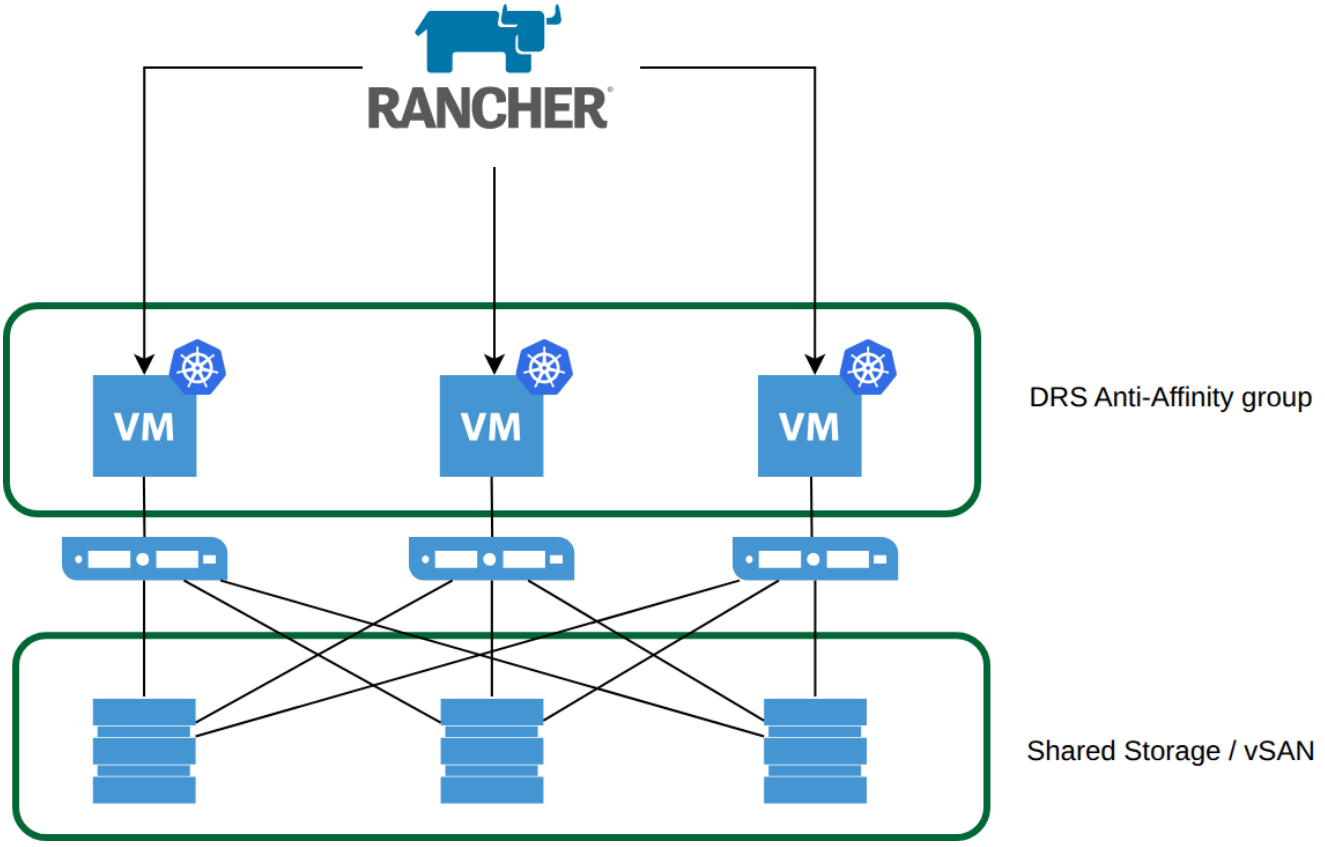



*Figure 12. Solution Overview*

**1. VM Considerations**

**Leverage VM Templates to Construct the Environment**

To facilitate consistency across the deployed Virtual Machines across the environment, consider the use of "Golden Images" in the form of VM templates. Packer can be used to accomplish this, adding greater customisation options.

**Leverage DRS Anti-Affinity Rules (Where Possible) to Separate Downstream Cluster Nodes Across ESXi Hosts**

Doing so will ensure node VM's are spread across multiple ESXi hosts - preventing a single point of failure at the host level.

**Leverage DRS Anti-Affinity Rules (Where Possible) to Separate Downstream Cluster Nodes Across Datastores**

Doing so will ensure node VM's are spread across multiple datastores - preventing a single point of failure at the datastore level.

**Configure VM's as Appropriate for Kubernetes**

It's important to follow K8s and etcd best practices when deploying your nodes, including disabling swap, double-checking you have full network connectivity between all machines in the cluster, using unique hostnames, MAC addresses, and product_uuids for every node.

**2. Network Considerations**

**Leverage Low Latency, High Bandwidth Connectivity Between ETCD Nodes**

Deploy etcd members within a single data center where possible to avoid latency overheads and reduce the likelihood of network partitioning. For most setups, 1Gb connections will suffice. For large clusters, 10Gb

connections can reduce the time taken to restore from backup.

#### Consistent IP Addressing for VM's

Each node used should have a static IP configured. In the case of DHCP, each node should have a DHCP reservation to make sure the node gets the same IP allocated.

### 3. Storage Considerations

#### Leverage SSD Drives for ETCD Nodes

ETCD is very sensitive to write latency. Therefore, leverage SSD disks where possible.

### 4. Backups and Disaster Recovery

#### Perform Regular Downstream Cluster Backups

Kubernetes uses etcd to store all its data - from configuration, state and metadata. Backing this up is crucial in the event of disaster recovery.

#### Back up Downstream Node VMs

Incorporate the Rancher downstream node VM's within a standard VM backup policy.

## Creating Credentials in the VMware vSphere Console

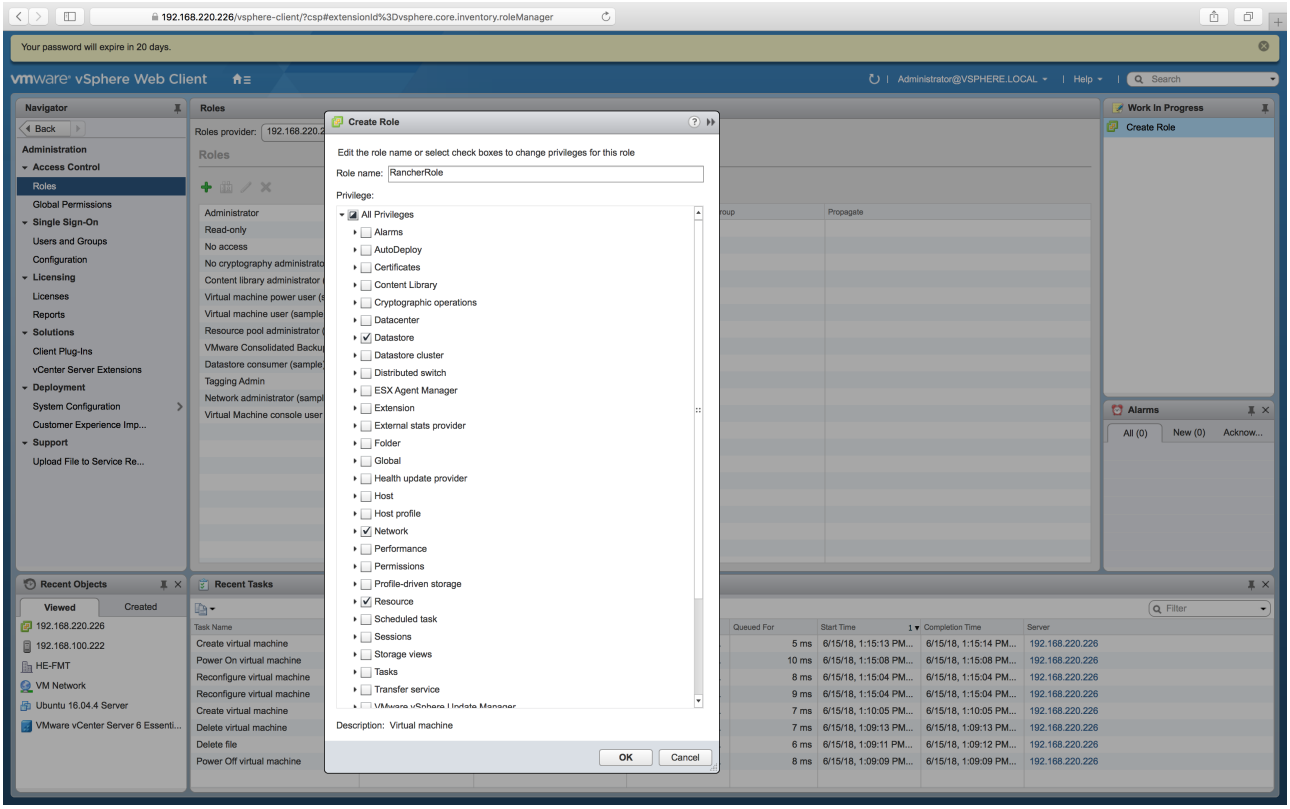
This section describes how to create a VMware vSphere username and password. You must provide these vSphere credentials to Rancher, which allows Rancher to provision resources in vSphere.

The following table lists the permissions required for the vSphere user account:

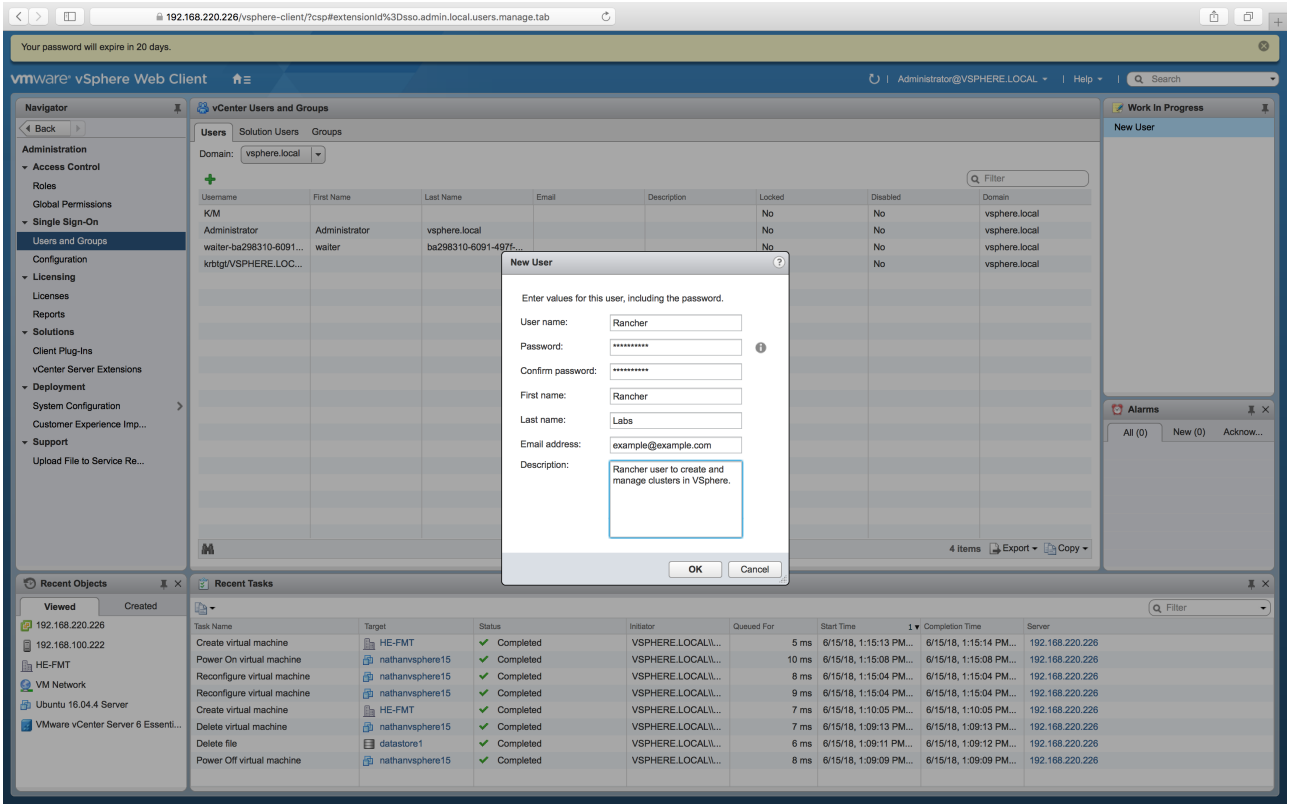
| Privilege Group | Operations |
|---|---|
| Cns Privileges | Searchable |
| Content library | Read Storage<br>(Required only if deploying Template from Content Library) |
| Cryptographic operations | Direct Access |
| Datastore | AllocateSpace<br>Browse<br>FileManagement (Low level file operations)<br>UpdateVirtualMachineFiles<br>UpdateVirtualMachineMetadata |
| Global | Set custom attribute |
| Network | Assign |
| Resource | AssignVMToPool |
| Virtual Machine | Config (All)<br>GuestOperations (All)<br>Interact (All)<br>Inventory (All)<br>Provisioning (All) |
| vSphere Tagging | Assign or Unassign vSphere Tag<br>Assign or Unassign vSphere Tag on Object |

The following steps create a role with the required privileges and then assign it to a new user in the vSphere console:

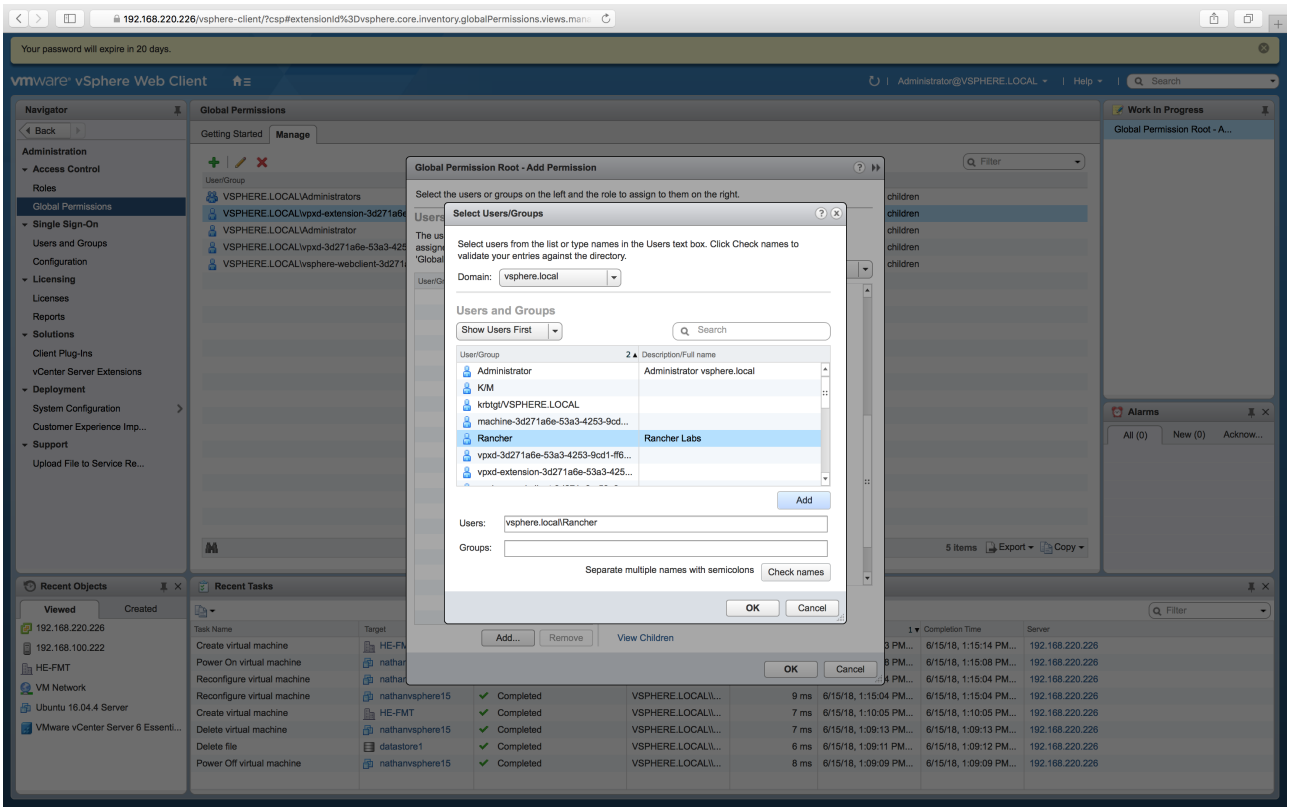
1. From the **vSphere** console, go to the **Administration** page.
2. Go to the **Roles** tab.
3. Create a new role. Give it a name and select the privileges listed in the permissions table above.

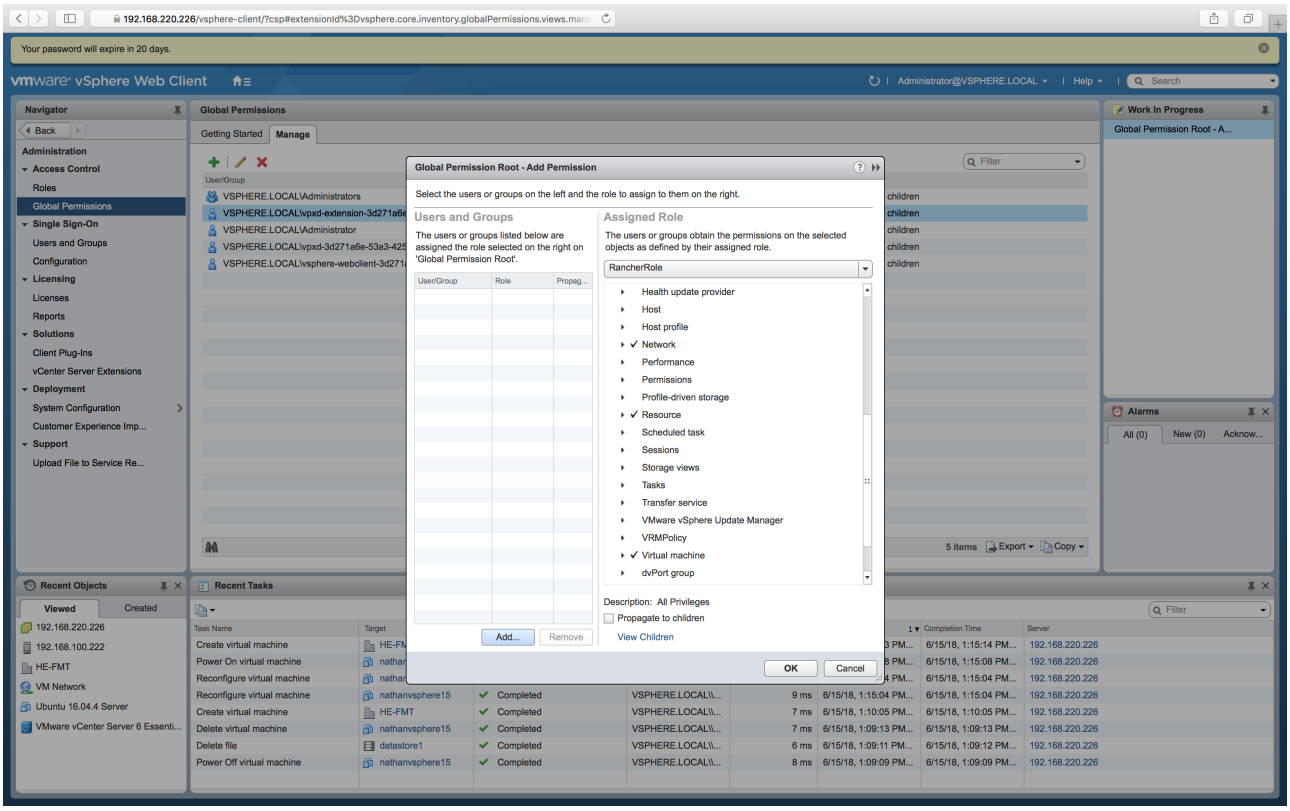



4. Go to the **Users and Groups** tab.
5. Create a new user. Fill out the form and then click **OK**. Make sure to note the username and password, because you need it when creating cloud credentials in Rancher.




6. Go to the **Global Permissions** tab.
7. Create a new Global Permission. Add the user you created earlier and assign it the role you created earlier. Click **OK**.






**Result:** You now have credentials that Rancher can use to manipulate vSphere resources.

### Creating a VMware vSphere Virtual Machine Template

Creating virtual machines in a repeatable and reliable fashion can often be difficult. VMware vSphere offers the ability to build one VM that can then be converted to a template. The template can then be used to create identically configured VMs. Rancher leverages this capability to create identical RKE/K3s nodes.

In order to leverage the template to create new VMs, Rancher has some specific requirements that the VM must have pre-installed. After you configure the VM with these requirements, you need to prepare the VM before creating the template. Finally, once preparation is complete, the VM can be converted to a template and moved into a content library.

#### Requirements

There is specific tooling required for both Linux and Windows VMs to be usable by the vSphere node driver. The most critical dependency is cloud-init [https://cloud-init.io/] for Linux and cloudbase-init [https://cloudbase.it/cloudbase-init/] for Windows. Both of these are used for provisioning the VMs by configuring the hostname and by setting up the SSH access and the default Rancher user. Users can add more content to these as desired if other configuration is needed. In addition, other requirements are listed below for reference.



If you have any specific firewall rules or configuration, you will need to add this to the VM before creating a template.

#### Linux Dependencies

The packages that need to be installed on the template are listed below. These will have slightly different names based on distribution; for example, some distributions ship these dependencies by default. The cluster provisioner will automatically install the dependencies required for Kubernetes. The dependencies listed below are required for the functioning of the Rancher cluster provisioner (not for Kubernetes):

- curl
- wget
- git
- net-tools
- unzip
- apparmor-parser
- ca-certificates
- cloud-init
- cloud-guest-utils
- cloud-image-utils
- growpart (part of cloud-guest-utils)
- cloud-initramfs-growroot
- open-iscsi
- openssh-server
- open-vm-tools [https://docs.vmware.com/en/VMware-Tools/11.3.0/com.vmware.vsphere.vmwaretools.doc/GUID-8B6EA5B7-453B-48AA-92E5-DB7F061341D1.html]

#### Windows Dependencies

The list of packages that need to be installed on the template is as follows:

- Windows Container Feature
- cloudbase-init [https://cloudbase.it/cloudbase-init/#download]

#### Creating a Template

You may either manually create your VM or you can utilize other alternatives to create your VM.

#### Manual Creation

1. Manually create your VM by following these instructions [https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-AE8AFBF1-75D1-4172-988C-378C35C9FAF2.html] from VMware. Once you have a VM running, you can manually install the dependencies listed above to configure the VM correctly for the vSphere node driver.
2. Customize as needed based on your specific environment and requirements.
3. Proceed with the final preparation before creating your template.

#### Alternatives to Manual Creation

Other alternative options to create VMs are listed below:

- VMware PowerCLI [https://developer.vmware.com/powercli]
- Packer [https://www.packer.io/]
- SaltStack [https://saltproject.io/]
- Ansible [https://www.ansible.com/]

Packer is a frequently-used alternative. Refer to this reference [https://github.com/vmware-samples/packer-examples-for-vsphere] for examples of its usage with vSphere.

#### Preparing Your VM

After creating a VM with all the required dependencies (and any additional required items), you must perform the most critical step next: preparing the VM to be turned into a template. This preparation will reset critical data such as the VM hostname, IPs, etc., to prevent that information from being brought into a new VM. If you fail to perform this step, you could create a VM with the same hostname, IP address, etc.

Note that these preparatory steps differ between Linux and Windows.

#### Linux Preparation

The commands below will reset your VM in Linux:

```
# Cleaning logs.
if [ -f /var/log/audit/audit.log ]; then
  cat /dev/null > /var/log/audit/audit.log
fi
if [ -f /var/log/wtmp ]; then
  cat /dev/null > /var/log/wtmp
fi
if [ -f /var/log/lastlog ]; then
  cat /dev/null > /var/log/lastlog
fi

# Cleaning udev rules.
if [ -f /etc/udev/rules.d/70-persistent-net.rules ]; then
  rm /etc/udev/rules.d/70-persistent-net.rules
fi

# Cleaning the /tmp directories
rm -rf /tmp/*
rm -rf /var/tmp/*

# Cleaning the SSH host keys
rm -f /etc/ssh/ssh_host_*

# Cleaning the machine-id
truncate -s 0 /etc/machine-id
rm /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id

# Cleaning the shell history
unset HISTFILE
history -cw
```

```
echo > ~/.bash_history
rm -fr /root/.bash_history

# Truncating hostname, hosts, resolv.conf and setting hostname to
localhost
truncate -s 0 /etc/{hostname,hosts,resolv.conf}
hostnamectl set-hostname localhost

# Clean cloud-init
cloud-init clean -s -l
```

#### Windows Preparation

Windows has a utility called sysprep [https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/sysprep--generalize--a-windows-installation?view=windows-11] that is used to generalize an image and reset the same items listed above for Linux. The command is as follows:

```
sysprep.exe /generalize /shutdown /oobe
```

#### Converting to a Template

1. Shut down and stop the VM.
2. Right-click on the VM in the inventory list and select **Template**.
3. Click on **Convert to Template**.

**Result:** Once the process has completed, a template will be available for use.

For additional information on converting a VM to a template, see the VMware guide [https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-5B3737CC-28DB-4334-BD18-6E12011CDC9F.html].

#### Moving to a Content library

Rancher has the ability to use templates provided by a content library. Content libraries store and manage content within vSphere, and they also offer the ability to publish and share that content.

Below are some helpful links on content libraries:

- Create a content library [https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-2A0F1C13-7336-45CE-B211-610D39A6E1F4.html]
- Clone the template to the content library [https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-AC1545F0-F8BA-4CD2-96EB-21B3DFAA1DC1.html]

#### Other Resources

Here is a list of additional resources that may be useful:

- Tutorial for creating a Linux template [https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/manage/hybrid/server/best-practices/vmware-ubuntu-template]

- Tutorial for creating a Windows template [https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/manage/hybrid/server/best-practices/vmware-windows-template]

### Provisioning Kubernetes Clusters in VMware vSphere

In this section, you'll learn how to use Rancher to deploy an RKE2 [https://documentation.suse.com/cloudnative/rke2/] or K3s [https://documentation.suse.com/cloudnative/k3s/] Kubernetes cluster in VMware vSphere.

First, you set up your vSphere cloud credentials in Rancher.

Then you creates a vSphere cluster in Rancher, and when configuring the new cluster, you define machine pools for it. Each machine pool has a Kubernetes role of etcd, controlplane, or worker. Rancher installs Kubernetes on the new nodes, and it sets up each node with the Kubernetes role defined by the machine pool.

- Preparation in vSphere
- Creating a vSphere Cluster

#### Preparation in VMware vSphere

This section describes the requirements for setting up vSphere so that Rancher can provision VMs and clusters.

##### Create Credentials in VMware vSphere

Before proceeding to create a cluster, you must ensure that you have a vSphere user with sufficient permissions.

Refer to this how-to guide for instructions on how to create a user in vSphere with the required permissions. These steps result in a username and password that you will need to provide to Rancher, which allows Rancher to provision resources in vSphere.

##### Network Permissions

It must be ensured that the hosts running the Rancher server are able to establish the following network connections:

- To the vSphere API on the vCenter server (usually port 443/TCP).
- To the Host API (port 443/TCP) on all ESXi hosts used to instantiate virtual machines for the clusters (*only required when using the ISO creation method*).
- To port 22/TCP and 2376/TCP on the created VMs

See Node Networking Requirements for a detailed list of port requirements applicable for creating nodes on an infrastructure provider.

##### Valid ESXi License for VMware vSphere API Access

The free ESXi license does not support API access. The vSphere servers must have a valid or evaluation ESXi license.

##### VM-VM Affinity Rules for Clusters with DRS

If you have a cluster with DRS enabled, setting up VM-VM Affinity Rules [https://docs.vmware.com/en/VMware-

vSphere/6.5/com.vmware.vsphere.resmgmt.doc/GUID-7297C302-378F-4AF2-9BD6-6EDB1E0A850A.html] is recommended. These rules allow VMs assigned the etcd and control-plane roles to operate on separate ESXi hosts when they are assigned to different node pools. This practice ensures that the failure of a single physical machine does not affect the availability of those planes.

#### Creating a VMware vSphere Cluster

##### 1. Create your cloud credentials

1. Click ☰ > **Cluster Management**.
2. Click **Cloud Credentials**.
3. Click **Create**.
4. Click **VMware vSphere**.
5. Enter your vSphere credentials. For help, refer to **Account Access** in the node template configuration reference.
6. Click **Create**.

**Result:** You have created the cloud credentials that are used to provision nodes in your cluster.

##### 2. Create your cluster

Use Rancher to create a Kubernetes cluster in vSphere.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **VMware vSphere**.
4. Enter a **Cluster Name** and use your vSphere cloud credentials.
5. Create a machine pool for each Kubernetes role. Refer to the best practices for recommendations on role assignments and counts.
    a. For each machine pool, define the machine configuration.
6. Use the **Cluster Configuration** to choose the version of Kubernetes that to install, what network provider to use and if you want to enable project network isolation. For help configuring the cluster, refer to the RKE2 and K3s cluster configuration references.
7. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
8. If you want to dynamically provision persistent storage or other infrastructure later, you will need to enable the vSphere cloud provider by modifying the cluster YAML file. For details, refer to in-tree vSphere cloud provider docs and out-of-tree vSphere cloud provider docs.
9. Review your options to confirm they're correct. Then click **Create**.

**Result:**

Your cluster is created and assigned a state of **Provisioning**. Rancher is standing up your cluster.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

#### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.
- **Provision Storage:** For an example of how to provision storage in vSphere using Rancher, refer to this section. In order to dynamically provision storage in vSphere, the vSphere provider must be enabled. For details, refer to in-tree vSphere cloud provider docs and out-of-tree vSphere cloud provider docs.

### VMware vSphere Node Template Configuration

#### Account Access

| Parameter | Required | Description |
|---|---|---|
| Cloud Credentials | * | Your vSphere account access information, stored in a cloud credential. |

Your cloud credential has these fields:

| Credential Field | Description |
|---|---|
| vCenter or ESXi Server | Enter the vCenter or ESXi hostname/IP. ESXi is the virtualization platform where you create and run virtual machines and virtual appliances. vCenter Server is the service through which you manage multiple hosts connected in a network and pool host resources. |
| Port | Configure the port of the vCenter or ESXi server. |
| Username and password | Enter your vSphere login username and password. |

#### Scheduling

Choose what hypervisor the virtual machine will be scheduled to.

The fields in the **Scheduling** section should auto-populate with the data center and other scheduling options that are available to you in vSphere.

| Field | Required | Explanation |
|---|---|---|
| Data Center | * | Choose the name/path of the data center where the VM will be scheduled. |

| Field | Required | Explanation |
|---|---|---|
| Resource Pool | | Name of the resource pool to schedule the VMs in. Resource pools can be used to partition available CPU and memory resources of a standalone host or cluster, and they can also be nested. Leave blank for standalone ESXi. If not specified, the default resource pool is used. |
| Data Store | * | If you have a data store cluster, you can toggle the **Data Store** field. This lets you select a data store cluster where your VM will be scheduled to. If the field is not toggled, you can select an individual disk. |
| Folder | | Name of a folder in the datacenter to create the VMs in. Must already exist. The VM folders in this dropdown menu directly correspond to your VM folders in vSphere. The folder name should be prefaced with `vm/` in your vSphere config file. |
| Host | | The IP of the host system to schedule VMs in. Leave this field blank for a standalone ESXi or for a cluster with DRS (Distributed Resource Scheduler). If specified, the host system's pool will be used and the **Resource Pool** parameter will be ignored. |
| Graceful Shutdown Timeout | | The amount of time, in seconds, that Rancher waits before deleting virtual machines on a cluster. If set to `0`, graceful shutdown is disabled. Only accepts integer values. |

#### Instance Options

In the **Instance Options** section, configure the number of vCPUs, memory, and disk size for the VMs created by this template.

| Parameter | Required | Description |
|---|---|---|
| CPUs | * | Number of vCPUS to assign to VMs. |
| Memory | * | Amount of memory to assign to VMs. |
| Disk | * | Size of the disk (in MB) to attach to the VMs. |
| Creation method | * | The method for setting up an operating system on the node. The operating system can be installed from an ISO or from a VM template. Depending on the creation method, you will also have to specify a VM template, content library, existing VM, or ISO. For more information on creation methods, refer to About VM Creation Methods. |

| Parameter | Required | Description |
|---|---|---|
| Cloud Init | | URL of a `cloud-config.yml` file or URL to provision VMs with. This file allows further customization of the operating system, such as network configuration, DNS servers, or system daemons. The operating system must support `cloud-init`. |
| Networks | | Name(s) of the network to attach the VM to. |
| Configuration Parameters used for guestinfo | | Additional configuration parameters for the VMs. These correspond to the Advanced Settings [https://kb.vmware.com/s/article/1016098] in the vSphere console. Example use cases include providing RancherOS guestinfo [https://rancher.com/docs/os/v1.x/en/installation/cloud/vmware-esxi/#vmware-guestinfo] parameters or enabling disk UUIDs for the VMs (`disk.EnableUUID=TRUE`). |

#### About VM Creation Methods

In the **Creation method** field, configure the method used to provision VMs in vSphere. Available options include creating VMs that boot from a RancherOS ISO or creating VMs by cloning from an existing virtual machine or VM template [https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.vm_admin.doc/GUID-F7BF0E6B-7C4F-4E46-8BBF-76229AEA7220.html].

The existing VM or template may use any modern Linux operating system that is configured with support for cloud-init [https://cloudinit.readthedocs.io/en/latest/] using the NoCloud datasource [https://canonical-cloud-init.readthedocs-hosted.com/en/latest/reference/datasources/nocloud.html].

Choose the way that the VM will be created:

- **Deploy from template: Data Center:** Choose a VM template that exists in the data center that you selected.
- **Deploy from template: Content Library:** First, select the Content Library [https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.vm_admin.doc/GUID-254B2CE8-20A8-43F0-90E8-3F6776C2C896.html] that contains your template, then select the template from the populated list **Library templates**.
- **Clone an existing virtual machine:** In the **Virtual machine** field, choose an existing VM that the new VM will be cloned from.
- **Install from boot2docker ISO:** Ensure that the **OS ISO URL** field contains the URL of a VMware ISO release for RancherOS (`rancheros-vmware.iso`). Note that this URL must be accessible from the nodes running your Rancher server installation.

#### Networks

The node template now allows a VM to be provisioned with multiple networks. In the **Networks** field, you can now click **Add Network** to add any networks available to you in vSphere.

#### Node Tags and Custom Attributes

Tags allow you to attach metadata to objects in the vSphere inventory to make it easier to sort and search for these objects.

For tags, all your vSphere tags will show up as options to select from in your node template.

In the custom attributes, Rancher will let you select all the custom attributes you have already set up in vSphere. The custom attributes are keys and you can enter values for each one.



Custom attributes are a legacy feature that will eventually be removed from vSphere.

#### cloud-init

Cloud-init [https://cloudinit.readthedocs.io/en/latest/] allows you to initialize your nodes by applying configuration on the first boot. This may involve things such as creating users, authorizing SSH keys or setting up the network.

To make use of cloud-init initialization, create a cloud config file using valid YAML syntax and paste the file content in the **Cloud Init** field. Refer to the cloud-init documentation. [https://cloudinit.readthedocs.io/en/latest/topics/examples.html] for a commented set of examples of supported cloud config directives.

Note that cloud-init is not supported when using the ISO creation method.

#### Engine Options

In the **Engine Options** section of the node template, you can configure the container daemon. You may want to specify the container version or a container image registry mirror.



If you're provisioning Red Hat Enterprise Linux (RHEL) or CentOS nodes, leave the **Docker Install URL** field as the default value, or select **none**. This will bypass a check for Docker installation, as Docker is already installed on these node types.

If you set **Docker Install URL** to a value other than the default or **none**, you might see an error message such as the following: `Error creating machine: RHEL ssh command error: command: sudo -E yum install -y curl err: exit status 1 output: Updating Subscription Management repositories.`

### Graceful Shutdown for VMware vSphere Virtual Machines

In Rancher v2.8.3 and later, you can configure the graceful shutdown of virtual machines (VMs) for VMware vSphere node driver clusters. Graceful shutdown introduces a delay before the VM is forcibly deleted, which allows time for terminating any running processes and open connections.

In RKE2/K3s, you can set up graceful shutdown when you create the cluster, or edit the cluster configuration to add it afterward.



Since Rancher can't detect the platform of an imported cluster, you cannot enable graceful shutdown on VMware vSphere clusters you have imported.

#### Enable Graceful Shutdown During VMware vSphere Cluster Creation

In RKE2/K3s, you can configure new VMware vSphere clusters with graceful shutdown for VMs:

1. Click ☰ > **Cluster Management**.
2. Click **Create** and select **VMware vSphere** to provision a new cluster.
3. Under **Machine Pools** › **Scheduling**, in the **Graceful Shutdown Timeout** field, enter an integer value greater than 0. The value you enter is the amount of time in seconds Rancher waits before deleting VMs on the cluster. If the value is set to 0, graceful shutdown is disabled.

#### Enable Graceful Shutdown in Existing SUSE® Rancher Prime: RKE2/SUSE® Rancher Prime: K3s Clusters

In RKE2/K3s, you can edit the configuration of an existing VMware vSphere cluster to enable graceful shutdown, which adds a delay before deleting VMs.

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, find the VMware vSphere hosted cluster you want to edit. Click ☰ at the end of the row associated with the cluster. Select **Edit Config**.
3. Under **Machine Pools** › **Scheduling**, in the **Graceful Shutdown Timeout** field, enter an integer value greater than 0. The value you enter is the amount of time in seconds Rancher waits before deleting VMs on the cluster. If the value is set to 0, graceful shutdown is disabled.

# Launching Kubernetes on Existing Custom Nodes

When you create a custom cluster, Rancher can use RKE2/K3s to create a Kubernetes cluster in on-prem bare-metal servers, on-prem virtual machines, or in any node hosted by an infrastructure provider.

To use this option, you need access to the servers that will be part of your Kubernetes cluster. Provision each server according to the requirements. Then, run the command provided in the Rancher UI on each server to convert it into a Kubernetes node.

This section describes how to set up a custom cluster.

## Creating a Cluster with Custom Nodes



*Want to use Windows hosts as Kubernetes workers?*

See Configuring Custom Clusters for Windows before you start.

#### 1. Provision a Linux Host

Begin creation of a custom cluster by provisioning a Linux host. Your host can be:

- A cloud-host virtual machine (VM)
- An on-prem VM
- A bare-metal server

If you want to reuse a node from a previous custom cluster, clean the node before using it in a cluster again. If you reuse a node that hasn't been cleaned, cluster provisioning may fail.

Provision the host according to the installation requirements and the checklist for production-ready clusters.



*IPv6-only cluster*

For an IPv6-only cluster, ensure that your operating system correctly configures the

`/etc/hosts` file.

```
::1       localhost
```

**2. Create the Custom Cluster**

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **Custom**.
4. Enter a **Cluster Name**.
5. Use the **Cluster Configuration** section to set up the cluster. For more information, see RKE2 Cluster Configuration Reference and K3s Cluster Configuration Reference.



*Windows nodes*

To learn more about using Windows nodes as Kubernetes workers, see Launching Kubernetes on Windows Clusters.

6. Click **Create**.

   **Result:** The UI redirects to the **Registration** page, where you can generate the registration command for your nodes.

7. From **Node Role**, select the roles you want a cluster node to fill. You must provision at least one node for each role: etcd, worker, and control plane. A custom cluster requires all three roles to finish provisioning. For more information on roles, see Roles for Nodes in Kubernetes Clusters.



*Bare-Metal Server*

If you plan to dedicate bare-metal servers to each role, you must provision a bare-metal server for each role (i.e., provision multiple bare-metal servers).

8. **Optional:** Click **Show Advanced** to configure additional settings such as specifying the IP address(es), overriding the node hostname, or adding labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] or taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to the node.



- The **Node Public IP** and **Node Private IP** fields can accept either a single address or a comma-separated list of addresses (for example: `10.0.0.5,2001:db8::1`).



*IPv6-only or Dual-stack Cluster*

- In both IPv6-only and dual-stack clusters, you should specify the node's **IPv6 address** as the **Node Private IP**.

9. Copy the command displayed on screen to your clipboard.
10. Log in to your Linux host using your preferred shell, such as PuTTy or a remote Terminal connection. Run the command copied to your clipboard.



Repeat steps 7-10 if you want to dedicate specific hosts to specific node roles. Repeat the

steps as many times as needed.

**Result:**

The cluster is created and transitions to the **Updating** state while Rancher initializes and provisions cluster components.

You can access your cluster after its state is updated to **Active**.

**Active** clusters are assigned two Projects:

- `Default`, containing the `default` namespace
- `System`, containing the `cattle-system`, `traefik`, `kube-public`, and `kube-system` namespaces

**3. Amazon Only: Tag Resources**

If you have configured your cluster to use Amazon as **Cloud Provider**, tag your AWS resources with a cluster ID.

Amazon Documentation: Tagging Your Amazon EC2 Resources [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html]



You can use Amazon EC2 instances without configuring a cloud provider in Kubernetes. You only have to configure the cloud provider if you want to use specific Kubernetes cloud provider functionality. For more information, see Kubernetes Cloud Providers [https://github.com/kubernetes/website/blob/release-1.18/content/en/docs/concepts/cluster-administration/cloud-providers.md]

The following resources need to be tagged with a `ClusterID`:

- **Nodes**: All hosts added in Rancher.
- **Subnet**: The subnet used for your cluster
- **Security Group**: The security group used for your cluster.



Do not tag multiple security groups. Tagging multiple groups generates an error when creating Elastic Load Balancer.

The tag that should be used is:

```
Key=kubernetes.io/cluster/<CLUSTERID>, Value=owned
```

`<CLUSTERID>` can be any string you choose. However, the same string must be used on every resource you tag. Setting the tag value to `owned` informs the cluster that all resources tagged with the `<CLUSTERID>` are owned and managed by this cluster.

If you share resources between clusters, you can change the tag to:

```
Key=kubernetes.io/cluster/CLUSTERID, Value=shared
```

## Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through Rancher. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.

## Configuring Storage Classes in Azure

If you are using Azure VMs for your nodes, you can use Azure files [https://docs.microsoft.com/en-us/azure/aks/azure-files-dynamic-pv] as a StorageClass for the cluster.

In order to have the Azure platform create the required storage resources, follow these steps:

1. Configure the Azure cloud provider.
2. Configure `kubectl` to connect to your cluster.
3. Copy the `ClusterRole` and `ClusterRoleBinding` manifest for the service account:

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: system:azure-cloud-provider
rules:
- apiGroups: ['']
resources: ['secrets']
verbs:     ['get','create']
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: system:azure-cloud-provider
roleRef:
kind: ClusterRole
apiGroup: rbac.authorization.k8s.io
name: system:azure-cloud-provider
subjects:
- kind: ServiceAccount
name: persistent-volume-binder
```

```
namespace: kube-system
```

4. Create these in your cluster using one of the follow command.

```
# kubectl create -f <MANIFEST>
```

## Networking Requirements for Host Gateway (L2bridge)

This section describes how to configure custom Windows clusters that are using *Host Gateway (L2bridge)* mode.

### Disabling Private IP Address Checks

If you are using *Host Gateway (L2bridge)* mode and hosting your nodes on any of the cloud services listed below, you must disable the private IP address checks for both your Linux or Windows hosts on startup. To disable this check for each node, follow the directions provided by each service below.

| Service | Directions to disable private IP address checks |
|---|---|
| Amazon EC2 | Disabling Source/Destination Checks [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html#EIP_Disable_SrcDestCheck] |
| Google GCE | Enabling IP Forwarding for Instances [https://cloud.google.com/vpc/docs/using-routes#canipforward] (By default, a VM cannot forward a packet originated by another VM) |
| Azure VM | Enable or Disable IP Forwarding [https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-network-interface#enable-or-disable-ip-forwarding] |

### Cloud-hosted VM Routes Configuration

If you are using the **Host Gateway (L2bridge)** [https://github.com/coreos/flannel/blob/master/Documentation/backends.md#host-gw] backend of Flannel, all containers on the same node belong to a private subnet, and traffic routes from a subnet on one node to a subnet on another node through the host network.

- When worker nodes are provisioned on AWS, virtualization clusters, or bare metal servers, make sure they belong to the same layer 2 subnet. If the nodes don't belong to the same layer 2 subnet, `host-gw` networking will not work.
- When worker nodes are provisioned on GCE or Azure, they are not on the same layer 2 subnet. Nodes on GCE and Azure belong to a routable layer 3 network. Follow the instructions below to configure GCE and Azure so that the cloud network knows how to route the host subnets on each node.

To configure host subnet routing on GCE or Azure, first run the following command to find out the host subnets on each worker node:

```
kubectl get nodes -o custom-columns
=nodeName:.metadata.name,nodeIP:status.addresses[0].address,routeDestina
tion:.spec.podCIDR
```

Then follow the instructions for each cloud provider to configure routing rules for each node:

| Service | Instructions |
|---|---|
| Google GCE | For GCE, add a static route for each node: Adding a Static Route [https://cloud.google.com/vpc/docs/using-routes#addingroute]. |
| Azure VM | For Azure, create a routing table: Custom Routes: User-defined [https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview#user-defined]. |

## Launching Kubernetes on Windows Clusters

When provisioning a custom cluster Rancher uses RKE2 to install Kubernetes on your existing nodes.

In a Windows cluster provisioned with Rancher, the cluster must contain both Linux and Windows nodes. The Kubernetes controlplane can only run on Linux nodes, and the Windows nodes can only have the worker role. Windows nodes can only be used for deploying workloads.

For a summary of Kubernetes features supported in Windows, see the Kubernetes documentation on supported functionality and limitations for using Kubernetes with Windows [https://kubernetes.io/docs/setup/production-environment/windows/intro-windows-in-kubernetes/#supported-functionality-and-limitations] or the guide for scheduling Windows containers in Kubernetes [https://kubernetes.io/docs/setup/production-environment/windows/user-guide-windows-containers/].

### SUSE® Rancher Prime: RKE2 Features for Windows Clusters

Listed below are the primary RKE2 features for Windows cluster provisioning:

- Windows Containers with RKE2 powered by containerd
- Added provisioning of Windows RKE2 custom clusters directly from the Rancher UI
- Calico and Flannel CNI for Windows RKE2 custom clusters
- SAC releases of Windows Server (2004 and 20H2) are included in the technical preview



Rancher will allow Windows workload pods to deploy on both Windows and Linux worker nodes by default. When creating mixed clusters in RKE2, you must edit the `nodeSelector` in the chart to direct the pods to be placed onto a compatible Windows node. Refer to the Kubernetes documentation [https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#nodeselector] for more information on how to use `nodeSelector` to assign pods to nodes.

- HostProcess containers in Windows RKE2 are supported in Kubernetes v1.24.1 and up. See the upstream documentation [https://kubernetes.io/docs/tasks/configure-pod-container/create-hostprocess-pod/] for more information.

### General Requirements

The general networking and operating system requirements for Windows nodes are the same as for other Rancher installations.

#### OS Requirements

Our support for Windows Server and Windows containers match the Microsoft official lifecycle for LTSC (Long-Term Servicing Channel) and SAC (Semi-Annual Channel).

For the support lifecycle dates for Windows Server, see the Microsoft Documentation. [https://docs.microsoft.com/en-us/windows-server/get-started/windows-server-release-info]

#### Kubernetes Version

For more information regarding Kubernetes component versions, see the support matrices for RKE2 versions [https://www.suse.com/suse-rke2/support-matrix/all-supported-versions/].

#### Node Requirements

The hosts in the cluster need to have at least:

- 2 core CPUs
- 5 GB memory
- 50 GB disk space

Rancher will not provision the node if the node does not meet these requirements.

#### Networking Requirements

Before provisioning a new cluster, be sure that you have already installed Rancher on a device that accepts inbound network traffic. This is required in order for the cluster nodes to communicate with Rancher. If you have not already installed Rancher, please refer to the installation documentation before proceeding with this guide.

Rancher supports Windows using Calico and Flannel as network providers.

If you are configuring DHCP options sets for an AWS virtual private cloud, note that in the `domain-name` option field, only one domain name can be specified. According to the DHCP options documentation: [https://docs.aws.amazon.com/vpc/latest/userguide/VPC_DHCP_Options.html]



Some Linux operating systems accept multiple domain names separated by spaces. However, other Linux operating systems and Windows treat the value as a single domain, which results in unexpected behavior. If your DHCP options set is associated with a VPC that has instances with multiple operating systems, specify only one domain name.

#### Rancher on VMware vSphere with ESXi 6.7u2 and above

If you are using Rancher on VMware vSphere with ESXi 6.7u2 or later with Red Hat Enterprise Linux 8.3, CentOS 8.3, or SUSE Enterprise Linux 15 SP2 or later, it is necessary to disable the `vmxnet3` virtual network adapter hardware offloading feature. Failure to do so will result in all network connections between pods on different cluster nodes to fail with timeout errors. All connections from Windows pods to critical services running on Linux nodes, such as CoreDNS, will fail as well. It is also possible that external connections may fail. This issue is the result of Linux distributions enabling the hardware offloading feature in `vmxnet3` and a bug in the `vmxnet3` hardware offloading feature that results in the discarding of packets for guest overlay traffic. To address this issue, it is necessary disable the `vmxnet3` hardware offloading feature. This setting does not survive reboot, so it is necessary to disable on every boot. The recommended course of action is to create a systemd unit file at `/etc/systemd/system/disable_hw_offloading.service`, which

disables the vmxnet3 hardware offloading feature on boot. A sample systemd unit file which disables the vmxnet3 hardware offloading feature is as follows. Note that <VM network interface> must be customized to the host vmxnet3 network interface, e.g., ens192:

```
[Unit]
Description=Disable vmxnet3 hardware offloading feature

[Service]
Type=oneshot
ExecStart=ethtool -K <VM network interface> tx-udp_tnl-segmentation off
ExecStart=ethtool -K <VM network interface> tx-udp_tnl-csum-segmentation
off
StandardOutput=journal

[Install]
WantedBy=multi-user.target
```

Then set the appropriate permissions on the systemd unit file:

```
chmod 0644 /etc/systemd/system/disable_hw_offloading.service
```

Finally, enable the systemd service:

```
systemctl enable disable_hw_offloading.service
```

#### Architecture Requirements

The Kubernetes cluster management nodes (etcd and controlplane) must be run on Linux nodes.

The worker nodes, which is where your workloads will be deployed on, will typically be Windows nodes, but there must be at least one worker node that is run on Linux in order to run the Rancher cluster agent, DNS, metrics server, and Ingress related containers.

#### Recommended Architecture

We recommend the minimum three-node architecture listed in the table below, but you can always add more Linux and Windows workers to scale up your cluster for redundancy:

| Node | Operating System | Kubernetes Cluster Role(s) | Purpose |
|---|---|---|---|
| Node 1 | Linux (Ubuntu Server 18.04 recommended) | Control plane, etcd, worker | Manage the Kubernetes cluster |
| Node 2 | Linux (Ubuntu Server 18.04 recommended) | Worker | Support the Rancher Cluster agent, Metrics server, DNS, and Ingress for the cluster |

| Node | Operating System | Kubernetes Cluster Role(s) | Purpose |
|---|---|---|---|
| Node 3 | Windows (Windows Server core version 1809 or above required, version 2022 recommended) | Worker | Run your Windows containers |

#### Container Requirements

Windows requires that containers must be built on the same Windows Server version that they are being deployed on. Therefore, containers must be built on Windows Server core version 1809 or above. If you have existing containers built for an earlier Windows Server core version, they must be re-built on Windows Server core version 1809 or above.

#### Cloud Provider Specific Requirements

If you set a Kubernetes cloud provider in your cluster, some additional steps are required. You might want to set a cloud provider if you want to want to leverage a cloud provider's capabilities, for example, to automatically provision storage, load balancers, or other infrastructure for your cluster. Refer to this page for details on how to configure a cloud provider cluster of nodes that meet the prerequisites.

If you are using the GCE (Google Compute Engine) cloud provider, you must do the following:

- Enable the GCE cloud provider in the `cluster.yml` by following these steps.
- When provisioning the cluster in Rancher, choose **Custom cloud provider** as the cloud provider in the Rancher UI.

### Tutorial: How to Create a Cluster with Windows Support

This tutorial describes how to create a Rancher-provisioned cluster with the three nodes in the recommended architecture.

To set up a cluster with support for Windows nodes and containers, you will need to complete the tasks below.

#### 1. Provision Hosts

To begin provisioning a cluster on existing nodes with Windows support, prepare your hosts.

Your hosts can be:

- Cloud-hosted VMs
- VMs from virtualization clusters
- Bare-metal servers

You will provision three nodes:

- One Linux node, which manages the Kubernetes control plane, stores your `etcd`, and optionally can be a worker node
- A second Linux node, which will be another worker node
- The Windows node, which will run your Windows containers as a worker node

| Node | Operating System |
|---|---|
| Node 1 | Linux (Ubuntu Server 18.04 recommended) |
| Node 2 | Linux (Ubuntu Server 18.04 recommended) |
| Node 3 | Windows (Windows Server core version 1809 or above required, version 2022 recommended) |

If your nodes are hosted by a **Cloud Provider** and you want automation support such as loadbalancers or persistent storage devices, your nodes have additional configuration requirements. For details, see Selecting Cloud Providers.

#### 2. Create the Cluster on Existing Nodes

The instructions for creating a Windows cluster on existing nodes are very similar to the general instructions for creating a custom cluster with some Windows-specific requirements.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, click **Create**.
3. Click **Custom**.
4. Enter a name for your cluster in the **Cluster Name** field.
5. In the **Kubernetes Version** dropdown menu, select a supported Kubernetes version.
6. In the **Container Network** field, select either **Calico** or **Flannel**.
7. Click **Create**.

#### 3. Add Nodes to the Cluster

This section describes how to register your Linux and Worker nodes to your cluster. You will run a command on each node, which will install the rancher-system-agent and allow Rancher to manage each node.

##### Add Linux Master Node

In this section, we fill out a form on the Rancher UI to get a custom command to install the Rancher agent on the Linux master node. Then we will copy the command and run it on our Linux master node to register the node in the cluster.

The first node in your cluster should be a Linux host that has both the **Control Plane** and **etcd** roles. At a minimum, both of these roles must be enabled for this node, and this node must be added to your cluster before you can add Windows hosts.

1. After cluster creation, navigate to the **Registration** tab.
2. In **Step 1** under the **Node Role** section, select all three roles. Although you can choose only the **etcd** and **Control Plane** roles, we recommend selecting all three.
3. Optional: If you click **Show Advanced**, you can configure additional settings such as specifying the IP address(es), overriding the node hostname, or adding labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] or taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to the node.
4. In **Step 2**, under the **Registration** section, copy the command displayed on the screen to your clipboard.
5. SSH into your Linux host and run the command that you copied to your clipboard.

**Result:**

Your cluster is created and assigned a state of **Updating**. Rancher is standing up your cluster.

It may take a few minutes for the node to register and appear under the **Machines** tab.

You'll be able to access the cluster once its state changes to **Active**.

#### Add Linux Worker Node

In this section, we run a command to register the Linux worker node to the cluster.

After the initial provisioning of your cluster, your cluster only has a single Linux host. Next, we add another Linux `worker` host, which will be used to support *Rancher cluster agent*, *Metrics server*, *DNS* and *Ingress* for your cluster.

1. After cluster creation, navigate to the **Registration** tab.
2. In **Step 1** under the **Node Role** section, select **Worker**.
3. Optional: If you click **Show Advanced**, you can configure additional settings such as specifying the IP address(es), overriding the node hostname, or to add labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] or taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to the node.
4. In **Step 2**, under the **Registration** section, copy the command displayed on the screen to your clipboard.
5. SSH into your Linux host and run the command that you copied to your clipboard.

**Result:**

The **Worker** role is installed on your Linux host, and the node registers with Rancher. It may take a few minutes for the node to be registered in your cluster.



Taints on Linux Worker Nodes

For each Linux worker node added into the cluster, the following taints will be added to Linux worker node. By adding this taint to the Linux worker node, any workloads added to the Windows cluster will be automatically scheduled to the Windows worker node. If you want to schedule workloads specifically onto the Linux worker node, you will need to add tolerations to those workloads.

| Taint Key | Taint Value | Taint Effect |
|---|---|---|
| `cattle.io/os` | `linux` | `NoSchedule` |

#### Add a Windows Worker Node

In this section, we run a command to register the Windows worker node to the cluster.



The registration command to add the Windows workers only appears after the cluster is running with Linux etcd, control plane, and worker nodes.

1. After cluster creation, navigate to the **Registration** tab.
2. In **Step 1** under the **Node Role** section, select **Worker**.

3. Optional: If you click **Show Advanced**, you can configure additional settings such as specifying the IP address(es), overriding the node hostname, or adding labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] or taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to the node.
4. In **Step 2**, under the **Registration** section, copy the command for Windows workers displayed on the screen to your clipboard.
5. Log in to your Windows host using your preferred tool, such as Microsoft Remote Desktop [https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients]. Run the command copied to your clipboard in the **PowerShell Console** as an Administrator.
6. Optional: Repeat these instructions if you want to add more Windows nodes to your cluster.

**Result:**

The **Worker** role is installed on your Windows host, and the node registers with Rancher. It may take a few minutes for the node to be registered in your cluster.

You now have a Windows Kubernetes cluster.

#### Optional Next Steps

After creating your cluster, you can access it through the Rancher UI. As a best practice, we recommend setting up these alternate ways of accessing your cluster:

- **Access your cluster with the kubectl CLI:** Follow these steps to access clusters with kubectl on your workstation. In this case, you will be authenticated through the Rancher server's authentication proxy, then Rancher will connect you to the downstream cluster. This method lets you manage the cluster without the Rancher UI.
- **Access your cluster with the kubectl CLI, using the authorized cluster endpoint:** Follow these steps to access your cluster with kubectl directly, without authenticating through the Rancher server. We recommend setting up this alternative method to access your cluster so that in case you can't connect to Rancher, you can still access the cluster.

### Configuration for Storage Classes in Azure

If you are using Azure VMs for your nodes, you can use Azure files [https://docs.microsoft.com/en-us/azure/aks/azure-files-dynamic-pv] as a StorageClass for the cluster. For details, refer to this section.

## Windows and Linux Cluster Feature Parity

Windows clusters do not share the same feature support as Linux clusters.

The following chart describes the feature parity between Windows and Linux on Rancher:

| Component | Linux | Windows |
|---|---|---|
| **Distributions** | | |
| RKE2 | Supported | Supported |
| K3S | Supported | Not Supported |
| EKS | Supported | Not Supported |
| GKE | Supported | Not Supported |

| Component | Linux | Windows |
|---|---|---|
| AKS | Supported | Not Supported |
| **Rancher Components** | | |
| Server | Supported | Not Supported |
| Agent | Supported | Supported |
| Fleet | Supported | Supported |
| EKS Operator | Supported | Not Supported |
| AKS Operator | Not Supported | Not Supported |
| GKE Operator | Not Supported | Not Supported |
| Monitoring/Alerting v2 | Supported | Supported |
| Logging v2 | Supported | Supported |
| Istio | Supported | Not Supported |
| Catalog v2 | Supported | Not Supported |
| OPA | Supported | Not Supported |
| Longhorn | Supported | Not Supported |
| CIS Scans | Supported | Not Supported |
| Backup/Restore Operator | Supported | Not Supported |
| **CNI / Add-ons** | | |
| Flannel | Supported | Supported |
| Canal | Supported | Not Supported |
| Calico | Supported | Supported (RKE2 Only) |
| Cilium | Supported | Not Supported |
| Multus | Supported | Not Supported |
| Traefik | Supported | Not Supported |
| NGINX Ingress | Supported | Not Supported |

For updated information on feature support, you may visit rancher/windows [https://github.com/rancher/windows] on GitHub.

# Configuration

## SUSE® Rancher Prime: K3s Cluster Configuration Reference

This section covers the configuration options that are available in Rancher for a new or existing K3s Kubernetes cluster.

### Overview

You can configure the Kubernetes options one of two ways:

- Rancher UI: Use the Rancher UI to select options that are commonly customized when setting up a Kubernetes cluster.
- Cluster Config File: Instead of using the Rancher UI to choose Kubernetes options for the cluster, advanced users can create a K3s config file. Using a config file lets you set any of the options

[https://rancher.com/docs/k3s/latest/en/installation/install-options/] available during a K3s installation.

### Editing Clusters in the Rancher UI

The Rancher UI provides two ways to edit a cluster:

1. With a form.
2. With YAML.

#### Editing Clusters with a Form

The form covers the most frequently needed options for clusters.

To edit your cluster,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster you want to configure and click ☰ > **Edit Config**.

#### Editing Clusters in YAML

For a complete reference of configurable options for K3s clusters in YAML, see the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/configuration.html].

To edit your cluster with YAML:

1. Click ☰ > **Cluster Management**.
2. Go to the cluster you want to configure and click ☰ > **Edit as YAML**.
3. Edit the RKE options under the `rkeConfig` directive.

### Configuration Options in the Rancher UI

#### Machine Pool Configuration

This subsection covers generic machine pool configurations. For specific infrastructure provider configurations, refer to the following:

- Azure
- DigitalOcean
- Amazon EC2
- Google GCE

#### Pool Name

The name of the machine pool.

#### Machine Count

The number of machines in the pool.

#### Roles

Option to assign etcd, control plane, and worker roles to nodes.

#### Advanced

#### Auto Replace

The amount of time nodes can be unreachable before they are automatically deleted and replaced.

#### Drain Before Delete

Enables draining nodes by evicting all pods before the node is deleted.

#### Kubernetes Node Labels

Add labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] to nodes to help with organization and object selection.

For details on label syntax requirements, see the Kubernetes documentation. [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set]

#### Taints

Add taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to nodes, to prevent pods from being scheduled to or executed on the nodes, unless the pods have matching tolerations.

#### Cluster Configuration

#### Basics

#### Kubernetes Version

The version of Kubernetes installed on your cluster nodes.

For details on upgrading or rolling back Kubernetes, refer to Upgrading Kubernetes.

#### Pod Security Admission Configuration Template

The default pod security admission configuration template for the cluster.

#### Encrypt Secrets

Option to enable or disable secrets encryption. When enabled, secrets are encrypted using a AES-CBC key. If disabled, previously encrypted secrets are not readable until encryption is enabled again. Refer to the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/security/secrets-encryption.html] for details.

#### Project Network Isolation

If your network provider allows project network isolation, you can choose whether to enable or disable inter-project communication.

#### SELinux

Option to enable or disable SELinux [https://documentation.suse.com/cloudnative/k3s/latest/en/advanced.html#_selinux_support] support.

#### CoreDNS

By default, CoreDNS [https://coredns.io/] is installed as the default DNS provider. If CoreDNS is not installed, an alternate DNS provider must be installed yourself. Refer to the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/networking-services.html#_coredns] for details.

#### Klipper Service LB

Option to enable or disable the Klipper [https://github.com/rancher/klipper-lb] service load balancer. Refer to the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/networking-services.html#_service_load_balancer] for details.

#### Traefik Ingress

Option to enable or disable the Traefik [https://traefik.io/] HTTP reverse proxy and load balancer. For more details and configuration options, see the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/networking-services.html#_traefik_ingress_controller].

#### Local Storage

Option to enable or disable local storage [https://documentation.suse.com/cloudnative/k3s/latest/en/storage.html] on the node(s).

#### Metrics Server

Option to enable or disable the metrics server [https://github.com/kubernetes-incubator/metrics-server]. If enabled, ensure port 10250 is opened for inbound TCP traffic.

#### Add-On Config

Additional Kubernetes manifests, managed as a Add-on [https://kubernetes.io/docs/concepts/cluster-administration/addons/], to apply to the cluster on startup. Refer to the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/packaged-components.html#_auto_deploying_manifests_addons] for details.

#### Agent Environment Vars

Option to set environment variables for K3s agents [https://documentation.suse.com/cloudnative/k3s/latest/en/architecture.html]. The environment variables can be set using key value pairs. Refer to the K3 documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/configuration.html#_configuration_file] for more details.

#### etcd

#### Automatic Snapshots

Option to enable or disable recurring etcd snapshots. If enabled, users have the option to configure the frequency of snapshots. For details, refer to the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/cli/etcd-snapshot.html#_creating_snapshots].

#### Metrics

Option to choose whether to expose etcd metrics to the public or only within the cluster.

#### Networking

#### Cluster CIDR

IPv4/IPv6 network CIDRs to use for pod IPs (default: `10.42.0.0/16`).

Example values:

- IPv4-only: `10.42.0.0/16`
- IPv6-only: `2001:cafe:42::/56`
- Dual-stack: `10.42.0.0/16,2001:cafe:42::/56`

For additional requirements and limitations related to dual-stack or IPv6-only networking, see the following resources:

- K3s documentation: Dual-stack (IPv4 + IPv6) Networking [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/basic-network-options.html#_dual_stack_ipv4_ipv6_networking]
- K3s documentation: Single-stack IPv6 Networking [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/basic-network-options.html#_single_stack_ipv6_networking]



You must configure the Service CIDR when you first create the cluster. You cannot enable the Service CIDR on an existing cluster after it starts.

#### Service CIDR

IPv4/IPv6 network CIDRs to use for service IPs (default: `10.43.0.0/16`).

Example values:

- IPv4-only: `10.43.0.0/16`
- IPv6-only: `2001:cafe:43::/112`
- Dual-stack: `10.43.0.0/16,2001:cafe:43::/112`

For additional requirements and limitations related to dual-stack or IPv6-only networking, see the following resources:

- K3s documentation: Dual-stack (IPv4 + IPv6) Networking [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/basic-network-options.html#_dual_stack_ipv4_ipv6_networking]
- K3s documentation: Single-stack IPv6 Networking [https://documentation.suse.com/cloudnative/k3s/latest/en/networking/basic-network-options.html#_single_stack_ipv6_networking]



You must configure the Service CIDR when you first create the cluster. You cannot enable the Service CIDR on an existing cluster after it starts.

#### Cluster DNS

IPv4 Cluster IP for coredns service. Should be in your service-cidr range (default: `10.43.0.10`).

#### Cluster Domain

Select the domain for the cluster. The default is `cluster.local`.

#### NodePort Service Port Range

Option to change the range of ports that can be used for NodePort services [https://kubernetes.io/docs/concepts/services-networking/service/#nodeport]. The default is `30000-32767`.

#### Truncate Hostnames

Option to truncate hostnames to 15 characters or fewer. You can only set this field during the initial creation of the cluster. You can't enable or disable the 15-character limit after cluster creation.

This setting only affects machine-provisioned clusters. Since custom clusters set hostnames during their own node creation process, which occurs outside of Rancher, this field doesn't restrict custom cluster hostname length.

Truncating hostnames in a cluster improves compatibility with Windows-based systems. Although Kubernetes allows hostnames up to 63 characters in length, systems that use NetBIOS restrict hostnames to 15 characters or fewer.

#### TLS Alternate Names

Add hostnames or IPv4/IPv6 addresses as Subject Alternative Names on the server TLS cert.

#### Stack Preference

Choose the networking stack for the cluster. This option affects:

- The address used for health and readiness probes of components such as Calico, etcd, kube-apiserver, kube-scheduler, kube-controller-manager, and kubelet.
- The server URL in the `authentication-token-webhook-config-file` for the Authorized Cluster Endpoint.
- The `advertise-client-urls` setting for etcd during snapshot restoration.

Options are `ipv4`, `ipv6`, `dual`:

- When set to `ipv4`, the cluster uses `127.0.0.1`
- When set to `ipv6`, the cluster uses `[::1]`
- When set to `dual`, the cluster uses `localhost`

The stack preference must match the cluster's networking configuration:

- Set to `ipv4` for IPv4-only clusters
- Set to `ipv6` for IPv6-only clusters
- Set to `dual` for dual-stack clusters



Ensuring the loopback address configuration is correct is critical for successful cluster provisioning. For more information, refer to the Node Requirements page.

#### Authorized Cluster Endpoint

Authorized Cluster Endpoint can be used to directly access the Kubernetes API server, without requiring communication through Rancher.

For more detail on how an authorized cluster endpoint works and why it is used, refer to the architecture section.

We recommend using a load balancer with the authorized cluster endpoint. For details, refer to the recommended architecture section.

#### Registries

Select the image repository to pull Rancher images from. For more details and configuration options, see the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/private-registry.html].

#### Upgrade Strategy

#### Control Plane Concurrency

Select how many nodes can be upgraded at the same time. Can be a fixed number or percentage.

#### Worker Concurrency

Select how many nodes can be upgraded at the same time. Can be a fixed number or percentage.

#### Drain Nodes (Control Plane)

Option to remove all pods from the node prior to upgrading.

#### Drain Nodes (Worker Nodes)

Option to remove all pods from the node prior to upgrading.

#### Advanced

Option to set kubelet options for different nodes. For available options, refer to the Kubernetes documentation [https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/].

### Cluster Config File Reference

Editing clusters in YAML allows you to set configurations that are already listed in Configuration Options in the Rancher UI, as well as set Rancher-specific parameters.

*Click to Expand* **Example Cluster Config File Snippet**

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  cloudCredentialSecretName: cattle-global-data:cc-fllv6
  clusterAgentDeploymentCustomization: {}
  fleetAgentDeploymentCustomization: {}
  kubernetesVersion: v1.26.7+k3s1
  localClusterAuthEndpoint: {}
```

```
  rkeConfig:
    additionalManifest: ""
    chartValues: {}
    etcd:
      snapshotRetention: 5
      snapshotScheduleCron: 0 */5 * * *
    machineGlobalConfig:
      disable-apiserver: false
      disable-cloud-controller: false
      disable-controller-manager: false
      disable-etcd: false
      disable-kube-proxy: false
      disable-network-policy: false
      disable-scheduler: false
      etcd-expose-metrics: false
      kube-apiserver-arg:
        - audit-policy-file=/etc/rancher/k3s/user-audit-policy.yaml
        - audit-log-path=/etc/rancher/k3s/user-audit.logs
      profile: null
      secrets-encryption: false
    machinePools:
    - controlPlaneRole: true
      etcdRole: true
      machineConfigRef:
        kind: Amazonec2Config
        name: nc-test-pooll-pwl5h
      name: pooll
      quantity: 1
      unhealthyNodeTimeout: 0s
      workerRole: true
    machineSelectorConfig:
      - config:
          docker: false
          protect-kernel-defaults: false
          selinux: false
    machineSelectorFiles:
      - fileSources:
          - configMap:
              name: ''
            secret:
              name: audit-policy
              items:
                - key: audit-policy
                  path: /etc/rancher/k3s/user-audit-policy.yaml
        machineLabelSelector:
```

```
        matchLabels:
          rke.cattle.io/control-plane-role: 'true'
    registries: {}
    upgradeStrategy:
      controlPlaneConcurrency: '1'
      controlPlaneDrainOptions:
        deleteEmptyDirData: true
        disableEviction: false
        enabled: false
        force: false
        gracePeriod: -1
        ignoreDaemonSets: true
        ignoreErrors: false
        postDrainHooks: null
        preDrainHooks: null
        skipWaitForDeleteTimeoutSeconds: 0
        timeout: 120
      workerConcurrency: '1'
      workerDrainOptions:
        deleteEmptyDirData: true
        disableEviction: false
        enabled: false
        force: false
        gracePeriod: -1
        ignoreDaemonSets: true
        ignoreErrors: false
        postDrainHooks: null
        preDrainHooks: null
        skipWaitForDeleteTimeoutSeconds: 0
        timeout: 120
```

**additionalManifest**

Specify additional manifests to deliver to the control plane nodes.

The value is a String, and will be placed at the path `/var/lib/rancher/k3s/server/manifests/rancher/addons.yaml` on target nodes.

Example:

```
additionalManifest: |-
  apiVersion: v1
  kind: Namespace
  metadata:
    name: name-xxxx
```



If you want to customize system charts, you should use the `chartValues` field as described below.

Alternatives, such as using a HelmChartConfig to customize the system charts via `additionalManifest`, can cause unexpected behavior, due to having multiple HelmChartConfigs for the same chart.

**chartValues**

Specify the values for the system charts installed by K3s.

For more information about how K3s manages packaged components, please refer to K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/installation/packaged-components.html].

Example:

```
chartValues:
    chart-name:
        key: value
```

**machineGlobalConfig**

Specify K3s configurations. Any configuration change made here will apply to every node. The configuration options available in the standalone version of k3s [https://documentation.suse.com/cloudnative/k3s/latest/en/cli/server.html] can be applied here.

Example:

```
machineGlobalConfig:
    etcd-arg:
        - key1=value1
        - key2=value2
```

To make it easier to put files on nodes beforehand, Rancher expects the following values to be included in the configuration, while K3s expects the values to be entered as file paths:

- private-registry
- flannel-conf

Rancher delivers the files to the path `/var/lib/rancher/k3s/etc/config-files/<option>` in target nodes, and sets the proper options in the K3s server.

Example:

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  rkeConfig:
```

```
    machineGlobalConfig:
      private-registry: |
        mirrors:
          docker.io:
            endpoint:
              - "http://mycustomreg.com:5000"
        configs:
          "mycustomreg:5000":
            auth:
              username: xxxxxx # this is the registry username
              password: xxxxxx # this is the registry password
```

#### machineSelectorConfig

`machineSelectorConfig` is the same as `machineGlobalConfig` except that a label selector can be specified with the configuration. The configuration will only be applied to nodes that match the provided label selector.

Multiple `config` entries are allowed, each specifying their own `machineLabelSelector`. A user can specify `matchExpressions`, `matchLabels`, both, or neither. Omitting the `machineLabelSelector` section of this field has the same effect as putting the config in the `machineGlobalConfig` section.

Example:

```
machineSelectorConfig:
  - config:
      config-key: config-value
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
        key1: value1
        key2: value2
```

#### machineSelectorFiles



This feature is available in Rancher v2.7.2 and later.

Deliver files to nodes, so that the files can be in place before initiating K3s server or agent processes. The content of the file is retrieved from either a secret or a configmap. The target nodes are filtered by the `machineLabelSelector`.

Example :

```
machineSelectorFiles:
  - fileSources:
      - secret:
          items:
            - key: example-key
              path: path-to-put-the-file-on-nodes
              permissions: 644 (optional)
              hash: base64-encoded-hash-of-the-content (optional)
          name: example-secret-name
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
        key1: value1
        key2: value2
  - fileSources:
      - configMap:
          items:
            - key: example-key
              path: path-to-put-the-file-on-nodes
              permissions: 644 (optional)
              hash: base64-encoded-hash-of-the-content (optional)
          name: example-configmap-name
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
        key1: value1
        key2: value2
```

The secret or configmap must meet the following requirements:

1. It must be in the `fleet-default` namespace where the Cluster object exists.

2. It must have the annotation `rke.cattle.io/object-authorized-for-clusters: cluster-name1,cluster-name2`, which permits the target clusters to use it.



Rancher Dashboard provides an easy-to-use form for creating the secret or configmap.

Example:

```
apiVersion: v1
data:
  audit-policy: >-

IyBMb2cgYWxsIHJlcXVlc3RzIGF0IHRoZSBNZXRhZGF0YSBsZXZlbC4KYXBpVmVyc2lvbjog
YXVkaXQuazhzLmlvL3YxCmtpbmQ6IFBvbGljeQpydWxlczoKLSBsZXZlbDogTWV0YWRhdGE=
kind: Secret
metadata:
  annotations:
    rke.cattle.io/object-authorized-for-clusters: cluster1
  name: name1
  namespace: fleet-default
```

## RKE2

### SUSE® Rancher Prime: RKE2 Cluster Configuration Reference

This section covers the configuration options that are available in Rancher for a new or existing RKE2 Kubernetes cluster.

#### Overview

You can configure the Kubernetes options in one of the two following ways:

- Rancher UI: Use the Rancher UI to select options that are commonly customized when setting up a Kubernetes cluster.
- Cluster Config File: Instead of using the Rancher UI to choose Kubernetes options for the cluster, advanced users can create an RKE2 config file. Using a config file allows you to set many additional options [https://documentation.suse.com/cloudnative/rke2/latest/en/install/configuration.html] available for an RKE2 installation.

#### Editing Clusters in the Rancher UI

The Rancher UI provides two ways to edit a cluster:

1. With a form.
2. With YAML.

#### Editing Clusters with a Form

The form covers the most frequently needed options for clusters.

To edit your cluster,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster you want to configure and click ☰ > **Edit Config**.

#### Editing Clusters in YAML

For a complete reference of configurable options for RKE2 clusters in YAML, see the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/configuration.html].

To edit your cluster in YAML:

1. Click ☰ > **Cluster Management**.
2. Go to the cluster you want to configure and click ☰ > **Edit as YAML**.
3. Edit the RKE options under the `rkeConfig` directive.

#### Configuration Options in the Rancher UI

##### Machine Pool Configuration

This subsection covers generic machine pool configurations. For specific infrastructure provider configurations, refer to the following:

- Azure
- DigitalOcean
- Amazon EC2
- Google GCE

##### Pool Name

The name of the machine pool.

##### Machine Count

The number of machines in the pool.

##### Roles

Option to assign etcd, control plane, and worker roles to nodes.

##### Advanced

##### Auto Replace

The amount of time nodes can be unreachable before they are automatically deleted and replaced.

##### Drain Before Delete

Enables draining nodes by evicting all pods before the node is deleted.

#### Kubernetes Node Labels

Add labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/] to nodes to help with organization and object selection.

For details on label syntax requirements, see the Kubernetes documentation. [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set]

#### Taints

Add taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/] to nodes, to prevent pods from being scheduled to or executed on the nodes, unless the pods have matching tolerations.

### Cluster Configuration

#### Basics

#### Kubernetes Version

The version of Kubernetes installed on your cluster nodes.

For details on upgrading or rolling back Kubernetes, refer to Upgrading Kubernetes.

#### Container Network Provider

The Network Provider [https://kubernetes.io/docs/concepts/cluster-administration/networking/] that the cluster uses.



After you launch the cluster, you cannot change your network provider. Therefore, choose which network provider you want to use carefully, as Kubernetes doesn't allow switching between network providers. Once a cluster is created with a network provider, changing network providers would require you to tear down the entire cluster and all its applications.

Out of the box, Rancher is compatible with the following network providers:

- Canal [https://github.com/projectcalico/canal]
- Cilium [https://cilium.io/]*
- Calico [https://docs.projectcalico.org/v3.11/introduction/]
- Flannel [https://github.com/flannel-io/flannel]
- Multus [https://github.com/k8snetworkplumbingwg/multus-cni]
- When using project network isolation in the Cilium CNI, it is possible to enable cross-node ingress routing. Click the CNI provider docs to learn more.

For more details on the different networking providers and how to configure them, please view our RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html].



When using `cilium` or `multus,cilium` as your container network interface provider, ensure the **Enable IPv6 Support** option is also enabled.

#### Cloud Provider

You can configure a Kubernetes cloud provider. If you want to use dynamically provisioned volumes and storage in Kubernetes, typically you must select the specific cloud provider in order to use it. For example, if you want to use Amazon EBS, you would need to select the `aws` cloud provider.



If the cloud provider you want to use is not listed as an option, you will need to use the config file option to configure the cloud provider. Please reference this documentation [https://rancher.com/docs/rke/latest/en/config-options/cloud-providers/] on how to configure the cloud provider.

#### Pod Security Admission Configuration Template

The default pod security admission configuration template for the cluster.

#### Worker Compliance Profile

Select a compliance benchmark to validate the system configuration against.

#### Project Network Isolation

If your network provider allows project network isolation, you can choose whether to enable or disable inter-project communication.

Project network isolation is available if you are using any RKE2 network plugin that supports the enforcement of Kubernetes network policies, such as Canal.

#### CoreDNS

By default, CoreDNS [https://coredns.io/] is installed as the default DNS provider. If CoreDNS is not installed, an alternate DNS provider must be installed yourself. Refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/networking_services.html#_coredns] for additional CoreDNS configurations.

#### Ingress

**Ingress-NGINX**



**Ingress-NGINX EOL:** The community `ingress-nginx` controller reached End-of-Life (EOL) March 2026 and is deprecated in RKE2 v1.36. SUSE Rancher Prime provisioned RKE2 versions will continue to receive `ingress-nginx` support and CVE fixes (8+) for the entire lifecycle of RKE2 v1.32, v1.33, v1.34, v1.35 and v1.36. Traefik is the recommended migration path for SUSE Rancher RKE2 environments, it is recommended to migrate as soon as possible. Refer to Migrating from Ingress NGINX to Traefik in a Rancher provisioned RKE2 Cluster [https://support.scc.suse.com/s/kb/Migrating-from-Ingress-NGINX-to-Traefik-in-a-Rancher-provisioned-RKE2-Cluster?language=en_US] for further details.

This is the legacy ingress option to enable Ingress-NGINX within the cluster. Refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/networking_services.html#_ingress_controller] for additional configuration options.

**Traefik**

If you want to expose your applications in a high-availability configuration, and you're hosting your nodes with a cloud provider that doesn't have a native load-balancing feature, enable this option to use Traefik Ingress within the cluster. Refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/networking_services.html#_ingress_controller] for additional configuration options.

#### Metrics Server

Option to enable or disable Metrics Server [https://rancher.com/docs/rke/latest/en/config-options/add-ons/metrics-server/].

Each cloud provider capable of launching a cluster using RKE2 can collect metrics and monitor for your cluster nodes. Enable this option to view your node metrics from your cloud provider's portal.

#### Add-On Config

Additional Kubernetes manifests, managed as an Add-on [https://kubernetes.io/docs/concepts/cluster-administration/addons/], to apply to the cluster on startup. Refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/add-ons/helm.html#_automatically_deploying_manifests_and_helm_charts] for details.

#### Agent Environment Vars

Option to set environment variables for Rancher agents. The environment variables can be set using key value pairs. Refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/reference/linux_agent_config.html] for more details.

#### etcd

#### Automatic Snapshots

Option to enable or disable recurring etcd snapshots. If enabled, users have the option to configure the frequency of snapshots. For details, refer to the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/backup_restore.html#_creating_snapshots]. Note that with RKE2, snapshots are stored on each etcd node. This varies from RKE1 which only stores one snapshot per cluster.

#### Metrics

Option to choose whether to expose etcd metrics to the public or only within the cluster.

#### Networking

#### Cluster CIDR

IPv4 and/or IPv6 network CIDRs to use for pod IPs (default: `10.42.0.0/16`).

Example values:

- IPv4-only: `10.42.0.0/16`
- IPv6-only: `2001:cafe:42::/56`
- Dual-stack: `10.42.0.0/16,2001:cafe:42::/56`

For additional requirements and limitations related to dual-stack or IPv6-only networking, see the following resources:

- RKE2 documentation: Dual-stack configuration [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html#_dual_stack_configuration]
- RKE2 documentation: IPv6-only setup [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html#_ipv6_setup]



- You must configure the Service CIDR when you first create the cluster. You cannot enable the Service CIDR on an existing cluster after it starts.
- When using `cilium` or `multus,cilium` as your container network interface provider, ensure the **Enable IPv6 Support** option is also enabled.

#### Service CIDR

IPv4/IPv6 network CIDRs to use for service IPs (default: `10.43.0.0/16`).

Example values:

- IPv4-only: `10.43.0.0/16`
- IPv6-only: `2001:cafe:43::/112`
- Dual-stack: `10.43.0.0/16,2001:cafe:43::/112`

For additional requirements and limitations related to dual-stack or IPv6-only networking, see the following resources:

- RKE2 documentation: Dual-stack configuration [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html#_dual_stack_configuration]
- RKE2 documentation: IPv6-only setup [https://documentation.suse.com/cloudnative/rke2/latest/en/networking/basic_network_options.html#_ipv6_setup]



- You must configure the Service CIDR when you first create the cluster. You cannot enable the Service CIDR on an existing cluster after it starts.
- When using `cilium` or `multus,cilium` as your container network interface provider, ensure the **Enable IPv6 Support** option is also enabled.

#### Cluster DNS

IPv4 Cluster IP for coredns service. Should be in your service-cidr range (default: `10.43.0.10`).

#### Cluster Domain

Select the domain for the cluster. The default is `cluster.local`.

#### NodePort Service Port Range

Option to change the range of ports that can be used for NodePort services [https://kubernetes.io/docs/concepts/services-networking/service/#nodeport]. The default is `30000-32767`.

#### Truncate Hostnames

Option to truncate hostnames to 15 characters or fewer. You can only set this field during the initial creation of the cluster. You can't enable or disable the 15-character limit after cluster creation.

This setting only affects machine-provisioned clusters. Since custom clusters set hostnames during their own node creation process, which occurs outside of Rancher, this field doesn't restrict custom cluster hostname length.

Truncating hostnames in a cluster improves compatibility with Windows-based systems. Although Kubernetes allows hostnames up to 63 characters in length, systems that use NetBIOS restrict hostnames to 15 characters or fewer.

#### TLS Alternate Names

Add hostnames or IPv4/IPv6 addresses as Subject Alternative Names on the server TLS cert.

#### Stack Preference

Choose the networking stack for the cluster. This option affects:

- The address used for health and readiness probes of components such as Calico, etcd, kube-apiserver, kube-scheduler, kube-controller-manager, and kubelet.
- The server URL in the `authentication-token-webhook-config-file` for the Authorized Cluster Endpoint.
- The `advertise-client-urls` setting for etcd during snapshot restoration.

Options are `ipv4`, `ipv6`, `dual`:

- When set to `ipv4`, the cluster uses `127.0.0.1`
- When set to `ipv6`, the cluster uses `[::1]`
- When set to `dual`, the cluster uses `localhost`

The stack preference must match the cluster's networking configuration:

- Set to `ipv4` for IPv4-only clusters
- Set to `ipv6` for IPv6-only clusters
- Set to `dual` for dual-stack clusters



Ensuring the loopback address configuration is correct is critical for successful cluster provisioning. For more information, refer to the Node Requirements page.

#### Authorized Cluster Endpoint

Authorized Cluster Endpoint can be used to directly access the Kubernetes API server, without requiring communication through Rancher.

This is enabled by default in Rancher-launched Kubernetes clusters, using the IP of the node with the `controlplane` role and the default Kubernetes self signed certificates.

For more detail on how an authorized cluster endpoint works and why it is used, refer to the architecture

section.

We recommend using a load balancer with the authorized cluster endpoint. For details, refer to the recommended architecture section.

#### Registries

Select the image repository to pull Rancher images from. For more details and configuration options, see the RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/containerd_registry_configuration.html].

#### Upgrade Strategy

#### Control Plane Concurrency

Select how many nodes can be upgraded at the same time. Can be a fixed number or percentage.

#### Worker Concurrency

Select how many nodes can be upgraded at the same time. Can be a fixed number or percentage.

#### Drain Nodes (Control Plane)

Option to remove all pods from the node prior to upgrading.

#### Drain Nodes (Worker Nodes)

Option to remove all pods from the node prior to upgrading.

#### Advanced

Option to set kubelet options for different nodes. For available options, refer to the Kubernetes documentation [https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/].

#### Cluster Config File Reference

Editing clusters in YAML allows you to set the options available [https://documentation.suse.com/cloudnative/rke2/latest/en/install/configuration.html] in an RKE2 installation, including those already listed in Configuration Options in the Rancher UI, as well as set Rancher-specific parameters.

*Example Cluster Config File Snippet*

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  cloudCredentialSecretName: cattle-global-data:cc-s879v
  kubernetesVersion: v1.25.12+rke2r1
  localClusterAuthEndpoint: {}
  rkeConfig:
    additionalManifest: ""
    chartValues:
      rke2-calico: {}
    etcd:
```

```
    snapshotRetention: 5
    snapshotScheduleCron: 0 */5 * * *
  machineGlobalConfig:
    cni: calico
    disable-kube-proxy: false
    etcd-expose-metrics: false
    profile: null
    kube-apiserver-arg:
      - audit-policy-file=/etc/rancher/rke2/user-audit-policy.yaml
      - audit-log-path=/etc/rancher/rke2/user-audit.logs
  machinePools:
  - controlPlaneRole: true
    etcdRole: true
    machineConfigRef:
      kind: Amazonec2Config
      name: nc-test-pooll-pwl5h
    name: pooll
    quantity: 1
    unhealthyNodeTimeout: 0s
    workerRole: true
  machineSelectorConfig:
  - config:
      protect-kernel-defaults: false
  machineSelectorFiles:
    - fileSources:
        - configMap:
            name: ''
          secret:
            name: audit-policy
            items:
              - key: audit-policy
                path: /etc/rancher/rke2/user-audit-policy.yaml
      machineLabelSelector:
        matchLabels:
          rke.cattle.io/control-plane-role: 'true'
  registries: {}
  upgradeStrategy:
    controlPlaneConcurrency: "1"
    controlPlaneDrainOptions:
      deleteEmptyDirData: true
      enabled: true
      gracePeriod: -1
      ignoreDaemonSets: true
      timeout: 120
    workerConcurrency: "1"
```

```
    workerDrainOptions:
      deleteEmptyDirData: true
      enabled: true
      gracePeriod: -1
      ignoreDaemonSets: true
      timeout: 120
```

**additionalManifest**

Specify additional manifests to deliver to the control plane nodes.

The value is a String, and will be placed at the path `/var/lib/rancher/rke2/server/manifests/rancher/addons.yaml` on target nodes.

Example:

```
additionalManifest: |-
  apiVersion: v1
  kind: Namespace
  metadata:
    name: name-xxxx
```



If you want to customize system charts, you should use the `chartValues` field as described below.

Alternatives, such as using a HelmChartConfig to customize the system charts via `additionalManifest`, can cause unexpected behavior, due to having multiple HelmChartConfigs for the same chart.

**chartValues**

Specify the values for the system charts installed by RKE2.

For more information about how RKE2 manages packaged components, please refer to RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/add-ons/helm.html].

Example:

```
chartValues:
    chart-name:
        key: value
```

**machineGlobalConfig**

Specify RKE2 configurations. Any configuration change made here will apply to every node. The configuration options available in the standalone version of RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/reference/server_config.html] can be applied here.

Example:

```
machineGlobalConfig:
    etcd-arg:
        - key1=value1
        - key2=value2
```

There are some configuration options that can't be changed when provisioning via Rancher:

- data-dir (folder to hold state), which defaults to `/var/lib/rancher/rke2`.

To make it easier to put files on nodes beforehand, Rancher expects the following values to be included in the configuration, while RKE2 expects the values to be entered as file paths:

- audit-policy-file
- cloud-provider-config
- private-registry

Rancher delivers the files to the path `/var/lib/rancher/rke2/etc/config-files/<option>` in target nodes, and sets the proper options in the RKE2 server.

Example:

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  rkeConfig:
    machineGlobalConfig:
      audit-policy-file: |
        apiVersion: audit.k8s.io/v1
        kind: Policy
        rules:
        - level: RequestResponse
          resources:
          - group: ""
            resources:
            - pods
```

#### machineSelectorConfig

`machineSelectorConfig` is the same as `machineGlobalConfig` except that a label selector can be specified with the configuration. The configuration will only be applied to nodes that match the provided label selector.

Multiple `config` entries are allowed, each specifying their own `machineLabelSelector`. A user can specify `matchExpressions`, `matchLabels`, both, or neither. Omitting the `machineLabelSelector` section of this field has the same effect as putting the config in the `machineGlobalConfig` section.

Example:

```
machineSelectorConfig:
  - config:
      config-key: config-value
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
        key1: value1
        key2: value2
```

**machineSelectorFiles**



This feature is available in Rancher v2.7.2 and later.

Deliver files to nodes, so that the files can be in place before initiating RKE2 server or agent processes. The content of the file is retrieved from either a secret or a configmap. The target nodes are filtered by the `machineLabelSelector`.

Example :

```
machineSelectorFiles:
  - fileSources:
      - secret:
          items:
            - key: example-key
              path: path-to-put-the-file-on-nodes
              permissions: 644 (optional)
              hash: base64-encoded-hash-of-the-content (optional)
          name: example-secret-name
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
```

```
        key1: value1
        key2: value2
  - fileSources:
      - configMap:
          items:
            - key: example-key
              path: path-to-put-the-file-on-nodes
              permissions: 644 (optional)
              hash: base64-encoded-hash-of-the-content (optional)
          name: example-configmap-name
    machineLabelSelector:
      matchExpressions:
        - key: example-key
          operator: string # Valid operators are In, NotIn, Exists and
DoesNotExist.
          values:
            - example-value1
            - example-value2
      matchLabels:
        key1: value1
        key2: value2
```

The secret or configmap must meet the following requirements:

1. It must be in the `fleet-default` namespace where the Cluster object exists.
2. It must have the annotation `rke.cattle.io/object-authorized-for-clusters: cluster-name1,cluster-name2`, which permits the target clusters to use it.



Rancher Dashboard provides an easy-to-use form for creating the secret or configmap.

Example:

```
apiVersion: v1
data:
  audit-policy: >-

IyBMb2cgYWxsIHJlcXVlc3RzIGF0IHRoZSBNZXRhZGF0YSBsZXZlbC4KYXBpVmVyc2lvbjog
YXVkaXQuazhzLmlvL3YxCmtpbmQ6IFBvbGljeQpydWxlczoKLSBsZXZlbDogTWV0YWRhdGE=
kind: Secret
metadata:
  annotations:
    rke.cattle.io/object-authorized-for-clusters: cluster1
  name: name1
  namespace: fleet-default
```

## Configuring Native CAPI Infrastructure Providers to Provision RKE2 Clusters

### Overview

Rancher can provision RKE2 clusters using native CAPI infrastructure providers, such as CAPA [https://cluster-api-aws.sigs.k8s.io/] (Cluster API Provider AWS) and CAPV [https://github.com/kubernetes-sigs/cluster-api-provider-vsphere] (Cluster API Provider vSphere).

Standard RKE2 provisioning relies on Rancher's internal bootstrap and control plane providers alongside Rancher Node Drivers (via `rancher/machine`) as the infrastructure provider. This new mode allows you to substitute Rancher Node Drivers with native CAPI infrastructure providers while retaining Rancher's bootstrap and control plane logic.

You can provision and manage native CAPI clusters either directly through the Rancher UI or manually using `kubectl`.

This guide provides steps for provisioning through the Rancher UI as well as example manifests for manual provisioning using CAPA and CAPV. SUSE® Rancher Prime: Cluster API [https://documentation.suse.com/cloudnative/cluster-api/latest/en/index.html] is the supported method for installing CAPA and CAPV. Refer to the documentation of each provider for more details on available options and adapt these examples to your needs.



Provisioning with a native CAPI infrastructure provider and Rancher as a bootstrap and control plane provider is distinct from using SUSE® Rancher Prime: Cluster API [https://documentation.suse.com/cloudnative/cluster-api/latest/en/index.html] and the CAPRKE2 [https://caprke2.docs.rancher.com/] provider to provision a RKE2 cluster and subsequently import it into Rancher.

By default, the `Standard User` and `Create Clusters` global permissions allow assigned users to create and update CAPI infrastructure objects in the `fleet-default` namespace. These permissions also allow users to get and list namespaced `capiproviders` resources in the `turtles-capi.cattle.io` API group.



*Plain-text credentials in CAPIProvider resources*

Do not store credentials or other sensitive values in the `.spec.variables` field of a `CAPIProvider` resource. Values in this field are stored in plain text and can be viewed by any user who has permission to access provider configurations.

To securely manage credentials, use provider-specific identity objects, such as `AWSClusterStaticIdentity` or `VSphereClusterIdentity`. Create these identity objects and reference them directly from infrastructure cluster objects, such as `AWSCluster` or `VSphereCluster`.

### Limitations and requirements

- Windows worker nodes are currently not supported.
- SSH access to cluster nodes through the Rancher UI is not supported.
- Kubernetes cloud provider requirements: a cloud-specific Kubernetes provider for the infrastructure where the downstream cluster runs is required (e.g., the Kubernetes AWS Cloud Provider [https://cloud-provider-aws.sigs.k8s.io/] for CAPA or the `rancher-vsphere-cpi` chart for CAPV).

#### General steps

For both CAPA and CAPV, the general steps are as follows:

1. Install Rancher.
2. Install a CAPI infrastructure provider, either CAPA or CAPV.
3. Set-up an identity resource for the provider, or use an identity resource mirrored from a cloud credential. Automatic mirroring is currently supported only for CAPA's `AWSClusterStaticIdentity`.
4. Create the downstream cluster and infrastructure resources either through the **Rancher UI** or by applying manifests via **`kubectl`**.

After provisioning, the cluster appears in the Rancher Cluster Management list (click ☰ **> Cluster Management**).

To view the progress of the provisioning process and troubleshoot, refer to the status of the various CAPI and Rancher provisioning resources in the local cluster:

1. Click ☰, then click on the icon for your local cluster.
2. Use the dropdown menu at the top to filter for **All Namespaces**.
3. From the sidebar, select **More Resources > Cluster Provisioning**.

The logs for the infrastructure provider deployment (e.g. `capa-controller-manager`) also show useful information.

#### Installing the infrastructure provider

Rancher allows installing the required infrastructure provider through SUSE® Rancher Prime: Cluster API. Refer to the documentation [https://documentation.suse.com/cloudnative/cluster-api/latest/en/user/providers.html] to install the CAPA or CAPV provider.

#### Provisioning a cluster via the Rancher UI

Rancher provides native UI workflows to provision RKE2 clusters using CAPI infrastructure providers without requiring manual YAML creation.

Current support for installable UI extensions is limited to CAPA.

#### Prerequisites

- Ensure the desired CAPI provider (e.g., CAPA) is installed and active in the local cluster via SUSE® Rancher Prime: Cluster API. Providers can also be installed in the UI via YAML by navigating to the **Providers > Cluster API Providers** view and clicking **Create from YAML**.
- Install the UI extension required for the provider (e.g., `SUSE CAPI CAPA UI` for CAPA) in order to provision via UI.

#### CAPI Cluster creation steps

1. Navigate to ☰ **> Cluster Management**.
2. Click **Create**.
3. Under the **Create a cluster using Cluster API** section select your desired CAPI provider after the respective UI extension is installed (e.g., **CAPI AWS** for CAPA).

4. **Cluster Options**:
   - Enter a **Cluster Name**.
   - Select the target **Kubernetes Version** (RKE2).
   - Choose or configure the **Infrastructure Identity / Cloud Credential** for the provider.
   - Note that there are certain preconfigured defaults for the cluster:
     - **Cluster Provider**: This is set to `external` for provisioning with a native CAPI infrastructure provider.
     - **Additional Manifest**: An additional manifest is used to install Helm charts necessary for cluster provisioning (e.g. `aws-cloud-controller-manager` for CAPA).
     - **machineSelectorConfig**: Currently not supported in the UI form.
   - **Machine Pools**:
     - Add one or more machine pools and assign node roles (**Control Plane**, **etcd**, **Worker**).
     - Configure the machine pool parameters (e.g. subnet, IAM instance profile name, machine image).
   - **Advanced Configuration**:
     - Configure CNI, security groups, or custom cloud-init user data as required by your environment.
5. Click **Create** to deploy the cluster.

Once the cluster is provisioned, it appears in the Rancher Cluster Management list (click **> Cluster Management**). Most cluster fields will be editable via **Edit Config**, however the **Cluster Name**, **Region**, and **VPC** fields are immutable after creation. Note that for CAPI AWS clusters, users will not be able to SSH into the CAPA machine pools.

#### Provisioning a cluster via `kubectl`

These examples use a single machine pool with all roles (control plane, etcd and worker) for simplicity.

Create the resources in your upstream cluster, and replace values within <> brackets.



Each machine pool defined in the `clusters.provisioning.cattle.io` resource should reference a different machine template.

##### CAPA

First, configure IAM as required by CAPA. Different permissions are needed by the CAPA controller in the upstream cluster and by the downstream cluster to enable the Kubernetes AWS Cloud Provider.

To do this, CAPA provides the `clusterawsadm` tool to generate and apply the required objects. Refer to the CAPA manual for more details [https://cluster-api-aws.sigs.k8s.io/topics/using-clusterawsadm-to-fulfill-prerequisites].

Then, configure the provider identity in the upstream cluster so that the CAPA provider can create resources on AWS.

To do this, create a cloud credential for AWS. Rancher creates the corresponding `AWSClusterStaticIdentity` with the same name, if the CAPA provider is installed. Rancher will not create a corresponding identity if a manually created identity of the same name already exists.



By default, the `Standard User` and `Create Clusters` global permissions allow

creating and then updating `AWSCluster` objects in the `fleet-default` namespace.

Manually created infrastructure provider identity objects, such as `AWSClusterStaticIdentity`, that target the `fleet-default` namespace can be used through CAPI infrastructure objects such as `AWSCluster`.

However, identity resources automatically mirrored by Rancher from cloud credentials can only be referenced by `AWSCluster` objects created or updated by users who have read access to the underlying cloud credential.

Currently Rancher only creates `AWSClusterStaticIdentity` objects from cloud credentials for CAPA.

Now, create the `AWSCluster` resource [https://cluster-api-aws.sigs.k8s.io/crd/#infrastructure.cluster.x-k8s.io%2fv1beta2]. This object defines the infrastructure configuration common to all machine pools.

CAPA creates VPCs, subnets, security groups and a load balancer in its default configuration, but additional rules must be configured to allow ports needed by Rancher and RKE2.



The following example defines security group rules for RKE2 and for calico in the default configuration. Adapt this example when using another CNI or different calico configurations.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSCluster
metadata:
  name: capa-lab
  namespace: fleet-default
spec:
  identityRef:
    kind: AWSClusterStaticIdentity
    name: <Name of the identity, e.g. cc-xxxxx>

  controlPlaneLoadBalancer:
    loadBalancerType: nlb
    healthCheckProtocol: TCP
    scheme: internal

  region: <e.g. us-east-1>

  network:
    additionalNodeIngressRules:
      - description: "RKE2 supervisor API"
        protocol: tcp
        fromPort: 9345
        toPort: 9345
        sourceSecurityGroupRoles:
          - controlplane
```

```
          - node
      - description: "ETCD client and peer"
        protocol: tcp
        fromPort: 2379
        toPort: 2380
        sourceSecurityGroupRoles:
          - controlplane
          - node
    cni:
      cniIngressRules:
        - description: "Calico VXLAN"
          protocol: udp
          fromPort: 4789
          toPort: 4789
        - description: "Calico Typha"
          protocol: tcp
          fromPort: 5473
          toPort: 5473
```

Next, create a machine template for the control plane machine pool. Create additional templates for every machine pool defined in the `clusters.provisioning.cattle.io` resource.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: AWSMachineTemplate
metadata:
  name: capa-lab-control-plane
  namespace: fleet-default
spec:
  template:
    spec:
      ami:
        # The ami requires cloud-init.
        id: <your ami>
      # This should correspond to the profile created through
clusterawsadm.
      # Worker or etcd-only nodes should use nodes.cluster-api-provider-
aws.sigs.k8s.io.
      iamInstanceProfile: control-plane.cluster-api-provider-
aws.sigs.k8s.io
      instanceType: t3.medium
      privateDnsName:
        hostnameType: resource-name
      # This refers to the name of an EC2 key pair.
      sshKeyName: <your ssh key>
      rootVolume:
```

```
        size: 16
      cloudInit:
        insecureSkipSecretsManager: true
```



The `insecureSkipSecretsManager` option is set to true to bypass the AWS secrets manager as a source of user-data for the provisioned instances. This source restricts the visibility of the user-data but has additional requirements on the AMI, such as the AWS CLI and a custom data source [https://github.com/kubernetes-sigs/image-builder/pull/1583] for cloud-init. Refer to the CAPA documentation [https://cluster-api-aws.sigs.k8s.io/topics/userdata-privacy] and the image-builder [https://github.com/kubernetes-sigs/image-builder] project for examples on how to build your own image to support enabling the secrets manager.

Refer to Rancher Security Best Practices for further information on how to restrict access to user-data.

Finally, create the Rancher `clusters.provisioning.cattle.io` resource and point to the CAPA cluster and machine template that were just created.

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
metadata:
  name: capa-lab
  namespace: fleet-default
spec:
  kubernetesVersion: <RKE2 version, e.g. v1.35.6+rke2r1>
  rkeConfig:
    # This is the ref to the infra cluster defined above.
    infrastructureRef:
      kind: AWSCluster
      name: capa-lab
      namespace: fleet-default
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
    machinePools:
      - name: ctrl
        controlPlaneRole: true
        etcdRole: true
        workerRole: true
        quantity: 3
        machineConfigRef:
          kind: AWSMachineTemplate
          name: capa-lab-control-plane
          namespace: fleet-default
          apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
    machineGlobalConfig:
      cni: calico
```

```
      disable-kube-proxy: false
      etcd-expose-metrics: false
      ingress-controller: traefik
      protect-kernel-defaults: false
      cloud-provider-name: external
    machineSelectorConfig:
    - config:
        disable-cloud-controller: true
      machineLabelSelector:
        matchExpressions:
          - key: rke.cattle.io/control-plane-role
            operator: In
            values:
              - 'true'
    - config:
        disable-cloud-controller: true
      machineLabelSelector:
        matchExpressions:
          - key: rke.cattle.io/etcd-role
            operator: In
            values:
              - 'true'
    # The AWS cloud controller definition. In this case, the controller
uses the IAM instance profile for its AWS credentials.
    additionalManifest: |-
      apiVersion: helm.cattle.io/v1
      kind: HelmChart
      metadata:
        name: aws-cloud-controller-manager
        namespace: kube-system
      spec:
        chart: aws-cloud-controller-manager
        repo: https://kubernetes.github.io/cloud-provider-aws
        targetNamespace: kube-system
        bootstrap: true
        valuesContent: |-
          hostNetworking: true
          nodeSelector:
            node-role.kubernetes.io/control-plane: "true"
          args:
            - --configure-cloud-routes=false
            - --v=5
            - --cloud-provider=aws
          tolerations:
                - key: node.cloudprovider.kubernetes.io/uninitialized
```

```
      value: "true"
      effect: NoSchedule
    - key: node-role.kubernetes.io/master
      effect: NoSchedule
    - key: node-role.kubernetes.io/control-plane
      effect: NoSchedule
    - key: node-role.kubernetes.io/etcd
      effect: NoExecute
```

#### CAPV

First, configure the provider identity in the upstream cluster so that the CAPV provider can create resources on your vSphere server. Refer to the manual for all identity options [https://github.com/kubernetes-sigs/cluster-api-provider-vsphere/blob/v1.15.2/docs/identity_management.md], and for general vSphere requirements [https://github.com/kubernetes-sigs/cluster-api-provider-vsphere/blob/v1.15.2/docs/getting_started.md].



By default, the `Standard User` and `Create Clusters` global permissions allow creating and then updating `VsphereCluster` objects in the `fleet-default` namespace.

Manually created infrastructure provider identity objects, such as `VSphereClusterIdentity` in the example below, that target the `fleet-default` namespace can be used through CAPI infrastructure objects such as `VsphereCluster`.

In this example, we'll use `VSphereClusterIdentity`.

Create a secret with your credentials:

```
apiVersion: v1
kind: Secret
metadata:
  name: capv-lab-credentials
  namespace: capv-system
type: Opaque
stringData:
  username: <your vSphere username>
  password: <your vSphere password>
```

Then, create the identity object that references the secret:

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: VSphereClusterIdentity
metadata:
  name: capv-lab-identity
spec:
  secretName: capv-lab-credentials
```

```
  allowedNamespaces:
    selector:
      # The namespace of the VSphereCluster for which this identity
      # is used when provisioning. All Rancher standard users are
      # able to create VSphereCluster objects in fleet-default that
      # reference this identity.
      matchLabels:
        kubernetes.io/metadata.name: fleet-default
```

Like for CAPA, it is also necessary to install the cloud provider for vSphere [https://github.com/kubernetes/cloud-provider-vsphere] in the downstream cluster.

To securely transfer the credentials for the CPI chart, you can enable the prebootstrap feature in Rancher. This can be done by enabling the `provisioningprebootstrap` feature flag and causes Rancher to restart.

Now, create the secret that is sent to the downstream cluster. If you use a different name to create the `clusters.provisioning.cattle.io` resource, make sure you update the `rke.cattle.io/object-authorized-for-clusters` annotation below.

```
# Credential secret synced to the downstream cluster for the vsphere CPI
chart.
apiVersion: v1
kind: Secret
metadata:
  name: vsphere-cpi-creds
  namespace: fleet-default
  annotations:
    # Can be a comma-separated list for multiple clusters, with no
spaces.
    rke.cattle.io/object-authorized-for-clusters: capv-lab
    provisioning.cattle.io/sync-bootstrap: "true"
    provisioning.cattle.io/sync-target-namespace: kube-system
type: Opaque
stringData:
  # Change the prefix of the key to match your vCenter host.
  <vsphere host>.username: <your vSphere username>
  <vsphere host>.password: <your vSphere password>
```

Now, create the `VSphereCluster` resource. This resource defines the infrastructure configuration common to all machine pools. Refer to the CAPV documentation for more configuration options.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: VSphereCluster
metadata:
  name: capv-lab
```

```
  namespace: fleet-default
spec:
  identityRef:
    kind: VSphereClusterIdentity
    name: capv-lab-identity
  server: <vsphere fqdn>
```

Next, create a machine template for the control plane machine pool. Create additional templates for every machine pool defined in the `clusters.provisioning.cattle.io` resource.

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
kind: VSphereMachineTemplate
metadata:
  name: capv-lab-control-plane
  namespace: fleet-default
spec:
  template:
    spec:
      datacenter: <datacenter>
      datastore: <datastore>
      diskGiB: 20
      folder: <your folder>
      memoryMiB: 4096
      network:
        devices:
        - dhcp4: true
          networkName: <your network>
      numCPUs: 2
      os: Linux
      resourcePool: <your resource pool>
      template: <your VM template>
```

Finally, create the Rancher `clusters.provisioning.cattle.io` resource and point to the CAPV cluster and machine template that were just created. Note that this example disables the CSI chart for simplicity. The CPI chart is required.

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
metadata:
  name: capv-lab
  namespace: fleet-default
spec:
  kubernetesVersion: <RKE2 version, e.g. v1.35.6+rke2r1>
  rkeConfig:
```

```
    infrastructureRef:
      kind: VSphereCluster
      name: capv-lab
      namespace: fleet-default
      apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
    machinePools:
      - name: ctrl
        controlPlaneRole: true
        etcdRole: true
        workerRole: true
        quantity: 3
        machineConfigRef:
          kind: VSphereMachineTemplate
          name: capv-lab-control-plane
          namespace: fleet-default
          apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
    machineGlobalConfig:
      cni: calico
      disable-kube-proxy: false
      etcd-expose-metrics: false
      ingress-controller: traefik
      protect-kernel-defaults: false
      disable:
        - rancher-vsphere-csi
      cloud-provider-name: rancher-vsphere
    chartValues:
      rancher-vsphere-cpi:
        vCenter:
          datacenters: <your datacenter>
          host: <vsphere fqdn>
          # The credential secret is transferred by the prebootstrap
mechanism,
          # and the cpi chart expects the default name (vsphere-cpi-
creds).
          credentialsSecret:
            generate: false
```

#### Changing machine templates

Machine templates for CAPI infrastructure providers, such as `AWSMachineTemplate` and `VSphereMachineTemplate` are usually immutable. To modify the configuration of the instances in a machine pool, create a new template with a different name, then edit the machine pool in `clusters.provisioning.cattle.io` to point to this new template. This causes all of the machines in that pool to be recreated with the new configuration. Remember to delete the old machine template if it is no longer in use.

#### Customizing user-data

You can define custom cloud-init user data for each machine pool in the `clusters.provisioning.cattle.io` resource. To do this, set the value of the `.spec.rkeConfig.machinePools.userdata.inlineUserdata` field to an inline yaml string in the cloud-config format. Rancher merges this user-data with the user-data it generates to bootstrap the cluster nodes.

For this provisioning method, Rancher generates the final user-data as a Jinja template [https://docs.cloud-init.io/en/latest/explanation/format/jinja.html#user-data-formats-jinja]. You can therefore add limited Jinja directives in your custom user-data, such as some cloud-init variable substitutions.



Do not include sensitive data in this field, as it is part of a resource other than a Secret.

The `.spec.rkeConfig.machinePools.userdata.inlineUserdata` field is only valid for native CAPI providers described in this document, and has no effect on clusters provisioned by Rancher through the standard method with node drivers. Rancher only generates user-data in the Jinja format for native CAPI providers.

Modifying the user-data field causes all of the machines in the pool to be recreated.

```
# Only some fields of the provisioning cluster resource are shown here.
apiVersion: provisioning.cattle.io/v1
kind: Cluster
metadata:
  name: capv-lab
  namespace: fleet-default
spec:
  rkeConfig:
    machinePools:
      - name: ctrl
        userdata:
          inlineUserdata: |
            runcmd:
              - ["echo", "Hello!"]
```

# Registering Existing Clusters

The cluster registration feature replaced the feature to import clusters.

The control that Rancher has to manage a registered cluster depends on the type of cluster. For details, see Management Capabilities for Registered Clusters.

## Prerequisites

#### Kubernetes Node Roles

Registered RKE Kubernetes clusters must have all three node roles - etcd, controlplane and worker. A cluster

with only controlplane components cannot be registered in Rancher.

For more information on RKE node roles, see the best practices.

#### Permissions

To register a cluster in Rancher, you must have `cluster-admin` privileges within that cluster. If you don't, grant these privileges to your user by running:

```
kubectl create clusterrolebinding cluster-admin-binding \
  --clusterrole cluster-admin \
  --user [USER_ACCOUNT]
```

Since, by default, Google Kubernetes Engine (GKE) doesn't grant the `cluster-admin` role, you must run these commands on GKE clusters before you can register them. To learn more about role-based access control for GKE, please see the official Google documentation [https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control].

#### Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), and Google Kubernetes Engine (GKE)

To successfully import or provision EKS, AKS, and GKE clusters from Rancher, the cluster must have at least one managed node group.

AKS clusters can be imported only if local accounts are enabled. If a cluster is configured to use Microsoft Entra ID for authentication, then Rancher will not be able to import it and report an error.

EKS Anywhere clusters can be imported/registered into Rancher with an API address and credentials, as with any downstream cluster. EKS Anywhere clusters are treated as imported clusters and do not have full lifecycle support from Rancher.

GKE Autopilot clusters aren't supported. See Compare GKE Autopilot and Standard [https://cloud.google.com/kubernetes-engine/docs/resources/autopilot-standard-feature-comparison] for more information about the differences between GKE modes.

## Registering a Cluster

1. Click ☰ **> Cluster Management**.
2. On the **Clusters** page, **Import Existing**.
3. Choose the type of cluster.
4. Use **Member Roles** to configure user authorization for the cluster. Click **Add Member** to add users that can access the cluster. Use the **Role** drop-down to set permissions for each user.
5. If you are importing a generic Kubernetes cluster in Rancher, perform the following steps for setup:
    a. Click **Agent Environment Variables** under **Cluster Options** to set environment variables for rancher cluster agent. The environment variables can be set using key value pairs. If rancher agent requires use of proxy to communicate with Rancher server, `HTTP_PROXY`, `HTTPS_PROXY` and `NO_PROXY` environment variables can be set using agent environment variables.
    b. Enable Project Network Isolation to ensure the cluster supports Kubernetes `NetworkPolicy` resources. Users can select the **Project Network Isolation** option under the **Advanced Options**

dropdown to do so.

c. Configure the version management feature for imported RKE2 and K3s clusters.

6. Click **Create**.
7. The prerequisite for `cluster-admin` privileges is shown (see **Prerequisites** above), including an example command to fulfil the prerequisite.
8. Copy the `kubectl` command to your clipboard and run it on a node where kubeconfig is configured to point to the cluster you want to import. If you are unsure it is configured correctly, run `kubectl get nodes` to verify before running the command shown in Rancher.
9. If you are using self-signed certificates, you will receive the message `certificate signed by unknown authority`. To work around this validation, copy the command starting with `curl` displayed in Rancher to your clipboard. Then run the command on a node where kubeconfig is configured to point to the cluster you want to import.
10. When you finish running the command(s) on your node, click **Done**.



The `NO_PROXY` environment variable is not standardized, and the accepted format of the value can differ between applications. When configuring the `NO_PROXY` variable for Rancher, the value must adhere to the format expected by Golang.

Specifically, the value should be a comma-delimited string which only contains IP addresses, CIDR notation, domain names, or special DNS labels (e.g. `*`). For a full description of the expected value format, refer to the **upstream Golang documentation** [https://pkg.go.dev/golang.org/x/net/http/httpproxy#Config].

**Result:**

- Your cluster is registered and assigned a state of **Pending**. Rancher is deploying resources to manage your cluster.
- You can access your cluster after its state is updated to **Active**.
- **Active** clusters are assigned two Projects: `Default` (containing the namespace `default`) and `System` (containing the namespaces `cattle-system`, `traefik`, `kube-public` and `kube-system`, if present).



You can not re-register a cluster that is currently active in a Rancher setup.

**Configuring an Imported EKS, AKS or GKE Cluster with Terraform**

You should define **only** the minimum fields that Rancher requires when importing an EKS, AKS or GKE cluster with Terraform. This is important as Rancher will overwrite what was in the cluster configuration with any config that the user has provided.



Even a small difference between the current cluster and a user-provided config could have unexpected results.

The minimum config fields required by Rancher to import EKS clusters with Terraform using `eks_config_v2` are as follows:

- cloud_credential_id
- name

- region
- imported (this field should always be set to `true` for imported clusters)

Example YAML configuration for imported EKS clusters:

```
resource "rancher2_cluster" "my-eks-to-import" {
  name        = "my-eks-to-import"
  description = "Terraform EKS Cluster"
  eks_config_v2 {
    cloud_credential_id = rancher2_cloud_credential.aws.id
    name                = var.aws_eks_name
    region              = var.aws_region
    imported            = true
  }
}
```

You can find additional examples for other cloud providers in the Rancher2 Terraform Provider documentation [https://registry.terraform.io/providers/rancher/rancher2/latest/docs/resources/cluster].

## Management Capabilities for Registered Clusters

The control that Rancher has to manage a registered cluster depends on the type of cluster.

- Features for All Registered Clusters
- Additional Features for Registered RKE2 and K3s Clusters
- Additional Features for Registered EKS, AKS and GKE Clusters

### Features for All Registered Clusters

After registering a cluster, the cluster owner can:

- Manage cluster access through role-based access control
- Enable monitoring, alerts and notifiers
- Enable logging
- Enable Istio
- Manage projects and workloads

### Additional Features for Registered SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Clusters

K3s [https://documentation.suse.com/cloudnative/k3s/latest/en/introduction.html] is a lightweight, fully compliant Kubernetes distribution for edge installations.

RKE2 [https://documentation.suse.com/cloudnative/rke2/latest/en/introduction.html] is Rancher's next-generation Kubernetes distribution for datacenter and cloud installations.

When an RKE2 or K3s cluster is registered in Rancher, Rancher will recognize it. The Rancher UI will expose features available to all registered clusters, along with the following options for editing and upgrading the

cluster:

- Enable or disable version management
- Upgrade the Kubernetes version when version management is enabled
- Configure the upgrade strategy when version management is enabled
- View a read-only version of the cluster's configuration arguments and environment variables used to launch each node

**Additional Features for Registered EKS, AKS, and GKE Clusters**

Rancher handles registered EKS, AKS, or GKE clusters similarly to clusters created in Rancher. However, Rancher doesn't destroy registered clusters when you delete them through the Rancher UI.

When you create an EKS, AKS, or GKE cluster in Rancher, then delete it, Rancher destroys the cluster. When you delete a registered cluster through Rancher, the Rancher server *disconnects* from the cluster. The cluster remains live, although it's no longer in Rancher. You can still access the deregistered cluster in the same way you did before you registered it.

See Cluster Management Capabilities by Cluster Type for more information about what features are available for managing registered clusters.

## Day 2 Operations for Imported RKE2 and K3s Clusters



Rancher does not support Day 2 operations for imported RKE2 clusters with Windows worker nodes.

You can perform Day 2 operations directly from the Rancher UI on the following clusters:

- RKE2 and K3s clusters imported into Rancher using the **Generic** option.
- RKE2 clusters provisioned with CAPRKE2 [https://caprke2.docs.rancher.com/] and imported into Rancher by SUSE® Rancher Prime: Cluster API.

The available operations are rotating encryption keys and creating, viewing, and restoring local `etcd` snapshots.

Rancher coordinates these operations across the applicable server nodes. You don't need to run the corresponding RKE2 or K3s commands manually on each node.



Rancher runs one Day 2 operation at a time for each cluster. A new operation remains pending until the current operation finishes.

**Prerequisites**

Before you perform a Day 2 operation, ensure that the following prerequisites are met:

- **Rancher Version:** Rancher v2.15 or later.
- **Feature Flag:** The `imported-day-2-ops` feature flag is active by default.
- **Cluster Enablement:** Day 2 operations are enabled for the cluster. See Configuring Day 2 Operations.
- **Cluster Health:** The cluster is in the **Active** state, its Kubernetes API is reachable, and the `rancher-`

`system-agent` has registered on every cluster node.

- **Version Management:** For clusters imported using the **Generic** option, version management is enabled so Rancher can install or update the required `rancher-system-agent` support. See Configuring Version Management for RKE2 and K3s Clusters.
- **Installation Layout:** Clusters imported using the **Generic** option use the standard RKE2 or K3s installation layout, including the default data and configuration directories and systemd service names. Custom data directories, custom configuration paths, and IPv6-only server networking aren't supported.
- **Datastore:** For snapshot and restore operations, the cluster uses embedded `etcd`. K3s clusters that use SQLite or an external datastore aren't supported for these operations.
- **Snapshot Storage:** Day 2 snapshot operations support only local `etcd` snapshots. Creating or restoring S3-backed snapshots isn't supported.
- **Server Token:** For snapshot and restore operations, every applicable `etcd` server node has a non-empty `token:` value in `/etc/rancher/rke2/config.yaml`, `/etc/rancher/k3s/config.yaml`, or a YAML or JSON file under `/etc/rancher/rke2/config.yaml.d/` or `/etc/rancher/k3s/config.yaml.d/`. A token supplied only through an environment variable, command-line argument, `token-file`, `/var/lib/rancher/rke2/server/token`, or `/var/lib/rancher/k3s/server/token` doesn't meet this prerequisite.
- **Secrets Encryption:** For encryption-key rotation, secrets encryption is enabled in the RKE2 or K3s configuration.
- **Permissions:** Your Rancher account is a global administrator, a **Cluster Owner**, or has an equivalent custom role with access to the `operation.cattle.io` resources.



The server token is sensitive information. Do not expose it in command output, logs, support bundles, or documentation examples.

#### Configuring Day 2 Operations

The `imported-day-2-ops` feature flag is active by default. If the feature flag is inactive, day 2 operations are disabled regardless of the global setting or per-cluster annotation. Changing the feature flag requires a Rancher restart. For instructions, see Feature Flags.

The `imported-cluster-day2-ops-enabled` global setting is `true` by default. This setting applies to eligible new and existing clusters that inherit the global setting.



When you upgrade Rancher to v2.15, Rancher automatically starts installing or updating `rancher-system-agent` support on existing eligible RKE2 and K3s clusters imported using the **Generic** option. This applies to clusters that use the enabled global default and meet the prerequisites in the previous section. If a cluster is temporarily unreachable, Rancher retries when it becomes available.

##### Preventing Automatic Agent Installation During an Upgrade

To prevent automatic installation on all eligible RKE2 and K3s clusters imported using the **Generic** option, add `imported-day-2-ops=false` to the `CATTLE_FEATURES` environment variable in the Helm values used for the upgrade:

```
extraEnv:
```

```
  - name: CATTLE_FEATURES
    value: "imported-day-2-ops=false"
```

Merge this entry into the existing `extraEnv` list in your values file. If `CATTLE_FEATURES` already contains other feature flags, retain them in the comma-separated value when you add `imported-day-2-ops=false`.

To prevent automatic installation on an individual cluster, edit its `Cluster.management.cattle.io` resource in the Rancher management cluster and add the `rancher.io/operations-enabled: "false"` annotation before upgrading:

```
metadata:
  annotations:
    rancher.io/operations-enabled: "false"
```

Changing the feature flag requires feature-management permissions. Changing the global setting requires settings-management permissions. Running operations requires access to the applicable `operation.cattle.io` resources.

The annotation accepts the following values:

- `true`: Enables day 2 operations for the cluster.
- `false`: Disables day 2 operations for the cluster.
- `system-default`: Uses the `imported-cluster-day2-ops-enabled` global setting.

If the annotation is absent or has another value, the cluster also inherits the global setting.



When you disable day 2 operations, Rancher waits for the current operation to finish before removing the support resources. When you re-enable the functionality, allow time for Rancher to reconcile those resources before starting an operation.

#### Encryption Key Rotation

For clusters with secrets encryption enabled, Rancher coordinates encryption-key rotation across the cluster's server nodes and verifies that the nodes converge on the new key.

##### Steps to Rotate Encryption Keys



Before you rotate the keys, create a current `etcd` snapshot and verify that it can be used for recovery. If rotation fails, an `etcd` restore might be required.

1. Click ☰ **> Cluster Management**.
2. Locate the cluster.
3. Click ☰ **> Rotate Encryption Keys**.
4. Review the confirmation prompt, then click **Rotate**.
5. Click the **Operations** tab and wait for the encryption key rotation operation to reach the **Succeeded** state.

6. Verify that the cluster is in the **Active** state.



Don't make cluster configuration changes while key rotation is in progress.

### Managing `etcd` Snapshots and Restores

#### Taking an `etcd` Snapshot



Creating a snapshot sequentially restarts the RKE2 or K3s server service on the `etcd` nodes. The Kubernetes API might be briefly unavailable during the operation.

1. Click ☰ > **Cluster Management** and select the cluster.
2. Click the **Snapshots** tab.
3. Click **Snapshot Now**.
4. Wait for the snapshot state to indicate that the operation succeeded.

#### Viewing Existing `etcd` Snapshots

To view the snapshots discovered for the cluster:

1. Click ☰ > **Cluster Management**.
2. Click the name of the cluster.
3. Click the **Snapshots** tab.
4. Snapshots are grouped by storage location. The table displays local snapshots associated with the cluster, including the following details when available:
   - **State:** The snapshot creation state.
   - **Name:** The snapshot identifier or filename.
   - **Size:** The snapshot archive's file size.
   - **Age:** The time elapsed since the snapshot was created.

#### Restoring an `etcd` Snapshot



Restoring an `etcd` snapshot replaces the current Kubernetes state with the state stored in the snapshot. The Kubernetes API and workloads might be temporarily unavailable while Rancher shuts down and restarts server nodes. Don't make cluster configuration changes during the restore.



You can restore only successful snapshots created after the cluster was imported into Rancher. The restore action is unavailable for snapshots that predate the import.

1. Click ☰ > **Cluster Management** and select the cluster.
2. Click the **Snapshots** tab.
3. Locate the local snapshot to restore, then click ☰ > **Restore**.
4. Review the confirmation prompt, then click **Restore**.
5. Click the **Operations** tab and wait for the snapshot restore operation to reach the **Succeeded** state.

6. Verify that the cluster is in the **Active** state and that the expected workloads are available.

## Configuring Version Management for SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Clusters



When version management is enabled for an imported cluster, upgrading it outside of Rancher may lead to unexpected consequences.



Rancher does not support managing Kubernetes versions for imported RKE2 clusters with Windows worker nodes.

The version management feature for imported RKE2 and K3s clusters can be configured using one of the following options:

- **Global default** (default): Inherits behavior from the global **imported-cluster-version-management** setting.
- **True**: Enables version management, allowing users to control the Kubernetes version and upgrade strategy of the cluster through Rancher.
- **False**: Disables version management, enabling users to manage the cluster's Kubernetes version independently, outside of Rancher.

You can define the default behavior for newly created clusters or existing ones set to "Global default" by modifying the **imported-cluster-version-management** setting.

Changes to the global **imported-cluster-version-management** setting take effect during the cluster's next reconciliation cycle.



If version management is enabled for a cluster, Rancher will deploy the `system-upgrade-controller` app, along with the associated Plans and other required Kubernetes resources, to the cluster. If version management is disabled, Rancher will remove these components from the cluster.

## Configuring SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Cluster Upgrades



It is a Kubernetes best practice to back up the cluster before upgrading. When upgrading a high-availability K3s cluster with an external database, back up the database in whichever way is recommended by the relational database provider.

The **concurrency** is the maximum number of nodes that are permitted to be unavailable during an upgrade. If number of unavailable nodes is larger than the **concurrency,** the upgrade will fail. If an upgrade fails, you may need to repair or remove failed nodes before the upgrade can succeed.

- **Controlplane concurrency:** The maximum number of server nodes to upgrade at a single time; also the maximum unavailable server nodes
- **Worker concurrency:** The maximum number worker nodes to upgrade at the same time; also the maximum unavailable worker nodes

In the RKE2 and K3s documentation, controlplane nodes are called server nodes. These nodes run the

Kubernetes master, which maintains the desired state of the cluster. By default, these controlplane nodes have the capability to have workloads scheduled to them by default.

Also in the RKE2 and K3s documentation, nodes with the worker role are called agent nodes. Any workloads or pods that are deployed in the cluster can be scheduled to these nodes by default.

### Debug Logging and Troubleshooting for Registered SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Clusters

Nodes are upgraded by the system upgrade controller running in the downstream cluster. Based on the cluster configuration, Rancher deploys two plans [https://github.com/rancher/system-upgrade-controller#example-upgrade-plan] to upgrade nodes: one for controlplane nodes and one for workers. The system upgrade controller follows the plans and upgrades the nodes.

To enable debug logging on the system upgrade controller deployment, edit the configmap [https://github.com/rancher/system-upgrade-controller/blob/50a4c8975543d75f1d76a8290001d87dc298bdb4/manifests/system-upgrade-controller.yaml#L32] to set the debug environment variable to true. Then restart the `system-upgrade-controller` pod.

Logs created by the `system-upgrade-controller` can be viewed by running this command:

```
kubectl logs -n cattle-system system-upgrade-controller
```

The current status of the plans can be viewed with this command:

```
kubectl get plans -A -o yaml
```



If the cluster becomes stuck in upgrading, restart the `system-upgrade-controller`.

To prevent issues when upgrading, the Kubernetes upgrade best practices [https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/] should be followed.

### Authorized Cluster Endpoint Support for SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Clusters

Rancher supports Authorized Cluster Endpoints (ACE) for registered RKE2 and K3s clusters. Enabling ACE requires manual configuration on the downstream cluster. For more information, see Authorized Cluster Endpoints.



- Perform these steps on each downstream control plane node individually.
- This procedure modifies the downstream cluster configuration and deploys `kube-api-authn-webhook`. If a future ACE release updates `kube-api-authn-webhook`, you must update it manually. For details, see About the Authentication Webhook.

To enable ACE, complete the following steps on each downstream control plane node:

1. Create `/var/lib/rancher/{rke2,k3s}/kube-api-authn-webhook.yaml` with the following

content:

```
apiVersion: v1
kind: Config
clusters:
- name: Default
  cluster:
    insecure-skip-tls-verify: true
    server: http://127.0.0.1:6440/v1/authenticate
users:
- name: Default
  user:
    insecure-skip-tls-verify: true
current-context: webhook
contexts:
- name: webhook
  context:
    user: Default
    cluster: Default
```

2. Add the following content to `/etc/rancher/{rke2,k3s}/config.yaml` (create the file if it does not exist):

```
kube-apiserver-arg:
  - authentication-token-webhook-config-
file=/var/lib/rancher/{rke2,k3s}/kube-api-authn-webhook.yaml
```

3. Restart the service:

```
sudo systemctl restart {rke2,k3s}-server
```

4. Complete ACE enablement in the Rancher UI:
   a. Navigate to the imported cluster and select ⬔ > **Edit Config**.
   b. Under **Cluster Configuration**, select the **Networking** tab.
   c. Select **Enabled** for **Authorized Endpoint**.
   d. Optional: Enter a fully qualified domain name (FQDN) or certificate details.



- If specified, the **FQDN** must point to the downstream cluster.
- Provide certificate information only if a load balancer in front of the downstream cluster uses an untrusted certificate. If the certificate is valid, leave **CA Certificates** blank.

## Annotating Registered Clusters

For all types of registered Kubernetes clusters except for RKE2 and K3s Kubernetes clusters, Rancher doesn't have any information about how the cluster is provisioned or configured.

Therefore, when Rancher registers a cluster, it assumes that several capabilities are disabled by default. Rancher assumes this in order to avoid exposing UI options to the user even when the capabilities are not enabled in the registered cluster.

However, if the cluster has a certain capability, a user of that cluster might still want to select the capability for the cluster in the Rancher UI. In order to do that, the user will need to manually indicate to Rancher that certain capabilities are enabled for the cluster.

By annotating a registered cluster, it is possible to indicate to Rancher that a cluster was given additional capabilities outside of Rancher.

The following annotation indicates Ingress capabilities. Note that the values of non-primitive objects need to be JSON encoded, with quotations escaped.

```
"capabilities.cattle.io/ingressCapabilities": "[
  {
    \"customDefaultBackend\": true,
    \"ingressProvider\": \"asdf\"
  }
]"
```

These capabilities can be annotated for the cluster:

- `ingressCapabilities`
- `loadBalancerCapabilities`
- `nodePoolScalingSupported`
- `nodePortRange`
- `taintSupport`

All the capabilities and their type definitions can be viewed in the Rancher API view, at `[Rancher Server URL]/v3/schemas/capabilities`.

To annotate a registered cluster,

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the custom cluster you want to annotate and click ☰ > **Edit Config**.
3. Expand the **Labels & Annotations** section.
4. Click **Add Annotation**.
5. Add an annotation to the cluster with the format `capabilities/<capability>: <value>` where `value` is the cluster capability that will be overridden by the annotation. In this scenario, Rancher is not aware of any capabilities of the cluster until you add the annotation.
6. Click **Save**.

**Result:** The annotation does not give the capabilities to the cluster, but it does indicate to Rancher that the cluster has those capabilities.

### Troubleshooting

This section lists some of the most common errors that may occur when importing a cluster and provides steps to troubleshoot them.

#### AKS

The following error may occur if local accounts are disabled in your cluster:

```
Error: Getting static credential is not allowed because this cluster is
set to disable local accounts.
```

To resolve this issue, enable local accounts before attempting to import the cluster again:

```
az aks update --resource-group <resource-group> --name <cluster-name>
--enable-local-accounts
```

## Registered Clusters

The commands/steps listed on this page can be used to check clusters that you are registering or that are registered in Rancher.

Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kubeconfig_from_imported_cluster.yml`)

### Rancher Agents

Communication to the cluster (Kubernetes API via cattle-cluster-agent) and communication to the nodes is done through Rancher agents.

If the cattle-cluster-agent cannot connect to the configured `server-url`, the cluster will remain in **Pending** state, showing `Waiting for full cluster configuration`.

#### cattle-node-agent



cattle-node-agents are only present in clusters created in Rancher with RKE.

Check if the cattle-node-agent pods are present on each node, have status **Running** and don't have a high count of Restarts:

```
kubectl -n cattle-system get pods -l app=cattle-agent -o wide
```

Example output:

```
NAME                          READY     STATUS    RESTARTS   AGE       IP
NODE
cattle-node-agent-4gc2p   1/1       Running   0          2h
x.x.x.x           worker-1
cattle-node-agent-8cxkk   1/1       Running   0          2h
x.x.x.x           etcd-1
cattle-node-agent-kzrlg   1/1       Running   0          2h
x.x.x.x           etcd-0
cattle-node-agent-nclz9   1/1       Running   0          2h
x.x.x.x           controlplane-0
cattle-node-agent-pwxp7   1/1       Running   0          2h
x.x.x.x           worker-0
cattle-node-agent-t5484   1/1       Running   0          2h
x.x.x.x           controlplane-1
cattle-node-agent-t8mtz   1/1       Running   0          2h
x.x.x.x           etcd-2
```

Check logging of a specific cattle-node-agent pod or all cattle-node-agent pods:

```
kubectl -n cattle-system logs -l app=cattle-agent
```

#### cattle-cluster-agent

Check if the cattle-cluster-agent pod is present in the cluster, has status **Running** and doesn't have a high count of Restarts:

```
kubectl -n cattle-system get pods -l app=cattle-cluster-agent -o wide
```

Example output:

```
NAME                                    READY     STATUS    RESTARTS
AGE       IP           NODE
cattle-cluster-agent-54d7c6c54d-ht9h4   1/1       Running   0
2h        x.x.x.x      worker-1
```

Check logging of cattle-cluster-agent pod:

```
kubectl -n cattle-system logs -l app=cattle-cluster-agent
```

# Cluster Administration

After you provision a cluster in Rancher, you can begin using powerful Kubernetes features to deploy and scale your containerized applications in development, testing, or production environments.



This section assumes a basic familiarity with Docker and Kubernetes. For a brief explanation of how Kubernetes components work together, refer to the concepts page.

## Managing Clusters in Rancher

After clusters have been provisioned into Rancher, cluster owners will need to manage these clusters. There are many different options of how to manage your cluster.

| Action | Rancher Launched Kubernetes Clusters | EKS, GKE and AKS Clusters[1] | Other Hosted Kubernetes Clusters | Non-EKS or GKE Registered Clusters |
|---|---|---|---|---|
| Using kubectl and a kubeconfig file to Access a Cluster | ⬕ | ⬕ | ⬕ | ⬕ |
| Managing Cluster Members | ⬕ | ⬕ | ⬕ | ⬕ |
| Editing and Upgrading Clusters | ⬕ | ⬕ | ⬕ | ⬕[2] |
| Managing Nodes | ⬕ | ⬕ | ⬕ | ⬕[3] |
| Managing Persistent Volumes and Storage Classes | ⬕ | ⬕ | ⬕ | ⬕ |
| Managing Projects, Namespaces and Workloads | ⬕ | ⬕ | ⬕ | ⬕ |
| Using App Catalogs | ⬕ | ⬕ | ⬕ | ⬕ |
| Configuring Tools (Alerts, Notifiers, Monitoring, Logging, Istio) | ⬕ | ⬕ | ⬕ | ⬕ |
| Running Security Scans | ⬕ | ⬕ | ⬕ | ⬕ |
| Ability to rotate certificates | ⬕ | ⬕ | | |
| Ability to backup and restore Rancher-launched clusters | ⬕ | ⬕ | | ⬕[4] |
| Cleaning Kubernetes components when clusters are no longer reachable from Rancher | ⬕ | | | |

1. Registered EKS, GKE and AKS clusters have the same options available as EKS, GKE and AKS clusters created from the Rancher UI. The difference is that when a registered cluster is deleted from the Rancher UI, it is not destroyed.
2. Cluster configuration options can't be edited for registered clusters, except for K3s and RKE2 clusters.

3. For registered cluster nodes, the Rancher UI exposes the ability to cordon, drain, and edit the node.
4. For registered clusters using etcd as a control plane, snapshots must be taken manually outside of the Rancher UI to use for backup and recovery.

# Configuring Tools

Rancher contains a variety of tools that aren't included in Kubernetes to assist in your DevOps operations. Rancher can integrate with external services to help your clusters run more efficiently. Tools are divided into the following categories:

- Alerts
- Notifiers
- Logging
- Monitoring
- Istio Service Mesh

Tools can be installed through **Apps.**

# Manage Clusters

## Best Practices for Disconnected Clusters

Rancher supports managing clusters that may not always be online due to network disruptions, control plane availability, or because all cluster nodes are down. At the moment there are no known issues with disconnected clusters in the latest released Rancher version.

While a managed cluster is disconnected from Rancher, management operations will be unavailable, and the Rancher UI will not allow navigation to the cluster. However, once the connection is reestablished, functionality is fully restored.

#### Best Practices for Managing Disconnected Clusters

- **Cluster Availability During Rancher Upgrades**: It is recommended to have all, or at least most, managed clusters online during a Rancher upgrade. The reason is that upgrading Rancher automatically upgrades the Rancher agent software running on managed clusters. Keeping the agent and Rancher versions aligned ensures consistent functionality. Any clusters that are disconnected during the upgrade will have their agents updated as soon as they reconnect.
- **Cleaning Up Disconnected Clusters**: Regularly remove clusters that will no longer reconnect to Rancher (e.g., clusters that have been decommissioned or destroyed). Keeping such clusters in the Rancher management system consumes unnecessary resources, which could impact Rancher's performance over time.
- **Certificate Rotation Considerations**: When designing processes that involve regularly shutting down clusters, whether connected to Rancher or not, take into account certificate rotation policies. For example, RKE2/K3s clusters may rotate certificates on startup if they exceeded their lifetime.

## Access Clusters

This section is about what tools can be used to access clusters managed by Rancher.

For information on how to give users permission to access a cluster, see the section on adding users to clusters.

For more information on roles-based access control, see this section.

For information on how to set up an authentication system, see this section.

### Clusters in Rancher UI

There are several paths to view and manage clusters through the Rancher UI.

#### Clusters Page

You can access the **Clusters** page from the ☰ menu:

1. Click ☰.
2. Select **Cluster Management**.

You can also access the **Clusters** page by clicking the **Manage** button above the clusters table on the Rancher UI **Home** page.

On the **Clusters** page, select ☰ at the end of each row to view a submenu with the following options:

- Kubectl Shell
- Download KubeConfig
- Copy KubeConfig to Clipboard
- Edit Config
- View YAML
- Download YAML

#### Cluster Dashboard

On the **Clusters** page, select the **Explore** button at the end of each row to view that cluster's **Cluster Dashboard**. You can also view the dashboard by clicking the name of a cluster in the table, then clicking the **Explore** button on the **Cluster** page.

The **Cluster Dashboard** is also accessible from the Rancher UI **Home** page, by clicking on the name of a cluster.

You can also access the **Cluster Dashboard** from the ☰ in the top navigation bar:

1. Click ☰.
2. Select the name of a cluster from the **Explore Cluster** menu option.

The **Cluster Dashboard** lists information about a specific cluster, such as number of nodes, memory usage, events, and resources.

### kubectl

You can use the Kubernetes command-line tool, kubectl [https://kubernetes.io/docs/reference/kubectl/overview/], to manage your clusters. You have two options for using kubectl:

- **Rancher kubectl shell:** Interact with your clusters by launching a kubectl shell available in the Rancher UI. This option requires no configuration actions on your part. For more information, see Accessing Clusters with kubectl Shell.
- **Terminal remote connection:** You can also interact with your clusters by installing kubectl [https://kubernetes.io/docs/tasks/tools/install-kubectl/] on your local desktop and then copying the cluster's kubeconfig file to your local `~/.kube/config` directory. For more information, see Accessing Clusters with kubectl and a kubeconfig File.

#### Rancher CLI

You can control your clusters by downloading Rancher's own command-line interface, Rancher CLI. This CLI tool can interact directly with different clusters and projects or pass them `kubectl` commands.

#### Rancher API

Finally, you can interact with your clusters over the Rancher API. Before you use the API, you must obtain an API key. To view the different resource fields and actions for an API object, open the API UI, which can be accessed by clicking on **View in API** for any Rancher UI object.

### Adding Users to Clusters

To provide a user access to view certain cluster-level resources and create new projects, assign the user a **Cluster Membership**. Cluster members can create projects and manage resources in those projects. However, not all resources, namespaces and workloads in a cluster are accessible by cluster members.



Want to provide a user with access to a *specific* project within a cluster? See Adding Project Members instead.

There are two contexts where you can add cluster members:

- Adding Members to a New Cluster

  You can add members to a cluster as you create it (recommended if possible).

- Adding Members to an Existing Cluster

  You can always add members to a cluster after a cluster is provisioned.

#### Editing Cluster Membership

Cluster administrators can edit the membership for a cluster, controlling which Rancher users can access the cluster and what features they can use.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster you want to add members to and click ☰ > **Edit Config**.
3. In the **Member Roles** tab, click **Add Member**.
4. Search for the user or group that you want to add to the cluster.



- At least 2 characters must be typed in the search box for results to appear.
- Users can be searched based on their username or display name.

- Search is prefix-based (e.g., a user named `Stan Dard` will appear when searching for `Sta`, but not when searching for `Dar`) and case-sensitive.

If external authentication is configured:

- Rancher returns users from your external authentication source as you type.



*Using AD but can't find your users?*

There may be an issue with your search attribute configuration. See Configuring Active Directory Authentication: Step 5.

- A drop-down allows you to add groups instead of individual users. The drop-down only lists groups that you, the logged in user, are part of.



If you are logged in as a local user, external users do not display in your search results. For more information, see External Authentication Configuration and Principal Users.

5. Assign the user or group **Cluster** roles.

   What are Cluster Roles?



For Custom Roles, you can modify the list of individual roles available for assignment.

- To add roles to the list, Add a Custom Role.
- To remove roles from the list, Lock/Unlock Roles.

**Result:** The chosen users are added to the cluster.

- To revoke cluster membership, select the user and click **Delete**. This action deletes membership, not the user.
- To modify a user's roles in the cluster, delete them from the cluster, and then re-add them with modified roles.

### How the Authorized Cluster Endpoint Works

This section describes how the kubectl CLI, the kubeconfig file, and the authorized cluster endpoint work together to allow you to access a downstream Kubernetes cluster directly, without authenticating through the Rancher server. It is intended to provide background information and context to the instructions for how to set up kubectl to directly access a cluster.

#### About the Kubeconfig File

The kubeconfig file is used to configure access to Kubernetes when used in conjunction with the kubectl command line tool (or other clients).

The kubeconfig file and its contents are specific to each cluster. It can be downloaded from the **Clusters** page in Rancher:

1. Click ☰ in the top left corner.
2. Select **Cluster Management**.

3. Find the cluster whose kubeconfig you want to download, and select ⬔ at the end of the row.
4. Select **Download KubeConfig** from the submenu.

You need a separate kubeconfig file for each cluster that you have access to in Rancher.

After you download the kubeconfig file, you are able to use the kubeconfig file and its Kubernetes contexts [https://kubernetes.io/docs/reference/kubectl/cheatsheet/#kubectl-context-and-configuration] to access your downstream cluster.

If admins have kubeconfig token generation turned off, the kubeconfig file requires that the Rancher CLI to be present in your PATH.

#### About the kube-api-auth Authentication Webhook

The `kube-api-auth` microservice is deployed to provide the user authentication functionality for the authorized cluster endpoint. When you access the user cluster using `kubectl`, the cluster's Kubernetes API server authenticates you by using the `kube-api-auth` service as a webhook.

During cluster provisioning, the file `/etc/kubernetes/kube-api-authn-webhook.yaml` is deployed and `kube-apiserver` is configured with `--authentication-token-webhook-config-file=/etc/kubernetes/kube-api-authn-webhook.yaml`. This configures the `kube-apiserver` to query `http://127.0.0.1:6440/v1/authenticate` to determine authentication for bearer tokens.

The scheduling rules for `kube-api-auth` are listed below:

| Component | nodeAffinity nodeSelectorTerms | nodeSelector | Tolerations |
|---|---|---|---|
| kube-api-auth | `beta.kubernetes.io/os:NotIn:windows`<br>`node-role.kubernetes.io/controlplane:In:"true"` | none | `operator:Exists` |

### Access a Cluster with Kubectl and kubeconfig

This section describes how to manipulate your downstream Kubernetes cluster with kubectl from the Rancher UI or from your workstation.

For more information on using kubectl, see Kubernetes Documentation: Overview of kubectl [https://kubernetes.io/docs/reference/kubectl/overview/].

#### Accessing Clusters with kubectl Shell in the Rancher UI

You can access and manage your clusters by logging into Rancher and opening the kubectl shell in the UI. No further configuration necessary.

1. Click ⬔ > **Cluster Management**.
2. Go to the cluster you want to access with kubectl and click **Explore**.
3. In the top navigation menu, click the **Kubectl Shell** button. Use the window that opens to interact with your Kubernetes cluster.

#### Accessing Clusters with kubectl from Your Workstation

This section describes how to download your cluster's kubeconfig file, launch kubectl from your workstation, and access your downstream cluster.

This alternative method of accessing the cluster allows you to authenticate with Rancher and manage your cluster without using the Rancher UI.



*Prerequisites:*

These instructions assume that you have already created a Kubernetes cluster, and that kubectl is installed on your workstation. For help installing kubectl, refer to the official Kubernetes documentation. [https://kubernetes.io/docs/tasks/tools/install-kubectl/]

1. Click ☰ in the top left corner.
2. Select **Cluster Management**.
3. Find the cluster whose kubeconfig you want to download, and select ☰ at the end of the row.
4. Select **Download KubeConfig** from the submenu.
5. Save the YAML file on your local computer. Move the file to `~/.kube/config`. Note: The default location that kubectl uses for the kubeconfig file is `~/.kube/config`, but you can use any directory and specify it using the `--kubeconfig` flag, as in this command:

   ```
   kubectl --kubeconfig /custom/path/kube.config get pods
   ```

6. From your workstation, launch kubectl. Use it to interact with your kubernetes cluster.

#### Note on Resources Created Using kubectl

Rancher will discover and show resources created by `kubectl`. However, these resources might not have all the necessary annotations on discovery. If an operation (for instance, scaling the workload) is done to the resource using the Rancher UI/API, this may trigger recreation of the resources due to the missing annotations. This should only happen the first time an operation is done to the discovered resource.

#### Authenticating Directly with a Downstream Cluster

This section intended to help you set up an alternative method to access a Rancher-launched cluster.

This method is only available RKE2 and K3s clusters that have the authorized cluster endpoint enabled. When Rancher creates the cluster, it generates a kubeconfig file that includes additional kubectl context(s) for accessing your cluster. This additional context allows you to use kubectl to authenticate with the downstream cluster without authenticating through Rancher. For a longer explanation of how the authorized cluster endpoint works, refer to this page.

On RKE2 and K3s clusters, you need to manually enable authorized cluster endpoints.

We recommend that as a best practice, you should set up this method to access your RKE2 and K3s clusters, so that just in case you can't connect to Rancher, you can still access the cluster.



*Prerequisites:*

The following steps assume that you have created a Kubernetes cluster and followed the

> steps to connect to your cluster with kubectl from your workstation.

To find the name of the context(s) in your downloaded kubeconfig file, run:

```
kubectl config get-contexts --kubeconfig /custom/path/kube.config
CURRENT   NAME                          CLUSTER
AUTHINFO      NAMESPACE
*         my-cluster                    my-cluster                    user-
46tmn
          my-cluster-controlplane-1     my-cluster-controlplane-1     user-
46tmn
```

In this example, when you use `kubectl` with the first context, `my-cluster`, you will be authenticated through the Rancher server.

With the second context, `my-cluster-controlplane-1`, you would authenticate with the authorized cluster endpoint, communicating with an downstream RKE2/K3s cluster directly.

We recommend using a load balancer with the authorized cluster endpoint. For details, refer to the recommended architecture section.

Now that you have the name of the context needed to authenticate directly with the cluster, you can pass the name of the context in as an option when running kubectl commands. The commands will differ depending on whether your cluster has an FQDN defined. Examples are provided in the sections below.

When `kubectl` works normally, it confirms that you can access your cluster while bypassing Rancher's authentication proxy.

#### Connecting Directly to Clusters with FQDN Defined

If an FQDN is defined for the cluster, a single context referencing the FQDN will be created. The context will be named `<CLUSTER_NAME>-fqdn`. When you want to use `kubectl` to access this cluster without Rancher, you will need to use this context.

Assuming the kubeconfig file is located at `~/.kube/config`:

```
kubectl --context <CLUSTER_NAME>-fqdn get nodes
```

Directly referencing the location of the kubeconfig file:

```
kubectl --kubeconfig /custom/path/kube.config --context <CLUSTER_NAME>-
fqdn get pods
```

#### Connecting Directly to Clusters without FQDN Defined

If there is no FQDN defined for the cluster, extra contexts will be created referencing the IP address of each node in the control plane. Each context will be named `<CLUSTER_NAME>-<NODE_NAME>`. When you want to use `kubectl` to access this cluster without Rancher, you will need to use this context.

Assuming the kubeconfig file is located at `~/.kube/config`:

```
kubectl --context <CLUSTER_NAME>-<NODE_NAME> get nodes
```

Directly referencing the location of the kubeconfig file:

```
kubectl --kubeconfig /custom/path/kube.config --context <CLUSTER_NAME>-
<NODE_NAME> get pods
```

## Removing Kubernetes Components from Nodes

This section describes how to disconnect a node from a Rancher-launched Kubernetes cluster and remove all of the Kubernetes components from the node. This process allows you to use the node for other purposes.

When you use Rancher to install Kubernetes on new nodes in an infrastructure provider, resources (containers/virtual network interfaces) and configuration items (certificates/configuration files) are created.

When removing nodes from your Rancher launched Kubernetes cluster (provided that they are in `Active` state), those resources are automatically cleaned, and the only action needed is to restart the node. When a node has become unreachable and the automatic cleanup process cannot be used, we describe the steps that need to be executed before the node can be added to a cluster again.

### What Gets Removed?

When cleaning nodes provisioned using Rancher, the following components are deleted based on the type of cluster node you're removing.

| Removed Component | Nodes Hosted by Infrastructure Provider | Custom Nodes | Hosted Cluster | Registered Nodes |
|---|---|---|---|---|
| The Rancher deployment namespace (`cattle-system` by default) | ⬕ | ⬕ | ⬕ | ⬕ |
| `serviceAccount`, `clusterRoles`, and `clusterRoleBindings` labeled by Rancher | ⬕ | ⬕ | ⬕ | ⬕ |
| Labels, Annotations, and Finalizers | ⬕ | ⬕ | ⬕ | ⬕ |
| Rancher Deployment | ⬕ | ⬕ | ⬕ | |
| Machines, clusters, projects, and user custom resource definitions (CRDs) | ⬕ | ⬕ | ⬕ | |
| All resources create under the `management.cattle.io` API Group | ⬕ | ⬕ | ⬕ | |

| Removed Component | Nodes Hosted by Infrastructure Provider | Custom Nodes | Hosted Cluster | Registered Nodes |
|---|---|---|---|---|
| All CRDs created by Rancher v2.x | ◪ | ◪ | ◪ | |

#### Removing a Node from a Cluster by Rancher UI

When the node is in `Active` state, removing the node from a cluster will trigger a process to clean up the node. Please restart the node after the automatic cleanup process is done to make sure any non-persistent data is properly removed.

**To restart a node:**

```
# using reboot
$ sudo reboot

# using shutdown
$ sudo shutdown -r now
```

#### Removing Rancher Components from a Cluster Manually

When a node is unreachable and removed from the cluster, the automatic cleaning process can't be triggered because the node is unreachable. Please follow the steps below to manually remove the Rancher components.



The commands listed below will remove data from the node. Make sure you have created a backup of files you want to keep before executing any of the commands as data will be lost.

##### Removing Rancher Components from Registered Clusters

For registered clusters, the process for removing Rancher is a little different. You have the option of simply deleting the cluster in the Rancher UI, or your can run a script that removes Rancher components from the nodes. Both options make the same deletions.

After the registered cluster is detached from Rancher, the cluster's workloads will be unaffected and you can access the cluster using the same methods that you did before the cluster was registered into Rancher.

**By UI / API**



This process will remove data from your cluster. Make sure you have created a backup of files you want to keep before executing the command, as data will be lost.

After you initiate the removal of a registered cluster using the Rancher UI (or API), the following events occur.

1. Rancher creates a `serviceAccount` that it uses to remove the Rancher components from the cluster. This account is assigned the clusterRole [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-and-clusterrole] and clusterRoleBinding [https://kubernetes.io/docs/reference/access-

authn-authz/rbac/#rolebinding-and-clusterrolebinding] permissions, which are required to remove the Rancher components.

2. Using the `serviceAccount`, Rancher schedules and runs a job [https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/] that cleans the Rancher components off of the cluster. This job also references the `serviceAccount` and its roles as dependencies, so the job deletes them before its completion.
3. Rancher is removed from the cluster. However, the cluster persists, running the native version of Kubernetes.

**Result:** All components listed for registered clusters in What Gets Removed? are deleted.

**By Script**

Rather than cleaning registered cluster nodes using the Rancher UI, you can run a script instead.



*Prerequisite*

Install kubectl [https://kubernetes.io/docs/tasks/tools/install-kubectl/].

1. Open a web browser, navigate to GitHub [https://github.com/rancher/rancher/blob/main/cleanup/user-cluster.sh], and download `user-cluster.sh`.
2. Make the script executable by running the following command from the same directory as `user-cluster.sh`:

```
chmod +x user-cluster.sh
```

3. **Air Gap Environments Only:** Open `user-cluster.sh` and replace `yaml_url` with the URL in `user-cluster.yml`.

   If you don't have an air gap environment, skip this step.

4. From the same directory, run the script and provide the `rancher/rancher-agent` image version which should be equal to the version of Rancher used to manage the cluster. (`<RANCHER_VERSION>`):



Add the `-dry-run` flag to preview the script's outcome without making changes.

```
./user-cluster.sh rancher/rancher-
agent:<RANCHER_VERSION>
```

**Result:** The script runs. All components listed for registered clusters in What Gets Removed? are deleted.

# Cleaning up Nodes

**RKE2**



For instructions on cleaning nodes in RKE2 clusters that weren't deployed by Rancher, see the official RKE2 documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/install/uninstall.html] on uninstalling clusters.

You need to remove the following components from Rancher-provisioned RKE2 nodes:

- The rancher-system-agent, which connects to Rancher and installs and manages RKE2.
- RKE2 itself.

```
<h3><a id="_removing_rancher-system-agent"></a>Removing rancher-system-agent</h3>
```

To remove the rancher-system-agent, run the system-agent-uninstall.sh [https://github.com/rancher/system-agent/blob/main/system-agent-uninstall.sh] script:

```
curl https://raw.githubusercontent.com/rancher/system-agent/main/system-agent-uninstall.sh > system-agent-uninstall.sh
sudo sh system-agent-uninstall.sh
```

```
<h3><a id="_removing_rke2"></a>Removing RKE2</h3>
```

To remove the RKE2 installation, run the `rke2-uninstall` script that is already present on the node:

```
sudo rke2-uninstall.sh
```

**K3s**

You need to remove the following components from Rancher-provisioned K3s nodes:

- The rancher-system-agent, which connects to Rancher and installs and manages K3s.
- K3s itself.

```
<h3><a id="_removing_rancher-system-agent"></a>Removing rancher-system-agent</h3>
```

To remove the rancher-system-agent, run the system-agent-uninstall.sh [https://github.com/rancher/system-agent/blob/main/system-agent-uninstall.sh] script:

```
curl https://raw.githubusercontent.com/rancher/system-agent/main/system-agent-uninstall.sh > system-agent-uninstall.sh
sudo sh system-agent-uninstall.sh
```

```
<h3><a id="_removing_k3s"></a>Removing K3s</h3>
```

To remove the K3s installation, run the `k3s-uninstall` script that is already present on the node:

```
sudo k3s-uninstall.sh
```

#### Directories and Files

The following directories are used when adding a node to a cluster, and should be removed. You can remove a directory using `rm -rf /directory_name`.



Depending on the role you assigned to the node, certain directories may or may not be present on the node.

**RKE2**

| Directories |
| --- |
| /etc/ceph |
| /etc/cni |
| /etc/kubernetes |
| /etc/rancher |
| /opt/cni |
| /run/calico |
| /run/flannel |
| /run/secrets/kubernetes.io |
| /var/lib/calico |
| /var/lib/cni |
| /var/lib/etcd |
| /var/lib/kubelet |
| /var/lib/rancher |
| /var/lib/weave |
| /var/log/containers |
| /var/log/pods |
| /var/run/calico |

**To clean the directories:**

```
rm -rf /etc/ceph \
       /etc/cni \
       /etc/kubernetes \
       /etc/rancher \
       /opt/cni \
       /run/calico \
       /run/flannel \
       /run/secrets/kubernetes.io \
```

```
        /var/lib/calico \
        /var/lib/cni \
        /var/lib/etcd \
        /var/lib/kubelet \
        /var/lib/rancher \
        /var/lib/weave \
        /var/log/containers \
        /var/log/pods \
        /var/run/calico
```

**K3s**

| Directories |
| --- |
| /etc/ceph |
| /etc/cni |
| /etc/kubernetes |
| /etc/rancher |
| /etc/systemd/system/k3s |
| /opt/cni |
| /run/calico |
| /run/flannel |
| /run/secrets/kubernetes.io |
| /usr/local/bin/k3s |
| /var/lib/calico |
| /var/lib/cni |
| /var/lib/etcd |
| /var/lib/kubelet |
| /var/lib/rancher |
| /var/lib/weave |
| /var/log/containers |
| /var/log/pods |
| /var/run/calico |

**To clean the directories:**

```
rm -rf /etc/ceph \
        /etc/cni \
        /etc/kubernetes \
        /etc/rancher \
        /etc/systemd/system/k3s \
        /opt/cni \
        /run/calico \
        /run/flannel \
```

```
        /run/secrets/kubernetes.io \
        /usr/local/bin/k3s \
        /var/lib/calico \
        /var/lib/cni \
        /var/lib/etcd \
        /var/lib/kubelet \
        /var/lib/rancher \
        /var/lib/weave \
        /var/log/containers \
        /var/log/pods \
        /var/run/calico
```

#### Network Interfaces and Iptables

The remaining two components that are changed/configured are (virtual) network interfaces and iptables rules. Both are non-persistent to the node, meaning that they will be cleared after a restart of the node. To remove these components, a restart is recommended.

**To restart a node:**

```
# using reboot
$ sudo reboot

# using shutdown
$ sudo shutdown -r now
```

If you want to know more on (virtual) network interfaces or iptables rules, please see the specific subjects below.

#### Network Interfaces



Depending on the network provider configured for the cluster the node was part of, some of the interfaces will or won't be present on the node.

| Interfaces |
| --- |
| `flannel.1` |
| `cni0` |
| `tunl0` |
| `caliXXXXXXXXXXX` (random interface names) |
| `vethXXXXXXXX` (random interface names) |

**To list all interfaces:**

```
# Using ip
ip address show
```

```
# Using ifconfig
ifconfig -a
```

**To remove an interface:**

```
ip link delete interface_name
```

#### Iptables



Depending on the network provider configured for the cluster the node was part of, some of the chains will or won't be present on the node.

Iptables rules are used to route traffic from and to containers. The created rules are not persistent, so restarting the node will restore iptables to its original state.

| Chains |
| --- |
| `cali-failsafe-in` |
| `cali-failsafe-out` |
| `cali-fip-dnat` |
| `cali-fip-snat` |
| `cali-from-hep-forward` |
| `cali-from-host-endpoint` |
| `cali-from-wl-dispatch` |
| `cali-fw-caliXXXXXXXXXXX` (random chain names) |
| `cali-nat-outgoing` |
| `cali-pri-kns.NAMESPACE` (chain per namespace) |
| `cali-pro-kns.NAMESPACE` (chain per namespace) |
| `cali-to-hep-forward` |
| `cali-to-host-endpoint` |
| `cali-to-wl-dispatch` |
| `cali-tw-caliXXXXXXXXXXX` (random chain names) |
| `cali-wl-to-host` |
| `KUBE-EXTERNAL-SERVICES` |
| `KUBE-FIREWALL` |
| `KUBE-MARK-DROP` |
| `KUBE-MARK-MASQ` |
| `KUBE-NODEPORTS` |
| `KUBE-SEP-XXXXXXXXXXXXXXXX` (random chain names) |
| `KUBE-SERVICES` |
| `KUBE-SVC-XXXXXXXXXXXXXXXX` (random chain names) |

**To list all iptables rules:**

```
iptables -L -t nat
iptables -L -t mangle
iptables -L
```

## Create Kubernetes Persistent Volumes and Storage Classes

When deploying an application that needs to retain data, you'll need to create persistent storage. Persistent storage allows you to store application data external from the pod running your application. This storage practice allows you to maintain application data, even if the application's pod fails.

The documents in this section assume that you understand the Kubernetes concepts of persistent volumes, persistent volume claims, and storage classes. For more information, refer to the section on how storage works.

### Prerequisites

To set up persistent storage, the `Manage Volumes` role is required.

If you are provisioning storage for a cluster hosted in the cloud, the storage and cluster hosts must have the same cloud provider.

For provisioning new storage with Rancher, the cloud provider must be enabled. For details on enabling cloud providers, refer to this page.

For attaching existing persistent storage to a cluster, the cloud provider does not need to be enabled.

### Setting up Existing Storage

The overall workflow for setting up existing storage is as follows:

1. Set up your persistent storage. This may be storage in an infrastructure provider, or it could be your own storage.
2. Add a persistent volume (PV) that refers to the persistent storage.
3. Add a persistent volume claim (PVC) that refers to the PV.
4. Mount the PVC as a volume in your workload.

For details and prerequisites, refer to this page.

### Dynamically Provisioning New Storage in Rancher

The overall workflow for provisioning new storage is as follows:

1. Add a StorageClass and configure it to use your storage provider. The StorageClass could refer to storage in an infrastructure provider, or it could refer to your own storage.
2. Add a persistent volume claim (PVC) that refers to the storage class.
3. Mount the PVC as a volume for your workload.

For details and prerequisites, refer to this page.

#### SUSE Storage Storage

SUSE Storage [https://documentation.suse.com/cloudnative/storage/1.11/en/longhorn-documentation.html] is a lightweight, reliable and easy-to-use distributed block storage system for Kubernetes.

Longhorn is free, open source software. Originally developed by Rancher Labs, it is now being developed as a sandbox project of the Cloud Native Computing Foundation. It can be installed on any Kubernetes cluster with Helm, with kubectl, or with the Rancher UI.

If you have a pool of block storage, Longhorn can help you provide persistent storage to your Kubernetes cluster without relying on cloud providers. For more information about Longhorn features, refer to the documentation. [https://documentation.suse.com/cloudnative/storage/1.11/en/introduction/introduction.html]

Rancher v2.5 simplified the process of installing Longhorn on a Rancher-managed cluster. For more information, see Cloud Native Storage with Longhorn.

#### Provisioning Storage Examples

We provide examples of how to provision storage with NFS, vSphere, and Amazon's EBS.

#### GlusterFS Volumes

In clusters that store data on GlusterFS volumes, you may experience an issue where pods fail to mount volumes after restarting the `kubelet`. For details on preventing this from happening, refer to this page.

#### iSCSI Volumes

In Rancher Launched Kubernetes clusters that store data on iSCSI volumes, you may experience an issue where kubelets fail to automatically connect with iSCSI volumes. For details on resolving this issue, refer to this page.

#### hostPath Volumes

Both K3s and RKE2 support mounting hostPath volumes using the Rancher Local Path Provisioner [https://github.com/rancher/local-path-provisioner]. For configuration information, depending on your distribution refer to K3s - Volumes and Storage [https://documentation.suse.com/cloudnative/k3s/latest/en/add-ons/storage.html#_setting_up_the_local_storage_provider] or RKE2 - Advanced Options and Configuration [https://documentation.suse.com/cloudnative/rke2/latest/en/advanced.html#_extra_control_plane_component_volume_mounts].

#### Migrating VMware vSphere Cloud Provider from In-tree to Out-of-tree

Kubernetes is moving away from maintaining cloud providers in-tree. vSphere has an out-of-tree cloud provider that can be used by installing the vSphere cloud provider and cloud storage plugins.

For instructions on how to migrate from the in-tree vSphere cloud provider to out-of-tree, and manage the existing VMs post migration, refer to this page.

#### Related Links

- Kubernetes Documentation: Storage [https://kubernetes.io/docs/concepts/storage/]

### GlusterFS Volumes



This section only applies to RKE clusters.

In clusters that store data on GlusterFS volumes, you may experience an issue where pods fail to mount volumes after restarting the `kubelet`. The logging of the `kubelet` will show: `transport endpoint is not connected`. To prevent this from happening, you can configure your cluster to mount the `systemd-run` binary in the `kubelet` container. There are two requirements before you can change the cluster configuration:

- The node needs to have the `systemd-run` binary installed (this can be checked by using the command `which systemd-run` on each cluster node)
- The `systemd-run` binary needs to be compatible with Debian OS on which the hyperkube image is based (this can be checked using the following command on each cluster node, replacing the image tag with the Kubernetes version you want to use)

```
docker run -v /usr/bin/systemd-run:/usr/bin/systemd-run -v
/usr/lib/x86_64-linux-gnu/libcrypto.so.3:/usr/lib/x86_64-linux-
gnu/libcrypto.so.3 -v /lib/systemd/libsystemd-shared-
249.so:/lib/systemd/libsystemd-shared-249.so --entrypoint
/usr/bin/systemd-run rancher/hyperkube:v1.26.14-rancher1 --version
```



Before updating your Kubernetes YAML to mount the `systemd-run` binary, make sure the `systemd` package is installed on your cluster nodes. If this package isn't installed *before* the bind mounts are created in your Kubernetes YAML, Docker will automatically create the directories and files on each node and will not allow the package install to succeed.

```
services:
  kubelet:
    extra_binds:
      - "/usr/bin/systemd-run:/usr/bin/systemd-run"
      - "/usr/lib/x86_64-linux-gnu/libcrypto.so.3:/usr/lib/x86_64-linux-
gnu/libcrypto.so.3"
      - "/lib/systemd/libsystemd-shared-249.so:/lib/systemd/libsystemd-
shared-249.so"
```

After the cluster has finished provisioning, you can check the `kubelet` container logging to see if the functionality is activated by looking for the following logline:

```
Detected OS with systemd
```

### How Persistent Storage Works

A persistent volume (PV) is a piece of storage in the Kubernetes cluster, while a persistent volume claim

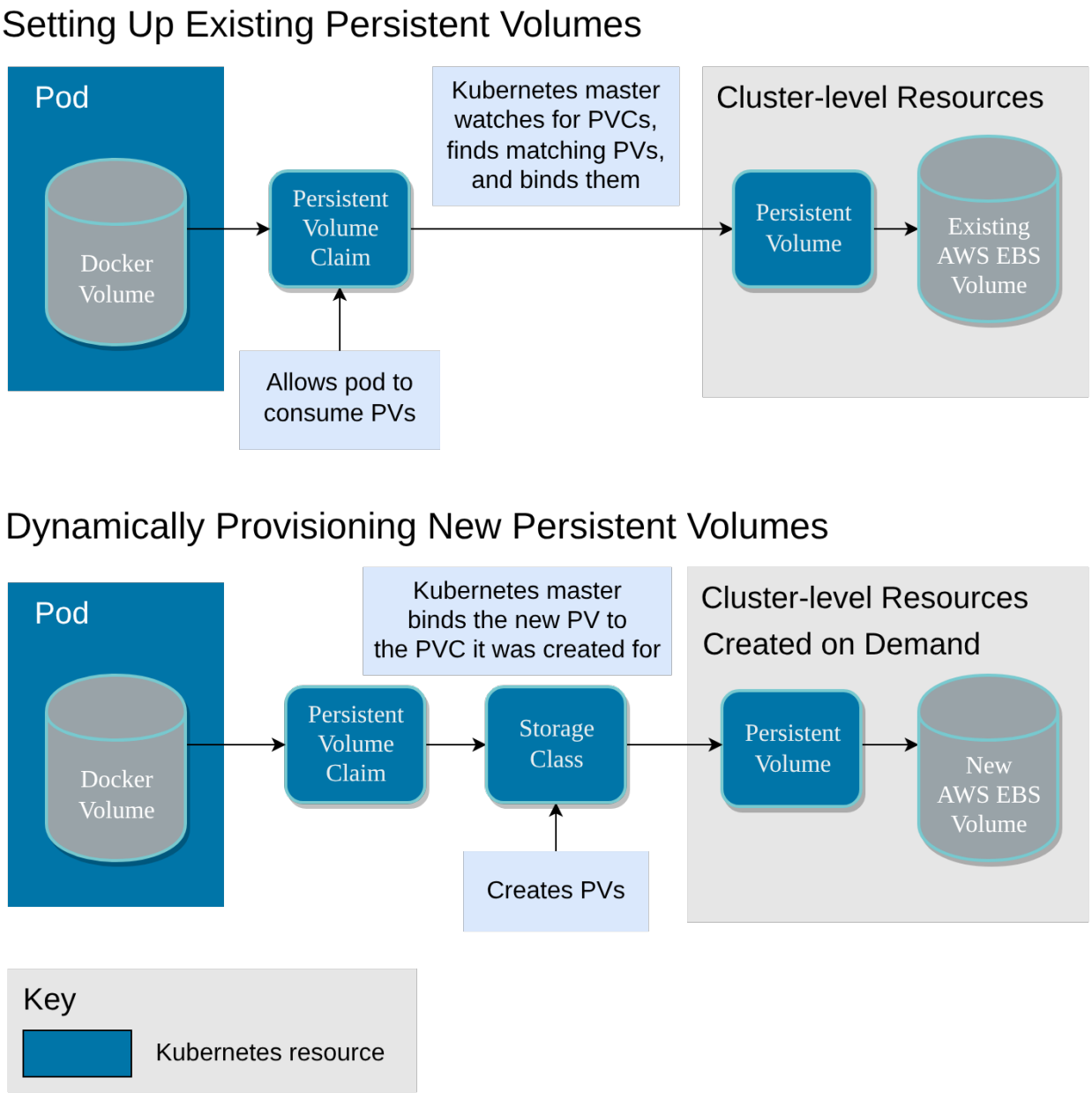
(PVC) is a request for storage.

There are two ways to use persistent storage in Kubernetes:

- Use an existing persistent volume
- Dynamically provision new persistent volumes

To use an existing PV, your application will need to use a PVC that is bound to a PV, and the PV should include the minimum resources that the PVC requires.

For dynamic storage provisioning, your application will need to use a PVC that is bound to a storage class. The storage class contains the authorization to provision new persistent volumes.




For more information, refer to the official Kubernetes documentation on storage [https://kubernetes.io/docs/concepts/storage/volumes/]

#### About Persistent Volume Claims

Persistent volume claims (PVCs) are objects that request storage resources from your cluster. They're similar to a voucher that your deployment can redeem for storage access. A PVC is mounted into a workloads as a

volume so that the workload can claim its specified share of the persistent storage.

To access persistent storage, a pod must have a PVC mounted as a volume. This PVC lets your deployment application store its data in an external location, so that if a pod fails, it can be replaced with a new pod and continue accessing its data stored externally, as though an outage never occurred.

Each Rancher project contains a list of PVCs that you've created, available from **Resources › Workloads › Volumes**. You can reuse these PVCs when creating deployments in the future.

#### PVCs are Required for Both New and Existing Persistent Storage

A PVC is required for pods to use any persistent storage, regardless of whether the workload is intended to use storage that already exists, or the workload will need to dynamically provision new storage on demand.

If you are setting up existing storage for a workload, the workload mounts a PVC, which refers to a PV, which corresponds to existing storage infrastructure.

If a workload should request new storage, the workload mounts PVC, which refers to a storage class, which has the capability to create a new PV along with its underlying storage infrastructure.

Rancher lets you create as many PVCs within a project as you'd like.

You can mount PVCs to a deployment as you create it, or later, after the deployment is running.

#### Setting up Existing Storage with a PVC and PV

Your pods can store data in volumes, [https://kubernetes.io/docs/concepts/storage/volumes/] but if the pod fails, that data is lost. To solve this issue, Kubernetes offers persistent volumes (PVs), which are Kubernetes resources that correspond to external storage disks or file systems that your pods can access. If a pod crashes, its replacement pod can access the data in persistent storage without any data loss.

PVs can represent a physical disk or file system that you host on premise, or a vendor-hosted storage resource, such as Amazon EBS or Azure Disk.

Creating a persistent volume in Rancher will not create a storage volume. It only creates a Kubernetes resource that maps to an existing volume. Therefore, before you can create a persistent volume as a Kubernetes resource, you must have storage provisioned.



PVs are created at the cluster level, which means that in a multi-tenant cluster, teams with access to separate namespaces could have access to the same PV.

#### Binding PVs to PVCs

When pods are set up to use persistent storage, they mount a persistent volume claim (PVC) that is mounted the same way as any other Kubernetes volume. When each PVC is created, the Kubernetes master considers it to be a request for storage and binds it to a PV that matches the minimum resource requirements of the PVC. Not every PVC is guaranteed to be bound to a PV. According to the Kubernetes documentation, [https://kubernetes.io/docs/concepts/storage/persistent-volumes/]

> Claims will remain unbound indefinitely if a matching volume does not exist. Claims will be bound as matching volumes become available. For example, a cluster provisioned with many 50Gi PVs would not match a PVC requesting 100Gi.

> The PVC can be bound when a 100Gi PV is added to the cluster.

In other words, you can create unlimited PVCs, but they will only be bound to PVs if the Kubernetes master can find a sufficient PV that has at least the amount of disk space required by the PVC.

To dynamically provision new storage, the PVC mounted in the pod would have to correspond to a storage class instead of a persistent volume.

#### Provisioning New Storage with a PVC and Storage Class

Storage Classes allow you to create PVs dynamically without having to create persistent storage in an infrastructure provider first.

For example, if a workload is bound to a PVC and the PVC refers to an Amazon EBS Storage Class, the storage class can dynamically create an EBS volume and a corresponding PV.

The Kubernetes master will then bind the newly created PV to your workload's PVC, allowing your workload to use the persistent storage.

### Dynamically Provisioning New Storage in SUSE Rancher Prime

This section describes how to provision new persistent storage for workloads in Rancher.

This section assumes that you understand the Kubernetes concepts of storage classes and persistent volume claims. For more information, refer to the section on how storage works.

New storage is often provisioned by a cloud provider such as Amazon EBS. However, new storage doesn't have to be in the cloud.

If you have a pool of block storage, and you don't want to use a cloud provider, Longhorn could help you provide persistent storage to your Kubernetes cluster. For more information, see Cloud Native Storage with Longhorn.

To provision new storage for your workloads, follow these steps:

1. Add a storage class and configure it to use your storage.
2. Use the Storage Class for Pods Deployed with a StatefulSet.

#### Prerequisites

- To set up persistent storage, the `Manage Volumes` role is required.
- If you are provisioning storage for a cluster hosted in the cloud, the storage and cluster hosts must have the same cloud provider.
- The cloud provider must be enabled. For details on enabling cloud providers, refer to this page.
- Make sure your storage provisioner is available to be enabled.

The following storage provisioners are enabled by default:

| Name | Plugin |
|---|---|
| Amazon EBS Disk | `aws-ebs` |
| AzureFile | `azure-file` |

| Name | Plugin |
|---|---|
| AzureDisk | `azure-disk` |
| Google Persistent Disk | `gce-pd` |
| VMware vSphere Volume | `vsphere-volume` |
| Local | `local` |
| Network File System | `nfs` |
| hostPath | `host-path` |

To use a storage provisioner that is not on the above list, you will need to use a feature flag to enable unsupported storage drivers.

#### 1. Add a storage class and configure it to use your storage

These steps describe how to set up a storage class at the cluster level.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you want to dynamically provision persistent storage volumes and click **Explore**.
3. Click **Storage** › **Storage Classes**.
4. Click **Create**.
5. Enter a name for your storage class.
6. From the **Provisioner** drop-down, select the service that you want to use to dynamically provision storage volumes. For example, if you have a Amazon EC2 cluster and you want to use cloud storage for it, use the `Amazon EBS Disk` provisioner.
7. In the **Parameters** tab, fill out the information required for the service to dynamically provision storage volumes. Each provisioner requires different information to dynamically provision storage volumes. Consult the service's documentation for help on how to obtain this information.
8. Click **Create**.

**Result:** The storage class is available to be consumed by a PVC.

For full information about the storage class parameters, refer to the official Kubernetes documentation [https://kubernetes.io/docs/concepts/storage/storage-classes/#parameters].

#### 2. Use the Storage Class for Pods Deployed with a StatefulSet

StatefulSets manage the deployment and scaling of Pods while maintaining a sticky identity for each Pod. In this StatefulSet, we will configure a VolumeClaimTemplate. Each Pod managed by the StatefulSet will be deployed with a PersistentVolumeClaim based on this VolumeClaimTemplate. The PersistentVolumeClaim will refer to the StorageClass that we created. Therefore, when each Pod managed by the StatefulSet is deployed, it will be bound to dynamically provisioned storage using the StorageClass defined in its PersistentVolumeClaim.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you want to add use the StorageClass for a workload and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Click **Create**.

5. Click **StatefulSet**.
6. In the **Volume Claim Templates** tab, click **Add Claim Template**.
7. Enter a name for the persistent volume.
8. In the **StorageClass** field, select the StorageClass that will dynamically provision storage for pods managed by this StatefulSet.
9. In the **Mount Point** field, enter the path that the workload will use to access the volume.
10. Click **Launch**.

**Result:** When each Pod managed by the StatefulSet is deployed, it will make a request for the specified amount of disk space to the Kubernetes master. If a PV with the specified resources is available when the workload is deployed, the Kubernetes master will bind the PV to Pod with a compatible PVC.

To attach the PVC to an existing workload,

1. Click ☰ **> Cluster Management**.
2. Go to the cluster where you want to add use the StorageClass for a workload and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Go to the workload that will use storage provisioned with the StorageClass that you cared at click ☰ **> Edit Config**.
5. In the **Volume Claim Templates** section, click **Add Claim Template**.
6. Enter a persistent volume name.
7. In the **StorageClass** field, select the StorageClass that will dynamically provision storage for pods managed by this StatefulSet.
8. In the **Mount Point** field, enter the path that the workload will use to access the volume.
9. Click **Save**.

**Result:** The workload will make a request for the specified amount of disk space to the Kubernetes master. If a PV with the specified resources is available when the workload is deployed, the Kubernetes master will bind the PV to the PVC. If not, Rancher will provision new persistent storage.

#### iSCSI Volumes

In Rancher Launched Kubernetes clusters that store data on iSCSI volumes, you may experience an issue where kubelets fail to automatically connect with iSCSI volumes. This failure is likely due to an incompatibility issue involving the iSCSI initiator tool. You can resolve this issue by installing the iSCSI initiator tool on each of your cluster nodes.

Rancher Launched Kubernetes clusters storing data on iSCSI volumes leverage the iSCSI initiator tool [http://www.open-iscsi.com/], which is embedded in the kubelet's `rancher/hyperkube` Docker image. From each kubelet (i.e., the *initiator*), the tool discovers and launches sessions with an iSCSI volume (i.e., the *target*). However, in some instances, the versions of the iSCSI initiator tool installed on the initiator and the target may not match, resulting in a connection failure.

If you encounter this issue, you can work around it by installing the initiator tool on each node in your cluster. You can install the iSCSI initiator tool by logging into your cluster nodes and entering one of the following commands:

| Platform | Package Name | Install Command |
|---|---|---|
| Ubuntu/Debian | `open-iscsi` | `sudo apt install open-iscsi` |
| RHEL | `iscsi-initiator-utils` | `yum install iscsi-initiator-utils -y` |

After installing the initiator tool on your nodes, edit the YAML for your cluster, editing the kubelet configuration to mount the iSCSI binary and configuration, as shown in the sample below.



- Before updating your Kubernetes YAML to mount the iSCSI binary and configuration, make sure either the `open-iscsi` (deb) or `iscsi-initiator-utils` (yum) package is installed on your cluster nodes. If this package isn't installed *before* the bind mounts are created in your Kubernetes YAML, Docker will automatically create the directories and files on each node and will not allow the package install to succeed.
- The example YAML below does not apply to K3s, but only to RKE clusters. Since the K3s kubelet does not run in a container, adding extra binds is not necessary. However, all iSCSI tools must still be installed on your K3s nodes.

```
services:
  kubelet:
    extra_binds:
      - "/etc/iscsi:/etc/iscsi"
      - "/sbin/iscsiadm:/sbin/iscsiadm"
```

If you're using RHEL 7.9, you need to mount the file `/usr/lib64/libcrypto.so.10` as shown in the sample below.

```
services:
  kubelet:
    extra_binds:
      - "/etc/iscsi:/etc/iscsi"
      - "/sbin/iscsiadm:/sbin/iscsiadm"
      - "/usr/lib64/libcrypto.so.10:/usr/lib/libcrypto.so.10"
```

If you're using RHEL 8.6, you need to mount the file `/usr/lib64/libcrypto.so.1.1` as shown in the sample below.

```
services:
  kubelet:
    extra_binds:
      - "/etc/iscsi:/etc/iscsi"
      - "/sbin/iscsiadm:/sbin/iscsiadm"
      - "/usr/lib64/libcrypto.so.1.1:/usr/lib/libcrypto.so.1.1"
```



To verify iscsiadm is working properly, you can run the command `docker exec kubelet iscsiadm --version` on any node.

#### Setting up Existing Storage

This section describes how to set up existing persistent storage for workloads in Rancher.



This section assumes that you understand the Kubernetes concepts of persistent volumes and persistent volume claims. For more information, refer to the section on how storage works.

To set up storage, follow these steps:

1. Set up persistent storage.
2. Add a PersistentVolume that refers to the persistent storage.
3. Use the Storage Class for Pods Deployed with a StatefulSet.

##### Prerequisites

- To create a persistent volume as a Kubernetes resource, you must have the `Manage Volumes` role.
- If you are provisioning storage for a cluster hosted in the cloud, the storage and cluster hosts must have the same cloud provider.

##### 1. Set up persistent storage

Creating a persistent volume in Rancher will not create a storage volume. It only creates a Kubernetes resource that maps to an existing volume. Therefore, before you can create a persistent volume as a Kubernetes resource, you must have storage provisioned.

The steps to set up a persistent storage device will differ based on your infrastructure. We provide examples of how to set up storage using vSphere, NFS, or Amazon's EBS.

If you have a pool of block storage, and you don't want to use a cloud provider, Longhorn could help you provide persistent storage to your Kubernetes cluster. For more information, see Cloud Native Storage with Longhorn.

##### 2. Add a PersistentVolume that refers to the persistent storage

These steps describe how to set up a PersistentVolume at the cluster level in Kubernetes.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you want to add a persistent volume and click **Explore**.
3. In the left navigation bar, click **Storage** › **Persistent Volumes**.
4. Click **Create**.
5. Enter a **Name** for the persistent volume.
6. Select the **Volume Plugin** for the disk type or service that you're using. When adding storage to a cluster that's hosted by a cloud provider, use the cloud provider's plug-in for cloud storage. For example, if you have a Amazon EC2 cluster and you want to use cloud storage for it, you must use the `Amazon EBS Disk` volume plugin.

7. Enter the **Capacity** of your volume in gigabytes.
8. Complete the **Plugin Configuration** form. Each plugin type requires information specific to the vendor of disk type. For help regarding each plugin's form and the information that's required, refer to the plug-in's vendor documentation.
9. Optional: In the **Customize** form, configure the access modes. [https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes] This options sets how many nodes can access the volume, along with the node read/write permissions. The Kubernetes Documentation [https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes] includes a table that lists which access modes are supported by the plugins available.
10. Optional: In the **Customize** form, configure the mount options. [https://kubernetes.io/docs/concepts/storage/persistent-volumes/#mount-options] Each volume plugin allows you to specify additional command line options during the mounting process. Consult each plugin's vendor documentation for the mount options available.
11. Click **Create**.

**Result:** Your new persistent volume is created.

#### 3. Use the Storage Class for Pods Deployed with a StatefulSet

StatefulSets manage the deployment and scaling of Pods while maintaining a sticky identity for each Pod. In this StatefulSet, we will configure a VolumeClaimTemplate. Each Pod managed by the StatefulSet will be deployed with a PersistentVolumeClaim based on this VolumeClaimTemplate. The PersistentVolumeClaim will refer to the PersistentVolume that we created. Therefore, when each Pod managed by the StatefulSet is deployed, it will be bound a PersistentVolume as defined in its PersistentVolumeClaim.

You can configure storage for the StatefulSet during or after workload creation.

The following steps describe how to assign existing storage to a new StatefulSet:

1. Click ☰ **> Cluster Management**.
2. Go to the cluster where you want to configure storage for the StatefulSet and click **Explore**.
3. In the left navigation bar, click **Workload › StatefulSets**.
4. Click **Create**.
5. Choose the namespace where the workload will be deployed.
6. Enter a name for the StatefulSet.
7. On the **Volume Claim Templates** tab, click **Add Claim Template**.
8. Click **Use an existing Persistent Volume**.
9. In the Persistent Volumes field, select the Persistent Volume that you created.
10. In the **Mount Point** field, enter the path that the workload will use to access the volume.
11. Click **Launch**.

**Result:** When the workload is deployed, it will make a request for the specified amount of disk space to the Kubernetes master. If a PV with the specified resources is available when the workload is deployed, the Kubernetes master will bind the PV to the PVC.

The following steps describe how to assign persistent storage to an existing workload:

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you want to configure storage for the StatefulSet and click **Explore**.
3. In the left navigation bar, click **Workload** › **StatefulSets**.
4. Go to the workload that you want to add the persistent storage to. Click ☰ > **Edit**.
5. On the **Volume Claim Templates** tab, click **Add Claim Template**.
6. Click **Use an existing Persistent Volume**.
7. In the Persistent Volumes field, select the Persistent Volume that you created.
8. In the **Mount Point** field, enter the path that the workload will use to access the volume.
9. Click **Launch**.

**Result:** The workload will make a request for the specified amount of disk space to the Kubernetes master. If a PV with the specified resources is available when the workload is deployed, the Kubernetes master will bind the PV to the PVC.

### Using an External Ceph Driver

These instructions are about using the external Ceph driver in an RKE2 cluster. If you are using RKE, additional steps are required. For details, refer to this section.

#### Requirements

Make sure ceph-common and xfsprogs packages are installed on SLE worker nodes.

#### Using the Ceph Driver with RKE

The resources below are fully compatible with RKE based clusters, but there is a need to do an additional kubelet configuration for RKE.

On RKE clusters, the kubelet component is running in a Docker container and doesn't have access to the host's kernel modules as rbd and libceph by default.

To solve this limitation, you can either run `modprobe rbd` on worker nodes, or configure the kubelet containers to automatically mount the `/lib/modules` directory from the host into the container.

For the kubelet configuration, put the following lines into the `cluster.yml` file prior to RKE cluster provisioning. You can also modify the `cluster.yml` later in the Rancher UI by clicking on **Edit Cluster** › **Edit as YAML** and restarting the worker nodes.

```
services:
  kubelet:
    extra_binds:
      - '/lib/modules:/lib/modules:ro'
```

For more information about the `extra_binds` directive, refer to this section. [https://rancher.com/docs/rke/latest/en/config-options/services/services-extras/#extra-binds]

#### Installing the ceph-csi driver on an SUSE® Rancher Prime: RKE2 cluster



These steps are needed for dynamic RBD provisioning only.

For more information about the `ceph-csi-rbd` chart, refer to this page. [https://github.com/ceph/ceph-csi/blob/devel/charts/ceph-csi-rbd/README.md]

To get details about your SES cluster, run:

```
ceph mon dump
```

Read its output:

```
dumped monmap epoch 3
epoch 3
fsid 79179d9d-98d8-4976-ab2e-58635caa7235
last_changed 2021-02-11T10:56:42.110184+0000
created 2021-02-11T10:56:22.913321+0000
min_mon_release 15 (octopus)
0: [v2:10.85.8.118:3300/0,v1:10.85.8.118:6789/0] mon.a
1: [v2:10.85.8.123:3300/0,v1:10.85.8.123:6789/0] mon.b
2: [v2:10.85.8.124:3300/0,v1:10.85.8.124:6789/0] mon.c
```

Later you'll need the fsid and mon addresses values.

#### Install the ceph-csi Driver Using Helm

Run these commands:

```
helm repo add ceph-csi https://ceph.github.io/csi-charts
helm repo update
helm search repo ceph-csi -l
helm inspect values ceph-csi/ceph-csi-rbd > ceph-csi-rbd-values.yaml
```

Modify the `ceph-csi-rbd-values.yaml` file and keep there only the required changes:

```
# ceph-csi-rbd-values.yaml
csiConfig:
  - clusterID: "79179d9d-98d8-4976-ab2e-58635caa7235"
    monitors:
      - "10.85.8.118:6789"
      - "10.85.8.123:6789"
      - "10.85.8.124:6789"
provisioner:
  name: provisioner
```

```
  replicaCount: 2
```

Make sure the ceph monitors are reachable from the RKE2 cluster, for example, by ping.

```
kubectl create namespace ceph-csi-rbd
helm install --namespace ceph-csi-rbd ceph-csi-rbd ceph-csi/ceph-csi-rbd
--values ceph-csi-rbd-values.yaml
kubectl rollout status deployment ceph-csi-rbd-provisioner -n ceph-csi-
rbd
helm status ceph-csi-rbd -n ceph-csi-rbd
```

in case you'd like to modify the configuration directly via Helm, you may adapt the `ceph-csi-rbd-values.yaml` file and call:

```
helm upgrade \
  --namespace ceph-csi-rbd ceph-csi-rbd ceph-csi/ceph-csi-rbd --values
ceph-csi-rbd-values.yaml
```

#### Creating RBD Ceph Resources

```
# Create a ceph pool:
ceph osd pool create myPool 64 64

# Create a block device pool:
rbd pool init myPool

# Create a block device image:
rbd create -s 2G myPool/image

# Create a block device user and record the key:
ceph auth get-or-create-key client.myPoolUser mon "allow r" osd "allow
class-read object_prefix rbd_children, allow rwx pool=myPool" | tr -d
'\n' | base64
QVFDZ0R5VmdyRk9KREJBQTJ5b2s5RlE2NUdSWExRQndhVVBwWXc9PQ==

# Encode the ceph user myPoolUser into a bash64 hash:
echo "myPoolUser" | tr -d '\n' | base64
bXlQb29sVXNlcg==

# Create a block device admin user and record the key:
ceph auth get-or-create-key client.myPoolAdmin mds 'allow *' mgr 'allow
*' mon 'allow *' osd 'allow * pool=myPool' | tr -d '\n' | base64
QVFCK0hDVmdXSjQ1T0JBQXBrc0VtcVhlZFpjc0JwaStIcmU5M3c9PQ==
```

```
# Encode the ceph user myPoolAdmin into a bash64 hash:
echo "myPoolAdmin" | tr -d '\n' | base64
bXlQb29sQWRtaW4=
```

#### Configure RBD Ceph Access Secrets

##### User Account

For static RBD provisioning (the image within the ceph pool must exist), run these commands:

```
cat > ceph-user-secret.yaml << EOF
apiVersion: v1
kind: Secret
metadata:
  name: ceph-user
  namespace: default
type: kubernetes.io/rbd
data:
  userID: bXlQb29sVXNlcg==
  userKey: QVFDZ0R5VmdyRk9KREJBQTJ5b2s5RlE2NUdSWExRQndhVVBwWXc9PQ==
EOF

kubectl apply -f ceph-user-secret.yaml
```

##### Admin Account

For dynamic RBD provisioning (used for automatic image creation within a given ceph pool), run these commands:

```
cat > ceph-admin-secret.yaml << EOF
apiVersion: v1
kind: Secret
metadata:
  name: ceph-admin
  namespace: default
type: kubernetes.io/rbd
data:
  userID: bXlQb29sQWRtaW4=
  userKey: QVFCK0hDVmdXSjQ1T0JBQXBrc0VtcVhlZFpjc0JwaStIcmU5M3c9PQ==
EOF

kubectl apply -f ceph-admin-secret.yaml
```

#### Create RBD Testing Resources

##### Using RBD in Pods

```
# pod
cat > ceph-rbd-pod-inline.yaml << EOF
apiVersion: v1
kind: Pod
metadata:
  name: ceph-rbd-pod-inline
spec:
  containers:
  - name: ceph-rbd-pod-inline
    image: busybox
    command: ["sleep", "infinity"]
    volumeMounts:
    - mountPath: /mnt/ceph_rbd
      name: volume
  volumes:
  - name: volume
    rbd:
      monitors:
      - 10.85.8.118:6789
      - 10.85.8.123:6789
      - 10.85.8.124:6789
      pool: myPool
      image: image
      user: myPoolUser
      secretRef:
        name: ceph-user
      fsType: ext4
      readOnly: false
EOF

kubectl apply -f ceph-rbd-pod-inline.yaml
kubectl get pod
kubectl exec pod/ceph-rbd-pod-inline -- df -k | grep rbd
```

##### Using RBD in Persistent Volumes

```
# pod-pvc-pv
cat > ceph-rbd-pod-pvc-pv-allinone.yaml << EOF
apiVersion: v1
kind: PersistentVolume
metadata:
```

```
  name: ceph-rbd-pv
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  rbd:
    monitors:
    - 10.85.8.118:6789
    - 10.85.8.123:6789
    - 10.85.8.124:6789
    pool: myPool
    image: image
    user: myPoolUser
    secretRef:
      name: ceph-user
    fsType: ext4
    readOnly: false
---
kind: PersistentVolumeClaim
apiVersion: vl
metadata:
  name: ceph-rbd-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
---
apiVersion: vl
kind: Pod
metadata:
  name: ceph-rbd-pod-pvc-pv
spec:
  containers:
  - name: ceph-rbd-pod-pvc-pv
    image: busybox
    command: ["sleep", "infinity"]
    volumeMounts:
    - mountPath: /mnt/ceph_rbd
      name: volume
  volumes:
  - name: volume
    persistentVolumeClaim:
```

```
      claimName: ceph-rbd-pvc
EOF

kubectl apply -f ceph-rbd-pod-pvc-pv-allinone.yaml
kubectl get pv,pvc,pod
kubectl exec pod/ceph-rbd-pod-pvc-pv -- df -k | grep rbd
```

#### Using RBD in Storage Classes

This example is for dynamic provisioning. The ceph-csi driver is needed.

```
# pod-pvc-sc
cat > ceph-rbd-pod-pvc-sc-allinone.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ceph-rbd-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: rbd.csi.ceph.com
parameters:
   clusterID: 79179d9d-98d8-4976-ab2e-58635caa7235
   pool: myPool
   imageFeatures: layering
   csi.storage.k8s.io/provisioner-secret-name: ceph-admin
   csi.storage.k8s.io/provisioner-secret-namespace: default
   csi.storage.k8s.io/controller-expand-secret-name: ceph-admin
   csi.storage.k8s.io/controller-expand-secret-namespace: default
   csi.storage.k8s.io/node-stage-secret-name: ceph-admin
   csi.storage.k8s.io/node-stage-secret-namespace: default
reclaimPolicy: Delete
allowVolumeExpansion: true
mountOptions:
   - discard
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ceph-rbd-sc-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

```
  storageClassName: ceph-rbd-sc
---
apiVersion: v1
kind: Pod
metadata:
  name: ceph-rbd-pod-pvc-sc
spec:
  containers:
  - name:  ceph-rbd-pod-pvc-sc
    image: busybox
    command: ["sleep", "infinity"]
    volumeMounts:
    - mountPath: /mnt/ceph_rbd
      name: volume
  volumes:
  - name: volume
    persistentVolumeClaim:
      claimName: ceph-rbd-sc-pvc
EOF

kubectl apply -f ceph-rbd-pod-pvc-sc-allinone.yaml
kubectl get pv,pvc,sc,pod
kubectl exec pod/ceph-rbd-pod-pvc-sc -- df -k | grep rbd
```

**SUSE® Rancher Prime: RKE2 Server/Master Provisioning**

```
sudo su
curl -sfL https://get.rke2.io | sh -
systemctl enable --now rke2-server

cat > /root/.bashrc << EOF
export PATH=$PATH:/var/lib/rancher/rke2/bin/
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
EOF

cat /var/lib/rancher/rke2/server/node-token
token:
K10ca0c38d4ff90d8b80319ab34092e315a8b732622e6adf97bc9eb0536REDACTED::ser
ver:ec0308000b8a6b595da000efREDACTED
```

**SUSE® Rancher Prime: RKE2 Agent/Worker provisioning**

```
mkdir -p /etc/rancher/rke2/
```

```
cat > /etc/rancher/rke2/config.yaml << EOF
server: https://10.100.103.23:9345
token:
K10ca0c38d4ff90d8b80319ab34092e315a8b732622e6adf97bc9eb0536REDACTED::ser
ver:ec0308000b8a6b595da000efREDACTED
EOF

curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
systemctl enable --now rke2-agent.service
```

To import the cluster into Rancher, click ☰ > **Cluster Management**. Then on the **Clusters** page, click **Import Existing**. Then run the provided kubectl command on the server/master node.

#### Tested Versions

OS for running RKE2 nodes: JeOS SLE15-SP2 with installed kernel-default-5.3.18-24.49

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:
"v1.18.4", GitCommit:"c96aede7b5205121079932896c4ad89bb93260af",
GitTreeState:"clean", BuildDate:"2020-06-22T12:00:00Z",
GoVersion:"go1.13.11", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.7+rke2r1",
GitCommit:"1dd5338295409edcfff11505e7bb246f0d325d15",
GitTreeState:"clean", BuildDate:"2021-01-20T01:50:52Z",
GoVersion:"go1.15.5b5", Compiler:"gc", Platform:"linux/amd64"}

helm version
version.BuildInfo{Version:"3.4.1",
GitCommit:"c4e74854886b2efe3321e185578e6db9be0a6e29",
GitTreeState:"clean", GoVersion:"go1.14.12"}
```

Kubernetes version on RKE2 cluster: v1.19.7+rke2r1

#### Troubleshooting

In case you are using SUSE's ceph-rook based on SES7, it might be useful to expose the monitors on hostNetwork by editing `rook-1.4.5/ceph/cluster.yaml` and setting `spec.network.hostNetwork=true`.

Also for operating the ceph-rook cluster, it is useful to deploy a toolbox on the Kubernetes cluster where ceph-rook is provisioned by `kubectl apply -f rook-1.4.5/ceph/toolbox.yaml` Then all the ceph related commands can be executed in the toolbox pod, for example, by running `kubectl exec -it -n rook-ceph rook-ceph-tools-686d8b8bfb-2nvqp -- bash`

Operating with the ceph - basic commands:

```
ceph osd pool stats
ceph osd pool delete myPool myPool --yes-i-really-really-mean-it
rbd list -p myPool
> csi-vol-f5d3766c-7296-lleb-b32a-c2b045952d38
> image
```

Delete the image: `rbd rm csi-vol-f5d3766c-7296-lleb-b32a-c2b045952d38 -p myPool`

CephFS commands in rook toolbox:

```
ceph -s
ceph fs ls
ceph fs fail cephfs
ceph fs rm cephfs --yes-i-really-mean-it
ceph osd pool delete cephfs_data cephfs_data --yes-i-really-really-mean
-it
ceph osd pool delete cephfs_metadata cephfs_metadata --yes-i-really
-really-mean-it
```

To prepare a cephfs filesystem, you can run this command on a rook cluster:

```
kubectl apply -f rook-1.4.5/ceph/filesystem.yaml
```

#### NFS Storage

Before you can use the NFS storage volume plug-in with Rancher deployments, you need to provision an NFS server.



- If you already have an NFS share, you don't need to provision a new NFS server to use the NFS volume plugin within Rancher. Instead, skip the rest of this procedure and complete adding storage.
- This procedure demonstrates how to set up an NFS server using Ubuntu, although you should be able to use these instructions for other Linux distros (e.g. Debian, RHEL, Arch Linux, etc.). For official instruction on how to create an NFS server using another Linux distro, consult the distro's documentation.



*Recommended:*

To simplify the process of managing firewall rules, use NFSv4.

1. Using a remote Terminal connection, log into the Ubuntu server that you intend to use for NFS storage.
2. Enter the following command:

```
sudo apt-get install nfs-kernel-server
```

3. Enter the command below, which sets the directory used for storage, along with user access rights. Modify the command if you'd like to keep storage at a different directory.

```
mkdir -p /nfs && chown nobody:nogroup /nfs
```

   - The `-p /nfs` parameter creates a directory named `nfs` at root.
   - The `chown nobody:nogroup /nfs` parameter allows all access to the storage directory.

4. Create an NFS exports table. This table sets the directory paths on your NFS server that are exposed to the nodes that will use the server for storage.

   a. Open `/etc/exports` using your text editor of choice.

   b. Add the path of the `/nfs` folder that you created in step 3, along with the IP addresses of your cluster nodes. Add an entry for each IP address in your cluster. Follow each address and its accompanying parameters with a single space that is a delimiter.

```
 /nfs <IP_ADDRESS1>(rw,sync,no_subtree_check) <IP_ADDRESS2>
(rw,sync,no_subtree_check) <IP_ADDRESS3>(rw,sync,no_subtree_check)
```



You can replace the IP addresses with a subnet. For example: `10.212.50.12/24`

   c. Update the NFS table by entering the following command:

```
exportfs -ra
```

5. Open the ports used by NFS.

   a. To find out what ports NFS is using, enter the following command:

```
rpcinfo -p | grep nfs
```

   b. Open the ports [https://help.ubuntu.com/lts/serverguide/firewall.html.en] that the previous command outputs. For example, the following command opens port 2049:

```
sudo ufw allow 2049
```

**Result:** Your NFS server is configured to be used for storage with your Rancher nodes.

#### What's Next?

Within Rancher, add the NFS server as a storage volume and/or storage class. After adding the server, you can use it for storage for your deployments.

#### Creating Persistent Storage in Amazon's EBS

This section describes how to set up Amazon's Elastic Block Store in EC2.

1. From the EC2 console, go to the **ELASTIC BLOCK STORE** section in the left panel and click **Volumes**.
2. Click **Create Volume**.
3. Optional: Configure the size of the volume or other options. The volume should be created in the same availability zone as the instance it will be attached to.
4. Click **Create Volume**.
5. Click **Close**.

**Result:** Persistent storage has been created.

For details on how to set up the newly created storage in Rancher, refer to the section on setting up existing storage.

### Provisioning Storage Examples

Rancher supports persistent storage with a variety of volume plugins. However, before you use any of these plugins to bind persistent storage to your workloads, you have to configure the storage itself, whether its a cloud-based solution from a service-provider or an on-prem solution that you manage yourself.

For your convenience, Rancher offers documentation on how to configure some of the popular storage methods:

- NFS
- vSphere
- EBS

#### VMware vSphere Storage

To provide stateful workloads with VMware vSphere storage, we recommend creating a vSphereVolume StorageClass. This practice dynamically provisions vSphere storage when workloads request volumes through a PersistentVolumeClaim.

In order to dynamically provision storage in vSphere, the vSphere provider must be enabled. See the following pages for more: Out-of-tree vSphere and in-tree vSphere.

#### Prerequisites

In order to provision vSphere volumes in a cluster created with the RKE2, the vSphere cloud provider [https://rancher.com/docs/rke/latest/en/config-options/cloud-providers/vsphere] must be explicitly enabled in the cluster options.

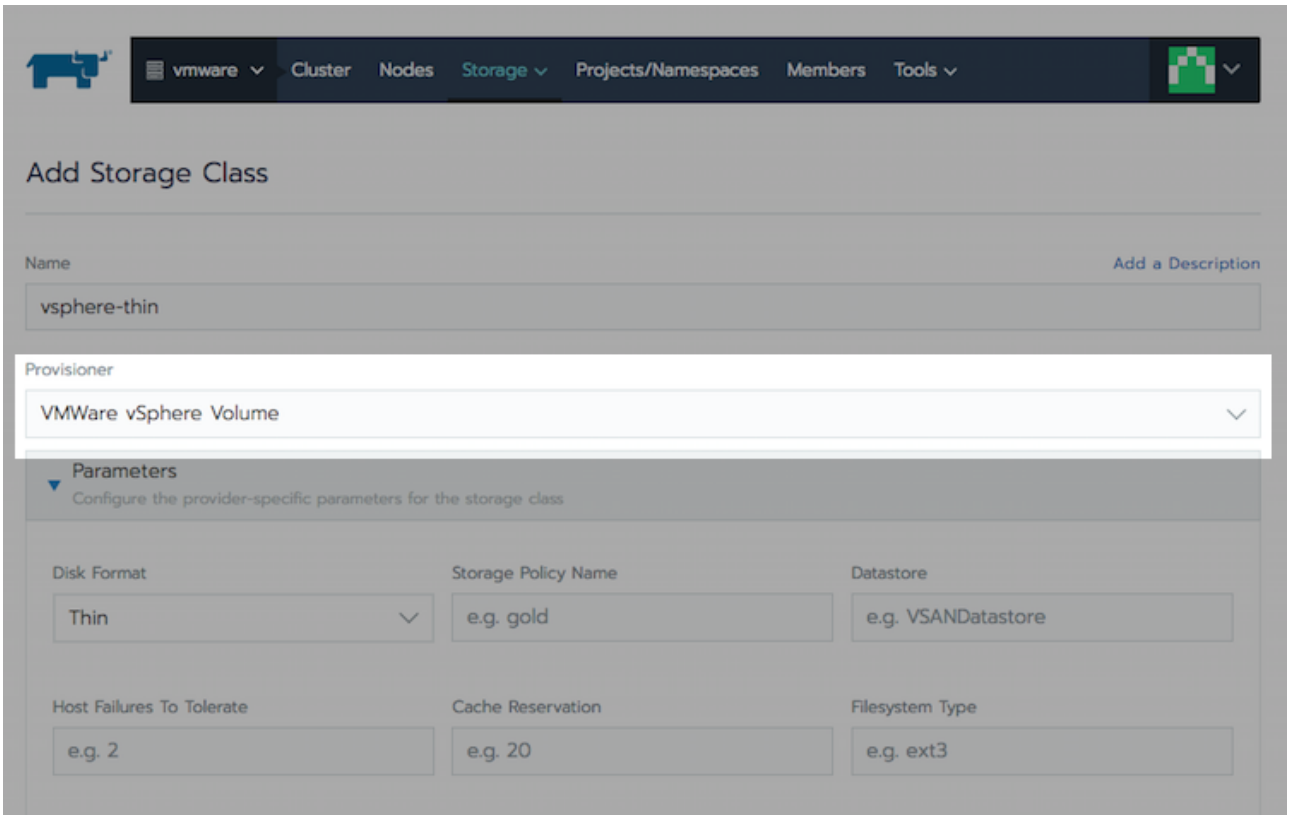
#### Creating a StorageClass



The following steps can also be performed using the `kubectl` command line tool. See Kubernetes documentation on persistent volumes [https://kubernetes.io/docs/concepts/storage/persistent-volumes/] for details.

1. Click ☰ **> Cluster Management**.

2. Choose the cluster you want to provide vSphere storage to and click **Explore**.
3. In the left navigation bar, select **Storage › StorageClasses**.
4. Click **Create**.
5. Enter a **Name** for the StorageClass.
6. Under **Provisioner**, select **VMWare vSphere Volume**.



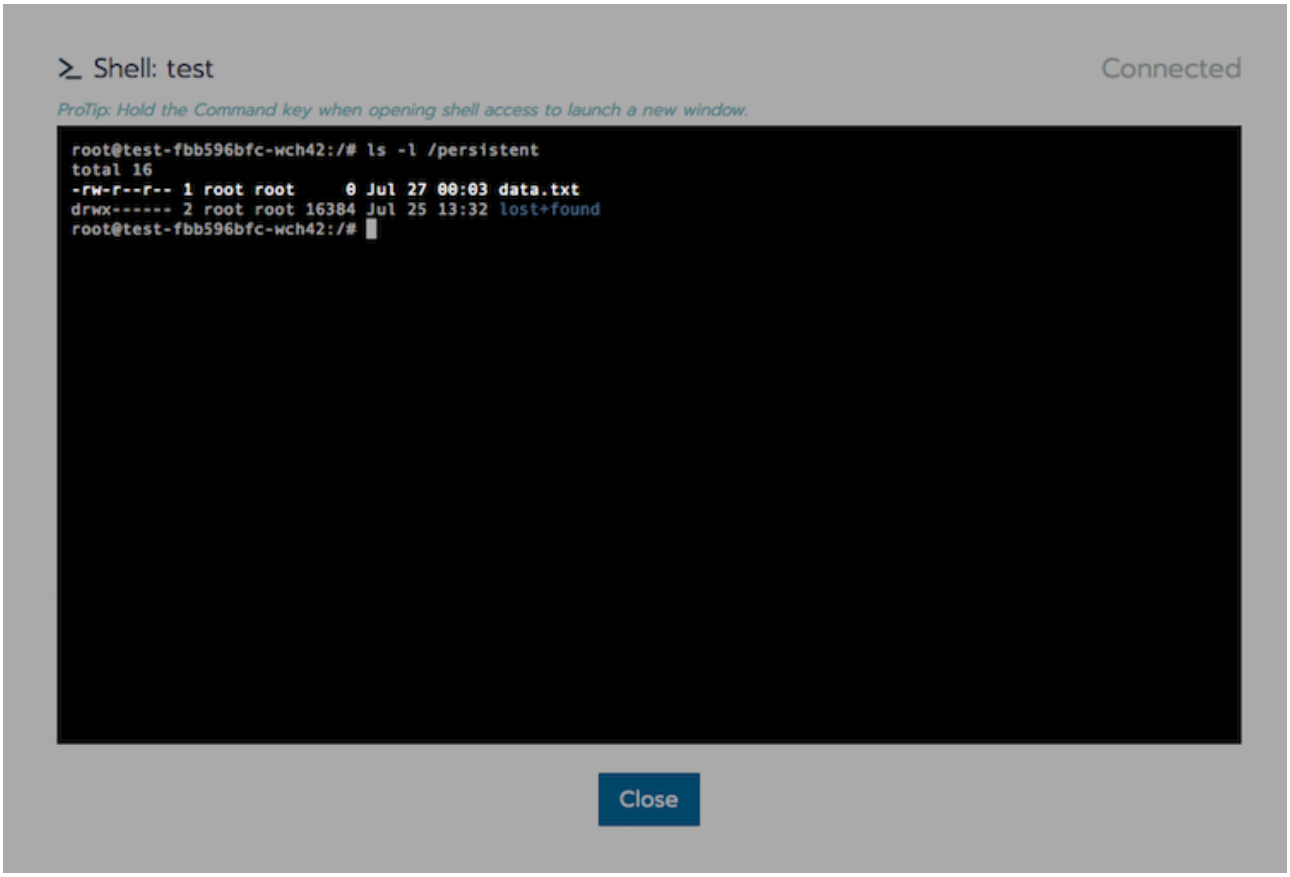

7. Optionally, specify additional properties for this storage class under **Parameters**. Refer to the vSphere storage documentation [https://github.com/vmware-archive/vsphere-storage-for-kubernetes/blob/master/documentation/storageclass.md] for details.
8. Click **Create**.

#### Creating a Workload with a VMware vSphere Volume

1. In the left navigation bar, click **Workload**.
2. Click **Create**.
3. Click **StatefulSet**.
4. In the **Volume Claim Templates** tab, click **Add Claim Template**.
5. Enter a persistent volume name.
6. In the Storage Class field, select the vSphere StorageClass that you created.
7. Enter the required **Capacity** for the volume. Then click **Define**.
8. Assign a path in the **Mount Point** field. This is the full path where the volume will be mounted in the container file system, e.g. `/persistent`.
9. Click **Create**.

#### Verifying Persistence of the Volume

1. In the left navigation bar, click **Workload › Pods**.
2. Go to the workload you just created and click ⋮ **> Execute Shell**.
3. Note the directory at root where the volume has been mounted to (in this case `/persistent`).
4. Create a file in the volume by executing the command `touch /<volumeMountPoint>/data.txt`.
5. Close the shell window.
6. Click on the name of the workload to reveal detail information.
7. Click ⋮ **> Delete**.
8. Observe that the pod is deleted. Then a new pod is scheduled to replace it so that the workload maintains its configured scale of a single stateful pod.
9. Once the replacement pod is running, click **Execute Shell**.
10. Inspect the contents of the directory where the volume is mounted by entering `ls -l /<volumeMountPoint>`. Note that the file you created earlier is still present.




#### Why to Use StatefulSets Instead of Deployments

You should always use StatefulSets [https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/] for workloads consuming vSphere storage, as this resource type is designed to address a VMDK block storage caveat.

Since vSphere volumes are backed by VMDK block storage, they only support an access mode [https://kubernetes.io/docs/concepts/storage/persistent-volumes/#persistentvolumeclaims] of `ReadWriteOnce`. This setting restricts the volume so that it can only be mounted to a single pod at a time, unless all pods consuming that volume are co-located on the same node. This behavior makes a deployment resource

unusable for scaling beyond a single replica if it consumes vSphere volumes.

Even using a deployment resource with just a single replica may result in a deadlock situation while updating the deployment. If the updated pod is scheduled to a node different from where the existing pod lives, it will fail to start because the VMDK is still attached to the other node.

#### Related Links

- vSphere Storage for Kubernetes [https://github.com/vmware-archive/vsphere-storage-for-kubernetes/tree/master/documentation]
- Kubernetes Persistent Volumes [https://kubernetes.io/docs/concepts/storage/persistent-volumes/]

## Cluster Templates

Cluster templates encompass both Kubernetes configuration and node pool configuration, allowing a single template to contain all the information Rancher needs to provision new nodes in a cloud provider and install Kubernetes on those nodes.

### Overview

Cluster templates are provided as Helm charts. To use them, you will need to clone and fork the templates, change them according to your use case, and then install the Helm charts on the Rancher management cluster. When the Helm chart is installed on the Rancher management cluster, a new cluster resource is created, which Rancher uses to provision the new cluster.

After the cluster is provisioned using the template, no changes to the template will affect the cluster. After the cluster is created from the cluster template, its configuration and infrastructure can change, because no restrictions are enforced by cluster templates.

#### Kubernetes Distribution

Cluster templates can use any Kubernetes distribution. For now, we provide an example with an RKE2 Kubernetes cluster. We may provide more examples in the future using other Kubernetes distributions.

#### Versioning

Rancher doesn't manage version control for cluster templates. Version control is handled in the repository containing the template's Helm chart.

### SUSE® Rancher Prime: RKE2 Cluster Template

See the example repository for an RKE2 cluster template [https://github.com/rancher/cluster-template-examples].

### Adding a Cluster Template to Rancher

In this section, you'll learn how to add the cluster template to the `local` cluster's chart repo list. The result is that Rancher will include the cluster template as an option when users install new Kubernetes clusters.



*Prerequisites:*

- You will need permission to install Helm charts on the `local` Rancher cluster.
- To make the chart viewable in the cluster creation form, both the chart and the index.yaml file must have the annotation, `catalog.cattle.io/type: cluster-`

`template`.

1. Go to a cluster template example repository. Rancher's examples are in this GitHub repository. [https://github.com/rancher/cluster-template-examples]
2. Fork the repository.
3. Optional: Edit the cluster options by editing the `values.yaml` file. For help editing the file, see the cluster template's Helm chart README.
4. Add the chart repository to Rancher. Click ☰ > **Cluster Management**.
5. Go to the `local` cluster and click **Explore.**
6. In the left navigation bar, click **Apps › Chart Repositories.**
7. Click **Create.**
8. Enter a name for the cluster template repository.
9. Click **Git Repository containing Helm chart definitions.**
10. In the **Git Repo URL** field, enter the URL for the repository. For example, `https://github.com/rancher/cluster-template-examples.git`.
11. In the **Git Branch** field, enter the branch to use as the source for the template. Rancher's example repository uses `main`.
12. Click **Create.**

**Result:** The cluster template available from the **Apps** in Rancher's `local` cluster. It can now be used to deploy clusters.

#### Creating a Cluster from a Cluster Template



*Prerequisites:*

- You will need permission to provision new Kubernetes clusters.
- You will need cloud credentials for provisioning infrastructure using the template.
- In order to show in the form for creating new clusters, the cluster template's Helm chart must have the `catalog.cattle.io/type: cluster-template` annotation.

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, click **Create.**
3. Click the name of your cluster template.
4. Finish installing the Helm chart.

**Result:** After Rancher provisions the new cluster, it is managed in the same way as any other Rancher-launched Kubernetes cluster. You can configure any options through the UI if the cluster template has options for the user to choose from.

#### Updating a Cluster Created from a Cluster Template

You can update any clusters using a template from the **Apps › Installed Apps** page, given there is a new version of a template being used by those clusters.

### Deploying Clusters from a Template with SUSE® Rancher Prime: Continuous Delivery



*Prerequisites:*

- You will need permission to provision new Kubernetes clusters.
- You will need cloud credentials for provisioning infrastructure using the template.
- In order to show in the form for creating new clusters, the cluster template's Helm chart must have the `catalog.cattle.io/type:cluster-template` annotation.
- In order to use a template as part of continuous delivery/GitOps, the cluster template needs to be deployed in the `fleet-local` namespace of the `local` cluster.
- All values must be set in the `values.yaml` of the template.
- Fleet repositories must follow these guidelines [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/explanations/gitrepo-content.html]. For RKE2 cluster templates, that means a `fleet.yaml` file must be added to the repository.

1. Click ☰ **> Cluster Management**.
2. On the **Clusters** page, click **Create.**
3. Click **Create Cluster from Template.**

**Result:** After Rancher provisions the new cluster, it is managed by Fleet.

### Uninstalling Cluster Templates

1. Click ☰ **> Cluster Management**.
2. Go to the `local` cluster and click **Apps › Chart Repositories.**
3. Go to the chart repository for your cluster template and click ☰ **> Delete.**
4. Confirm the deletion.

**Result:** The cluster template is uninstalled. This action does not affect clusters created with the cluster template.

An admin with access to the `local` cluster can also remove a cluster deployed via cluster templates through the **Apps › Installed Apps** page.

### Configuration Options

Cluster templates are flexible enough that they can be used to configure all of the following options:

- Node configuration
- Node pools
- Pre-specified cloud credentials
- Enable/configure an authorized cluster endpoint to get kubectl access to the cluster without using Rancher as a proxy
- Install Rancher V2 monitoring
- Kubernetes version
- Assign cluster members

- Infrastructure configuration such as AWS VPC/subnets or vSphere data center
- Cloud provider options
- Pod security options
- Network providers
- Ingress controllers
- Network security configuration
- Network plugins
- Private registry URL and credentials
- Add-ons
- Kubernetes options, including configurations for Kubernetes components such as kube-api, kube-controller, kubelet, and services

For details on how to configure the template, refer to the cluster template's Helm chart README.

## Nodes and Machine Pools

After you launch a Kubernetes cluster in Rancher, you can manage individual nodes from the cluster's **Node** tab.

1. Click ☰ in the top left corner.
2. Select **Cluster Management**.
3. Find the cluster whose nodes you want to manage, and click the **Explore** button at the end of the row.
4. Select **Nodes** from the left navigation.

Depending on the option used to provision the cluster, there are different node options available.



If you want to manage the *cluster* and not individual nodes, see xref:[Editing Clusters].

### Node Options Available for Each Cluster Creation Option

The following table lists which node options are available for each type of cluster in Rancher. Click the links in the **Option** column for more detailed information about each feature.

| Option | Nodes Hosted by an Infrastructure Provider | Custom Node | Hosted Cluster | Registered EKS Nodes | All Other Registered Nodes | Description |
|---|---|---|---|---|---|---|
| Cordon | ☐ | ☐ | ☐ | ☐ | ☐ | Marks the node as unschedulable. |
| Drain | ☐ | ☐ | ☐ | ☐ | ☐ | Marks the node as unschedulable *and* evicts all pods. |

| Option | Nodes Hosted by an Infrastructure Provider | Custom Node | Hosted Cluster | Registered EKS Nodes | All Other Registered Nodes | Description |
|---|---|---|---|---|---|---|
| Edit | ◪ | ◪ | ◪ | ◪ | ◪ | Enter a custom name, description, label, or taints for a node. |
| View API | ◪ | ◪ | ◪ | ◪ | ◪ | View API data. |
| Delete | ◪ | ◪ | | * | * | Deletes defective nodes from the cluster. |
| Download Keys | ◪ | | | | | Download SSH key in order to SSH into the node. |
| Node Scaling | ◪ | | | ◪ | | Scale the number of nodes in the node pool up or down. |

- Delete option accessible via View API

#### Nodes Hosted by an Infrastructure Provider

Machine pools are available when you provision Rancher-launched Kubernetes clusters on nodes that are hosted in an infrastructure provider.

#### Nodes Provisioned by Hosted Kubernetes Providers

Options for managing nodes hosted by a Kubernetes provider are somewhat limited in Rancher. Rather than using the Rancher UI to make edits such as scaling the number of nodes up or down, edit the cluster directly.

#### Registered Nodes

Although you can deploy workloads to a registered cluster using Rancher, you cannot manage individual cluster nodes. All management of imported cluster nodes must take place outside of Rancher.

### Managing and Editing Individual Nodes

Editing a node lets you:

- Change its name
- Change its description
- Add labels [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/]
- Add/Remove taints [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/]

To manage individual nodes, browse to the cluster that you want to manage and then select **Nodes** from the main menu. You can open the options menu for a node by clicking its ◪ icon (**...**).

#### Viewing a Node in the Rancher API

Select this option to view the node's API endpoints.

#### Deleting a Node

Use **Delete** to remove defective nodes from the cloud provider.

When you delete a defective node, Rancher can automatically replace it with an identically provisioned node if the node is in a machine pool and auto-replace is enabled.



If your cluster is hosted by an infrastructure provider, and you want to scale your cluster down instead of deleting a defective node, scale down rather than delete.

#### Scaling Nodes

For nodes hosted by an infrastructure provider, you can scale the number of nodes in each machine pool by using the scale controls. This option isn't available for other cluster types.

#### SSH into a Node Hosted by an Infrastructure Provider

For nodes hosted by an infrastructure provider, you have the option of downloading its SSH key so that you can connect to it remotely from your desktop.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to SSH into a node and click the name of the cluster.
3. On the **Machine Pools** tab, find the node that you want to remote into and click ☰ **> Download SSH Key**. A ZIP file containing files used for SSH is then downloaded.
4. Extract the ZIP file to any location.
5. Open Terminal. Change your location to the extracted ZIP file.
6. Enter the following command:

```
ssh -i id_rsa root@<IP_OF_HOST>
```

#### Cordoning a Node

*Cordoning* a node marks it as unschedulable. This feature is useful for performing short tasks on the node during small maintenance windows, like reboots, upgrades or decommissions. When you're done, power back on and make the node schedulable again by uncordoning it.

#### Draining a Node

*Draining* is the process of first cordoning the node, and then evicting all its pods. This feature is useful for performing node maintenance (like kernel upgrades or hardware maintenance). It prevents new pods from deploying to the node while redistributing existing pods so that users don't experience service interruption.

- For pods with a replica set, the pod is replaced by a new pod that is scheduled to a new node. Additionally, if the pod is part of a service, then clients are automatically redirected to the new pod.

- For pods with no replica set, you need to bring up a new copy of the pod, and assuming it is not part of a service, redirect clients to it.

You can drain nodes that are in either a `cordoned` or `active` state. When you drain a node, the node is cordoned, the nodes are evaluated for conditions they must meet to be drained, and then (if it meets the conditions) the node evicts its pods.

However, you can override the conditions draining when you initiate the drain. You're also given an opportunity to set a grace period and timeout value.

#### Aggressive and Safe Draining Options

When you configure the upgrade strategy for the cluster, you can enable node draining. If node draining is enabled, you are able to configure how pods are deleted and rescheduled.

- **Aggressive Mode**

  In this mode, pods won't get rescheduled to a new node, even if they do not have a controller. Kubernetes expects you to have your own logic that handles the deletion of these pods.

  Kubernetes also expects the implementation to decide what to do with pods using emptyDir. If a pod uses emptyDir to store local data, you might not be able to safely delete it, since the data in the emptyDir is deleted once the pod is removed from the node. Choosing aggressive mode deletes these pods.

- **Safe Mode**

  If a node has stand-alone pods or ephemeral data it is cordoned but not drained.

#### Grace Period

The timeout given to each pod for cleaning things up so they have a chance to exit gracefully. For example, when pods might need to finish any outstanding requests, roll back transactions or save state to an external storage. If negative, the default value specified in the pod is used.

#### Timeout

The amount of time drain should continue to wait before giving up.



*Kubernetes Known Issue:*

The timeout setting [https://github.com/kubernetes/kubernetes/pull/64378] was not enforced while draining a node before Kubernetes 1.12.

#### Drained and Cordoned State

If there's any error related to user input, the node enters a `cordoned` state because the drain failed. You can either correct the input and attempt to drain the node again, or you can abort by uncordoning the node.

If the drain continues without error, the node enters a `draining` state. You'll have the option to stop the drain when the node is in this state, which then stops the drain process and changes the node's state to `cordoned`.

Once drain successfully completes, the node is in a state of `drained`. You can then power off or delete the node.

**Want to know more about cordon and drain?** See the Kubernetes documentation [https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/].

#### Labeling a Node to be Ignored by Rancher

Certain solutions, such as F5's BIG-IP integration, may require creating a node that is never registered to a cluster.

Since the node never finishes registering, it is always shown as unhealthy in the Rancher UI.

In that case, you may want to label the node to be ignored by Rancher so that Rancher only shows nodes as unhealthy when they are actually failing.

You can label nodes to be ignored by using a setting in the Rancher UI, or by using `kubectl`.



There is an open issue [https://github.com/rancher/rancher/issues/24172] in which nodes labeled to be ignored can get stuck in an updating state.

##### Labeling Nodes to be Ignored with kubectl

To add a node that is ignored by Rancher, use `kubectl` to create a node that has the following label:

```
cattle.rancher.io/node-status: ignore
```

**Result**: If you add the node to a cluster, Rancher skips syncing with this node. The node can still be part of the cluster and can be listed with `kubectl`.

If the label is added before the node is added to the cluster, the node is not shown in the Rancher UI.

If the label is added after the node is added to a Rancher cluster, the node is not removed from the UI.

If you delete the node from the Rancher server using the Rancher UI or API, the node is not removed from the cluster if the `nodeName` is listed in the Rancher settings in the Rancher API under `v3/settings/ignore-node-name`.

### Projects and Kubernetes Namespaces with SUSE Rancher Prime

A namespace is a Kubernetes concept that allows a virtual cluster within a cluster, which is useful for dividing the cluster into separate "virtual clusters" that each have their own access control and resource quotas.

A project is a group of namespaces, and it is a concept introduced by Rancher. Projects allow you to manage multiple namespaces as a group and perform Kubernetes operations in them. You can use projects to support multi-tenancy, so that a team can access a project within a cluster without having access to other projects in the same cluster.



Projects are de-emphasized on the UI because it is not required to create Kubernetes resources within a project scope. However, resources such as Secrets can still be created in a project scope.

This section describes how projects and namespaces work with Rancher.

### About Namespaces

A namespace is a concept introduced by Kubernetes. According to the official Kubernetes documentation on namespaces, [https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/]



> Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces. [...] Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you should not need to create or think about namespaces at all.

Namespaces provide the following functionality:

- **Providing a scope for names:** Names of resources need to be unique within a namespace, but not across namespaces. Namespaces can not be nested inside one another and each Kubernetes resource can only be in one namespace.
- **Resource quotas:** Namespaces provide a way to divide cluster resources between multiple users.

You can assign resources at the project level so that each namespace in the project can use them. You can also bypass this inheritance by assigning resources explicitly to a namespace.

You can assign the following resources directly to namespaces:

- Workloads
- Load Balancers/Ingress
- Service Discovery Records
- Persistent Volume Claims
- Certificates
- ConfigMaps
- Registries
- Secrets

To manage permissions in a vanilla Kubernetes cluster, cluster admins configure role-based access policies for each namespace. With Rancher, user permissions are assigned on the project level instead, and permissions are automatically inherited by any namespace owned by the particular project.

For more information on creating and moving namespaces, see Namespaces.

#### Role-based access control issues with namespaces and kubectl

Because projects are a concept introduced by Rancher, kubectl does not have the capability to restrict the creation of namespaces to a project the creator has access to.

This means that when standard users with project-scoped permissions create a namespaces with `kubectl`, it may be unusable because `kubectl` doesn't require the new namespace to be scoped within a certain project.

If your permissions are restricted to the project level, it is better to create a namespace through Rancher to ensure that you will have permission to access the namespace.

If a standard user is a project owner, the user will be able to create namespaces within that project. The

Rancher UI will prevent that user from creating namespaces outside the scope of the projects they have access to.

### About Projects

In terms of hierarchy:

- Clusters contain projects
- Projects contain namespaces

You can use projects to support multi-tenancy, so that a team can access a project within a cluster without having access to other projects in the same cluster.

In the base version of Kubernetes, features like role-based access rights or cluster resources are assigned to individual namespaces. A project allows you to save time by giving an individual or a team access to multiple namespaces simultaneously.

You can use projects to perform actions such as:

- Assign users to a group of namespaces (i.e., project membership).
- Assign users specific roles in a project. A role can be owner, member, read-only, or custom.
- Assign resources to the project.

When you create a cluster, two projects are automatically created within it:

- Default Project
- System Project

#### The Cluster's Default Project

When you provision a cluster with Rancher, it automatically creates a `default` project for the cluster. This is a project you can use to get started with your cluster, but you can always delete it and replace it with projects that have more descriptive names.

If you don't have a need for more than the default namespace, you also do not need more than the **Default** project in Rancher.

If you require another level of organization beyond the **Default** project, you can create more projects in Rancher to isolate namespaces, applications and resources.

#### The System Project

When troubleshooting, you can view the `system` project to check if important namespaces in the Kubernetes system are working properly. This easily accessible project saves you from troubleshooting individual system namespace containers.

To open it, open the cluster view and click **Cluster › Projects/Namespaces**. This view shows all of the namespaces in the `system` project.

The `system` project:

- Is automatically created when you provision a cluster.

- Lists all namespaces that exist in `v3/settings/system-namespaces`, if they exist.
- Allows you to add more namespaces or move its namespaces to other projects.
- Cannot be deleted because it's required for cluster operations.



In RKE2/K3s clusters where the project network isolation option is enabled, the `system` project overrides the project network isolation option so that it can communicate with other projects, collect logs, and check health.

#### Project Authorization

Standard users are only authorized for project access in two situations:

- An administrator, cluster owner or cluster member explicitly adds the standard user to the project's **Members** tab.
- Standard users can access projects that they create themselves.

#### Creating Projects

This section describes how to create a new project with a name and with optional pod security policy, members, and resource quotas.

1. Name a new project.
2. Recommended: Add project members.
3. Optional: Add resource quotas.

##### Name a New Project

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to project in and click **Explore**.
3. Click **Cluster › Projects/Namespaces**.
4. Click **Create Project**.
5. Enter a **Project Name**.

##### Recommended: Add Project Members

Use the **Members** section to provide other users with project access and roles.

By default, your user is added as the project `Owner`.



*Notes on Permissions:*

- Users assigned the `Owner` or `Member` role for a project automatically inherit the `namespace creation` role. However, this role is a Kubernetes ClusterRole [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-and-clusterrole], meaning its scope extends to all projects in the cluster. Therefore, users explicitly assigned the `Owner` or `Member` role for a project can create namespaces in other projects they're assigned to, even with only the `Read Only` role assigned.
- By default, the Rancher role of `project-member` inherits from the `Kubernetes-edit` role, and the `project-owner` role inherits from the `Kubernetes-admin` role.

As such, both `project-member` and `project-owner` roles will allow for namespace management, including the ability to create and delete namespaces.

- Choose `Custom` to create a custom role on the fly: Custom Project Roles.

To add members:

1. In the **Members** tab, click **Add**.
2. From the **Select Member** field, search for a user or group that you want to assign project access. You can only search for groups if external authentication is enabled.



- At least 2 characters must be typed in the search box for results to appear.
- Users can be searched based on their username or display name.
- Search is prefix-based (e.g., a user named `Stan Dard` will appear when searching for `Sta`, but not when searching for `Dar`) and case-sensitive.

3. In the **Project Permissions** section, choose a role. For more information, refer to the documentation on project roles.

#### Optional: Add Resource Quotas

Resource quotas limit the resources that a project (and its namespaces) can consume. For more information, see Resource Quotas.

To add a resource quota,

1. In the **Resource Quotas** tab, click **Add Resource**.
2. Select a **Resource Type**. For more information, see Resource Quotas..
3. Enter values for the **Project Limit** and the **Namespace Default Limit**.
4. **Optional:** Specify **Container Default Resource Limit**, which will be applied to every container started in the project. The parameter is recommended if you have CPU or Memory limits set by the Resource Quota. It can be overridden on per an individual namespace or a container level. For more information, see Container Default Resource Limit
5. Click **Create**.

**Result:** Your project is created. You can view it from the cluster's **Projects/Namespaces** view.

| Field | Description |
|---|---|
| Project Limit | The overall resource limit for the project. |
| Namespace Default Limit | The default resource limit available for each namespace. This limit is propagated to each namespace in the project when created. The combined limit of all project namespaces shouldn't exceed the project limit. |

### Deleting Projects

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster the project is attached to, and click **Explore**.
3. Click **Cluster › Projects/Namespaces**.

4. Find the project you want to delete, and click ⋮.
5. Select **Delete**.

When you delete a project, any namespaces that were formerly associated with the project will remain on the cluster. You can find these namespaces in the Rancher UI, in the **Not in a Project** tab of the **Projects/Namespaces** page. You can reassign these namespaces to a project by moving them.

#### Further Reading

You can create projects and project members more programmatically via the Public API. See the project workflow doc for more information.

## Certificate Rotation



Rotating Kubernetes certificates may result in your cluster being temporarily unavailable as components are restarted. For production environments, it's recommended to perform this action during a maintenance window.

By default, Kubernetes clusters require certificates and Rancher launched Kubernetes clusters automatically generate certificates for the Kubernetes components. Rotating these certificates is important before the certificates expire as well as if a certificate is compromised. After the certificates are rotated, the Kubernetes components are automatically restarted.

Certificates can be rotated for the following services:

- admin
- api-server
- controller-manager
- scheduler
- rke2-controller
- rke2-server
- cloud-controller
- etcd
- auth-proxy
- kubelet
- kube-proxy



For users who didn't rotate their webhook certificates, and they have expired after one year, please see this page for help.

#### Certificate Rotation

Rancher launched Kubernetes clusters have the ability to rotate the auto-generated certificates through the UI.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to rotate certificates for and click ⋮ **> Rotate**

**Certificates**.

3. Select which certificates that you want to rotate.
   - Rotate all Service certificates (keep the same CA)
   - Rotate an individual service and choose one of the services from the drop-down menu
4. Click **Save**.

**Results:** The selected certificates will be rotated and the related services will be restarted to start using the new certificate.

#### Additional Notes

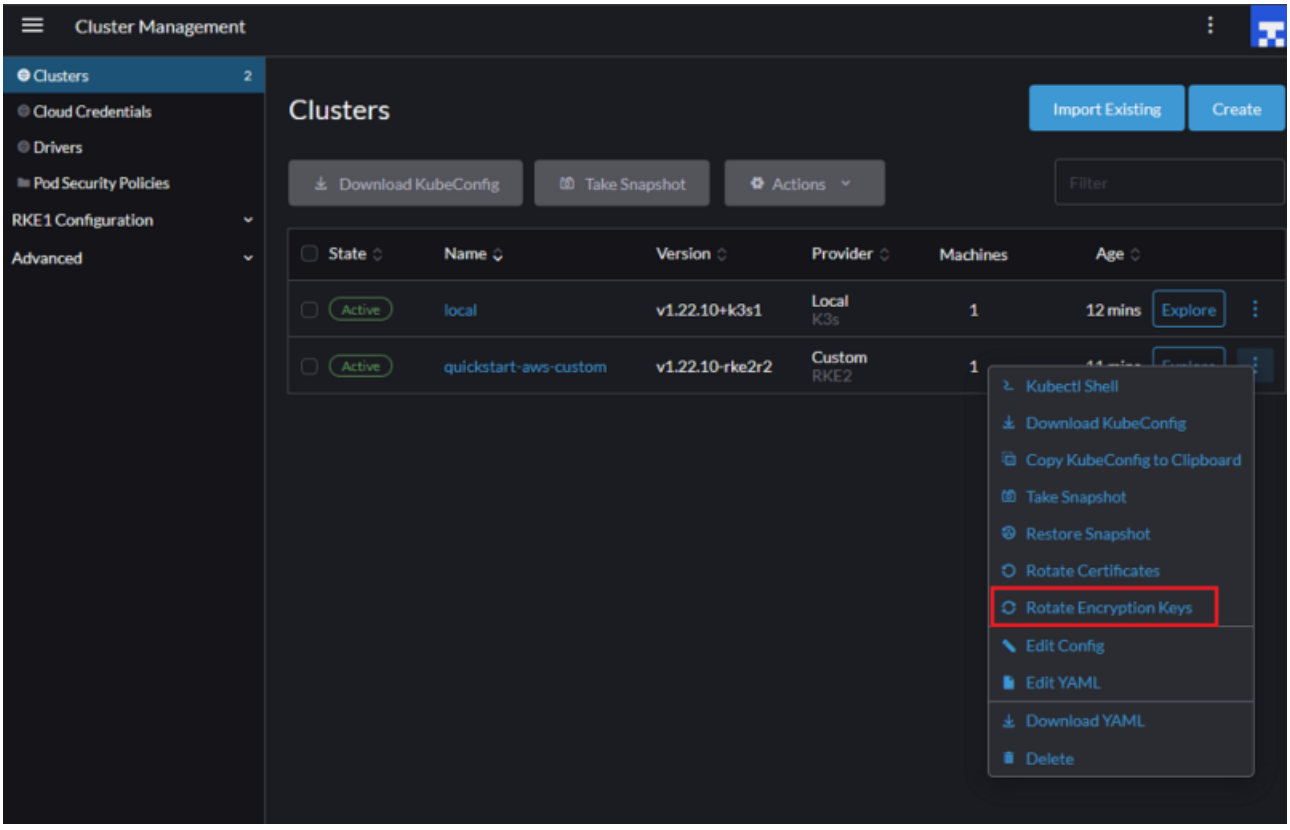
In RKE2, both etcd and control plane nodes are treated as the same `server` concept. As such, when rotating certificates of services specific to either of these components will result in certificates being rotated on both. The certificates will only change for the specified service, but you will see nodes for both components go into an updating state. You may also see worker only nodes go into an updating state. This is to restart the workers after a certificate change to ensure they get the latest client certs.

### Encryption Key Rotation

Encryption key rotation is enabled by default and cannot be disabled.

To rotate keys in the Rancher UI:

1. Click ☰ > **Cluster Management**.
2. Select ☰ > **Rotate Encryption Keys** on the far right of the screen next to your chosen cluster:




For more information on RKE2 secrets encryption config, please see the RKE2 docs

[https://documentation.suse.com/cloudnative/rke2/latest/en/security/secrets_encryption.html].

# Kubernetes Resources Setup

You can view and manipulate all of the custom resources and CRDs in a Kubernetes cluster from the Rancher UI.

## Workloads

Deploy applications to your cluster nodes using workloads, which are objects that contain pods that run your apps, along with metadata that set rules for the deployment's behavior. Workloads can be deployed within the scope of the entire clusters or within a namespace.

When deploying a workload, you can deploy from any image. There are a variety of _workload_types[workload types to choose from which determine how your application should run.

Following a workload deployment, you can continue working with it. You can:

- Upgrade the workload to a newer version of the application it's running.
- Roll back a workload to a previous version, if an issue occurs during upgrade.
- Add a sidecar, which is a workload that supports a primary workload.

## Load Balancing and Ingress

### Load Balancers

After you launch an application, it's only available within the cluster. It can't be reached externally.

If you want your applications to be externally accessible, you must add a load balancer to your cluster. Load balancers create a gateway for external connections to access your cluster, provided that the user knows the load balancer's IP address and the application's port number.

Rancher supports two types of load balancers:

- Layer-4 Load Balancers
- Layer-7 Load Balancers

For more information, see load balancers.

#### Ingress

Load Balancers can only handle one IP address per service, which means if you run multiple services in your cluster, you must have a load balancer for each service. Running multiples load balancers can be expensive. You can get around this issue by using an ingress.

Ingress is a set of rules that act as a load balancer. Ingress works in conjunction with one or more ingress controllers to dynamically route service requests. When the ingress receives a request, the ingress controller(s) in your cluster program the load balancer to direct the request to the correct service based on service subdomains or path rules that you've configured.

For more information, see Ingress.

## Service Discovery

After you expose your cluster to external requests using a load balancer and/or ingress, it's only available by IP address. To create a resolveable hostname, you must create a service record, which is a record that maps an IP address, external hostname, DNS record alias, workload(s), or labelled pods to a specific hostname.

For more information, see Service Discovery.

## Applications

Besides launching individual components of an application, you can use the Rancher catalog to start launching applications, which are Helm charts.

## Kubernetes Resources

Within the context of a Rancher project or namespace, *resources* are files and data that support operation of your pods. Within Rancher, certificates, registries, and secrets are all considered resources. However, Kubernetes classifies resources as different types of secrets [https://kubernetes.io/docs/concepts/configuration/secret/]. Therefore, within a single project or namespace, individual resources must have unique names to avoid conflicts. Although resources are primarily used to carry sensitive information, they have other uses as well.

Resources include:

- Certificates: Files used to encrypt/decrypt data entering or leaving the cluster.
- ConfigMaps: Files that store general configuration information, such as a group of config files.
- Secrets: Files that store sensitive data like passwords, tokens, or keys.
- Registries: Files that carry credentials used to authenticate with private registries.

## Kubernetes Workloads and Pods

You can build any complex containerized application in Kubernetes using two basic constructs: pods and workloads. Once you build an application, you can expose it for access either within the same cluster or on the Internet using a third construct: services.

### Pods

*Pods* [https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/] are one or more containers that share network namespaces and storage volumes. Most pods have only one container. Therefore when we discuss *pods*, the term is often synonymous with *containers*. You scale pods the same way you scale containers—by having multiple instances of the same pod that implement a service. Usually pods get scaled and managed by the workload.

### Workloads

*Workloads* are objects that set deployment rules for pods. Based on these rules, Kubernetes performs the deployment and updates the workload with the current state of the application. Workloads let you define the rules for application scheduling, scaling, and upgrade.

#### Workload Types

Kubernetes divides workloads into different types. The most popular types supported by Kubernetes are:

- Deployments [https://kubernetes.io/docs/concepts/workloads/controllers/deployment/]

  *Deployments* are best used for stateless applications (i.e., when you don't have to maintain the workload's state). Pods managed by deployment workloads are treated as independent and disposable. If a pod encounters disruption, Kubernetes removes it and then recreates it. An example application would be an Nginx web server.

- StatefulSets [https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/]

  *StatefulSets*, in contrast to deployments, are best used when your application needs to maintain its identity and store data. An application would be something like Zookeeper—an application that requires a database for storage.

- DaemonSets [https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/]

  *Daemonsets* ensures that every node in the cluster runs a copy of pod. For use cases where you're collecting logs or monitoring node performance, this daemon-like workload works best.

- Jobs [https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/]

  *Jobs* launch one or more pods and ensure that a specified number of them successfully terminate. Jobs are best used to run a finite task to completion as opposed to managing an ongoing desired application state.

- CronJobs [https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/]

  *CronJobs* are similar to jobs. CronJobs, however, runs to completion on a cron-based schedule.

### Services

In many use cases, a workload has to be either:

- Accessed by other workloads in the cluster.
- Exposed to the outside world.

You can achieve these goals by creating a *Service*. Services are mapped to the underlying workload's pods using a selector/label approach (view the code samples) [https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#service-and-replicationcontroller]. Rancher UI simplifies this mapping process by automatically creating a service along with the workload, using the service port and type that you select.

#### Service Types

There are several types of services available in Rancher. The descriptions below are sourced from the Kubernetes Documentation [https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types].

- **ClusterIP**

  > Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster. This is the default

> `ServiceType`.

- **NodePort**

  > Exposes the service on each Node's IP at a static port (the `NodePort`). A `ClusterIP` service, to which the `NodePort` service will route, is automatically created. You'll be able to contact the `NodePort` service, from outside the cluster, by requesting `<NodeIP>:<NodePort>`.

- **LoadBalancer**

  > Exposes the service externally using a cloud provider's load balancer. `NodePort` and `ClusterIP` services, to which the external load balancer will route, are automatically created.

#### Workload Options

This section of the documentation contains instructions for deploying workloads and using workload options.

- Deploy Workloads
- Upgrade Workloads
- Rollback Workloads

#### Related Links

- Services [https://kubernetes.io/docs/concepts/services-networking/service/]

#### Deploying Workloads

Deploy a workload to run an application in one or more containers.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to upgrade a workload and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Click **Create**.
5. Choose the type of workload.
6. Select the namespace where the workload will be deployed.
7. Enter a **Name** for the workload.
8. From the **Container Image** field, enter the name of the Docker image that you want to deploy to the project, optionally prefacing it with the registry host (e.g. `quay.io`, `registry.gitlab.com`, etc.). During deployment, Rancher pulls this image from the specified public or private registry. If no registry host is provided, Rancher will pull the image from Docker Hub [https://hub.docker.com/explore/]. Enter the name exactly as it appears in the registry server, including any required path, and optionally including the desired tag (e.g. `registry.gitlab.com/user/path/image:tag`). If no tag is provided, the `latest` tag will be automatically used.

9. Either select an existing namespace, or click **Add to a new namespace** and enter a new namespace.
10. Click **Add Port** to enter a port mapping, which enables access to the application inside and outside of the cluster . For more information, see Services.
11. Configure the remaining options:

    - **Environment Variables**

      Use this section to either specify environment variables for your workload to consume on the fly, or to pull them from another source, such as a secret or ConfigMap.

    - **Node Scheduling**
    - **Health Check**
    - **Volumes**

      Use this section to add storage for your workload. You can manually specify the volume that you want to add, use a persistent volume claim to dynamically create a volume for the workload, or read data for a volume to use from a file such as a ConfigMap.

      When you are deploying a Stateful Set, you should use a Volume Claim Template when using Persistent Volumes. This will ensure that Persistent Volumes are created dynamically when you scale your Stateful Set.

    - **Scaling/Upgrade Policy**

    *Amazon Note for Volumes:*

    To mount an Amazon EBS volume: **In Amazon AWS [https://aws.amazon.com/], the nodes must be in the same Availability Zone and possess IAM permissions to attach/unattach volumes.** The cluster must be using the AWS cloud provider [https://github.com/kubernetes/website/blob/release-1.18/content/en/docs/concepts/cluster-administration/cloud-providers.md#aws] option. For more information on enabling this option see Creating an Amazon EC2 Cluster or Creating a Custom Cluster.

12. Click **Show Advanced Options** and configure:

    - **Command**
    - **Networking**
    - **Labels & Annotations**
    - **Security and Host Config**

13. Click **Launch**.

**Result:** The workload is deployed to the chosen namespace. You can view the workload's status from the project's **Workloads** view.

#### Adding a Sidecar

A *sidecar* is a container that extends or enhances the main container in a pod. The main container and the sidecar share a pod, and therefore share the same network space and storage. You can add sidecars to existing workloads by using the **Add a Sidecar** option.

1. In the upper left corner, click ☰ **> Cluster Management**.

2. Go to the cluster where you want to add a sidecar and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Find the workload that you want to extend. Select **> + Add Sidecar**.
5. Enter a **Name** for the sidecar.
6. In the **General** section, select a sidecar type. This option determines if the sidecar container is deployed before or after the main container is deployed.

   - **Standard Container:**

     The sidecar container is deployed after the main container.

   - **Init Container:**

     The sidecar container is deployed before the main container.

7. From the **Container Image** field, enter the name of the container image that you want to deploy in support of the main container. During deployment, Rancher pulls this image from Docker Hub [https://hub.docker.com/explore/]. Enter the name exactly as it appears on Docker Hub.
8. Set the remaining options. You can read about them in Deploying Workloads.
9. Click **Launch**.

**Result:** The sidecar is deployed according to your parameters. Following its deployment, you can view the sidecar by selecting **icon (...) > Edit** for the main deployment.

#### Related Links

- The Distributed System ToolKit: Patterns for Composite Containers [https://kubernetes.io/blog/2015/06/the-distributed-system-toolkit-patterns/]

### Upgrading Workloads

When a new version of an application image is released on Docker Hub, you can upgrade any workloads running a previous version of the application to the new one.

1. In the upper left corner, click **> Cluster Management**.
2. Go to the cluster where you want to upgrade a workload and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Find the workload that you want to upgrade and select **> Edit Config**.
5. Update the **Container Image** and any options that you want to change.
6. Review and edit the workload's **Scaling and Upgrade Policy**.

   These options control how the upgrade rolls out to containers that are currently running. For example, for scalable deployments, you can choose whether you want to stop old pods before deploying new ones, or vice versa, as well as the upgrade batch size.

7. Click **Save**.

**Result:** The workload begins upgrading its containers, per your specifications. Note that scaling up the deployment or updating the upgrade/scaling policy won't result in the pods recreation.

#### Rolling Back Workloads

Sometimes there is a need to rollback to the previous version of the application, either for debugging purposes or because an upgrade did not go as planned.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to upgrade a workload and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Find the workload that you want to rollback and select ☰ > **Rollback**.
5. Choose the revision that you want to roll back to. Click **Rollback**.

**Result:** Your workload reverts to the previous version that you chose. Wait a few minutes for the action to complete.

### Horizontal Pod Autoscaler

The Horizontal Pod Autoscaler [https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/] (HPA) is a Kubernetes feature that allows you to configure your cluster to automatically scale the services it's running up or down.

Rancher provides some additional features to help manage HPAs, depending on the version of Rancher.

You can create, manage, and delete HPAs using the Rancher UI. It only supports HPA in the `autoscaling/v2beta2` API.

#### Managing HPAs

The way that you manage HPAs is different based on your version of the Kubernetes API:

- **For Kubernetes API version autoscaling/V2beta1:** This version of the Kubernetes API lets you autoscale your pods based on the CPU and memory utilization of your application.
- **For Kubernetes API Version autoscaling/V2beta2:** This version of the Kubernetes API lets you autoscale your pods based on CPU and memory utilization, in addition to custom metrics.

You can create, manage, and delete HPAs using the Rancher UI. From the Rancher UI you can configure the HPA to scale based on CPU and memory utilization. For more information, refer to Managing HPAs with the Rancher UI. To scale the HPA based on custom metrics, you still need to use `kubectl`. For more information, refer to Configuring HPA to Scale Using Custom Metrics with Prometheus.

Clusters created in Rancher v2.0.7 and higher automatically have all the requirements needed (metrics-server and Kubernetes cluster configuration) to use HPA.

#### Testing HPAs with a Service Deployment

You can see your HPA's current number of replicas by going to your project and clicking **Resources › HPA**. For more information, refer to Get HPA Metrics and Status.

You can also use `kubectl` to get the status of HPAs that you test with your load testing tool. For more information, refer to Testing HPAs with kubectl.

### Background Information on HPAs

The Horizontal Pod Autoscaler [https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/] (HPA) is a Kubernetes feature that allows you to configure your cluster to automatically scale the services it's running up or down. This section provides explanation on how HPA works with Kubernetes.

#### Why Use Horizontal Pod Autoscaler?

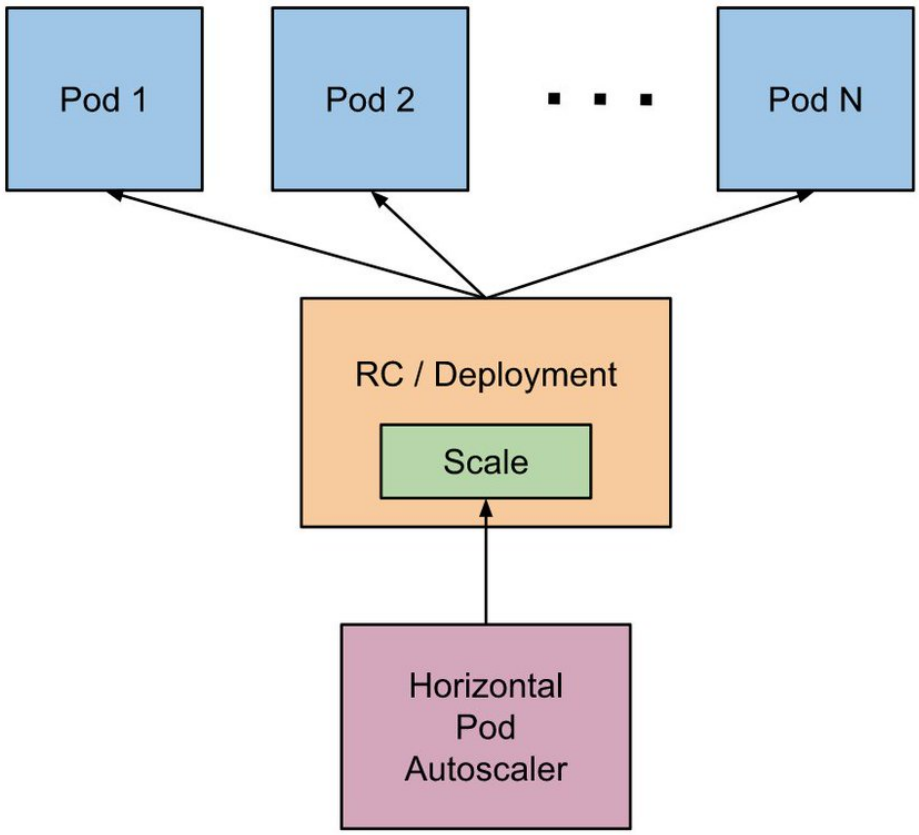
Using HPA, you can automatically scale the number of pods within a replication controller, deployment, or replica set up or down. HPA automatically scales the number of pods that are running for maximum efficiency. Factors that affect the number of pods include:

- A minimum and maximum number of pods allowed to run, as defined by the user.
- Observed CPU/memory use, as reported in resource metrics.
- Custom metrics provided by third-party metrics application like Prometheus, Datadog, etc.

HPA improves your services by:

- Releasing hardware resources that would otherwise be wasted by an excessive number of pods.
- Increase/decrease performance as needed to accomplish service level agreements.

#### How HPA Works




HPA is implemented as a control loop, with a period controlled by the `kube-controller-manager` flags below:

| Flag | Default | Description |
|---|---|---|
| `--horizontal-pod-autoscaler-sync-period` | `30s` | How often HPA audits resource/custom metrics in a deployment. |
| `--horizontal-pod-autoscaler-downscale-delay` | `5m0s` | Following completion of a downscale operation, how long HPA must wait before launching another downscale operations. |
| `--horizontal-pod-autoscaler-upscale-delay` | `3m0s` | Following completion of an upscale operation, how long HPA must wait before launching another upscale operation. |

For full documentation on HPA, refer to the Kubernetes Documentation [https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/].

#### Horizontal Pod Autoscaler API Objects

HPA is an API resource in the Kubernetes `autoscaling` API group. The current stable version is `autoscaling/v1`, which only includes support for CPU autoscaling. To get additional support for scaling based on memory and custom metrics, use the beta version instead: `autoscaling/v2beta1`.

For more information about the HPA API object, see the HPA GitHub Readme [https://github.com/kubernetes/design-proposals-archive/blob/main/autoscaling/horizontal-pod-autoscaler.md#horizontalpodautoscaler-object].

### Managing HPAs with the SUSE Rancher Prime UI

The Rancher UI supports creating, managing, and deleting HPAs. You can configure CPU or memory usage as the metric that the HPA uses to scale.

If you want to create HPAs that scale based on other metrics than CPU and memory, refer to Configuring HPA to Scale Using Custom Metrics with Prometheus.

#### Creating an HPA

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster you want to create an HPA in and click **Explore**.
3. In the left navigation bar, click **Service Discovery** › **HorizontalPodAutoscalers**.
4. Click **Create**.
5. Select a **Namespace** for the HPA.
6. Enter a **Name** for the HPA.
7. Select a **Target Reference** as scale target for the HPA.
8. Specify the **Minimum Replicas** and **Maximum Replicas** for the HPA.
9. Configure the metrics for the HPA. You can choose memory or CPU usage as the metric that will cause the HPA to scale the service up or down. In the **Quantity** field, enter the percentage of the workload's memory or CPU usage that will cause the HPA to scale the service. To configure other HPA metrics, including metrics available from Prometheus, you need to manage HPAs using kubectl.
10. Click **Create** to create the HPA.



*Result:*

The HPA is deployed to the chosen namespace. You can view the HPA's status from the project's Resources > HPA view.

#### Get HPA Metrics and Status

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster that has the HPA and click **Explore**.
3. In the left navigation bar, click **Service Discovery** › **HorizontalPodAutoscalers**. The **HorizontalPodAutoscalers** page shows the number of current replicas.

For more detailed metrics and status of a specific HPA, click the name of the HPA. This leads to the HPA detail page.

#### Deleting an HPA

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster that has the HPA you want to delete and click **Explore**.
3. In the left navigation bar, click **Service Discovery** › **HorizontalPodAutoscalers**.
4. Click **Resources** › **HPA**.
5. Find the HPA which you would like to delete and click ☰ > **Delete**.
6. Click **Delete** to confirm.



*Result:*

The HPA is deleted from the current cluster.

### Managing HPAs with kubectl

This section describes HPA management with `kubectl`. This document has instructions for how to:

- Create an HPA
- Get information on HPAs
- Delete an HPA
- Configure your HPAs to scale with CPU or memory utilization
- Configure your HPAs to scale using custom metrics, if you use a third-party tool such as Prometheus for metrics

You can create, view, and delete HPAs from the Rancher UI. You can also configure them to scale based on CPU or memory usage from the Rancher UI. For more information, refer to Managing HPAs with the Rancher UI. For scaling HPAs based on other metrics than CPU or memory, you still need `kubectl`.

#### Basic kubectl Command for Managing HPAs

If you have an HPA manifest file, you can create, manage, and delete HPAs using `kubectl`:

- Creating HPA
  - With manifest: `kubectl create -f <HPA_MANIFEST>`

- Without manifest (Just support CPU): `kubectl autoscale deployment hello-world --min=2 --max=5 --cpu-percent=50`

- Getting HPA info
  - Basic: `kubectl get hpa hello-world`
  - Detailed description: `kubectl describe hpa hello-world`
- Deleting HPA
  - `kubectl delete hpa hello-world`

#### HPA Manifest Definition Example

The HPA manifest is the config file used for managing an HPA with `kubectl`.

The following snippet demonstrates use of different directives in an HPA manifest. See the list below the sample to understand the purpose of each directive.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: hello-world
spec:
  scaleTargetRef:
    apiVersion: extensions/v1beta1
    kind: Deployment
    name: hello-world
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
  - type: Resource
    resource:
      name: memory
      targetAverageValue: 100Mi
```

| Directive | Description |
|---|---|
| `apiVersion: autoscaling/v2beta1` | The version of the Kubernetes `autoscaling` API group in use. This example manifest uses the beta version, so scaling by CPU and memory is enabled. |
| `name: hello-world` | Indicates that HPA is performing autoscaling for the `hello-word` deployment. |
| `minReplicas: 1` | Indicates that the minimum number of replicas running can't go below 1. |

| Directive | Description |
|---|---|
| `maxReplicas: 10` | Indicates the maximum number of replicas in the deployment can't go above 10. |
| `targetAverageUtilization: 50` | Indicates the deployment will scale pods up when the average running pod uses more than 50% of its requested CPU. |
| `targetAverageValue: 100Mi` | Indicates the deployment will scale pods up when the average running pod uses more that 100Mi of memory. |

#### Configuring HPA to Scale Using Resource Metrics (CPU and Memory)

Clusters created in Rancher v2.0.7 and higher have all the requirements needed (metrics-server and Kubernetes cluster configuration) to use Horizontal Pod Autoscaler.

Run the following commands to check if metrics are available in your installation:

```
$ kubectl top nodes
NAME                              CPU(cores)   CPU%      MEMORY(bytes)
MEMORY%
node-controlplane   196m         9%        1623Mi          42%
node-etcd           80m          4%        1090Mi          28%
node-worker         64m          3%        1146Mi          29%
$ kubectl -n kube-system top pods
NAME                                    CPU(cores)   MEMORY(bytes)
canal-pgldr                             18m          46Mi
canal-vhkgr                             20m          45Mi
canal-x5q5v                             17m          37Mi
canal-xknnz                             20m          37Mi
kube-dns-7588d5b5f5-298j2               0m           22Mi
kube-dns-autoscaler-5db9bbb766-t24hw    0m           5Mi
metrics-server-97bc649d5-jxrlt          0m           12Mi
$ kubectl -n kube-system logs -l k8s-app=metrics-server
I1002 12:55:32.172841       1 heapster.go:71] /metrics-server
--source=kubernetes.summary_api:https://kubernetes.default.svc?kubeletHt
tps=true&kubeletPort=10250&useServiceAccount=true&insecure=true
I1002 12:55:32.172994       1 heapster.go:72] Metrics Server version
v0.2.1
I1002 12:55:32.173378       1 configs.go:61] Using Kubernetes client
with master "https://kubernetes.default.svc" and version
I1002 12:55:32.173401       1 configs.go:62] Using kubelet port 10250
I1002 12:55:32.173946       1 heapster.go:128] Starting with Metric Sink
I1002 12:55:32.592703       1 serving.go:308] Generated self-signed cert
(apiserver.local.config/certificates/apiserver.crt,
apiserver.local.config/certificates/apiserver.key)
I1002 12:55:32.925630       1 heapster.go:101] Starting Heapster API
server...
[restful] 2018/10/02 12:55:32 log.go:33: [restful/swagger] listing is
```

```
available at https:///swaggerapi
[restful] 2018/10/02 12:55:32 log.go:33: [restful/swagger]
https:///swaggerui/ is mapped to folder /swagger-ui/
I1002 12:55:32.928597       1 serve.go:85] Serving securely on
0.0.0.0:443
```

#### Configuring HPA to Scale Using Custom Metrics with Prometheus

You can configure HPA to autoscale based on custom metrics provided by third-party software. The most common use case for autoscaling using third-party software is based on application-level metrics (i.e., HTTP requests per second). HPA uses the `custom.metrics.k8s.io` API to consume these metrics. This API is enabled by deploying a custom metrics adapter for the metrics collection solution.

For this example, we are going to use Prometheus [https://prometheus.io/]. We are beginning with the following assumptions:

- Prometheus is deployed in the cluster.
- Prometheus is configured correctly and collecting proper metrics from pods, nodes, namespaces, etc.
- Prometheus is exposed at the following URL and port: `http://prometheus.mycompany.io:80`

Prometheus is available for deployment in the Rancher v2.0 catalog. Deploy it from Rancher catalog if it isn't already running in your cluster.

For HPA to use custom metrics from Prometheus, package k8s-prometheus-adapter [https://github.com/DirectXMan12/k8s-prometheus-adapter] is required in the `kube-system` namespace of your cluster. To install `k8s-prometheus-adapter`, we are using the Helm chart available at banzai-charts [https://github.com/banzaicloud/banzai-charts].

1. Initialize Helm in your cluster.

```
  # kubectl -n kube-system create serviceaccount tiller
  kubectl create clusterrolebinding tiller --clusterrole cluster-
admin --serviceaccount=kube-system:tiller
  helm init --service-account tiller
```

2. Clone the `banzai-charts` repo from GitHub:

```
  # git clone https://github.com/banzaicloud/banzai-charts
```

3. Install the `prometheus-adapter` chart, specifying the Prometheus URL and port number.

```
  # helm install --name prometheus-adapter banzai-charts/prometheus-
adapter --set
prometheus.url="http://prometheus.mycompany.io",prometheus.port="80"
--namespace kube-system
```

4. Check that `prometheus-adapter` is running properly. Check the service pod and logs in the `kube-system` namespace.
5. Check that the service pod is `Running`. Enter the following command. ` # kubectl get pods -n kube-system ` From the resulting output, look for a status of `Running`. `+ NAME READY STATUS RESTARTS AGE … prometheus-adapter-prometheus-adapter-568674d97f-hbzfx 1/1 Running 0 7h … +`
6. Check the service logs to make sure the service is running correctly by entering the command that follows. ` # kubectl logs prometheus-adapter-prometheus-adapter-568674d97f-hbzfx -n kube-system ` Then review the log output to confirm the service is running.

```
.Prometheus Adaptor Logs [%collapsible] ====== ...
I0724 10:18:45.696679       1 round_trippers.go:436] GET
https://10.43.0.1:443/api/v1/namespaces/default/pods?labelSelector=ap
p%3Dhello-world 200 OK in 2 milliseconds
I0724 10:18:45.696695       1 round_trippers.go:442] Response
Headers:
I0724 10:18:45.696699       1 round_trippers.go:445]     Date: Tue,
24 Jul 2018 10:18:45 GMT
I0724 10:18:45.696703       1 round_trippers.go:445]     Content-
Type: application/json
I0724 10:18:45.696706       1 round_trippers.go:445]     Content-
Length: 2581
I0724 10:18:45.696766       1 request.go:836] Response Body:
{"kind":"PodList","apiVersion":"v1","metadata":{"selfLink":"/api/v1/n
amespaces/default/pods","resourceVersion":"6237"},"items":[{"metadata
":{"name":"hello-world-54764dfbf8-q6l82","generateName":"hello-world-
54764dfbf8-
","namespace":"default","selfLink":"/api/v1/namespaces/default/pods/h
ello-world-54764dfbf8-q6l82","uid":"484cb929-8f29-11e8-99d2-
067cac34e79c","resourceVersion":"4066","creationTimestamp":"2018-07-
24T10:06:50Z","labels":{"app":"hello-world","pod-template-
hash":"1032089694"},"annotations":{"cni.projectcalico.org/podIP":"10.
42.0.7/32"},"ownerReferences":[{"apiVersion":"extensions/v1beta1","ki
nd":"ReplicaSet","name":"hello-world-54764dfbf8","uid":"4849b9b1-
8f29-11e8-99d2-
067cac34e79c","controller":true,"blockOwnerDeletion":true}]},"spec":{
"volumes":[{"name":"default-token-
ncvts","secret":{"secretName":"default-token-
ncvts","defaultMode":420}}],"containers":[{"name":"hello-
world","image":"rancher/hello-
world","ports":[{"containerPort":80,"protocol":"TCP"}],"resources":{"
requests":{"cpu":"500m","memory":"64Mi"}},"volumeMounts":[{"name":"de
fault-token-
ncvts","readOnly":true,"mountPath":"/var/run/secrets/kubernetes.io/se
rviceaccount"}],"terminationMessagePath":"/dev/termination-
```

```
log","terminationMessagePolicy":"File","imagePullPolicy":"Always"}],"
restartPolicy":"Always","terminationGracePeriodSeconds":30,"dnsPolicy
":"ClusterFirst","serviceAccountName":"default","serviceAccount":"def
ault","nodeName":"34.220.18.140","securityContext":{},"schedulerName"
:"default-scheduler","tolerations":[{"key":"node.kubernetes.io/not-
ready","operator":"Exists","effect":"NoExecute","tolerationSeconds":3
00},{"key":"node.kubernetes.io/unreachable","operator":"Exists","effe
ct":"NoExecute","tolerationSeconds":300}]},"status":{"phase":"Running
","conditions":[{"type":"Initialized","status":"True","lastProbeTime"
:null,"lastTransitionTime":"2018-07-
24T10:06:50Z"},{"type":"Ready","status":"True","lastProbeTime":null,"
lastTransitionTime":"2018-07-
24T10:06:54Z"},{"type":"PodScheduled","status":"True","lastProbeTime"
:null,"lastTransitionTime":"2018-07-
24T10:06:50Z"}],"hostIP":"34.220.18.140","podIP":"10.42.0.7","startTi
me":"2018-07-24T10:06:50Z","containerStatuses":[{"name":"hello-
world","state":{"running":{"startedAt":"2018-07-
24T10:06:54Z"}},"lastState":{},"ready":true,"restartCount":0,"image":
"rancher/hello-world:latest","imageID":"docker-
pullable://rancher/hello-
world@sha256:4b1559cb4b57ca36fa2b313a3c7dde77480laa3a2047930d94ella45
168bc053","containerID":"docker://cce4df5fc0408f03d4adf82c90de222f64c
302bf7a04belc82d584ec31530773"}],"qosClass":"Burstable"}}]}
I0724 10:18:45.699525       1 api.go:74] GET http://prometheus-
server.prometheus.34.220.18.140.xip.io/api/v1/query?query=sum%28rate%
28container_fs_read_seconds_total%7Bpod_name%3D%22hello-world-
54764dfbf8-
q6l82%22%2Ccontainer_name%21%3D%22POD%22%2Cnamespace%3D%22default%22%
7D%5B5m%5D%29%29+by+%28pod_name%29&time=1532427525.697 200 OK
I0724 10:18:45.699620       1 api.go:93] Response Body:
{"status":"success","data":{"resultType":"vector","result":[{"metric"
:{"pod_name":"hello-world-54764dfbf8-
q6l82"},"value":[1532427525.697,"0"]}]}}
I0724 10:18:45.699939       1 wrap.go:42] GET
/apis/custom.metrics.k8s.io/v1beta1/namespaces/default/pods/%2A/fs_re
ad?labelSelector=app%3Dhello-world: (12.431262ms) 200 [[kube-
controller-manager/v1.10.1 (linux/amd64)
kubernetes/d4ab475/system:serviceaccount:kube-system:horizontal-pod-
autoscaler] 10.42.0.0:24268]
I0724 10:18:51.727845       1 request.go:836] Request Body:
{"kind":"SubjectAccessReview","apiVersion":"authorization.k8s.io/v1be
ta1","metadata":{"creationTimestamp":null},"spec":{"nonResourceAttrib
utes":{"path":"/","verb":"get"},"user":"system:anonymous","group":["s
ystem:unauthenticated"]},"status":{"allowed":false}}
... ======
```

7. Check that the metrics API is accessible from kubectl.

   - If you are accessing the cluster directly, enter your Server URL in the kubectl config in the following format: `https://<Kubernetes_URL>:6443`.

     ```
     # kubectl get --raw /apis/custom.metrics.k8s.io/v1beta1
     ```

     If the API is accessible, you should receive output that's similar to what follows.

     - *API Response*

       {"kind":"APIResourceList","apiVersion":"v1","groupVersion":"custom.metrics.k8s.io/v1beta1","resources":[{"name":"pods/fs_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_rss","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_period","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_throttled","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_time","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_read","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_sector_writes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_user","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/last_seen","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/tasks_state","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_quota","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/start_time_seconds","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_write","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_cache","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_periods","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_throttled_periods","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads_merged","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_working_set_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/network_udp_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_inodes_free","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_inodes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_time_weighted","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_failures","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_swap","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_shares","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_swap_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_current","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_failcnt","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes_merged","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"

name":"pods/network_tcp_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_max_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_reservation_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_load_average_10s","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_system","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_sector_reads","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]}]}

- If you are accessing the cluster through Rancher, enter your Server URL in the kubectl config in the following format: `https://<RANCHER_URL>/k8s/clusters/<CLUSTER_ID>`. Add the suffix `/k8s/clusters/<CLUSTER_ID>` to API path.

```
# kubectl get --raw
/k8s/clusters/<CLUSTER_ID>/apis/custom.metrics.k8s.io/v1beta1
```

If the API is accessible, you should receive output that's similar to what follows.

- *API Response*

{"kind":"APIResourceList","apiVersion":"v1","groupVersion":"custom.metrics.k8s.io/v1beta1","resources":[{"name":"pods/fs_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_rss","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_period","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_throttled","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_time","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_read","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_sector_writes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_user","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/last_seen","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/tasks_state","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_quota","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/start_time_seconds","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_write","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_cache","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_periods","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_cfs_throttled_periods","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads_merged","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_working_set_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/network_udp_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_inodes_free","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_inodes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_time_weighted","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_failures","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"po

ds/memory_swap","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_cpu_shares","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_swap_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_io_current","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_failcnt","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_writes_merged","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/network_tcp_usage","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/memory_max_usage_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/spec_memory_reservation_limit_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_load_average_10s","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/cpu_system","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_reads_bytes","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]},{"name":"pods/fs_sector_reads","singularName":"","namespaced":true,"kind":"MetricValueList","verbs":["get"]}]}

**Testing HPAs with kubectl**

This document describes how to check the status of your HPAs after scaling them up or down with your load testing tool. For information on how to check the status from the Rancher UI (at least version 2.3.x), refer to Managing HPAs with the Rancher UI.

For HPA to work correctly, service deployments should have resources request definitions for containers. Follow this hello-world example to test if HPA is working correctly.

1. Configure `kubectl` to connect to your Kubernetes cluster.
2. Copy the `hello-world` deployment manifest below.

*Hello World Manifest*

+

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  labels:
    app: hello-world
  name: hello-world
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
```

```
    strategy:
      rollingUpdate:
        maxSurge: 1
        maxUnavailable: 0
      type: RollingUpdate
    template:
      metadata:
        labels:
          app: hello-world
      spec:
        containers:
        - image: rancher/hello-world
          imagePullPolicy: Always
          name: hello-world
          resources:
            requests:
              cpu: 500m
              memory: 64Mi
          ports:
          - containerPort: 80
            protocol: TCP
        restartPolicy: Always
  ---
  apiVersion: v1
  kind: Service
  metadata:
    name: hello-world
    namespace: default
  spec:
    ports:
    - port: 80
      protocol: TCP
      targetPort: 80
    selector:
      app: hello-world
```

+

3. Deploy it to your cluster.

```
# kubectl create -f <HELLO_WORLD_MANIFEST>
```

4. Copy one of the HPAs below based on the metric type you're using:

*Hello World HPA: Resource Metrics*

+

```
    apiVersion: autoscaling/v2betal
    kind: HorizontalPodAutoscaler
    metadata:
      name: hello-world
      namespace: default
    spec:
      scaleTargetRef:
        apiVersion: extensions/v1betal
        kind: Deployment
        name: hello-world
      minReplicas: 1
      maxReplicas: 10
      metrics:
      - type: Resource
        resource:
          name: cpu
          targetAverageUtilization: 50
      - type: Resource
        resource:
          name: memory
          targetAverageValue: 1000Mi
```

+

+

```
  apiVersion: autoscaling/v2betal
  kind: HorizontalPodAutoscaler
  metadata:
    name: hello-world
    namespace: default
  spec:
    scaleTargetRef:
      apiVersion: extensions/v1betal
      kind: Deployment
      name: hello-world
    minReplicas: 1
    maxReplicas: 10
    metrics:
    - type: Resource
      resource:
        name: cpu
```

```
        targetAverageUtilization: 50
    - type: Resource
      resource:
        name: memory
        targetAverageValue: 100Mi
    - type: Pods
      pods:
        metricName: cpu_system
        targetAverageValue: 20m
```

+

1. View the HPA info and description. Confirm that metric data is shown.

   *Resource Metrics*

   1. Enter the following commands. ` # kubectl get hpa NAME REFERENCE TARGETS MINPODS MAXPODS REPLICAS AGE hello-world Deployment/hello-world 1253376 / 100Mi, 0% / 50% 1 10 1 6m # kubectl describe hpa Name: hello-world Namespace: default Labels: <none> Annotations: <none> CreationTimestamp: Mon, 23 Jul 2018 20:21:16 +0200 Reference: Deployment/hello-world Metrics: ( current / target ) resource memory on pods: 1253376 / 100Mi resource cpu on pods (as a percentage of request): 0% (0) / 50% Min replicas: 1 Max replicas: 10 Conditions: Type Status Reason Message ---- ------ ------ ------- AbleToScale True ReadyForNewScale the last scale time was sufficiently old as to warrant a new scale ScalingActive True ValidMetricFound the HPA was able to successfully calculate a replica count from memory resource ScalingLimited False DesiredWithinRange the desired count is within the acceptable range Events: <none> `

1. Enter the following command. ` # kubectl describe hpa ` You should receive the output that follows. ` Name: hello-world Namespace: default Labels: <none> Annotations: <none> CreationTimestamp: Tue, 24 Jul 2018 18:36:28 +0200 Reference: Deployment/hello-world Metrics: ( current / target ) resource memory on pods: 3514368 / 100Mi "cpu_system" on pods: 0 / 20m resource cpu on pods (as a percentage of request): 0% (0) / 50% Min replicas: 1 Max replicas: 10 Conditions: Type Status Reason Message ---- ------ ------ ------- AbleToScale True ReadyForNewScale the last scale time was sufficiently old as to warrant a new scale ScalingActive True ValidMetricFound the HPA was able to successfully calculate a replica count from memory resource ScalingLimited False DesiredWithinRange the desired count is within the acceptable range Events: <none> `

1. Generate a load for the service to test that your pods autoscale as intended. You can use any load-testing tool (Hey, Gatling, etc.), but we're using Hey [https://github.com/rakyll/hey].
2. Test that pod autoscaling works as intended.

   **To Test Autoscaling Using Resource Metrics:**

   *Upscale to 2 Pods: CPU Usage Up to Target*

   Use your load testing tool to scale up to two pods based on CPU Usage.

a. View your HPA.

```
# kubectl describe hpa
```

You should receive output similar to what follows.

```
 Name:                                                  hello-
world
 Namespace:                                             default
 Labels:                                                <none>
 Annotations:                                           <none>
 CreationTimestamp:                                     Mon, 23
Jul 2018 22:22:04 +0200
 Reference:
Deployment/hello-world
 Metrics:                                               ( current
/ target )
   resource memory on pods:                             10928128
/ 100Mi
   resource cpu on pods  (as a percentage of request):  56% (
280m) / 50%
 Min replicas:                                          1
 Max replicas:                                          10
 Conditions:
   Type            Status  Reason              Message
   ----            ------  ------              -------
   AbleToScale     True    SucceededRescale    the HPA controller
was able to update the target scale to 2
   ScalingActive   True    ValidMetricFound    the HPA was able
to successfully calculate a replica count from cpu resource
utilization (percentage of request)
   ScalingLimited  False   DesiredWithinRange  the desired count
is within the acceptable range
 Events:
   Type    Reason             Age   From
Message
   ----    ------             ----  ----
-------
   Normal  SuccessfulRescale  13s   horizontal-pod-autoscaler
New size: 2; reason: cpu resource utilization (percentage of
request) above target
```

b. Enter the following command to confirm you've scaled to two pods.

```
# kubectl get pods
```

You should receive output similar to what follows:

```
NAME                                                READY
STATUS    RESTARTS   AGE
hello-world-54764dfbf8-k8ph2                        1/1
Running   0          1m
hello-world-54764dfbf8-q6l4v                        1/1
Running   0          3h
```

Use your load testing tool to upscale to 3 pods based on CPU usage with `horizontal-pod-autoscaler-upscale-delay` set to 3 minutes.

a. Enter the following command.

```
# kubectl describe hpa
```

You should receive output similar to what follows

```
Name:                                                  hello-
world
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Mon, 23
Jul 2018 22:22:04 +0200
Reference:
Deployment/hello-world
Metrics:                                               (
current / target )
  resource memory on pods:                             9424896
/ 100Mi
  resource cpu on pods  (as a percentage of request):  66%
(333m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    SucceededRescale    the HPA
controller was able to update the target scale to 3
```

```
     ScalingActive   True    ValidMetricFound    the HPA was able
to successfully calculate a replica count from cpu resource
utilization (percentage of request)
     ScalingLimited  False   DesiredWithinRange  the desired
count is within the acceptable range
   Events:
     Type    Reason             Age   From
Message
     ----    ------             ----  ----
-------
     Normal  SuccessfulRescale  4m    horizontal-pod-autoscaler
New size: 2; reason: cpu resource utilization (percentage of
request) above target
     Normal  SuccessfulRescale  16s   horizontal-pod-autoscaler
New size: 3; reason: cpu resource utilization (percentage of
request) above target
```

b. Enter the following command to confirm three pods are running.

```
# kubectl get pods
```

You should receive output similar to what follows.

```
   NAME                                                  READY
STATUS    RESTARTS   AGE
   hello-world-54764dfbf8-f46kh                          0/1
Running   0          1m
   hello-world-54764dfbf8-k8ph2                          1/1
Running   0          5m
   hello-world-54764dfbf8-q6l4v                          1/1
Running   0          3h
```

*Downscale to 1 Pod: All Metrics Below Target*

Use your load testing to scale down to 1 pod when all metrics are below target for `horizontal-pod-autoscaler-downscale-delay` (5 minutes by default).

a. Enter the following command.

```
   # kubectl describe hpa
```

You should receive output similar to what follows.

```
   Name:                                                  hello-
world
   Namespace:                                             default
   Labels:                                                <none>
   Annotations:                                           <none>
   CreationTimestamp:                                     Mon, 23
Jul 2018 22:22:04 +0200
   Reference:
Deployment/hello-world
   Metrics:                                               ( current
/ target )
     resource memory on pods:                             10070016 /
100Mi
     resource cpu on pods  (as a percentage of request):  0% (0) /
50%
   Min replicas:                                          1
   Max replicas:                                          10
   Conditions:
     Type            Status  Reason              Message
     ----            ------  ------              -------
     AbleToScale     True    SucceededRescale    the HPA controller
was able to update the target scale to 1
     ScalingActive   True    ValidMetricFound    the HPA was able to
successfully calculate a replica count from memory resource
     ScalingLimited  False   DesiredWithinRange  the desired count
is within the acceptable range
   Events:
     Type    Reason             Age   From
Message
     ----    ------             ----  ----
-------
     Normal  SuccessfulRescale  10m   horizontal-pod-autoscaler  New
size: 2; reason: cpu resource utilization (percentage of request)
above target
     Normal  SuccessfulRescale  6m    horizontal-pod-autoscaler  New
size: 3; reason: cpu resource utilization (percentage of request)
above target
     Normal  SuccessfulRescale  1s    horizontal-pod-autoscaler  New
size: 1; reason: All metrics below target
```

+ **To Test Autoscaling Using Custom Metrics:**

+ .Upscale to 2 Pods: CPU Usage Up to Target

*Details*

Use your load testing tool to upscale two pods based on CPU usage.

a. Enter the following command. ` # kubectl describe hpa ` You should receive output similar to what follows. ` Name: hello-world Namespace: default Labels: <none> Annotations: <none> CreationTimestamp: Tue, 24 Jul 2018 18:01:11 +0200 Reference: Deployment/hello-world Metrics: ( current / target ) resource memory on pods: 8159232 / 100Mi "cpu_system" on pods: 7m / 20m resource cpu on pods (as a percentage of request): 64% (321m) / 50% Min replicas: 1 Max replicas: 10 Conditions: Type Status Reason Message ---- ------ ------ ------- AbleToScale True SucceededRescale the HPA controller was able to update the target scale to 2 ScalingActive True ValidMetricFound the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request) ScalingLimited False DesiredWithinRange the desired count is within the acceptable range Events: Type Reason Age From Message ---- ------ ---- ---- ------- Normal SuccessfulRescale 16s horizontal-pod-autoscaler New size: 2; reason: cpu resource utilization (percentage of request) above target `

b. Enter the following command to confirm two pods are running. ` # kubectl get pods ` You should receive output similar to what follows. ` NAME READY STATUS RESTARTS AGE hello-world-54764dfbf8-5pfdr 1/1 Running 0 3s hello-world-54764dfbf8-q6l82 1/1 Running 0 6h `

*Upscale to 3 Pods: CPU Usage Up to Target*

Use your load testing tool to scale up to three pods when the cpu_system usage limit is up to target. .. Enter the following command.

+

```
# kubectl describe hpa
```

+ You should receive output similar to what follows:

+

```
   Name:                                                  hello-world
   Namespace:                                             default
   Labels:                                                <none>
   Annotations:                                           <none>
   CreationTimestamp:                                     Tue, 24 Jul
2018 18:01:11 +0200
   Reference:
Deployment/hello-world
   Metrics:                                               ( current /
target )
     resource memory on pods:                             8374272 /
100Mi
     "cpu_system" on pods:                                27m / 20m
     resource cpu on pods  (as a percentage of request):  71% (357m) /
50%
   Min replicas:                                          1
   Max replicas:                                          10
```

```
   Conditions:
     Type            Status  Reason              Message
     ----            ------  ------              -------
     AbleToScale     True    SucceededRescale    the HPA controller
was able to update the target scale to 3
     ScalingActive   True    ValidMetricFound    the HPA was able to
successfully calculate a replica count from cpu resource utilization
(percentage of request)
     ScalingLimited  False   DesiredWithinRange  the desired count is
within the acceptable range
   Events:
     Type    Reason             Age   From
Message
     ----    ------             ----  ----
-------
     Normal  SuccessfulRescale  3m    horizontal-pod-autoscaler  New
size: 2; reason: cpu resource utilization (percentage of request)
above target
     Normal  SuccessfulRescale  3s    horizontal-pod-autoscaler  New
size: 3; reason: pods metric cpu_system above target
```

a. Enter the following command to confirm three pods are running.

```
# kubectl get pods
```

You should receive output similar to what follows:

```
# kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
hello-world-54764dfbf8-5pfdr  1/1       Running   0          3m
hello-world-54764dfbf8-m2hrl  1/1       Running   0          1s
hello-world-54764dfbf8-q6l82  1/1       Running   0          6h
```

*Upscale to 4 Pods: CPU Usage Up to Target*

Use your load testing tool to upscale to four pods based on CPU usage. `horizontal-pod-autoscaler-upscale-delay` is set to three minutes by default.

a. Enter the following command. ` # kubectl describe hpa ` You should receive output similar to what follows. ` Name: hello-world Namespace: default Labels: <none> Annotations: <none> CreationTimestamp: Tue, 24 Jul 2018 18:01:11 +0200 Reference: Deployment/hello-world Metrics: ( current / target ) resource memory on pods: 8374272 / 100Mi "cpu_system" on pods: 27m / 20m resource cpu on pods (as a percentage of request): 71% (357m) / 50% Min replicas: 1 Max replicas: 10 Conditions: Type Status Reason Message ---- ------ ------ ------- AbleToScale True SucceededRescale the HPA controller was able to update the target scale to 3 ScalingActive True ValidMetricFound the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of

request) ScalingLimited False DesiredWithinRange the desired count is within the acceptable range Events: Type Reason Age From Message ---- ------ ---- ---- ------- Normal SuccessfulRescale 5m horizontal-pod-autoscaler New size: 2; reason: cpu resource utilization (percentage of request) above target Normal SuccessfulRescale 3m horizontal-pod-autoscaler New size: 3; reason: pods metric cpu_system above target Normal SuccessfulRescale 4s horizontal-pod-autoscaler New size: 4; reason: cpu resource utilization (percentage of request) above target `

b. Enter the following command to confirm four pods are running.

```
# kubectl get pods
```

You should receive output similar to what follows.

```
  NAME                              READY     STATUS    RESTARTS   AGE
  hello-world-54764dfbf8-2p9xb   1/1       Running   0          5m
  hello-world-54764dfbf8-5pfdr   1/1       Running   0          2m
  hello-world-54764dfbf8-m2hrl   1/1       Running   0          1s
  hello-world-54764dfbf8-q6l82   1/1       Running   0          6h
```

*Downscale to 1 Pod: All Metrics Below Target*

Use your load testing tool to scale down to one pod when all metrics below target for `horizontal-pod-autoscaler-downscale-delay`.

a. Enter the following command.

```
 # kubectl describe hpa
```

You should receive similar output to what follows.

```
     Name:                                                  hello-
world
     Namespace:                                             default
     Labels:                                                <none>
     Annotations:                                           <none>
     CreationTimestamp:                                     Tue, 24
Jul 2018 18:01:11 +0200
     Reference:
Deployment/hello-world
     Metrics:                                               (
current / target )
       resource memory on pods:                             8101888
/ 100Mi
       "cpu_system" on pods:                                8m / 20m
       resource cpu on pods  (as a percentage of request):  0% (0) /
50%
```

```
     Min replicas:                                         1
     Max replicas:                                         10
     Conditions:
       Type            Status  Reason              Message
       ----            ------  ------              -------
       AbleToScale     True    SucceededRescale    the HPA
controller was able to update the target scale to 1
       ScalingActive   True    ValidMetricFound    the HPA was able
to successfully calculate a replica count from memory resource
       ScalingLimited  False   DesiredWithinRange  the desired count
is within the acceptable range
     Events:
       Type    Reason             Age   From
Message
       ----    ------             ----  ----
-------
       Normal  SuccessfulRescale  10m    horizontal-pod-autoscaler
New size: 2; reason: cpu resource utilization (percentage of
request) above target
       Normal  SuccessfulRescale  8m    horizontal-pod-autoscaler
New size: 3; reason: pods metric cpu_system above target
       Normal  SuccessfulRescale  5m    horizontal-pod-autoscaler
New size: 4; reason: cpu resource utilization (percentage of
request) above target
       Normal   SuccessfulRescale             13s
horizontal-pod-autoscaler  New size: 1; reason: All metrics below
target
```

b. Enter the following command to confirm a single pods is running.

```
     # kubectl get pods
```

You should receive output similar to what follows.

```
     NAME                           READY     STATUS    RESTARTS
AGE
     hello-world-54764dfbf8-q6l82   1/1       Running   0
6h
```

## Load Balancer and Ingress Controller Setup within SUSE Rancher Prime

Within Rancher, you can set up load balancers and ingress controllers to redirect service requests.

### Load Balancers

After you launch an application, the app is only available within the cluster. It can't be reached from outside the cluster.

If you want your applications to be externally accessible, you must add a load balancer or ingress to your cluster. Load balancers create a gateway for external connections to access your cluster, provided that the user knows the load balancer's IP address and the application's port number.

Rancher supports two types of load balancers:

- Layer-4 Load Balancers
- Layer-7 Load Balancers

For more information, see load balancers.

#### Load Balancer Limitations

Load Balancers have a couple of limitations you should be aware of:

- Load Balancers can only handle one IP address per service, which means if you run multiple services in your cluster, you must have a load balancer for each service. Running multiples load balancers can be expensive.
- If you want to use a load balancer with a Hosted Kubernetes cluster (i.e., clusters hosted in GKE, EKS, or AKS), the load balancer must be running within that cloud provider's infrastructure. Please review the compatibility tables regarding support for load balancers based on how you've provisioned your clusters:
- Support for Layer-4 Load Balancing
- Support for Layer-7 Load Balancing

### Ingress

As mentioned in the limitations above, the disadvantages of using a load balancer are:

- Load Balancers can only handle one IP address per service.
- If you run multiple services in your cluster, you must have a load balancer for each service.
- It can be expensive to have a load balancer for every service.

In contrast, when an ingress is used as the entry point into a cluster, the ingress can route traffic to multiple services with greater flexibility. It can map multiple HTTP requests to services without individual IP addresses for each service.

Therefore, it is useful to have an ingress if you want multiple services to be exposed with the same IP address, the same Layer 7 protocol, or the same privileged node-ports: 80 and 443.

Ingress works in conjunction with one or more ingress controllers to dynamically route service requests. When the ingress receives a request, the ingress controller(s) in your cluster direct the request to the correct service based on service subdomains or path rules that you've configured.

Your ingress, which creates a port of entry to your cluster similar to a load balancer, can reside within your cluster or externally.

Ingress can provide other functionality as well, such as SSL termination, name-based virtual hosting, and

more.

- For more information on how to set up ingress in Rancher, see Ingress.
- For complete information about ingress and ingress controllers, see the Kubernetes Ingress Documentation [https://kubernetes.io/docs/concepts/services-networking/ingress/].

### Layer 4 and Layer 7 Load Balancing

Kubernetes supports load balancing in two ways: Layer-4 Load Balancing and Layer-7 Load Balancing.

#### Layer-4 Load Balancer

Layer-4 load balancer (or the external load balancer) forwards traffic to Nodeports. Layer-4 load balancer allows you to forward both HTTP and TCP traffic.



It is possible to deploy a cluster with a non-cloud load balancer, such as MetalLB. [https://metallb.universe.tf/] However, that use case is more advanced than the Layer-4 load balancer supported by a cloud provider, and it is not configurable in Rancher.

#### Support for Layer-4 Load Balancing

Support for layer-4 load balancer varies based on the underlying cloud provider.

| Cluster Deployment | Layer-4 Load Balancer Support |
|---|---|
| Amazon EKS | Supported by AWS cloud provider |
| Google GKE | Supported by GCE cloud provider |
| Azure AKS | Supported by Azure cloud provider |
| Third-party MetalLB | third-party Ingress* |

#### Layer-7 Load Balancer

Layer-7 load balancer (or the ingress controller) supports host and path-based load balancing and SSL termination. Layer-7 load balancer only forwards HTTP and HTTPS traffic and therefore they listen on ports 80 and 443 only. Cloud providers such as Amazon and Google support layer-7 load balancer.

#### Support for Layer-7 Load Balancing

Support for layer-7 load balancer varies based on the underlying cloud provider.

| Cluster Deployment | Layer-7 Load Balancer Support |
|---|---|
| Amazon EKS | Supported by AWS cloud provider |
| Google GKE | Supported by GKE cloud provider |
| Azure AKS | Not Supported |

#### Host Names in Layer-7 Load Balancer

Some cloud-managed layer-7 load balancers (such as the ALB ingress controller on AWS) expose DNS addresses for ingress rules. You need to map (via CNAME) your domain name to the DNS address generated by the layer-7 load balancer.

Other layer-7 load balancers, such as the Google Load Balancer, directly expose one or more IP addresses.

Google Load Balancer provides a single routable IP address. You can do either of the following:

1. Configure your own DNS to map (via A records) your domain name to the IP addresses exposes by the Layer-7 load balancer.
2. Ask Rancher to generate an xip.io host name for your ingress rule. Rancher will take one of your exposed IPs, say `a.b.c.d`, and generate a host name `<ingressname>.<namespace>.a.b.c.d.xip.io`.

The benefit of using xip.io is that you obtain a working entrypoint URL immediately after you create the ingress rule. Setting up your own domain name, on the other hand, requires you to configure DNS servers and wait for DNS to propagate.

#### Related Links

- Create an External Load Balancer [https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/]

### Adding Ingresses

Ingresses can be added for workloads to provide load balancing, SSL termination and host/path based routing.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster that you want to add an ingress to and click **Explore**.
3. Click **Service Discovery** › **Ingresses**.
4. Click **Create**.
5. Select an existing **Namespace** from the drop-down list.
6. Enter a **Name** for the ingress.
7. Create ingress forwarding **Rules**. If any of your ingress rules handle requests for encrypted ports, add a certificate to encrypt/decrypt communications.
8. **Optional:** click **Add Rule** to create additional ingress rules. For example, after you create ingress rules to direct requests for your hostname, you'll likely want to create a default backend to handle 404s.
9. Click **Create** at the bottom right.

**Result:** Your ingress is added to the project. The ingress begins enforcing your ingress rules.

### Configuring an Ingress

#### Specify a hostname to use

If you use this option, Ingress routes requests for a hostname to the service or workload that you specify.

1. Specify a path of type `Prefix` and specify a path such as `/`.
2. Add a **Target Service**.
3. **Optional:** If you want to specify a workload or service when a request is sent to a particular hostname path, add a **Path** for the target. For example, if you want requests for `www.mysite.com/contact-us` to be sent to a different service than `www.mysite.com`, enter `/contact-us` in the **Path** field. Typically, the first rule that you create does not include a path.
4. Enter the **Port** number that each target operates on.

#### Certificates



You must have an SSL certificate that Ingress can use to encrypt and decrypt communications. For more information, see Adding SSL Certificates.

1. To create an Ingress controller, click the **Certificates** tab.
2. Click **Add Certificate**.
3. Select a **Certificate - Secret Name** from the drop-down list.
4. Enter the host using encrypted communication.
5. To add more hosts that use the same certificate, click **Add Hosts**.

#### Labels and Annotations

Please refer to the Traefik documentation for the full list of Ingress NGINX annotations that are supported [https://doc.traefik.io/traefik/reference/routing-configuration/kubernetes/ingress-nginx/#annotations-support] and unsupported [https://doc.traefik.io/traefik/reference/routing-configuration/kubernetes/ingress-nginx/#unsupported-annotations] by Traefik's kubernetesIngressNginx provider.

### Guide to Ingress NGINX Retirement

The Kubernetes SIG Network and the Security Response Committee announced the retirement of the Ingress NGINX project [https://kubernetes.io/blog/2025/11/11/ingress-nginx-retirement/]. Upstream best-effort maintenance, and all associated upstream releases, bug fixes, or security updates, ended March 2026.

To support users during this transition,

SUSE provides extended support timelines and clear migration paths to Traefik.

This guide centralizes the information on how to proceed based on your specific deployment scenario.

#### Support and Timelines

For organizations that cannot migrate immediately, SUSE Rancher Prime LTS provides extended support for RKE2 version 1.35.



*Extended support for Ingress NGINX*

SUSE Rancher Prime LTS provides patches for critical and important Common Vulnerabilities and Exposures (CVEs) for the Ingress NGINX component through December 2027. No new features will be added during this period.

#### Migration Paths by Environment

For organizations ready to move, SUSE offers a supported path to Traefik.

Traefik includes a compatibility layer that can interpret many existing Ingress NGINX annotations. Identify your cluster environment below to find the correct migration approach.

#### Standalone RKE2 Clusters

For standalone or imported RKE2 clusters currently using Ingress NGINX, follow the official migration guide for standalone RKE2 clusters [https://support.scc.suse.com/s/kb/Migrating-from-Ingress-NGINX-to-Traefik-in-a-

standalone-RKE2-cluster?language=en_US].

The migration process involves a four-phase strategy:

1. **Dual ingress controller setup:** Enable Traefik alongside Ingress NGINX using temporary, non-conflicting ports.
2. **Parallel migration and validation:** Duplicate Ingress resources and test Traefik's handling of the existing annotations.
3. **Final switchover:** Remove Ingress NGINX and configure Traefik to use the standard ports.
4. **Cleanup:** Delete the legacy Ingress objects.

#### Rancher Server on RKE2 (Local Clusters)

When migrating a Rancher local cluster, the Rancher Ingress resource requires specific handling to prevent lockouts. Follow the specific guide for migrating the Rancher Ingress to Traefik in an RKE2 cluster [https://support.scc.suse.com/s/kb/How-to-migrate-the-Rancher-Ingress-to-Traefik-in-an-RKE2-cluster?language=en_US]. This guide builds upon the standalone migration phases but includes steps tailored to the Rancher management server.

#### Downstream RKE2 Clusters (Provisioned by Rancher)

For RKE2 clusters provisioned and managed by SUSE Rancher Prime, migration options are integrated directly into the user interface.



This migration option is only possible in Rancher v2.13.5+ and v2.14.

The cluster configuration interface provides a Dual Mode migration option. This allows you to safely test and migrate traffic from Ingress NGINX to Traefik directly from the cluster management screen.

#### Rancher on Managed Kubernetes (Amazon EKS, Azure AKS, Google GKE)

If you run SUSE Rancher Prime on a managed Kubernetes service such as Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), or Google Kubernetes Engine (GKE), the recommendation is to migrate to Traefik.

SUSE Rancher Prime customers can leverage the Application Collection to deploy and manage the Traefik proxy.

#### Impact on Other SUSE Products

The retirement of Ingress NGINX impacts other

SUSE products

in the following ways:

- SUSE Storage: There is no impact on the Longhorn backend. However, administrators must reconfigure their Ingress to upgrade the Longhorn UI. For more information, refer to Create an Ingress with Basic Authentication Traefik [https://documentation.suse.com/cloudnative/storage/1.11/en/longhorn-system/system-access/longhorn-ingress-traefik.html].
- SUSE® Rancher Prime: Continuous Delivery: Configuring a webhook service is affected. Refer to the Fleet documentation [https://documentation.suse.com/cloudnative/continuous-delivery/latest/en/how-tos-for-users/

webhook.html] for more information.

- Ingress NGINX references in documentation for other projects have been removed.

## Services

Pod configuration is managed by Deployments, StatefulSets and Daemonsets, whereas services direct traffic to pods using selectors.

For every workload (with at least one port configured) created, a complementing Service Discovery entry is created. This Service Discovery entry enables DNS resolution for the workload's pods using the following naming convention: `<workload>.<namespace>.svc.cluster.local`.

You can create additional services so that a given namespace resolves with one or more external IP addresses, an external hostname, an alias to another DNS record, other workloads, or a set of pods that match a selector that you create.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Go to the cluster where you want to add a service and click **Explore**.
3. Click **Service Discovery › Services**.
4. Click **Create**.
5. Choose the type of service you want to create.
6. Select a **Namespace** from the drop-down list.
7. Enter a **Name** for the service. This name is used for DNS resolution.
8. Fill out the rest of the form. For help, refer to the upstream Kubernetes documentation about services. [https://kubernetes.io/docs/concepts/services-networking/service/]
9. Click **Create**.

**Result:** A new service is created.

- You can view the record by from the project's **Service Discovery** tab.
- When you visit the new DNS name for the new record that you created (`<recordname>.<namespace>.svc.cluster.local`), it resolves the chosen namespace.

#### Related Links

- Adding entries to Pod /etc/hosts with HostAliases [https://kubernetes.io/docs/concepts/services-networking/add-entries-to-pod-etc-hosts-with-host-aliases/]

## ConfigMaps

While most types of Kubernetes secrets store sensitive information, ConfigMaps [https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/] store general configuration information, such as a group of config files. Because ConfigMaps don't store sensitive information, they can be updated automatically, and therefore don't require their containers to be restarted following update (unlike most secret types, which require manual updates and a container restart to take effect).

ConfigMaps accept key value pairs in common string formats, like config files or JSON blobs. After you upload a config map, any workload can reference it as either an environment variable or a volume mount.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster that has the workload that should reference a ConfigMap and click **Explore**.
3. In the left navigation bar, click **More Resources** › **Core** › **ConfigMaps**.
4. Click **Create**.
5. Enter a **Name** for the Config Map.



Kubernetes classifies ConfigMaps as secrets [https://kubernetes.io/docs/concepts/configuration/secret/], and no two secrets in a project or namespace can have duplicate names. Therefore, to prevent conflicts, your ConfigMaps must have a unique name among the other certificates, registries, and secrets within your workspace.

6. Select the **Namespace** you want to add Config Map to.
7. On the **Data** tab, add a key-value pair to your ConfigMap. Add as many values as you need. You can add multiple key value pairs to the ConfigMap by copying and pasting. Alternatively, use **Read from File** to add the data. Note: If you need to store sensitive data, use a secret, not a ConfigMap.
8. Click **Create**.

**Result:** Your ConfigMap is added to the namespace. You can view it in the Rancher UI from the **Resources** › **Config Maps** view.

**What's Next?**

Now that you have a ConfigMap added to a namespace, you can add it to a workload that you deploy from the namespace of origin. You can use the ConfigMap to specify information for you application to consume, such as:

- Application environment variables.
- Specifying parameters for a Volume mounted to the workload.

For more information on adding ConfigMaps to a workload, see Deploying Workloads.

## Kubernetes Registry and Container Image Registry

Registries are Kubernetes secrets containing credentials used to authenticate with private container registries [https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/].

The word "registry" can mean two things, depending on whether it is used to refer to a container or Kubernetes registry:

- A **Container image registry** (formerly "Docker registry") contains container images that you can pull and deploy. The registry is a stateless, scalable server side application that stores and lets you distribute container images.
- The **Kubernetes registry** is an image pull secret that your deployment uses to authenticate with an image registry.

Deployments use the Kubernetes registry secret to authenticate with a private image registry and then pull a container image hosted on it.

Currently, deployments pull the private registry credentials automatically only if the workload is created in

the Rancher UI and not when it is created via kubectl.

**Creating a Registry in Namespaces**



*Prerequisite:*

You must have an available private registry already deployed.

If you need to create a private registry, refer to the documentation pages for your respective runtime:

- Containerd [https://github.com/containerd/containerd/blob/main/docs/cri/config.md#registry-configuration].
  - Nerdctl commands and managed registry services [https://github.com/containerd/nerdctl/blob/main/docs/registry.md].
- Docker [https://docs.docker.com/registry/deploying/].

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to add a registry and click **Explore**.
3. In the left navigation, click either **Storage** › **Secrets** or **More Resources** › **Core** › **Secrets**.
4. Click **Create**.
5. Click **Registry**.
6. Enter a **Name** for the registry.



Kubernetes classifies secrets, certificates, and registries all as secrets [https://kubernetes.io/docs/concepts/configuration/secret/], and no two secrets in a project or namespace can have duplicate names. Therefore, to prevent conflicts, your registry must have a unique name among all secrets within your workspace.

7. Select a namespace for the registry.
8. Select the website that hosts your private registry. Then enter credentials that authenticate with the registry. For example, if you use Docker Hub, provide your Docker Hub username and password.
9. Click **Save**.

**Result:**

- Your secret is added to the namespace you chose.
- You can view the secret in the Rancher UI by clicking either **Storage** › **Secrets** or **More Resources** › **Core** › **Secrets**.
- Any workload that you create in the Rancher UI will have the credentials to access the registry if the workload is within the registry's scope.

**Creating a Registry in Projects**



*Prerequisites:*

You must have an available private registry already deployed.

If you need to create a private registry, refer to the documentation pages for your respective runtime:

- Containerd [https://github.com/containerd/containerd/blob/main/docs/cri/config.md#registry-configuration].
  - Nerdctl commands and managed registry services [https://github.com/containerd/nerdctl/blob/main/docs/registry.md].
- Docker [https://docs.docker.com/registry/deploying/].

Before v2.6, secrets were required to be in a project scope. Projects are no longer required, and you may use the namespace scope instead. As a result, the Rancher UI was updated to reflect this new functionality. However, you may still create a project-scoped registry if desired. Use the following steps to do so:

1. In the upper left corner, click ☰ > **Global Settings** in the dropdown.
2. Click **Feature Flags**.
3. Go to the `legacy` feature flag and click **Activate**.
4. In the upper left corner, click ☰ > **Cluster Management**.
5. Go to the cluster where you want to add a registry and click **Explore**.
6. In the left navigation, click either **Storage** › **Secrets** or **More Resources** › **Core** › **Secrets**.
7. Click **Create**.
8. Click **Registry**.
9. In the top navigation bar, filter to see only one project.
10. Enter a **Name** for the registry.



Kubernetes classifies secrets, certificates, and registries all as secrets [https://kubernetes.io/docs/concepts/configuration/secret/], and no two secrets in a project or namespace can have duplicate names. Therefore, to prevent conflicts, your registry must have a unique name among all secrets within your workspace.

11. Select a namespace for the registry.
12. Select the website that hosts your private registry. Then enter credentials that authenticate with the registry. For example, if you use Docker Hub, provide your Docker Hub username and password.
13. Click **Save**.

**Result:**

- Your secret is added to the individual project you chose.
- You can view the secret in the Rancher UI by clicking either **Storage** › **Secrets** or **More Resources** › **Core** › **Secrets**.
- Any workload that you create in the Rancher UI will have the credentials to access the registry if the workload is within the registry's scope.



Project-scoped registries on the local cluster are only visible when a single project is selected.

### Using a Private Registry

You can deploy a workload with an image from a private registry through the Rancher UI, or with `kubectl`.

#### Using the Private Registry with the Rancher UI

To deploy a workload with an image from your private registry,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Go to the cluster where you want to deploy a workload and click **Explore**.
3. Click **Workload**.
4. Click **Create**.
5. Select the type of workload you want to create.
6. Enter a unique name for the workload and choose a namespace.
7. In the **Container Image** field, enter the URL of the path to the image in your private registry. For example, if your private registry is on Quay.io, you could use `quay.io/<Quay profile name>/<Image name>`.
8. Click **Create**.

**Result:** Your deployment should launch, authenticate using the private registry credentials you added in the Rancher UI, and pull the container image that you specified.

#### Using the Private Registry with kubectl

When you create the workload using `kubectl`, you need to configure the pod so that its YAML has the path to the image in the private registry. You also have to create and reference the registry secret because the pod only automatically gets access to the private registry credentials if it is created in the Rancher UI.

The secret has to be created in the same namespace where the workload gets deployed.

Below is an example `pod.yml` for a workload that uses an image from a private registry. In this example, the pod uses an image from Quay.io, and the .yml specifies the path to the image. The pod authenticates with the registry using credentials stored in a Kubernetes secret called `testquay`, which is specified in `spec.imagePullSecrets` in the `name` field:

```
apiVersion: vl
kind: Pod
metadata:
  name: private-reg
spec:
  containers:
  - name: private-reg-container
    image: quay.io/<Quay profile name>/<image name>
  imagePullSecrets:
  - name: testquay
```

In this example, the secret named `testquay` is in the default namespace.

You can use `kubectl` to create the secret with the private registry credentials. This command creates the secret named `testquay`:

```
kubectl create secret docker-registry testquay \
    --docker-server=quay.io \
    --docker-username=<Profile name> \
    --docker-password=<password>
```

To see how the secret is stored in Kubernetes, you can use this command:

```
kubectl get secret testquay --output="jsonpath={.data.
\.dockerconfigjson}" | base64 --decode
```

The result looks like this:

```
{"auths":{"quay.io":{"username":"<Profile name>","password":
"<password>","auth":"c29lbXlhbGo6dGVzdGFiYzEyMw=="}}}
```

After the workload is deployed, you can check if the image was pulled successfully:

```
kubectl get events
```

The result should look like this:

```
14s         Normal    Scheduled          Pod    Successfully assigned
default/private-reg2 to minikube
11s         Normal    Pulling            Pod    pulling image
"quay.io/<Profile name>/<image name>"
10s         Normal    Pulled             Pod    Successfully pulled
image "quay.io/<Profile name>/<image name>"
```

For more information, refer to the Kubernetes documentation on creating a pod that uses your secret. [https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/#create-a-pod-that-uses-your-secret]

# Upgrading and Rolling Back Kubernetes

Following an upgrade to the latest version of Rancher, downstream Kubernetes clusters can be upgraded to use the latest supported version of Kubernetes.

## Tested Kubernetes Versions

Before a new version of Rancher is released, it's tested with the latest minor versions of Kubernetes to ensure

compatibility. For details on which versions of Kubernetes were tested on each Rancher version, refer to the support maintenance terms. [https://rancher.com/support-maintenance-terms/all-supported-versions/rancher-v2.6.0/]

### Recommended Best Practice for Upgrades

When upgrading the Kubernetes version of a cluster, we recommend that you:

1. Take a snapshot.
2. Initiate a Kubernetes upgrade.
3. If the upgrade fails, revert the cluster to the pre-upgrade Kubernetes version. This is achieved by selecting the **Restore etcd and Kubernetes version** option. This will return your cluster to the pre-upgrade kubernetes version before restoring the etcd snapshot.

The restore operation will work on a cluster that is not in a healthy or active state.

### Upgrading the Kubernetes Version



*Prerequisites:*

- The options below are available for Rancher-launched Kubernetes clusters and Registered K3s Kubernetes clusters.
- The following options also apply to imported RKE2 clusters that you have registered. If you import a cluster from an external cloud platform but don't register it, you won't be able to upgrade the Kubernetes version from Rancher.
- Before upgrading Kubernetes, back up your cluster.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to upgrade and click ☰ **> Edit Config**.
3. From the **Kubernetes Version** drop-down, choose the version of Kubernetes that you want to use for the cluster.
4. Click **Save**.

**Result:** Kubernetes begins upgrading for the cluster.

### Rolling Back

A cluster can be restored to a backup in which the previous Kubernetes version was used. For more information, refer to the following sections:

- Backing up a cluster
- Restoring a cluster from backup

### Configuring the Upgrade Strategy

The upgrade strategy can be configured in the Rancher UI, or by editing the `cluster.yml`. More advanced options are available by editing the `cluster.yml`.

#### Configuring the Maximum Unavailable Worker Nodes in the Rancher UI

From the Rancher UI, the maximum number of unavailable worker nodes can be configured. During a cluster upgrade, worker nodes will be upgraded in batches of this size.

By default, the maximum number of unavailable worker is defined as 10 percent of all worker nodes. This number can be configured as a percentage or as an integer. When defined as a percentage, the batch size is rounded down to the nearest node, with a minimum of one node.

To change the default number or percentage of worker nodes,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to upgrade and click ☰ **> Edit Config**.
3. In the **Upgrade Strategy** tab, enter the **Worker Concurrency** as a fixed number or percentage. To get this number, you can take the number of nodes in your cluster and subtract the max unavailable nodes.
4. Click **Save**.

**Result:** The cluster is updated to use the new upgrade strategy.

#### Enabling Draining Nodes During Upgrades from the Rancher UI

To enable draining each node during a cluster upgrade,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to enable node draining and click ☰ **> Edit Config**.
3. Click ☰ **> Edit**.
4. In the **Upgrade Strategy** tab, go to the **Drain nodes** field and click **Yes**. Node draining is configured separately for control plane and worker nodes.
5. Configure the options for how pods are deleted. For more information about each option, refer to this section.
6. Optionally, configure a grace period. The grace period is the timeout given to each pod for cleaning things up, so they will have chance to exit gracefully. Pods might need to finish any outstanding requests, roll back transactions or save state to some external storage. If this value is negative, the default value specified in the pod will be used.
7. Optionally, configure a timeout, which is the amount of time the drain should continue to wait before giving up.
8. Click **Save**.

**Result:** The cluster is updated to use the new upgrade strategy.



- There is a known issue [https://github.com/rancher/rancher/issues/25478] in which the Rancher UI doesn't show the state of etcd and controlplane as drained, even though they are being drained.
- During an upgrade, nodes may be drained even when no user-visible YAML changes are present. This can occur if non-dynamic configuration files are updated or if a new `system-agent-installer` image is introduced. In such cases, Rancher generates a new upgrade plan, resulting in a new plan hash. When `Upgrade Strategy` is set to `Drain nodes`, this plan change can trigger node draining.

#### Configuring the Upgrade Strategy in the cluster.yml

More advanced upgrade strategy configuration options are available by editing the `cluster.yml`.

## Troubleshooting

No upgrade will proceed if the number of unavailable nodes exceeds the configured maximum.

If an upgrade stops, you may need to fix an unavailable node or remove it from the cluster before the upgrade can continue.

A failed node could be in various states:

- Powered off
- Unavailable
- User drains a node while upgrade is in process, so there are no kubelets on the node
- The upgrade itself failed

If the max unavailable number of nodes is reached during an upgrade, Rancher user clusters will be stuck in updating state and not move forward with upgrading any other control plane nodes. It will continue to evaluate the set of unavailable nodes in case one of the nodes becomes available. If the node cannot be fixed, you must remove the node in order to continue the upgrade.

## Upgrading Kubernetes without Upgrading SUSE Rancher Prime

The RKE metadata feature allows you to provision clusters with new versions of Kubernetes as soon as they are released, without upgrading Rancher. This feature is useful for taking advantage of patch versions of Kubernetes, for example, if you want to upgrade to Kubernetes v1.14.7 when your Rancher server originally supported v1.14.6.



The Kubernetes API can change between minor versions. Therefore, we don't support introducing minor Kubernetes versions, such as introducing v1.15 when Rancher currently supports v1.14. You would need to upgrade Rancher to add support for minor Kubernetes versions.

Rancher's Kubernetes metadata contains information specific to the Kubernetes version that Rancher uses to provision RKE clusters. Rancher syncs the data periodically and creates custom resource definitions (CRDs) for **system images, service options** and **addon templates**. Consequently, when a new Kubernetes version is compatible with the Rancher server version, the Kubernetes metadata makes the new version available to Rancher for provisioning clusters. The metadata gives you an overview of the information that the Rancher Kubernetes Engine [https://rancher.com/docs/rke/latest/en/] (RKE) uses for deploying various Kubernetes versions.

This table below describes the CRDs that are affected by the periodic data sync.



Only administrators can edit metadata CRDs. It is recommended not to update existing objects unless explicitly advised.

| Resource | Description | Rancher API URL |
|---|---|---|
| System Images | List of system images used to deploy Kubernetes through RKE. | `<RANCHER_SERVER_URL>/v3/rkek8ssystemimages` |
| Service Options | Default options passed to Kubernetes components like `kube-api`, `scheduler`, `kubelet`, `kube-proxy`, and `kube-controller-manager` | `<RANCHER_SERVER_URL>/v3/rkek8sserviceoptions` |
| Addon Templates | YAML definitions used to deploy addon components like Canal, Calico, Flannel, Weave, Kube-dns, CoreDNS, `metrics-server`, `nginx-ingress` | `<RANCHER_SERVER_URL>/v3/rkeaddons` |

Administrators might configure the RKE metadata settings to do the following:

- Refresh the Kubernetes metadata, if a new patch version of Kubernetes comes out and they want Rancher to provision clusters with the latest version of Kubernetes without having to upgrade Rancher
- Change the metadata URL that Rancher uses to sync the metadata, which is useful for air gap setups if you need to sync Rancher locally instead of with GitHub
- Prevent Rancher from auto-syncing the metadata, which is one way to prevent new and unsupported Kubernetes versions from being available in Rancher

#### Refresh Kubernetes Metadata

The option to refresh the Kubernetes metadata is available for administrators by default, or for any user who has the **Manage Cluster Drivers** global role.

To force Rancher to refresh the Kubernetes metadata, a manual refresh action is available:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the left navigation menu, click **Drivers**.
3. Click **Refresh Kubernetes Metadata**.

You can configure Rancher to only refresh metadata when desired by setting `refresh-interval-minutes` to `0` (see below) and using this button to perform the metadata refresh manually when desired.

##### Configuring the Metadata Synchronization



Only administrators can change these settings.

The RKE metadata config controls how often Rancher syncs metadata and where it downloads data from. You can configure the metadata from the settings in the Rancher UI, or through the Rancher API at the endpoint `v3/settings/rke-metadata-config`.

The way that the metadata is configured depends on the Rancher version.

To edit the metadata config in Rancher,

1. In the upper left corner, click ☰ > **Global Settings**.
2. Go to the **rke-metadata-config** section. Click ☰ > **Edit Setting**.
3. You can optionally fill in the following parameters:

- `refresh-interval-minutes`: This is the amount of time that Rancher waits to sync the metadata. To disable the periodic refresh, set `refresh-interval-minutes` to 0.
- `url`: This is the HTTP path that Rancher fetches data from. The path must be a direct path to a JSON file. For example, the default URL for Rancher v2.4 is `https://releases.rancher.com/kontainer-driver-metadata/release-v2.4/data.json`.

    a. Click **Save**.

If you don't have an air gap setup, you don't need to specify the URL where Rancher gets the metadata, because the default setting is to pull from Rancher's metadata Git repository. [https://github.com/rancher/kontainer-driver-metadata/blob/dev-v2.5/data/data.json]

However, if you have an air gap setup, you will need to mirror the Kubernetes metadata repository in a location available to Rancher. Then you need to change the URL to point to the new location of the JSON file.

#### Air Gap Setups

Rancher relies on a periodic refresh of the `rke-metadata-config` to download new Kubernetes version metadata if it is supported with the current version of the Rancher server. For a table of compatible Kubernetes and Rancher versions, refer to the service terms section. [https://rancher.com/support-maintenance-terms/all-supported-versions/rancher-v2.2.8/]

If you have an air gap setup, you might not be able to get the automatic periodic refresh of the Kubernetes metadata from Rancher's Git repository. In that case, you should disable the periodic refresh to prevent your logs from showing errors. Optionally, you can configure your metadata settings so that Rancher can sync with a local copy of the RKE metadata.

To sync Rancher with a local mirror of the RKE metadata, an administrator would configure the `rke-metadata-config` settings to point to the mirror. For details, refer to Configuring the Metadata Synchronization.

After new Kubernetes versions are loaded into the Rancher setup, additional steps would be required in order to use them for launching clusters. Rancher needs access to updated system images. While the metadata settings can only be changed by administrators, any user can download the Rancher system images and prepare a private container image registry for them.

To download the system images for the private registry:

1. Click ☰ in the top left corner.
2. Click **About** at the bottom of the left navigation.
3. Download the OS specific image lists for Linux or Windows.
4. Download `rancher-images.txt`.
5. Prepare the private registry using the same steps during the air gap install, but instead of using the `rancher-images.txt` from the releases page, use the one obtained from the previous steps.

**Result:** The air gap installation of ncher can now sync the Kubernetes metadata. If you update your private registry when new versions of Kubernetes are released, you can provision clusters with the new version without having to upgrade Rancher.

## Backing up a Cluster

In the Rancher UI, etcd backup and recovery for Rancher launched Kubernetes clusters can be easily performed.

Rancher recommends configuring recurrent `etcd` snapshots for all production clusters. Additionally, one-time snapshots can be taken as well.

Snapshots of the etcd database are taken and saved either locally onto the etcd nodes or to a S3 compatible target. The advantages of configuring S3 is that if all etcd nodes are lost, your snapshot is saved remotely and can be used to restore the cluster.

### How Snapshots Work

#### Snapshot Components

**RKE**

When Rancher creates a snapshot, it includes three components:

- The cluster data in etcd
- The Kubernetes version
- The cluster configuration in the form of the `cluster.yml`

Because the Kubernetes version is now included in the snapshot, it is possible to restore a cluster to a prior Kubernetes version.

**RKE2/K3s**

Rancher delegates snapshot creation to the downstream Kubernetes engine. When the Kubernetes engine creates a snapshot, it includes three components:

- The cluster data in etcd
- The Kubernetes version
- The cluster configuration

Because the Kubernetes version is included in the snapshot, it is possible to restore a cluster to a prior Kubernetes version while also restoring an etcd snapshot.

The multiple components of the snapshot allow you to select from the following options if you need to restore a cluster from a snapshot:

- **Restore just the etcd contents:** This restore is similar to restoring to snapshots in Rancher before v2.4.0.
- **Restore etcd and Kubernetes version:** This option should be used if a Kubernetes upgrade is the reason that your cluster is failing, and you haven't made any cluster configuration changes.
- **Restore etcd, Kubernetes versions and cluster configuration:** This option should be used if you changed both the Kubernetes version and cluster configuration when upgrading.

It is always recommended to take a new snapshot before performing any configuration changes or upgrades.

**RKE**

For each etcd node in the cluster, the etcd cluster health is checked. If the node reports that the etcd cluster is healthy, a snapshot is created from it and optionally uploaded to S3.

The snapshot is stored in `/opt/rke/etcd-snapshots`. If the directory is configured on the nodes as a shared mount, it will be overwritten. On S3, the snapshot will always be from the last node that uploads it, as all etcd nodes upload it and the last will remain.

In the case when multiple etcd nodes exist, any created snapshot is created after the cluster has been health checked, so it can be considered a valid snapshot of the data in the etcd cluster.

**RKE2/K3s**

Snapshots are enabled by default.

The snapshot directory defaults to `/var/lib/rancher/<RUNTIME>/server/db/snapshots`, where `<RUNTIME>` is either `rke2` or `k3s`.

In RKE2, snapshots are stored on each etcd node. If you have multiple etcd or etcd + control-plane nodes, you will have multiple copies of local etcd snapshots.

# Snapshot Naming Conventions

**RKE**

The name of the snapshot is auto-generated. The `--name` option can be used to override the name of the snapshot when creating one-time snapshots with the RKE CLI.

When Rancher creates a snapshot of an RKE cluster, the snapshot name is based on the type (whether the snapshot is manual or recurring) and the target (whether the snapshot is saved locally or uploaded to S3). The naming convention is as follows:

- `m` stands for manual
- `r` stands for recurring
- `l` stands for local
- `s` stands for S3

Some example snapshot names are:

- c-9dmxz-rl-8b2cx
- c-9dmxz-ml-kr56m
- c-9dmxz-ms-t6bjb
- c-9dmxz-rs-8gxc8

**RKE2/K3s**

The name of the snapshot is auto-generated. The `--name` option can be used to override the base name of the snapshot when creating one-time snapshots with the RKE2 or K3s CLI.

When Rancher creates a snapshot of an RKE2 or K3s cluster, the snapshot name is based on the type (whether the snapshot is manual or recurring) and the target (whether the snapshot is saved locally or uploaded to S3). The naming convention is as follows:

`<name>-<node>-<timestamp>`

`<name>`: is the base name set by `--name` and can be one of the following

- `etcd-snapshot` is prepended on recurring snapshots
- `on-demand` is prepended on manual, on-demand snapshots

`<node>`: Node is the name of the node that the snapshot was taken on.

`<timestamp>` is a unix-time stamp of the snapshot creation date.

Some example snapshot names are:

- `on-demand-my-super-rancher-k8s-node1-1652288934`
- `on-demand-my-super-rancher-k8s-node2-1652288936`
- `etcd-snapshot-my-super-rancher-k8s-node1-1652289945`
- `etcd-snapshot-my-super-rancher-k8s-node2-1652289948`

## How Restoring from a Snapshot Works

**RKE**

On restore, the following process is used:

1. The snapshot is retrieved from S3, if S3 is configured.
2. The snapshot is unzipped (if zipped).
3. One of the etcd nodes in the cluster serves that snapshot file to the other nodes.
4. The other etcd nodes download the snapshot and validate the checksum so that they all use the same snapshot for the restore.
5. The cluster is restored and post-restore actions will be done in the cluster.

**RKE2/K3s**

On restore, Rancher delivers a few sets of plans to perform a restoration. A set of phases are used, namely:

- Started
- Shutdown
- Restore
- RestartCluster
- Finished

If the etcd snapshot restore fails, the phase will be set to `Failed`.

1. The etcd snapshot restore request is received, and depending on `restoreRKEConfig`, the cluster configuration/kubernetes version are reconciled.
2. The phase is set to `Started`.
3. The phase is set to `Shutdown`, and the entire cluster is shut down using plans that run the distribution `killall.sh` script. A new init node is elected. If the snapshot being restored is a local snapshot, the node that the snapshot resides on will be selected as the init node. If the snapshot is being restored from S3, the existing init node will be used.
4. The phase is set to `Restore`, and the init node has the snapshot restored onto it.
5. The phase is set to `RestartCluster`, and the cluster is restarted/rejoined to the new init node that has the freshly restored snapshot information.
6. The phase is set to `Finished`, and the cluster is deemed successfully restored. The `cattle-cluster-agent` will reconnect, and the cluster will finish reconciliation.

# Configuring Recurring Snapshots

**RKE**

Select how often you want recurring snapshots to be taken as well as how many snapshots to keep. The amount of time is measured in hours. With timestamped snapshots, the user has the ability to do a point-in-time recovery.

By default, Rancher launched Kubernetes clusters are configured to take recurring snapshots (saved to local disk). To protect against local disk failure, using the S3 Target or replicating the path on disk is advised.

During cluster provisioning or editing the cluster, the configuration for snapshots can be found in the advanced section for **Cluster Options**. Click on **Show advanced options**.

In the **Advanced Cluster Options** section, there are several options available to configure:

| Option | Description | Default Value |
|---|---|---|
| etcd Snapshot Backup Target | Select where you want the snapshots to be saved. Options are either local or in S3 | local |
| Recurring etcd Snapshot Enabled | Enable/Disable recurring snapshots | Yes |
| Recurring etcd Snapshot Creation Period | Time in hours between recurring snapshots | 12 hours |
| Recurring etcd Snapshot Retention Count | Number of snapshots to retain | 6 |

**RKE2/K3s**

Set the schedule for how you want recurring snapshots to be taken as well as how many snapshots to keep. The schedule is conventional cron format. The retention policy dictates the number of snapshots matching a name to keep per node.

By default, Rancher launched Kubernetes clusters are configured to take recurring snapshots (saved to local disk) every 5 hours starting at 12 AM. To protect against local disk failure, using the S3 Target or replicating the path on disk is advised.

During cluster provisioning or editing the cluster, the configuration for snapshots can be found under **Cluster Configuration**. Click on **etcd**.

| Option | Description | Default Value |
|---|---|---|
| Recurring etcd Snapshot Enabled | Enable/Disable recurring snapshots | Yes |
| Recurring etcd Snapshot Creation Period | Cron schedule for recurring snapshot | `0 */5 * * *` |
| Recurring etcd Snapshot Retention Count | Number of snapshots to retain | 5 |

# One-Time Snapshots

**RKE**

In addition to recurring snapshots, you may want to take a "one-time" snapshot. For example, before upgrading the Kubernetes version of a cluster it's best to backup the state of the cluster to protect against upgrade failure.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, navigate to the cluster where you want to take a one-time snapshot.
3. Click ☰ **> Take Snapshot**.

**RKE2/K3s**

In addition to recurring snapshots, you may want to take a "one-time" snapshot. For example, before upgrading the Kubernetes version of a cluster it's best to backup the state of the cluster to protect against upgrade failure.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, navigate to the cluster where you want to take a one-time snapshot.
3. Navigate to the `Snapshots` tab and click `Snapshot Now`

```
<h3><a id="_how_taking_one_time_snapshots_works"></a>How Taking One-Time Snapshots Works</h3>
```

On one-time snapshot creation, the Rancher delivers a few sets of plans to perform snapshot creation. A set of phases are used, namely:

- Started
- RestartCluster
- Finished

If the etcd snapshot creation fails, the phase will be set to `Failed`.

1. The etcd snapshot creation request is received.
2. The phase is set to `Started`. All etcd nodes in the cluster receive a plan to create an etcd snapshot, per the cluster configuration.
3. The phase is set to `RestartCluster`, and the plans on every etcd node are reset to the original

plan for the etcd nodes.

4. The phase is set to `Finished`.

**Result:** Based on your snapshot backup target, a one-time snapshot will be taken and saved in the selected backup target.

### Snapshot Backup Targets

Rancher supports two different backup targets:

- Local Target
- S3 Target

#### Local Backup Target

**RKE**

By default, the `local` backup target is selected. The benefits of this option is that there is no external configuration. Snapshots are automatically saved locally to the etcd nodes in the Rancher launched Kubernetes clusters in `/opt/rke/etcd-snapshots`. All recurring snapshots are taken at configured intervals. The downside of using the `local` backup target is that if there is a total disaster and *all* etcd nodes are lost, there is no ability to restore the cluster.

**RKE2/K3s**

By default, the `local` backup target is selected. The benefits of this option is that there is no external configuration. Snapshots are automatically saved locally to the etcd nodes in the Rancher launched Kubernetes clusters in `/var/lib/rancher/<runtime>/server/db/snapshots` where `<runtime>` is either `k3s` or `rke2`. All recurring snapshots are taken per the cron schedule. The downside of using the `local` backup target is that if there is a total disaster and *all* etcd nodes are lost, there is no ability to restore the cluster.

#### S3 Backup Target

We recommend that you use the S3 backup target. It lets you store snapshots externally, on an S3 compatible backend. Since the snapshots aren't stored locally, you can still restore the cluster even if you lose all etcd nodes.

Although the S3 target offers advantages over local backup, it does require extra configuration.



If you use an S3 backup target, make sure that every cluster has its own bucket or folder. Rancher populates snapshot information from any available snapshot listed in the S3 bucket or folder configured for that cluster.

| Option | Description | Required |
|---|---|---|
| S3 Bucket Name | Name of S3 bucket to store backups | * |
| S3 Region | S3 region for the backup bucket | |
| S3 Region Endpoint | S3 regions endpoint for the backup bucket | * |

| Option | Description | Required |
|---|---|---|
| S3 Access Key | S3 access key with permission to access the backup bucket | * |
| S3 Secret Key | S3 secret key with permission to access the backup bucket | * |
| Custom CA Certificate | A custom certificate used to access private S3 backends | |

#### Using a custom CA certificate for S3

The backup snapshot can be stored on a custom S3 backup like minio [https://min.io/]. If the S3 back end uses a self-signed or custom certificate, provide a custom certificate using the `Custom CA Certificate` option to connect to the S3 backend.

#### IAM Support for Storing Snapshots in S3

The S3 backup target supports using IAM authentication to AWS API in addition to using API credentials. An IAM role gives temporary permissions that an application can use when making API calls to S3 storage. To use IAM authentication, the following requirements must be met:

- The cluster etcd nodes must have an instance role that has read/write access to the designated backup bucket.
- The cluster etcd nodes must have network access to the specified S3 endpoint.
- The Rancher Server worker node(s) must have an instance role that has read/write to the designated backup bucket.
- The Rancher Server worker node(s) must have network access to the specified S3 endpoint.

To give an application access to S3, refer to the AWS documentation on Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances. [https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html]

### Viewing Available Snapshots

The list of all available snapshots for the cluster is available in the Rancher UI.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. In the **Clusters** page, go to the cluster where you want to view the snapshots and click its name.
3. Click the **Snapshots** tab to view the list of saved snapshots. These snapshots include a timestamp of when they were created.

### Safe Timestamps (RKE)

Snapshot files are timestamped to simplify processing the files using external tools and scripts, but in some S3 compatible backends, these timestamps were unusable.

The option `safe_timestamp` is added to support compatible file names. When this flag is set to `true`, all special characters in the snapshot filename timestamp are replaced.

This option is not available directly in the UI, and is only available through the `Edit as Yaml` interface.

## Restoring a Cluster from Backup

Etcd backup and recovery for Rancher launched Kubernetes clusters can be easily performed. Snapshots of the etcd database are taken and saved either locally onto the etcd nodes or to a S3 compatible target. The advantages of configuring S3 is that if all etcd nodes are lost, your snapshot is saved remotely and can be used to restore the cluster.

Rancher recommends enabling the ability to set up recurring snapshots of etcd, but one-time snapshots can easily be taken as well. Rancher allows restore from saved snapshots.

Clusters can also be restored to a prior Kubernetes version and cluster configuration.

### Viewing Available Snapshots

The list of all available snapshots for the cluster is available.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the **Clusters** page, go to the cluster where you want to view the snapshots and click the name of the cluster.
3. Click the **Snapshots** tab. The listed snapshots include a timestamp of when they were created.

### Restoring a Cluster from a Snapshot

If your Kubernetes cluster is broken, you can restore the cluster from a snapshot.

Snapshots are composed of the cluster data in etcd, the Kubernetes version, and the cluster configuration in the `cluster.yml`. These components allow you to select from the following options when restoring a cluster from a snapshot:

- **Restore just the etcd contents:** This restore is similar to restoring to snapshots in Rancher before v2.4.0.
- **Restore etcd and Kubernetes version:** This option should be used if a Kubernetes upgrade is the reason that your cluster is failing, and you haven't made any cluster configuration changes.
- **Restore etcd, Kubernetes versions and cluster configuration:** This option should be used if you changed both the Kubernetes version and cluster configuration when upgrading.

When rolling back to a prior Kubernetes version, the upgrade strategy options are ignored. Worker nodes are not cordoned or drained before being reverted to the older Kubernetes version, so that an unhealthy cluster can be more quickly restored to a healthy state.



*Prerequisite:*

To restore snapshots from S3, the cluster needs to be configured to take recurring snapshots on S3.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. In the **Clusters** page, go to the cluster where you want to view the snapshots and click the name of the cluster.
3. Click the **Snapshots** tab to view the list of saved snapshots.
4. Go to the snapshot you want to restore and click ☰ > **Restore**.
5. Select a **Restore Type**.

6. Click **Restore**.

**Result:** The cluster will go into `updating` state and the process of restoring the `etcd` nodes from the snapshot will start. The cluster is restored when it returns to an `active` state.

#### Restoring a Cluster From a Snapshot When the controlplane/etcd Are Completely Unavailable

In a disaster recovery scenario, the control plane and etcd nodes managed by Rancher in a downstream cluster may no longer be available or functioning. The cluster can be rebuilt by adding control plane and etcd nodes again, followed by restoring from an available snapshot.

If you have a complete cluster failure, you must remove all etcd nodes/machines from your cluster before you can add a "new" etcd node for restore.



If you are using local snapshots, it is **VERY** important that you ensure you back up the corresponding snapshot you want to restore from the `/var/lib/rancher/<k3s/rke2>/server/db/snapshots/` folder on the etcd node you are going to be removing. You can copy the snapshot onto your new node in the `/var/lib/rancher/<k3s/rke2>/server/db/snapshots/` folder. Furthermore, if using local snapshots and restoring to a new node, restoration cannot be done via the UI as of now.

1. Remove all etcd and control plane nodes from your cluster.
   a. In the upper left corner, click ☰ **> Cluster Management**.
   b. In the **Clusters** page, go to the cluster where you want to remove nodes.
   c. In the **Machines** tab, click ☰ **> Delete** on each node you want to delete. Initially, you will see the nodes hang in a `deleting` state, but once all etcd nodes are deleting, they will be removed together. This is due to the fact that Rancher sees all etcd nodes deleting and proceeds to "short circuit" the etcd safe-removal logic.
2. After all etcd and control plane nodes are removed, add the new etcd / control plane node that you are planning to restore from. Assign the new node the role etcd and control plane (etcd, controlplane).
   - If the node was previously in a cluster, clean the node first.
   - For custom clusters, go to the **Registration** tab and check the box for `etcd, controlplane, and worker`. Then copy and run the registration command on your node.
   - For node driver clusters, a new node is provisioned automatically.

     At this point, Rancher will indicate that restoration from etcd snapshot is required.
3. Restore from an etcd snapshot.

   As the etcd node is a clean node, you may need to manually create the `/var/lib/rancher/<k3s/rke2>/server/db/snapshots/` path.

   - For S3 snapshots, restore using the UI.
     i. Click the **Snapshots** tab to view the list of saved snapshots.
     ii. Go to the snapshot you want to restore and click ☰ **> Restore**.
     iii. Select a **Restore Type**.

iv. Click **Restore**.

- For local snapshots, restore using the UI is **not** available.

  i. In the upper right corner, click ⋮ **> Edit YAML**.

  ii. The example YAML below can be added under your `rkeConfig` to configure the etcd restore:

```
...
rkeConfig:
  etcdSnapshotRestore:
    name: <string> # This field is required. Refers to the
filename of the associated etcdsnapshot object.
...
```

4. After restoration is successful, you can scale your etcd nodes back up to the desired redundancy.

# Namespaces

Within Rancher, you can further divide projects into different namespaces [https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/], which are virtual clusters within a project backed by a physical cluster. Should you require another level of organization beyond projects and the `default` namespace, you can use multiple namespaces to isolate applications and resources.

Although you assign resources at the project level so that each namespace in the project can use them, you can override this inheritance by assigning resources explicitly to a namespace.

Resources that you can assign directly to namespaces include:

- Workloads
- Load Balancers/Ingress
- Service Discovery Records
- Persistent Volume Claims
- Certificates
- ConfigMaps
- Registries
- Secrets

To manage permissions in a vanilla Kubernetes cluster, cluster admins configure role-based access policies for each namespace. With Rancher, user permissions are assigned on the project level instead, and permissions are automatically inherited by any namespace owned by the particular project.



If you create a namespace with `kubectl`, it may be unusable because `kubectl` doesn't require your new namespace to be scoped within a project that you have access to. If your permissions are restricted to the project level, it is better to create a namespace through Rancher to ensure that you will have permission to access the namespace.

## Creating Namespaces

Create a new namespace to isolate apps and resources in a project.



When working with project resources that you can assign to a namespace (i.e., workloads, certificates, ConfigMaps, etc.) you can create a namespace on the fly.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to create a namespace and click **Explore**.
3. Click menu:Cluster[Projects/Namespaces].
4. Go to the project where you want to add a namespace and click **Create Namespace**. Alternately, go to **Not in a Project** to create a namespace not associated with a project.
5. **Optional:** If your project has Resource Quotas in effect, you can override the default resource **Limits** (which places a cap on the resources that the namespace can consume).
6. Enter a **Name** and then click **Create**.

**Result:** Your namespace is added to the project. You can begin assigning cluster resources to the namespace.

## Moving Namespaces to Another Project

Cluster admins and members may occasionally need to move a namespace to another project, such as when you want a different team to start using the application.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to move a namespace and click **Explore**.
3. Click menu:Cluster[Projects/Namespaces].
4. Go to the namespace you want to move and click ☰ > **Move**.
5. Select the namespace(s) that you want to move to a different project. Then click **Move**. You can move multiple namespaces at one.



- Don't move the namespaces in the `System` project. Moving these namespaces can adversely affect cluster networking.
- You cannot move a namespace into a project that already has a resource quota configured.
- If you move a namespace from a project that has a quota set to a project with no quota set, the quota is removed from the namespace.

6. Choose a new project for the new namespace and then click **Move**. Alternatively, you can remove the namespace from all projects by selecting **None**.

**Result:** Your namespace is moved to a different project (or is unattached from all projects). If any project resources are attached to the namespace, the namespace releases them and then attached resources from the new project.

## Editing Namespace Resource Quotas

You can always override the namespace default limit to provide a specific namespace with access to more

(or less) project resources.

For more information, see how to edit namespace resource quotas.

# Project Administration

*Projects* are objects introduced in Rancher that help organize namespaces in your Kubernetes cluster. You can use projects to create multi-tenant clusters, which allows a group of users to share the same underlying resources without interacting with each other's applications.

In terms of hierarchy:

- Clusters contain projects
- Projects contain namespaces

Within Rancher, projects allow you to manage multiple namespaces as a single entity. In native Kubernetes, which does not include projects, features like role-based access rights or cluster resources are assigned to individual namespaces. In clusters where multiple namespaces require the same set of access rights, assigning these rights to each individual namespace can become tedious. Even though all namespaces require the same rights, there's no way to apply those rights to all of your namespaces in a single action. You'd have to repetitively assign these rights to each namespace!

Rancher projects resolve this issue by allowing you to apply resources and access rights at the project level. Each namespace in the project then inherits these resources and policies, so you only have to assign them to the project once, rather than assigning them to each individual namespace.

You can use projects to perform actions like:

- cluster-admin/project-admin/add-users-to-projects.pdf[Assign users access to a group of namespaces]
- Assign users specific roles in a project. A role can be owner, member, read-only, or custom
- Set resource quotas
- Manage namespaces
- Configure tools

## Authorization

Non-administrative users are only authorized for project access after an administrator, cluster owner or member, or project owner adds them to the project's **Members** tab.

Whoever creates the project automatically becomes a project owner.

## Switching between Projects

To switch between projects, use the drop-down available in the navigation bar. Alternatively, you can switch between projects directly in the navigation bar.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to switch projects and click **Explore**.
3. In the top navigation bar, select the project that you want to open.

## Adding Users to Projects

If you want to provide a user with access and permissions to *specific* projects and resources within a cluster, assign the user a project membership.

You can add members to a project as it is created, or add them to an existing project.



Want to provide a user with access to *all* projects within a cluster? See Adding Cluster Members instead.

### Adding Members to a New Project

You can add members to a project as you create it (recommended if possible). For details on creating a new project, refer to the cluster administration section.

### Adding Members to an Existing Project

Following project creation, you can add users as project members so that they can access its resources.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to add members to a project and click **Explore**.
3. Click **Cluster › Projects/Namespaces**.
4. Go to the project where you want to add members. Next to the **Create Namespace** button above the project name, click ⋮. Select **Edit Config**.
5. In the **Members** tab, click **Add**.
6. Search for the user or group that you want to add to the project.



- At least 2 characters must be typed in the search box for results to appear.
- Users can be searched based on their username or display name.
- Search is prefix-based (e.g., a user named `Stan Dard` will appear when searching for `Sta`, but not when searching for `Dar`) and case-sensitive.

If external authentication is configured:

- Rancher returns users from your external authentication source as you type.
- A drop-down allows you to add groups instead of individual users. The dropdown only lists groups that you, the logged in user, are included in.



If you are logged in as a local user, external users do not display in your search results.

7. Assign the user or group **Project** roles.

What are Project Roles?



- Users assigned the `Owner` or `Member` role for a project automatically inherit the `namespace creation` role. However, this role is a Kubernetes ClusterRole [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-and-clusterrole], meaning its scope extends to all projects in the cluster. Therefore, users explicitly assigned the `Owner` or `Member` role for a project can create or delete namespaces

in other projects they're assigned to, even with only the `Read Only` role assigned.

- By default, the Rancher role of `project-member` inherits from the `Kubernetes-edit` role, and the `project-owner` role inherits from the `Kubernetes-admin` role. As such, both `project-member` and `project-owner` roles will allow for namespace management, including the ability to create and delete namespaces.
- For `Custom` roles, you can modify the list of individual roles available for assignment.
  - To add roles to the list, Add a Custom Role.
  - To remove roles from the list, Lock/Unlock Roles.

**Result:** The chosen users are added to the project.

- To revoke project membership, select the user and click **Delete**. This action deletes membership, not the user.
- To modify a user's roles in the project, delete them from the project, and then re-add them with modified roles.

## Project Resource Quotas

In situations where several teams share a cluster, one team may overconsume the resources available: CPU, memory, storage, services, Kubernetes objects like pods or secrets, and so on. To prevent this overconsumption, you can apply a *resource quota*, which is a Rancher feature that limits the resources available to a project or namespace.

This page is a how-to guide for creating resource quotas in existing projects.

Resource quotas can also be set when a new project is created. For details, refer to the section on creating new projects.

Resource quotas in Rancher include the same functionality as the native version of Kubernetes [https://kubernetes.io/docs/concepts/policy/resource-quotas/]. In Rancher, resource quotas have been extended so that you can apply them to projects. For details on how resource quotas work with projects in Rancher, refer to this page.

### Applying Resource Quotas to Existing Projects

Edit resource quotas when:

- You want to limit the resources that a project and its namespaces can use.
- You want to scale the resources available to a project up or down when a resource quota is already in effect.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to apply a resource quota and click **Explore**.
3. Click **Cluster › Projects/Namespaces**.
4. Make sure that the **Projects/Namespaces** page is in **Group by Project** view mode. image::edit-project-config-for-resource-quotas-group-by-project.png[Screenshot highlighting the "Group by Project" icon, above the list of projects. It resembles a folder.]

5. Find the project that you want to add a resource quota to, and select the ⋮ that's on the same row as the project's name. image::edit-project-config-for-resource-quotas-dots.png[Screenshot highlighting triple dots icon at the end of the same row as the project name.]
6. Select **Edit Config**.
7. Expand **Resource Quotas** and click **Add Resource**. Alternatively, you can edit existing quotas.
8. Select a Resource Type. For more information on types, see the quota type reference.
9. Enter values for the **Project Limit** and the **Namespace Default Limit**.

| Field | Description |
|---|---|
| Project Limit | The overall resource limit for the project. |
| Namespace Default Limit | The default resource limit available for each namespace. This limit is propagated to each namespace in the project. The combined limit of all project namespaces shouldn't exceed the project limit. |

10. **Optional:** Add more quotas.
11. Click **Create**.

**Result:** The resource quota is applied to your project and namespaces. When you add more namespaces in the future, Rancher validates that the project can accommodate the namespace. If the project can't allocate the resources, you may still create namespaces, but they will be given a resource quota of 0. Subsequently, Rancher will not allow you to create any resources restricted by this quota.

#### Advanced: Beyond the basic Resource Quotas

The set of resource quotas listed in the **Resource Type** dropdown of **Edit Config** is limited. For quotas outside of that set use **Edit Config** and **Add Resource** as already described, and select **Custom** as the resource type. This enables the edit field **Resource Identifier** for the entry of the necessary identifier. Some examples of identifiers are:

- `requests.nvidia.com/gpu`
- `gold.storageclass.storage.k8s.io/requests.storage`
- `count/podtemplates`



While it is possible to specify `Custom` which refer to quotas in the basic builtin set it is currently **strongly** recommended to use the builtin fields for them instead. Also, in case of conflicts, i.e. specifying a quota for a resource in both its builtin field and via `Custom`, the data found in the builtin field has priority and the data in `Custom` is ignored.

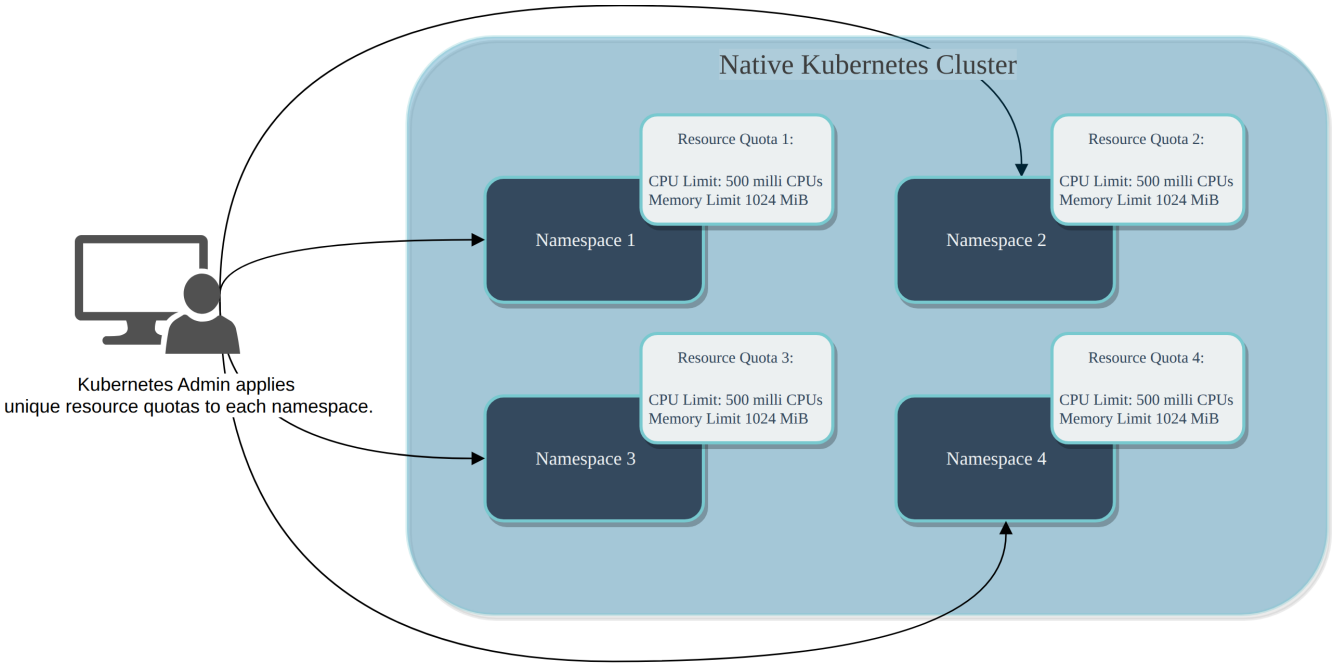
#### How Resource Quotas Work in SUSE Rancher Prime Projects

Resource quotas in Rancher include the same functionality as the native version of Kubernetes [https://kubernetes.io/docs/concepts/policy/resource-quotas/]. However, in Rancher, resource quotas have been extended so that you can apply them to projects.

In a standard Kubernetes deployment, resource quotas are applied to individual namespaces. However, you cannot apply the quota to your namespaces simultaneously with a single action. Instead, the resource quota must be applied multiple times.

In the following diagram, a Kubernetes administrator is trying to enforce a resource quota without Rancher. The administrator wants to apply a resource quota that sets the same CPU and memory limit to every namespace in his cluster (`Namespace 1-4`) . However, in the base version of Kubernetes, each namespace requires a unique resource quota. The administrator has to create four different resource quotas that have the same specs configured (`Resource Quota 1-4`) and apply them individually.

^Base Kubernetes: Unique Resource Quotas Being Applied to Each Namespace^




Resource quotas are a little different in Rancher. In Rancher, you apply a resource quota to the project, and then the quota propagates to each namespace, whereafter Kubernetes enforces your limits using the native version of resource quotas. If you want to change the quota for a specific namespace, you can override it.

The resource quota includes two limits, which you set while creating or editing a project:

- **Project Limits:**

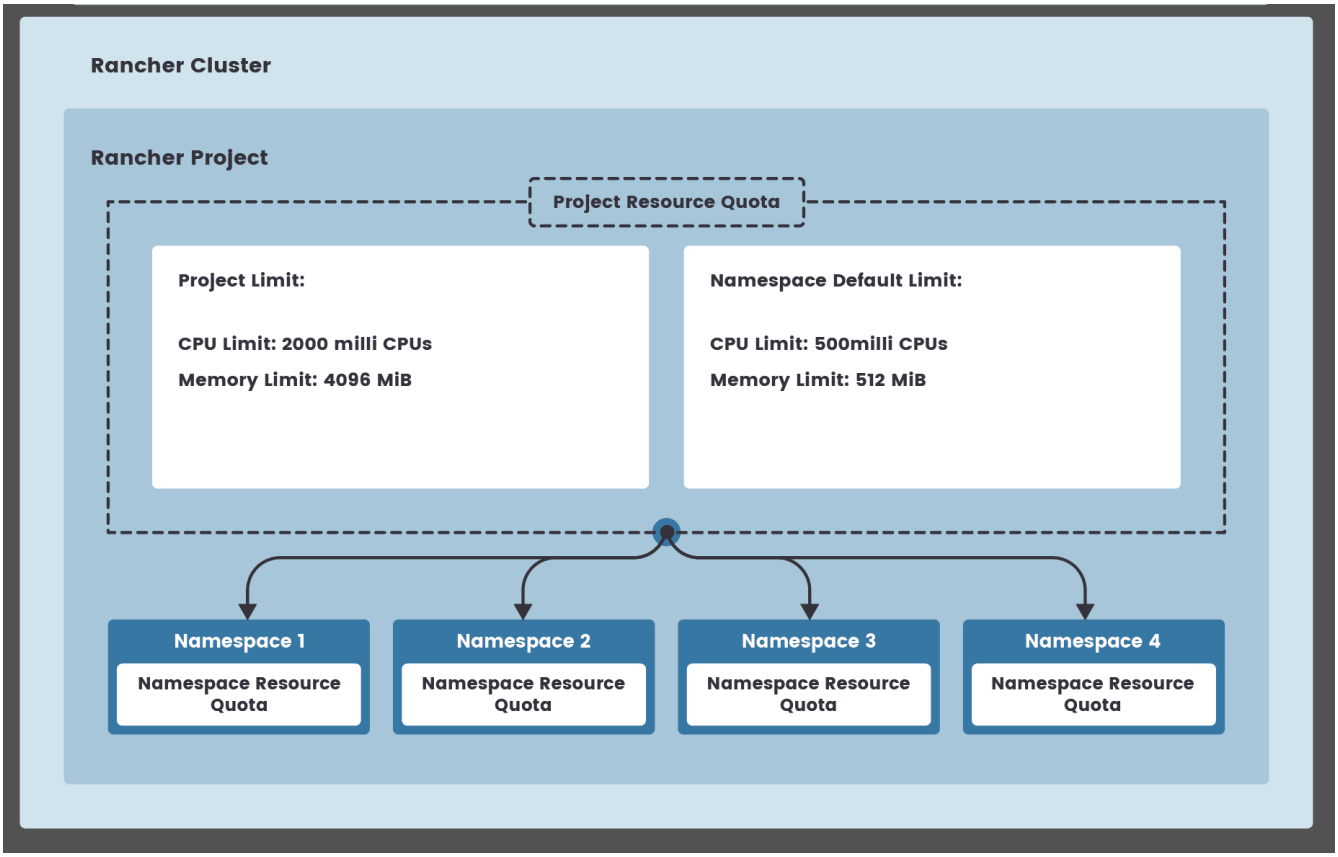
  This set of values configures a total limit for each specified resource shared among all namespaces in the project.

- **Namespace Default Limits:**

  This set of values configures the default quota limit available for each namespace for each specified resource. When a namespace is created in the project without overrides, this limit is automatically bound to the namespace and enforced.

In the following diagram, a Rancher administrator wants to apply a resource quota that sets the same CPU and memory limit for every namespace in their project (`Namespace 1-4`). However, in Rancher, the administrator can set a resource quota for the project (`Project Resource Quota`) rather than individual namespaces. This quota includes resource limits for both the entire project (`Project Limit`) and individual namespaces (`Namespace Default Limit`). Rancher then propagates the `Namespace Default Limit` quotas to each namespace (`Namespace Resource Quota`) when created.

^Rancher: Resource Quotas Propagating to Each Namespace^




Let's highlight some more nuanced functionality for namespaces created ***within*** the Rancher UI. If a quota is deleted at the project level, it will also be removed from all namespaces contained within that project, despite any overrides that may exist. Further, updating an existing namespace default limit for a quota at the project level will not result in that value being propagated to existing namespaces in the project; the updated value will only be applied to newly created namespaces in that project. To update a namespace default limit for existing namespaces you can delete and subsequently recreate the quota at the project level with the new default value. This will result in the new default value being applied to all existing namespaces in the project.

Before creating a namespace in a project, Rancher compares the amounts of the project's available resources and requested resources, regardless of whether they come from the default or overridden limits. If the requested resources exceed the remaining capacity in the project for those resources, Rancher will assign the namespace the remaining capacity for that resource.

However, this is not the case with namespaces created ***outside*** of Rancher's UI. For namespaces created via `kubectl`, Rancher will assign a resource quota that has a **zero** amount for any resource that requested more capacity than what remains in the project.

To create a namespace in an existing project via `kubectl`, use the `field.cattle.io/projectId` annotation. To override the default requested quota limit, use the `field.cattle.io/resourceQuota` annotation. You can also configure these settings in the Rancher UI. For more information see override default limits in namespaces.

Note that Rancher will only override limits for resources that are defined on the project quota.

```
apiVersion: v1
kind: Namespace
metadata:
```

```
  annotations:
    field.cattle.io/projectId: <YOUR_CLUSTER_ID>:<YOUR_PROJECT_ID>
    field.cattle.io/resourceQuota: '{"limit":{"limitsCpu":"100m",
"configMaps": "50"}}'
  name: my-ns
```

In this example, if the project's quota does not include configMaps in its list of resources, then Rancher will ignore `configMaps` in this override.

Users are advised to either use the `extended` map to configure additional custom limits for resources not built into the project, for all namespaces in the project, or to create dedicated `ResourceQuota` objects in specific namespaces for the same, just for these namespaces. Resource quotas are native Kubernetes objects, and Rancher will ignore user-defined quotas in namespaces belonging to a project with a quota, thus giving users more control.

The following table explains the key differences between the two quota types.

| Rancher Resource Quotas | Kubernetes Resource Quotas |
|---|---|
| Applies to projects and namespaces. | Applies to namespaces only. |
| Creates resource pool for all namespaces in a project. | Applies static resource limits to individual namespaces. |
| Applies resource quotas to namespaces through propagation. | Applies only to the assigned namespace. |

#### `field.cattle.io/resourceQuota` annotation

The `field.cattle.io/resourceQuota` annotation is a JSON object containing a single key, `limit`. The value of `limit` is a JSON object with keys that map to resource quota types.

All key values are strings, except for `extended`. The `extended` key contains a JSON object that maps arbitrary resource references to string values.

| Field Key | Resource Quota Type |
|---|---|
| `limitsCpu` | CPU Limit |
| `requestsCpu` | CPU Reservation |
| `limitsMemory` | Memory Limit |
| `requestsMemory` | Memory Reservation |
| `requestsStorage` | Storage Reservation |
| `servicesLoadBalancers` | Services Load Balancers |
| `servicesNodePorts` | Services Node Ports |
| `pods` | Pods |
| `services` | Services |
| `configmaps` | ConfigMaps |
| `persistentVolumeClaims` | Persistent Volume Claims |
| `replicationControllers` | Replications Controllers |
| `secrets` | Secrets |
| `extended` | Custom** |

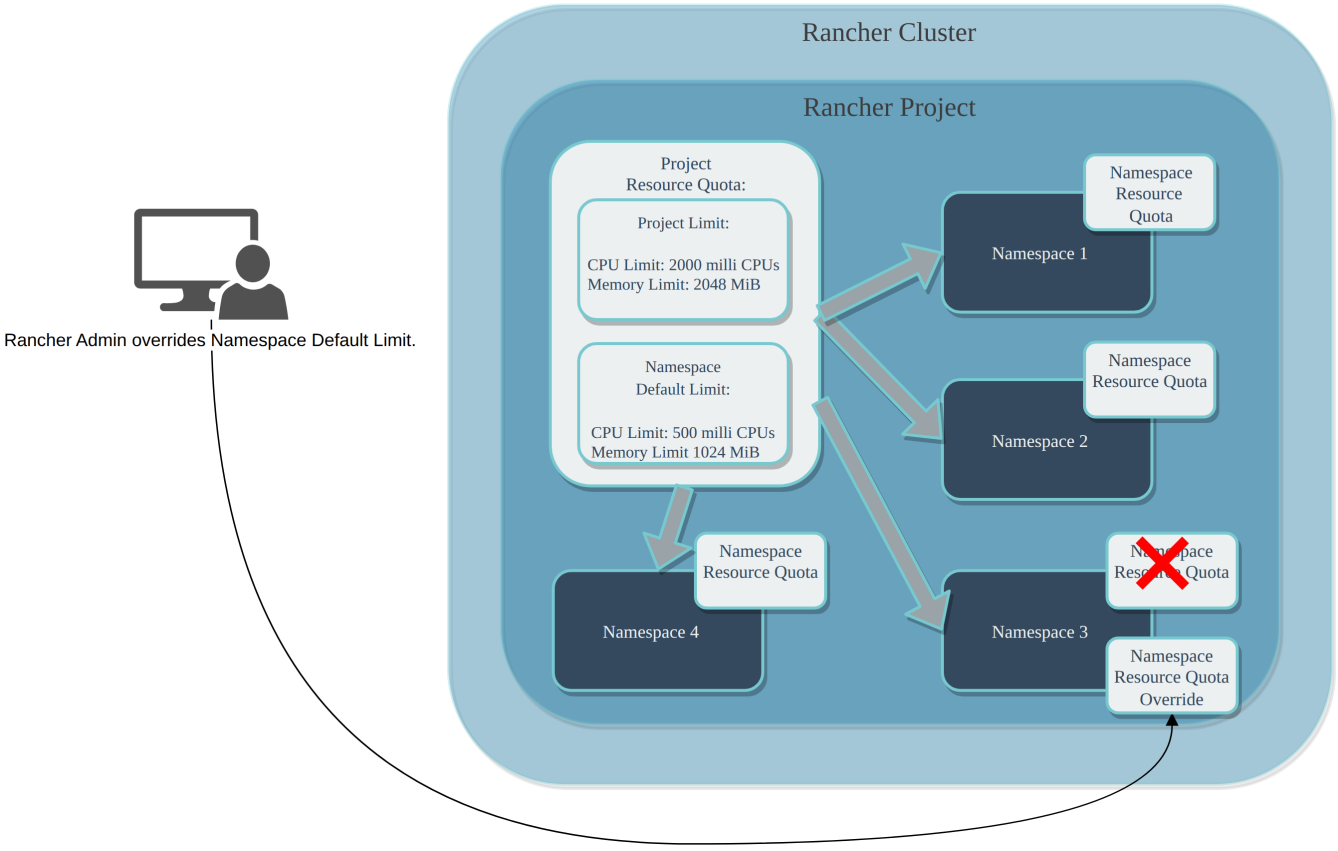
```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    # [...]
    field.cattle.io/resourceQuota:
'{"limit":{"limitsCpu":"100m","extended":{"requests.nvidia.com/gpu":"4"}
}}'
# [...]
```

**Overriding the Default Limit for a Namespace**

Although the **Namespace Default Limit** propagates from the project to each namespace when created, in some cases, you may need to increase (or decrease) the quotas for a specific namespace. In this situation, you can override the default limits by editing the namespace.

In the diagram below, the Rancher administrator has a resource quota in effect for their project. However, the administrator wants to override the cpu limits for `Namespace 3` so that it has more resources available. Therefore, the administrator raises the cpu limits for `Namespace 3` so that the namespace can access more resources.

^Namespace Default Limit Override^




How to: Editing Namespace Resource Quotas

#### Editing Namespace Resource Quotas

If there is a resource quota configured for a project, you can override the namespace default limit to provide a specific namespace with access to more (or less) project resources.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to edit a namespace resource quota and click **Explore**.
3. Click **Cluster** › **Projects/Namespaces**.
4. Find the namespace for which you want to edit the resource quota. Click ☰ > **Edit Config**.
5. Edit the resource limits. These limits determine the resources available to the namespace. The limits must be set within the configured project limits.

   For more information about each **Resource Type**, see the type reference.



- If a resource quota is not configured for the project, these options will not be available.
- If you enter limits that exceed the configured project limits, Rancher will not let you save your edits.

**Result:** Your override is applied to the namespace's resource quota.

#### Resource Quota Type Reference

When you create a resource quota, you are configuring the pool of resources available to the project. Rancher supports the use of arbitrary resource references and their quotas. This allows you to utilize all upstream Kubernetes `ResourceQuota` [https://kubernetes.io/docs/concepts/policy/resource-quotas/#types-of-resource-quota] types when managing project resource quotas.

You can set resource limits for the following predefined resource types, where the `Custom` type enables specification of arbitrary resources and their quotas.



Support for arbitrary resource references using the `Custom` type does not cover resources in the `ext.cattle.io` API group.

| Resource Type | Description |
|---|---|
| CPU Limit* | The maximum amount of CPU (in millicores [https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#meaning-of-cpu]) allocated to the project/namespace.[1] |
| CPU Reservation* | The minimum amount of CPU (in millicores) guaranteed to the project/namespace.[1] |
| Memory Limit* | The maximum amount of memory (in bytes [https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#meaning-of-memory]) allocated to the project/namespace.[1] |
| Memory Reservation* | The minimum amount of memory (in bytes) guaranteed to the project/namespace.[1] |

| Resource Type | Description |
|---|---|
| Storage Reservation | The minimum amount of storage (in gigabytes) guaranteed to the project/namespace. |
| Services Load Balancers | The maximum number of load balancers services that can exist in the project/namespace. |
| Services Node Ports | The maximum number of node port services that can exist in the project/namespace. |
| Pods | The maximum number of pods that can exist in the project/namespace in a non-terminal state (i.e., pods with a state of `.status.phase in (Failed, Succeeded)` equal to true). |
| Services | The maximum number of services that can exist in the project/namespace. |
| ConfigMaps | The maximum number of ConfigMaps that can exist in the project/namespace. |
| Persistent Volume Claims | The maximum number of persistent volume claims that can exist in the project/namespace. |
| Replications Controllers | The maximum number of replication controllers that can exist in the project/namespace. |
| Secrets | The maximum number of secrets that can exist in the project/namespace. |
| Custom** | The specification of arbitrary resources and their quotas, beyond the resource types built into projects, as listed above. |



*

When setting resource quotas, if you set anything related to CPU or Memory (i.e. limits or reservations) on a project or namespace, all containers will require a respective CPU or Memory field set during creation. A container default resource limit can be set at the same time to avoid the need to explicitly set these limits for every workload. See the Kubernetes documentation [https://kubernetes.io/docs/concepts/policy/resource-quotas/#requests-vs-limits] for more details on why this is required.



**

For example:

- `requests.nvidia.com/gpu: 4`
- `gold.storageclass.storage.k8s.io/requests.storage: 500Gi`
- `count/podtemplates: 10`

See the Kubernetes documentation [https://kubernetes.io/docs/concepts/policy/resource-quotas/#quota-for-extended-resources] for many more examples.

### Setting Container Default Resource Limits

When setting resource quotas, if you set anything related to CPU or Memory (i.e. limits or reservations) on a project / namespace, all containers will require a respective CPU or Memory field set during creation. See the Kubernetes documentation [https://kubernetes.io/docs/concepts/policy/resource-quotas/#requests-vs-limits] for more details on why this is required.

To avoid setting these limits on each and every container during workload creation, a default container resource limit can be specified on the namespace.

#### Editing the Container Default Resource Limit

Edit the container default resource limit when:

- You have a CPU or Memory resource quota set on a project, and want to supply the corresponding default values for a container.
- You want to edit the default container resource limit.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to edit the default resource limit and click **Explore**.
3. Click **Cluster › Projects/Namespaces**.
4. Find the project that you want to edit the container default resource limit. From that project, select ☰ **> Edit Config**.
5. Expand **Container Default Resource Limit** and edit the values.

#### Resource Limit Propagation

When the default container resource limit is set at a project level, the parameter will be propagated to any namespace created in the project after the limit has been set. For any existing namespace in a project, this limit will not be automatically propagated. You will need to manually set the default container resource limit for any existing namespaces in the project in order for it to be used when creating any containers.

You can set a default container resource limit on a project and launch any catalog applications.

Once a container default resource limit is configured on a namespace, the default will be pre-populated for any containers created in that namespace. These limits/reservations can always be overridden during workload creation.

#### Container Resource Quota Types

The following resource limits can be configured:

| Resource Type | Description |
|---|---|
| CPU Limit | The maximum amount of CPU (in millicores [https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#meaning-of-cpu]) allocated to the container. |
| CPU Reservation | The minimum amount of CPU (in millicores) guaranteed to the container. |
| Memory Limit | The maximum amount of memory (in bytes) allocated to the container. |
| Memory Reservation | The minimum amount of memory (in bytes) guaranteed to the container. |
| NVIDIA GPU Limit/Reservation | The amount of GPUs allocated to the container. The limit and reservation are always the same for GPUs. |

# Helm Charts and Apps

In this section, you'll learn how to manage Helm chart repositories and apps in Rancher.

## How Helm Charts Work in Rancher

Helm chart repositories in Rancher are managed using **Apps**.

Rancher uses a catalog-like system to import bundles of charts from repositories and then uses those charts to either deploy custom Kubernetes applications or Rancher's tools such as Monitoring or Istio. Rancher tools come as pre-loaded repositories which deploy as standalone Helm charts. Any additional repositories are only added to the current cluster.

### Catalogs, Apps, and the Rancher UI

Rancher v2.4 and earlier, repositories of ready-to-deploy applications were called "catalogs". These repositories were managed through the **Catalogs** section of the UI.

Rancher v2.5 replaced the former catalog system with a new **Apps & Marketplace** feature.

Since Rancher v2.6.5, the **Apps & Marketplace** feature is named **Apps** in the UI.

### Versioning Scheme

The Rancher feature charts versioning scheme is centered around the major version of the charts and the `+up` annotation for upstream charts, where applicable.

**Major Version:** The major versions of feature charts are tied to particular minor versions of Rancher. When you upgrade to a new Rancher minor version, you should ensure that all of your feature charts are also upgraded to the correct release line for the chart.

**Charts based on upstream:** When you upgrade, make sure that the upstream chart version is compatible with your Rancher version. The `+up` annotation for the chart indicates which upstream version the Rancher chart is tracking. For example, `100.x.x+up16.6.0` for Monitoring tracks upstream kube-prometheus-stack `16.6.0` with some additional Rancher patches.

When upgrading Rancher versions, don't downgrade the version of the chart that you are using. For example, if you are using a version of Monitoring that is later than `16.6.0` in Rancher v2.5, you shouldn't upgrade to `100.x.x+up16.6.0`. Instead, you should upgrade to the appropriate version in the next release.

#### Prerelease Versions

Prereleases adhere to the specification [https://semver.org/#spec-item-9] defined by Semantic Versioning 2.0.0 [https://semver.org/]. For example, a Helm chart with a version of `0.1.3-dev.12ab4f` is considered a prerelease. Prerelease versions are not displayed by default and must be configured to do so.

To display prerelease versions:

1. Click on your user avatar in the upper right corner.
2. Click **Preferences**.
3. Under **Helm Charts**, select **Include Prerelease Versions**.

## Access Charts

The **Charts** page contains all Rancher, Partner, and Custom charts. You can filter charts by selecting the left-most dropdown menu:

- Rancher tools such as Logging or Monitoring are listed under the **Rancher** label.
- Partner charts are under the **Partners** label.
- Custom charts are listed under the name of their respective repository.

All three types of charts are deployed and managed in the same way.



Apps managed by the Cluster Manager (the global view in the legacy Rancher UI) continue to be managed only by the Cluster Manager, and apps managed with **Apps** in the new UI must be managed only by **Apps**.

To access the **Charts** page:

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose charts you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Charts**.

## Manage Repositories

The **Repositories** page lists your Helm repositories. These include traditional Helm endpoints which have an index.yaml, and Git repositories that are cloned and point to a specific branch. To use custom charts, add your repository here. After you add a repository, you can access custom charts in the **Charts** page, listed under the name of the repository.

To access the **Repositories** page:

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Repositories**.

### Add Custom Git Repositories

To add a custom Git repository that contains your Helm charts or cluster template definitions:

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Repositories**.
4. Click **Create**.
5. Select the target, **Git repository containing Helm chart...**.
6. You must enter a name and a Git repository URL. The other fields, including the description, are optional.

Enter an alternative branch name if you don't want to pull from whichever branch the repo owner has set as the default. Usually, the default branch is named either `main` or `master`.

7. Click **Create** to add the repository.

After you add a chart repository to Rancher, it becomes available immediately.

#### Add Custom Helm Chart Repositories

You can add your own Helm chart repositories to serve chart packages to Rancher. You can use any HTTP server, as long as the server can respond to GET requests and serve YAML files and tar archives.

For more information on Helm chart repositories, see the official Helm docs [https://helm.sh/docs/topics/chart_repository/].

To add a custom Helm chart repository to Rancher:

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Repositories**.
4. Click **Create**.
5. Select the target, **http(s) URL to an index generated by Helm**.
6. Enter a repo name and the index URL address of the chart repository.
7. Click **Create** to add the repository.

#### Add Private Git/Helm Chart Repositories

You can add private Git or Helm chart repositories with SSH key credentials or an HTTP basic auth secret, such as a username and password.

#### Add a Private CA to Repositories

To add a private CA to Helm chart repositories, you must add a base64 encoded copy of the CA certificate in DER format to the `spec.caBundle field` of the chart repo, such as `openssl x509 -outform der -in ca.pem | base64 -w0`. Instructions are the same for both Git-based and HTTP-based repositories:

1. Click ☰. Under **Explore Cluster** in the left navigation menu, select a cluster.
2. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Repositories**.
3. Find the row associated with the Git-based or HTTP-based repository you want to add a private CA to, and click ☰ **> Edit YAML**.
4. Set the `caBundle` value, as in the following example:

```
    # [...]
    spec:
      caBundle: MIIFXzCC...DArGmr4=

MIIFXzCCA0egAwIBAgIUWNy8WrvSkgNzV0zdWRP79j9cVcEwDQYJKoZIhvcNAQELBQAwPzEL
```

```
MAkGAlUEBhMCVVMxCzAJBgNVBAgMAkNBMRQwEgYDVQQKDAtNeU9yZywgSW5jLjENMAsGAlUE
AwwEcm9vdDAeFw0yMTEyMTQwODMyMTdaFw0yNDEwMDMwODMyMT
        ...

nDxZ/tNXt/WPJr/PgEB3hQdInDWYMg7vGO0Oz00G5kWg0sJ0ZTSoAl0ZwdjIdGEeKljlNlPy
AqpQ+uDnmx6DW+zqfYtLnc/g6GuLLVPamraqN+gyU8CHwAWPNjZonFN9Vpg0PIklI2zuOc4E
HifoTAXSpnjfzfyAxCaZsnTptimlPFJJqAMj+FfDArGmr4=
    # [...]
```



*Helm chart repositories with authentication*

The Repo.Spec contains a `disableSameOriginCheck` value. This value allows you to bypass the same origin checks, sending the repository Authentication information as a Basic Auth Header with all API calls. This is not recommended but can be used as a temporary solution in cases of non-standard Helm chart repositories, such as those that have redirects to a different origin URL.

To use this feature for an existing Helm chart repository, follow previous steps up to edit the YAML. On the `spec` portion of the YAML file, add `disableSameOriginCheck` and set it to `true`.

```yaml
# [...]
spec:
  disableSameOriginCheck: true
# [...]
```

#### Add Custom OCI Chart Repositories

Helm v3 introduced storing Helm charts as Open Container Initiative (OCI) [https://opencontainers.org/about/overview/] artifacts in container registries. With Rancher v2.9.0, you can add OCI-based Helm chart repositories [https://helm.sh/docs/topics/registries/] alongside HTTP-based and Git-based repositories. This means you can deploy apps that are stored as OCI artifacts. For more information, see Using OCI Helm Chart Repositories.

#### Helm Compatibility

Only Helm 3 compatible charts are supported.

#### Refresh Chart Repositories

The **Refresh** button can be used to sync changes from selected Helm chart repositories on the **Repositories** page.

To refresh a chart repository:

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.

3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Repositories**.
4. Use the toggle next to the **State** field to select all repositories, or toggle specified chart repositories to sync changes.
5. Click **Refresh**.
6. The ⋮ at the end of each chart repository row also includes a **Refresh** option, which can be clicked to refresh the respective repository.

Non-Airgap Rancher installations upon refresh will reflect any chart repository changes immediately and you will see the **State** field for updated repositories move from `In Progress` to `Active` once the action is completed.

Airgap installations where Rancher is configured to use the packaged copy of Helm system charts (`useBundledSystemChart=true`) will only refer to the system-chart [https://github.com/rancher/system-charts] repository that comes bundled and will not be able to be refreshed or synced.

#### Refresh Interval

By default, Rancher syncs each Helm repository every hour (3600 seconds). You can configure the frequency using the `refreshInterval` field of the `ClusterRepo` custom resource definition (CRD).

To modify the refresh interval of a chart repository:

1. Select ⋮ **> Cluster Management**.
2. Locate the cluster containing the repository you want to modify, and click **Explore**.
3. In the left navigation menu, select **Apps > Repositories**.
4. Locate the repository, and select ⋮ **> Edit Config**.
5. Select **Enable Refresh Interval**.
6. In the **Refresh Interval** field, enter a number.
7. From the **Unit** dropdown menu, select a time unit (**secs**, **mins**, **hours**, or **days**).
8. Click **Save**.

**Result:** Rancher syncs the repository according to the configured interval.

### Enable/Disable Helm Chart Repositories

Helm repositories are enabled by default.

To disable a chart repository:

1. Click ⋮ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps > Repositories**.
4. Find the repository you want to disable, and click ⋮ **> Edit YAML**.
5. Set the **Enabled** field under **Spec** to **false**.
6. Click **Save**.

7. When you disable a repository, updates are disabled and new changes to the clusterRepo are not applied.

To enable a chart repository:

1. Click ☰ > **Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps > Repositories**.
4. Find the repository you want to disable, and click ☰ > **Edit YAML**.
5. Set the **Enabled** field under **Spec** to **true**.
6. Click **Save**.

## Deploy and Upgrade Charts

To install and deploy a chart:

1. Click ☰ > **Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation menu on the **Cluster Dashboard**, click **Apps › Charts**.
4. Select a chart, and click **Install**.

Rancher and Partner charts may have extra configurations available through custom pages or questions.yaml files. However, all chart installations can modify the values.yaml and other basic settings. After you click **Install**, a Helm operation job is deployed, and the console for the job is displayed.



If the cluster uses an authenticated registry, Rancher automatically injects the required pull secrets into the Helm values for charts from the built-in Rancher catalog. This behavior does not apply to charts with custom pull secrets specified during installation or charts from custom repositories. For more information, see Authenticated Private Registries.

To view all recent changes, click **Apps › Recent Operations** in the left navigation menu. From there you can view the calls, conditions, events, and logs.

After installing a chart, you can view it by clicking **Apps › Installed Apps** in the left navigation menu. You can upgrade or delete the installation, and see further details. Upgrading uses the same forms and values as you saw during initial installation.

Most Rancher tools have additional pages located in the toolbar below the **Apps** section to help manage and use the features. These pages include links to dashboards, forms to easily add Custom Resources, and additional information.



If you are upgrading your chart using **Customize Helm options before upgrade**, and your chart contains immutable fields, using the `--force` option may result in errors. This is because some objects in Kubernetes can't be changed after they're created. To prevent this error:

- Use the default upgrade option (i.e don't use `--force`).
- Uninstall the existing chart and install the upgraded chart.
- Delete the resources with immutable fields from the cluster before performing a forced upgrade.

## Limitations

- Dashboard apps or Rancher feature charts can't be installed using the Rancher CLI.
- When determining the most recent version to display for the **Upgradable** column on the **Apps › Installed Apps** page, rather than only considering versions of the Helm chart from the repository it was installed from, Rancher considers versions of the Helm chart from all repositories on the cluster.

  For example, suppose you install `cert-manager` v1.13.0 from repository A, where v1.14.0 is now the most recent version available. In this case, you expect **Upgradable** to display v1.14.0. However, if the cluster also has access to repository B where v1.15.0 of `cert-manager` is available, then **Upgradable** displays v1.15.0 even though the original installation used repository A.

## Creating Apps



For a complete walkthrough of developing charts, see the Chart Template Developer's Guide [https://helm.sh/docs/chart_template_guide/] in the official Helm documentation.

### Chart Types

Rancher supports two different types of charts: Helm charts and Rancher charts.

#### Helm Charts

Native Helm charts include an application along with other software required to run it. When deploying native Helm charts, you can provide the chart's parameter values in a YAML editor.

#### Rancher Charts

Rancher charts are native Helm charts with two files that enhance user experience: `app-readme.md` and `questions.yaml`. Read more about them in Additional Files for Rancher Charts.

Rancher charts add simplified chart descriptions and configuration forms to make the application deployment easy. Rancher users do not need to read through the entire list of Helm variables to understand how to launch an application.

### Chart Directory Structure

You can provide Helm Charts in a standard, HTTP based Helm Repository. For more information see the Chart Repository Guide [https://helm.sh/docs/topics/chart_repository] in the official Helm documentation.

Alternatively you can organize your charts in a Git Repository and directly add this to the App Marketplace.

The following table demonstrates the directory structure for a Git repository. The `charts` directory is the top level directory under the repository base. Adding the repository to Rancher will expose all charts contained within it. The `questions.yaml`, `README.md`, and `requirements.yml` files are specific to Rancher charts, but are optional for chart customization.

```
<Repository-Base>/
 ▯
 ▯▯▯ charts/
 ▯   ▯▯▯ <Application Name>/      # This directory name will be surfaced
in the Rancher UI as the chart name
 ▯   ▯   ▯▯▯ <App Version>/   # Each directory at this level provides
different app versions that will be selectable within the chart in the
Rancher UI
 ▯   ▯   ▯   ▯▯▯ Chart.yaml   # Required Helm chart information file.
 ▯   ▯   ▯   ▯▯▯ questions.yaml   # Form questions displayed within the
Rancher UI. Questions display in Configuration Options.*
 ▯   ▯   ▯   ▯▯▯ README.md         # Optional: Helm Readme file
displayed within Rancher UI. This text displays in Detailed
Descriptions.
 ▯   ▯   ▯   ▯▯▯ requirements.yml  # Optional: YAML file listing
dependencies for the chart.
 ▯   ▯   ▯   ▯▯▯ values.yml        # Default configuration values for
the chart.
 ▯   ▯   ▯   ▯▯▯ templates/        # Directory containing templates
that, when combined with values.yml, generates Kubernetes YAML.
```

#### Additional Files for Rancher Charts

Before you create your own custom catalog, you should have a basic understanding about how a Rancher chart differs from a native Helm chart. Rancher charts differ slightly from Helm charts in their directory structures. Rancher charts include two files that Helm charts do not.

- `app-readme.md`

  A file that provides descriptive text in the chart's UI header.

- `questions.yml`

  A file that contains questions for a form. These form questions simplify deployment of a chart. Without it, you must configure the deployment using a values YAML config, which is more difficult. The following images display the difference between a Rancher chart (which includes `questions.yml`) and a native Helm chart (which does not).

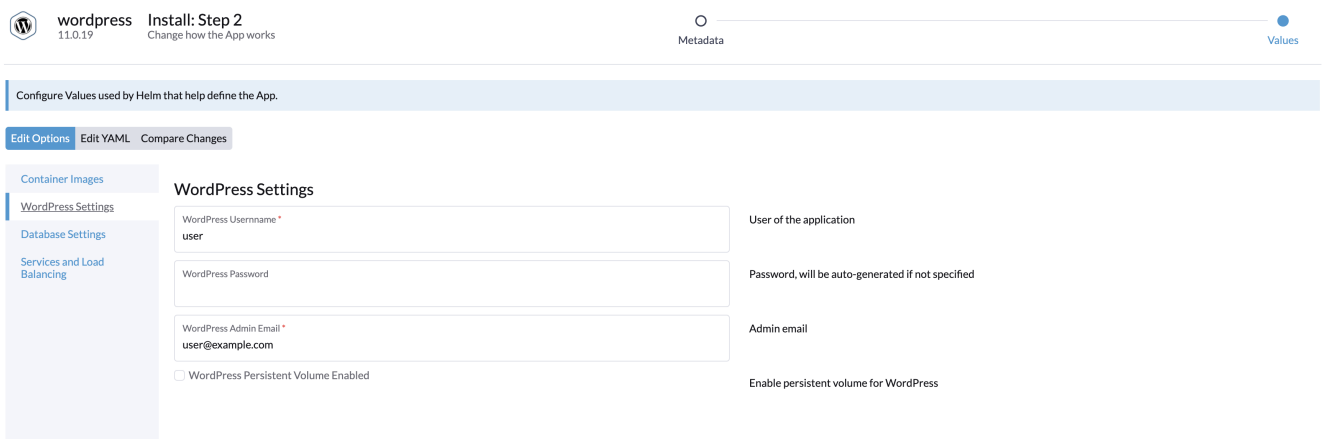



*Figure 13. Rancher Chart with `questions.yml`*

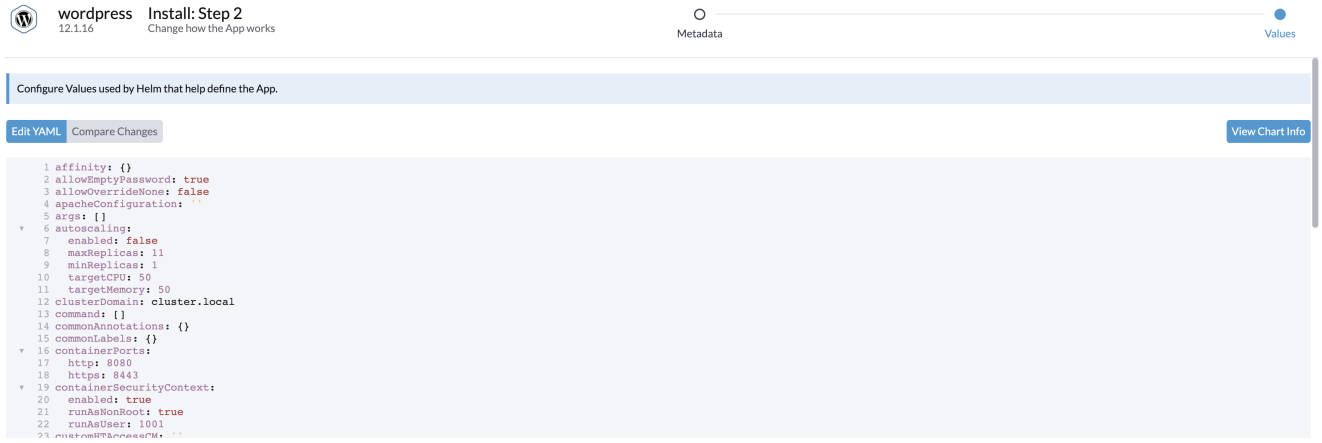



*Figure 14. Helm Chart without `questions.yml`*

### Chart.yaml

#### Annotations

Rancher supports additional annotations that you can add to the `Chart.yaml` file. These annotations allow you to define application dependencies or configure additional UI defaults:

| Annotation | Description | Example |
|---|---|---|
| catalog.cattle.io/auto-install | If set, will install the specified chart in the specified version before installing this chart | other-chart-name=1.0.0 |
| catalog.cattle.io/display-name | A display name that should be displayed in the App Marketplace instead of the chart name | Display Name of Chart |
| catalog.cattle.io/namespace | A fixed namespace where the chart should be deployed in. If set, this can't be changed by the user | fixed-namespace |
| catalog.cattle.io/release-name | A fixed release name for the Helm installation. If set, this can't be changed by the user | fixed-release-name |
| catalog.cattle.io/requests-cpu | Total amount of CPU that should be unreserved in the cluster. If less CPU is available, a warning will be shown | 2000m |

| Annotation | Description | Example |
|---|---|---|
| catalog.cattle.io/requests-memory | Total amount of memory that should be unreserved in the cluster. If less memory is available, a warning will be shown | 2Gi |
| catalog.cattle.io/os | Restricts the OS where this chart can be installed. Possible values: `linux`, `windows`. Default: no restriction | linux |

#### Keywords

With the `keywords` option in the `Chart.yaml` file it is possible to provide a list of categories for sorting your application in the Rancher UI, like `infrastructure`, `monitoring` and more.

#### questions.yml

Inside the `questions.yml`, most of the content will be around the questions to ask the end user, but there are some additional fields that can be set in this file.

#### Min/Max Rancher versions

For each chart, you can add the minimum and/or maximum Rancher version, which determines whether or not this chart is available to be deployed from Rancher.



Even though Rancher release versions are prefixed with a `v`, there is *no* prefix for the release version when using this option.

```
rancher_min_version: 2.3.0
rancher_max_version: 2.3.99
```

#### Question Variable Reference

This reference contains variables that you can use in `questions.yml` nested under `questions:`.

| Variable | Type | Required | Description |
|---|---|---|---|
| variable | string | true | Define the variable name specified in the `values.yml` file, using `foo.bar` for nested objects. |
| label | string | true | Define the UI label. |
| description | string | false | Specify the description of the variable. |
| type | string | false | Default to `string` if not specified (current supported types are string, multiline, boolean, int, enum, password, storageclass, hostname, pvc, secret and cloudcredential). |
| required | bool | false | Define if the variable is required or not (true \| false) |

| Variable | Type | Required | Description |
|---|---|---|---|
| default | string | false | Specify the default value. Only used if there is no corresponding value in the `values.yml` file. |
| group | string | false | Group questions by input value. |
| min_length | int | false | Min character length. |
| max_length | int | false | Max character length. |
| min | int | false | Min integer value. |
| max | int | false | Max integer value. |
| options | []string | false | Specify the options when the variable type is `enum`, for example: options:<br>- "ClusterIP"<br>- "NodePort"<br>- "LoadBalancer" |
| valid_chars | string | false | Regular expression for input chars validation. |
| invalid_chars | string | false | Regular expression for invalid input chars validation. |
| subquestions | []subquestion | false | Add an array of subquestions. |
| show_if | string | false | Show current variable if conditional variable is true. For example `show_if: "serviceType=Nodeport"` |
| show_subquestion_if | string | false | Show subquestions if is true or equal to one of the options. for example `show_subquestion_if: "true"` |



`subquestions[]` cannot contain `subquestions` or `show_subquestions_if` keys, but all other keys in the above table are supported.

## Using OCI-Based Helm Chart Repositories

Helm v3 introduced storing Helm charts as Open Container Initiative (OCI) [https://opencontainers.org/about/overview/] artifacts in container registries. With Rancher v2.9.0, you can add OCI-based Helm chart repositories [https://helm.sh/docs/topics/registries/] alongside HTTP-based and Git-based repositories. This means that you can deploy apps that are stored as OCI artifacts.

### Add an OCI-Based Helm Chart Repository

To add an OCI-based Helm chart repository through the Rancher UI:

1. Click ☰ > **Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.

3. In the left navigation bar, select menu:Apps[Repositories].
4. Click **Create**.
5. Enter a **Name** for the registry. Select **OCI Repository** as the target.
6. Enter the **OCI Repository Host URL** for the registry. The registry endpoint must not contain anything besides OCI Helm Chart artifacts. The artifacts should all have unique names. If you attempt to add an endpoint that contains any other kinds of files or artifacts, the OCI repository will not be added.



You can use the **OCI URL** field to fine-tune how many charts from the registry are available for installation on Rancher. More generic endpoints target more charts, as the following examples demonstrate:

```
- `oci://<registry-host>`: Every chart in the registry
becomes available for installation, regardless of namespace
or tag.
- `oci://<registry-host>/<namespace>`: Every chart in the
specified namespace within the registry becomes available
for installation.
- `oci://<registry-host>/<namespace>/<chart-name>`: Only
the specified chart and any associated tags or versions of
that chart become available for installation.
- `oci://<registry-host>/<namespace>/<chart-name>:<tag>`:
Only the chart with the specified tag becomes available for
installation.
```

1. Set up authentication. Select **Basicauth** from the authentication field and enter a username and password as required. Otherwise, create or select an **Authentication** secret. See Authentication for a full description.
2. (optional) Enter a base64 encoded DER certificate in the **CA Cert Bundle** field. This field is for cases where you have a private OCI-based Helm chart repository and need Rancher to trust its certificates.
3. (optional) To allow insecure connections without performing an SSL check, select **Skip TLS Verification**. To force Rancher to use HTTP instead of HTTPS to send requests to the repository, select **Insecure Plain Http**.
4. (optional) If your repository has a rate limiting policy and may respond with status code `429 Too Many Requests`, you may want to fill out the fields under **Exponential Back Off**:

   - **Min Wait**: The minimum duration in seconds that Rancher should wait before retrying. The default is `1` second.
   - **Max Wait**: The maximum duration in seconds that Rancher should wait before retrying. The default is 5 second.
   - **Max Number of Retries**: The default is 5 retries.

   Once these values are set, Rancher responds to the `429` status code by staggering requests based on the minimum and maximum wait values. The wait time between retries increases exponentially, until Rancher has sent the maximum number of retries set. See Rate Limiting for more details.

5. Add any labels and annotations.

6. Click **Create**.

It may take some time for the OCI repository to activate. This is particularly true if the OCI endpoint contains multiple namespaces.

#### Authentication for OCI-Based Helm Chart Repositories

Rancher supports BasicAuth for OCI registries. You must create a **BasicAuth** Kubernetes secret [https://kubernetes.io/docs/concepts/configuration/secret/#basic-authentication-secret]. You can also create the secret through the Rancher UI.

The CRD that is linked to the OCI-based Helm repository is `ClusterRepo`.

#### View Helm Charts in OCI-Based Helm Chart Repositories

To view Helm charts in the OCI-based Helm chart repository after it achieves an `Active` state:

1. Click ☰. Under **Explore Cluster** in the left navigation menu, select a cluster.
2. Click menu:Apps[Charts].
3. Select the OCI-based Helm chart repository from the dropdown.

#### Refresh an OCI-Based Helm Chart Repository

Rancher automatically refreshes the OCI-based Helm chart repository every 6 hours.

If you need to update immediately, you can perform a manual refresh.

#### Update an OCI-Based Helm Chart Repository Configuration

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation bar, select menu:Apps[Repositories].
4. Find the row associated with the OCI-based Helm chart repository, and click ☰.
5. From the submenu, select **Edit Config**.

#### Delete an OCI-Based Helm Chart Repository

1. Click ☰ **> Cluster Management**.
2. Find the name of the cluster whose repositories you want to access. Click **Explore** at the end of the cluster's row.
3. In the left navigation bar, select menu:Apps[Repositories].
4. Select the row associated with the OCI-based Helm chart repository, and click **Delete**.

#### Size Limitations of OCI-Based Helm Chart Repositories in Rancher

Due to security concerns, there are limitations on how large of a Helm chart you can deploy through an OCI-based repository, and how much metadata you can use to describe the Helm charts within a single OCI endpoint.

Rancher can deploy OCI Helm charts up to 20 MB in size.

#### Rate Limiting of OCI-Based Helm Chart Repositories

Different OCI registries implement rate limiting in different ways.

Most servers return a `Retry-After` header, indicating how long to wait before rate limiting is lifted.

Docker Hub returns a `429` status code when it completes all allocated requests. It also returns a `RateLimit-Remaining` header which describes the rate limiting policy.

Rancher currently checks for the `Retry-After` header. It also handles Docker Hub-style responses (status code `429` and the `RateLimit-Remaining` header) and automatically waits before making a new request. When handling `Retry-After` or Docker Hub-style responses, Rancher ignores `ExponentialBackOff` values.

If you have an OCI-based Helm chart repository which doesn't implement the `Retry-After` or `RateLimit-Remaining` headers, and think you may be rate-limited at some point, fill out the fields under **Exponential Back Off** when you add the repository.

For example, if you have an OCI-based Helm chart repository that doesn't return a `Retry-After` header, but you know that the server allows 50 requests in 24 hours, you can provide Rancher a **Min Wait** value of **86400** seconds, a **Max Wait** value of **90000** seconds, and a **Max Number of Retries** value of **1**. Then, if Rancher gets rate limited by the server, Rancher will wait for 24 hours before trying again. The request should succeed as Rancher hasn't sent any other requests in the previous 24 hours.

#### Troubleshooting OCI-based Helm Registries

- To enhance logging information, enable the debug option while deploying Rancher.
- If there is any discrepancy between the repository contents and Rancher, you should refresh the cluster repository as a first resort. If the discrepancy persists, delete the OCI-based Helm chart repository from Rancher and add it again. Deleting the repository won't delete any Helm charts that are already installed.
- Apps installed through OCI-based Helm chart repositories are subject to a known issue with how Rancher displays upgradeable version information. See the Limitations section of **Helm Charts and Apps** for more details.

# Security

## SUSE Rancher Prime Security Guides

```
<table width="100%">
  <tr style="vertical-align: top;text-align: center;border-bottom-style:
none;border-top-style: solid;border-top-width: 2px;border-top-color:
#c0c2c4;">
    <td width="30%">
      <h4>Security policy</h4><br/>
      <p style={{padding: '8px'}}>Rancher Labs supports responsible
disclosure, and endeavours to resolve all issues in a reasonable time
frame. </p>
    </td>
    <td width="30%">
      <h4>Reporting process</h4><br/>
      <p style={{padding: '8px'}}>Please submit possible security issues by
emailing <a href="mailto:security-rancher@suse.com">security-
rancher@suse.com</a> .</p>
    </td>
    <td width="30%">
      <h4>Announcements</h4><br/>
      <p style={{padding:'8px'}}>Subscribe to the <a
href="https://forums.rancher.com/c/announcements">Rancher announcements
forum</a> for release updates.</p>
    </td>
  </tr>
</table>
```

Security is at the heart of all Rancher features. From integrating with all the popular authentication tools and services, to an enterprise grade RBAC capability, Rancher makes your Kubernetes clusters even more secure.

On this page, we provide security related documentation along with resources to help you secure your Rancher installation and your downstream Kubernetes clusters.

### SUSE Security Integration with Rancher

NeuVector is an open-source, container-focused security application that is now integrated into Rancher. NeuVector provides production security, DevOps vulnerability protection, and a container firewall, et al. Please see the Rancher docs and the SUSE Security docs [https://documentation.suse.com/cloudnative/security/5.4/en/overview.html] for more information.

### Running a Compliance Security Scan on a Kubernetes Cluster

Rancher leverages kube-bench [https://github.com/aquasecurity/kube-bench] to run a security scan to check whether Kubernetes is deployed according to security best practices.

When Rancher runs a Compliance scan on a cluster, it generates a report showing the results of each test, including a summary with the number of passed, skipped and failed tests. The report also includes remediation steps for any failed tests.

For details, refer to the section on security scans.

### SELinux RPM

We provide three RPMs (RPM Package Manager) that enable Rancher products to function properly on SELinux-enforcing hosts: `rancher-selinux`, `rke2-selinux` and `k3s-selinux`. For details, see this page.

### Rancher Hardening Guide

The Rancher Hardening Guide is based on controls and best practices found in the CIS Kubernetes Benchmark [https://www.cisecurity.org/benchmark/kubernetes/] from the Center for Internet Security.

The hardening guides provide prescriptive guidance for hardening a production installation of Rancher. See Rancher's guides for Self Assessment of the CIS Kubernetes Benchmark for the full list of security controls.

> The hardening guides describe how to secure the nodes in your cluster, and it is recommended to follow a hardening guide before installing Kubernetes.

Each version of the hardening guide is intended to be used with specific versions of the CIS Kubernetes Benchmark, Kubernetes, and Rancher.

### The CIS Benchmark and Self-Assessment

The benchmark self-assessment is a companion to the Rancher security hardening guide. While the hardening guide shows you how to harden the cluster, the benchmark guide is meant to help you evaluate the level of security of the hardened cluster.

This guide walks through the various controls and provide updated example commands to audit compliance in Rancher created clusters. The original benchmark documents can be downloaded from the CIS website [https://www.cisecurity.org/benchmark/kubernetes/].

Each version of Rancher's self-assessment guide corresponds to specific versions of the hardening guide, Rancher, Kubernetes, and the CIS Benchmark.

### Third-party Penetration Test Reports

Rancher periodically hires third parties to perform security audits and penetration tests of the Rancher software stack. The environments under test follow the Rancher provided hardening guides at the time of the testing. Previous penetration test reports are available below.

Results:

- Cure53 Pen Test - July 2019 [https://releases.rancher.com/documents/security/pen-tests/2019/RAN-01-cure53-report.final.pdf]
- Untamed Theory Pen Test - March 2019 [https://releases.rancher.com/documents/security/pen-tests/2019/UntamedTheory-Rancher_SecurityAssessment-20190712_v5.pdf]

Please note that new reports are no longer shared or made publicly available.

## Rancher Security Advisories and CVEs

Rancher is committed to informing the community of security issues in our products. For the list of CVEs (Common Vulnerabilities and Exposures) for issues we have resolved, refer to this page.

## Kubernetes Security Best Practices

For recommendations on securing your Kubernetes cluster, refer to the Kubernetes Cluster Security Best Practices [https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/] guide.

## Rancher Security Best Practices

For recommendations on securing your Rancher Manager deployments, refer to the Rancher Security Best Practices guide.

## Rancher Kubernetes Distributions (K3s/RKE2) Self-Assessment and Hardening Guides

Rancher uses the following Kubernetes distributions:

- **RKE2** [https://documentation.suse.com/en-us/cloudnative/rke2/] is a fully conformant Kubernetes distribution that focuses on security and compliance within the U.S. Federal Government sector.
- **K3s** [https://documentation.suse.com/en-us/cloudnative/k3s/] is a fully conformant, lightweight Kubernetes distribution. It is easy to install, with half the memory requirement of upstream Kubernetes, all in a binary of less than 100 MB.

To harden a Kubernetes cluster that's running a distribution other than those listed, refer to your Kubernetes provider's docs.

### Hardening Guides and Benchmark Versions

Each self-assessment guide is accompanied by a hardening guide. These guides were tested alongside the listed Rancher releases. Each self-assessment guide was tested on a specific Kubernetes version and CIS benchmark version. If a CIS benchmark has not been validated for your Kubernetes version, you can use the existing guides until a guide for your version is added.

### RKE2 Guides

| Type | Kubernetes Version | CIS Benchmark Version | Self Assessment Guide | Hardening Guide |
|---|---|---|---|---|
| Standalone RKE2 | Kubernetes v1.27 | CIS v1.9 | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/cis_self_assessment19.html] | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/hardening_guide.html] |

| Type | Kubernetes Version | CIS Benchmark Version | Self Assessment Guide | Hardening Guide |
|---|---|---|---|---|
| Standalone RKE2 | Kubernetes v1.28 | CIS v1.10 | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/cis_self_assessment110.html] | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/hardening_guide.html] |
| Standalone RKE2 | Kubernetes v1.29 to v1.31 | CIS v1.11 | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/cis_self_assessment111.html] | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/hardening_guide.html] |
| Standalone RKE2 | Kubernetes v1.32 and above | CIS v1.12 | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/cis_self_assessment112.html] | Link [https://documentation.suse.com/cloudnative/rke2/latest/en/security/hardening_guide.html] |

### K3s Guides

| Type | Kubernetes Version | CIS Benchmark Version | Self Assessment Guide | Hardening Guide |
|---|---|---|---|---|
| Standalone K3s | Kubernetes v1.27 | CIS v1.9 | {url-k3s-self-assessment-1.9}[Link] | Link [https://documentation.suse.com/cloudnative/k3s/latest/en/security/hardening-guide.html] |
| Standalone K3s | Kubernetes v1.28 | CIS v1.10 | {url-k3s-self-assessment-1.10}[Link] | Link [https://documentation.suse.com/cloudnative/k3s/latest/en/security/hardening-guide.html] |
| Standalone K3s | Kubernetes v1.29 to v1.31 | CIS v1.11 | {url-k3s-self-assessment-1.11}[Link] | Link [https://documentation.suse.com/cloudnative/k3s/latest/en/security/hardening-guide.html] |
| Standalone K3s | Kubernetes v1.32 and above | CIS v1.12 | {url-k3s-self-assessment-1.12}[Link] | Link [https://documentation.suse.com/cloudnative/k3s/latest/en/security/hardening-guide.html] |

# Security Advisories and CVEs

Rancher is committed to informing the community of security issues in our products. Rancher will publish security advisories and CVEs (Common Vulnerabilities and Exposures) for issues we have resolved. New security advisories are also published in Rancher's GitHub security page [https://github.com/rancher/rancher/security/advisories].

| ID | Description | Date | Resolution |
|---|---|---|---|
| CVE-2026-88809 [https://github.com/rancher/rancher/security/advisories/GHSA-f9gf-wh3x-jx3v] | Fixed a security vulnerability in the Rancher meta proxy endpoint where the destination allowlist could not prevent a request from being forwarded, allowing an authenticated user to reach internal services through the Rancher server. The allowlist is now enforced in the request handler. | 29 October 2026 | Rancher v2.15.3, v2.14.7, v2.13.11, v2.12.15, v2.11.19 |
| CVE-2026-88810 [https://github.com/rancher/webhook/security/advisories/GHSA-f898-2c8h-r7jc] | Fixed a security vulnerability in Rancher's RBAC validation where the `inheritedNamespacedRules` field of a `GlobalRole` was not checked for privilege escalation, allowing a user with delegated global role management permissions to grant themselves permissions in downstream clusters beyond those they already held. | 29 October 2026 | Rancher v2.15.3 |
| CVE-2026-88811 [https://github.com/rancher/rancher/security/advisories/GHSA-x82x-2qj6-5rm2] | Fixed a security vulnerability in the Rancher Manager meta proxy where the caller's own Rancher credentials were not consistently removed from requests forwarded to third-party cloud provider hosts, allowing a signed-in user's session credential or an API client's Rancher token to be disclosed to the receiving host. The proxy now strips the caller's credentials on every request path. | 29 October 2026 | Rancher v2.15.3, v2.14.7, v2.13.11, v2.12.15, v2.11.19 |
| CVE-2026-104623 [https://github.com/rancher/rancher/security/advisories/GHSA-8x6f-2v58-x6p7] | Fixed a security vulnerability in Rancher where the authorization check protecting the local cluster shell was not applied to every API route that could reach it, allowing any authenticated user, including users with no permissions on the `local` cluster, to obtain an unprivileged shell pod on the management cluster. The authorization check is now enforced consistently for all request forms. | 29 October 2026 | Rancher v2.15.3, v2.14.7, v2.13.11, v2.12.15, v2.11.19 |
| CVE-2026-94290 [https://github.com/rancher/fleet/security/advisories/GHSA-qfmw-gw96-ffcr] | Fixed a privilege escalation vulnerability in Fleet's agent management controller where a Fleet cluster object created by an unprivileged user could reference the Rancher management cluster's own administrative credentials, allowing escalation to cluster-admin on the local cluster. | 29 October 2026 | Rancher v2.15.3, v2.14.7, v2.13.11, and v2.12.15 with Fleet v0.16.3, v0.15.8, v0.14.12, and v0.13.17 |
| CVE-2026-104615 [https://github.com/rancher/fleet/security/advisories/GHSA-gx8v-2929-9q95] | Fixed a cluster identity takeover vulnerability in Fleet where a registration request could claim an existing cluster's identity using a different registration token, granting access to that cluster's Fleet management namespace. | 29 October 2026 | Rancher v2.15.3, v2.14.7, v2.13.11, and v2.12.15 with Fleet v0.16.3, v0.15.8, v0.14.12, and v0.13.17 |
| CVE-2026-88808 [https://github.com/rancher/fleet/security/advisories/GHSA-q9v4-358v-r8q5] | Fixed a vulnerability in Fleet where bundle `downstreamResources` were copied to downstream clusters with the agent's privileged credentials, allowing a tenant to write Secrets and ConfigMaps, and create namespaces, outside of their authorization scope. | 23 September 2026 | Rancher v2.15.2, v2.14.6, v2.13.10 |
| CVE-2026-88805 [https://github.com/rancher/rancher/security/advisories/GHSA-6vpq-mf48-9794] | Fixed a security vulnerability in Rancher where logging out cleared the browser session cookies but did not delete the corresponding public API session token on the server, so a previously captured session token remained valid until its natural expiry. Only deployments using public API session tokens were affected. | 23 September 2026 | Rancher v2.15.2 |

| ID | Description | Date | Resolution |
|---|---|---|---|
| CVE-2026-88804 [https://github.com/rancher/rancher/security/advisories/GHSA-992f-xh8r-jg2f] | Fixed a security vulnerability in Rancher where public UI settings exposed on unauthenticated read-only API routes could be modified by a remote unauthenticated user, allowing content to be stored that executes in the browser of users opening the Rancher login page and potentially exposing the local administrator bootstrap password or an active administrator session. The unauthenticated settings routes are now strictly read-only. | 23 September 2026 | Rancher v2.15.2, v2.14.6, v2.13.10, v2.12.14, v2.11.18 |
| CVE-2026-93540 [https://github.com/rancher/fleet/security/advisories/GHSA-m93g-8438-2cgg] | Fixed a privilege management vulnerability in Fleet where namespace labels and annotations were applied without the bundle service account's authorization, allowing unauthorized changes to namespace metadata in multi-tenant deployments. | 23 September 2026 | Rancher v2.15.1, v2.14.6, v2.13.10, v2.12.14 |
| CVE-2026-93539 [https://github.com/rancher/fleet/security/advisories/GHSA-8vfv-33cg-g75q] | Fixed a vulnerability in Fleet's Git webhook receiver that allowed unauthenticated, cross-namespace modification of the `GitRepo` polling interval when no webhook secret was configured. | 23 September 2026 | Rancher v2.15.2 |
| CVE-2026-93538 [https://github.com/rancher/fleet/security/advisories/GHSA-h9p5-fp5h-qpqr] | Fixed a cross-tenant authorization issue in Fleet where labels supplied during agent-initiated cluster registration could cause a cluster to match another tenant's bundle targets, disclosing that tenant's manifests and resolved values Secrets. | 23 September 2026 | Rancher v2.15.1, v2.14.5, v2.13.9, v2.12.13, v2.11.17 |
| CVE-2026-93537 [https://github.com/rancher/fleet/security/advisories/GHSA-wpfm-r97v-3j4h] | Fixed a path traversal vulnerability in Fleet where untrusted bundle content could reference Helm `valuesFiles` entries outside the bundle directory, disclosing their contents through the generated Bundle. | 23 September 2026 | Rancher v2.15.2, v2.14.6, v2.13.10, v2.12.14, v2.11.18 |
| CVE-2026-75036 [https://github.com/rancher/fleet/security/advisories/GHSA-x9m6-xcjr-hpjp] | Fixed a vulnerability in Fleet's Helm template preprocessing where untrusted bundle content could cause the Fleet controller to disclose cluster metadata and information about hosts reachable from the controller. | 27 August 2026 | Rancher v2.15.1, v2.14.5, v2.13.9, v2.12.13, v2.11.17 |
| CVE-2026-75035 [https://github.com/rancher/rancher/security/advisories/GHSA-h923-rr66-g2w5] | Fixed a security vulnerability in the Rancher imperative ext Token API (`ext.cattle.io/v1` tokens) where any authenticated user can list and watch other users' tokens by supplying a crafted label selector, disclosing token metadata and the stored bearer token hash. | 27 August 2026 | Rancher v2.15.1, Rancher v2.14.5, Rancher v2.13.9 |
| CVE-2026-75034 [https://github.com/rancher/rancher/security/advisories/GHSA-j637-8xgr-436x] | Fixed a security vulnerability in Rancher's SAML authentication handler where the assertion replay protection was ineffective in high-availability (multi-replica) deployments, allowing a captured SAML assertion to be replayed against other replicas to create additional authenticated sessions as the victim. All SAML providers were affected. | 27 August 2026 | Rancher v2.15.1, Rancher v2.14.5, Rancher v2.13.9 |
| CVE-2026-75033 [https://github.com/rancher/rancher/security/advisories/GHSA-5hf4-f4mp-g6h4] | Fixed a security vulnerability in Rancher's project-scoped `secretsController` where Project Secrets were propagated to namespaces based only on the `field.cattle.io/projectId` annotation, without validating that the referenced project belonged to the same cluster. A user with namespace-create permissions on one downstream cluster could spoof this annotation to leak Project Secrets from a project in a different cluster they were not authorized to access. | 27 August 2026 | Rancher v2.15.1, Rancher v2.14.5, Rancher v2.13.9, Rancher v2.12.13 |

| ID | Description | Date | Resolution |
|---|---|---|---|
| CVE-2026-71404 [https://github.com/rancher/rancher/security/advisories/GHSA-92jp-phmj-wv65] | Fixed a security vulnerability in Rancher's GlobalRole controller where a user with delegated `create` or `update` permissions for `GlobalRole` resources could set the `authz.management.cattle.io/cr-name` annotation to target an arbitrary existing `ClusterRole` (such as the built-in `cluster-admin`). The controller then overwrites the rules without an ownership check, stripping permissions from all bound principals and causing a persistent, cluster-wide RBAC lockout on the local management cluster. | 27 August 2026 | Rancher v2.15.1, Rancher v2.14.5, Rancher v2.13.9 |
| CVE-2026-71403 [https://github.com/rancher/rancher/security/advisories/GHSA-wfvm-w99r-cjgc] | Fixed a security vulnerability in the SUSE Rancher Norman `/v3/users` API where the update path did not enforce immutability of the `username`, `principalIds`, and `displayName` fields on a `User` resource. A caller holding `update` permissions on `users` could rebind a user's external principal identity, enabling account takeover during the next external login or creating a denial-of-service condition that left existing role bindings inaccessible. | 27 August 2026 | Rancher v2.15.1, Rancher v2.14.5, Rancher v2.13.9 |

# SUSE Rancher Prime Security Best Practices

## Restrict Public Access to /version and /rancherversion Path

The upstream (local) Rancher instance provides information about the Rancher version it is running and the Go version that was used to build it. That information is accessible via the `/version` path, which is used for tasks such as automating version bumps, or confirming that a deployment was successful. The upstream instance also provides Rancher version information accessible via the `/rancherversion` path.

Adversaries can misuse this information to identify the running Rancher version and cross-relate it with potential bugs to exploit. If your upstream Rancher instance is publicly available on the web, use a Layer 7 firewall to block `/version` and `/rancherversion`.

See OWASP Web Application Security Testing - Enumerate Infrastructure and Application Admin Interfaces [https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/05-Enumerate_Infrastructure_and_Application_Admin_Interfaces.html] for more information on protecting your server.

## Session Management

Some environments may require additional security controls for session management. For example, you may want to limit users' concurrent active sessions or restrict which geolocations those sessions can be initiated from. Such features are not supported by Rancher out of the box.

If you require such features, combine Layer 7 firewalls with external authentication providers.

## Use External Load Balancers to Protect Vulnerable Ports

You should protect the following ports behind an external load balancer that has SSL offload enabled:

- **K3s:** Port 6443, used by the Kubernetes API.
- **RKE2:** Port 6443, used by the Kubernetes API, and port 9345, used for node registration.

These ports have TLS SAN certificates which list nodes' public IP addresses. An attacker could use that information to gain unauthorized access or monitor activity on the cluster. Protecting these ports helps mitigate against nodes' public IP addresses being disclosed to potential attackers.

#### Rancher Username Policy

By default, Kubernetes does not provide enforcement mechanisms for baseline username policies. In Rancher, this means that any enforcement of username formats, naming conventions, or baseline policies is expected to be handled by the external identity provider's policies, if such policies are in place.

In Rancher, `admin` is the default username for the Administrator user, as highlighted here

Examples of potential baseline policies include:

- Requiring usernames to follow an organizational convention (e.g.,`firstname.lastname`)
- Enforcing minimum or maximum length requirements
- Disallowing certain special characters
- Preventing impersonation by disallowing reserved names (e.g., `admin`, `root`)

Without these controls at the identity provider layer, there is a risk of inconsistent or insecure username practices, which can complicate access audits and lead to privilege escalation attempts.



Rancher currently enforces only a minimum password length.

**Recommendation:** We strongly advice that customers:

- Review and configure username baseline policies directly in their external identity providers (e.g., LDAP, Active Directory, SAML, or OIDC).
- Ensure that those policies align with the organization's security and compliance requirements.
- Regularly audit user accounts to detect naming inconsistencies or policy violations.

For more information, see:

- OWASP Cheat Sheet: Authentication [https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html]
- OWASP Identity Management [https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication]

#### WAF Rules

Rancher is designed to support a wide range of deployment scenarios, including environments where customers may programmatically automate the creation or provisioning of large numbers of clusters. Imposing strict application-level limits within Rancher itself could interfere with legitimate workloads that require dynamic scaling.

For example:

- CI/CD pipelines may create and tear down clusters frequently.
- Self-service portals could provision clusters on-demand for developers.
- Test environments may generate high volumes of API calls.

**Risk:** Without appropriate rate limiting, adversaries could exploit unauthenticated or authenticated endpoints to:

- Exhaust resources (Denial of Service).
- Inflate storage costs.
- Degrade performance for legitimate users.

**Recommendation:** The most effective way to mitigate this risk is to implement rate limiting and abuse protection at the infrastructure or Web Application Firewall (WAF) layer. This approach allows thresholds to be tuned for each environment's expected usage and scaling characteristics. Some examples of controls can be:

- Configuring a Web Application Firewall or API Gateway to enforce rate limits on sensitive operations, such as cluster creation and provisioning.
- Defining thresholds based on baseline workload expectations (e.g., max requests per minute per client).
- Monitoring logs and alerting on anomalies to detect potential abuse.
- Apply a resource quota, which is a Rancher feature that limits the resources available to a project or namespace.

For more information, see:

- Project Resource Quotas
- OWASP API Security Top 10 - API4:2019 - Lack of Resources & Rate Limiting [https://owasp.org/API-Security/editions/2023/en/0xa4-lack-of-resources-rate-limiting/]
- OWASP Cheat Sheet: Rate Limiting [https://cheatsheetseries.owasp.org/cheatsheets/Rate_Limiting_Cheat_Sheet.html]

**Rancher Username Policy**

By default, Kubernetes does not provide enforcement mechanisms for baseline username policies. In Rancher, this means that any enforcement of username formats, naming conventions, or baseline policies is expected to be handled by the external identity provider's policies, if such policies are in place.

In Rancher, `admin` is the default username for the Administrator user, as highlighted here

Examples of potential baseline policies include:

- Requiring usernames to follow an organizational convention (e.g.,`firstname.lastname`)
- Enforcing minimum or maximum length requirements
- Disallowing certain special characters
- Preventing impersonation by disallowing reserved names (e.g., `admin`, `root`)

Without these controls at the identity provider layer, there is a risk of inconsistent or insecure username practices, which can complicate access audits and lead to privilege escalation attempts.



Rancher currently enforces only a minimum password length.

**Recommendation:** We strongly advice that customers:

- Review and configure username baseline policies directly in their external identity providers (e.g., LDAP, Active Directory, SAML, or OIDC).
- Ensure that those policies align with the organization's security and compliance requirements.
- Regularly audit user accounts to detect naming inconsistencies or policy violations.

For more information, see:

- OWASP Cheat Sheet: Authentication [https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html]
- OWASP Identity Management [https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication]

**WAF Rules**

Rancher is designed to support a wide range of deployment scenarios, including environments where customers may programmatically automate the creation or provisioning of large numbers of clusters. Imposing strict application-level limits within Rancher itself could interfere with legitimate workloads that require dynamic scaling.

For example:

- CI/CD pipelines may create and tear down clusters frequently.
- Self-service portals could provision clusters on-demand for developers.
- Test environments may generate high volumes of API calls.

**Risk:** Without appropriate rate limiting, adversaries could exploit unauthenticated or authenticated endpoints to:

- Exhaust resources (Denial of Service).
- Inflate storage costs.
- Degrade performance for legitimate users.

**Recommendation:** The most effective way to mitigate this risk is to implement rate limiting and abuse protection at the infrastructure or Web Application Firewall (WAF) layer. This approach allows thresholds to be tuned for each environment's expected usage and scaling characteristics. Some examples of controls can be:

- Configuring a Web Application Firewall or API Gateway to enforce rate limits on sensitive operations, such as cluster creation and provisioning.
- Defining thresholds based on baseline workload expectations (e.g., max requests per minute per client).
- Monitoring logs and alerting on anomalies to detect potential abuse.
- Apply a resource quota, which is a Rancher feature that limits the resources available to a project or namespace.

For more information, see:

- Project Resource Quotas
- OWASP API Security Top 10 - API4:2019 - Lack of Resources & Rate Limiting [https://owasp.org/API-Security/editions/2023/en/0xa4-lack-of-resources-rate-limiting/]

- OWASP Cheat Sheet: Rate Limiting [https://cheatsheetseries.owasp.org/cheatsheets/Rate_Limiting_Cheat_Sheet.html]

**Restrict access to user-data**

To launch clusters on infrastructure providers, Rancher relies on cloud-init user-data to provide the necessary information to bootstrap each instance. User-data is a parameter defined on the cloud provider for each instance on creation.

Some cloud infrastructure providers expose this user-data to the instance through the "IMDS" (instance metadata service), accessible through a well-known link-local IP address.

Processes in the instance, including pods, can access this metadata depending on the provider and configuration.

In addition, some providers also expose this user-data to provider operators with sufficient access to read information about the instance through the provider console or cli.

Some examples of controls to restrict access to user-data are:

- Define Network Policies to limit access from pods to IMDS.
- On AWS EC2, require IMDSv2 [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-IMDS-new-instances.html] and set a hop limit of `1`, to limit access from pods to IMDS. Note that this does not prevent access from pods with `hostNetwork`.
- Define firewall rules to limit access from processes to IMDS.
- Restrict read access to instance information on the cloud provider API, which can include user-data.

Refer to the provider documentation for further information on how to limit access to IMDS, e.g. on AWS [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-metadata-limiting-access.html].

# SUSE Rancher Prime Webhook

Rancher-Webhook is an essential component of Rancher that works in conjunction with Kubernetes to enhance security and enable critical features for Rancher-managed clusters.

It integrates with Kubernetes' extensible admission controllers, as described in the Kubernetes documentation [https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-controllers/], which allows Rancher-Webhook to inspect specific requests sent to the Kubernetes API server, and add custom validations and mutations to the requests that are specific to Rancher. Rancher-Webhook manages the resources to be validated using the `rancher.cattle.io ValidatingWebhookConfiguration` and the `rancher.cattle.io MutatingWebhookConfiguration` objects, and will override any manual edits.

Rancher deploys Rancher-Webhook as a separate deployment and service in both local and downstream clusters. Rancher manages Rancher-Webhook using Helm. It's important to note that Rancher may override modifications made by users to the Helm release. To safely modify these values see Customizing Rancher-Webhook Configuration.

Each Rancher version is designed to be compatible with a single version of the webhook. The compatible versions are provided below for convenience.



Rancher manages deployment and upgrade of the webhook. Under most circumstances,

> no user intervention should be needed to ensure that the webhook version is compatible with the version of Rancher that you are running.

| Rancher Version | Webhook Version | Availability in Prime | Availability in Community |
|---|---|---|---|
| v2.15.3 | v0.11.4 | ⬕ | ⬕ |
| v2.15.2 | v0.11.3 | ⬕ | ⬕ |
| v2.15.1 | v0.11.1 | ⬕ | ⬕ |
| v2.15.0 | v0.11.0 | ✗ | ⬕ |

## Why Do We Need It?

Rancher-Webhook is crucial for Rancher to protect clusters against malicious attacks and enable various features. Rancher relies on the Rancher-Webhook as an integral part of its functionality. Without the webhook, Rancher would not be a complete product. It provides essential protection for Rancher-managed clusters, preventing security vulnerabilities and ensuring the consistency and stability of the cluster.

## What Resources Does the Webhook Validate?

You can find an in-progress list of the resources that the webhook validates in the webhook's repo [https://github.com/rancher/webhook/blob/release/v0.4/docs.md]. These docs are organized by group/version and resource (top-level header is group/version, next level header is resource). Checks specific to one version can be found by viewing the `docs.md` file associated with a particular tag (note that webhook versions prior to `v0.3.6` won't have this file).

## Bypassing the Webhook

Sometimes, you must bypass Rancher's webhook validation to perform emergency restore operations or fix other critical issues. The bypass operation is exhaustive, meaning no webhook validations or mutations apply when you use it. It is not possible to bypass some validations or mutations and have others still apply - they are either all bypassed or all active.



> Rancher's webhook provides critical security protections. Bypassing the webhook should only be done by administrators in specific scenarios, after all other options have been exhausted. In addition, permission to bypass the webhook should be carefully controlled, and never given to users who are not admins.

To bypass the webhook, impersonate both the `rancher-webhook-sudo` service account and the `system:masters` group (both are required):

```
kubectl create -f example.yaml --as=system:serviceaccount:cattle-system:rancher-webhook-sudo --as-group=system:masters
```

## Customizing Rancher-Webhook Configuration

You can add custom Helm values when you install Rancher-Webhook via Helm. During a Helm install of the Rancher-Webhook chart, Rancher checks for custom Helm values. These custom values must be defined in a ConfigMap named `rancher-config`, in the `cattle-system` namespace, under the data key, `rancher-webhook`. The value of this key must be valid YAML.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rancher-config
  namespace: cattle-system
  labels:
    app.kubernetes.io/part-of: "rancher"
data:
  rancher-webhook: '{"port": 9553, "priorityClassName": "system-node-
critical"}'
```

Rancher redeploys the Rancher-Webhook chart when changes to the ConfigMap values are detected.

#### Customizing Rancher-Webhook During Rancher Installation

When you use Helm to install the Rancher chart, you can add custom Helm values to the Rancher-Webhook of the local cluster. All values in the Rancher-Webhook chart are accessible as nested variables under the `webhook` name.

These values are synced to the `rancher-config` ConfigMap during installation.

```
helm install rancher rancher-prime/rancher \
  --namespace cattle-system \
  ...
  --set webhook.port=9553 \
  --set webhook.priorityClassName="system-node-critical"
```

### Common Issues

#### EKS Cluster with Calico CNI

Users running an EKS cluster with Calico CNI may run into errors when the Kubernetes API server attempts to contact the Rancher-Webhook. One workaround for this issue, as documented by Calico [https://docs.tigera.io/calico/latest/getting-started/kubernetes/managed-public-cloud/eks#install-eks-with-calico-networking] involves setting `hostNetwork=true` for the webhook deployment. You can change this value by adding the Helm value `global.hostNetwork=true` to the `rancher-config` ConfigMap. See Customizing Rancher-Webhook Configuration for more info.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: rancher-config
  namespace: cattle-system
  labels:
    app.kubernetes.io/part-of: "rancher"
```

```
data:
  rancher-webhook: '{"global": {"hostNetwork": true}}'
```



This temporary workaround may violate an environment's security policy. This workaround also requires that port 9443 is unused on the host network.



By default, Helm stores information as secrets. Secrets are a resource that some webhook versions validate. In these cases, directly update the deployment with the `hostNetwork=true` value using kubectl, then update the webhook configuration as specified above.

#### Private GKE Cluster

When using a private GKE cluster, errors may occur that prevent the Kubernetes API server from communicating with the webhook. The following error message may appear:

```
Internal error occurred: failed calling webhook
"rancher.cattle.io.namespaces.create-non-kubesystem": failed to call
webhook: Post "https://rancher-webhook.cattle-
system.svc:443/v1/webhook/validation/namespaces?timeout=10s": context
deadline exceeded
```

This issue occurs because firewall rules restrict communication between the API server and the private cluster. To resolve this communication problem, users must add firewall rules to allow the GKE control plane to communicate with the Rancher-Webhook on port 9443. Please refer to the GKE documentation [https://cloud.google.com/kubernetes-engine/docs/how-to/private-clusters#add_firewall_rules] for detailed information and steps on updating the firewall rules.

#### Application Fails to Deploy Due to rancher-webhook Blocking Access

The webhook provides extra validations on namespaces [https://github.com/rancher/webhook/blob/release/v0.4/docs.md#psa-label-validation]. One of these validations ensures that users can only update PSA relevant labels if they have the proper permissions (`updatepsa` for `projects` in `management.cattle.io`). This can result in specific operators, such as Tigera or Trident, failing when they attempt to deploy namespaces with PSA labels. There are several ways to resolve this issue:

- Configure the application to create a namespace with no PSA labels. If users wish to apply a PSA to these namespaces, they can add them to a project with the desired PSA after configuration. See the docs on PSS and PSA resources for instructions on how.
  - This is the preferred option, though not all applications can be configured in this fashion.
- Manually grant the operator permissions to manage PSAs for namespaces.
  - This option will introduce security risks, since the operator will now be able to set the PSA for the namespaces it has access to. This could allow the operator to deploy a privileged pod, or effect cluster takeover through other means.
- A user account with the proper permissions can pre-create the namespace with the appropriate configuration.

- This option depends on the ability of the application to handle existing resources.

Another one of these validations ensures that the user has the proper permissions to update the `field.cattle.io/projectId` annotation on a namespace. This is the `manage-namespaces` permission for `projects` in `management.cattle.io`.

## Issues on Specific Versions



The following is an incomplete list of high-severity issues affecting specific Rancher/webhook versions. In most cases, these issues can be resolved by upgrading to a more recent Rancher version.

### Incompatible Webhook Version on Rollback



This affects rolling back to Rancher v2.7.5 or earlier.

If you roll back to Rancher v2.7.5 or earlier, you may see webhook versions that are too recent to be compatible with downstream clusters running pre-v2.7.5 version of Rancher. This may cause various incompatibility issues. For example, project members may be unable to create namespaces. In addition, when you roll back to versions before the webhook was installed in downstream clusters, the webhook may remain installed, which can result in similar incompatibility issues.

To help alleviate these issues, you can run the adjust-downstream-webhook [https://github.com/rancherlabs/support-tools/tree/master/adjust-downstream-webhook] shell script after roll back. This script selects and installs the proper webhook version (or removes the webhook entirely) for the corresponding Rancher version.

## Rotation of Expired Webhook Certificates

For Rancher versions that have `rancher-webhook` installed, certain versions created certificates that will expire after one year. It will be necessary for you to rotate your webhook certificate if the certificate did not renew.

In Rancher v2.6.3 and up, rancher-webhook deployments will automatically renew their TLS certificate when it is within 30 or fewer days of its expiration date. If you are using v2.6.2 or below, there are two methods to work around this issue:

### 1. Users with Cluster Access, Run the Following Commands:

```
kubectl delete secret -n cattle-system cattle-webhook-tls
kubectl delete
mutatingwebhookconfigurations.admissionregistration.k8s.io --ignore-not
-found=true rancher.cattle.io
kubectl delete pod -n cattle-system -l app=rancher-webhook
```

### 2. Users with No Cluster Access Via `kubectl`:

1. Delete the `cattle-webhook-tls` secret in the `cattle-system` namespace in the local cluster.
2. Delete the `rancher.cattle.io` mutating webhook

3. Delete the `rancher-webhook` pod in the `cattle-system` namespace in the local cluster.



The webhook certificate expiration issue is not specific to `cattle-webhook-tls` as listed in the examples. You will fill in your expired certificate secret accordingly.

## Hardening the SUSE Rancher Prime Webhook

Rancher Webhook is an important component within Rancher, playing a role in enforcing security requirements for Rancher and its workloads. To decrease its attack surface, access to it should be limited to the only valid caller it has: the Kubernetes API server. This can be done by using network policies and authentication independently or in conjunction with each other to harden the webhook against attacks.

### Block External Traffic Using Network Policies

The webhook is only expected to accept requests from the Kubernetes API server. By default, however, the webhook can accept traffic from any source. If you are using a CNI that supports Network Policies, you can create a policy that blocks traffic that doesn't originate from the API server.

The built-in NetworkPolicy resource in Kubernetes can't block or admit traffic from the cluster hosts, and the `kube-apiserver` process is always running on the host network. Therefore, you must use the advanced network policy resources from the CNI in use. Examples for Calico and Cilium follow. Consult the documentation for your CNI for more details.

#### Calico

Use the NetworkPolicy resource in the `crd.projectcalico.org/v1` API group. Use the selector `app == 'rancher-webhook'` to create a rule for the webhook, and set the CIDR of the control plane hosts as the ingress source:

```
apiVersion: crd.projectcalico.org/v1
kind: NetworkPolicy
metadata:
  name: allow-k8s
  namespace: cattle-system
spec:
  selector: app == 'rancher-webhook'
  types:
    - Ingress
  ingress:
    - action: Allow
      protocol: TCP
      source:
        nets:
        - 192.168.42.0/24 # CIDR of the control plane host. May list
more than 1 if the hosts are in different subnets.
      destination:
        selector:
          app == 'rancher-webhook'
```

#### Cilium

Use the CiliumNetworkPolicy resource in the `cilium.io/v2` API group. Add the `host` and `remote-node` keys to the `fromEntities` ingress rule. This blocks in-cluster and external traffic while allowing traffic from the hosts.

```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: allow-k8s
  namespace: cattle-system
spec:
  endpointSelector:
    matchLabels:
      app: rancher-webhook
  ingress:
    - fromEntities:
      - host
      - remote-node
```

### Require the Kubernetes API Server to Authenticate to the Webhook

The webhook should only accept requests from the Kubernetes API server. By default, the webhook doesn't require clients to authenticate to it. It will accept any request. You can configure the webhook to require credentials so that only the API server can access it. More information can be found in the Kubernetes documentation [https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-controllers/#authenticate-apiservers].

1. Configure the API server to present a client certificate to the webhook, pointing to an AdmissionConfiguration file to configure the ValidatingAdmissionWebhook and MutatingAdmissionWebhook plugins:

   ```
   # /etc/rancher/admission/admission.yaml
   apiVersion: apiserver.config.k8s.io/v1
   kind: AdmissionConfiguration
   plugins:
   - name: ValidatingAdmissionWebhook
     configuration:
       apiVersion: apiserver.config.k8s.io/v1
       kind: WebhookAdmissionConfiguration
       kubeConfigFile: "/etc/rancher/admission/kubeconfig"
   - name: MutatingAdmissionWebhook
     configuration:
       apiVersion: apiserver.config.k8s.io/v1
       kind: WebhookAdmissionConfiguration
       kubeConfigFile: "/etc/rancher/admission/kubeconfig"
   ```

This is also the same config file where other admission plugins are configured, such as PodSecurity. If your distro or your setup uses additional admission plugins, configure those as well. For example, add RKE2's PodSecurity configuration [https://documentation.suse.com/cloudnative/rke2/latest/en/security/pod_security_standards.html] to this file.

2. Create the kubeconfig file that the admission plugins refer to. Rancher Webhook only supports client certificate authentication, so generate a TLS key pair, and set the kubeconfig to use either `client-certificate` and `client-key` or `client-certificate-data` and `client-key-data`. For example:

```
# /etc/rancher/admission/kubeconfig
apiVersion: v1
kind: Config
users:
- name: 'rancher-webhook.cattle-system.svc'
  user:
    client-certificate: /path/to/client/cert
    client-key: /path/to/client/key
```

3. Start the kube-apiserver binary with the flag `--admission-control-config-file` pointing to your AdmissionConfiguration file. The way to do this varies by distro, and it isn't supported universally, such as in hosted Kubernetes providers. Consult the documentation for your Kubernetes distribution.

   For RKE2, `rke2-server` can be started with a config file like so:

```
# /etc/rancher/rke2/config.yaml
kube-apiserver-arg:
- admission-control-config-
file=/etc/rancher/admission/admission.yaml
kube-apiserver-extra-mount:
- /etc/rancher/admission:/etc/rancher/admission:ro
```



Some distros set this flag by default. If your distro provisions its own AdmissionConfiguration, you must include it in your custom admission control config file. For example, RKE2 installs an AdmissionConfiguration file at `/etc/rancher/rke2/rke2-pss.yaml`, which configures the PodSecurity admission plugin. Setting `admission-control-config-file` in config.yaml will override this essential security setting. To include both plugins, consult the Default Pod Security Standards documentation [https://documentation.suse.com/cloudnative/rke2/latest/en/security/pod_security_standards.html] and copy the appropriate plugin configuration to your admission.yaml.

4. If you're using Rancher to provision your cluster using existing nodes, create these files on the node before you provision them.

   If you're using Rancher to provision your cluster on new nodes, allow the provisioning to complete, then use the provided SSH key and IP address to connect to the nodes, and place the RKE2 config file in the

`/etc/rancher/rke2/config.yaml.d/` directory.

5. After the cluster is configured with these credentials, configure the Rancher cluster agent to enable authentication in the webhook. Create a file containing these chart values:

```
# values.yaml
auth:
  clientCA: <base64-encoded certificate authority that signed the kube-apiserver's client certificates>
  allowedCNs:
  - <list of Common Names identifying the kube-apiserver's client certificates.>
  - <if not provided, any cert signed by the given CA will be accepted.>
```

6. Create a configmap in the `cattle-system` namespace on the provisioned cluster with these values:

```
kubectl --namespace cattle-system create configmap rancher-config --from-file=rancher-webhook=values.yaml
```

The webhook will restart with these values.

## SELinux RPM

Security-Enhanced Linux (SELinux) [https://en.wikipedia.org/wiki/Security-Enhanced_Linux] is a security enhancement to Linux.

Developed by Red Hat, it is an implementation of mandatory access controls (MAC) on Linux. Mandatory access controls allow an administrator of a system to define how applications and users can access different resources such as files, devices, networks and inter-process communication. SELinux also enhances security by making an OS restrictive by default.

After being historically used by government agencies, SELinux is now an industry standard and enabled by default on SUSE distributions such as SUSE Linux Enterprise 16, openSUSE distributions such as openSUSE Tumbleweed, and RHEL-based OS such as Rocky Linux. To check whether SELinux is enabled and enforced on your system, use `getenforce`:

```
# getenforce
Enforcing
```

We provide three RPMs (RPM Package Manager) that enable Rancher products to function properly on SELinux-enforcing hosts:

- `rancher-selinux`
- `rke2-selinux`

- k3s-selinux

## About rancher-selinux

To allow Rancher to work with SELinux, some functionality has to be manually enabled for the SELinux nodes. To help with that, Rancher provides an SELinux RPM.



*Why SELinux?*

By assigning a dedicated SELinux type to each container, we ensure that containers are limited to their minimal needs and cannot pivot to other resources if compromised.

The `rancher-selinux` RPM contains a set of SELinux policies designed to grant the necessary privileges to various Rancher components running on Linux systems with SELinux enabled.

Source code, policy definitions, and development guides are available in the rancher-selinux GitHub repository [https://github.com/rancher/rancher-selinux].

### Supported Operating Systems

*Table 9. Supported operating systems and repository paths*

| Operating System | Version | Repo path |
|---|---|---|
| openSUSE Leap / SLE / SLE Micro | 16 / 6 | `sle/16` |
| openSUSE MicroOS / Tumbleweed | Rolling | `microos` |
| RHEL / CentOS / Rocky Linux | 9 | `centos/9` |
| RHEL / CentOS / Rocky Linux | 10 | `centos/10` |

All policies are covered by end-to-end tests. For the most up-to-date compatibility details, see the rancher-selinux support matrix [https://github.com/rancher/rancher-selinux#support-matrix].

### Installing the rancher-selinux RPM

Install `rancher-selinux` on all SELinux-enabled nodes in the cluster. Replace `<repo-path>` with the corresponding path value from Supported operating systems and repository paths.

#### openSUSE Leap, SLE, SLE Micro, and MicroOS

```
cat << EOF > /etc/zypp/repos.d/rancher.repo
[rancher]
name=Rancher
baseurl=https://rpm.rancher.io/rancher/production/<repo-path>/noarch
enabled=1
gpgcheck=1
gpgkey=https://rpm.rancher.io/public.key
EOF

zypper --gpg-auto-import-keys refresh
zypper install -y rancher-selinux
```

On transactional systems such as SLE Micro and openSUSE MicroOS, replace `zypper install` with:

```
transactional-update pkg install rancher-selinux
reboot
```

#### RHEL, CentOS, and Rocky Linux

```
cat << EOF > /etc/yum.repos.d/rancher.repo
[rancher]
name=Rancher
baseurl=https://rpm.rancher.io/rancher/production/<repo-path>/noarch
enabled=1
gpgcheck=1
gpgkey=https://rpm.rancher.io/public.key
EOF

dnf install -y rancher-selinux
```

#### Verifying the installation

Verify that the Rancher SELinux policy modules are installed:

```
semodule -l | grep rancher
```

### Configuring Applications to Work with SELinux

Installing the `rancher-selinux` RPM on the host does not automatically confine applications. Applications must be explicitly configured to run in an allowed SELinux container domain provided by the RPM.

To configure supported Helm charts to be SELinux aware, set `global.seLinux.enabled` to `true` in the `values.yaml` file when installing or upgrading the chart.

*Table 10. Dedicated SELinux types by chart*

| Chart | Container | SELinux type |
|---|---|---|
| Rancher Monitoring | node-exporter | `prom_node_exporter_t` |
| Rancher Monitoring | pushprox | `rke_kubereader_t` |
| Rancher Logging | fluentbit | `rke_logreader_t` |
| Rancher AI | rancher-ai-agent | `rancher_aiagent_container_t` |
| Rancher AI | rancher-ai-mcp | `rancher_aimcp_container_t` |



Only the components listed in Dedicated SELinux types by chart receive a dedicated SELinux type. Other Rancher components and general workloads use the default `container_t` domain provided by `container-selinux`.

### About rke2-selinux

`rke2-selinux` provides policies for RKE2. It is installed automatically when the RKE2 installer script detects that it is running on an RPM-based distro.

The `rke2-selinux` GitHub repository is here. [https://github.com/rancher/rke2-selinux]

For more information about installing RKE2 on SELinux-enabled hosts, see the RKE2 documentation. [https://documentation.suse.com/cloudnative/rke2/latest/en/install/methods.html#rpm]

### About k3s-selinux

`k3s-selinux` provides policies for K3s. It is installed automatically when the K3s installer script detects that it is running on an RPM-based distro.

The `k3s-selinux` GitHub repository is here [https://github.com/rancher/k3s-selinux].

For more information about installing K3s on SELinux-enabled hosts, see the K3s documentation [https://documentation.suse.com/cloudnative/k3s/latest/en/advanced.html#_selinux_support].

## Compliance Scans

Rancher can run a security scan to check whether a cluster is deployed according to security best practices as defined in Kubernetes security benchmarks, such as the ones provided by STIG, BSI or CIS. The Compliance scans can run on any Kubernetes cluster, including hosted Kubernetes providers such as EKS, AKS, and GKE.

The `rancher-compliance` app leverages kube-bench, [https://github.com/aquasecurity/kube-bench] an open-source tool from Aqua Security, to check the compliance of clusters against Kubernetes Benchmarks. Also, to generate a cluster-wide report, the application utilizes Sonobuoy [https://github.com/vmware-tanzu/sonobuoy] for report aggregation.

### About the CIS Benchmark

The Center for Internet Security is a 501(c)(3) non-profit organization, formed in October 2000, with a mission to "identify, develop, validate, promote, and sustain best practice solutions for cyber defense and build and lead communities to enable an environment of trust in cyberspace". The organization is headquartered in East Greenbush, New York, with members including large corporations, government agencies, and academic institutions.

CIS Benchmarks are best practices for the secure configuration of a target system. CIS Benchmarks are developed through the generous volunteer efforts of subject matter experts, technology vendors, public and private community members, and the CIS Benchmark Development team.

Sign up [https://learn.cisecurity.org/benchmarks] at the CIS website to view the official Benchmark documents.

### About the Generated Report

Each scan generates a report that can be viewed in the Rancher UI and downloaded in CSV or XCCDF format.

By default, the CIS Benchmark v1.6 is used.

The Benchmark version is included in the generated report.

The Benchmark provides recommendations of two types: Automated and Manual. Recommendations marked as Manual in the Benchmark are not included in the generated report.

Some tests are designated as "Not Applicable." These tests will not be run on any CIS scan because of the way that Rancher provisions RKE2/K3s clusters. For information on how test results can be audited, and why some tests are designated to be not applicable, refer to Rancher's self-assessment guide for the corresponding Kubernetes version.

#### Report Formats

Compliance scan reports can be downloaded in two formats from the Rancher UI:

- **CSV:** A flat file containing all test results for the scan. This is the default download format.
- **XCCDF (XML):** A per-node report in XCCDF 1.2 [https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/xccdf] format, packaged as a `.zip` file containing one `.xml` file per node in the cluster. XCCDF output is compatible with compliance management tools such as STIG Manager [https://github.com/nuwcdivnpt/stig-manager] and is suitable for submission to security assessors.

XCCDF output is available for any benchmark format, including CIS and STIG. To include additional metadata in the XCCDF output, such as title, publisher, platform, or STIG-specific check and CCI identifiers, add an optional `metadata.yaml` file to the benchmark's ConfigMap. Refer to Compliance Scan Guides for details.

For information on how to download reports in either format, refer to View Reports.

The CSV report contains the following information:

| Column in Report | Description |
|---|---|
| `id` | The ID number of the CIS Benchmark. |
| `description` | The description of the CIS Benchmark test. |
| `remediation` | What needs to be fixed in order to pass the test. |
| `state` | Indicates if the test passed, failed, was skipped, or was not applicable. |
| `node_type` | The node role, which affects which tests are run on the node. Master tests are run on controlplane nodes, etcd tests are run on etcd nodes, and node tests are run on the worker nodes. |
| `audit` | This is the audit check that `kube-bench` runs for this test. |
| `audit_config` | Any configuration applicable to the audit script. |
| `test_info` | Test-related info as reported by `kube-bench`, if any. |
| `commands` | Test-related commands as reported by `kube-bench`, if any. |
| `config_commands` | Test-related configuration data as reported by `kube-bench`, if any. |
| `actual_value` | The test's actual value, present if reported by `kube-bench`. |
| `expected_result` | The test's expected result, present if reported by `kube-bench`. |

Refer to the table in the cluster hardening guide for information on which versions of Kubernetes, the Benchmark, Rancher, and our cluster hardening guide correspond to each other. Also refer to the hardening

guide for configuration files of CIS-compliant clusters and information on remediating failed tests.

### Test Profiles

The following profiles are available:

- Generic CIS 1.6
- Generic CIS 1.20
- Generic CIS 1.23
- RKE2 permissive 1.6
- RKE2 hardened 1.6
- RKE2 permissive 1.20
- RKE2 hardened 1.20
- RKE2 permissive 1.23
- RKE2 hardened 1.23
- K3s permissive 1.6
- K3s hardened 1.6
- K3s permissive 1.20
- K3s hardened 1.20
- K3s permissive 1.23
- K3s hardened 1.23
- AKS
- EKS
- GKE

You also have the ability to customize a profile by saving a set of tests to skip.

All profiles will have a set of not applicable tests that will be skipped during the CIS scan. These tests are not applicable based on how a RKE2/K3s cluster manages Kubernetes.

There are two types of RKE2/K3s cluster scan profiles:

- **Permissive:** This profile has a set of tests that have been will be skipped as these tests will fail on a default RKE2/K3s Kubernetes cluster. Besides the list of skipped tests, the profile will also not run the not applicable tests.
- **Hardened:** This profile will not skip any tests, except for the non-applicable tests.

The EKS and GKE cluster scan profiles are based on CIS Benchmark versions that are specific to those types of clusters.

In order to pass the "Hardened" profile, you will need to follow the steps on the hardening guide and use the `cluster.yml` defined in the hardening guide to provision a hardened cluster.

The default profile and the supported CIS benchmark version depends on the type of cluster that will be scanned:

The `rancher-compliance` supports the CIS 1.9 Benchmark version.

- For RKE2 Kubernetes clusters, the RKE2 Permissive 1.9 profile is the default.
- EKS and GKE have their own CIS Benchmarks published by `kube-bench`. The corresponding test profiles are used by default for those clusters.
- For cluster types other than RKE2, EKS and GKE, the Generic CIS 1.5 profile will be used by default.

## About Skipped and Not Applicable Tests

For now, only user-defined skipped tests are marked as skipped in the generated report.

Any skipped tests that are defined as being skipped by one of the default profiles are marked as not applicable.

## Roles-based Access Control

For information about permissions, refer to this page

## Configuration

For more information about configuring the custom resources for the scans, profiles, and benchmark versions, refer to this page

## How-to Guides

Please refer to the Compliance Scan Guides to learn how to run compliance scans.

## Roles-based Access Control

This section describes the permissions required to use the rancher-compliance App.

The rancher-compliance is a cluster-admin only feature by default.

However, the `rancher-compliance` chart installs these two default `ClusterRoles`:

- compliance-admin
- compliance-view

In Rancher, only cluster owners and global administrators have `compliance-admin` access by default.

### Cluster-Admin Access

Rancher Compliance Scans is a cluster-admin only feature by default. This means only the Rancher global admins, and the cluster's cluster-owner can:

- Install/Uninstall the rancher-compliance App
- See the navigation links for Compliance CRDs - ClusterScanBenchmarks, ClusterScanProfiles, ClusterScans
- List the default ClusterScanBenchmarks and ClusterScanProfiles
- Create/Edit/Delete new ClusterScanProfiles

- Create/Edit/Delete a new ClusterScan to run the Compliance scan on the cluster
- View and Download the ClusterScanReport created after the ClusterScan is complete

**Summary of Default Permissions for Kubernetes Default Roles**

The rancher-compliance creates three `ClusterRoles` and adds the Compliance Benchmark CRD access to the following default K8s `ClusterRoles`:

| ClusterRole created by chart | Default K8s ClusterRole | Permissions given with Role |
|---|---|---|
| `compliance-admin` | `admin` | Ability to CRUD clusterscanbenchmarks, clusterscanprofiles, clusterscans, clusterscanreports CR |
| `compliance-view` | `view` | Ability to List® clusterscanbenchmarks, clusterscanprofiles, clusterscans, clusterscanreports CR |

By default only cluster-owner role will have ability to manage and use `rancher-compliance` feature.

The other Rancher roles (cluster-member, project-owner, project-member) do not have any default permissions to manage and use rancher-compliance resources.

But if a cluster-owner wants to delegate access to other users, they can do so by creating ClusterRoleBindings between these users and the above Compliance ClusterRoles manually. There is no automatic role aggregation supported for the `rancher-compliance` ClusterRoles.

## Compliance Scan Guides

- Install rancher-compliance
- Uninstall rancher-compliance
- Run a Scan
- Run a Scan Periodically on a Schedule
- View Reports
- Enable Alerting for rancher-compliance
- Configure Alerts for Periodic Scan on a Schedule
- Create a Custom Compliance Version to Run

## Install SUSE Rancher Prime Compliance

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to install Compliance and click **Explore**.
3. In the left navigation bar, click **Apps > Charts**.
4. Click **Compliance**
5. Click **Install**.

**Result:** The compliance scan application is deployed on the Kubernetes cluster.

## Uninstall SUSE Rancher Prime Compliance

1. From the **Cluster Dashboard,** go to the left navigation bar and click **Apps > Installed Apps**.
2. Go to the `compliance-operator-system` namespace and check the boxes next to `rancher-compliance-crd` and `rancher-compliance`.
3. Click **Delete** and confirm **Delete**.

**Result:** The `rancher-compliance` application is uninstalled.

## Configuration

This configuration reference is intended to help you manage the custom resources created by the `rancher-compliance` application. These resources are used for performing compliance scans on a cluster, skipping tests, setting the test profile that will be used during a scan, and other customization.

To configure the custom resources, go to the **Cluster Dashboard** To configure the compliance scans,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to configure compliance scans and click **Explore**.
3. In the left navigation bar, click **Compliance**.

### Scans

A scan is created to trigger a compliance scan on the cluster based on the defined profile. A report is created after the scan is completed.

When configuring a scan, you need to define the name of the scan profile that will be used with the `scanProfileName` directive.

An example ClusterScan custom resource is below:

```
apiVersion: compliance.cattle.io/v1
kind: ClusterScan
metadata:
  name: scan-smnr9
spec:
  scanProfileName: cis-1.10-profile
```

### Profiles

A profile contains the configuration for the compliance scan, which includes the benchmark version to use and any specific tests to skip in that benchmark.



By default, a few ClusterScanProfiles are installed as part of the `rancher-compliance` chart. If a user edits these default benchmarks or profiles, the next chart update will reset them back. So it is advisable for users to not edit the default ClusterScanProfiles.

Users can clone the ClusterScanProfiles to create custom profiles.

Skipped tests are listed under the `skipTests` directive.

When you create a new profile, you will also need to give it a name.

An example `ClusterScanProfile` is below:

```
apiVersion: compliance.cattle.io/v1
kind: ClusterScanProfile
metadata:
  annotations:
    clusterscanprofile.compliance.cattle.io/builtin: 'true'
    meta.helm.sh/release-name: rancher-compliance
    meta.helm.sh/release-namespace: compliance-operator-system
  creationTimestamp: '2025-09-15T18:09:52Z'
  generation: 1
  labels:
    app.kubernetes.io/managed-by: Helm
  name: cis-1.10-profile
  resourceVersion: '93582'
  uid: 0baad187-1157-46ac-982d-014338847c27
spec:
  benchmarkVersion: cis-1.10
  skipTests:
    - '1.1.20'
    - '1.1.21'
```

**Benchmark Versions**

A benchmark version is the name of benchmark to run using `kube-bench`, as well as the valid configuration parameters for that benchmark.

A `ClusterScanBenchmark` defines the compliance `BenchmarkVersion` name and test configurations. The `BenchmarkVersion` name is a parameter provided to the `kube-bench` tool.

By default, a few `BenchmarkVersion` names and test configurations are packaged as part of the compliance scan application. When this feature is enabled, these default BenchmarkVersions will be automatically installed and available for users to create a ClusterScanProfile.



If the default BenchmarkVersions are edited, the next chart update will reset them back. Therefore we don't recommend editing the default ClusterScanBenchmarks.

A ClusterScanBenchmark consists of the fields:

- `Cluster provider`: This is the cluster provider name for which this benchmark is applicable. For example: RKE2, EKS, GKE, etc. Leave it empty if this benchmark can be run on any cluster type.
- `MinKubernetesVersion`: Specifies the cluster's minimum kubernetes version necessary to run this benchmark. Leave it empty if there is no dependency on a particular Kubernetes version.

- `MaxKubernetesVersion`: Specifies the cluster's maximum Kubernetes version necessary to run this benchmark. Leave it empty if there is no dependency on a particular k8s version.
- `customBenchmarkConfigMapName`: The name of the ConfigMap containing the custom benchmark config files. Required when creating a custom benchmark version.
- `customBenchmarkConfigMapNamespace`: The namespace where the custom benchmark ConfigMap is located. Required when creating a custom benchmark version.

An example `ClusterScanBenchmark` for a default benchmark is below:

```
apiVersion: compliance.cattle.io/v1
kind: ClusterScanBenchmark
metadata:
  annotations:
    meta.helm.sh/release-name: rancher-compliance
    meta.helm.sh/release-namespace: compliance-operator-system
  creationTimestamp: '2025-09-15T18:09:52Z'
  generation: 1
  labels:
    app.kubernetes.io/managed-by: Helm
  name: cis-1.10
  resourceVersion: '93569'
  uid: 309e543e-9102-4091-be91-08d7af7fb7a7
spec:
  clusterProvider: ''
  minKubernetesVersion: 1.28.0
```

The following example shows a `ClusterScanBenchmark` for a custom benchmark that references a ConfigMap:

```
apiVersion: compliance.cattle.io/v1
kind: ClusterScanBenchmark
metadata:
  name: rke2-cis-1.8-custom
spec:
  clusterProvider: rke2
  minKubernetesVersion: 1.28.0
  customBenchmarkConfigMapName: rke2-cis-1.8-custom
  customBenchmarkConfigMapNamespace: compliance-operator-system
```

#### XCCDF Metadata for Custom Benchmarks

When you download compliance scan reports in XCCDF format, you can enrich the output with benchmark-specific metadata by adding an optional `metadata.yaml` file to the custom benchmark ConfigMap. This file is mounted at the configuration root alongside `config.yaml`, and its contents are read when Rancher generates XCCDF reports. This applies to any benchmark format, including CIS and STIG.

If `metadata.yaml` is not present, the XCCDF output uses default placeholder values. For example, the title defaults to `Kubernetes CIS Benchmark <benchmarkVersion>`, and other fields default to `not-applicable`.

For instructions on adding `metadata.yaml` to a custom benchmark ConfigMap, refer to Creating a Custom Benchmark Version.

#### `metadata.yaml` Schema

The `metadata.yaml` file is a flat YAML file placed at the root of the custom benchmark directory. The following fields are supported:

| Field | Description |
|---|---|
| `title` | The benchmark title, used as the XCCDF `<title>` element. Defaults to `Kubernetes CIS Benchmark <benchmarkVersion>` if not set. |
| `description` | A description of the benchmark. |
| `publisher` | The name of the organization publishing the benchmark. |
| `contributor` | The name of the contributor. |
| `creator` | The name of the creator. |
| `source` | A URL or identifier for the source of the benchmark. |
| `platform` | A CPE identifier for the target platform, for example `cpe:/a:kubernetes:kubernetes`. |
| `notice` | Legal or usage notice text. |
| `noticeId` | An identifier for the notice element. |
| `frontMatter` | Introductory text to include before the benchmark content. |
| `rearMatter` | Closing text to include after the benchmark content. |
| `referenceHref` | A URL for the benchmark reference. |
| `referenceTitle` | The title of the benchmark reference. |
| `referenceType` | The type of the benchmark reference. |
| `referenceSubject` | The subject of the benchmark reference. |
| `referenceIdentifier` | An identifier for the benchmark reference. |
| `checkHref` | A URL for the check content reference. |
| `checkName` | The name of the check content reference. |
| `clusterName` | Overrides the cluster name used in the XCCDF `<target>` element. |
| `benchmarkId` | Overrides the XCCDF benchmark ID. Defaults to `xccdf_compliance-operator_benchmark_kubernetes`. |

`metadata.yaml` also supports an optional `checks` map for per-check or per-group rule overrides. Each key is either a check ID, which is applied as an exact match, or a group ID, which is applied to all checks in the group. This applies to any benchmark format, including CIS and STIG.

Each entry under `checks` supports the following fields:

| Field | Description |
|---|---|
| `ruleId` | The STIG rule ID, for example `SV-254553r1016525_rule`. This overrides the default compliance-operator rule ID in the XCCDF output. |
| `version` | The rule version string. |
| `severity` | Overrides the rule severity with `high`, `medium`, or `low`. |
| `fixId` | The ID of the fix element. |
| `checkId` | The check system identifier, which maps to the XCCDF `<check system>` attribute. |
| `idents` | A list of XCCDF ident objects, each with `system` and `value` fields. STIG benchmarks use `http://cyber.mil/cci` as the system. CIS benchmarks use the CIS controls URL. |

**Example `metadata.yaml` for a CIS Benchmark**

```yaml
title: "My Custom CIS Benchmark"
publisher: "My Organization"
contributor: "My Team"
creator: "My Team"
description: "Custom CIS benchmark for foo clusters."
platform: "cpe:/a:kubernetes:kubernetes"
notice: "For internal use only."
noticeId: "internal-notice"
referenceHref: "https://example.com/benchmark"
referenceTitle: "My Benchmark Reference"
source: "https://example.com"
```

**Example `metadata.yaml` for a STIG Benchmark**

```yaml
title: "RKE2 STIG Benchmark"
publisher: "Defense Information Systems Agency"
platform: "cpe:/a:kubernetes:kubernetes"
referenceHref: "https://public.cyber.mil/stigs/"
referenceTitle: "DISA STIG for Kubernetes"
checks:
  V-254553:
    ruleId: SV-254553r1016525_rule
    version: SV-254553r1016525_rule
    severity: high
    fixId: F-254553
    checkId: C-254553
    idents:
      - system: "http://cyber.mil/cci"
        value: "CCI-000366"
```

## Creating a Custom Benchmark Version for Running a Cluster Scan

Each Benchmark Version defines a set of test configuration files that define the Compliance tests to be run by the kube-bench [https://github.com/aquasecurity/kube-bench] tool. The `rancher-compliance` application installs a few default Benchmark Versions which are listed under Compliance Benchmark application menu.

But there could be some Kubernetes cluster setups that require custom configurations of the Benchmark tests. For example, the path to the Kubernetes config files or certs might be different than the standard location where the upstream CIS Benchmarks look for them.

But in the following cases, a custom configuration or remediation may be required:

- Non-standard file locations: When Kubernetes binaries, configuration or certificate paths deviate from upstream benchmark defaults.

  Example: Unlike traditional Kubernetes, K3s bundles control plane components into a single binary. Therefore, `--anonymous-auth` flag presence and configuration should be verified in K3s' logs (`journalctl`), not via `kube-apiserver` process checks (`ps`).

- Alternative risk mitigations: If a setup doesn't meet a check but has an equally effective compensating control with justification. Or simply is not concerned by the check requirement because of its design.

  Example: By default, K3s embeds the api server within the k3s process. There is no API server pod specification file, so verifying the latter's file permissions is not required.

#### 1. Prepare the Custom Benchmark Version ConfigMap

To create a custom benchmark version, first you need to create a ConfigMap containing the benchmark version's config files and upload it to your Kubernetes cluster where you want to run the scan.

To prepare a custom benchmark version ConfigMap, suppose we want to add a custom Benchmark Version named `foo`.

1. Create a directory named `foo` and inside this directory, place all the config YAML files that the kube-bench [https://github.com/aquasecurity/kube-bench] tool looks for. For example, here are the config YAML files for a Generic CIS 1.5 Benchmark Version https://github.com/aquasecurity/kube-bench/tree/master/cfg/cis-1.5
2. Place the complete `config.yaml` file, which includes all the components that should be tested.
3. Add the Benchmark version name to the `target_mapping` section of the `config.yaml`:

   ```
   target_mapping:
     "foo":
       - "master"
       - "node"
       - "controlplane"
       - "etcd"
       - "policies"
   ```

4. (Optional) To enrich XCCDF report output with benchmark metadata, create a `metadata.yaml` file in the same directory and include it in the ConfigMap. For example:

```
title: "My Custom Benchmark"
publisher: "My Organization"
platform: "cpe:/a:kubernetes:kubernetes"
```

Refer to Configuration Reference for all supported fields.

5. Upload this directory to your Kubernetes Cluster by creating a ConfigMap:

```
 kubectl create configmap -n <namespace> foo --from-file=<path to
directory foo>
```

**2. Add a Custom Benchmark Version to a Cluster**

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to add a custom benchmark and click **Explore**.
3. In the left navigation bar, click **Compliance > Benchmark Version**.
4. Click **Create**.
5. Enter the **Name** and a description for your custom benchmark version.
6. Choose the cluster provider that your benchmark version applies to.
7. Choose the ConfigMap you have uploaded from the dropdown.
8. Add the minimum and maximum Kubernetes version limits applicable, if any.
9. Click **Create**.

**3. Create a New Profile for the Custom Benchmark Version**

To run a scan using your custom benchmark version, you need to add a new Profile pointing to this benchmark version.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to add a custom benchmark and click **Explore**.
3. In the left navigation bar, click **Compliance > Scan**.
4. Click **Create**.
5. Provide a **Name** and description. In this example, we name it `foo-profile`.
6. Choose the Benchmark Version from the dropdown.
7. Click **Create**.

**4. Run a Scan Using the Custom Benchmark Version**

Once the Profile pointing to your custom benchmark version `foo` has been created, you can create a new Scan to run the custom test configs in the Benchmark Version.

To run a scan,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to add a custom benchmark and click **Explore**.
3. In the left navigation bar, click **CIS Benchmark** › **Scan**.
4. Click **Create**.
5. Choose the new cluster scan profile.
6. Click **Create**.

**Result:** A report is generated with the scan results. To see the results, click the name of the scan that appears.

## Run a Scan

When a ClusterScan custom resource is created, it launches a new compliance scan on the cluster for the chosen ClusterScanProfile.



There is currently a limitation of running only one compliance scan at a time for a cluster. If you create multiple ClusterScan custom resources, they will be run one after the other by the operator, and until one scan finishes, the rest of the ClusterScan custom resources will be in the "Pending" state.

To run a scan,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to run a compliance scan and click **Explore**.
3. Click **Compliance > Scan**.
4. Click **Create**.
5. Choose a cluster scan profile. The profile determines which CIS Benchmark version will be used and which tests will be performed. If you choose the Default profile, then the Compliance Operator will choose a profile applicable to the type of Kubernetes cluster it is installed on.
6. Click **Create**.

**Result:** A report is generated with the scan results. To see the results, click the name of the scan that appears.

## Run a Scan Periodically on a Schedule

To run a ClusterScan on a schedule,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to run a Compliance scan and click **Explore**.
3. Click **Compliance > Scan**.
4. Choose a cluster scan profile. The profile determines which CIS Benchmark version will be used and which tests will be performed. If you choose the Default profile, then the Compliance Operator will choose a profile applicable to the type of Kubernetes cluster it is installed on.
5. Choose the option **Run scan on a schedule**.
6. Enter a valid cron schedule expression [https://en.wikipedia.org/wiki/Cron#CRON_expression] in the field **Schedule**.

7. Choose a **Retention** count, which indicates the number of reports maintained for this recurring scan. By default this count is 3. When this retention limit is reached, older reports will get purged.
8. Click **Create**.

**Result:** The scan runs and reschedules to run according to the cron schedule provided. The **Next Scan** value indicates the next time this scan will run again.

A report is generated with the scan results every time the scan runs. To see the latest results, click the name of the scan that appears.

You can also see the previous reports by choosing the report from the **Reports** dropdown on the scan detail page.

## View Reports

To view the generated Compliance scan reports,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to run a Compliance scan and click **Explore**.
3. Click **Compliance > Scan**.
4. The **Scans** page will show the generated reports. To see a detailed report, go to a scan report and click the name.

You can download the report from the Scans list or from the scan detail page in one of two formats: * **CSV:** A flat file containing all test results for the scan. * **XCCDF (XML):** A per-node report packaged as a `.zip` file with one `.xml` file per node in the cluster.

To get the verbose version of the compliance scan results, run the following command on the cluster that was scanned. Note that the scan must be completed before this can be done.

```
export REPORT="scan-report-name"
kubectl get clusterscanreports.compliance.cattle.io $REPORT -o json |jq
".spec.reportJSON | fromjson" | jq -r ".actual_value_map_data" | base64
-d | gunzip | jq .
```

## Enable Alerting for SUSE Rancher Prime Compliance

Alerts can be configured to be sent out for a scan that runs on a schedule.



*Prerequisite:*

Before enabling alerts for `rancher-compliance`, make sure to install the `rancher-monitoring` application and configure the Receivers and Routes. For more information, see this section.

While configuring the routes for `rancher-compliance` alerts, you can specify the matching using the key-value pair `job: rancher-compliance-scan`. An example route configuration is here.

While installing or upgrading the `rancher-compliance` Helm chart, set the following flag to `true` in the `values.yaml`:

```
alerts:
  enabled: true
```

### Configure Alerts for Periodic Scan on a Schedule

It is possible to run a ClusterScan on a schedule.

A scheduled scan can also specify if you should receive alerts when the scan completes.

Alerts are supported only for a scan that runs on a schedule.

The compliance application supports two types of alerts:

- Alert on scan completion: This alert is sent out when the scan run finishes. The alert includes details including the ClusterScan's name and the ClusterScanProfile name.
- Alert on scan failure: This alert is sent out if there are some test failures in the scan run or if the scan is in a `Fail` state.



*Prerequisite*

Before enabling alerts for `rancher-compliance`, make sure to install the `rancher-monitoring` application and configure the Receivers and Routes. For more information, see this section.

While configuring the routes for `rancher-compliance` alerts, you can specify the matching using the key-value pair `job: rancher-compliance-scan`. An example route configuration is here.

To configure alerts for a scan that runs on a schedule,

1. Please enable alerts on the `rancher-compliance` application. For more information, see this page.
2. In the upper left corner, click ☰ **> Cluster Management**.
3. On the **Clusters** page, go to the cluster where you want to run a Compliance scan and click **Explore**.
4. Click **compliance > Scan**.
5. Click **Create**.
6. Choose a cluster scan profile. The profile determines which CIS Benchmark version will be used and which tests will be performed. If you choose the Default profile, then the Compliance Operator will choose a profile applicable to the type of Kubernetes cluster it is installed on.
7. Choose the option **Run scan on a schedule**.
8. Enter a valid cron schedule expression [https://en.wikipedia.org/wiki/Cron#CRON_expression] in the field **Schedule**.
9. Check the boxes next to the Alert types under **Alerting**.
10. Optional: Choose a **Retention Count**, which indicates the number of reports maintained for this recurring scan. By default this count is 3. When this retention limit is reached, older reports will get

purged.

11. Click **Create**.

**Result:** The scan runs and reschedules to run according to the cron schedule provided. Alerts are sent out when the scan finishes if routes and receiver are configured under `rancher-monitoring` application.

A report is generated with the scan results every time the scan runs. To see the latest results, click the name of the scan that appears.

### Create a Custom Compliance Version for Running a Cluster Scan

There could be some Kubernetes cluster setups that require custom configurations of the Compliance tests. For example, the path to the Kubernetes config files or certs might be different than the standard location where the upstream Compliance look for them.

It is now possible to create a custom compliance version for running a cluster scan using the `rancher-compliance` application.

For details, see this page.

## Pod Security Standards (PSS) & Pod Security Admission (PSA)

Pod Security Standards (PSS) [https://kubernetes.io/docs/concepts/security/pod-security-standards/] and Pod Security Admission (PSA) [https://kubernetes.io/docs/concepts/security/pod-security-admission/] define security restrictions for a broad set of workloads. They became available and were turned on by default in Kubernetes v1.23, and replace Pod Security Policies (PSP) [https://kubernetes.io/docs/concepts/security/pod-security-policy/] in Kubernetes v1.25 and above.

PSS define security levels for workloads. PSAs describe requirements for pod security contexts and related fields. PSAs reference PSS levels to define security restrictions.

### Pod Security Admission Configuration Templates

Rancher offers PSA configuration templates. These are pre-defined security configurations that you can apply to a cluster. Rancher admins (or those with the right permissions) can create, manage, and edit PSA templates.

#### Rancher on PSA-restricted Clusters

Rancher system namespaces are also affected by the restrictive security policies described by PSA templates. You need to exempt Rancher's system namespaces after you assign the template, or else the cluster won't operate correctly. See Pod Security Admission (PSA) Configuration Templates for more details.

For a complete file which has all the exemptions you need to run Rancher, please refer to this sample Admission Configuration.

## Pod Security Admission (PSA) Configuration Templates

Pod Security admission (PSA) configuration templates are a Rancher custom-defined resource (CRD), available in Rancher v2.7.2 and above. The templates provide pre-defined security configurations that you

can apply to a cluster:



The policies shipped by default in Rancher aim to provide a trade-off between security and convenience. If a more strict policy configuration is needed, users are able to craft such policies themselves based on their specific requirements. In the case Rancher policies are preferred, you will need to deploy admission controllers that block the creation of any exempted namespaces that won't be used within your environments.

- `rancher-privileged`: The most permissive configuration. It doesn't restrict the behavior of any pods. This allows for known privilege escalations. This policy has no exemptions.
- `rancher-restricted`: A heavily restricted configuration that follows current best practices for hardening pods. You must make namespace-level exemptions for Rancher components.

## Assign a Pod Security Admissions (PSA) Configuration Template

You can assign a PSA template at the same time that you create a downstream cluster. You can also add a template by configuring an existing downstream cluster.

### Assign a Template During Cluster Creation

**RKE2 and K3s**

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, click the **Create** button.
3. Select a provider.
4. On the **Cluster: Create** page, go to **Basics** › **Security**.
5. In the **Pod Security Admission Configuration Template** dropdown menu, select the template you want to assign.
6. Click **Create**.

```
<h3><a id="_assign_a_template_to_an_existing_cluster"></a>Assign a
Template to an Existing Cluster</h3>
```

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Find the cluster you want to update in the **Clusters** table, and click the ☰.
3. Select **Edit Config** .
4. In the **Pod Security Admission Configuration Template** dropdown menu, select the template you want to assign.
5. Click **Save**.

**RKE1**

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, click the **Create** button.
3. Select a provider.
4. On the **Add Cluster** page, under **Cluster Options**, click **Advanced Options**.

5. In the **Pod Security Admission Configuration Template** dropdown menu, select the template you want to assign.
6. Click **Create**.

<h3><a id="_assign_a_template_to_an_existing_cluster"></a>Assign a Template to an Existing Cluster</h3>

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Find the cluster you want to update in the **Clusters** table, and click the ☰.
3. Select **Edit Config**.
4. On the **Edit Cluster** page, go to **Cluster Options › Advanced Options**.
5. In the **Pod Security Admission Configuration Template**, select the template you want to assign.
6. Click **Save**.

## Add or Edit a Pod Security Admissions (PSA) Configuration Template

If you have administrator privileges, you can customize security restrictions and permissions by creating additional PSA templates, or by editing existing templates.



If you edit an existing PSA template while it is still in use, changes will be applied to all clusters that have been assigned to that template.

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Click **Advanced** to open the dropdown menu.
3. Select **Pod Security Admissions**.
4. Find the template you want to modify, and click the ☰.
5. Select **Edit Config** to edit the template.
6. When you're done editing the configuration, click **Save**.

### Allow Non-Admin Users to Manage PSA Templates

If you want to allow other users to manage templates, you can bind that user to a role that grants all verbs (`"*"`) on `management.cattle.io/podsecurityadmissionconfigurationtemplates`.



Any user that is bound to the above permission will be able to change the restriction levels on *all* managed clusters which use a given PSA template, including ones that they have no permissions on.

## Exempting Required Rancher Namespaces

When you run Rancher on a Kubernetes cluster that enforces a restrictive security policy by default, you'll need to exempt the following namespaces, otherwise the policy might prevent Rancher system pods from running properly.

- `calico-apiserver`

- calico-system
- cattle-alerting
- cattle-capi-system
- cattle-csp-adapter-system
- cattle-elemental-system
- cattle-epinio-system
- cattle-externalip-system
- cattle-fleet-local-system
- cattle-fleet-system
- cattle-gatekeeper-system
- cattle-global-data
- cattle-global-nt
- cattle-impersonation-system
- cattle-istio
- cattle-istio-system
- cattle-logging
- cattle-logging-system
- cattle-monitoring-system
- cattle-neuvector-system
- cattle-prometheus
- cattle-provisioning-capi-system
- cattle-resources-system
- cattle-scc-system
- cattle-sriov-system
- cattle-system
- cattle-turtles-system
- cattle-ui-plugin-system
- cattle-windows-gmsa-system
- cert-manager
- cis-operator-system
- compliance-operator-system
- fleet-default
- fleet-local
- istio-system
- kube-node-lease
- kube-public

- `kube-system`
- `longhorn-system`
- `rancher-alerting-drivers`
- `rancher-compliance-system`
- `security-scan`
- `sr-operator-system`
- `tigera-operator`
- `traefik`

Rancher, some Rancher owned charts, and RKE2 and K3s distributions all use these namespaces. A subset of the listed namespaces are already exempt in the built-in Rancher `rancher-restricted` policy, for use in downstream clusters. For a complete template which has all the exemptions you need to run Rancher, please refer to this sample Admission Configuration.

### Exempting Namespaces

If you assign the `rancher-restricted` template to a cluster, by default the restrictions are applied across the entire cluster at the namespace level. To exempt certain namespaces from this highly restricted policy, do the following:

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Click **Advanced** to open the dropdown menu.
3. Select **Pod Security Admissions**.
4. Find the template you want to modify, and click the ☰.
5. Select **Edit Config**.
6. Click the **Namespaces** checkbox under **Exemptions** to edit the **Namespaces** field.
7. When you're done exempting namespaces, click **Save**.



You need to update the target cluster to make the new template take effect in that cluster. An update can be triggered by editing and saving the cluster without changing values.

## Sample PodSecurityConfiguration

The following PodSecurityConfiguration contains the required Rancher namespace exemptions for a `rancher-restricted` cluster to run properly.

```
apiVersion: apiserver.config.k8s.io/v1
kind: AdmissionConfiguration
plugins:
  - name: PodSecurity
    configuration:
      apiVersion: pod-security.admission.config.k8s.io/v1
      kind: PodSecurityConfiguration
      defaults:
```

```
        enforce: "restricted"
        enforce-version: "latest"
        audit: "restricted"
        audit-version: "latest"
        warn: "restricted"
        warn-version: "latest"
      exemptions:
        namespaces:
        - calico-apiserver
        - calico-system
        - cattle-alerting
        - cattle-capi-system
        - cattle-csp-adapter-system
        - cattle-elemental-system
        - cattle-epinio-system
        - cattle-externalip-system
        - cattle-fleet-local-system
        - cattle-fleet-system
        - cattle-gatekeeper-system
        - cattle-global-data
        - cattle-global-nt
        - cattle-impersonation-system
        - cattle-istio
        - cattle-istio-system
        - cattle-logging
        - cattle-logging-system
        - cattle-monitoring-system
        - cattle-neuvector-system
        - cattle-prometheus
        - cattle-provisioning-capi-system
        - cattle-resources-system
        - cattle-scc-system
        - cattle-sriov-system
        - cattle-system
        - cattle-turtles-system
        - cattle-ui-plugin-system
        - cattle-windows-gmsa-system
        - cert-manager
        - cis-operator-system
        - compliance-operator-system
        - fleet-default
        - fleet-local
        - istio-system
        - kube-node-lease
        - kube-public
```

```
        - kube-system
        - longhorn-system
        - rancher-alerting-drivers
        - rancher-compliance-system
        - security-scan
        - sr-operator-system
        - tigera-operator
        - traefik
      runtimeClasses: []
      usernames: []
```

# Secrets

Secrets [https://kubernetes.io/docs/concepts/configuration/secret/#overview-of-secrets] store sensitive data like passwords, tokens, or keys. They may contain one or more key value pairs.



This page is about secrets in general. For details on setting up a private registry, refer to the section on registries.

When configuring a workload, you are able to choose which secrets to include. Like config maps, secrets can be referenced by workloads as either an environment variable or a volume mount.

Mounted secrets are updated automatically unless they are mounted as subpath volumes. For details on how updated secrets are propagated, refer to the Kubernetes documentation. [https://kubernetes.io/docs/concepts/configuration/secret/#mounted-secrets-are-updated-automatically]

## Creating Secrets in Namespaces

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to add a secret and click **Explore**.
3. To navigate to secrets, you may click either **Storage** › **Secrets** or **More Resources** › **Core** › **Secrets**.
4. Select the **Namespaced** tab.
5. Click **Create**.
6. Select the type of secret you want to create.
7. Select a **Namespace** for the secret.
8. Enter a **Name** for the secret.



Kubernetes classifies secrets, certificates, and registries all as secrets [https://kubernetes.io/docs/concepts/configuration/secret/], and no two secrets in a namespace can have duplicate names. Therefore, to prevent conflicts, your secret must have a unique name among all secrets within your workspace.

9. From **Data**, click **Add** to add a key-value pair. Add as many values as you need.



You can add multiple key value pairs to the secret by copying and pasting.

[Add multiple key value pairs to the secret by copying and pasting.] | *../en/_images/bulk-key-values.gif*

10. Click **Save**.

**Result:** Your secret is added to the namespace you chose. You can view the secret in the Rancher UI by clicking either **Storage › Secrets** or **More Resources › Core › Secrets**.

Mounted secrets are updated automatically unless they are mounted as subpath volumes. For details on how updated secrets are propagated, refer to the Kubernetes documentation. [https://kubernetes.io/docs/concepts/configuration/secret/#mounted-secrets-are-updated-automatically]

## Creating Secrets in Projects

When creating a secret in a project scope, the secret is copied into all namespaces within the project.

#### Creating a Project Scoped Secret in the UI

1. In the upper left corner, click ☰ **> Cluster Management**.
2. Go to the cluster where you want to add a secret and click **Explore**.
3. To navigate to secrets, you may click either **Storage > Secrets** or **More Resources > Core > Secrets**.
4. Select the **Project Scoped** tab.
5. Click **Create Project Scoped Secret**.
6. Select the type of secret you want to create.
7. Select a **Project** for the secret.
8. Enter a **Name** for the secret.



Kubernetes classifies secrets, certificates, and registries all as secrets [https://kubernetes.io/docs/concepts/configuration/secret/], and no two secrets in a namespace can have duplicate names. If you create a project scoped secret that has the same name as an existing secret in one of the project namespaces, the existing secret is overwritten.

9. From **Data**, click **Add** to add a key-value pair. Add as many values as you need.



You can add multiple key-value pairs to the secret by copying and pasting.

[Add multiple key-value pairs to the secret by copying and pasting.] | *../en/_images/bulk-key-values.gif*

10. Click **Save**.

**Result:** Your secret is added to each namespace within the project. You can view the secret in the Rancher UI by clicking either **Storage > Secrets** or **More Resources > Core > Secrets**.

#### Creating a Project Scoped Secret with kubectl

Project scoped secrets work by creating the original secret on the management cluster in what's known as the "Project Backing Namespace". Rancher stores important project related information in this namespace. You can find it in the `status.backingNamespace` field in the project CRD, or by doing `kubectl get projects -A` in the management cluster.

In order for the secret to be acknowledged by Rancher as a project scoped secret, it also needs the label `management.cattle.io/project-scoped-secret: <projectID>`.

Example yaml:

```
apiVersion: v1
data:
  key: ZG9n
kind: Secret
metadata:
  labels:
    management.cattle.io/project-scoped-secret: p-vwxyz
  name: test-secret
  namespace: c-abcl23-p-vwxyz
type: Opaque
```

In the above YAML, the namespace is the backing namespace of project `p-vwxyz` and the project scoped secret label references the projectID. When applied to the management cluster, all namespaces within the project `p-vwxyz` contain a copy of `test-secret`.

### What's Next?

Now that you have a secret added to a namespace, you can add it to a workload that you deploy.

For more information on adding secret to a workload, see Deploying Workloads.

## Opening Ports with firewalld



Enabling firewalld can cause serious network communication problems.

For proper network function, firewalld must be disabled on systems running RKE2. Firewalld conflicts with Canal [https://documentation.suse.com/cloudnative/rke2/latest/en/known_issues.html#_firewalld_conflicts_with_default_networking], RKE2's default networking stack.

Firewalld must also be disabled on systems running Kubernetes 1.19 and later.

If you enable firewalld on systems running Kubernetes 1.18 or earlier, understand that this may cause networking issues. CNIs in Kubernetes dynamically update iptables and networking rules independently of any external firewalls, such as firewalld. This can cause unexpected behavior when the CNI and the external firewall conflict.

Some distributions of Linux derived from RHEL, [https://en.wikipedia.org/wiki/Red_Hat_Enterprise_Linux#Rebuilds] including Oracle Linux, may have default firewall rules that block communication with Helm.

For example, one Oracle Linux image in AWS has REJECT rules that stop Helm from communicating with Tiller:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             state
RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state NEW
tcp dpt:ssh
REJECT     all  --  anywhere             anywhere             reject-
with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
REJECT     all  --  anywhere             anywhere             reject-
with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

You can check the default firewall rules with this command:

```
sudo iptables --list
```

This section describes how to use `firewalld` to apply the firewall port rules for nodes in a high-availability Rancher server cluster.

### Prerequisite

Install v7.x or later of`firewalld`:

```
yum install firewalld
systemctl start firewalld
systemctl enable firewalld
```

### Applying Firewall Port Rules

In the Rancher high-availability installation instructions, the Rancher server is set up on three nodes that have all three Kubernetes roles: etcd, controlplane, and worker. If your Rancher server nodes have all three roles, run the following commands on each node:

```
firewall-cmd --permanent --add-port=22/tcp
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
```

```
firewall-cmd --permanent --add-port=2376/tcp
firewall-cmd --permanent --add-port=2379/tcp
firewall-cmd --permanent --add-port=2380/tcp
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --permanent --add-port=9099/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10254/tcp
firewall-cmd --permanent --add-port=30000-32767/tcp
firewall-cmd --permanent --add-port=30000-32767/udp
```

If your Rancher server nodes have separate roles, use the following commands based on the role of the node:

```
# For etcd nodes, run the following commands:
firewall-cmd --permanent --add-port=2376/tcp
firewall-cmd --permanent --add-port=2379/tcp
firewall-cmd --permanent --add-port=2380/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --permanent --add-port=9099/tcp
firewall-cmd --permanent --add-port=10250/tcp

# For control plane nodes, run the following commands:
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --permanent --add-port=2376/tcp
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --permanent --add-port=9099/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10254/tcp
firewall-cmd --permanent --add-port=30000-32767/tcp
firewall-cmd --permanent --add-port=30000-32767/udp

# For worker nodes, run the following commands:
firewall-cmd --permanent --add-port=22/tcp
firewall-cmd --permanent --add-port=80/tcp
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --permanent --add-port=2376/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --permanent --add-port=9099/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10254/tcp
firewall-cmd --permanent --add-port=30000-32767/tcp
firewall-cmd --permanent --add-port=30000-32767/udp
```

After the `firewall-cmd` commands have been run on a node, use the following command to enable the firewall rules:

```
firewall-cmd --reload
```

**Result:** The firewall is updated so that Helm can communicate with the Rancher server nodes.

# Encrypting HTTP Communication

When you create an ingress within Rancher/Kubernetes, you must provide it with a secret that includes a TLS private key and certificate, which are used to encrypt and decrypt communications that come through the ingress. You can make certificates available for ingress use by adding the certificate to the ingress deployment.



*Prerequisite:*

You must have a TLS private key and certificate available to upload.

## 1. Create a Secret

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to deploy your ingress and click **More Resources** › **Core** › **Secrets**.
3. Click **Create**.
4. Click **TLS Certificate**.
5. Enter a name for the secret. Note: Your secret must have a unique name among the other certificates, registries, and secrets within your project/workspace.
6. In the **Private Key** field, either copy and paste your certificate's private key into the text box (include the header and footer), or click **Read from a file** to browse to the private key on your file system. If possible, we recommend using **Read from a file** to reduce likelihood of error. Note: Private key files end with an extension of `.key`.
7. In the **Certificate** field, either copy and paste your certificate into the text box (include the header and footer), or click **Read from a file** to browse to the certificate on your file system. If possible, we recommend using **Read from a file** to reduce likelihood of error. Note: Certificate files end with an extension of `.crt`.
8. Click **Create**.

## 2. Add the Secret to an Ingress

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the cluster where you want to deploy your ingress and click **Service Discovery** › **Ingresses**.
3. Click **Create**.
4. Select the **Namespace** of the ingress.
5. Enter a **Name** for the ingress.
6. In the **Certificates** tab, select the secret containing your certificate and private key.

7. Click **Create**.

## What's Next?

Now you can add the certificate when launching an ingress within the current project or namespace. For more information, see Adding Ingress.

# Integrations

Prime is the Rancher ecosystem's enterprise offering, with additional security, extended lifecycles, and access to Prime-exclusive documentation. Rancher Prime installation assets are hosted on a trusted SUSE registry, owned and managed by Rancher. The trusted Prime registry includes only stable releases that have been community-tested.

Prime also offers options for production support, as well as add-ons to your subscription that tailor to your commercial needs.

To learn more and get started with Rancher Prime, please visit this page [https://www.rancher.com/quick-start].

To learn more about Rancher's ecosystem, refer to the following integrations:

- Kubernetes Distributions
- SUSE Virtualization
- SUSE Storage
- SUSE Security
- SUSE Security Admission Controller
- SUSE® Rancher Prime: OS Manager
- SUSE® Rancher Prime: Continuous Delivery
- SUSE® Observability
- Kubernetes on the Desktop with Rancher Desktop
- SUSE® Rancher Prime: Cluster API

# Kubernetes Distributions

## SUSE® Rancher Prime: K3s

K3s is a lightweight, fully compliant Kubernetes distribution designed for a range of use cases, including edge computing, IoT, CI/CD, development and embedding Kubernetes into applications. It simplifies Kubernetes management by packaging the system as a single binary, using sqlite3 as the default storage, and offering a user-friendly launcher. K3s includes essential features like local storage and load balancing, Helm chart controller and the Traefik CNI. It minimizes external dependencies and provides a streamlined Kubernetes experience. K3s was donated to the CNCF as a Sandbox Project in June 2020.

#### SUSE® Rancher Prime: K3s with Rancher

- Rancher allows easy provision of K3s across a range of platforms including Amazon EC2, DigitalOcean, Azure, vSphere, or existing servers.
- Standard Rancher management of Kubernetes clusters including all outlined cluster management capabilities.

## SUSE® Rancher Prime: RKE2

RKE2 is a compliant Kubernetes distribution developed by Rancher. It is specifically designed for security and compliance within the U.S. Federal Government sector.

Primary characteristics of RKE2 include:

1. **Security and Compliance Focus**: RKE2 places a strong emphasis on security and compliance, operating under a "secure by default" framework, making it suitable for government services and highly regulated industries like finance and healthcare.
2. **CIS Kubernetes Benchmark Conformance**: RKE2 comes pre-configured to meet the CIS Kubernetes Hardening Benchmark (currently supporting v1.23 and v1.7), with minimal manual intervention required.
3. **FIPS 140-2 Compliance**: RKE2 complies with the FIPS 140-2 standard using FIPS-validated crypto modules for its components.
4. **Embedded etcd**: RKE2 defaults to using an embedded etcd as its data store. This aligns it more closely with standard Kubernetes practices, allowing better integration with other Kubernetes tools and reducing the risk of misconfiguration.
5. **Alignment with Upstream Kubernetes**: RKE2 aims to stay closely aligned with upstream Kubernetes, reducing the risk of non-conformance that may occur when using distributions that deviate from standard Kubernetes practices.
6. **Multiple CNI Support**: RKE2 offers support for multiple Container Network Interface (CNI) plugins, including Cilium, Calico, and Multus. This is essential for use cases such as telco distribution centers and factories with various production facilities.

### SUSE® Rancher Prime: RKE2 with Rancher

- Rancher allows easy provision of RKE2 across a range of platforms including Amazon EC2, DigitalOcean, Azure, vSphere, or existing servers.
- Standard Rancher management of Kubernetes clusters including all outlined cluster management capabilities.

## Extensions

Extensions allow users, developers, partners, and customers to extend and enhance the Rancher UI. In addition, users can make changes and create enhancements to their UI functionality independent of Rancher releases. Extensions will enable users to build on top of Rancher to better tailor it to their respective environments. Note that users will also have the ability to update to new versions as well as roll back to a previous version.

Extensions are Helm charts that can only be installed once into a cluster; therefore, these charts have been simplified and separated from the general Helm charts listed under **Apps**.

Examples of built-in Rancher extensions are Fleet, Explorer, and Harvester. Examples of other extensions that use the Extensions API that can be manually added are Kubewarden and Elemental.

### Prerequisites

> 1. You must log in as an administrator to view and interact with the extensions management page.
> 2. You must enable extension support.

## Enabling Extension Support in Rancher

Rancher v2.9.0 and later includes extension support.

You can confirm if extension support is enabled by checking the `uiextension` feature flag. Any changes to this feature flag cause the Rancher pod to restart.

When you enable extension support for the first time, it creates resources, such as the CRDs, so that UI Extensions can work. When extension support is disabled, it disables the endpoints and does not cache any files. However, it does not remove any CRs or delete any extensions that were installed before. If re-enabled, it exposes the required endpoints again and creates CRDs as needed. The extensions that were already installed load after the Rancher pod restarts.

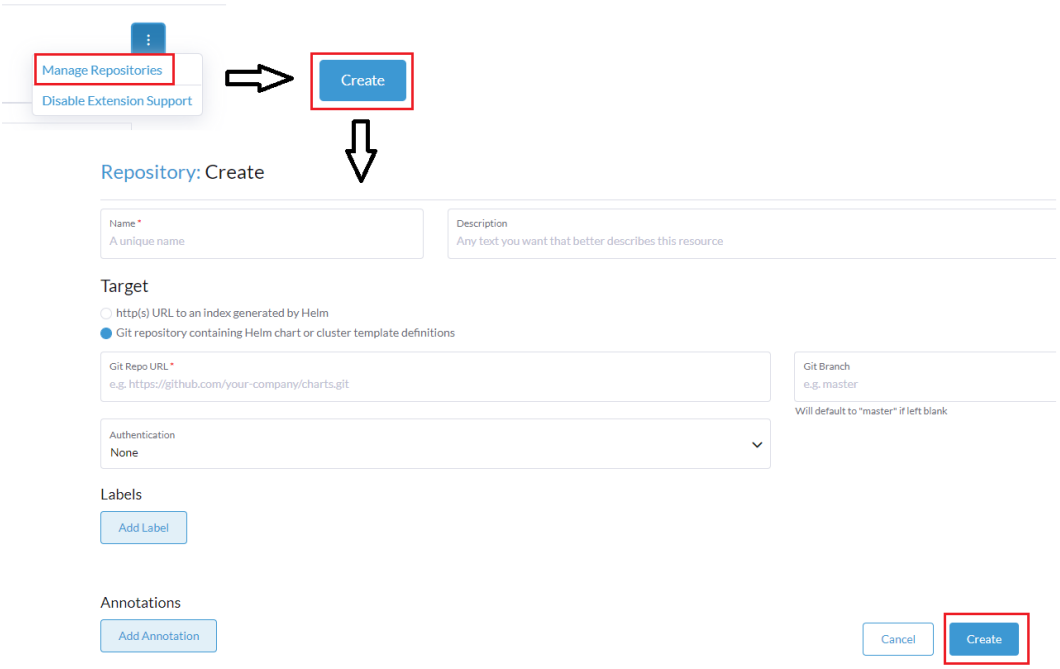
### Caching Extension Files

By default, Rancher caches every extension file in the file system. You can change that behavior by setting `plugin.noCache` to `true`.

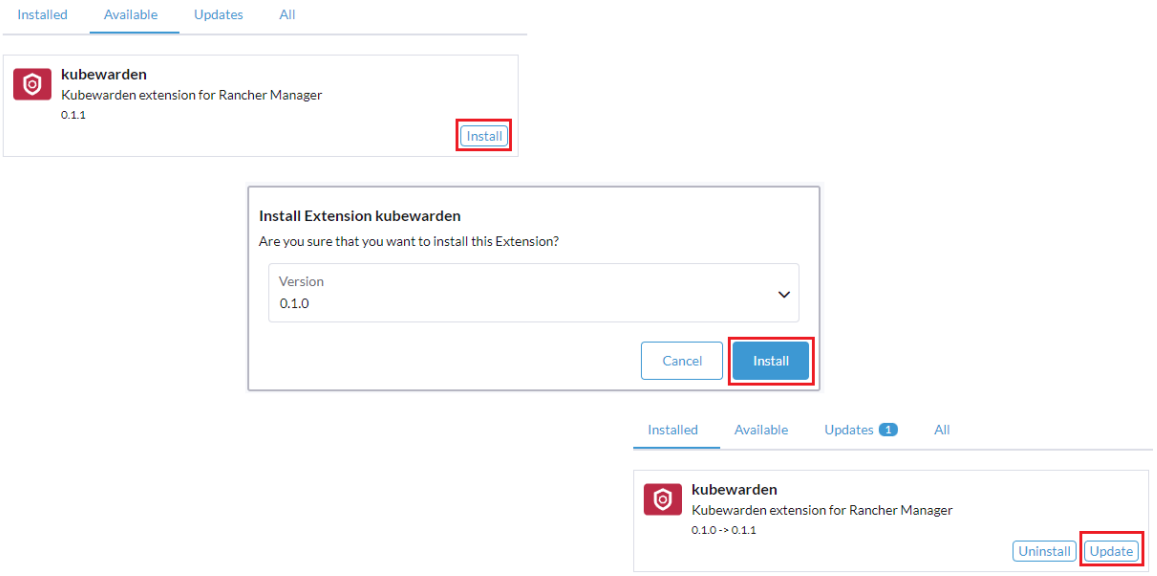
Rancher does have a cached file size limit of 30MB. If an extension has a file bigger than that, the cache is disabled and `plugin.noCache` is set to `true`, regardless of user input.

## Installing Extensions

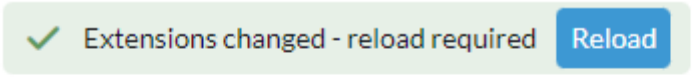
1. Click ☰ > **Extensions** under **Configuration**.
2. On the **Extensions** page, select the **Available** tab to choose the extensions that you want to install.
3. If no extensions are listed as available, you can manually add the repos:
   a. On the upper right, click ☰ > **Manage Repositories** > **Create**.
   b. Add the desired repo name, making sure to also specify the Git repo URL and the Git branch.
   c. Click **Create** in the lower right again to complete.



4. Under the **Available** tab, click **Install** on the desired extension and version, as in the example below. You can also update your extension from this screen. The button to **Update** appears on the extension card if an update is available.




5. Click **Reload** after your extension successfully installs to check its status. Updates to the UI aren't visible until you reload the page.




## Updating and Upgrading Extensions

1. Click ☰ > **Extensions** under **Configuration**.
2. Select the **Updates** tab.
3. Click **Update**.

If there is a new version of the extension, there will also be an **Update** button visible on the associated card for the extension in the **Available** tab.

## Deleting Extensions

1. Click ☰, then click on the name of your local cluster.
2. From the sidebar, select **Apps** › **Installed Apps**.
3. Find the name of the chart you want to delete and select the checkbox next to it.
4. Click **Delete**.

## Deleting Extension Repositories

1. Click ☰ > **Extensions** under **Configuration**.
2. On the top right, click ☰ > **Manage Repositories**.
3. Find the name of the extension repository you want to delete. Select the checkbox next to the repository name, then click **Delete**.

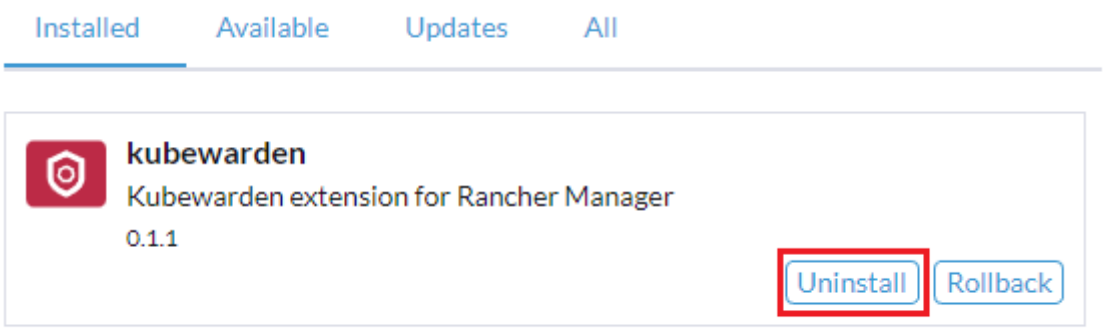
## Deleting Extension Repository Container Images

1. Click ☰, then select **Extensions**, under **Configuration**.
2. On the top right, click ☰ > **Manage Extension Catalogs**.
3. Find the name of the container image you want to delete, then click ☰ > **Uninstall**.

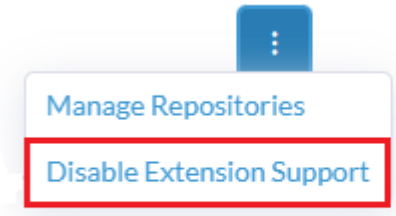
## Uninstalling Extensions

There are two ways to uninstall or disable an extension:

1. Under the **Installed** tab, click the **Uninstall** button on the extension you wish to remove.




2. On the extensions management page, click ☰ > **Disable Extension Support**. This will disable all installed extensions.






You must reload the page after disabling extensions or display issues may occur.

## Enabling Unauthenticated Access to an Extension

In Rancher v2.9.0 and later, you can allow unauthenticated access to an extension. You may want to enable unauthenticated access if the extension enables a new locale or adds custom branding. By default, all extensions require user authentication to load.

To enable unauthenticated access to an extension, set `plugin.noAuth` to `true` in the CR used by the extension.

## Developing Extensions

To learn how to develop your own extensions, refer to the official Getting Started [https://rancher.github.io/dashboard/extensions/extensions-getting-started] guide.

## Working with Extensions in an Air-gapped Environment

If you intend to work with extensions in an air-gapped environment, you must perform some extra steps before you can complete certain tasks.

### Accessing Rancher UI Extensions in an Air-Gapped Environment

Rancher provides some extensions, such as Kubewarden and Elemental, through the `ui-plugin-catalog` container image at https://hub.docker.com/r/rancher/ui-plugin-catalog/tags. If you're trying to install these extensions in an air-gapped environment, you must make the `ui-plugin-catalog` image accessible.

1. Mirror the `ui-plugin-catalog` image to a private registry:

```
export REGISTRY_ENDPOINT=<my-private-registry-endpoint> # e.g. "my-private-registry.com"
docker pull rancher/ui-plugin-catalog:<tag>
docker tag rancher/ui-plugin-catalog:<tag> $REGISTRY_ENDPOINT/rancher/ui-plugin-catalog:<tag>
docker push $REGISTRY_ENDPOINT/rancher/ui-plugin-catalog:<tag>
```

2. Use the mirrored image to create a Kubernetes deployment [https://kubernetes.io/docs/concepts/workloads/controllers/deployment/]:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ui-plugin-catalog
  namespace: cattle-ui-plugin-system
  labels:
    catalog.cattle.io/ui-extensions-catalog-image: ui-plugin-catalog
spec:
  replicas: 1
  selector:
    matchLabels:
      catalog.cattle.io/ui-extensions-catalog-image: ui-plugin-catalog
  template:
    metadata:
      namespace: cattle-ui-plugin-system
```

```
      labels:
        catalog.cattle.io/ui-extensions-catalog-image: ui-plugin-
catalog
    spec:
      containers:
      - name: server
        image: <my-private-registry-endpoint>/rancher/ui-plugin-
catalog:<tag>
        imagePullPolicy: Always
      imagePullSecrets:
        - name: <my-registry-credentials>
```

3. Expose the deployment by creating a ClusterIP service [https://kubernetes.io/docs/concepts/services-networking/service/#type-clusterip]:

```
apiVersion: vl
kind: Service
metadata:
name: ui-plugin-catalog-svc
namespace: cattle-ui-plugin-system
spec:
ports:
** name: catalog-svc-port
port: 8080
protocol: TCP
targetPort: 8080
selector:
  catalog.cattle.io/ui-extensions-catalog-image: ui-plugin-catalog
type: ClusterIP
```

4. Create a ClusterRepo that targets the ClusterIP service:

```
  apiVersion: catalog.cattle.io/vl
  kind: ClusterRepo
  metadata:
 name: ui-plugin-catalog-repo
  spec:
 url: http://ui-plugin-catalog-svc.cattle-ui-plugin-system:8080
```

After you successfully set up these resources, you can install the extensions from the `ui-plugin-charts` manifest into your air-gapped environment.

#### Importing and Installing Extensions in an Air-gapped Environment

1. Find the address of the container image repository that you want to import as an extension. You should import and use the latest tagged version of the image to ensure you receive the latest features and security updates.
   - **(Optional)** If the container image is private: Create a registry secret within the `cattle-ui-plugin-system` namespace. Enter the domain of the image address in the **Registry Domain Name** field.
2. Click ☰, then select **Extensions**, under **Configuration**.
3. On the top right, click ☰ **> Manage Extension Catalogs**.
4. Select the **Import Extension Catalog** button.
5. Enter the image address in the **Catalog Image Reference** field.
   - **(Optional)** If the container image is private, select the secret you just created from the **Pull Secrets** drop-down menu.
6. Click **Load**. The extension will now be **Pending**.
7. Return to the **Extensions** page.
8. Select the **Available** tab, and click **Reload** to make sure that the list of extensions is up to date.
9. Find the extension you just added, and click **Install**.

#### Updating and Upgrading an Extensions Repository in an Air-gapped Environment

Extensions repositories that aren't air-gapped are automatically updated. If the repository is air-gapped, you must update it manually.

First, mirror the latest changes to your private registry by following the same steps for initially importing and installing an extension repository.

After you mirror the latest changes, follow these steps:

1. Click ☰ **> Local**.
2. From the sidebar, select **Workloads › Deployments**.
3. From the namespaces dropdown menu, select **cattle-ui-plugin-system**.
4. Find the **cattle-ui-plugin-system** namespace.
5. Select the `ui-plugin-catalog` deployment.
6. Click ☰ **> Edit config**.
7. Update the **Container Image** field within the deployment's container with the latest image.
8. Click **Save**.

# SUSE Virtualization

Introduced in Rancher v2.6.1, Harvester is an open-source hyper-converged infrastructure (HCI) software built on Kubernetes. Harvester installs on bare metal servers and provides integrated virtualization and distributed storage capabilities. Although Harvester operates using Kubernetes, it does not require knowledge of Kubernetes concepts, making it more user-friendly.

## SUSE Virtualization with Rancher

With Rancher Prime and Harvester, IT operators now have access to an enterprise-ready, simple-to-use infrastructure platform that cohesively manages their virtual machines and Kubernetes clusters alongside one another. For more information on the support offering, see the Support Matrix [https://www.suse.com/suse-harvester/support-matrix/all-supported-versions/harvester-v1-7-x/]. With the Rancher Virtualization Management feature, users can import and manage multiple Harvester clusters. Leveraging the Rancher's authentication feature and RBAC control for multi-tenancy support.

## Overview

Introduced in Rancher v2.6.1, SUSE Virtualization [https://documentation.suse.com/cloudnative/virtualization/v1.7/en/introduction/overview.html] is an open-source hyper-converged infrastructure (HCI) software built on Kubernetes. Harvester installs on bare metal servers and provides integrated virtualization and distributed storage capabilities. Although Harvester operates using Kubernetes, it does not require users to know Kubernetes concepts, making it a more user-friendly application.

### Feature Flag

The Harvester feature flag is used to manage access to the Virtualization Management (VM) page in Rancher where users can navigate directly to Harvester clusters and access the Harvester UI. The Harvester feature flag is enabled by default. Click here for more information on feature flags in Rancher.

To navigate to the Harvester cluster, click ☰ > **Virtualization Management**. From Harvester Clusters page, click one of the clusters listed to go to the single Harvester cluster view.

- If the Harvester feature flag is enabled, Harvester clusters will be filtered out from any pages or apps (such as Continuous Delivery with Fleet) that list Kubernetes clusters.
- If the Harvester feature flag is disabled, and a Harvester cluster is imported, the Harvester cluster will be shown in the Rancher cluster list in the Cluster Management page. Harvester clusters will only be shown on the cluster list when the feature flag is off.
- With the Harvester integration, Harvester clusters can now be imported into Rancher as a cluster type `Harvester`.
- Users may import a Harvester cluster only on the Virtualization Management page. Importing a cluster on the Cluster Management page is not supported, and a warning will advise you to return to the VM page to do so.

### SUSE Virtualization Node Driver

The SUSE Virtualization node driver [https://documentation.suse.com/cloudnative/virtualization/v1.7/en/integrations/rancher/node-driver/node-driver.html] is generally available for K3s and RKE2 options in Rancher. The node driver is available whether or not the Harvester feature flag is enabled. Note that the node driver is off by default. Users may create K3s or RKE2 clusters on Harvester only from the Cluster Management page.

Harvester allows `.ISO` images to be uploaded and displayed through the Harvester UI, but this is not supported in the Rancher UI. This is because `.ISO` images usually require additional setup that interferes with a clean deployment (without requiring user intervention), and they are not typically used in cloud environments.

See Provisioning Drivers for more information on node drivers in Rancher.

#### Port Requirements

The port requirements for the Harvester cluster can be found here [https://documentation.suse.com/cloudnative/virtualization/v1.7/en/installation-setup/requirements.html#_network_requirements].

In addition, other networking considerations are as follows:

- Be sure to enable VLAN trunk ports of the physical switch for VM VLAN networks.
- Follow the networking setup guidance here [https://documentation.suse.com/cloudnative/virtualization/v1.7/en/networking/cluster-network.html].

For other port requirements for other guest clusters, such as K3s and RKE2, please see these docs [https://documentation.suse.com/cloudnative/virtualization/v1.7/en/installation-setup/requirements.html#_port_requirements_for_k3s_and_rke2_clusters].

# SUSE Storage

Longhorn is an official Cloud Native Computing Foundation project (CNCF) [https://cncf.io/] project that delivers a powerful cloud-native distributed storage platform for Kubernetes that can run anywhere. When combined with Rancher, Longhorn makes the deployment of highly available persistent block storage in your Kubernetes environment easy, fast and reliable.

## SUSE Storage with Rancher

With Rancher Prime and Longhorn, users can easily deploy with 1-click via the Rancher catalog and conduct lifecycle management for managed clusters; empowering the user to install and upgrade, together with draining operation for graceful operations. Longhorn with Rancher also provides mixed cluster support with Windows, Rancher hosted images, UI Proxy access through Rancher, and Rancher monitoring with Longhorn metrics.

## Overview

SUSE Storage [https://documentation.suse.com/cloudnative/storage/1.11/en/longhorn-documentation.html] is a lightweight, reliable, and easy-to-use distributed block storage system for Kubernetes.

Longhorn is free, open source software. Originally developed by Rancher Labs, it is now being developed as a sandbox project of the Cloud Native Computing Foundation. It can be installed on any Kubernetes cluster with Helm, with kubectl, or with the Rancher UI. You can learn more about its architecture here. [https://documentation.suse.com/cloudnative/storage/1.11/en/introduction/concepts.html]

With Longhorn, you can:

- Use Longhorn volumes as persistent storage for the distributed stateful applications in your Kubernetes cluster
- Partition your block storage into Longhorn volumes so that you can use Kubernetes volumes with or without a cloud provider
- Replicate block storage across multiple nodes and data centers to increase availability
- Store backup data in external storage such as NFS or AWS S3
- Create cross-cluster disaster recovery volumes so that data from a primary Kubernetes cluster can be quickly recovered from backup in a second Kubernetes cluster

- Schedule recurring snapshots of a volume, and schedule recurring backups to NFS or S3-compatible secondary storage
- Restore volumes from backup
- Upgrade Longhorn without disrupting persistent volumes

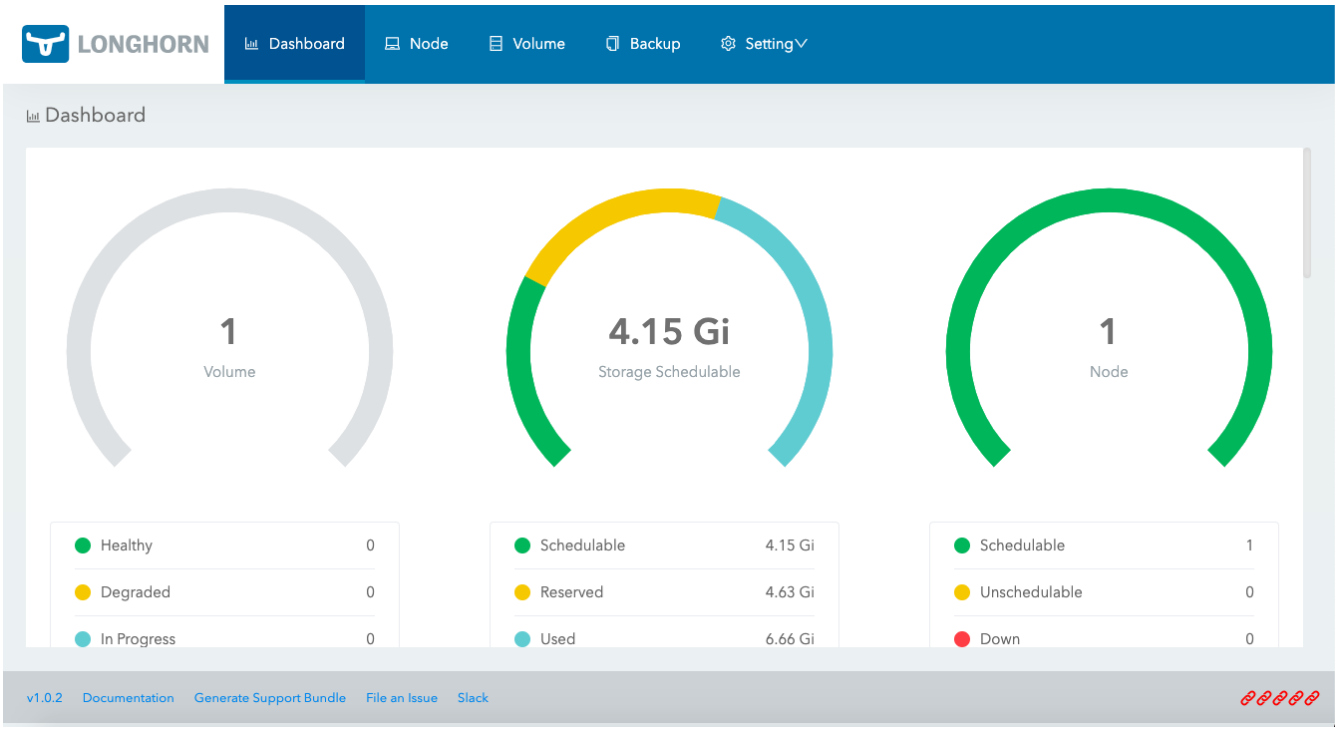



*Figure 15. Longhorn Dashboard*

### Installing SUSE Storage with Rancher

1. Fulfill all Installation Requirements. [https://documentation.suse.com/cloudnative/storage/1.11/en/installation-setup/requirements.html]
2. Go to the cluster where you want to install Longhorn.
3. Click **Apps**.
4. Click **Charts**.
5. Click **Longhorn**.
6. Optional: To customize the initial settings, click **Longhorn Default Settings** and edit the configuration. For help customizing the settings, refer to the SUSE Storage documentation. [https://documentation.suse.com/cloudnative/storage/1.11/en/longhorn-system/settings.html]
7. Click **Install**.

**Result:** Longhorn is deployed in the Kubernetes cluster.

### Accessing SUSE Storage from the Rancher UI

1. Go to the cluster where Longhorn is installed. In the left navigation menu, click **Longhorn**.
2. On this page, you can edit Kubernetes resources managed by Longhorn. To view the Longhorn UI, click the **Longhorn** button in the **Overview** section.

**Result:** You will be taken to the Longhorn UI, where you can manage your Longhorn volumes and their replicas in the Kubernetes cluster, as well as secondary backups of your Longhorn storage that may exist in

another Kubernetes cluster or in S3.

#### Uninstalling SUSE Storage from the Rancher UI

1. Go to the cluster where Longhorn is installed and click **Apps**.
2. Click **Installed Apps**.
3. Go to the `longhorn-system` namespace and check the boxes next to the `longhorn` and `longhorn-crd` apps.
4. Click **Delete,** and confirm **Delete**.

**Result:** Longhorn is uninstalled.

#### GitHub Repository

The Longhorn project is available here. [https://github.com/longhorn/longhorn]

#### Documentation

The Longhorn documentation is here. [https://documentation.suse.com/cloudnative/storage/1.11/en/longhorn-documentation.html]

#### Architecture

Longhorn creates a dedicated storage controller for each volume and synchronously replicates the volume across multiple replicas stored on multiple nodes.

The storage controller and replicas are themselves orchestrated using Kubernetes.

You can learn more about its architecture here. [https://documentation.suse.com/cloudnative/storage/1.11/en/introduction/concepts.html]

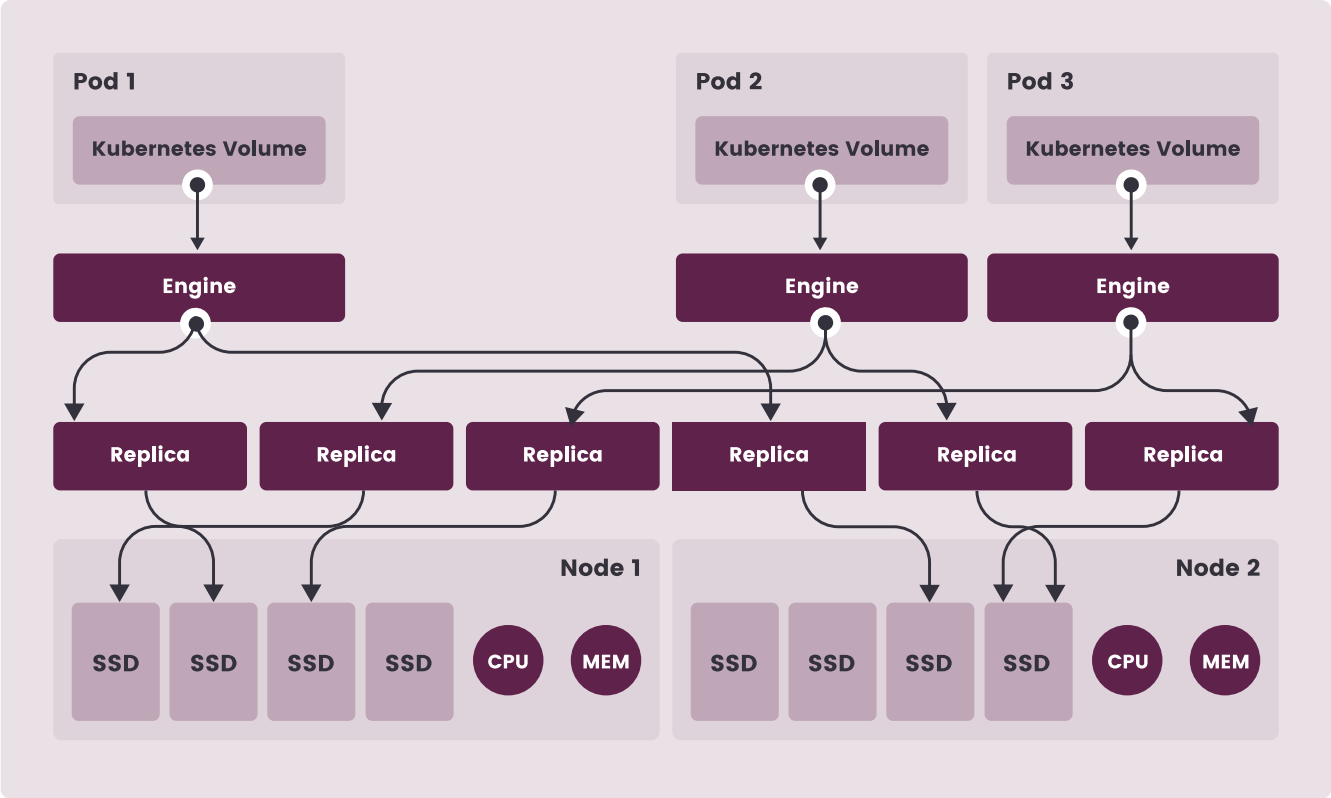



*Figure 16. Longhorn Architecture*

# SUSE Security

NeuVector is the only 100% open source, Zero Trust container security platform. Continuously scan throughout the container lifecycle. Remove security roadblocks. Bake in security policies at the start to maximize developer agility. NeuVector provides vulnerability and compliance scanning and management from build to production. The unique NeuVector run-time protection protects network connections within and ingress/egress to the cluster with a Layer7 container firewall. Additionally, NeuVector monitors process and file activity in containers and on hosts to stop unauthorized activity.

## SUSE Security with Rancher

All NeuVector features are available through Rancher with integrated deployment and single-sign on to the NeuVector console. Rancher cluster admins are able to deploy and manage the NeuVector deployment on their clusters and easily configure NeuVector through Helm values, configMaps, custom resource definitions (CRDs) and the NeuVector console.

With NeuVector and Rancher:

- Deploy, manage and secure multiple clusters.
- Manage and report vulnerabilities and compliance results for Rancher workloads and nodes.

## SUSE Security Prime with Cloud Native

The NeuVector UI Extension for Rancher Manager is available and supported for Rancher Prime and NeuVector Prime customers. This extension provides:

- Automated deployment of NeuVector, including the Rancher Prime NeuVector Extension dashboard.
- Access to important security information from each cluster, such as critical security events, vulnerability

scan results, and ingress/egress exposures.

- Integrated vulnerability (CVE) and compliance scan results directly in Rancher resources such as nodes and containers/pods.
- Integrated actions such as manual triggers of scans on Rancher resources.

## Overview

NeuVector 5.x [https://documentation.suse.com/cloudnative/security/] is an open-source container-centric security platform that is integrated with Rancher. NeuVector offers real-time compliance, visibility, and protection for critical applications and data during runtime. NeuVector provides a firewall, container process/file system monitoring, security auditing with CIS benchmarks, and vulnerability scanning. For more information on Rancher security, please see the security documentation.

NeuVector can be enabled through a Helm chart that may be installed either through **Apps** or through the **Cluster Tools** button in the Rancher UI. Once the Helm chart is installed, users can easily deploy and manage NeuVector clusters within Rancher [https://documentation.suse.com/cloudnative/security/5.4/en/rancher.html#_deploy_and_manage_suse_security_through_rancher_extensions_or_apps_marketplace].

### Installing SUSE Security with Rancher

The Harvester Helm Chart is used to manage access to the NeuVector UI in Rancher where users can navigate directly to deploy and manage their NeuVector clusters.

**To navigate to and install the NeuVector chart through Apps:**

1. Click ☰ **> Cluster Management**.
2. On the Clusters page, go to the cluster where you want to deploy NeuVector, and click **Explore**.
3. Go to **Apps › Charts**, and install **NeuVector** from the chart repo.
4. Different cluster types require different container runtimes. When configuring Helm chart values, go to the **Container Runtime** section, and select your runtime in accordance with the cluster type. Finally, click **Install** again.

Some examples are as follows:

- K3s and RKE2: `k3scontainerd`
- AKS: `containerd` for v1.19 and up
- EKS: `docker` for v1.22 and below; `containerd` for v1.23 and up
- GKE: `containerd` (see the Google docs [https://cloud.google.com/kubernetes-engine/docs/concepts/using-containerd] for more)



Only one container runtime engine may be selected at a time during installation.

**To navigate to and install the NeuVector chart through Cluster Tools:**

1. Click ☰ **> Cluster Management**.
2. On the Clusters page, go to the cluster where you want to deploy NeuVector, and click **Explore**.
3. Click on **Cluster Tools** at the bottom of the left navigation bar.
4. Repeat step 4 above to select your container runtime accordingly, then click **Install** again.

### Accessing SUSE Security from the Rancher UI

1. Navigate to the cluster explorer of the cluster where NeuVector is installed. In the left navigation bar, click **NeuVector**.
2. Click the external link to go to the NeuVector UI. Once the link is selected, users must accept the `END USER LICENSE AGREEMENT` to access the NeuVector UI.

### Uninstalling SUSE Security from the Rancher UI

**To uninstall from Apps:**

1. Click ☰ **> Cluster Management**.
2. Under **Apps**, click **Installed Apps**.
3. Under `cattle-neuvector-system`, select both the NeuVector app (and the associated CRD if desired), then click **Delete**.

**To uninstall from Cluster Tools:**

1. Click ☰ **> Cluster Management**.
2. Click on **Cluster Tools** at the bottom-left of the screen, then click on the trash can icon under the NeuVector chart. Select `Delete the CRD associated with this app` if desired, then click **Delete**.

### GitHub Repository

The NeuVector project is available here [https://github.com/neuvector/neuvector].

### Documentation

The NeuVector documentation is here [https://documentation.suse.com/cloudnative/security/].

### Architecture

The NeuVector security solution contains four types of security containers: Controllers, Enforcers, Managers, and Scanners. A special container called an All-in-One is also provided to combine the Controller, Enforcer, and Manager functions all in one container, primarily for Docker-native deployments. There is also an Updater which, when run, will update the CVE database.

- **Controller:** Manages the NeuVector Enforcer container; provides REST APIs for the management console.
- **Enforcer:** Enforces security policies.
- **Manager:** Provides a web-UI and CLI console to manage the NeuVector platform.
- **All-in-One:** Includes the Controller, Enforcer, and Manager.
- **Scanner:** Performs the vulnerability and compliance scanning for images, containers, and nodes.
- **Updater:** Updates the CVE database for Neuvector (when run); redeploys scanner pods.

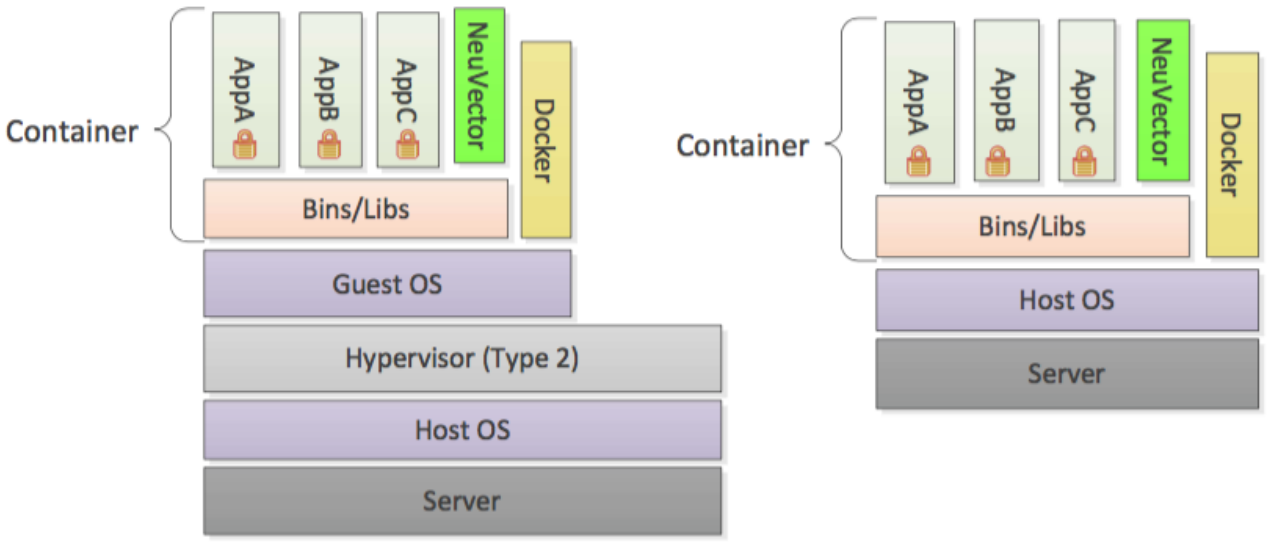



*Figure 17. NeuVector Security Containers*

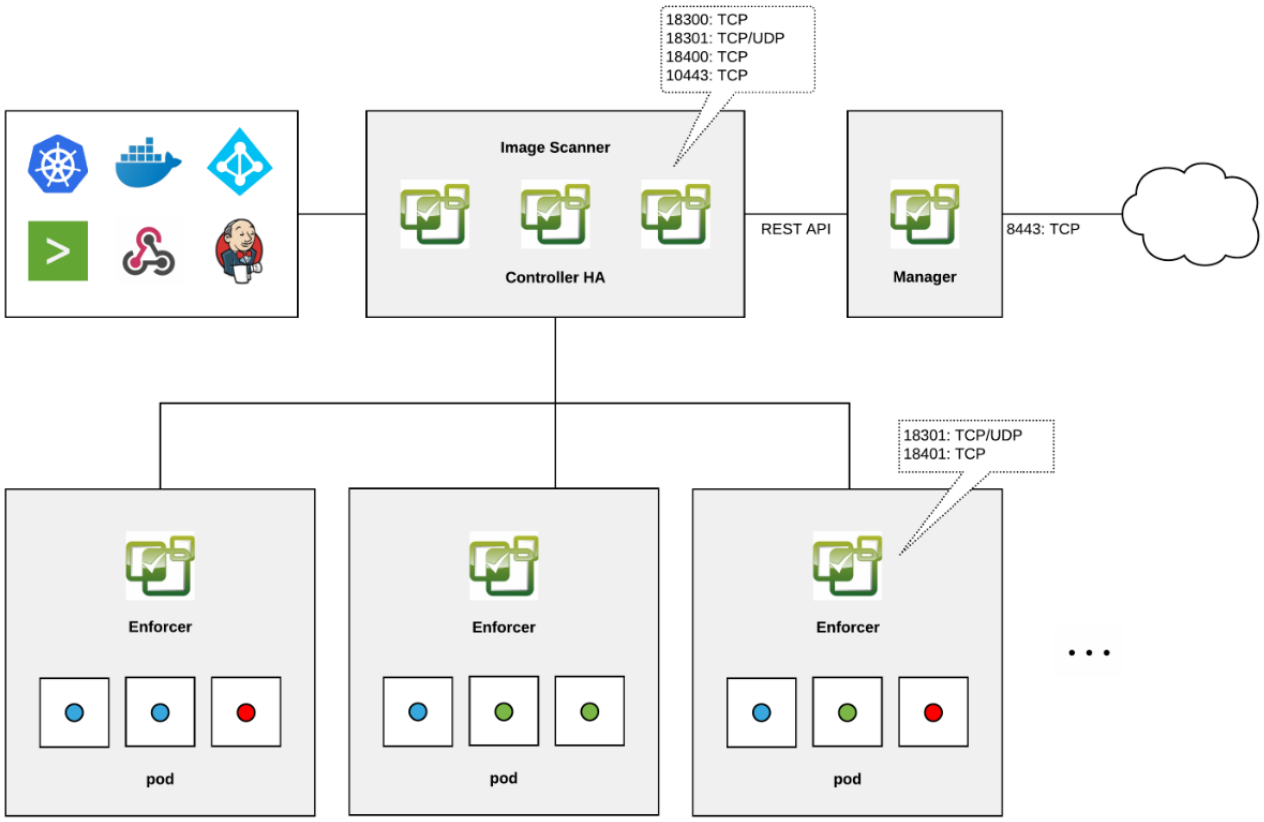



*Figure 18. NeuVector Architecture*

To learn more about NeuVector's architecture, please refer here [https://documentation.suse.com/cloudnative/security/5.4/en/overview.html#_architecture].

### CPU and Memory Allocations

Below are the minimum recommended computing resources for the NeuVector chart installation in a default deployment. Note that the resource limit is not set.

| Container | CPU - Request | Memory - Request |
|---|---|---|
| Controller | 3 (1GB 1vCPU needed per controller) | * |
| Enforcer | On all nodes (500MB .5vCPU) | 1GB |
| Manager | 1 (500MB .5vCPU) | * |
| Scanner | 3 (100MB .5vCPU) | * |

- Minimum 1GB of memory total required for Controller, Manager, and Scanner containers combined.

**Hardened Cluster Support - Calico and Canal**

**RKE2**

- NeuVector components Controller and Enforcer are deployable if PSP is set to true.

**Applicable to NeuVector chart version 100.0.0+up2.2.0 only:**

- For Manager, Scanner, and Updater components, additional configuration is required as shown below:

```
kubectl patch deploy neuvector-manager-pod -n cattle-neuvector-system
--patch '{"spec":{"template":{"spec":{"securityContext":{"runAsUser":
5400}}}}}'
kubectl patch deploy neuvector-scanner-pod -n cattle-neuvector-system
--patch '{"spec":{"template":{"spec":{"securityContext":{"runAsUser":
5400}}}}}'
kubectl patch cronjob neuvector-updater-pod -n cattle-neuvector-
system --patch
'{"spec":{"jobTemplate":{"spec":{"template":{"spec":{"securityContext
":{"runAsUser": 5400}}}}}}}'
```

You will need to set additional configuration for your hardened cluster environment.



You must update your config in both RKE2 and K3s hardened clusters as shown below.

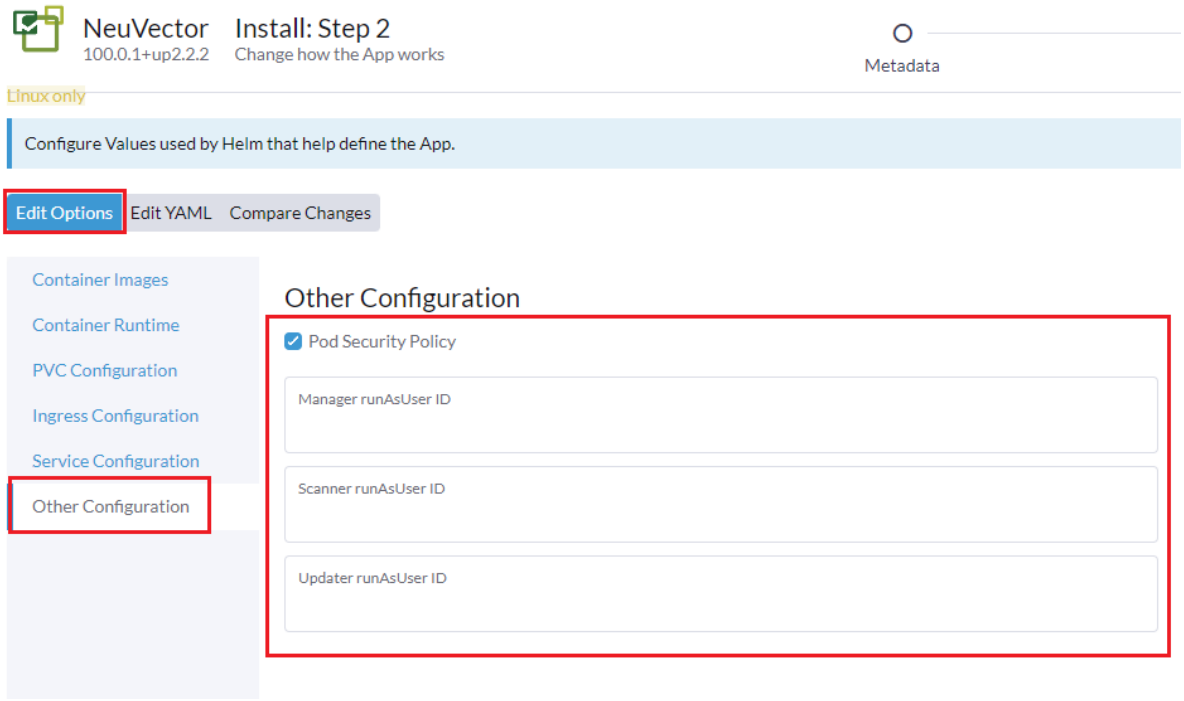
a. Click ☰ > **Cluster Management**.

b. Go to the cluster that you created and click **Explore**.

c. In the left navigation bar, click **Apps**.

d. Install (or upgrade to) NeuVector version `100.0.1+up2.2.2`.

   i. Under **Edit Options** > **Other Configuration**, enable **Pod Security Policy** by checking the box. Note that you must also enter a value greater than `zero` for `Manager runAsUser ID`, `Scanner runAsUser ID`, and `Updater runAsUser ID`:




e. Click **Install** at the bottom-right to complete.

**SELinux-enabled Cluster Support - Calico and Canal**

To enable SELinux on RKE2 clusters, follow the steps below:

- NeuVector components Controller and Enforcer are deployable if PSP is set to true.

**Applicable to NeuVector chart version 100.0.0+up2.2.0 only:**

- For Manager, Scanner, and Updater components, additional configuration is required as shown below:

```
kubectl patch deploy neuvector-manager-pod -n cattle-neuvector-system
--patch '{"spec":{"template":{"spec":{"securityContext":{"runAsUser":
5400}}}}}'
kubectl patch deploy neuvector-scanner-pod -n cattle-neuvector-system
--patch '{"spec":{"template":{"spec":{"securityContext":{"runAsUser":
5400}}}}}'
kubectl patch cronjob neuvector-updater-pod -n cattle-neuvector-system
--patch
'{"spec":{"jobTemplate":{"spec":{"template":{"spec":{"securityContext":{
"runAsUser": 5400}}}}}}}'
```

**Cluster Support in an Air-Gapped Environment**

- All NeuVector components are deployable on a cluster in an air-gapped environment without any additional configuration needed.

**Support Limitations**

- Only admins and cluster owners are currently supported.

- Fleet multi-cluster deployment is not supported.
- NeuVector is not supported on a Windows cluster.

**Other Limitations**

- Currently, NeuVector feature chart installation fails when a NeuVector partner chart already exists. To work around this issue, uninstall the NeuVector partner chart and reinstall the NeuVector feature chart.
- Sometimes when the controllers are not ready, the NeuVector UI is not accessible from the Rancher UI. During this time, controllers will try to restart, and it takes a few minutes for the controllers to be active.
- Container runtime is not auto-detected for different cluster types when installing the NeuVector chart. To work around this, you can specify the runtime manually.

# SUSE Security Admission Controller

Kubewarden is a Policy Engine that secures and helps manage your cluster resources. It allows for validation and mutation of resource requests via policies, including context-aware policies and verifying image signatures. It can run policies in monitor or enforcing mode and provides an overview of the state of the cluster.

Kubewarden aims to be the Universal Policy Engine by enabling and simplifying Policy as Code. Kubewarden policies are compiled into WebAssembly: they are small (400KBs ~ 2MBs), sandboxed, secure, and portable. It aims to be universal by catering to each persona in your organization:

- Policy User: manage and declare policies using Kubernetes Custom Resources, reuse existing policies written in Rego (OPA and Gatekeeper). Test the policies outside the cluster in CI/CD.
- Policy Developer: write policies in your preferred Wasm-compiling language (Rego, Go, Rust, C#, Swift, Typescript, and more to come). Reuse the ecosystem of tools, libraries, and workflows you already know.
- Policy Distributor: policies are OCI artifacts, serve them through your OCI repository and use industry standards in your infrastructure, like Software-Bill-Of-Materials and artifact signatures.
- Cluster Operator: Kubewarden is modular (OCI registry, PolicyServers, Audit Scanner, Controller). Configure your deployment to suit your needs, segregating different tenants. Get an overview of past, current, and possible violations across the cluster with the Audit Scanner and the PolicyReports.
- Kubewarden Integrator: use it as a platform to write new Kubewarden modules and custom policies.

## SUSE Security Admission Controller with Rancher

Kubewarden's upstream Helm charts are fully integrated as Rancher Apps, providing a UI for the install options. The charts also come with defaults that respect the Rancher stack (for example: not policing Rancher system namespaces), and default PolicyServer and Policies. Users have access to all Kubewarden features and can deploy PolicyServers and Policies manually by interacting with the Kubernetes API (e.g.: using kubectl).

Kubewarden provides a full replacement of the removed Kubernetes Pod Security Policies. Kubewarden also integrates with the new Pod Security Admission feature introduced by a recent version of Kubernetes by augmenting its security capabilities.

### SUSE Security Admission Controller with Cloud Native

The available Rancher UI Extension for Kubewarden integrates it into the Rancher UI. The UI Extension automates the installation and configuration of the Kubewarden stack and configures access to the policies maintained by SUSE. The UI Extension provides access to a curated catalog of ready-to-use policies. Using the UI Extension, one can browse, install, and configure these policies.

The UI Extension provides an overview of the Kubewarden stack components and their behavior. This includes access to the Kubewarden metrics and trace events. An operator can understand the impact of policies on the cluster and troubleshoot issues.

In addition, the UI Extension provides the Policy Reporter UI, which gives a visual overview of the compliance status of the Kubernetes cluster. With this UI, an operator can quickly identify all non-compliant Kubernetes resources, understand the reasons for violations and act accordingly. All of this with the support offering of Rancher Prime.

## SUSE® Rancher Prime: OS Manager

Elemental enables cloud-native host management. Elemental allows you to onboard any machine in any location, whether its in a datacenter or on the edge, and integrate them seamlessly into Kubernetes while managing your workflows (e.g., OS updates).

### SUSE® Rancher Prime: OS Manager with Rancher

Elemental in Rancher:

- Is Kubernetes native, which allows you to manage the OS via Elemental in Kubernetes clusters.
- Is nondisruptive from a Kubernetes operational perspective.
- Is declarative and GitOps friendly.
- Allows OCI Image-based flows, which are trusted, deterministic, and predictable.
- Works at scale. It enables fleet-sized OS management.

#### When should I use SUSE® Rancher Prime: OS Manager?

- Elemental enables cloud-native OS management from Rancher manager. It works with any OS (e.g., SLE Micro vanilla).
- Elemental allows cloud-native management for machines in datacenters and on the edge.
- Elemental is flexible and allows platform teams to perform all kind of workflows across their fleet of machines.

### SUSE® Rancher Prime: OS Manager with Cloud Native

- Deeply integrated already as GUI Extension in Rancher.
- Extends the Rancher story to the OS. Working perfectly with SLE Micro for Rancher today.

## SUSE® Rancher Prime: Continuous Delivery

Fleet orchestrates and manages the continuous delivery of applications through the supply chain for fleets of

clusters. Fleet organizes the supply chain to help teams deliver with confidence and trust in a timely manner using GitOps as a safe operating model.

## SUSE® Rancher Prime: Continuous Delivery with Rancher

Many users often manage over 10 clusters at a time. Given the proliferation of clusters, continuous delivery is an important part of Rancher. Fleet ensures a reliable continuous delivery experience using GitOps, which is a safe and increasingly common operating model.

#### When should I use SUSE® Rancher Prime: Continuous Delivery?

- I need to deploy my monitoring stack (e.g., Grafana, Prometheus) across geographical regions, each with different retention policies.
- I am a platform operator and want to provision clusters with all components using a scalable and safe operating model (GitOps).
- I am an application developer and want my latest changes to automatically go into my development environment.

## SUSE® Rancher Prime: Continuous Delivery with Cloud Native

Fleet is already deeply integrated as the Continuous Delivery tool and GitOps Engine in Rancher.

## Overview

Continuous Delivery with Fleet is GitOps at scale. Fleet is designed to manage up to a million clusters. It's also lightweight enough that it works great for a single cluster [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-operators/installation.html#_default_installation] too, but it really shines when you get to a large scale [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-operators/installation.html#_configuration_for_multi_cluster]. By large scale we mean either a lot of clusters, a lot of deployments, or a lot of teams in a single organization.

Fleet is a separate project from Rancher, and can be installed on any Kubernetes cluster with Helm.

#### Architecture

For information about how Fleet works, see the Architecture page.

#### Accessing SUSE® Rancher Prime: Continuous Delivery in the Rancher UI

Fleet comes preinstalled in Rancher and is managed by the **Continuous Delivery** option in the Rancher UI. For additional information on Continuous Delivery and other Fleet troubleshooting tips, refer here [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/troubleshooting.html].

Users can leverage continuous delivery to deploy their applications to the Kubernetes clusters in the git repository without any manual operation by following **gitops** practice.

Follow the steps below to access Continuous Delivery in the Rancher UI:

1. Click ☰ **> Continuous Delivery**.
2. Select your namespace at the top of the menu, noting the following:
    - By default, **fleet-default** is selected which includes all downstream clusters that are registered

through Rancher.

- You may switch to **fleet-local**, which only contains the **local** cluster, or you may create your own workspace to which you may assign and move clusters.
- You can then manage clusters by clicking on **Clusters** on the left navigation bar.

3. Click on **Gitrepos** on the left navigation bar to deploy the gitrepo into your clusters in the current workspace.
4. Select your git repository [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-users/gitrepo-add.html] and target clusters/cluster group [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-users/gitrepo-targets.html]. You can also create the cluster group in the UI by clicking on **Cluster Groups** from the left navigation bar.
5. Once the gitrepo is deployed, you can monitor the application through the Rancher UI.

#### Windows Support

For details on support for clusters with Windows nodes, see the Windows Support page.

#### GitHub Repository

The Fleet Helm charts are available here [https://github.com/rancher/fleet/releases].

#### Using SUSE® Rancher Prime: Continuous Delivery Behind a Proxy

For details on using Fleet behind a proxy, see the Using Fleet Behind a Proxy page.

#### Helm Chart Dependencies

In order for Helm charts with dependencies to deploy successfully, you must run a manual command (as listed below), as it is up to the user to fulfill the dependency list. If you do not do this and proceed to clone your repository and run `helm install`, your installation will fail because the dependencies will be missing.

The Helm chart in the git repository must include its dependencies in the `charts/` subdirectory. To do this, manually run `helm dependencies update $chart` or run `helm dependencies build $chart` locally, then commit the complete `charts/` directory to your git repository. Ensure you update your commands with the applicable parameters to your environment.

#### Fleet resource configuration

Fleet allows you to configuring CPU and memory resource requests and limits for both management cluster controllers and downstream cluster agents. When running Fleet through Rancher, apply these changes using the `rancher-config` ConfigMap rather than via Helm directly.

For detailed configuration options, refer to Resource Limits in Fleet [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/how-tos-for-operators/resource-limits.html].

#### Troubleshooting

- **Known Issue**: clientSecretName and helmSecretName secrets for Fleet gitrepos are not included in the backup nor restore created by the backup-restore-operator. We will update the community once a permanent solution is in place.
- **Temporary Workaround**: By default, user-defined secrets are not backed up in Fleet. It is necessary to

recreate secrets if performing a disaster recovery restore or migration of Rancher into a fresh cluster. To modify resourceSet to include extra resources you want to backup, refer to docs here [https://github.com/rancher/backup-restore-operator#user-flow].

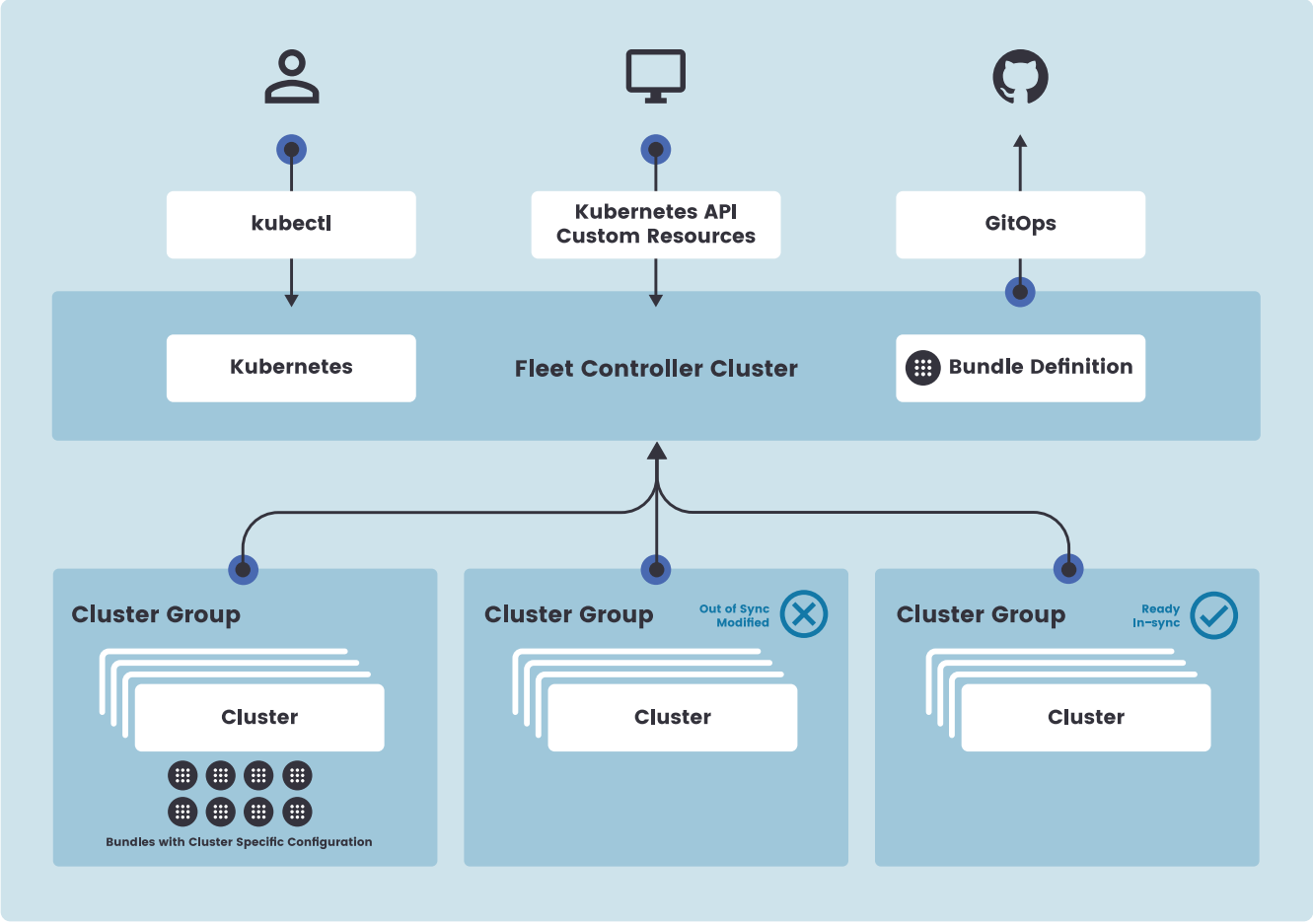
- **Debug logging**: To enable debug logging of Fleet components, create a new **fleet** entry in the existing **rancher-config** ConfigMap in the **cattle-system** namespace with the value `{"debug": 1, "debugLevel": 1}`. The Fleet application restarts after you save the ConfigMap.

#### Documentation

See the SUSE® Rancher Prime: Continuous Delivery documentation [https://documentation.suse.com/cloudnative/continuous-delivery/v0.16/en/index.html] to learn more.

## SUSE® Rancher Prime: Continuous Delivery Architecture

Fleet can manage deployments from git of raw Kubernetes YAML, Helm charts, or Kustomize or any combination of the three. Regardless of the source, all resources are dynamically turned into Helm charts, and Helm is used as the engine to deploy everything in the cluster. This gives you a high degree of control, consistency, and auditability. Fleet focuses not only on the ability to scale, but to give one a high degree of control and visibility to exactly what is installed on the cluster.




## Using SUSE® Rancher Prime: Continuous Delivery Behind a Proxy

In this section, you'll learn how to enable Fleet in a setup that has a Rancher server with a public IP a Kubernetes cluster that has no public IP, but is configured to use a proxy.

Rancher does not establish connections with registered downstream clusters. The Rancher agent deployed

on the downstream cluster must be able to establish the connection with Rancher.

To set up Fleet to work behind a proxy, you will need to set the **Agent Environment Variables** for the downstream cluster. These are cluster-level configuration options.

Through the Rancher UI, you can configure these environment variables for any cluster type, including registered and custom clusters. The variables can be added while editing an existing cluster or while provisioning a new cluster.

For public downstream clusters, it is sufficient to set the required environment variables in the Rancher UI.

For private nodes or private clusters, the environment variables need to be set on the nodes themselves. Then the environment variables are configured from the Rancher UI, typically when provisioning a custom cluster or when registering the private cluster. For an example of how to set the environment variables on Ubuntu node in a K3s Kubernetes cluster, see this section.

#### Required Environment Variables

When adding Fleet agent environment variables for the proxy, replace `<PROXY_IP>` with your private proxy IP.



The `NO_PROXY` environment variable is not standardized, and the accepted format of the value can differ between applications. When configuring the `NO_PROXY` variable for Rancher, the value must adhere to the format expected by Golang.

Specifically, the value should be a comma-delimited string which only contains IP addresses, CIDR notation, domain names, or special DNS labels (e.g. `*`). For a full description of the expected value format, refer to the **upstream Golang documentation** [https://pkg.go.dev/golang.org/x/net/http/httpproxy#Config].

| Variable Name | Value |
|---|---|
| `HTTP_PROXY` | http://<PROXY_IP>:8888 |
| `HTTPS_PROXY` | https://<PROXY_IP>:8888 |
| `NO_PROXY` | 127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16,.svc,.cluster.local |

#### Setting Environment Variables in the Rancher UI

To add the environment variable to an existing cluster:

**RKE2/K3s**

1. Click ☰ **> Cluster Management**.
2. Go to the cluster where you want to add environment variables and click ☰ **> Edit Config**.
3. Click **Agent Environment Vars** under **Cluster configuration**.
4. Click **Add**.
5. Enter the required environment variables
6. Click **Save**.

**RKE**

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where you want to add environment variables and click ☰ > **Edit Config**.
3. Click **Advanced Options**.
4. Click **Add Environment Variable**.
5. Enter the required environment variables
6. Click **Save**.

**Result:** The Fleet agent works behind a proxy.

#### Setting Environment Variables on Private Nodes

For private nodes and private clusters, the proxy environment variables need to be set on the nodes themselves, as well as configured from the Rancher UI.

This example shows how the environment variables would be set up on an Ubuntu node in a K3s Kubernetes cluster:

```
ssh -o ForwardAgent=yes ubuntu@<public_proxy_ip>
ssh <k3s_ip>
export proxy_private_ip=<private_proxy_ip>
export HTTP_PROXY=http://${proxy_private_ip}:8888
export HTTPS_PROXY=http://${proxy_private_ip}:8888
export NO_PROXY
=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16,.svc,.cluster.local
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
```

## Deploying applications from the SUSE Application Collection

SUSE® Rancher Prime: Continuous Delivery provides a dedicated deployment flow for the **SUSE Application Collection**, serving as a third option alongside GitRepos and standard HelmOps under **App Bundles**. Built on a specialized HelmOps configuration, this flow allows you to browse, configure, and distribute curated application bundles across downstream clusters natively through the Rancher UI without creating a GitRepo.

#### Prerequisites

- SUSE Rancher Prime v2.15.1 or later.
- An active account on https://apps.rancher.io/.

#### Authenticating with the SUSE Application Collection

Before deploying applications for the first time, you must authenticate SUSE Rancher Prime with your SUSE Application Collection account:

1. Log in to https://apps.rancher.io/.
2. Select your user profile/login menu and generate a new **Access Token**.

3. Copy the access token for use in the Rancher UI setup.



Credentials are saved automatically upon initial verification. Subsequent visits to the SUSE Application Collection menu will load the application catalog without requiring you to re-enter your credentials.

#### Deploying an application bundle

##### UI workflow

1. In the left navigation bar, select the **Continuous Delivery** icon.
2. From the main menu, select **App Bundles**.
3. On the App Bundles view, select **SUSE Application Collection**.
4. If accessing the tab for the first time, enter your credentials:
    a. Enter your **Username**.
    b. Enter your **Access Token** acquired from `apps.rancher.io`.
5. After your credentials are verified, browse the application catalog and select the Helm chart you wish to install.
6. Select the desired chart **Version**.
7. Configure the target clusters, target namespaces, and any custom Helm values.
8. Click **Create**. Continuous Delivery will deploy the selected application bundle to your designated downstream clusters.

#### Secret management and image pull handling

When deploying applications from the SUSE Application Collection to downstream clusters, Continuous Delivery automatically handles container image authentication:

- **Automatic `imagePullSecrets` generation:** Continuous Delivery provisions and manages the necessary `imagePullSecrets` across target namespaces in downstream clusters.
- **Seamless image pulls:** Workload nodes pull container images directly from the registry without requiring manual credential configuration on downstream clusters.
- **Credential lifecycle:** Any updates to registry authentication details are automatically propagated down to downstream cluster secrets by Continuous Delivery.



*Do not modify SUSE Application Collection charts directly in Rancher*

Always manage applications deployed via Continuous Delivery exclusively through your Continuous Delivery App Bundle definitions. Direct modifications to Rancher-managed charts outside of Continuous Delivery will **not** be reflected in your Continuous Delivery HelmApps, leading to configuration drift and potential synchronization errors.

## Windows Support

Prior to Rancher v2.5.6, the `agent` did not have native Windows manifests on downstream clusters with Windows nodes. This would result in a failing `agent` pod for the cluster.

If you are upgrading from an older version of Rancher to v2.5.6+, you can deploy a working `agent` with the

following workflow *in the downstream cluster*:

1. Cordon all Windows nodes.
2. Apply the below toleration to the `agent` workload.
3. Uncordon all Windows nodes.
4. Delete all `agent` pods. New pods should be created with the new toleration.
5. Once the `agent` pods are running, and auto-update is enabled for Fleet, they should be updated to a Windows-compatible `agent` version.

```
tolerations:
- effect: NoSchedule
  key: cattle.io/os
  operator: Equal
  value: linux
```

# SUSE® Observability

SUSE Observability is a complete observability solution that provides deep insights into the health of your clusters and nodes, and the workloads running on them. Designed to give you clear visibility into your entire Kubernetes environment, SUSE Observability's full-stack approach allows you to seamlessly explore everything from services to infrastructure within a single platform, eliminating the need for multiple observability tools.

SUSE Observability securely collects and correlates data, offering actionable insights into both existing and potential issues in your cluster. This helps you address current problems swiftly and take preventative measures against future challenges.

The intuitive dashboards highlight problem areas and offer remediation steps, guiding you from issue identification to root cause analysis, and ultimately to resolution, in the quickest possible time.

For more information and to set up SUSE Observability in your SUSE Rancher-managed Kubernetes cluster, please refer to the documentation [https://documentation.suse.com/en-us/cloudnative/suse-observability/].

# Kubernetes on the Desktop with Rancher Desktop

Rancher Desktop bundles together essential tools for developing and testing cloud-native applications from your desktop.

If you're working from your local machine on apps intended for cloud environments, you normally need a lot of preparation. You need to select a container run-time, install Kubernetes and popular utilities, and possibly set up a virtual machine. Installing components individually and getting them to work together can be a time-consuming process.

To reduce the complexity, Rancher Desktop offers teams the following key features:

- Simple and easy installation on macOS, Linux and Windows operating systems.
- K3s, a ready-to-use, light-weight Kubernetes distribution.

- The ability to easily switch between Kubernetes versions.
- A GUI-based cluster dashboard powered by Rancher to explore your local cluster.
- Freedom to choose your container engine: dockerd (moby) or containerd.
- Preference settings to configure the application to suit your needs.
- Bundled tools required for your container, for Kubernetes-based development, and for operation workflows.
- Periodic updates to keep bundled tools up to date.
- Integration with popular tools/IDEs, including VS Code and Skaffold.
- Image & Registry access control.
- Support for Docker extensions.

Visit the Rancher Desktop [https://rancherdesktop.io] website and read the docs [https://docs.rancherdesktop.io/] to learn more.

To install Rancher Desktop on your machine, refer to the installation guide [https://docs.rancherdesktop.io/getting-started/installation].

### Trying Rancher on Rancher Desktop

Rancher Desktop offers the setup and tools you need to easily try out containerized, Helm-based applications. You can get started with the Rancher Kubernetes Management platform using Rancher Desktop, by following this how-to guide [https://docs.rancherdesktop.io/how-to-guides/rancher-on-rancher-desktop].

## SUSE® Rancher Prime: Cluster API

SUSE® Rancher Prime: Cluster API [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/index.html] is a Kubernetes Operator [https://kubernetes.io/docs/concepts/extend-kubernetes/operator/#operators-in-kubernetes] that manages the lifecycle of provisioned Kubernetes clusters by integrating Cluster API (CAPI) and Rancher where it is automatically deployed as a Rancher system component. With SUSE® Rancher Prime: Cluster API, you can:

- Import CAPI clusters into Rancher, by installing the Rancher Cluster Agent in CAPI provisioned clusters.
- Manage the CAPI Operator [https://cluster-api-operator.sigs.k8s.io/] using the CAPIProvider [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/reference/capiprovider.html] resource.

The Overview section outlines installing CAPI providers and further information including Supply-chain Levels for Software Artifacts (SLSA) security compliance. For more details, see the SUSE® Rancher Prime: Cluster API documentation [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/index.html].

### Overview



The core CAPI controller manifest is embedded in a ConfigMap, which simplifies operation in air-gapped environments.

SUSE® Rancher Prime: Cluster API is automatically deployed as a system component in the `cattle-turtles-system` namespace. This includes:

- The `rancher-turtles-controller-manager` pod, which handles all CAPI-related resources.
- Core CAPI CRDs and the core CAPI controller.
- The `CAPIProvider` custom resource, which allows declarative definition of CAPI providers.

#### Security

As defined by Supply-chain Levels for Software Artifacts (SLSA) [https://slsa.dev/spec/v1.0/about], SLSA is a set of incrementally adoptable guidelines for supply chain security, established by industry consensus. The specification set by SLSA is useful for both software producers and consumers: producers can follow SLSA's guidelines to make their software supply chain more secure, and consumers can use SLSA to make decisions about whether to trust a software package.

SUSE® Rancher Prime: Cluster API meets SLSA Level 3 [https://slsa.dev/spec/v1.0/levels#build-l3] requirements as an appropriate hardened build platform, with consistent build processes, and provenance distribution. For more information, visit the SUSE® Rancher Prime: Cluster API Security [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/security/slsa.html] document.

#### Installing CAPI Providers

With SUSE® Rancher Prime: Cluster API embedded in Rancher, CAPI providers are no longer bundled by default. Providers are managed declaratively using the `CAPIProvider` custom resource.



This is a required component and must be installed for cluster provisioning functionality to work. It is not recommended that you uninstall SUSE® Rancher Prime: Cluster API.

A separate providers chart for Prime users is available for installing certified providers. For more information, refer to Install certified providers [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/user/providers.html].

For example, to define a provider:

```
apiVersion: turtles-capi.cattle.io/v1alpha1
kind: CAPIProvider
metadata:
  name: aws
  namespace: cattle-turtles-system
spec:
  type: infrastructure
  name: aws
```

For a full list of configuration options, see the CAPIProvider reference [https://documentation.suse.com/cloudnative/cluster-api/v0.27/en/reference/capiprovider.html]. For more information about setting up providers, see Configuring Native CAPI Infrastructure Providers to Provision RKE2 Clusters.

# Observability

## Cluster Tools for Logging, Monitoring, and Visibility

Rancher contains a variety of tools that aren't included in Kubernetes to assist in your DevOps operations. Rancher can integrate with external services to help your clusters run more efficiently. Tools are divided into following categories:

### Logging

Logging is helpful because it allows you to:

- Capture and analyze the state of your cluster
- Look for trends in your environment
- Save your logs to a safe location outside of your cluster
- Stay informed of events like a container crashing, a pod eviction, or a node dying
- More easily debug and troubleshoot problems

Rancher can integrate with Elasticsearch, splunk, kafka, syslog, and fluentd.

For more information, refer to the logging documentation here.

### Monitoring and Alerts

Using Rancher, you can monitor the state and processes of your cluster nodes, Kubernetes components, and software deployments through integration with Prometheus [https://prometheus.io/], a leading open-source monitoring solution.

After monitoring is enabled, you can set up alerts and notifiers that provide the mechanism to receive them.

Notifiers are services that inform you of alert events. You can configure notifiers to send alert notifications to staff best suited to take corrective action. Notifications can be sent with Slack, email, PagerDuty, WeChat, and webhooks.

Alerts are rules that trigger those notifications. Before you can receive alerts, you must configure one or more notifier in Rancher. The scope for alerts can be set at either the cluster or project level.

For more information, refer to the monitoring documentation here.

### Istio

Istio [https://istio.io/] is an open-source tool that makes it easier for DevOps teams to observe, control, troubleshoot, and secure the traffic within a complex network of microservices.

Rancher's integration with Istio was improved in Rancher v2.5.

For more information, refer to the Istio documentation here.

### Compliance Scans

Rancher can run a security scan to check whether Kubernetes is deployed according to security best practices as defined in most recognized Kubernetes Security Benchmarks, such as STIG.

For more information, refer to the Compliance scan documentation here.

## Project Tools for Logging, Monitoring, and Visibility

Rancher contains a variety of tools that aren't included in Kubernetes to assist in your DevOps operations. Rancher can integrate with external services to help your clusters run more efficiently.

### Notifiers and Alerts

Notifiers and alerts are two features that work together to inform you of events in the Rancher system. Before they can be enabled, the monitoring application must be installed.

Notifiers are services that inform you of alert events. You can configure notifiers to send alert notifications to staff best suited to take corrective action. Notifications can be sent with Slack, email, PagerDuty, WeChat, and webhooks.

Alerts are rules that trigger those notifications. Before you can receive alerts, you must configure one or more notifier in Rancher. The scope for alerts can be set at either the cluster or project level.

### Logging

Logging is helpful because it allows you to:

- Capture and analyze the state of your cluster
- Look for trends in your environment
- Save your logs to a safe location outside of your cluster
- Stay informed of events like a container crashing, a pod eviction, or a node dying
- More easily debug and troubleshoot problems

Rancher can integrate with Elasticsearch, splunk, kafka, syslog, and fluentd.

For details, refer to the logging section.

### Monitoring

Using Rancher, you can monitor the state and processes of your cluster nodes, Kubernetes components, and software deployments through integration with Prometheus [https://prometheus.io/], a leading open-source monitoring solution. For details, refer to the monitoring section.

## SUSE Rancher Prime Integration with Logging Services

The Logging operator [https://kube-logging.github.io/docs/] now powers Rancher's logging solution in place of the former, in-house solution.

## Enabling Logging

You can enable the logging for a Rancher managed cluster by going to the Apps page and installing the logging app.

1. Go to the cluster where you want to install logging and click **Apps**.
2. Click the **Logging** app.
3. Scroll to the bottom of the Helm chart README and click **Install**.

**Result:** The logging app is deployed in the `cattle-logging-system` namespace.

## Uninstall Logging

1. Go to the cluster where you want to install logging and click **Apps**.
2. Click **Installed Apps**.
3. Go to the `cattle-logging-system` namespace and check the boxes for `rancher-logging` and `rancher-logging-crd`.
4. Click **Delete**.
5. Confirm **Delete**.

**Result** `rancher-logging` is uninstalled.

## Architecture

For more information about how the logging application works, see this section.

## Role-based Access Control

Rancher logging has two roles, `logging-admin` and `logging-view`. For more information on how and when to use these roles, see this page.

## Configuring Logging Custom Resources

To manage `Flows`, `ClusterFlows`, `Outputs`, and `ClusterOutputs`,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to configure logging custom resources and click **Explore**.
3. In the left navigation bar, click **Logging**.

### Flows and ClusterFlows

For help with configuring `Flows` and `ClusterFlows`, see this page.

### Outputs and ClusterOutputs

For help with configuring `Outputs` and `ClusterOutputs`, see this page.

### Using a Custom HostTailer Image

To use a custom image for the `HostTailer` resource, you need to specify the image in the `containerOverrides` section of each `fileTailer` of the `HostTailer` resource.

```
apiVersion: logging-extensions.banzaicloud.io/v1alpha1
kind: HostTailer
metadata:
  name: cluster-system-log
spec:
  workloadMetaOverrides:
    annotations: {}
    labels: {}
  fileTailers:
    - disabled: false
      name: kubelet-log
      path: /var/lib/rancher/rke2/agent/logs/*.log
      containerOverrides:
        image: <your_registry>/<your_image>:<your_tag>
    - disabled: false
      name: containerd-log
      path: /var/lib/rancher/rke2/agent/containerd/*.log
      containerOverrides:
        image: <your_registry>/<your_image>:<your_tag>
    - name: kube-audit
      path: /var/log/kube-audit/audit-log.json
      disabled: false
      containerOverrides:
        image: <your_registry>/<your_image>:<your_tag>
```

## Configuring the Logging Helm Chart

For a list of options that can be configured when the logging application is installed or upgraded, see this page.

#### Windows Support

You can enable logging from Windows nodes.

#### Working with a Custom Docker Root Directory

For details on using a custom Docker root directory, see this section.

#### Working with Taints and Tolerations

For information on how to use taints and tolerations with the logging application, see this page.

#### Logging V2 with SELinux

For information on enabling the logging application for SELinux-enabled nodes, see this section.

#### Additional Logging Sources

By default, Rancher collects logs for control plane components and node components for all cluster types. In some cases additional logs can be collected. For details, see this section.

## Troubleshooting

#### Resource Exhaustion of `inotify` Watchers and File Descriptors

When enabling the **Logging** app on Linux systems that heavily monitor the filesystem, you may encounter `Too many open files` or `CrashLoopBackOff` failures related to applications that leverage `inotify` to watch for file changes.

This happens because the Linux kernel caps the number of files a user can open and the number of directory paths a subsystem can watch simultaneously. To resolve this, you must explicitly increase your `inotify` system limits.

Below are example commands an admin user can run to increase system limits for `inotify` user instances and watches:

```
sysctl -w fs.inotify.max_user_instances=8192
sysctl -w fs.inotify.max_user_watches=524288
```

#### The Logging Buffer Overloads Pods

Depending on your configuration, the default buffer size may be too large and cause pod failures. One way to reduce the load is to lower the logger's flush interval. This prevents logs from overfilling the buffer. You can also add more flush threads to handle moments when many logs are attempting to fill the buffer at once.

For a more complete description of how to configure the logging buffer to suit your organization's needs, see the official Logging operator documentation on buffers [https://kube-logging.github.io/docs/configuration/plugins/outputs/buffer/] and on Fluentd configuration [https://kube-logging.github.io/docs/logging-infrastructure/fluentd/].

#### The `cattle-logging` Namespace Being Recreated

If your cluster previously deployed logging from the global view in the legacy Rancher UI, you may encounter an issue where its `cattle-logging` namespace is continually being recreated.

The solution is to delete all `clusterloggings.management.cattle.io` and `projectloggings.management.cattle.io` custom resources from the cluster specific namespace in the management cluster. The existence of these custom resources causes Rancher to create the `cattle-logging` namespace in the downstream cluster if it does not exist.

The cluster namespace matches the cluster ID, so we need to find the cluster ID for each cluster.

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster you want to get the ID of and click **Explore**.

3. Copy the `<cluster-id>` portion from one of the URLs below. The `<cluster-id>` portion is the cluster namespace name.

```
# Cluster Management UI
https://<your-url>/c/<cluster-id>/

# Cluster Dashboard
https://<your-url>/dashboard/c/<cluster-id>/
```

Now that we have the `<cluster-id>` namespace, we can delete the CRs that cause `cattle-logging` to be continually recreated. *Warning:* ensure that logging, the version installed from the global view in the legacy Rancher UI, is not currently in use.

```
kubectl delete crd clusterloggings.management.cattle.io -n <cluster-id>
kubectl delete crd projectloggings.management.cattle.io -n <cluster-id>
```

## Logging Architecture

This section summarizes the architecture of the Rancher logging application.

For more details about how the Logging operator works, see the official documentation. [https://kube-logging.github.io/docs/#architecture]

### How the Logging Operator Works

The Logging operator automates the deployment and configuration of a Kubernetes logging pipeline. It deploys and configures a Fluent Bit DaemonSet on every node to collect container and application logs from the node file system.

Fluent Bit queries the Kubernetes API and enriches the logs with metadata about the pods, and transfers both the logs and the metadata to Fluentd. Fluentd receives, filters, and transfers logs to multiple `Outputs`.

The following custom resources are used to define how logs are filtered and sent to their `Outputs`:

- A `Flow` is a namespaced custom resource that uses filters and selectors to route log messages to the appropriate `Outputs`.
- A `ClusterFlow` is used to route cluster-level log messages.
- An `Output` is a namespaced resource that defines where the log messages are sent.
- A `ClusterOutput` defines an `Output` that is available from all `Flows` and `ClusterFlows`.

Each `Flow` must reference an `Output`, and each `ClusterFlow` must reference a `ClusterOutput`.

The following figure from the Logging Operator documentation [https://kube-logging.github.io/docs/#architecture] shows the new logging architecture:

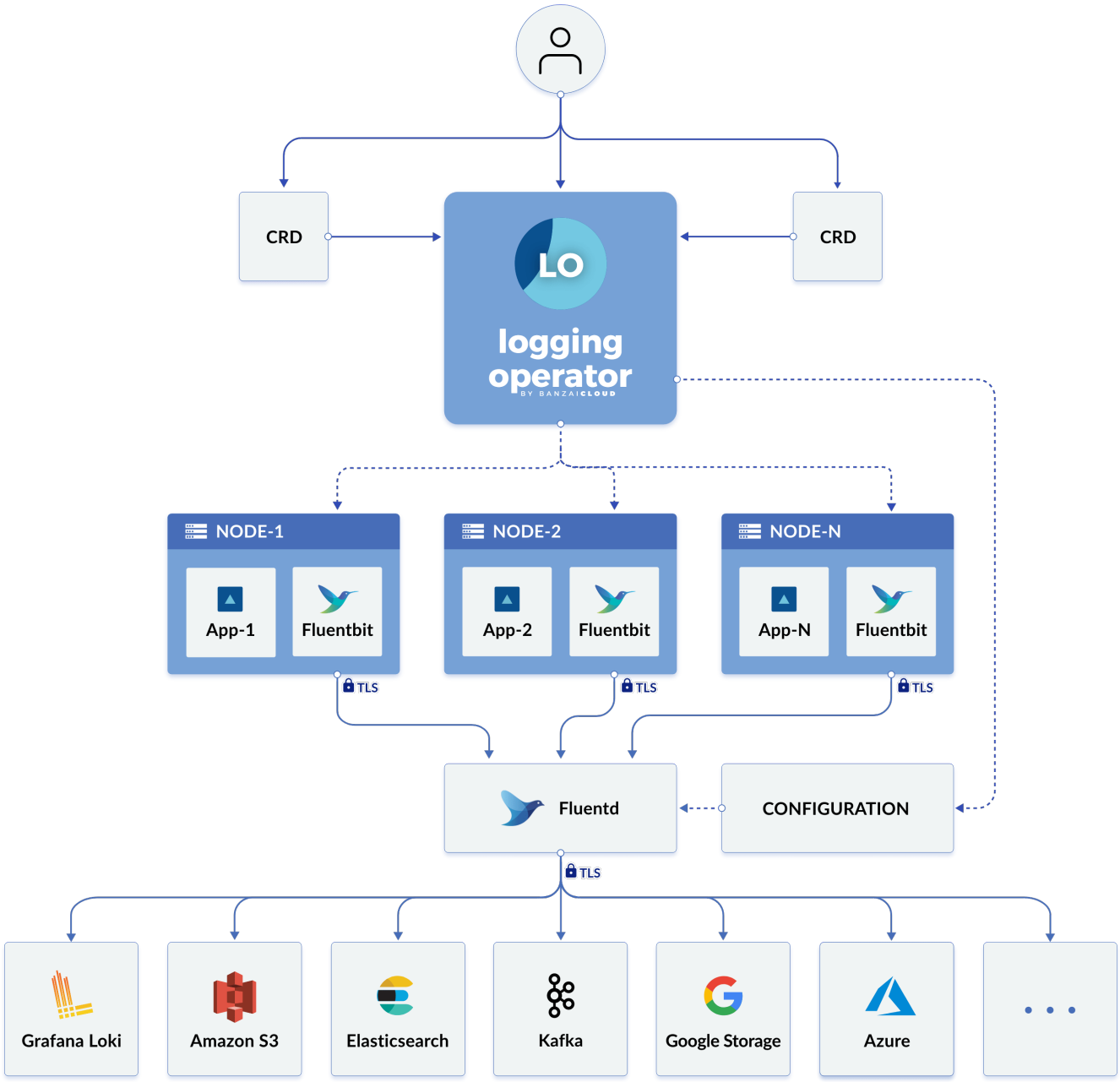



*Figure 19. How the Logging Operator Works with Fluentd and Fluent Bit*

## Logging Best Practices

In this guide, we recommend best practices for cluster-level logging and application logging.

- Cluster-level Logging
- Application Logging
- General Best Practices

Before Rancher v2.5, logging in Rancher has historically been a pretty static integration. There were a fixed list of aggregators to choose from (ElasticSearch, Splunk, Kafka, Fluentd and Syslog), and only two configuration points to choose (Cluster-level and Project-level).

Rancher provides a flexible experience for log aggregation. With the logging feature, administrators and users alike can deploy logging that meets fine-grained collection criteria while offering a wider array of destinations and configuration options.

"Under the hood", Rancher logging uses the Logging operator [https://github.com/kube-logging/logging-operator]. We provide manageability of this operator (and its resources), and tie that experience in with managing your Rancher clusters.

### Cluster-level Logging

#### Cluster-wide Scraping

For some users, it is desirable to scrape logs from every container running in the cluster. This usually coincides with your security team's request (or requirement) to collect all logs from all points of execution.

In this scenario, it is recommended to create at least two *ClusterOutput* objects - one for your security team (if you have that requirement), and one for yourselves, the cluster administrators. When creating these objects take care to choose an output endpoint that can handle the significant log traffic coming from the entire cluster. Also make sure to choose an appropriate index to receive all these logs.

Once you have created these *ClusterOutput* objects, create a *ClusterFlow* to collect all the logs. Do not define any *Include* or *Exclude* rules on this flow. This will ensure that all logs from across the cluster are collected. If you have two *ClusterOutputs*, make sure to send logs to both of them.

#### Kubernetes Components

*ClusterFlows* have the ability to collect logs from all containers on all hosts in the Kubernetes cluster. This works well in cases where those containers are part of a Kubernetes pod.

A future release of Rancher will include the source container name which will enable filtering of these component logs. Once that change is made, you will be able to customize a *ClusterFlow* to retrieve **only** the Kubernetes component logs, and direct them to an appropriate output.

### Application Logging

Best practice not only in Kubernetes but in all container-based applications is to direct application logs to `stdout`/`stderr`. The container runtime will then trap these logs and do **something** with them - typically writing them to a file. Depending on the container runtime (and its configuration), these logs can end up in any number of locations.

In the case of writing the logs to a file, Kubernetes helps by creating a `/var/log/containers` directory on each host. This directory symlinks the log files to their actual destination (which can differ based on configuration or container runtime).

Rancher logging will read all log entries in `/var/log/containers`, ensuring that all log entries from all containers (assuming a default configuration) will have the opportunity to be collected and processed.

#### Specific Log Files

Log collection only retrieves `stdout`/`stderr` logs from pods in Kubernetes. But what if we want to collect logs from other files that are generated by applications? Here, a log streaming sidecar (or two) may come in handy.

The goal of setting up a streaming sidecar is to take log files that are written to disk, and have their contents streamed to `stdout`. This way, the Logging Operator can pick up those logs and send them to your desired output.

To set this up, edit your workload resource (e.g. Deployment) and add the following sidecar definition:

```
...
containers:
- args:
  - -F
  - /path/to/your/log/file.log
  command:
  - tail
  image: busybox
  name: stream-log-file-[name]
  volumeMounts:
  - mountPath: /path/to/your/log
    name: mounted-log
...
```

This will add a container to your workload definition that will now stream the contents of (in this example) `/path/to/your/log/file.log` to `stdout`.

This log stream is then automatically collected according to any *Flows* or *ClusterFlows* you have setup. You may also wish to consider creating a *Flow* specifically for this log file by targeting the name of the container. See example:

```
...
spec:
  match:
  - select:
      container_names:
      - stream-log-file-name
...
```

**General Best Practices**

- Where possible, output structured log entries (e.g. `syslog`, JSON). This makes handling of the log entry easier as there are already parsers written for these formats.
- Try to provide the name of the application that is creating the log entry, in the entry itself. This can make troubleshooting easier as Kubernetes objects do not always carry the name of the application as the object name. For instance, a pod ID may be something like `myapp-098kjhsdf098sdf98` which does not provide much information about the application running inside the container.
- Except in the case of collecting all logs cluster-wide, try to scope your *Flow* and *ClusterFlow* objects tightly. This makes it easier to troubleshoot when problems arise, and also helps ensure unrelated log entries do not show up in your aggregator. An example of tight scoping would be to constrain a *Flow* to a single *Deployment* in a namespace, or perhaps even a single container within a *Pod*.
- Keep the log verbosity down except when troubleshooting. High log verbosity poses a number of issues, chief among them being **noise**: significant events can be drowned out in a sea of `DEBUG` messages. This is somewhat mitigated with automated alerting and scripting, but highly verbose logging still places an inordinate amount of stress on the logging infrastructure.

- Where possible, try to provide a transaction or request ID with the log entry. This can make tracing application activity across multiple log sources easier, especially when dealing with distributed applications.

## Role-based Access Control for Logging

Rancher logging has two roles, `logging-admin` and `logging-view`.

- `logging-admin` gives users full access to namespaced `Flows` and `Outputs`
- `logging-view` allows users to *view* namespaced `Flows` and `Outputs`, and `ClusterFlows` and `ClusterOutputs`



*Why choose one role over the other?*

Edit access to `ClusterFlow` and `ClusterOutput` resources is powerful. Any user with it has edit access for all logs in the cluster.

In Rancher, the cluster administrator role is the only role with full access to all `rancher-logging` resources. Cluster members are not able to edit or read any logging resources. Project owners and members have the following privileges:

| Project Owners | Project Members |
|---|---|
| able to create namespaced `Flows` and `Outputs` in their projects' namespaces | only able to view the `Flows` and `Outputs` in projects' namespaces |
| can collect logs from anything in their projects' namespaces | cannot collect any logs in their projects' namespaces |

Both project owners and project members require at least *one* namespace in their project to use logging. If they do not, then they may not see the logging button in the top nav dropdown.

## rancher-logging Helm Chart Options

### Enable/Disable Windows Node Logging

You can enable or disable Windows node logging by setting `global.cattle.windows.enabled` to either `true` or `false` in the `values.yaml`.

By default, Windows node logging will be enabled if the Cluster Dashboard UI is used to install the logging application on a Windows cluster.

In this scenario, setting `global.cattle.windows.enabled` to `false` will disable Windows node logging on the cluster. When disabled, logs will still be collected from Linux nodes within the Windows cluster.



Currently an issue [https://github.com/rancher/rancher/issues/32325] exists where Windows nodeAgents are not deleted when performing a `helm upgrade` after disabling Windows logging in a Windows cluster. In this scenario, users may need to manually remove the Windows nodeAgents if they are already installed.

### Working with a Custom Docker Root Directory

If using a custom Docker root directory, you can set `global.dockerRootDirectory` in `values.yaml`.

This will ensure that the Logging CRs created will use your specified path rather than the default Docker `data-root` location.

Note that this only affects Linux nodes.

If there are any Windows nodes in the cluster, the change will not be applicable to those nodes.

#### Adding NodeSelector Settings and Tolerations for Custom Taints

You can add your own `nodeSelector` settings and add `tolerations` for additional taints by editing the logging Helm chart values. For details, see this page.

#### Enabling the Logging Application to Work with SELinux



*Requirements:*

Logging v2 was tested with SELinux on RHEL/CentOS 7 and 8.

Security-Enhanced Linux (SELinux) [https://en.wikipedia.org/wiki/Security-Enhanced_Linux] is a security enhancement to Linux. After being historically used by government agencies, SELinux is now industry standard and is enabled by default on CentOS 7 and 8.

To use Logging v2 with SELinux, we recommend installing the `rancher-selinux` RPM according to these instructions.

Then, when installing the logging application, configure the chart to be SELinux aware by changing `global.seLinux.enabled` to `true` in the `values.yaml`.

#### Additional Logging Sources

By default, Rancher collects logs for control plane components [https://kubernetes.io/docs/concepts/overview/components/#control-plane-components] and node components [https://kubernetes.io/docs/concepts/overview/components/#node-components] for all cluster types.

In some cases, Rancher may be able to collect additional logs.

The following table summarizes the sources where additional logs may be collected for each node types:

| Logging Source | Linux Nodes (including in Windows cluster) | Windows Nodes |
|---|---|---|
| RKE2 | ⬕ | |
| K3s | ⬕ | |
| AKS | ⬕ | |
| EKS | ⬕ | |
| GKE | ⬕ | |

To enable hosted Kubernetes providers as additional logging sources, enable **Enable enhanced cloud provider logging** option when installing or upgrading the Logging Helm chart.

When enabled, Rancher collects all additional node and control plane logs the provider has made available, which may vary between providers

If you're already using a cloud provider's own logging solution such as AWS CloudWatch or Google Cloud

operations suite (formerly Stackdriver), it is not necessary to enable this option as the native solution will have unrestricted access to all logs.

#### Systemd Configuration

In Rancher logging, `SystemdLogPath` must be configured for K3s and RKE2 Kubernetes distributions.

K3s and RKE2 Kubernetes distributions log to journald, which is the subsystem of systemd that is used for logging. In order to collect these logs, the `systemdLogPath` needs to be defined. While the `run/log/journal` directory is used by default, some Linux distributions do not default to this path. For example, Ubuntu defaults to `var/log/journal`. To determine your `systemdLogPath` configuration, see steps below.

#### Steps for Systemd Configuration:

- Run `cat /etc/systemd/journald.conf | grep -E ^\#?Storage | cut -d"=" -f2` on one of your nodes.
- If `persistent` is returned, your `systemdLogPath` should be `/var/log/journal`.
- If `volatile` is returned, your `systemdLogPath` should be `/run/log/journal`.
- If `auto` is returned, check if `/var/log/journal` exists.
  - If `/var/log/journal` exists, then use `/var/log/journal`.
  - If `/var/log/journal` does not exist, then use `/run/log/journal`.



If any value not described above is returned, Rancher Logging will not be able to collect control plane logs. To address this issue, you will need to perform the following actions on every control plane node:

- Set `Storage=volatile` in journald.conf.
- Reboot your machine.
- Set `systemdLogPath` to `/run/log/journal`.

## Enabling the API Audit Log to Record System Events

You can enable the API audit log to record the sequence of system events initiated by individual users. You can know what happened, when it happened, who initiated it, and what cluster it affected. When you enable this feature, all requests to the Rancher API and all responses from it are written to a log.

You can enable API Auditing during Rancher installation or upgrade.

#### Enabling API Audit Log

The Audit Log is enabled and configured by passing environment variables to the Rancher server container. See the following to enable on your installation.

- Kubernetes Install

#### API Audit Log Options

The usage below defines rules about what the audit log should record and what data it should include:

| Parameter | Description |
|---|---|
| `AUDIT_LOG_ENABLED` | `false` - Disables the audit log (default setting).<br><br>`true` - Enables the audit log. |
| `AUDIT_LEVEL` | `0` - Log request and response metadata (default setting).<br>`1` - Log request and response headers.<br>`2` - Log request body<br>`3` - Log response body. Each log level is cumulative and each subsequent level logs the previous level data. Each log transaction for a request/response pair uses the same `auditID` value.<br><br>See Audit Level Logging for a table that displays what each setting logs. |
| `AUDIT_LOG_PATH` | Log path for Rancher Server API. Default path is `/var/log/auditlog/rancher-api-audit.log`. You can mount the log directory to host.<br><br>Usage Example: `AUDIT_LOG_PATH=/my/custom/path/` |
| `AUDIT_LOG_MAXAGE` | Defined the maximum number of days to retain old audit log files. Default is 10 days. |
| `AUDIT_LOG_MAXBACKUP` | Defines the maximum number of audit log files to retain. Default is 10. |
| `AUDIT_LOG_MAXSIZE` | Defines the maximum size in megabytes of the audit log file before it gets rotated. Default size is 100M. |

#### Audit Log Levels

The following table displays what parts of API transactions are logged for each `AUDIT_LEVEL` setting.

| `AUDIT_LEVEL` Setting | Metadata | Request Headers | Response Headers | Request Body | Response Body |
|---|---|---|---|---|---|
| `0` | ⬔ | | | | |
| `1` | ⬔ | ⬔ | ⬔ | | |
| `2` | ⬔ | ⬔ | ⬔ | ⬔ | |
| `3` | ⬔ | ⬔ | ⬔ | ⬔ | ⬔ |

### Audit Log Policies

Audit log policies allow end users to configure redactions using `AuditPolicy` cluster-scoped CRs in addition to the default redactions and filters.

All configured audit log policies are additive.

Redaction policies for headers utilize a regular expression (regex) engine to redact headers, while a JSONPath engine is used to redact request/response headers.

The JSONPath engine does not support the script or filter expressions. For getting started with JSONPath expressions, a good resource to consult is Stafan Goessner's article on JSONPath [https://goessner.net/articles/JsonPath/].

The structure of an audit policy CR is as follows:

```
apiVersion: auditlog.cattle.io/v1
kind: AuditPolicy
spec:
  enabled : true # true/false
  # list of API request filters
  filters:
    - action: allow # allow/deny
        # would allow logs sent to "/foo/some/endpoint" but not "/foo"
or "/foobar".
      requestURI: "/foo/.*"
  # additionalRedactions allows configuration of redactions on headers
using `jsonpath` expressions
  additionalRedactions:
    # redacts headers based on regex expressions
    - headers:
      - "Cache.*"
      # paths redacts information from request and response bodies based
on json path expressions
      paths:
      - "$.gitCommit"
  verbosity:
    level : 0 # matches the levels in the above audit log table
    # request allows fine-grained control over which request data
    # gets included. This overrides the behaviour of the generic
verbosity.level
    request:
      headers : true # true/false
      body : true # true/false
    # response allows fine-grained control over which response data
    # gets included. This overrides the behaviour of the generic
verbosity.level
    response:
      headers : true # true/false
      body: true # true/false
```

#### Examples

The following example shows only logging requests containing `login` in the request path to the audit log path:

```
apiVersion: auditlog.cattle.io/v1
kind: AuditPolicy
metadata:
```

```
  name: filters
spec:
  filters:
    - action: deny
      requestUri: ".*"
    - action: allow
      requestUri: ".*login.*"
```

The following example shows how to redact specific fields containing `gitCommint` in request/response bodies:

```
apiVersion: auditlog.cattle.io/v1
kind: AuditPolicy
metadata:
  name: redactions
spec:
  additionalRedactions:
    - paths:
      - "$.gitCommit"
```

#### Default redactions & filters

The audit log controller comes with default built-in redactions for common sensitive information.

#### Redacted headers

Generic headers:

- `Cookie`
- `Set-Cookie`
- `X-Api-Set-Cookie-Header`
- `Authorization`
- `X-Api-Tunnel-Params`
- `X-Api-Tunnel-Token`
- `X-Api-Auth-Header`
- `X-Amz-Security-Token`

#### Redacted body fields

Generic body fields:

- `credentials`
- `applicationSecret`
- `oauthCredential`
- `serviceAccountCredential`

- spKey
- spCert
- certificate
- privateKey
- secretsEncryptionConfig
- manifestUrl
- insecureWindowsNodeCommand
- insecureNodeCommand
- insecureCommand
- command
- nodeCommand
- windowsNodeCommand
- clientRandom

Generic body regex redactor:

- `".([pP]assword|[Kk]ube[Cc]onfig|[Tt]oken)."`

#### Cluster Driver

By default, any API request with fields tied to cluster drivers will have any non `public*` or `optional*` fields redacted by the audit log controller.

#### Redacted URIs

Any endpoint containing `secrets` or `configmaps` will have relevant fields stripped from both the request and response bodies. Additionally, any endpoint containing `/v3/imports/*` will have its URI redacted.

### Viewing API Audit Logs

#### Docker Install

Share the `AUDIT_LOG_PATH` directory (Default: `/var/log/auditlog`) with the host system. The log can be parsed by standard CLI tools or forwarded on to a log collection tool like Fluentd, Filebeat, Logstash, etc.

#### Kubernetes Install

Enabling the API Audit Log with the Helm chart install will create a `rancher-audit-log` sidecar container in the Rancher pod. This container will stream the log to standard output (stdout). You can view the log as you would any container log.

The `rancher-audit-log` container is part of the `rancher` pod in the `cattle-system` namespace.

#### CLI

```
kubectl -n cattle-system logs -f rancher-84d886bdbb-s4s69 rancher-audit-log
```

#### Shipping the Audit Log

You can enable Rancher's built in log collection and shipping for the cluster to ship the audit and other services logs to a supported collection endpoint. See Rancher Tools - Logging for details.

### Audit Log Samples

After you enable auditing, each API request or response is logged by Rancher in the form of JSON. Each of the following code samples provide examples of how to identify each API transaction.

#### Metadata Level

If you set your `AUDIT_LEVEL` to `0`, Rancher logs the metadata header for every API request, but neither the body nor the request and response headers. The metadata provides basic information about the API transaction, such as the transaction ID, the initiator of the transaction, the time it occurred, etc.


```
{
  "auditID": "40bd4e40-875b-4020-933e-4c4f4c4db366",
  "requestURI": "/v3/schemas",
  "user": {
      "name": "user-6j5s6",
      "group": [
          "system:authenticated",
          "system:cattle:authenticated"
      ],
      "extra": {
          "principalid": [
              "local://user-6j5s6"
          ],
          "requesthost": [
              "localhost:8443"
          ],
          "requesttokenid": [
              "token-zs42h"
          ],
          "username": [
              "admin"
          ]
      }
  },
  "method": "GET",
  "remoteAddr": "127.0.0.1:58652",
  "responseCode": 200,
  "requestTimestamp": "2025-06-30T11:13:25-04:00",
  "responseTimestamp": "2025-06-30T11:13:25-04:00"
}
```

#### Metadata and Headers Level

If you set your `AUDIT_LEVEL` to `1`, Rancher logs the metadata and the request and response headers for every API request.


```
{
    "auditID": "f8c83dc6-a080-4e2e-ab43-552bddf01716",
    "requestURI":
"/v1/apps.deployments?page=1&pagesize=100&sort=metadata.name&filter=meta
data.namespace!=p-npsl5&filter=metadata.namespace!=p-
nzp6c&filter=metadata.namespace!=cattle-fleet-clusters-
system&filter=metadata.namespace!=cattle-fleet-
system&filter=metadata.namespace!=cattle-global-
data&filter=metadata.namespace!=cattle-impersonation-
system&filter=metadata.namespace!=cattle-provisioning-capi-
system&filter=metadata.namespace!=cattle-
system&filter=metadata.namespace!=cattle-ui-plugin-
system&filter=metadata.namespace!=cluster-fleet-local-local-
1a3d67d0a899&filter=metadata.namespace!=fleet-
default&filter=metadata.namespace!=fleet-
local&filter=metadata.namespace!=kube-node-
lease&filter=metadata.namespace!=kube-
public&filter=metadata.namespace!=kube-
system&exclude=metadata.managedFields",
    "user": {
        "name": "user-6j5s6",
        "group": [
            "system:authenticated",
            "system:cattle:authenticated"
        ],
        "extra": {
            "principalid": [
                "local://user-6j5s6"
            ],
            "requesthost": [
                "localhost:8443"
            ],
            "requesttokenid": [
                "token-zs42h"
            ],
            "username": [
                "admin"
            ]
        }
    },
    "method": "GET",
```

```
    "remoteAddr": "127.0.0.1:58833",
    "responseCode": 200,
    "requestTimestamp": "2025-06-30T11:17:04-04:00",
    "responseTimestamp": "2025-06-30T11:17:04-04:00",
    "requestHeader": {
        "Accept": [
            "application/json"
        ],
        "Accept-Encoding": [
            "gzip, deflate, br, zstd"
        ],
        "Accept-Language": [
            "en-US,en;q=0.5"
        ],
        "Connection": [
            "keep-alive"
        ],
        "Cookie": [
            "[redacted]"
        ],
        "Referer": [

"https://localhost:8443/dashboard/c/local/explorer/apps.deployment"
        ],
        "Sec-Fetch-Dest": [
            "empty"
        ],
        "Sec-Fetch-Mode": [
            "cors"
        ],
        "Sec-Fetch-Site": [
            "same-origin"
        ],
        "User-Agent": [
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0)
Gecko/20100101 Firefox/140.0"
        ],
        "X-Api-Csrf": [
            "fccc690cab7b0c169b3fc6527edadef3"
        ]
    },
    "responseHeader": {
        "Cache-Control": [
            "no-cache, no-store, must-revalidate"
        ],
```

```
        "Content-Encoding": [
            "gzip"
        ],
        "Content-Type": [
            "application/json"
        ],
        "Expires": [
            "Wed 24 Feb 1982 18:42:00 GMT"
        ],
        "X-Api-Cattle-Auth": [
            "true"
        ],
        "X-Api-Schemas": [
            "https://localhost:8443/v1/schemas"
        ],
        "X-Content-Type-Options": [
            "nosniff"
        ]
    }
}
```

#### Metadata, Headers and Request Body Level

If you set your `AUDIT_LEVEL` to `2`, Rancher logs the metadata, the request and response headers and the request body for every API request.

The code sample below depicts an API request, with both its metadata, headers and request body.

#### Request

The code sample below depicts an API request, with both its metadata header and body.

```
{
    "auditID": "d1088a09-2a13-4450-970e-0d44bd2c49ee",
    "requestURI": "/v3/projects",
    "user": {
        "name": "user-6j5s6",
        "group": [
            "system:authenticated",
            "system:cattle:authenticated"
        ],
        "extra": {
            "principalid": [
                "local://user-6j5s6"
            ],
            "requesthost": [
```

```
                "localhost:8443"
            ],
            "requesttokenid": [
                "token-zs42h"
            ],
            "username": [
                "admin"
            ]
        }
    },
    "method": "POST",
    "remoteAddr": "127.0.0.1:49966",
    "responseCode": 201,
    "requestTimestamp": "2025-06-30T12:32:13-04:00",
    "responseTimestamp": "2025-06-30T12:32:13-04:00",
    "requestHeader": {
        "Accept": [
            "application/json"
        ],
        "Accept-Encoding": [
            "gzip, deflate, br, zstd"
        ],
        "Accept-Language": [
            "en-US,en;q=0.5"
        ],
        "Connection": [
            "keep-alive"
        ],
        "Content-Length": [
            "214"
        ],
        "Content-Type": [
            "application/json"
        ],
        "Cookie": [
            "[redacted]"
        ],
        "Impersonate-Extra-Principalid": [
            "local://user-6j5s6"
        ],
        "Impersonate-Extra-Requesthost": [
            "localhost:8443"
        ],
        "Impersonate-Extra-Requesttokenid": [
            "token-zs42h"
```

```
        ],
        "Impersonate-Extra-Username": [
            "admin"
        ],
        "Impersonate-Group": [
            "system:authenticated",
            "system:cattle:authenticated"
        ],
        "Impersonate-User": [
            "user-6j5s6"
        ],
        "Origin": [
            "https://localhost:8443"
        ],
        "Priority": [
            "u=0"
        ],
        "Referer": [
"https://localhost:8443/dashboard/c/local/explorer/management.cattle.io.project/create"
        ],
        "Sec-Fetch-Dest": [
            "empty"
        ],
        "Sec-Fetch-Mode": [
            "cors"
        ],
        "Sec-Fetch-Site": [
            "same-origin"
        ],
        "User-Agent": [
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0"
        ],
        "X-Api-Csrf": [
            "fccc690cab7b0c169b3fc6527edadef3"
        ]
    },
    "responseHeader": {
        "Cache-Control": [
            "no-cache, no-store, must-revalidate"
        ],
        "Content-Encoding": [
            "gzip"
```

```
        ],
        "Content-Type": [
            "application/json"
        ],
        "Expires": [
            "Wed 24 Feb 1982 18:42:00 GMT"
        ],
        "X-Api-Cattle-Auth": [
            "true"
        ],
        "X-Api-Schemas": [
            "https://localhost:8443/v3/project/schemas"
        ],
        "X-Content-Type-Options": [
            "nosniff"
        ]
    },
    "requestBody": {
        "annotations": {},
        "clusterId": "local",
        "containerDefaultResourceLimit": {},
        "creatorId": "local://user-6j5s6",
        "labels": {},
        "name": "example-project",
        "namespaceDefaultResourceQuota": {},
        "resourceQuota": {},
        "type": "project"
    }
}
```


#### Metadata, Headers, Request Body and Response Body Level

If you set your `AUDIT_LEVEL` to `3`, Rancher logs the metadata, request and response headers and request body and response.

The code sample below depicts an example of an API request with that information logged.


```
{
    "auditID": "a9549a5b-4351-4bd5-adcd-12f7ec667a6b",
    "requestURI": "/v3/projects",
    "user": {
        "name": "user-6j5s6",
        "group": [
            "system:authenticated",
            "system:cattle:authenticated"
```

```
        ],
        "extra": {
            "principalid": [
                "local://user-6j5s6"
            ],
            "requesthost": [
                "localhost:8443"
            ],
            "requesttokenid": [
                "token-zs42h"
            ],
            "username": [
                "admin"
            ]
        }
    },
    "method": "POST",
    "remoteAddr": "127.0.0.1:50454",
    "responseCode": 201,
    "requestTimestamp": "2025-06-30T12:42:24-04:00",
    "responseTimestamp": "2025-06-30T12:42:24-04:00",
    "requestHeader": {
        "Accept": [
            "application/json"
        ],
        "Accept-Encoding": [
            "gzip, deflate, br, zstd"
        ],
        "Accept-Language": [
            "en-US,en;q=0.5"
        ],
        "Connection": [
            "keep-alive"
        ],
        "Content-Length": [
            "214"
        ],
        "Content-Type": [
            "application/json"
        ],
        "Cookie": [
            "[redacted]"
        ],
        "Impersonate-Extra-Principalid": [
            "local://user-6j5s6"
```

```
        ],
        "Impersonate-Extra-Requesthost": [
            "localhost:8443"
        ],
        "Impersonate-Extra-Requesttokenid": [
            "token-zs42h"
        ],
        "Impersonate-Extra-Username": [
            "admin"
        ],
        "Impersonate-Group": [
            "system:authenticated",
            "system:cattle:authenticated"
        ],
        "Impersonate-User": [
            "user-6j5s6"
        ],
        "Origin": [
            "https://localhost:8443"
        ],
        "Priority": [
            "u=0"
        ],
        "Referer": [

"https://localhost:8443/dashboard/c/local/explorer/management.cattle.io.
project/create"
        ],
        "Sec-Fetch-Dest": [
            "empty"
        ],
        "Sec-Fetch-Mode": [
            "cors"
        ],
        "Sec-Fetch-Site": [
            "same-origin"
        ],
        "User-Agent": [
            "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0)
Gecko/20100101 Firefox/140.0"
        ],
        "X-Api-Csrf": [
            "fccc690cab7b0c169b3fc6527edadef3"
        ]
    },
```

```
    "responseHeader": {
        "Cache-Control": [
            "no-cache, no-store, must-revalidate"
        ],
        "Content-Encoding": [
            "gzip"
        ],
        "Content-Type": [
            "application/json"
        ],
        "Expires": [
            "Wed 24 Feb 1982 18:42:00 GMT"
        ],
        "X-Api-Cattle-Auth": [
            "true"
        ],
        "X-Api-Schemas": [
            "https://localhost:8443/v3/project/schemas"
        ],
        "X-Content-Type-Options": [
            "nosniff"
        ]
    },
    "requestBody": {
        "annotations": {},
        "clusterId": "local",
        "containerDefaultResourceLimit": {},
        "creatorId": "local://user-6j5s6",
        "labels": {},
        "name": "example-project",
        "namespaceDefaultResourceQuota": {},
        "resourceQuota": {},
        "type": "project"
    },
    "responseBody": {
        "actions": {
            "exportYaml": "https://localhost:8443/v3/projects/local:p-
qt6tq?action=exportYaml"
        },
        "annotations": {
            "authz.management.cattle.io/creator-role-bindings": "{
\"required\":[\"project-owner\"]}"
        },
        "backingNamespace": "local-p-qt6tq",
        "baseType": "project",
```

```
        "clusterId": "local",
        "containerDefaultResourceLimit": {
            "type": "/v3/schemas/containerResourceLimit"
        },
        "created": "2025-06-30T16:42:24Z",
        "createdTS": 1751301744000,
        "creatorId": "user-6j5s6",
        "id": "local:p-qt6tq",
        "labels": {
            "cattle.io/creator": "norman"
        },
        "links": {
            "basicAuths": "https://localhost:8443/v3/projects/local:p-
qt6tq/basicauths",
            "certificates": "https://localhost:8443/v3/projects/local:p-
qt6tq/certificates",
            "configMaps": "https://localhost:8443/v3/projects/local:p-
qt6tq/configmaps",
            "cronJobs": "https://localhost:8443/v3/projects/local:p-
qt6tq/cronjobs",
            "daemonSets": "https://localhost:8443/v3/projects/local:p-
qt6tq/daemonsets",
            "deployments": "https://localhost:8443/v3/projects/local:p-
qt6tq/deployments",
            "dnsRecords": "https://localhost:8443/v3/projects/local:p-
qt6tq/dnsrecords",
            "dockerCredentials":
"https://localhost:8443/v3/projects/local:p-qt6tq/dockercredentials",
            "horizontalPodAutoscalers":
"https://localhost:8443/v3/projects/local:p-
qt6tq/horizontalpodautoscalers",
            "ingresses": "https://localhost:8443/v3/projects/local:p-
qt6tq/ingresses",
            "jobs": "https://localhost:8443/v3/projects/local:p-
qt6tq/jobs",
            "namespacedBasicAuths":
"https://localhost:8443/v3/projects/local:p-qt6tq/namespacedbasicauths",
            "namespacedCertificates":
"https://localhost:8443/v3/projects/local:p-
qt6tq/namespacedcertificates",
            "namespacedDockerCredentials":
"https://localhost:8443/v3/projects/local:p-
qt6tq/namespaceddockercredentials",
            "namespacedSecrets":
"https://localhost:8443/v3/projects/local:p-qt6tq/namespacedsecrets",
```

```
            "namespacedServiceAccountTokens": "[redacted]",
            "namespacedSshAuths":
"https://localhost:8443/v3/projects/local:p-qt6tq/namespacedsshauths",
            "persistentVolumeClaims":
"https://localhost:8443/v3/projects/local:p-
qt6tq/persistentvolumeclaims",
            "pods": "https://localhost:8443/v3/projects/local:p-
qt6tq/pods",
            "projectNetworkPolicies":
"https://localhost:8443/v3/projects/local:p-
qt6tq/projectnetworkpolicies",
            "projectRoleTemplateBindings":
"https://localhost:8443/v3/projects/local:p-
qt6tq/projectroletemplatebindings",
            "remove": "https://localhost:8443/v3/projects/local:p-
qt6tq",
            "replicaSets": "https://localhost:8443/v3/projects/local:p-
qt6tq/replicasets",
            "replicationControllers":
"https://localhost:8443/v3/projects/local:p-
qt6tq/replicationcontrollers",
            "secrets": "https://localhost:8443/v3/projects/local:p-
qt6tq/secrets",
            "self": "https://localhost:8443/v3/projects/local:p-qt6tq",
            "serviceAccountTokens": "[redacted]",
            "services": "https://localhost:8443/v3/projects/local:p-
qt6tq/services",
            "sshAuths": "https://localhost:8443/v3/projects/local:p-
qt6tq/sshauths",
            "statefulSets": "https://localhost:8443/v3/projects/local:p-
qt6tq/statefulsets",
            "subscribe": "https://localhost:8443/v3/projects/local:p-
qt6tq/subscribe",
            "update": "https://localhost:8443/v3/projects/local:p-
qt6tq",
            "workloads": "https://localhost:8443/v3/projects/local:p-
qt6tq/workloads"
        },
        "name": "example-project",
        "namespaceDefaultResourceQuota": {
            "limit": {
                "type": "/v3/schemas/resourceQuotaLimit"
            },
            "type": "/v3/schemas/namespaceResourceQuota"
        },
```

```
        "namespaceId": null,
        "resourceQuota": {
            "limit": {
                "type": "/v3/schemas/resourceQuotaLimit"
            },
            "type": "/v3/schemas/projectResourceQuota",
            "usedLimit": {
                "type": "/v3/schemas/resourceQuotaLimit"
            }
        },
        "state": "active",
        "transitioning": "no",
        "transitioningMessage": "",
        "type": "project",
        "uuid": "b582603b-7826-4302-8393-792df26ll265"
    }
}
```


### Enabling the API Audit Log in Downstream Clusters

Kubernetes auditing provides a security-relevant chronological set of records about a cluster. Kube-apiserver performs auditing. Requests generate an event at each stage of its execution, which is then preprocessed according to a certain policy and written to a backend. The policy determines what's recorded and the backend persists the records.

You might want to configure the audit log as part of compliance with the Center for Internet Security (CIS) Kubernetes Benchmark controls.

For configuration details, refer to the official Kubernetes documentation [https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/].

**RKE2**

<h2><a id="_method_1_recommended_set_audit_policy_file_in_machineglobalconfig_or_machineselectorconfig"></a>Method 1 (Recommended): Set `audit-policy-file` in `machineGlobalConfig` or `machineSelectorConfig`</h2>

You can set `audit-policy-file` in the configuration file using either `machineGlobalConfig` or `machineSelectorConfig`.

When using `machineGlobalConfig`, Rancher delivers the file to the path `/var/lib/rancher/rke2/etc/config-files/audit-policy-file` on **all nodes** (both control plane and worker nodes), and sets the proper options in the RKE2 server. This may cause unwanted worker node reconciliation when the audit policy is modified.

To avoid worker node reconciliation, use `machineSelectorConfig` with a label selector to target only control plane nodes. This ensures that the audit policy file is only delivered to control plane nodes.

Example using `machineGlobalConfig`:

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  rkeConfig:
    machineGlobalConfig:
      audit-policy-file: |
        apiVersion: audit.k8s.io/v1
        kind: Policy
        rules:
          - level: RequestResponse
            resources:
              - group: ""
                resources:
                  - pods
```

Example using `machineSelectorConfig` (recommended to avoid worker node reconciliation):

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  rkeConfig:
    machineSelectorConfig:
      - config:
          audit-policy-file: |
            apiVersion: audit.k8s.io/v1
            kind: Policy
            rules:
              - level: RequestResponse
                resources:
                  - group: ""
                    resources:
                      - pods
        machineLabelSelector:
          matchLabels:
            rke.cattle.io/control-plane-role: 'true'
```

<h2><a id="_method_2_use_the_directives_machineselectorfiles_and_machineglobalconfig"></a>Method 2: Use the Directives, `machineSelectorFiles` and `machineGlobalConfig`</h2>



This feature is available in Rancher v2.7.2 and later.

You can use `machineSelectorFiles` to deliver the audit policy file to the control plane nodes, and `machineGlobalConfig` to set the options on kube-apiserver.

As a prerequisite, you must create a secret or configmap to be the source of the audit policy.

The secret or configmap must meet the following requirements:

1. It must be in the `fleet-default` namespace where the Cluster object exists.
2. It must have the annotation `rke.cattle.io/object-authorized-for-clusters: <cluster-name1>,<cluster-name2>` which permits the target clusters to use it.



Rancher Dashboard provides an easy-to-use form for creating the secret or configmap.

Example:

```
apiVersion: v1
data:
  audit-policy: >-

IyBMb2cgYWxsIHJlcXVlc3RzIGF0IHRoZSBNZXRhZGF0YSBsZXZlbC4KYXBpVmVyc2
lvbjogYXVkaXQuazhzLmlvL3YxCmtpbmQ6IFBvbGljeQpydWxlczoKLSBsZXZlbDog
TWV0YWRhdGE=
kind: Secret
metadata:
  annotations:
    rke.cattle.io/object-authorized-for-clusters: cluster1
  name: <name1>
  namespace: fleet-default
```

Enable and configure the audit log by editing the cluster in YAML, and utilizing the `machineSelectorFiles` and `machineGlobalConfig` directives.

Example:

```
apiVersion: provisioning.cattle.io/v1
kind: Cluster
spec:
  rkeConfig:
    machineGlobalConfig:
      kube-apiserver-arg:
        - audit-policy-file=<customized-path>/dev-audit-
policy.yaml
        - audit-log-path=<customized-path>/dev-audit.logs
    machineSelectorFiles:
      - fileSources:
```

```
          - configMap:
              name: ''
            secret:
              items:
                - key: audit-policy
                  path: <customized-path>/dev-audit-policy.yaml
              name: dev-audit-policy
        machineLabelSelector:
          matchLabels:
            rke.cattle.io/control-plane-role: 'true'
```

For more information about cluster configuration, refer to the RKE2 cluster configuration reference pages.

**K3s**



This feature is available in Rancher v2.7.2 and later.

You can use `machineSelectorFiles` to deliver the audit policy file to the control plane nodes, and `machineGlobalConfig` to set the options on kube-apiserver.

As a prerequisite, you must create a secret or configmap to be the source of the audit policy.

The secret or configmap must meet the following requirements:

1. It must be in the `fleet-default` namespace where the Cluster object exists.
2. It must have the annotation `rke.cattle.io/object-authorized-for-clusters: <cluster-name1>,<cluster-name2>` which permits the target clusters to use it.



Rancher Dashboard provides an easy-to-use form for creating the secret or configmap.

Example:

```
apiVersion: v1
data:
  audit-policy: >-

IyBMb2cgYWxsIHJlcXVlc3RzIGF0IHRoZSBNZXRhZGF0YSBsZXZlbC4KYXBpVmVyc2
lvbjogYXVkaXQuazhzLmlvL3YxCmtpbmQ6IFBvbGljeQpydWxlczoKLSBsZXZlbDog
TWV0YWRhdGE=
kind: Secret
metadata:
  annotations:
    rke.cattle.io/object-authorized-for-clusters: cluster1
  name: <name1>
  namespace: fleet-default
```

Enable and configure the audit log by editing the cluster in YAML, and utilizing the `machineSelectorFiles` and `machineGlobalConfig` directives.

Example:

```
apiVersion: provisioning.cattle.io/vl
kind: Cluster
spec:
  rkeConfig:
    machineGlobalConfig:
      kube-apiserver-arg:
        - audit-policy-file=<customized-path>/dev-audit-
policy.yaml
        - audit-log-path=<customized-path>/dev-audit.logs
    machineSelectorFiles:
      - fileSources:
          - configMap:
              name: ''
            secret:
              items:
                - key: audit-policy
                  path: <customized-path>/dev-audit-policy.yaml
              name: dev-audit-policy
        machineLabelSelector:
          matchLabels:
            rke.cattle.io/control-plane-role: 'true'
```

For more information about cluster configuration, refer to the K3s cluster configuration reference pages.

## Working with Taints and Tolerations

"Tainting" a Kubernetes node causes pods to repel running on that node.

Unless the pods have a `toleration` for that node's taint, they will run on other nodes in the cluster.

Taints and tolerations [https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/] can work in conjunction with the `nodeSelector` field [https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#nodeselector] within the PodSpec, which enables the *opposite* effect of a taint.

Using `nodeSelector` gives pods an affinity towards certain nodes.

Both provide choice for the what node(s) the pod will run on.

- Default Implementation in Rancher's Logging Stack
- Adding NodeSelector Settings and Tolerations for Custom Taints

#### Default Implementation in Rancher's Logging Stack

By default, Rancher taints all Linux nodes with `cattle.io/os=linux`, and does not taint Windows nodes. The logging stack pods have `tolerations` for this taint, which enables them to run on Linux nodes. Moreover, most logging stack pods run on Linux only and have a `nodeSelector` added to ensure they run on Linux nodes.

This example Pod YAML file shows a nodeSelector being used with a toleration:

```
apiVersion: vl
kind: Pod
# metadata...
spec:
  # containers...
  tolerations:
    - key: cattle.io/os
      operator: "Equal"
      value: "linux"
      effect: NoSchedule
  nodeSelector:
    kubernetes.io/os: linux
```

In the above example, we ensure that our pod only runs on Linux nodes, and we add a `toleration` for the taint we have on all of our Linux nodes.

You can do the same with Rancher's existing taints, or with your own custom ones.

#### Adding NodeSelector Settings and Tolerations for Custom Taints

If you would like to add your own `nodeSelector` settings, or if you would like to add `tolerations` for additional taints, you can pass the following to the chart's values.

```
tolerations:
  # insert tolerations...
nodeSelector:
  # insert nodeSelector...
```

These values will add both settings to the `fluentd`, `fluentbit`, and `logging-operator` containers. Essentially, these are global settings for all pods in the logging stack.

However, if you would like to add tolerations for *only* the `fluentbit` container, you can add the following to the chart's values.

```
fluentbit_tolerations:
  # insert tolerations list for fluentbit containers only...
```

## Custom Resource Configuration

### Flows and ClusterFlows

See the Logging operator documentation [https://kube-logging.github.io/docs/configuration/flow/] for the full details on how to configure `Flows` and `ClusterFlows`.

See Rancher Integration with Logging Services: Troubleshooting for how to resolve memory problems with the logging buffer.

#### Flows

A `Flow` defines which logs to collect and filter and which output to send the logs to.

The `Flow` is a namespaced resource, which means logs will only be collected from the namespace that the `Flow` is deployed in.

`Flows` can be configured by filling out forms in the Rancher UI.

For more details about the `Flow` custom resource, see FlowSpec. [https://kube-logging.github.io/docs/configuration/crds/v1beta1/flow_types/]

#### Matches

Match statements are used to select which containers to pull logs from.

You can specify match statements to select or exclude logs according to Kubernetes labels, container and host names. Match statements are evaluated in the order they are defined and processed only until the first matching select or exclude rule applies.

Matches can be configured by filling out the `Flow` or `ClusterFlow` forms in the Rancher UI.

For detailed examples on using the match statement, see the official documentation on log routing. [https://kube-logging.github.io/docs/configuration/log-routing/]

#### Filters

You can define one or more filters within a `Flow`. Filters can perform various actions on the logs, such as adding data, transforming the logs, or parsing values from the records. The filters in the `Flow` are applied in the same order they appear in the definition.

For a list of filters supported by the Logging operator, see the official documentation on Fluentd filters [https://kube-logging.github.io/docs/configuration/plugins/filters/].

Filters need to be configured in YAML.

#### Outputs

This `Output` will receive logs from the `Flow`. Because the `Flow` is a namespaced resource, the `Output` must reside in same namespace as the `Flow`.

`Outputs` can be referenced when filling out the `Flow` or `ClusterFlow` forms in the Rancher UI.

#### ClusterFlows

Matches, filters and `Outputs` are configured for `ClusterFlows` in the same way that they are configured for `Flows`. The key difference is that the `ClusterFlow` is scoped at the cluster level and can configure log collection across all namespaces.

`ClusterFlows` can be configured by filling out forms in the Rancher UI.

After `ClusterFlow` selects logs from all namespaces in the cluster, logs from the cluster will be collected and logged to the selected `ClusterOutput`.

#### YAML Example

The following example `Flow` transforms the log messages from the default namespace and sends them to an S3 `Output`:

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: Flow
metadata:
  name: flow-sample
  namespace: default
spec:
  filters:
    - parser:
        remove_key_name_field: true
        parse:
          type: nginx
    - tag_normaliser:
        format: ${namespace_name}.${pod_name}.${container_name}
  localOutputRefs:
    - s3-output
  match:
    - select:
        labels:
          app: nginx
```

### Outputs and ClusterOutputs

See the Logging operator documentation [https://kube-logging.github.io/docs/configuration/flow/] for the full details on how to configure `Flows` and `ClusterFlows`.

See Rancher Integration with Logging Services: Troubleshooting for how to resolve memory problems with the logging buffer.

#### Outputs

The `Output` resource defines where your `Flows` can send the log messages. `Outputs` are the final stage for a logging `Flow`.

The `Output` is a namespaced resource, which means only a `Flow` within the same namespace can access it.

You can use secrets in these definitions, but they must also be in the same namespace.

`Outputs` can be configured by filling out forms in the Rancher UI.

For the details of `Output` custom resource, see OutputSpec. [https://kube-logging.github.io/docs/configuration/crds/v1beta1/output_types/].

The Rancher UI provides forms for configuring the following `Output` types:

- Amazon ElasticSearch
- Azure Storage
- Cloudwatch
- Datadog
- Elasticsearch
- File
- Fluentd
- GCS
- Kafka
- Kinesis Stream
- LogDNA
- LogZ
- Loki
- New Relic
- Splunk
- SumoLogic
- Syslog

The Rancher UI provides forms for configuring the `Output` type, target, and access credentials if applicable.

For example configuration for each logging plugin supported by the logging operator, see the Logging operator documentation [https://kube-logging.github.io/docs/configuration/plugins/outputs/].

#### ClusterOutputs

`ClusterOutput` defines an `Output` without namespace restrictions. It is only effective when deployed in the same namespace as the logging operator.

`ClusterOutputs` can be configured by filling out forms in the Rancher UI.

For the details of the `ClusterOutput` custom resource, see ClusterOutput. [https://kube-logging.github.io/docs/configuration/crds/v1beta1/clusteroutput_types/]

#### YAML Examples

Once logging is installed, you can use these examples to help craft your own logging pipeline.

- Cluster Output to ElasticSearch
- Output to Splunk
- Output to Syslog
- Unsupported Outputs

#### Cluster Output to ElasticSearch

Let's say you wanted to send all logs in your cluster to an `elasticsearch` cluster. First, we create a cluster `Output`.

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: ClusterOutput
metadata:
    name: "example-es"
    namespace: "cattle-logging-system"
spec:
    elasticsearch:
      host: elasticsearch.example.com
      port: 9200
      scheme: http
```

We have created this `ClusterOutput`, without elasticsearch configuration, in the same namespace as our operator: `cattle-logging-system.`. Any time we create a `ClusterFlow` or `ClusterOutput`, we have to put it in the `cattle-logging-system` namespace.

Now that we have configured where we want the logs to go, let's configure all logs to go to that `ClusterOutput`.

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: ClusterFlow
metadata:
    name: "all-logs"
    namespace: "cattle-logging-system"
spec:
  globalOutputRefs:
    - "example-es"
```

We should now see our configured index with logs in it.

#### Output to Splunk

What if we have an application team who only wants logs from a specific namespaces sent to a `splunk` server? For this case, we can use namespaced `Outputs` and `Flows`.

Before we start, let's set up that team's application: `coolapp`.

```
apiVersion: v1
kind: Namespace
metadata:
  name: devteam
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coolapp
  namespace: devteam
  labels:
    app: coolapp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coolapp
  template:
    metadata:
      labels:
        app: coolapp
    spec:
      containers:
        - name: generator
          image: paynejacob/loggenerator:latest
```

With `coolapp` running, we will follow a similar path as when we created a `ClusterOutput`. However, unlike `ClusterOutputs`, we create our `Output` in our application's namespace.

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: Output
metadata:
  name: "devteam-splunk"
  namespace: "devteam"
spec:
  splunkHec:
    hec_host: splunk.example.com
    hec_port: 8088
    protocol: http
```

Once again, let's feed our `Output` some logs:

```
apiVersion: logging.banzaicloud.io/v1beta1
```

```
kind: Flow
metadata:
  name: "devteam-logs"
  namespace: "devteam"
spec:
  localOutputRefs:
    - "devteam-splunk"
```

#### Output to Syslog

Let's say you wanted to send all logs in your cluster to an `syslog` server. First, we create a `ClusterOutput`:

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: ClusterOutput
metadata:
  name: "example-syslog"
  namespace: "cattle-logging-system"
spec:
  syslog:
    buffer:
      timekey: 30s
      timekey_use_utc: true
      timekey_wait: 10s
      flush_interval: 5s
    format:
      type: json
      app_name_field: test
    host: syslog.example.com
    insecure: true
    port: 514
    transport: tcp
```

Now that we have configured where we want the logs to go, let's configure all logs to go to that `Output`.

```
apiVersion: logging.banzaicloud.io/v1beta1
kind: ClusterFlow
metadata:
  name: "all-logs"
  namespace: cattle-logging-system
spec:
  globalOutputRefs:
    - "example-syslog"
```

#### Unsupported Outputs

For the final example, we create an `Output` to write logs to a destination that is not supported out of the box:



*Note on syslog:*

`syslog` is a supported `Output`. However, this example still provides an overview on using unsupported plugins.

```
apiVersion: v1
kind: Secret
metadata:
  name: syslog-config
  namespace: cattle-logging-system
type: Opaque
stringData:
  fluent-bit.conf: |
    [INPUT]
        Name              forward
        Port              24224

    [OUTPUT]
        Name              syslog
        InstanceName      syslog-output
        Match             *
        Addr              syslog.example.com
        Port              514
        Cluster           ranchers

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fluentbit-syslog-forwarder
  namespace: cattle-logging-system
  labels:
    output: syslog
spec:
  selector:
    matchLabels:
      output: syslog
  template:
    metadata:
      labels:
        output: syslog
    spec:
```

```
      containers:
      - name: fluentbit
        image: paynejacob/fluent-bit-out-syslog:latest
        ports:
          - containerPort: 24224
        volumeMounts:
          - mountPath: "/fluent-bit/etc/"
            name: configuration
      volumes:
      - name: configuration
        secret:
          secretName: syslog-config
---
apiVersion: vl
kind: Service
metadata:
  name: syslog-forwarder
  namespace: cattle-logging-system
spec:
  selector:
    output: syslog
  ports:
    - protocol: TCP
      port: 24224
      targetPort: 24224
---
apiVersion: logging.banzaicloud.io/vlbetal
kind: ClusterFlow
metadata:
  name: all-logs
  namespace: cattle-logging-system
spec:
  globalOutputRefs:
    - syslog
---
apiVersion: logging.banzaicloud.io/vlbetal
kind: ClusterOutput
metadata:
  name: syslog
  namespace: cattle-logging-system
spec:
  forward:
    servers:
      - host: "syslog-forwarder.cattle-logging-system"
    require_ack_response: false
```

```
    ignore_network_errors_at_startup: false
```

Let's break down what is happening here. First, we create a deployment of a container that has the additional `syslog` plugin and accepts logs forwarded from another `fluentd`. Next we create an `Output` configured as a forwarder to our deployment. The deployment `fluentd` will then forward all logs to the configured `syslog` destination.

## Troubleshooting

### Log Levels

The following log levels are used in Rancher:

| Name | Description |
|---|---|
| `info` | Logs informational messages. This is the default log level. |
| `debug` | Logs more detailed messages that can be used to debug. |
| `trace` | Logs very detailed messages on internal functions. This is very verbose and can contain sensitive information. |

#### How to Configure a Log Level

##### Kubernetes Install

- Configure debug log level

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl -n cattle-system get pods -l app=rancher --no-headers -o
custom-columns=name:.metadata.name | while read rancherpod; do
kubectl -n cattle-system exec $rancherpod -c rancher -- loglevel
--set debug; done
OK
OK
OK
$ kubectl -n cattle-system logs -l app=rancher -c rancher
```

- Configure info log level

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl -n cattle-system get pods -l app=rancher --no-headers -o
custom-columns=name:.metadata.name | while read rancherpod; do
kubectl -n cattle-system exec $rancherpod -c rancher -- loglevel
--set info; done
OK
OK
OK
```

#### Docker Install

- Configure debug log level

```
$ docker exec -ti <container_id> loglevel --set debug
OK
$ docker logs -f <container_id>
```

- Configure info log level

```
$ docker exec -ti <container_id> loglevel --set info
OK
```

#### Rancher Machine Debug Logs

If you need to troubleshoot the creation of objects in your infrastructure provider of choice, `rancher-machine` debug logs might be helpful to you.

It's possible to enable debug logs for `rancher-machine` by setting environment variables when launching Rancher.

The `CATTLE_WHITELIST_ENVVARS` environment variable allows users to whitelist specific environment variables to be passed down to `rancher-machine` during provisioning.

The `MACHINE_DEBUG` variable enables debug logs in `rancher-machine`.

Thus, by setting `MACHINE_DEBUG=true` and adding `MACHINE_DEBUG` to the default list of variables in `CATTLE_WHITELIST_ENVVARS` (e.g. `CATTLE_WHITELIST_ENVVARS=HTTP_PROXY,HTTPS_PROXY,NO_PROXY,MACHINE_DEBUG`) it is possible to enable debug logs in `rancher-machine` when provisioning RKE1, RKE2 and k3s clusters.



Just like the `trace` log level above, `rancher-machine` debug logs can contain sensitive information.

#### Cattle-cluster-agent Debug Logs

The `cattle-cluster-agent` log levels can be set when you initialize downstream clusters.

When you create a cluster under **Cluster Configuration › Agent Environment Vars** you can set variables to define the log level.

- Trace-level logging: Set `CATTLE_TRACE` or `RANCHER_TRACE` to `true`
- Debug-level logging: Set `CATTLE_DEBUG` or `RANCHER_DEBUG` to `true`



The `cattle-cluster-agent` debug logs may contain sensitive information.

# Monitoring and Dashboards

## Rancher Monitoring Dashboards



Starting in Rancher v2.15.0, new monitoring installations use the decoupled `rancher-monitoring-dashboards` chart.

Rancher maintains the legacy `rancher-monitoring` chart to support existing deployments, but it does not receive new updates. Use the `rancher-monitoring-dashboards` chart for ongoing updates and support.

The `rancher-monitoring-dashboards` chart provides Grafana dashboards and Rancher UI integration for external Prometheus monitoring stacks, such as `kube-prometheus-stack`.

- **No bundled runtime:** Operates without depending on Rancher-shipped `kube-prometheus-stack` images or components. It operates independently and does not require installing the legacy `rancher-monitoring` chart. It installs a dashboard-only version of `rancher-monitoring`.
- **Prerequisites:** Requires uninstalling the `rancher-monitoring` chart and using a Prometheus infrastructure. Although several configurations are supported, installing `kube-prometheus-stack` is the simplest method.

### Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`

Follow these steps to migrate to `rancher-monitoring-dashboards`:

1. Uninstall the `rancher-monitoring` chart. See observability/monitoring-and-dashboards/uninstall-monitoring.pdf.
2. Install your own `kube-prometheus-stack`. See Install the `kube-prometheus-stack` chart.
3. Install `rancher-monitoring-dashboards`. See Install `rancher-monitoring-dashboards` chart.

If you need to preserve or migrate existing data, refer to the official Grafana and Prometheus documentation:

- Grafana | Back up Grafana [https://grafana.com/docs/grafana/latest/administration/back-up-grafana/]
- Prometheus | TSDB Admin APIs - Snapshot [https://prometheus.io/docs/prometheus/latest/querying/api/#snapshot]
- Prometheus | Storage [https://prometheus.io/docs/prometheus/latest/storage/]

### Rancher Monitoring Dashboards Installation

This section guides you through setting up your Prometheus stack and installing the `rancher-monitoring-dashboards` chart using the SUSE Application Collection (AppCo).

#### Create the AppCo credentials

First create the secret that will be used by Kubernetes to pull the images and login to Helm to pull the required chart from SUSE AppCo. The secret needs to be created in the `cattle-monitoring-system` namespace where the chart is installed.

Run the following commands to create the secret:

```
kubectl create namespace cattle-monitoring-system

kubectl create secret docker-registry application-collection --docker
-server=dp.apps.rancher.io --docker-username=<your_username> --docker
-password=<your_password> --namespace cattle-monitoring-system
```

Run the following command to login to Helm, replacing `<your_username>` and `<your_password>` with your SUSE AppCo credentials:

```
helm registry login dp.apps.rancher.io -u <your_username> -p
<your_password>
```

**Install the `kube-prometheus-stack` chart**

1. Create the `kube-prometheus-stack values.yaml` file. Configure the file with the following required settings to ensure `kube-prometheus-stack` integrates properly and supports embedded dashboards in `rancher-monitoring-dashboards`.

```
grafana:
  # NOTE: Do NOT set root_url or serve_from_sub_path here!
  grafana.ini:
    security:
      allow_embedding: true
    auth:
      disable_login_form: false
    auth.anonymous:
      enabled: true
      org_role: Viewer
    dashboards:
      default_home_dashboard_path: /tmp/dashboards/rancher-default-
home.json
    users:
      auto_assign_org_role: Viewer
# Prometheus configuration to pick up all ServiceMonitors
prometheus:
  prometheusSpec:
    serviceMonitorSelectorNilUsesHelmValues: false
    podMonitorSelectorNilUsesHelmValues: false

# No metric exporter by default

kubeEtcd:
```

```
  enabled: true
  serviceMonitor:
    enabled: false

kubeControllerManager:
  enabled: true
  serviceMonitor:
    enabled: false

kubeScheduler:
  enabled: true
  serviceMonitor:
    enabled: false

kubeProxy:
  enabled: true
  serviceMonitor:
    enabled: false
```

2. Install the `kube-prometheus-stack` chart in the `cattle-monitoring-system` namespace through the Rancher UI or via the Helm CLI, applying the `values.yaml` file from the previous step:

**UI**

1. Select ☰ **> Cluster Management**.
2. Locate your cluster and click **Explore**.
3. In the left navigation menu, select **Apps > Charts**.
4. Locate and select the **kube-prometheus-stack** chart.
5. Click **Install this version**.
6. In the **Namespace** dropdown, select **cattle-monitoring-system**. If the namespace does not exist:
    a. Select **Create a New Namespace** and enter `cattle-monitoring-system`.
    b. In the **Name** field, enter `kube-prometheus-stack`.
    c. Select **Customize Helm options before install**.
7. Click **Next**.
8. Select **Edit YAML** and paste or upload your `values.yaml` file.
9. Click **Next**.
10. Select and specify additional deployment options.
11. Click **Install**.

**Helm**

```
helm install kube-prometheus-stack
```

```
oci://dp.apps.rancher.io/charts/prometheus-operator \
  --namespace cattle-monitoring-system \
  --create-namespace \
  --debug \
  --wait \
  --set global.imagePullSecrets={application-collection} \
  -f values.yaml
```

**Result:** The Prometheus monitoring infrastructure is deployed in the `cattle-monitoring-system` namespace.

#### Install `rancher-monitoring-dashboards` chart

To deploy the `rancher-monitoring-dashboards` chart using the Rancher UI:

1. Select ☰ > **Cluster Management**.
2. Locate your cluster and click **Explore**.
3. In the left navigation menu, select **Apps > Charts**.
4. Locate and select the latest **rancher-monitoring-dashboards** chart.
5. Click **Install**.
6. Click **Next**.
7. The default `values.yaml` sets the upstream services name to the default value provided by the upstream `kube-prometheus-stack` and NGINX image. When using values provided by the SUSE Application Collection, you must modify the names of the services and the images. Check the SUSE AppCo [https://apps.rancher.io/] website for the latest patch of each image minor to take advantage of images with lower CVE impact when configuring. The following is an example with modified chart values:

   ☰ *Click to expand*

```yaml
alertmanagerProxy:
  enabled: true
  upstreamPort: 9093
  upstreamService: kube-prometheus-stack-prom-alertmanager
commonLabels: {}
dashboardArtifacts:
  annotations: {}
  cleanupOnUninstall: false
  enabled: true
  groups:
    backupRestore: true
    fleet: true
    ingressNginx: true
    logging:
      fluentbit: true
      fluentd: true
```

```
    performance: true
    rancherCore: true
  label: grafana_dashboard
  labelValue: '1'
  namespace: cattle-dashboards
  useExistingNamespace: false
dashboardIntegration:
  grafanaURL: ''
  prometheusURL: ''
defaultRules:
  create: false
etcd:
  enabled: false
extraManifests: []
global:
  cattle:
    clusterId: ''
    clusterName: ''
  clusterDomain: cluster.local
  disableProxyIPv6: false
  imagePullSecrets: ["application-collection"]
  kubectl:
    pullPolicy: IfNotPresent
    repository: rancher/kuberlr-kubectl
    tag: v7.0.1 #The latest minor/version from
https://github.com/rancher/kuberlr-kubectl/releases
  rbac:
    create: true
    userRoles:
      aggregateToDefaultRoles: true
      create: true
grafana:
  defaultDashboards:
    namespace: cattle-dashboards
  defaultDashboardsEnabled: true
  enabled: false
  forceDeployDashboards: false
  sidecar:
    dashboards:
      annotations: {}
      label: grafana_dashboard
      labelValue: '1'
grafanaProxy:
  enabled: true
  proxyPath: >-
```

```
    /api/v1/namespaces/cattle-monitoring-
system/services/http:rancher-monitoring-grafana:80/proxy
  upstreamPort: 80
  upstreamService: kube-prometheus-stack-grafana
hardened:
  enabled: false
  k3s:
    networkPolicy:
      enabled: false
hardenedNodeExporter:
  enabled: false
kubeStateMetrics:
  enabled: false
kubeTargetVersionOverride: ''
monitoringProxy:
  affinity: {}
  image:
    pullPolicy: IfNotPresent
    repository: dp.apps.rancher.io/containers/nginx
    tag: 1.31.5
  nodeSelector: {}
  replicas: 1
  resources:
    limits:
      cpu: 200m
      memory: 256Mi
    requests:
      cpu: 50m
      memory: 64Mi
  tolerations: []
nodeExporter:
  enabled: false
prometheusProxy:
  enabled: true
  upstreamPort: 9090
  upstreamService: kube-prometheus-stack-prom-prometheus
rancherMonitoring:
  enabled: true
  prometheusServiceAccount: kube-prometheus-stack-prom-prometheus
  selector: {}
```

Depending on your distribution you will also need to add the following modifications to your `values.yaml` file:

*K3s:*

```
k3sServer:
  enabled: true
etcd:
  enabled: false
```

*RKE2 or distributions with etcd:*

```
etcd:
  enabled: true
k3sServer:
  enabled: false
```

8. Click **Install**.

**Result:** Rancher deploys the `rancher-monitoring-dashboards` chart and integrates the dashboards into the Rancher UI.

#### Customization and Overrides

Use the configuration settings in this section to customize the `rancher-monitoring-dashboards` chart for your environment.

- Override default proxy images. See Monitoring Proxy Image Customization.
- Align upstream service names with your Prometheus stack and disable unused K3s components. See Service Name Overrides.

##### Monitoring Proxy Image Customization

A single NGINX pod (`monitoringProxy`) acts as a proxy for Grafana, Prometheus, and Alertmanager, ensuring the Kubernetes API server's service proxy (used by the Rancher UI) can reach them reliably. You can customize the NGINX image using the following configuration:

```
monitoringProxy:
  image:
    repository: <new/image>
    tag: <version>
```

##### Service Name Overrides

The default `values.yaml` assumes standard service naming conventions (for example, kube-prometheus-stack-<application>), as shown below:

```
grafanaProxy:
  upstreamService: kube-prometheus-stack-grafana
```

```
prometheusProxy:
  upstreamService: kube-prometheus-stack-prometheus

alertmanagerProxy:
  upstreamService: kube-prometheus-stack-alertmanager
```

This creates mirror services alongside the new chart, which the reverse proxy uses to enable Rancher UI functionality. If you modified the service names during installation, update the `upstreamService` configuration accordingly:

```
grafanaProxy:
  upstreamService: <upstream_service_name>

prometheusProxy:
  upstreamService: <upstream_service_name>

alertmanagerProxy:
  upstreamService: <upstream_service_name>
```



Make sure to disable `k3sServer` if you are not using K3s:

```
k3sServer:
  enabled: false
```

**FAQ**

**Q**: What is the new chart being introduced?

**A**: The `rancher-monitoring-dashboards` chart includes only dashboard artifacts and Rancher UI integration metadata. It does not include Prometheus, Grafana, Alertmanager, exporters, or PushProx. To deploy a full monitoring runtime, install `kube-prometheus-stack` separately from the Prometheus Community Helm repository.

**Q**: Will the legacy `rancher-monitoring` chart still be available?

**A**: Yes. Rancher v2.15 retains the Rancher v2.14 version of the `rancher-monitoring` chart, but you cannot install it directly as a new installation. Rancher v2.15 also includes an updated monitoring version available exclusively for upgrades.

**Q**: What happens to existing installations when upgrading to Rancher v2.15?

**A**: You can upgrade existing installations to the updated monitoring release in Rancher v2.15. To migrate fully to the new dashboards-only chart, follow the standard upstream migration path for Prometheus and Grafana.

**Q**: How does the Rancher UI locate Grafana and Prometheus?

**A**: In this release, monitoring service URLs (such as `rancher-monitoring-<service>`) remain hardcoded in the Rancher UI. Support for configurable, user-provided URLs is planned for a future release.

**Q**: How does the Rancher UI monitoring installation workflow change?

**A**: The Rancher UI installation workflow uses `kube-prometheus-stack` for runtime configuration options. The `rancher-monitoring-dashboards` chart exposes values only for dashboards and Rancher UI integration.

## Rancher Monitoring



The `rancher-monitoring` chart is no longer available in the UI for new installs, but it remains fully supported to upgrade existing `rancher-monitoring` deployments.

The `rancher-monitoring` application can quickly deploy leading open-source monitoring and alerting solutions onto your cluster.

Introduced in Rancher v2.5, the application is powered by Prometheus [https://prometheus.io/], Grafana [https://grafana.com/grafana/], Alertmanager [https://prometheus.io/docs/alerting/latest/alertmanager/], the Prometheus Operator [https://github.com/prometheus-operator/prometheus-operator], and the Prometheus adapter. [https://github.com/DirectXMan12/k8s-prometheus-adapter]

Using the `rancher-monitoring` application, you can quickly deploy leading open-source monitoring and alerting solutions onto your cluster.

### Features

Prometheus lets you view metrics from your Rancher and Kubernetes objects. Using timestamps, Prometheus lets you query and view these metrics in easy-to-read graphs and visuals, either through the Rancher UI or Grafana, which is an analytics viewing platform deployed along with Prometheus.

By viewing data that Prometheus scrapes from your cluster control plane, nodes, and deployments, you can stay on top of everything happening in your cluster. You can then use these analytics to better run your organization: stop system emergencies before they start, develop maintenance strategies, or restore crashed servers.

The monitoring application:

- Monitors the state and processes of your cluster nodes, Kubernetes components, and software deployments.
- Defines alerts based on metrics collected via Prometheus.
- Creates custom Grafana dashboards.
- Configures alert-based notifications via email, Slack, PagerDuty, etc. using Prometheus Alertmanager.
- Defines precomputed, frequently needed or computationally expensive expressions as new time series based on metrics collected via Prometheus.
- Exposes collected metrics from Prometheus to the Kubernetes Custom Metrics API via Prometheus Adapter for use in HPA.

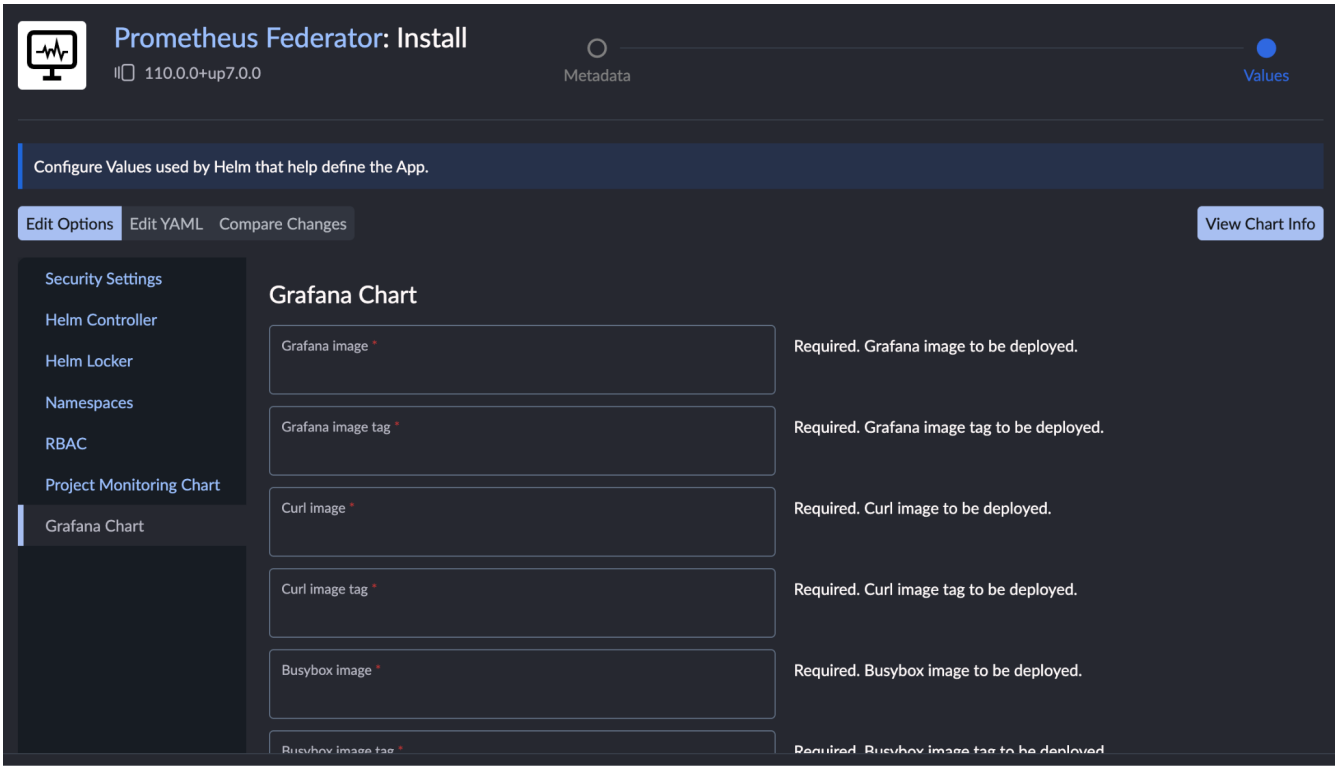
See How Monitoring Works for an explanation of how the monitoring components work together.

- **Mandatory Image Configuration:** Default component images (Grafana, Prometheus, Alertmanager, etc.) are no longer provided by the chart. You must explicitly supply image repositories and tags in your configuration.
- The `chartSource.repo` URL must point strictly to the repository root directory, do not append `/index.yaml` to the URL: https://raw.githubusercontent.com/rancher/ob-team-charts/refs/heads/main.
- Ensure you create the AppCo secret in the namespace `cattle-project-<project_id>-monitoring` and note this must be done for all projects. Include the `imagePullSecrets` in your `values.yaml` file to pull images from the SUSE AppCo registry:



```yaml
helmProjectOperator:
  valuesOverride:
    global:
      imagePullSecrets:
        - name: application-collection
    grafana:
      imagePullSecrets:
        - name: application-collection
```

If installing using the UI ensure all required fields under the **Project Monitoring Chart** and **Grafana Chart** tabs are filled respective to your `values.yaml` configuration, note the example UI view below. This changes the **images**, not the chart. At the moment it is not possible to choose what version of the Grafana chart is installed.




To supply the required Prometheus, Grafana, and Alertmanager images via Helm, specify them under

`helmProjectOperator.valuesOverride` when deploying Prometheus Federator. Check the SUSE AppCo [https://apps.rancher.io/] website for the latest patch of each image minor to take advantage of images with lower CVE impact when configuring:

```
helmProjectOperator:
  valuesOverride:
    grafana:
      downloadDashboardsImage:
        repository: dp.apps.rancher.io/containers/curl
        tag: 8.14.1
      image:
        repository: dp.apps.rancher.io/containers/grafana
        tag: 12.3.1
      imageRenderer:
        image:
          repository: dp.apps.rancher.io/containers/grafana-image-
renderer
          tag: v5.1.0
      initChownData:
        image:
          repository: dp.apps.rancher.io/containers/bci-busybox
          tag: 1.31.1
      sidecar:
        image:
          repository: dp.apps.rancher.io/containers/k8s-sidecar
          tag: 2.1.2
  chartSource:
    name: rancher-project-monitoring
    repo: https://raw.githubusercontent.com/rancher/ob-team-
charts/refs/heads/main
    version: 1.0.0
```

**Pushprox alternative using Prometheus Agent**

Previous versions of `rancher-monitoring` relied on a Rancher-maintained fork of PushProx to scrape metrics from components that bind exclusively to port `127.0.0.1`. These components include:

- `kube-proxy`
- `etcd`
- `kube-scheduler`
- `kube-controller-manager`

This localhost binding strategy is standard across Kubernetes distributions, including K3s and RKE2, and aligns with CIS benchmark recommendations. Upstream PushProx lacks the custom features present in the Rancher fork, making it an unsuitable replacement.

Note that authentication behavior varies across components:

- `kube-scheduler` and `kube-controller-manager` require a valid bearer token for scrape requests.
- `kube-proxy` and `etcd` do not enforce authentication, which remains secure because scrape requests operate entirely over the local loopback interface.

To collect these local metrics, deploy a Prometheus Agent on each control-plane node. The agent scrapes all four targets over loopback and forwards the collected metrics to the primary Prometheus instance via `remote_write`. This approach avoids creating new listening ports or exposing control-plane network interfaces, offering a safer alternative to tools like `socat`.

1. First enable `remote-write` on the previously installed `kube-prometheus-stack chart`:

```
helm upgrade kube-prometheus-stack
oci://dp.apps.rancher.io/charts/prometheus-operator \
    -n cattle-monitoring-system --reuse-values \
    --set prometheus.prometheusSpec.enableRemoteWriteReceiver=true \
    --wait
```

2. Create the service that the Prometheus Agent will use to send metrics to the main Prometheus instance:

```
apiVersion: vl
kind: Service
metadata:
  name: kube-prometheus-stack-prom-prometheus-write
  namespace: cattle-monitoring-system
spec:
  selector:
    app.kubernetes.io/name: prometheus
    operator.prometheus.io/name: kube-prometheus-stack-prom-
prometheus
  ports:
    - name: remote-write
      port: 9091
      targetPort: 9091
```

3. Create the CRD for the Prometheus Agent:

```
apiVersion: monitoring.coreos.com/vlalphal
kind: PrometheusAgent
metadata:
  name: control-plane-agent
  namespace: cattle-monitoring-system
spec:
  replicas: <number_of_replicas>
```

```
  hostNetwork: true
  listenLocal: true
  serviceAccountName: kube-prometheus-stack-prom-prometheus
  nodeSelector:
    node-role.kubernetes.io/control-plane: "true"
  tolerations:
    - operator: Exists
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchLabels:
              app.kubernetes.io/name: prometheus-agent
              operator.prometheus.io/name: control-plane-agent
          topologyKey: kubernetes.io/hostname
  scrapeConfigSelector:
    matchLabels:
      control-plane-agent: "true"
  remoteWrite:
    - url: https://kube-prometheus-stack-prom-prometheus-
write.cattle-monitoring-system.svc:9091/api/v1/write
      authorization:
        type: Bearer
        credentialsFile:
/var/run/secrets/kubernetes.io/serviceaccount/token
      tlsConfig:
        insecureSkipVerify: true
```

4. Create the `ScrapeConfig` that matches the `scrapeConfigSelector` above:

```
apiVersion: monitoring.coreos.com/v1alpha1
kind: ScrapeConfig
metadata:
  name: control-plane-agent-kube-proxy
  namespace: cattle-monitoring-system
  labels:
    control-plane-agent: "true"
spec:
  jobName: kube-proxy
  staticConfigs:
    - targets: ["127.0.0.1:10249"]
---
apiVersion: monitoring.coreos.com/v1alpha1
kind: ScrapeConfig
metadata:
```

```
  name: control-plane-agent-kube-etcd
  namespace: cattle-monitoring-system
  labels:
    control-plane-agent: "true"
spec:
  jobName: kube-etcd
  staticConfigs:
    - targets: ["127.0.0.1:2381"]
---
apiVersion: monitoring.coreos.com/vlalpha1
kind: ScrapeConfig
metadata:
  name: control-plane-agent-kube-scheduler
  namespace: cattle-monitoring-system
  labels:
    control-plane-agent: "true"
spec:
  jobName: kube-scheduler
  scheme: HTTPS
  tlsConfig:
    insecureSkipVerify: true
  authorization:
    credentials:
      name: kube-prometheus-stack-prom-prometheus-token
      key: token
  staticConfigs:
    - targets: ["127.0.0.1:10259"]
---
apiVersion: monitoring.coreos.com/vlalpha1
kind: ScrapeConfig
metadata:
  name: control-plane-agent-kube-controller-manager
  namespace: cattle-monitoring-system
  labels:
    control-plane-agent: "true"
spec:
  jobName: kube-controller-manager
  scheme: HTTPS
  tlsConfig:
    insecureSkipVerify: true
  authorization:
    credentials:
      name: kube-prometheus-stack-prom-prometheus-token
      key: token
  staticConfigs:
```

```
    - targets: ["127.0.0.1:10257"]
```

At this point, the Prometheus Agent is fully configured to scrape metrics from the control-plane components and forward them to the existing Prometheus instance.

#### Optional Security Hardening

Prometheus' `remote_write` receiver (`--web.enable-remote-write-receiver`) lacks built-in authentication capabilities, and `prometheus-operator` does not currently provide a CRD field to configure receiver authentication. For more details on this limitation, refer to the upstream prometheus-operator issue [https://github.com/prometheus-operator/prometheus-operator/issues/4200].

Leaving the receiver unauthenticated exposes the endpoint and increases the cluster's attack surface. To mitigate this risk, deploy `kube-rbac-proxy` as a sidecar container to enforce RBAC authentication and authorization in front of the `remote_write` receiver.

1. First create the `configmap` that will hold the configuration file for the proxy:

```
kubectl create configmap kube-rbac-proxy-remote-write-config \
  -n cattle-monitoring-system \
  --from-literal=config-file.yaml='authorization:
  resourceAttributes:
    namespace: cattle-monitoring-system
    apiVersion: v1
    resource: services
    subresource: remote-write
    name: kube-prometheus-stack-prom-prometheus-write'
```

2. Give the appropriate permissions to the `serviceAccount`:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kube-prometheus-stack-prom-prometheus-auth-delegator
subjects:
  - kind: ServiceAccount
    name: kube-prometheus-stack-prom-prometheus
    namespace: cattle-monitoring-system
roleRef:
  kind: ClusterRole
  name: system:auth-delegator
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
```

```
  name: control-plane-agent-remote-write
rules:
  - apiGroups: [""]
    resources: ["services/remote-write"]
    resourceNames: ["kube-prometheus-stack-prom-prometheus-write"]
    verbs: ["create"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: control-plane-agent-remote-write
subjects:
  - kind: ServiceAccount
    name: kube-prometheus-stack-prom-prometheus
    namespace: cattle-monitoring-system
roleRef:
  kind: ClusterRole
  name: control-plane-agent-remote-write
  apiGroup: rbac.authorization.k8s.io
```

3. Lastly, in the `kube-prometheus-stack` chart, under `prometheus.PrometheusSpec.containers` add the following:

```
containers:
  - name: kube-rbac-proxy-remote-write
    image: dp.apps.rancher.io/containers/kube-rbac-proxy:0.23.0
    args:
      - --secure-listen-address=0.0.0.0:9091
      - --upstream=http://127.0.0.1:9090/
      - --config-file=/etc/kube-rbac-proxy/config-file.yaml
      - --logtostderr=true
      - --v=2
    ports:
      - containerPort: 9091
        name: remote-write
    volumeMounts:
      - name: kube-rbac-proxy-config
        mountPath: /etc/kube-rbac-proxy
    resources:
      requests: { cpu: 10m, memory: 32Mi }
      limits: { cpu: 100m, memory: 64Mi }
volumes:
  - name: kube-rbac-proxy-config
    configMap:
```

```
      name: kube-rbac-proxy-remote-write-config
```

- Port numbers 10249, 10257, 10259, 2381, node role labels, and whether kube-proxy is loopback-bound can differ by distribution and Kubernetes version. Check with `ss -tln` on a node directly before using the configuration outside of K3s or RKE2.
- Resources requests and limits are examples and should be configured based on the cluster size and workload requirements.

## Default Components and Deployments

### Built-in Dashboards

By default, the monitoring application deploys Grafana dashboards (curated by the kube-prometheus [https://github.com/prometheus-operator/kube-prometheus] project) onto a cluster.

It also deploys an Alertmanager UI and a Prometheus UI. For more information about these tools, see Built-in Dashboards.

### Default Metrics Exporters

By default, Rancher Monitoring deploys exporters (such as node-exporter [https://github.com/prometheus/node_exporter] and kube-state-metrics [https://github.com/kubernetes/kube-state-metrics]).

These default exporters automatically scrape metrics for CPU and memory from all components of your Kubernetes cluster, including your workloads.

### Default Alerts

The monitoring application deploys some alerts by default. To see the default alerts, go to the Alertmanager UI and click **Expand all groups.**

### Components Exposed in the Rancher UI

For a list of monitoring components exposed in the Rancher UI, along with common use cases for editing them, see this section.

## Role-based Access Control

For more information on configuring access to monitoring, see this page.

Rancher and Project read permissions don't necessarily apply to monitoring resources. See monitoring-ui-view for more details.

## Guides

- Enable monitoring
- Uninstall monitoring
- Monitoring workloads

- Customizing Grafana dashboards
- Persistent Grafana dashboards
- Debugging high memory usage

## Configuration

### Configuring Monitoring Resources in Rancher

The configuration reference assumes familiarity with how monitoring components work together. For more information, see How Monitoring Works.

- ServiceMonitor and PodMonitor
- Receiver
- Route
- PrometheusRule
- Prometheus
- Alertmanager

### Configuring Helm Chart Options

For more information on `rancher-monitoring` chart options, including options to set resource limits and requests, see Helm Chart Options.

## Windows Cluster Support

To be able to fully deploy Monitoring V2 for Windows, all of your Windows hosts must have a minimum wins [https://github.com/rancher/wins] version of v0.1.0.

For more details on how to upgrade wins on existing Windows hosts, see Windows cluster support for Monitoring V2..

## Known Issues

There is a known issue [https://github.com/rancher/rancher/issues/28787#issuecomment-693611821] that K3s clusters require more than the allotted default memory. If you enable monitoring on a K3s cluster, set `prometheus.prometheusSpec.resources.memory.limit` to 2500 Mi and `prometheus.prometheusSpec.resources.memory.request` to 1750 Mi.

See Debugging High Memory Usage for advice and recommendations.

## Monitoring Best Practices



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Configuring sensible monitoring and alerting rules is vital for running any production workloads securely and reliably. This is not different when using Kubernetes and Rancher. Fortunately the integrated monitoring and alerting functionality makes this whole process a lot easier.

The Rancher monitoring documentation describes how you can set up a complete Prometheus and Grafana stack. Out of the box this will scrape monitoring data from all system and Kubernetes components in your cluster and provide sensible dashboards and alerts for them to get started. But for a reliable setup, you also need to monitor your own workloads and adapt Prometheus and Grafana to your own specific use cases and cluster sizes. This document aims to give you best practices for this.

### What to Monitor

Kubernetes itself, as well as applications running inside of it, form a distributed system where different components interact with each other. For the whole system and each individual component, you have to ensure performance, availability, reliability and scalability. A good resource with more details and information is Google's free Site Reliability Engineering Book [https://sre.google/sre-book/table-of-contents/], especially the chapter about Monitoring distributed systems [https://sre.google/sre-book/monitoring-distributed-systems/].

### Configuring Prometheus Resource Usage

When installing the integrated monitoring stack, Rancher allows to configure several settings that are dependent on the size of your cluster and the workloads running in it. This chapter covers these in more detail.

#### Storage and Data Retention

The amount of storage needed for Prometheus directly correlates to the amount of time series and labels that you store and the data retention you have configured. It is important to note that Prometheus is not meant to be used as a long-term metrics storage. Data retention time is usually only a couple of days and not weeks or months. The reason for this is that Prometheus does not perform any aggregation on its stored metrics. This is great because aggregation can dilute data, but it also means that the needed storage grows linearly over time without retention.

One way to calculate the necessary storage is to look at the average size of a storage chunk in Prometheus with this query

```
rate(prometheus_tsdb_compaction_chunk_size_bytes_sum[1h]) / rate
(prometheus_tsdb_compaction_chunk_samples_sum[1h])
```

Next, find out your data ingestion rate per second:

```
rate(prometheus_tsdb_head_samples_appended_total[1h])
```

and then multiply this with the retention time, adding a few percentage points as buffer:

```
average chunk size in bytes * ingestion rate per second * retention time
in seconds * 1.1 = necessary storage in bytes
```

You can find more information about how to calculate the necessary storage in this blog post [https://www.robustperception.io/how-much-disk-space-do-prometheus-blocks-use].

You can read more about the Prometheus storage concept in the Prometheus documentation [https://prometheus.io/docs/prometheus/latest/storage].

#### CPU and Memory Requests and Limits

In larger Kubernetes clusters Prometheus can consume quite a bit of memory. The amount of memory Prometheus needs directly correlates to the amount of time series and amount of labels it stores and the scrape interval in which these are filled.

You can find more information about how to calculate the necessary memory in this blog post [https://www.robustperception.io/how-much-ram-does-prometheus-2-x-need-for-cardinality-and-ingestion].

The amount of necessary CPUs correlate with the amount of queries you are performing.

#### Federation and Long-term Storage

Prometheus is not meant to store metrics for a long amount of time, but should only be used for short term storage.

In order to store some, or all metrics for a long time, you can leverage Prometheus' remote read/write [https://prometheus.io/docs/prometheus/latest/storage/#remote-storage-integrations] capabilities to connect it to storage systems like Thanos [https://thanos.io/], InfluxDB [https://www.influxdata.com/], M3DB [https://www.m3db.io/], or others. You can find an example setup in this blog post [https://rancher.com/blog/2020/prometheus-metric-federation].

### Scraping Custom Workloads

While the integrated Rancher Monitoring already scrapes system metrics from a cluster's nodes and system components, the custom workloads that you deploy on Kubernetes should also be scraped for data. For that you can configure Prometheus to do an HTTP request to an endpoint of your applications in a certain interval. These endpoints should then return their metrics in a Prometheus format.

In general, you want to scrape data from all the workloads running in your cluster so that you can use them for alerts or debugging issues. Often, you recognize that you need some data only when you actually need the metrics during an incident. It is good, if it is already scraped and stored. Since Prometheus is only meant to be a short-term metrics storage, scraping and keeping lots of data is usually not that expensive. If you are using a long-term storage solution with Prometheus, you can then still decide which data you are actually persisting and keeping there.

#### About Prometheus Exporters

Many 3rd party workloads, such as databases, queues, and web-servers, already support exposing metrics in a Prometheus format, or offer exporters that translate between the tool's metrics and a format that Prometheus understands. You can usually add these exporters as additional sidecar containers to the workload's Pods. Many Helm charts already include options to deploy the correct exporter. You can find a curated list of exports on ExporterHub [https://exporterhub.io/].

#### Prometheus support in Programming Languages and Frameworks

To get your own custom application metrics into Prometheus, you have to collect and expose these metrics directly from your application's code. Fortunately, there are already libraries and integrations available to

help with this for most popular programming languages and frameworks. One example for this is the Prometheus support in the Spring Framework [https://docs.spring.io/spring-metrics/docs/current/public/prometheus].

#### ServiceMonitors and PodMonitors

Once all of your workloads expose metrics in a Prometheus format, you must configure Prometheus to scrape them. Under the hood, Rancher uses the prometheus-operator [https://github.com/prometheus-operator/prometheus-operator]. This makes it easy to add scraping targets with ServiceMonitors and PodMonitors. Many Helm charts already include an option to create these monitors directly. You can also find more information in the Rancher documentation.

#### Prometheus Push Gateway

There are some workloads that are traditionally hard to scrape by Prometheus. Examples for these are short lived workloads like Jobs and CronJobs, or applications that do not allow sharing data between individual handled incoming requests, like PHP applications.

To still get metrics for these use cases, you can set up prometheus-pushgateways [https://github.com/prometheus/pushgateway]. The CronJob or PHP application would push metric updates to the pushgateway. The pushgateway aggregates and exposes them through an HTTP endpoint, which then can be scraped by Prometheus.

#### Prometheus Blackbox Monitor

Sometimes it is useful to monitor workloads from the outside. For this, you can use the Prometheus blackbox-exporter [https://github.com/prometheus/blackbox_exporter] which allows probing any kind of endpoint over HTTP, HTTPS, DNS, TCP and ICMP.

### Monitoring in a (Micro)Service Architecture

If you have a (micro)service architecture where multiple individual workloads within your cluster are communicating with each other, it is really important to have detailed metrics and traces about this traffic to understand how all these workloads are communicating with each other and where a problem or bottleneck may be.

Of course you can monitor all this internal traffic in all your workloads and expose these metrics to Prometheus. But this can quickly become quite work intensive. Service Meshes like Istio, which can be installed with a click in Rancher, can do this automatically and provide rich telemetry about the traffic between all services.

### Real User Monitoring

Monitoring the availability and performance of all your internal workloads is vitally important to run stable, reliable and fast applications. But these metrics only show you parts of the picture. To get a complete view it is also necessary to know how your end users are actually perceiving it. For this you can look into various Real user monitoring solutions [https://en.wikipedia.org/wiki/Real_user_monitoring].

### Security Monitoring

In addition to monitoring workloads to detect performance, availability or scalability problems, the cluster and the workloads running into it should also be monitored for potential security problems. A good starting point is to frequently run and alert on Compliance Scans which check if the cluster is configured according

to security best practices.

For the workloads, you can have a look at Kubernetes and Container security solutions like NeuVector [https://www.suse.com/products/neuvector/], Falco [https://falco.org/], Aqua Kubernetes Security [https://www.aquasec.com/solutions/kubernetes-container-security/], SysDig [https://sysdig.com/].

#### Setting up Alerts

Getting all the metrics into a monitoring systems and visualizing them in dashboards is great, but you also want to be pro-actively alerted if something goes wrong.

The integrated Rancher monitoring already configures a sensible set of alerts that make sense in any Kubernetes cluster. You should extend these to cover your specific workloads and use cases.

When setting up alerts, configure them for all the workloads that are critical to the availability of your applications. But also make sure that they are not too noisy. Ideally every alert you are receiving should be because of a problem that needs your attention and needs to be fixed. If you have alerts that are firing all the time but are not that critical, there is a danger that you start ignoring your alerts all together and then miss the real important ones. Less may be more here. Start to focus on the real important metrics first, for example alert if your application is offline. Fix all the problems that start to pop up and then start to create more detailed alerts.

If an alert starts firing, but there is nothing you can do about it at the moment, it's also fine to silence the alert for a certain amount of time, so that you can look at it later.

You can find more information on how to set up alerts and notification channels in the Rancher Documentation.

## Built-in Dashboards



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

#### Grafana UI

Grafana [https://grafana.com/grafana/] allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data driven culture.

To see the default dashboards for time series data visualization, go to the Grafana UI.

##### Customizing Grafana

To view and customize the PromQL queries powering the Grafana dashboard, see this page.

##### Persistent Grafana Dashboards

To create a persistent Grafana dashboard, see this page.

#### Access to Grafana

For information about role-based access control for Grafana, see this section.

### Alertmanager UI

When `rancher-monitoring` is installed, the Prometheus Alertmanager UI is deployed, allowing you to view your alerts and the current Alertmanager configuration.



This section assumes familiarity with how monitoring components work together. For more information about Alertmanager, see How Alertmanager Works.

#### Accessing the Alertmanager UI

The Alertmanager UI lets you see the most recently fired alerts.



*Prerequisite:*

The `rancher-monitoring` application must be installed.

To see the Alertmanager UI,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the Alertmanager UI, click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Alertmanager**.

**Result:** The Alertmanager UI opens in a new tab. For help with configuration, refer to the official Alertmanager documentation. [https://prometheus.io/docs/alerting/latest/alertmanager/]

For more information on configuring Alertmanager in Rancher, see this page.

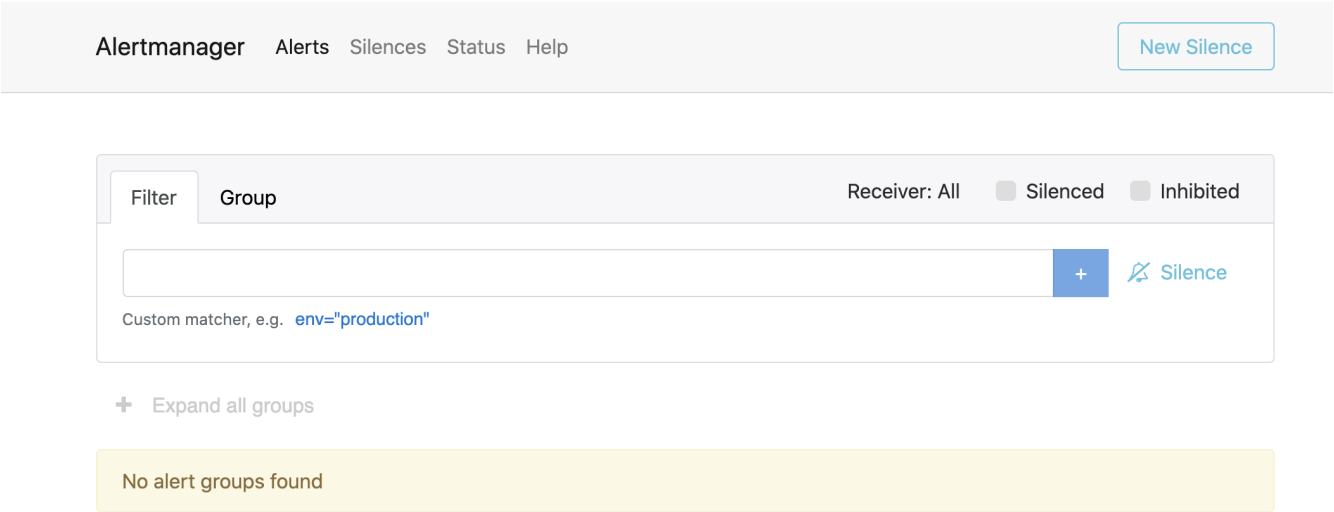



*Figure 20. The Alertmanager UI*

#### Viewing Default Alerts

To see alerts that are fired by default, go to the Alertmanager UI and click **Expand all groups**.

#### Prometheus UI

By default, the kube-state-metrics service [https://github.com/kubernetes/kube-state-metrics] provides a wealth of information about CPU and memory utilization to the monitoring application. These metrics cover Kubernetes resources across namespaces. This means that in order to see resource metrics for a service, you don't need to create a new ServiceMonitor for it. Because the data is already in the time series database, you can go to the Prometheus UI and run a PromQL query to get the information. The same query can be used to configure a Grafana dashboard to show a graph of those metrics over time.

To see the Prometheus UI, install `rancher-monitoring`. Then:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the Prometheus UI and click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Prometheus Graph**.

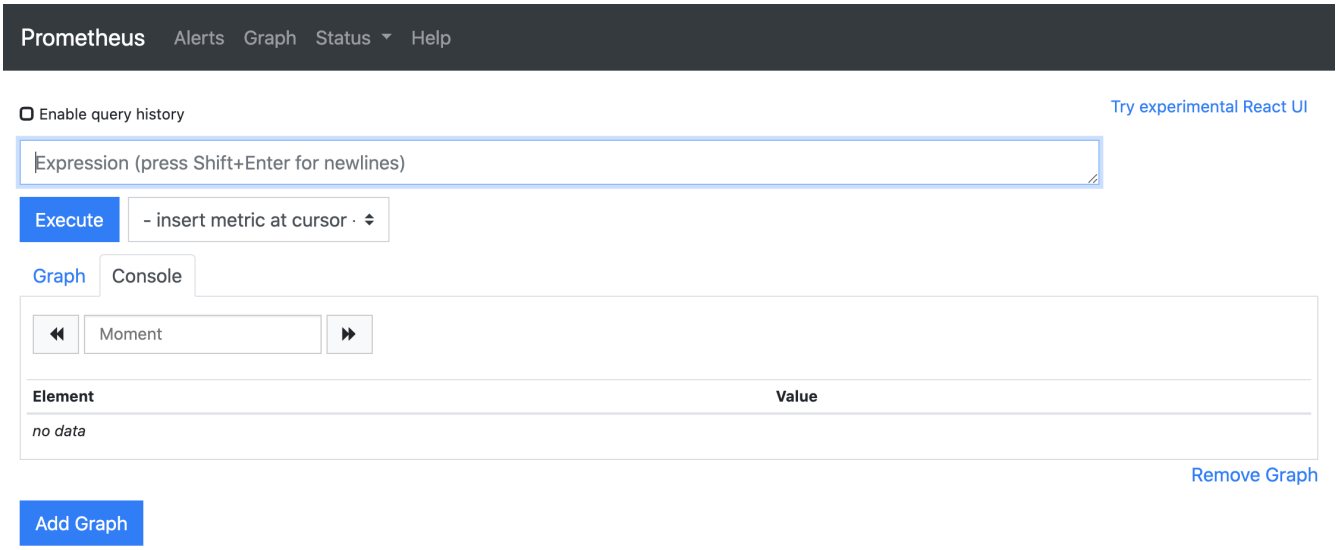



*Figure 21. Prometheus Graph UI*

#### Viewing the Prometheus Targets

To see what services you are monitoring, you will need to see your targets. Targets are set up by ServiceMonitors and PodMonitors as sources to scrape metrics from. You won't need to directly edit targets, but the Prometheus UI can be useful for giving you an overview of all of the sources of metrics that are being scraped.

To see the Prometheus Targets, install `rancher-monitoring`. Then:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the Prometheus targets and click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Prometheus Targets**.

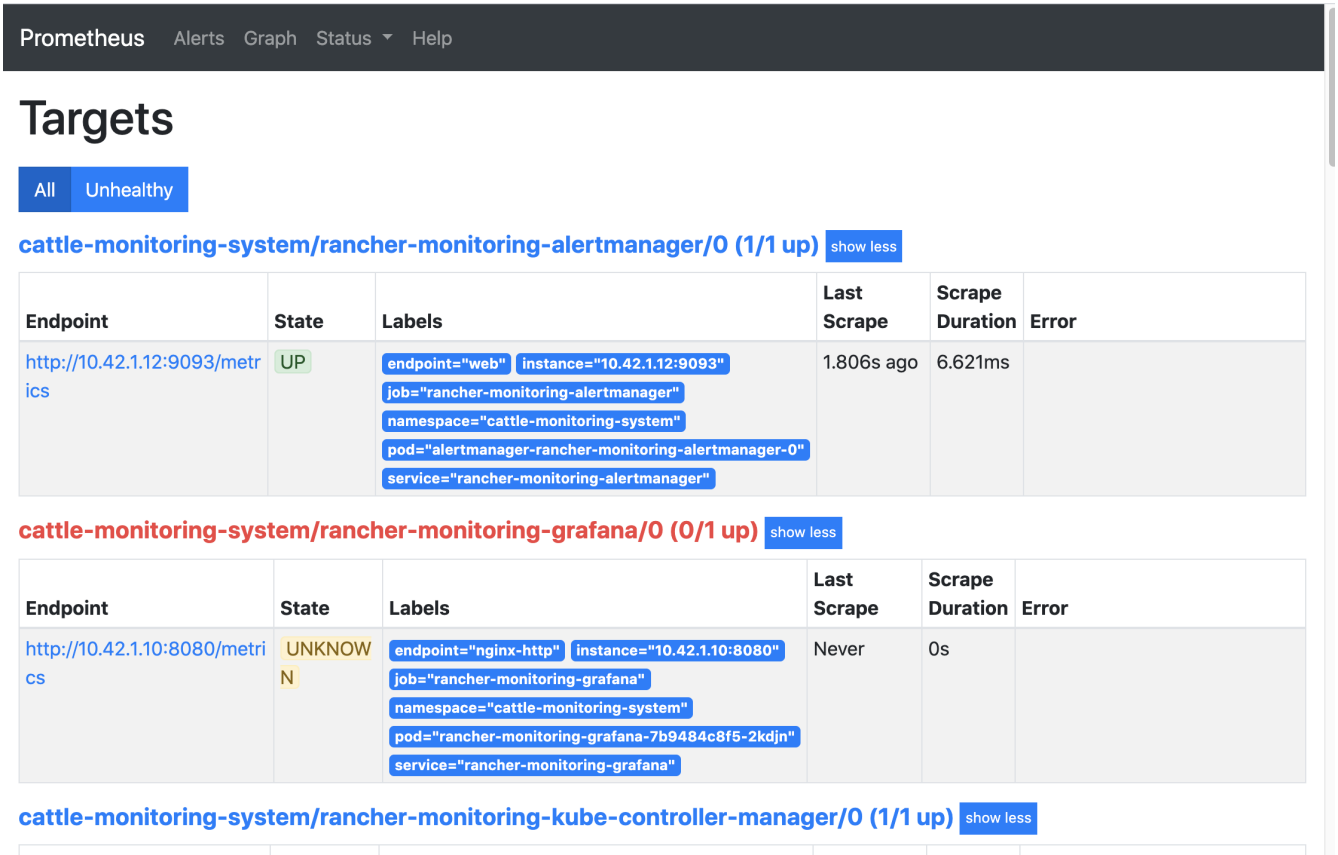



*Figure 22. Targets in the Prometheus UI*

#### Viewing the PrometheusRules

When you define a Rule (which is declared within a RuleGroup in a PrometheusRule resource), the spec of the Rule itself [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#rule] contains labels that are used by Alertmanager to figure out which Route should receive a certain Alert.

To see the PrometheusRules, install `rancher-monitoring`. Then:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the visualizations and click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Prometheus Rules**.

You can also see the rules in the Prometheus UI:

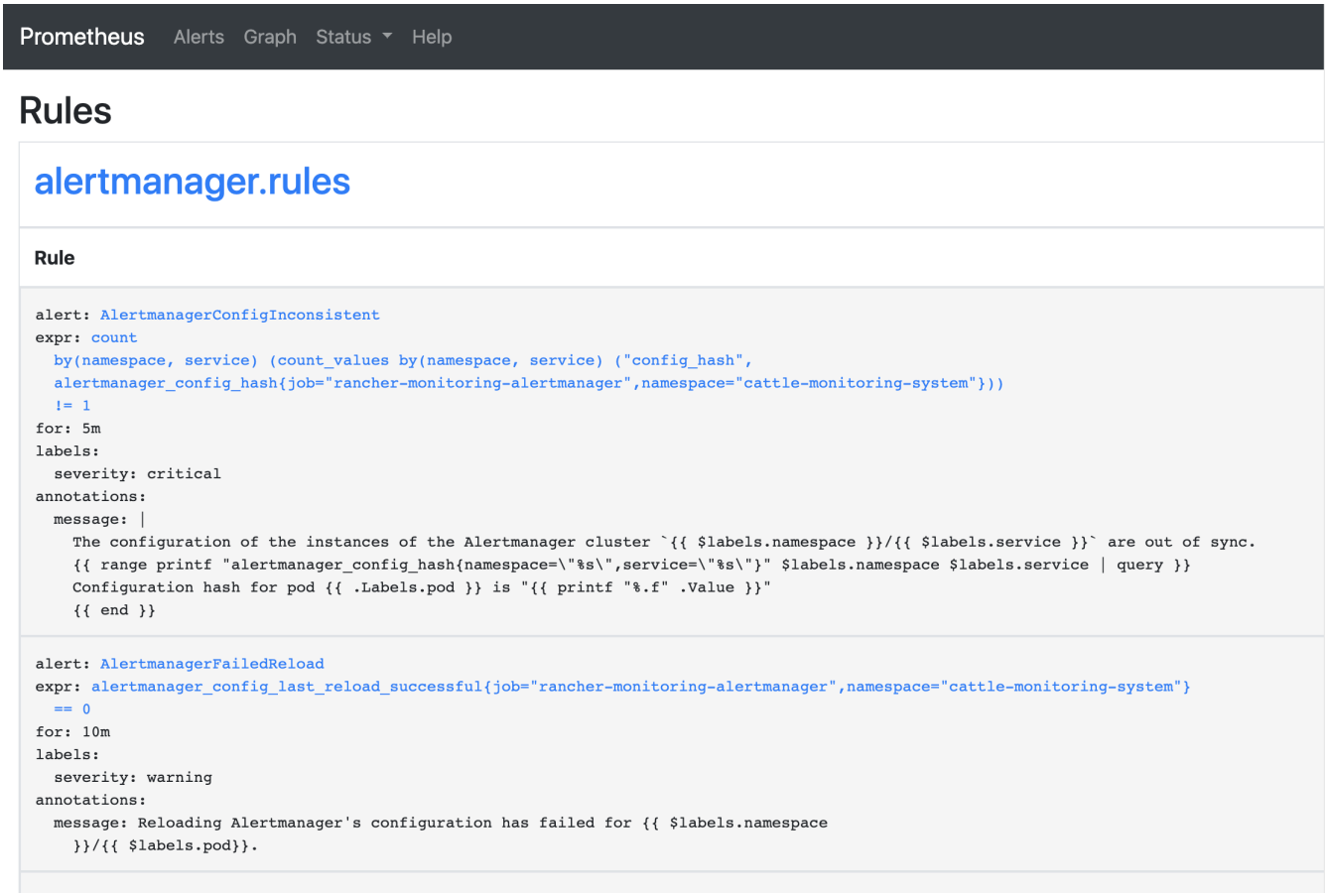



*Figure 23. Rules in the Prometheus UI*

For more information on configuring PrometheusRules in Rancher, see this page.

### Legacy UI

For information on the dashboards available in v2.2 to v2.4 of Rancher, before the introduction of the `rancher-monitoring` application, see the xref:[Rancher v2.0—v2.4 docs].

## How Monitoring Works



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

### 1. Architecture Overview

**The following sections describe how data flows through the Monitoring V2 application:**

#### Prometheus Operator

Prometheus Operator observes ServiceMonitors, PodMonitors, and PrometheusRules being created. When the Prometheus configuration resources are created, Prometheus Operator calls the Prometheus API to sync the new configuration. As the diagram at the end of this section shows, the Prometheus Operator acts as the intermediary between Prometheus and Kubernetes, calling the Prometheus API to synchronize Prometheus

with the monitoring-related resources in Kubernetes.

#### ServiceMonitors and PodMonitors

ServiceMonitors and PodMonitors declaratively specify targets, such as Services and Pods, that need to be monitored.

- Targets are scraped on a recurring schedule based on the configured Prometheus scrape interval, and the metrics that are scraped are stored into the Prometheus Time Series Database (TSDB).
- In order to perform the scrape, ServiceMonitors and PodMonitors are defined with label selectors that determine which Services or Pods should be scraped and endpoints that determine how the scrape should happen on the given target, e.g., scrape/metrics in TCP 10252, proxying through IP addr x.x.x.x.
- Out of the box, Monitoring V2 comes with certain pre-configured exporters that are deployed based on the type of Kubernetes cluster that it is deployed on. For more information, see Scraping and Exposing Metrics.

#### How PushProx Works

- Certain internal Kubernetes components are scraped via a proxy deployed as part of Monitoring V2 called **PushProx**. The Kubernetes components that expose metrics to Prometheus through PushProx are the following: `kube-controller-manager`, `kube-scheduler`, `etcd`, and `kube-proxy`.
- For each PushProx exporter, we deploy one PushProx client onto all target nodes. For example, a PushProx client is deployed onto all controlplane nodes for kube-controller-manager, all etcd nodes for kube-etcd, and all nodes for kubelet.
- We deploy exactly one PushProx proxy per exporter. The process for exporting metrics is as follows:
  1. The PushProx Client establishes an outbound connection with the PushProx Proxy.
  2. The client then polls the proxy for scrape requests that have come into the proxy.
  3. When the proxy receives a scrape request from Prometheus, the client sees it as a result of the poll.
  4. The client scrapes the internal component.
  5. The internal component responds by pushing metrics back to the proxy.

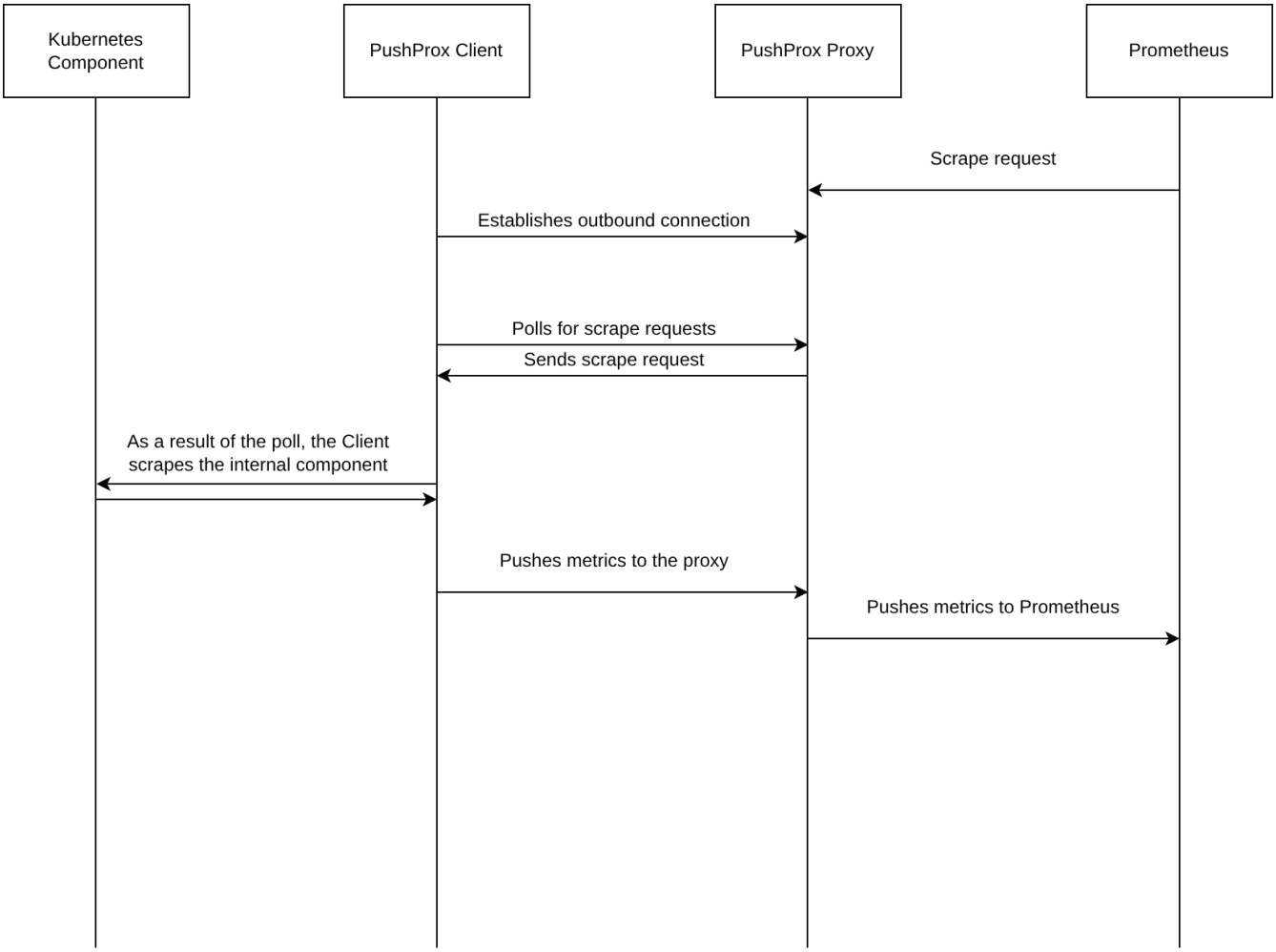



*Figure 24. Process for Exporting Metrics with PushProx*

#### PrometheusRules

PrometheusRules allow users to define rules for what metrics or time series database queries should result in alerts being fired. Rules are evaluated on an interval.

- **Recording rules** create a new time series based on existing series that have been collected. They are frequently used to precompute complex queries.
- **Alerting rules** run a particular query and fire an alert from Prometheus if the query evaluates to a non-zero value.

#### Alert Routing

Once Prometheus determines that an alert needs to be fired, alerts are forwarded to **Alertmanager**.

- Alerts contain labels that come from the PromQL query itself and additional labels and annotations that can be provided as part of specifying the initial PrometheusRule.
- Before receiving any alerts, Alertmanager will use the **routes** and **receivers** specified in its configuration to form a routing tree on which all incoming alerts are evaluated. Each node of the routing tree can specify additional grouping, labeling, and filtering that needs to happen based on the labels attached to the Prometheus alert. A node on the routing tree (usually a leaf node) can also specify that an alert that reaches it needs to be sent out to a configured Receiver, e.g., Slack, PagerDuty, SMS, etc. Note that Alertmanager will send an alert first to **alertingDriver**, then alertingDriver will send or forward alert to the proper destination.

- Routes and receivers are also stored in the Kubernetes API via the Alertmanager Secret. When the Secret is updated, Alertmanager is also updated automatically. Note that routing occurs via labels only (not via annotations, etc.).

## 2. How Prometheus Works

#### Storing Time Series Data

After collecting metrics from exporters, Prometheus stores the time series in a local on-disk time series database. Prometheus optionally integrates with remote systems, but `rancher-monitoring` uses local storage for the time series database.

Once stored, users can query this TSDB using PromQL, the query language for Prometheus.

PromQL queries can be visualized in one of two ways:

1. By supplying the query in Prometheus's Graph UI, which will show a simple graphical view of the data.
2. By creating a Grafana Dashboard that contains the PromQL query and additional formatting directives that label axes, add units, change colors, use alternative visualizations, etc.

#### Defining Rules for Prometheus

Rules define queries that Prometheus needs to execute on a regular `evaluationInterval` to perform certain actions, such as firing an alert (alerting rules) or precomputing a query based on others existing in its TSDB (recording rules). These rules are encoded in PrometheusRules custom resources. When PrometheusRule custom resources are created or updated, the Prometheus Operator observes the change and calls the Prometheus API to synchronize the set of rules that Prometheus is currently evaluating on a regular interval.

A PrometheusRule allows you to define one or more RuleGroups. Each RuleGroup consists of a set of Rule objects that can each represent either an alerting or a recording rule with the following fields:

- The name of the new alert or record
- A PromQL expression for the new alert or record
- Labels that should be attached to the alert or record that identify it (e.g. cluster name or severity)
- Annotations that encode any additional important pieces of information that need to be displayed on the notification for an alert (e.g. summary, description, message, runbook URL, etc.). This field is not required for recording rules.

Upon evaluating a rule [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#rule], Prometheus runs the provided PromQL query, adds the provided labels, and runs the appropriate action for the rule. If the rule triggers an alert, Prometheus also adds the provided annotations. For example, an Alerting Rule that adds `team: front-end` as a label to the provided PromQL query will append that label to the fired alert, which will allow Alertmanager to forward the alert to the correct Receiver.

#### Alerting and Recording Rules

Prometheus doesn't maintain the state of whether alerts are active. It fires alerts repetitively at every evaluation interval, relying on Alertmanager to group and filter the alerts into meaningful notifications.

The `evaluation_interval` constant defines how often Prometheus evaluates its alerting rules against the

time series database. Similar to the `scrape_interval`, the `evaluation_interval` also defaults to one minute.

The rules are contained in a set of rule files. Rule files include both alerting rules and recording rules, but only alerting rules result in alerts being fired after their evaluation.

For recording rules, Prometheus runs a query, then stores it as a time series. This synthetic time series is useful for storing the results of an expensive or time-consuming query so that it can be queried more quickly in the future.

Alerting rules are more commonly used. Whenever an alerting rule evaluates to a positive number, Prometheus fires an alert.

The Rule file adds labels and annotations to alerts before firing them, depending on the use case:

- Labels indicate information that identifies the alert and could affect the routing of the alert. For example, if when sending an alert about a certain container, the container ID could be used as a label.
- Annotations denote information that doesn't affect where an alert is routed, for example, a runbook or an error message.

### 3. How Alertmanager Works

The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of the following tasks:

- Deduplicating, grouping, and routing alerts to the correct receiver integration such as email, PagerDuty, or OpsGenie
- Silencing and inhibition of alerts
- Tracking alerts that fire over time
- Sending out the status of whether an alert is currently firing, or if it is resolved

#### Alerts Forwarded by alertingDrivers

When alertingDrivers are installed, this creates a `Service` that can be used as the receiver's URL for Teams or SMS, based on the alertingDriver's configuration. The URL in the Receiver points to the alertingDrivers; so the Alertmanager sends alert first to alertingDriver, then alertingDriver forwards or sends alert to the proper destination.

#### Routing Alerts to Receivers

Alertmanager coordinates where alerts are sent. It allows you to group alerts based on labels and fire them based on whether certain labels are matched. One top-level route accepts all alerts. From there, Alertmanager continues routing alerts to receivers based on whether they match the conditions of the next route.

While the Rancher UI forms only allow editing a routing tree that is two levels deep, you can configure more deeply nested routing structures by editing the Alertmanager Secret.

#### Configuring Multiple Receivers

By editing the forms in the Rancher UI, you can set up a Receiver resource with all the information Alertmanager needs to send alerts to your notification system.

By editing custom YAML in the Alertmanager or Receiver configuration, you can also send alerts to multiple notification systems. For more information, see the section on configuring Receivers.

### 4. Monitoring V2 Specific Components

Prometheus Operator introduces a set of Custom Resource Definitions [https://github.com/prometheus-operator/prometheus-operator#customresourcedefinitions] that allow users to deploy and manage Prometheus and Alertmanager instances by creating and modifying those custom resources on a cluster.

Prometheus Operator will automatically update your Prometheus configuration based on the live state of the resources and configuration options that are edited in the Rancher UI.

#### Resources Deployed by Default

By default, a set of resources curated by the kube-prometheus [https://github.com/prometheus-operator/kube-prometheus] project are deployed onto your cluster as part of installing the Rancher Monitoring Application to set up a basic Monitoring/Alerting stack.

The resources that get deployed onto your cluster to support this solution can be found in the `rancher-monitoring` [https://github.com/rancher/charts/tree/main/charts/rancher-monitoring] Helm chart, which closely tracks the upstream kube-prometheus-stack [https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack] Helm chart maintained by the Prometheus community with certain changes tracked in the CHANGELOG.md [https://github.com/rancher/charts/blob/main/charts/rancher-monitoring/CHANGELOG.md].

#### Default Exporters

Monitoring V2 deploys three default exporters that provide additional metrics for Prometheus to store:

1. `node-exporter`: exposes hardware and OS metrics for Linux hosts. For more information on `node-exporter`, refer to the upstream documentation [https://prometheus.io/docs/guides/node-exporter/].
2. `windows-exporter`: exposes hardware and OS metrics for Windows hosts (only deployed on Windows clusters). For more information on `windows-exporter`, refer to the upstream documentation [https://github.com/prometheus-community/windows_exporter].
3. `kube-state-metrics`: expose additional metrics that track the state of resources contained in the Kubernetes API (e.g., pods, workloads, etc.). For more information on `kube-state-metrics`, refer to the upstream documentation [https://github.com/kubernetes/kube-state-metrics/tree/master/docs].

ServiceMonitors and PodMonitors will scrape these exporters, as defined here. Prometheus stores these metrics, and you can query the results via either Prometheus's UI or Grafana.

See the architecture section for more information on recording rules, alerting rules, and Alertmanager.

#### Components Exposed in the Rancher UI

When the monitoring application is installed, you will be able to edit the following components in the Rancher UI:

| Component | Type of Component | Purpose and Common Use Cases for Editing |
|---|---|---|
| ServiceMonitor | Custom resource | Sets up Kubernetes Services to scrape custom metrics from. Automatically updates the scrape configuration in the Prometheus custom resource. |
| PodMonitor | Custom resource | Sets up Kubernetes Pods to scrape custom metrics from. Automatically updates the scrape configuration in the Prometheus custom resource. |
| Receiver | Configuration block (part of Alertmanager) | Modifies information on where to send an alert (e.g., Slack, PagerDuty, etc.) and any necessary information to send the alert (e.g., TLS certs, proxy URLs, etc.). Automatically updates the Alertmanager custom resource. |
| Route | Configuration block (part of Alertmanager) | Modifies the routing tree that is used to filter, label, and group alerts based on labels and send them to the appropriate Receiver. Automatically updates the Alertmanager custom resource. |
| PrometheusRule | Custom resource | Defines additional queries that need to trigger alerts or define materialized views of existing series that are within Prometheus's TSDB. Automatically updates the Prometheus custom resource. |

#### PushProx

PushProx allows Prometheus to scrape metrics across a network boundary, which prevents users from having to expose metrics ports for internal Kubernetes components on each node in a Kubernetes cluster.

Since the metrics for Kubernetes components are generally exposed on the host network of nodes in the cluster, PushProx deploys a DaemonSet of clients that sit on the hostNetwork of each node and make an outbound connection to a single proxy that is sitting on the Kubernetes API. Prometheus can then be configured to proxy scrape requests through the proxy to each client, which allows it to scrape metrics from the internal Kubernetes components without requiring any inbound node ports to be open.

Refer to Scraping Metrics with PushProx for more.

### 5. Scraping and Exposing Metrics

#### Defining what Metrics are Scraped

ServiceMonitors and PodMonitors define targets that are intended for Prometheus to scrape. The Prometheus custom resource [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/getting-started/design.md#prometheus] tells Prometheus which ServiceMonitors or PodMonitors it should use to find out where to scrape metrics from.

The Prometheus Operator observes the ServiceMonitors and PodMonitors. When it observes that they are created or updated, it calls the Prometheus API to update the scrape configuration in the Prometheus custom resource and keep it in sync with the scrape configuration in the ServiceMonitors or PodMonitors. This scrape configuration tells Prometheus which endpoints to scrape metrics from and how it will label the

metrics from those endpoints.

Prometheus scrapes all of the metrics defined in its scrape configuration at every `scrape_interval`, which is one minute by default.

The scrape configuration can be viewed as part of the Prometheus custom resource that is exposed in the Rancher UI.

#### How the Prometheus Operator Sets up Metrics Scraping

The Prometheus Deployment or StatefulSet scrapes metrics, and the configuration of Prometheus is controlled by the Prometheus custom resources. The Prometheus Operator watches for Prometheus and Alertmanager resources, and when they are created, the Prometheus Operator creates a Deployment or StatefulSet for Prometheus or Alertmanager with the user-defined configuration.

When the Prometheus Operator observes ServiceMonitors, PodMonitors, and PrometheusRules being created, it knows that the scrape configuration needs to be updated in Prometheus. It updates Prometheus by first updating the configuration and rules files in the volumes of Prometheus's Deployment or StatefulSet. Then it calls the Prometheus API to sync the new configuration, resulting in the Prometheus Deployment or StatefulSet to be modified in place.

#### How Kubernetes Component Metrics are Exposed

Prometheus scrapes metrics from deployments known as exporters, [https://prometheus.io/docs/instrumenting/exporters/] which export the time series data in a format that Prometheus can ingest. In Prometheus, time series consist of streams of timestamped values belonging to the same metric and the same set of labeled dimensions.

#### Scraping Metrics with PushProx

Certain internal Kubernetes components are scraped via a proxy deployed as part of Monitoring V2 called PushProx. For detailed information on PushProx, refer here and to the above architecture section.

#### Scraping Metrics

The following Kubernetes components are directly scraped by Prometheus:

- kubelet*
- Traefik**
- coreDns/kubeDns
- kube-api-server
- You can optionally use `hardenedKubelet.enabled` to use a PushProx, but that is not the default.
  - For RKE2 clusters, Traefik is deployed by default and treated as an internal Kubernetes component.

#### Scraping Metrics Based on Kubernetes Distribution

Metrics are scraped differently based on the Kubernetes distribution. For help with terminology, refer here. For details, see the table below:

*Table 11. How Metrics are Exposed to Prometheus*

| Kubernetes Component | RKE2 | KubeADM | K3s |
|---|---|---|---|
| kube-controller-manager | rke2ControllerManager.enabled | kubeAdmControllerManager.enabled | k3sServer.enabled |
| kube-scheduler | rke2Scheduler.enabled | kubeAdmScheduler.enabled | k3sServer.enabled |
| etcd | rke2Etcd.enabled | kubeAdmEtcd.enabled | Not available |
| kube-proxy | rke2Proxy.enabled | kubeAdmProxy.enabled | k3sServer.enabled |
| kubelet | Collects metrics directly exposed by kubelet | Collects metrics directly exposed by kubelet | Collects metrics directly exposed by kubelet |
| Traefik* | Collects metrics directly exposed by kubelet | Not available | Not available |
| coreDns/kubeDns | Collects metrics directly exposed by coreDns/kubeDns | Collects metrics directly exposed by coreDns/kubeDns | Collects metrics directly exposed by coreDns/kubeDns |
| kube-api-server | Collects metrics directly exposed by kube-api-server | Collects metrics directly exposed by kube-appi-server | Collects metrics directly exposed by kube-api-server |

- For RKE2 clusters, Traefik is deployed by default and treated as an internal Kubernetes component.

**Terminology**

- **kube-scheduler:** The internal Kubernetes component that uses information in the pod spec to decide on which node to run a pod.
- **kube-controller-manager:** The internal Kubernetes component that is responsible for node management (detecting if a node fails), pod replication and endpoint creation.
- **etcd:** The internal Kubernetes component that is the distributed key/value store which Kubernetes uses for persistent storage of all cluster information.
- **kube-proxy:** The internal Kubernetes component that watches the API server for pods/services changes in order to maintain the network up to date.
- **kubelet:** The internal Kubernetes component that watches the API server for pods on a node and makes sure they are running.
- **Traefik:** An Ingress controller for Kubernetes that can be used as a reverse proxy and load balancer.
- **coreDns/kubeDns:** The internal Kubernetes component responsible for DNS.
- **kube-api-server:** The main internal Kubernetes component that is responsible for exposing APIs for the other master components.

## PromQL Expression Reference



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

The PromQL expressions in this doc can be used to configure alerts.

For more information about querying the Prometheus time series database, refer to the official Prometheus documentation. [https://prometheus.io/docs/prometheus/latest/querying/basics/]

## Cluster Metrics

### Cluster CPU Utilization

| Catalog | Expression |
| --- | --- |
| Detail | `1 - (avg(irate(node_cpu_seconds_total{mode="idle"}[5m])) by (instance))` |
| Summary | `1 - (avg(irate(node_cpu_seconds_total{mode="idle"}[5m])))` |

### Cluster Load Average

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>load1</td><td>`sum(node_load1) by (instance) / count(node_cpu_seconds_total{mode="system"}) by (instance)`</td></tr><tr><td>load5</td><td>`sum(node_load5) by (instance) / count(node_cpu_seconds_total{mode="system"}) by (instance)`</td></tr><tr><td>load15</td><td>`sum(node_load15) by (instance) / count(node_cpu_seconds_total{mode="system"}) by (instance)`</td></tr></table> |
| Summary | <table><tr><td>load1</td><td>`sum(node_load1) by (instance) / count(node_cpu_seconds_total{mode="system"})`</td></tr><tr><td>load5</td><td>`sum(node_load5) by (instance) / count(node_cpu_seconds_total{mode="system"})`</td></tr><tr><td>load15</td><td>`sum(node_load15) by (instance) / count(node_cpu_seconds_total{mode="system"})`</td></tr></table> |

### Cluster Memory Utilization

| Catalog | Expression |
| --- | --- |
| Detail | `1 - sum(node_memory_MemAvailable_bytes) by (instance) / sum(node_memory_MemTotal_bytes) by (instance)` |
| Summary | `1 - sum(node_memory_MemAvailable_bytes) / sum(node_memory_MemTotal_bytes)` |

### Cluster Disk Utilization

| Catalog | Expression |
| --- | --- |
| Detail | `(sum(node_filesystem_size_bytes{device!="rootfs"}) by (instance) - sum(node_filesystem_free_bytes{device!="rootfs"}) by (instance)) / sum(node_filesystem_size_bytes{device!="rootfs"}) by (instance)` |
| Summary | `(sum(node_filesystem_size_bytes{device!="rootfs"}) - sum(node_filesystem_free_bytes{device!="rootfs"})) / sum(node_filesystem_size_bytes{device!="rootfs"})` |

**Cluster Disk I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>read</td><td>`sum(rate(node_disk_read_bytes_total[5m])) by (instance)`</td></tr><tr><td>written</td><td>`sum(rate(node_disk_written_bytes_total[5m])) by (instance)`</td></tr></table> |
| Summary | <table><tr><td>read</td><td>`sum(rate(node_disk_read_bytes_total[5m]))`</td></tr><tr><td>written</td><td>`sum(rate(node_disk_written_bytes_total[5m]))`</td></tr></table> |

**Cluster Network Packets**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive-dropped</td><td>sum(rate(node_network_receive_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>receive-errs</td><td>sum(rate(node_network_receive_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>receive-packets</td><td>sum(rate(node_network_receive_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>transmit-dropped</td><td>sum(rate(node_network_transmit_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>transmit-errs</td><td>sum(rate(node_network_transmit_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>transmit-packets</td><td>sum(rate(node_network_transmit_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>receive-dropped</td><td>sum(rate(node_network_receive_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>receive-errs</td><td>sum(rate(node_network_receive_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>receive-packets</td><td>sum(rate(node_network_receive_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>transmit-dropped</td><td>sum(rate(node_network_transmit_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>transmit-errs</td><td>sum(rate(node_network_transmit_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>transmit-packets</td><td>sum(rate(node_network_transmit_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr></table> |

#### Cluster Network I/O

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive</td><td>sum(rate(node_network_receive_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr><tr><td>transmit</td><td>sum(rate(node_network_transmit_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>receive</td><td>sum(rate(node_network_receive_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr><tr><td>transmit</td><td>sum(rate(node_network_transmit_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*"}[5m]))</td></tr></table> |

### Node Metrics

#### Node CPU Utilization

| Catalog | Expression |
| --- | --- |
| Detail | `avg(irate(node_cpu_seconds_total{mode!="idle", instance=~"$instance"}[5m])) by (mode)` |
| Summary | `1 - (avg(irate(node_cpu_seconds_total{mode="idle", instance=~"$instance"}[5m])))` |

#### Node Load Average

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>load1</td><td>sum(node_load1{instance=~"$instance"}) / count(node_cpu_seconds_total{mode="system",instance="$instance"})</td></tr><tr><td>load5</td><td>sum(node_load5{instance="$instance"}) / count(node_cpu_seconds_total{mode="system",instance="$instance"})</td></tr><tr><td>load15</td><td>sum(node_load15{instance="$instance"}) / count(node_cpu_seconds_total{mode="system",instance=~"$instance"})</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>load1</td><td>sum(node_load1{instance=~"$instance"}) / count(node_cpu_seconds_total{mode="system",instance="$instance"})</td></tr><tr><td>load5</td><td>sum(node_load5{instance="$instance"}) / count(node_cpu_seconds_total{mode="system",instance="$instance"})</td></tr><tr><td>load15</td><td>sum(node_load15{instance="$instance"}) / count(node_cpu_seconds_total{mode="system",instance=~"$instance"})</td></tr></table> |

**Node Memory Utilization**

| Catalog | Expression |
| --- | --- |
| Detail | `1 - sum(node_memory_MemAvailable_bytes{instance=~"$instance"}) / sum(node_memory_MemTotal_bytes{instance=~"$instance"})` |
| Summary | `` `1 - sum(node_memory_MemAvailable_bytes{instance=~"$instance"}) / sum(node_memory_MemTotal_bytes{instance=~"$instance"}) ` `` |

**Node Disk Utilization**

| Catalog | Expression |
| --- | --- |
| Detail | `(sum(node_filesystem_size_bytes{device!="rootfs",instance=~"$instance"}) by (device) - sum(node_filesystem_free_bytes{device!="rootfs",instance=~"$instance"}) by (device)) / sum(node_filesystem_size_bytes{device!="rootfs",instance=~"$instance"}) by (device)` |
| Summary | `(sum(node_filesystem_size_bytes{device!="rootfs",instance=~"$instance"}) - sum(node_filesystem_free_bytes{device!="rootfs",instance=~"$instance"})) / sum(node_filesystem_size_bytes{device!="rootfs",instance=~"$instance"})` |

**Node Disk I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>read</td><td>sum(rate(node_disk_read_bytes_total{instance="$instance"}[5m]))</td></tr><tr><td>written</td><td>sum(rate(node_disk_written_bytes_total{instance="$instance"}[5m]))</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>read</td><td>sum(rate(node_disk_read_bytes_total{instance="$instance"}[5m]))</td></tr><tr><td>written</td><td>sum(rate(node_disk_written_bytes_total{instance="$instance"}[5m]))</td></tr></table> |

**Node Network Packets**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive-dropped</td><td>sum(rate(node_network_receive_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>receive-errs</td><td>sum(rate(node_network_receive_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>receive-packets</td><td>sum(rate(node_network_receive_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>transmit-dropped</td><td>sum(rate(node_network_transmit_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>transmit-errs</td><td>sum(rate(node_network_transmit_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>transmit-packets</td><td>sum(rate(node_network_transmit_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>receive-dropped</td><td>sum(rate(node_network_receive_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>receive-errs</td><td>sum(rate(node_network_receive_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>receive-packets</td><td>sum(rate(node_network_receive_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>transmit-dropped</td><td>sum(rate(node_network_transmit_drop_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>transmit-errs</td><td>sum(rate(node_network_transmit_errs_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>transmit-packets</td><td>sum(rate(node_network_transmit_packets_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr></table> |

**Node Network I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive</td><td>sum(rate(node_network_receive_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr><tr><td>transmit</td><td>sum(rate(node_network_transmit_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m])) by (device)</td></tr></table> |
| Summary | <table><tr><td>receive</td><td>sum(rate(node_network_receive_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr><tr><td>transmit</td><td>sum(rate(node_network_transmit_bytes_total{device!~"lo \| veth.* \| docker.* \| flannel.* \| cali.* \| cbr.*",instance=~"$instance"}[5m]))</td></tr></table> |

### Etcd Metrics

#### Etcd Has a Leader

```
max(etcd_server_has_leader)
```

#### Number of Times the Leader Changes

```
max(etcd_server_leader_changes_seen_total)
```

#### Number of Failed Proposals

```
sum(etcd_server_proposals_failed_total)
```

#### GRPC Client Traffic

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>in</td><td>sum(rate(etcd_network_client_grpc_received_bytes_total[5m])) by (instance)</td></tr><tr><td>out</td><td>sum(rate(etcd_network_client_grpc_sent_bytes_total[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>in</td><td>sum(rate(etcd_network_client_grpc_received_bytes_total[5m]))</td></tr><tr><td>out</td><td>sum(rate(etcd_network_client_grpc_sent_bytes_total[5m]))</td></tr></table> |

#### Peer Traffic

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>in</td><td>sum(rate(etcd_network_peer_received_bytes_total[5m])) by (instance)</td></tr><tr><td>out</td><td>sum(rate(etcd_network_peer_sent_bytes_total[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>in</td><td>sum(rate(etcd_network_peer_received_bytes_total[5m]))</td></tr><tr><td>out</td><td>sum(rate(etcd_network_peer_sent_bytes_total[5m]))</td></tr></table> |

#### DB Size

| Catalog | Expression |
| --- | --- |
| Detail | sum(etcd_debugging_mvcc_db_total_size_in_bytes) by (instance) |
| Summary | sum(etcd_debugging_mvcc_db_total_size_in_bytes) |

#### Active Streams

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>lease-watch</td><td>sum(grpc_server_started_total{grpc_service="etcdserverpb.Lease",grpc_type="bidi_stream"}) by (instance) - sum(grpc_server_handled_total{grpc_service="etcdserverpb.Lease",grpc_type="bidi_stream"}) by (instance)</td></tr><tr><td>watch</td><td>sum(grpc_server_started_total{grpc_service="etcdserverpb.Watch",grpc_type="bidi_stream"}) by (instance) - sum(grpc_server_handled_total{grpc_service="etcdserverpb.Watch",grpc_type="bidi_stream"}) by (instance)</td></tr></table> |
| Summary | <table><tr><td>lease-watch</td><td>sum(grpc_server_started_total{grpc_service="etcdserverpb.Lease",grpc_type="bidi_stream"}) - sum(grpc_server_handled_total{grpc_service="etcdserverpb.Lease",grpc_type="bidi_stream"})</td></tr><tr><td>watch</td><td>sum(grpc_server_started_total{grpc_service="etcdserverpb.Watch",grpc_type="bidi_stream"}) - sum(grpc_server_handled_total{grpc_service="etcdserverpb.Watch",grpc_type="bidi_stream"})</td></tr></table> |

**Raft Proposals**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>applied</td><td>sum(increase(etcd_server_proposals_applied_total[5m])) by (instance)</td></tr><tr><td>committed</td><td>sum(increase(etcd_server_proposals_committed_total[5m])) by (instance)</td></tr><tr><td>pending</td><td>sum(increase(etcd_server_proposals_pending[5m])) by (instance)</td></tr><tr><td>failed</td><td>sum(increase(etcd_server_proposals_failed_total[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>applied</td><td>sum(increase(etcd_server_proposals_applied_total[5m]))</td></tr><tr><td>committed</td><td>sum(increase(etcd_server_proposals_committed_total[5m]))</td></tr><tr><td>pending</td><td>sum(increase(etcd_server_proposals_pending[5m]))</td></tr><tr><td>failed</td><td>sum(increase(etcd_server_proposals_failed_total[5m]))</td></tr></table> |

**RPC Rate**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>total</td><td>sum(rate(grpc_server_started_total{grpc_type="unary"}[5m])) by (instance)</td></tr><tr><td>fail</td><td>sum(rate(grpc_server_handled_total{grpc_type="unary",grpc_code!="OK"}[5m])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>total</td><td>sum(rate(grpc_server_started_total{grpc_type="unary"}[5m]))</td></tr><tr><td>fail</td><td>sum(rate(grpc_server_handled_total{grpc_type="unary",grpc_code!="OK"}[5m]))</td></tr></table> |

**Disk Operations**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>commit-called-by-backend</td><td>sum(rate(etcd_disk_backend_commit_duration_seconds_sum[lm])) by (instance)</td></tr><tr><td>fsync-called-by-wal</td><td>sum(rate(etcd_disk_wal_fsync_duration_seconds_sum[lm])) by (instance)</td></tr></table> |
| Summary | <table><tr><td>commit-called-by-backend</td><td>sum(rate(etcd_disk_backend_commit_duration_seconds_sum[lm]))</td></tr><tr><td>fsync-called-by-wal</td><td>sum(rate(etcd_disk_wal_fsync_duration_seconds_sum[lm]))</td></tr></table> |

**Disk Sync Duration**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>wal</td><td>histogram_quantile(0.99, sum(rate(etcd_disk_wal_fsync_duration_seconds_bucket[5m])) by (instance, le))</td></tr><tr><td>db</td><td>histogram_quantile(0.99, sum(rate(etcd_disk_backend_commit_duration_seconds_bucket[5m])) by (instance, le))</td></tr></table> |
| Summary | <table><tr><td>wal</td><td>sum(histogram_quantile(0.99, sum(rate(etcd_disk_wal_fsync_duration_seconds_bucket[5m])) by (instance, le)))</td></tr><tr><td>db</td><td>sum(histogram_quantile(0.99, sum(rate(etcd_disk_backend_commit_duration_seconds_bucket[5m])) by (instance, le)))</td></tr></table> |

## Kubernetes Components Metrics

### API Server Request Latency

| Catalog | Expression |
| --- | --- |
| Detail | `avg(apiserver_request_latencies_sum / apiserver_request_latencies_count) by (instance, verb) /1e+06` |
| Summary | `avg(apiserver_request_latencies_sum / apiserver_request_latencies_count) by (instance) /1e+06` |

### API Server Request Rate

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(apiserver_request_count[5m])) by (instance, code)` |
| Summary | `sum(rate(apiserver_request_count[5m])) by (instance)` |

### Scheduling Failed Pods

| Catalog | Expression |
| --- | --- |
| Detail | `sum(kube_pod_status_scheduled{condition="false"})` |
| Summary | `sum(kube_pod_status_scheduled{condition="false"})` |

### Controller Manager Queue Depth

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>volumes</td><td>`sum(volumes_depth) by instance`</td></tr><tr><td>deployment</td><td>`sum(deployment_depth) by instance`</td></tr><tr><td>replicaset</td><td>`sum(replicaset_depth) by instance`</td></tr><tr><td>service</td><td>`sum(service_depth) by instance`</td></tr><tr><td>serviceaccount</td><td>`sum(serviceaccount_depth) by instance`</td></tr><tr><td>endpoint</td><td>`sum(endpoint_depth) by instance`</td></tr><tr><td>daemonset</td><td>`sum(daemonset_depth) by instance`</td></tr><tr><td>statefulset</td><td>`sum(statefulset_depth) by instance`</td></tr><tr><td>replicationmanager</td><td>`sum(replicationmanager_depth) by instance`</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>volumes</td><td>sum(volumes_depth)</td></tr><tr><td>deployment</td><td>sum(deployment_depth)</td></tr><tr><td>replicaset</td><td>sum(replicaset_depth)</td></tr><tr><td>service</td><td>sum(service_depth)</td></tr><tr><td>serviceaccount</td><td>sum(serviceaccount_depth)</td></tr><tr><td>endpoint</td><td>sum(endpoint_depth)</td></tr><tr><td>daemonset</td><td>sum(daemonset_depth)</td></tr><tr><td>statefulset</td><td>sum(statefulset_depth)</td></tr><tr><td>replicationmanager</td><td>sum(replicationmanager_depth)</td></tr></table> |

**Scheduler E2E Scheduling Latency**

| Catalog | Expression |
| --- | --- |
| Detail | `histogram_quantile(0.99, sum(scheduler_e2e_scheduling_latency_microseconds_bucket) by (le, instance)) / 1e+06` |
| Summary | `sum(histogram_quantile(0.99, sum(scheduler_e2e_scheduling_latency_microseconds_bucket) by (le, instance)) / 1e+06)` |

**Scheduler Preemption Attempts**

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(scheduler_total_preemption_attempts[5m])) by (instance)` |
| Summary | `sum(rate(scheduler_total_preemption_attempts[5m]))` |

**Ingress Controller Connections**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>reading</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="reading"}) by (instance)</td></tr><tr><td>waiting</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="waiting"}) by (instance)</td></tr><tr><td>writing</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="writing"}) by (instance)</td></tr><tr><td>accepted</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="accepted"}[5m]))) by (instance)</td></tr><tr><td>active</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="active"}[5m]))) by (instance)</td></tr><tr><td>handled</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="handled"}[5m]))) by (instance)</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>reading</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="reading"})</td></tr><tr><td>waiting</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="waiting"})</td></tr><tr><td>writing</td><td>sum(nginx_ingress_controller_nginx_process_connections{state="writing"})</td></tr><tr><td>accepted</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="accepted"}[5m])))</td></tr><tr><td>active</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="active"}[5m])))</td></tr><tr><td>handled</td><td>sum(ceil(increase(nginx_ingress_controller_nginx_process_connections_total{state="handled"}[5m])))</td></tr></table> |

#### Ingress Controller Request Process Time

| Catalog | Expression |
| --- | --- |
| Detail | `topk(10, histogram_quantile(0.95,sum by (le, host, path)(rate(nginx_ingress_controller_request_duration_seconds_bucket{host!="_"}[5m]))))` |
| Summary | `topk(10, histogram_quantile(0.95,sum by (le, host)(rate(nginx_ingress_controller_request_duration_seconds_bucket{host!="_"}[5m]))))` |

### Rancher Logging Metrics

#### Fluentd Buffer Queue Rate

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(fluentd_output_status_buffer_queue_length[5m])) by (instance)` |
| Summary | `sum(rate(fluentd_output_status_buffer_queue_length[5m]))` |

#### Fluentd Input Rate

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(fluentd_input_status_num_records_total[5m])) by (instance)` |
| Summary | `sum(rate(fluentd_input_status_num_records_total[5m]))` |

#### Fluentd Output Errors Rate

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(fluentd_output_status_num_errors[5m])) by (type)` |

| Catalog | Expression |
| --- | --- |
| Summary | `sum(rate(fluentd_output_status_num_errors[5m]))` |

#### Fluentd Output Rate

| Catalog | Expression |
| --- | --- |
| Detail | `sum(rate(fluentd_output_status_num_records_total[5m])) by (instance)` |
| Summary | `sum(rate(fluentd_output_status_num_records_total[5m]))` |

### Workload Metrics

#### Workload CPU Utilization

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>cfs throttled seconds</td><td>`sum(rate(container_cpu_cfs_throttled_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>user seconds</td><td>`sum(rate(container_cpu_user_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>system seconds</td><td>`sum(rate(container_cpu_system_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>usage seconds</td><td>`sum(rate(container_cpu_usage_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr></table> |
| Summary | <table><tr><td>cfs throttled seconds</td><td>`sum(rate(container_cpu_cfs_throttled_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m]))`</td></tr><tr><td>user seconds</td><td>`sum(rate(container_cpu_user_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m]))`</td></tr><tr><td>system seconds</td><td>`sum(rate(container_cpu_system_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m]))`</td></tr><tr><td>usage seconds</td><td>`sum(rate(container_cpu_usage_seconds_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m]))`</td></tr></table> |

#### Workload Memory Utilization

| Catalog | Expression |
| --- | --- |
| Detail | `sum(container_memory_working_set_bytes{namespace="$namespace",pod_name=~"$podName", container_name!=""}) by (pod_name)` |
| Summary | `sum(container_memory_working_set_bytes{namespace="$namespace",pod_name=~"$podName", container_name!=""})` |

**Workload Network Packets**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive-packets</td><td>`sum(rate(container_network_receive_packets_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>receive-dropped</td><td>`sum(rate(container_network_receive_packets_dropped_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>receive-errors</td><td>`sum(rate(container_network_receive_errors_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>transmit-packets</td><td>`sum(rate(container_network_transmit_packets_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>transmit-dropped</td><td>`sum(rate(container_network_transmit_packets_dropped_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr><tr><td>transmit-errors</td><td>`sum(rate(container_network_transmit_errors_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)`</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>receive-packets</td><td>sum(rate(container_network_receive_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-dropped</td><td>sum(rate(container_network_receive_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-errors</td><td>sum(rate(container_network_receive_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-packets</td><td>sum(rate(container_network_transmit_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-dropped</td><td>sum(rate(container_network_transmit_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-errors</td><td>sum(rate(container_network_transmit_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

**Workload Network I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive</td><td>sum(rate(container_network_receive_bytes_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)</td></tr><tr><td>transmit</td><td>sum(rate(container_network_transmit_bytes_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)</td></tr></table> |
| Summary | <table><tr><td>receive</td><td>sum(rate(container_network_receive_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit</td><td>sum(rate(container_network_transmit_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

**Workload Disk I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>read</td><td>sum(rate(container_fs_reads_bytes_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)</td></tr><tr><td>write</td><td>sum(rate(container_fs_writes_bytes_total{namespace="$namespace",pod_name=~"$podName",container_name!=""}[5m])) by (pod_name)</td></tr></table> |
| Summary | <table><tr><td>read</td><td>sum(rate(container_fs_reads_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>write</td><td>sum(rate(container_fs_writes_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

## Pod Metrics

### Pod CPU Utilization

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>cfs throttled seconds</td><td>sum(rate(container_cpu_cfs_throttled_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr><tr><td>usage seconds</td><td>sum(rate(container_cpu_usage_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr><tr><td>system seconds</td><td>sum(rate(container_cpu_system_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr><tr><td>user seconds</td><td>sum(rate(container_cpu_user_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr></table> |

| Catalog | Expression |
| --- | --- |
| Summary | <table><tr><td>cfs throttled seconds</td><td>sum(rate(container_cpu_cfs_throttled_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName", container_name!=""}[5m]))</td></tr><tr><td>usage seconds</td><td>sum(rate(container_cpu_usage_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName", container_name!=""}[5m]))</td></tr><tr><td>system seconds</td><td>sum(rate(container_cpu_system_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName", container_name!=""}[5m]))</td></tr><tr><td>user seconds</td><td>sum(rate(container_cpu_user_seconds_total{container_name!="POD",namespace="$namespace",pod_name="$podName", container_name!=""}[5m]))</td></tr></table> |

**Pod Memory Utilization**

| Catalog | Expression |
| --- | --- |
| Detail | `sum(container_memory_working_set_bytes{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""}) by (container_name)` |
| Summary | `sum(container_memory_working_set_bytes{container_name!="POD",namespace="$namespace",pod_name="$podName",container_name!=""})` |

**Pod Network Packets**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive-packets</td><td>sum(rate(container_network_receive_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-dropped</td><td>sum(rate(container_network_receive_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-errors</td><td>sum(rate(container_network_receive_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-packets</td><td>sum(rate(container_network_transmit_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-dropped</td><td>sum(rate(container_network_transmit_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-errors</td><td>sum(rate(container_network_transmit_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |
| Summary | <table><tr><td>receive-packets</td><td>sum(rate(container_network_receive_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-dropped</td><td>sum(rate(container_network_receive_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>receive-errors</td><td>sum(rate(container_network_receive_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-packets</td><td>sum(rate(container_network_transmit_packets_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-dropped</td><td>sum(rate(container_network_transmit_packets_dropped_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit-errors</td><td>sum(rate(container_network_transmit_errors_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

**Pod Network I/O**

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>receive</td><td>sum(rate(container_network_receive_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit</td><td>sum(rate(container_network_transmit_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |
| Summary | <table><tr><td>receive</td><td>sum(rate(container_network_receive_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>transmit</td><td>sum(rate(container_network_transmit_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

#### Pod Disk I/O

| Catalog | Expression |
| --- | --- |
| Detail | <table><tr><td>read</td><td>sum(rate(container_fs_reads_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr><tr><td>write</td><td>sum(rate(container_fs_writes_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m])) by (container_name)</td></tr></table> |
| Summary | <table><tr><td>read</td><td>sum(rate(container_fs_reads_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr><tr><td>write</td><td>sum(rate(container_fs_writes_bytes_total{namespace="$namespace",pod_name="$podName",container_name!=""}[5m]))</td></tr></table> |

### Container Metrics

#### Container CPU Utilization

| Catalog | Expression |
| --- | --- |
| cfs throttled seconds | sum(rate(container_cpu_cfs_throttled_seconds_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m])) |
| usage seconds | sum(rate(container_cpu_usage_seconds_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m])) |
| system seconds | sum(rate(container_cpu_system_seconds_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m])) |
| user seconds | sum(rate(container_cpu_user_seconds_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m])) |

#### Container Memory Utilization

```
sum(container_memory_working_set_bytes{namespace="$namespace",pod_name="$podName",container_name="$containerName"})
```

#### Container Disk I/O

| Catalog | Expression |
|---|---|
| read | `sum(rate(container_fs_reads_bytes_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m]))` |
| write | `sum(rate(container_fs_writes_bytes_total{namespace="$namespace",pod_name="$podName",container_name="$containerName"}[5m]))` |

## Role-based Access Control



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

This section describes the expectations for RBAC for Rancher Monitoring.

#### Cluster Admins

By default, only those with the cluster-admin `ClusterRole` should be able to:

- Install the `rancher-monitoring` App onto a cluster and all other relevant configuration performed on the chart deploy
  - e.g. whether default dashboards are created, what exporters are deployed onto the cluster to collect metrics, etc.
- Create / modify / delete Prometheus deployments in the cluster via Prometheus CRs
- Create / modify / delete Alertmanager deployments in the cluster via Alertmanager CRs
- Persist new Grafana dashboards or datasources via creating ConfigMaps in the appropriate namespace
- Expose certain Prometheus metrics to the k8s Custom Metrics API for HPA via a Secret in the `cattle-monitoring-system` namespace

#### Users with Kubernetes ClusterRole-based Permissions

The `rancher-monitoring` chart installs the following three `ClusterRoles`. By default, they aggregate into the corresponding k8s `ClusterRoles`:

| ClusterRole | Aggregates To Default K8s ClusterRole |
|---|---|
| `monitoring-admin` | `admin` |
| `monitoring-edit` | `edit` |
| `monitoring-view` | `view` |

These `ClusterRoles` provide different levels of access to the Monitoring CRDs based on the actions that can be performed:

| CRDs (monitoring.coreos.com) | Admin | Edit | View |
|---|---|---|---|
| • `prometheuses`<br>• `alertmanagers` | Get, List, Watch | Get, List, Watch | Get, List, Watch |
| • `servicemonitors`<br>• `podmonitors`<br>• `prometheusrules` | * | * | Get, List, Watch |

On a high level, the following permissions are assigned by default as a result.

#### Users with Kubernetes Admin/Edit Permissions

Only those with the cluster-admin, admin or edit `ClusterRole` should be able to:

- Modify the scrape configuration of Prometheus deployments via ServiceMonitor and PodMonitor CRs
- Modify the alerting / recording rules of a Prometheus deployment via PrometheusRules CRs

#### Users with Kubernetes View Permissions

Only those with who have some Kubernetes `ClusterRole` should be able to:

- View the configuration of Prometheuses that are deployed within the cluster
- View the configuration of Alertmanagers that are deployed within the cluster
- View the scrape configuration of Prometheus deployments via ServiceMonitor and PodMonitor CRs
- View the alerting/recording rules of a Prometheus deployment via PrometheusRules CRs

#### Additional Monitoring Roles

Monitoring also creates additional `Roles` that are not assigned to users by default but are created within the cluster. They can be bound to a namespace by deploying a `RoleBinding` that references it. To define a `RoleBinding` with `kubectl` instead of through Rancher, click here.

Admins should use these roles to provide more fine-grained access to users:

| Role | Purpose |
|---|---|
| monitoring-config-admin | Allow admins to assign roles to users to be able to view / modify Secrets and ConfigMaps within the cattle-monitoring-system namespace. Modifying Secrets / ConfigMaps in this namespace could allow users to alter the cluster's Alertmanager configuration, Prometheus Adapter configuration, additional Grafana datasources, TLS secrets, etc. |

| Role | Purpose |
| --- | --- |
| monitoring-config-edit | Allow admins to assign roles to users to be able to view / modify Secrets and ConfigMaps within the cattle-monitoring-system namespace. Modifying Secrets / ConfigMaps in this namespace could allow users to alter the cluster's Alertmanager configuration, Prometheus Adapter configuration, additional Grafana datasources, TLS secrets, etc. |
| monitoring-config-view | Allow admins to assign roles to users to be able to view Secrets and ConfigMaps within the cattle-monitoring-system namespace. Viewing Secrets / ConfigMaps in this namespace could allow users to observe the cluster's Alertmanager configuration, Prometheus Adapter configuration, additional Grafana datasources, TLS secrets, etc. |
| monitoring-dashboard-admin | Allow admins to assign roles to users to be able to edit / view ConfigMaps within the cattle-dashboards namespace. ConfigMaps in this namespace will correspond to Grafana Dashboards that are persisted onto the cluster. |
| monitoring-dashboard-edit | Allow admins to assign roles to users to be able to edit / view ConfigMaps within the cattle-dashboards namespace. ConfigMaps in this namespace will correspond to Grafana Dashboards that are persisted onto the cluster. |
| monitoring-dashboard-view | Allow admins to assign roles to users to be able to view ConfigMaps within the cattle-dashboards namespace. ConfigMaps in this namespace will correspond to Grafana Dashboards that are persisted onto the cluster. |

#### Assigning Monitoring Roles through Custom Roles

Admins may assign custom roles in the Rancher UI for admin, editing, and viewing monitoring. These "roles" are created by default when the monitoring app is installed. Additionally, these roles are also deployed to the corresponding Kubernetes roles: admin, edit, and view `ClusterRoles`.



The UI won't offer `monitoring-admin`, `monitoring-edit`, and `monitoring-view` options when users are being added to a cluster. These monitoring roles can only be assigned by manually creating a custom role that inherits from Project Owner and Project Monitoring View roles.

1. Create the custom role:

```
1.1 Click **☰ > Users & Authentication > Roles**.

1.2 Select the appropriate tab, e.g., **Cluster** role. Then click
**Create Cluster Role**.

1.3 In the **Name** field, create a custom role such as `View
Monitoring`, `Edit Monitoring`, or `Admin Monitoring`.

1.4 Click **Inherit From > Add Resource**, then select the Kubernetes
role, as applicable, from the dropdown.
```

```
1.5 Click **Create**.
```

1. Assign the custom role to a new user:

```
2.1 Click **☰ > Cluster Management > Cluster Explore > Cluster > Cluster
Members > Add**.

2.2 Search for your new user name from **Select Member** options
displayed.

2.3 Assign the new custom role from **Cluster Permissions** to the new
user.

2.4 Click **Create**.
```

**Result:** The new user should now be able to see the monitoring tools.

#### Additional Monitoring ClusterRoles

Monitoring also creates additional `ClusterRoles` that aren't assigned to users by default but are created within the cluster. They aren't aggregated by default but can be bound to a namespace by deploying a `RoleBinding` or `ClusterRoleBinding` that references it. To define a `RoleBinding` with `kubectl` instead of through Rancher, click here.

| Role | Purpose |
|---|---|
| monitoring-ui-view | This ClusterRole allows users with write access to the project to view metrics graphs for the specified cluster in the Rancher UI. This is done by granting Read-only access to external Monitoring UIs. Users with this role have permission to list the Prometheus, Alertmanager, and Grafana endpoints and make GET requests to Prometheus, Alertmanager, and Grafana UIs through the Rancher proxy.<br><br>This role doesn't grant access to monitoring endpoints. As a result, users with this role won't be able to view cluster monitoring graphs and dashboards in the Rancher UI; however, they are able to access the monitoring Grafana, Prometheus, and Alertmanager UIs if provided those links. |



A user bound to the **View Monitoring** Rancher role and read-only project permissions can't view links in the Monitoring UI. They can still access external monitoring UIs if provided links to those UIs. If you wish to grant access to users with the **View Monitoring** role and read-only project permissions, move the `cattle-monitoring-system` namespace into the project.

#### Assigning Roles and ClusterRoles with kubectl

##### Using `kubectl create`

One method is to use either `kubectl create clusterrolebinding` or `kubectl create`

`rolebinding` to assign a `Role` or `ClusterRole`. This is shown in the following examples:

- Assign to a specific user:

**clusterrolebinding**

```
kubectl create clusterrolebinding my-binding
--clusterrole=monitoring-ui-view --user=u-l4npx
```

**rolebinding**

```
kubectl create rolebinding my-binding --clusterrole=monitoring-ui
-view --user=u-l4npx --namespace=my-namespace
```

- Assign to all authenticated users:

**clusterrolebinding**

```
kubectl create clusterrolebinding my-binding
--clusterrole=monitoring-ui-view --group=system:authenticated
```

**rolebinding**

```
kubectl create rolebinding my-binding --clusterrole=monitoring-ui
-view --group=system:authenticated --namespace=my-namespace
```

#### Using YAML Files

Another method is to define bindings in YAML files that you create. You must first configure the `RoleBinding` or `ClusterRoleBinding` with a YAML file. Then, apply the configuration changes by running the `kubectl apply` command.

- **Roles**: Below is an example YAML file to help you configure `RoleBindings` in Kubernetes. You'll need to fill in the name below.



Names are case-sensitive.

```
# monitoring-config-view-role-binding.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: monitoring-config-view
  namespace: cattle-monitoring-system
roleRef:
```

```
  kind: Role
  name: monitoring-config-view
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: u-b4qkhsnliz # this can be found via `kubectl get users -A`
  apiGroup: rbac.authorization.k8s.io
```

- **kubectl**: Below is an example of a `kubectl` command used to apply the binding you've created in the YAML file. Remember to fill in your YAML filename accordingly.

```
kubectl apply -f monitoring-config-view-role-binding.yaml
```

#### Users with Rancher Based Permissions

The relationship between the default roles deployed by Rancher (i.e. cluster-owner, cluster-member, project-owner, project-member), the default Kubernetes roles, and the roles deployed by the rancher-monitoring chart are detailed in the table below:

*Table 12. Default Rancher Permissions and Corresponding Kubernetes ClusterRoles*

| Rancher Role | Kubernetes Role | Monitoring ClusterRole / Role | ClusterRoleBinding or RoleBinding? |
|---|---|---|---|
| cluster-owner | cluster-admin | N/A | ClusterRoleBinding |
| cluster-member | admin | monitoring-admin | ClusterRoleBinding |
| project-owner | admin | monitoring-admin | RoleBinding within Project namespace |
| project-member | edit | monitoring-edit | RoleBinding within Project namespace |

In addition to these default roles, the following Rancher project roles can be applied to members of your cluster to provide access to monitoring. These Rancher roles are tied to ClusterRoles deployed by the monitoring chart:

*Table 13. Non-default Rancher Permissions and Corresponding Kubernetes ClusterRoles*

| Rancher Role | Kubernetes ClusterRole | Available In Rancher From | Available in Monitoring v2 From |
|---|---|---|---|
| View Monitoring* | monitoring-ui-view | 2.4.8+ | 9.4.204+ |



A user bound to the **View Monitoring** Rancher role and read-only project permissions can't view links in the Monitoring UI. They can still access external monitoring UIs if provided links to those UIs. If you wish to grant access to users with the **View Monitoring** role and read-only project permissions, move the `cattle-monitoring-system` namespace into the project.

##### Differences in 2.5.x

Users with the project-member or project-owners roles assigned will not be given access to either

Prometheus or Grafana in Rancher 2.5.x since we only create Grafana or Prometheus on a cluster-level.

In addition, while project owners will still be only able to add ServiceMonitors / PodMonitors that scrape resources within their project's namespace by default, PrometheusRules are not scoped to a single namespace / project. Therefore, any alert rules or recording rules created by project-owners within their project namespace will be applied across the entire cluster, although they will be unable to view / edit / delete any rules that were created outside the project's namespace.

#### Assigning Additional Access

If cluster-admins would like to provide additional admin/edit access to users outside of the roles offered by the rancher-monitoring chart, the following table identifies the potential impact:

| CRDs (monitoring.coreos.com) | Can it cause impact outside of a namespace / project? | Impact |
|---|---|---|
| `prometheuses` | Yes, this resource can scrape metrics from any targets across the entire cluster (unless the Operator itself is otherwise configured). | User will be able to define the configuration of new cluster-level Prometheus deployments that should be created in the cluster. |
| `alertmanagers` | No | User will be able to define the configuration of new cluster-level Alertmanager deployments that should be created in the cluster. Note: if you just want to allow users to configure settings like Routes and Receivers, you should just provide access to the Alertmanager Config Secret instead. |
| • `servicemonitors`<br>• `podmonitors` | No, not by default; this is configurable via `ignoreNamespaceSelectors` on the Prometheus CR. | User will be able to set up scrapes by Prometheus on endpoints exposed by Services / Pods within the namespace they are given this permission in. |
| `prometheusrules` | Yes, PrometheusRules are cluster-scoped. | User will be able to define alert or recording rules on Prometheus based on any series collected across the entire cluster. |

| k8s Resources | Namespace | Can it cause impact outside of a namespace / project? | Impact |
|---|---|---|---|
| • `secrets`<br>• `configmaps` | `cattle-monitoring-system` | Yes, Configs and Secrets in this namespace can impact the entire monitoring / alerting pipeline. | User will be able to create or edit Secrets / ConfigMaps such as the Alertmanager Config, Prometheus Adapter Config, TLS secrets, additional Grafana datasources, etc. This can have broad impact on all cluster monitoring / alerting. |
| • `secrets`<br>• `configmaps` | `cattle-dashboards` | Yes, Configs and Secrets in this namespace can create dashboards that make queries on all metrics collected at a cluster-level. | User will be able to create Secrets / ConfigMaps that persist new Grafana Dashboards only. |

### Role-based Access Control for Grafana

Rancher allows any users who are authenticated by Kubernetes and have access the Grafana service deployed by the Rancher Monitoring chart to access Grafana via the Rancher Dashboard UI. By default, all users who are able to access Grafana are given the Viewer [https://grafana.com/docs/grafana/latest/permissions/organization_roles/#viewer-role] role, which allows them to view any of the default dashboards deployed by Rancher.

However, users can choose to log in to Grafana as an Admin [https://grafana.com/docs/grafana/latest/permissions/organization_roles/#admin-role] if necessary. The default Admin username and password for the Grafana instance will be `admin`/`prom-operator`, but alternative credentials can also be supplied on deploying or upgrading the chart.

To see the Grafana UI, install `rancher-monitoring`. Then:

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the visualizations and click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Grafana**.

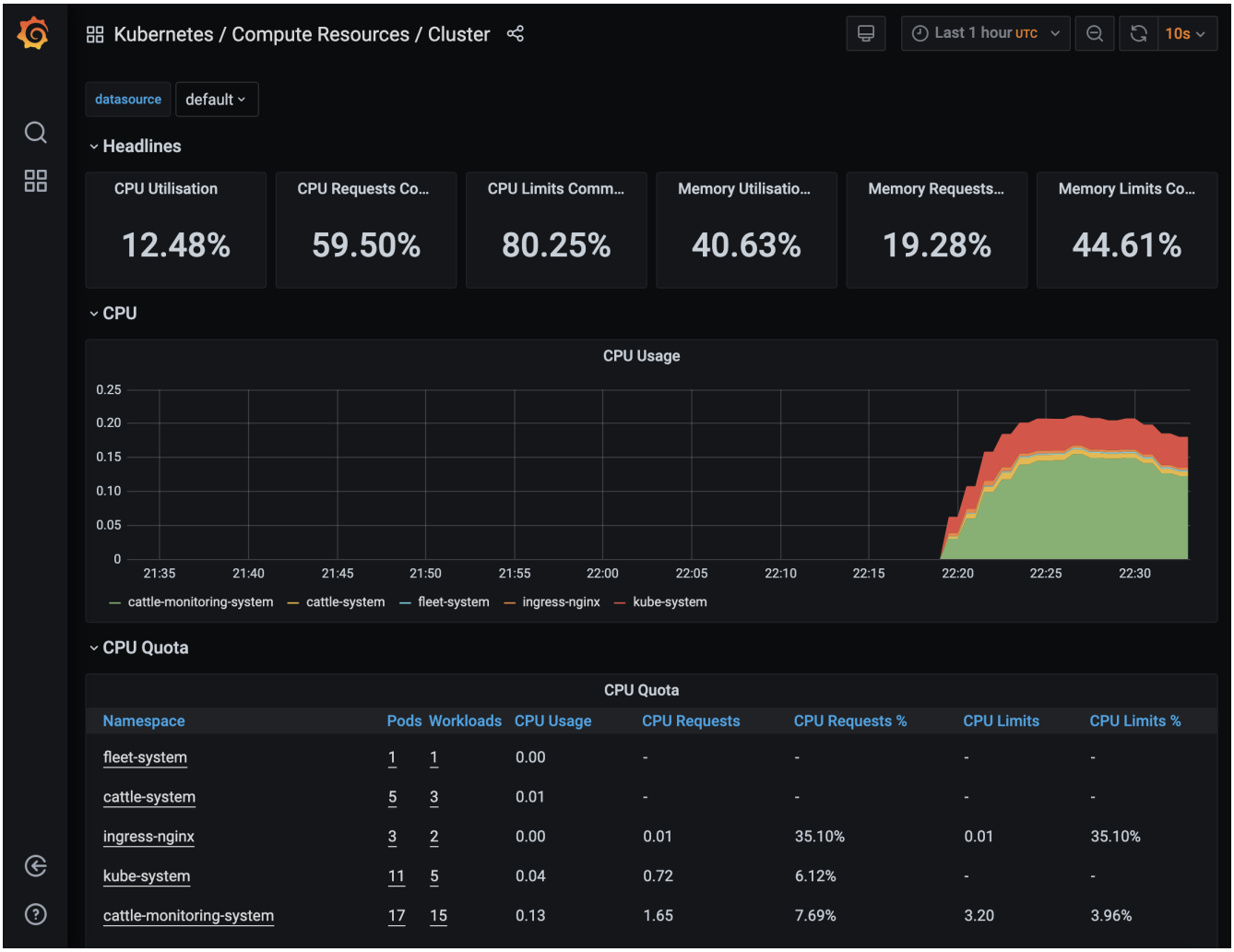



*Figure 25. Cluster Compute Resources Dashboard in Grafana*




*Figure 26. Default Dashboards in Grafana*

## Windows Cluster Support for Monitoring V2

> *Deprecation Notice*
>
> Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Monitoring V2 can be deployed on a Windows cluster to scrape metrics from Windows nodes using prometheus-community/windows_exporter [https://github.com/prometheus-community/windows_exporter] (previously named `wmi_exporter`).

### Cluster Requirements

Monitoring V2 for Windows can only scrape metrics from Windows hosts that have a minimum `wins` version of v0.1.0. To be able to fully deploy Monitoring V2 for Windows, all of your hosts must meet this requirement.

#### Upgrading Existing Clusters to wins v0.1.0

If the cluster was provisioned before Rancher 2.5.8 (even if the current Rancher version is 2.5.8), you will not be able to successfully deploy Monitoring V2 for Windows until you upgrade the wins version on each host to at least v0.1.0.

To facilitate this upgrade, Rancher 2.5.8 has released a brand new Helm chart called `rancher-wins-upgrader`.

1. Deploy `rancher-wins-upgrader` with the following override:

```
 # Masquerading bootstraps the wins-upgrader installation via
 # a previously whitelisted process path since the normal install
path,
 # c:\etc\rancher\wins\wins-upgrade.exe is not normally whitelisted.
 # In this case, we are using the previously whitelisted process
 # path used by Monitoring V1.
 masquerade:
   enabled: true
   as: c:\\etc\wmi-exporter\wmi-exporter.exe
```

2. Once all your hosts have been successfully upgraded, please ensure that you deploy the Helm chart once again with default values to avoid conflicts with the following settings:

```
 masquerade:
   enabled: false
```

**Result:** The hosts are ready for Monitoring V2 to be installed. You may choose to uninstall the `rancher-wins-upgrader` chart or keep it in your cluster to facilitate future upgrades.

For more information on how it can be used, please see the README.md [https://github.com/rancher/wins/blob/master/charts/rancher-wins-upgrader/README.md] of the chart.

## Enable Monitoring



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

As an administrator or cluster owner, you can configure Rancher to deploy Prometheus to monitor your Kubernetes cluster.

This page describes how to enable monitoring and alerting within a cluster using the new monitoring application.

You can enable monitoring with or without SSL.

### Requirements

- Allow traffic on port 9796 for each of your nodes. Prometheus scrapes metrics from these ports.
  - You may also need to allow traffic on port 10254 for each of your nodes, if PushProx is disabled (`ingressNginx.enabled` set to `false`), or you've upgraded from a previous Rancher version that had v1 monitoring already installed.
- Make sure that your cluster fulfills the resource requirements. The cluster should have at least 1950Mi memory available, 2700m CPU, and 50Gi storage. See Configuring Resource Limits and Requests for a breakdown of the resource limits and requests.



If you want to set up Alertmanager, Grafana or Ingress, it has to be done with the settings on the Helm chart deployment. It's problematic to create Ingress outside the deployment.

### Setting Resource Limits and Requests

The resource requests and limits can be configured when installing `rancher-monitoring`. To configure Prometheus resources from the Rancher UI, click **Apps › Monitoring** in the upper left corner.

For more information about the default limits, see this page.

### Install the Monitoring Application

#### Enable Monitoring for use without SSL

1. Click ☰ **> Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click **Cluster Tools** (bottom left corner).
4. Click **Install** by Monitoring.
5. Optional: Customize requests, limits and more for Alerting, Prometheus, and Grafana in the Values step. For help, refer to the configuration reference.

**Result:** The monitoring app is deployed in the `cattle-monitoring-system` namespace.

#### Enable Monitoring for use with SSL

1. Follow the steps on this page to create a secret in order for SSL to be used for alerts.
   - The secret should be created in the `cattle-monitoring-system` namespace. If it doesn't exist, create it first.
   - Add the `ca`, `cert`, and `key` files to the secret.
2. In the upper left corner, click ☰ **> Cluster Management**.
3. On the **Clusters** page, go to the cluster where you want to enable monitoring for use with SSL and click **Explore**.
4. Click **Apps › Charts**.
5. Click **Monitoring**.
6. Click **Install** or **Update**, depending on whether you have already installed Monitoring.

7. Check the box for **Customize Helm options before install** and click **Next**.
8. Click **Alerting**.
9. In the **Additional Secrets** field, add the secrets created earlier.

**Result:** The monitoring app is deployed in the `cattle-monitoring-system` namespace.

When creating a receiver, SSL-enabled receivers such as email or webhook will have a **SSL** section with fields for **CA File Path**, **Cert File Path**, and **Key File Path**. Fill in these fields with the paths to each of `ca`, `cert`, and `key`. The path will be of the form `/etc/alertmanager/secrets/name-of-file-in-secret`.

For example, if you created a secret with these key-value pairs:

```
ca.crt=`base64-content`
cert.pem=`base64-content`
key.pfx=`base64-content`
```

Then **Cert File Path** would be set to `/etc/alertmanager/secrets/cert.pem`.

#### Rancher Performance Dashboard

When monitoring is installed on the upstream (local) cluster, you are given basic health metrics about the Rancher pods, such as CPU and memory data. To get advanced metrics for your local Rancher server, you must additionally enable the Rancher Performance Dashboard for Grafana.

This dashboard provides access to the following advanced metrics:

- Handler Average Execution Times Over Last 5 Minutes
- Rancher API Average Request Times Over Last 5 Minutes
- Subscribe Average Request Times Over Last 5 Minutes
- Lasso Controller Work Queue Depth (Top 20)
- Number of Rancher Requests (Top 20)
- Number of Failed Rancher API Requests (Top 20)
- K8s Proxy Store Average Request Times Over Last 5 Minutes (Top 20)
- K8s Proxy Client Average Request Times Over Last 5 Minutes (Top 20)
- Cached Objects by GroupVersionKind (Top 20)
- Lasso Handler Executions (Top 20)
- Handler Executions Over Last 2 Minutes (Top 20)
- Total Handler Executions with Error (Top 20)
- Data Transmitted by Remote Dialer Sessions (Top 20)
- Errors for Remote Dialer Sessions (Top 20)
- Remote Dialer Connections Removed (Top 20)
- Remote Dialer Connections Added by Client (Top 20)



Profiling data (such as advanced memory or CPU analysis) is not present as it is a very

> context-dependent technique that's meant for debugging and not intended for normal observation.

#### Enabling the Rancher Performance Dashboard

To enable the Rancher Performance Dashboard:

**Helm**

Use the following options with the Helm CLI:

```
--set extraEnv\[0\].name="CATTLE_PROMETHEUS_METRICS" --set-string
extraEnv\[0\].value=true
```

You can also include the following snippet in your Rancher Helm chart's values.yaml file:

```
extraEnv:
  - name: "CATTLE_PROMETHEUS_METRICS"
    value: "true"
```

**UI**

1. Click ☰ > **Cluster Management**.
2. Go to the row of the `local` cluster and click **Explore**.
3. Click **Workloads** › **Deployments**.
4. Use the dropdown menu at the top to filter for **All Namespaces**.
5. Under the `cattle-system` namespace, go to the `rancher` row and click ☰ > **Edit Config**
6. Under **Environment Variables**, click **Add Variable**.
7. For **Type**, select `Key/Value Pair`.
8. For **Variable Name**, enter `CATTLE_PROMETHEUS_METRICS`.
9. For **Value**, enter `true`.
10. Click **Save** to apply the change.

#### Accessing the Rancher Performance Dashboard

1. Click ☰ > **Cluster Management**.
2. Go to the row of the `local` cluster and click **Explore**.
3. Click **Monitoring**
4. Select the **Grafana** dashboard.
5. From the sidebar, click **Search dashboards**.
6. Enter `Rancher Performance Debugging` and select it.

## Uninstall Monitoring



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click **Apps**.
4. Click **Installed Apps**.
5. Go to the `cattle-monitoring-system` namespace and check the boxes for `rancher-monitoring-crd` and `rancher-monitoring`.
6. Click **Delete**.
7. Confirm **Delete**.

**Result:** `rancher-monitoring` is uninstalled.



*Persistent Grafana Dashboards:*

For users who are using Monitoring V2 v9.4.203 or below, uninstalling the Monitoring chart will delete the cattle-dashboards namespace, which will delete all persisted dashboards, unless the namespace is marked with the annotation `helm.sh/resource-policy: "keep"`. This annotation is added by default in Monitoring V2 v14.5.100+ but can be manually applied on the cattle-dashboards namespace before an uninstall if an older version of the Monitoring chart is currently installed onto your cluster.

## Setting up Monitoring for a Workload



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

If you only need CPU and memory time series for the workload, you don't need to deploy a ServiceMonitor or PodMonitor because the monitoring application already collects metrics data on resource usage by default.

The steps for setting up monitoring for workloads depend on whether you want basic metrics such as CPU and memory for the workload, or whether you want to scrape custom metrics from the workload.

If you only need CPU and memory time series for the workload, you don't need to deploy a ServiceMonitor or PodMonitor because the monitoring application already collects metrics data on resource usage by default. The resource usage time series data is in Prometheus's local time series database.

Grafana shows the data in aggregate, but you can see the data for the individual workload by using a PromQL query that extracts the data for that workload. Once you have the PromQL query, you can execute the query individually in the Prometheus UI and see the time series visualized there, or you can use the query to customize a Grafana dashboard to display the workload metrics. For examples of PromQL queries for workload metrics, see this section.

To set up custom metrics for your workload, you will need to set up an exporter and create a new ServiceMonitor custom resource to configure Prometheus to scrape metrics from your exporter.

#### Display CPU and Memory Metrics for a Workload

By default, the monitoring application already scrapes CPU and memory.

To get some fine-grained detail for a particular workload, you can customize a Grafana dashboard to display the metrics for a particular workload.

#### Setting up Metrics Beyond CPU and Memory

For custom metrics, you will need to expose the metrics on your application in a format supported by Prometheus.

Then we recommend that you should create a new ServiceMonitor custom resource. When this resource is created, the Prometheus custom resource will be automatically updated so that its scrape configuration includes the new custom metrics endpoint. Then Prometheus will begin scraping metrics from the endpoint.

You can also create a PodMonitor to expose the custom metrics endpoint, but ServiceMonitors are more appropriate for the majority of use cases.

#### Monitoring Configuration Examples



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

##### ServiceMonitor

See the official prometheus-operator GitHub repo for an example ServiceMonitor [https://github.com/prometheus-operator/prometheus-operator/blob/main/example/prometheus-operator-crd/monitoring.coreos.com_servicemonitors.yaml] YAML.

##### PodMonitor

See the Prometheus Operator documentation [https://prometheus-operator.dev/docs/developer/getting-started/#using-podmonitors] for an example PodMonitor and an example Prometheus resource that refers to a PodMonitor.

##### PrometheusRule

A PrometheusRule contains the alerting and recording rules that you would usually place in a Prometheus rule file [https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/].

For a more fine-grained approach, the `ruleSelector` field on a Prometheus resource can select which PrometheusRules should be loaded onto Prometheus, based on the labels attached to the PrometheusRules resources.

See the Prometheus Operator documentation [https://prometheus-operator.dev/docs/developer/alerting/] for an example PrometheusRule.

#### Alertmanager Config

See the Rancher docs page on Receivers for an example Alertmanager config.

### Helm Chart Options



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

#### Configuring Resource Limits and Requests

The resource requests and limits can be configured when installing `rancher-monitoring`.

The default values are in the values.yaml [https://github.com/rancher/charts/blob/main/charts/rancher-monitoring/values.yaml] in the `rancher-monitoring` Helm chart.

The default values in the table below are the minimum required resource limits and requests.

| Resource Name | Memory Limit | CPU Limit | Memory Request | CPU Request |
|---|---|---|---|---|
| alertmanager | 500Mi | 1000m | 100Mi | 100m |
| grafana | 200Mi | 200m | 100Mi | 100m |
| kube-state-metrics subchart | 200Mi | 100m | 130Mi | 100m |
| prometheus-node-exporter subchart | 50Mi | 200m | 30Mi | 100m |
| prometheusOperator | 500Mi | 200m | 100Mi | 100m |
| prometheus | 2500Mi | 1000m | 1750Mi | 750m |
| **Total** | **3950Mi** | **2700m** | **2210Mi** | **1250m** |

At least 50Gi storage is recommended.

#### Trusted CA for Notifiers

If you need to add a trusted CA to your notifier, follow these steps:

1. Create the `cattle-monitoring-system` namespace.
2. Add your trusted CA secret to the `cattle-monitoring-system` namespace.
3. Deploy or upgrade the `rancher-monitoring` Helm chart. In the chart options, reference the secret in **Alerting › Additional Secrets**.

**Result:** The default Alertmanager custom resource will have access to your trusted CA.

#### Additional Scrape Configurations

If the scrape configuration you want cannot be specified via a ServiceMonitor or PodMonitor at the moment, you can provide an `additionalScrapeConfigSecret` on deploying or upgrading `rancher-monitoring`.

A scrape_config section [https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config] specifies a set of targets and parameters describing how to scrape them. In the general case, one scrape configuration specifies a single job.

An example of where this might be used is with Istio. For more information, see this section.

#### Configuring Applications Packaged within Monitoring v2

We deploy kube-state-metrics and node-exporter with monitoring v2. The node exporters are deployed as DaemonSets. Each of these entities are deployed as sub-charts through the monitoring v2 Helm chart, values.yaml.

We also deploy Grafana, which is not managed by Prometheus.

Many values aren't exposed in the top level chart. However, you can add values to the top level chart to override values that exist in the sub-charts.

#### Increase the Replicas of Alertmanager

As part of the chart deployment options, you can opt to increase the number of replicas of the Alertmanager deployed onto your cluster. The replicas can all be managed using the same underlying Alertmanager Config Secret. For more information on the Alertmanager Config Secret, refer to this section.

#### Configuring the Namespace for a Persistent Grafana Dashboard

To specify that you would like Grafana to watch for ConfigMaps across all namespaces, set this value in the `rancher-monitoring` Helm chart:

```
grafana.sidecar.dashboards.searchNamespace=ALL
```

Note that the RBAC roles exposed by the Monitoring chart to add Grafana Dashboards are still restricted to giving permissions for users to add dashboards in the namespace defined in `grafana.dashboards.namespace`, which defaults to `cattle-dashboards`.

### Receiver Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

The Alertmanager Config [https://prometheus.io/docs/alerting/latest/configuration/#configuration-file] Secret

contains the configuration of an Alertmanager instance that sends out notifications based on alerts it receives from Prometheus.



This section assumes familiarity with how monitoring components work together. For more information about Alertmanager, see this section.

#### Creating Receivers in the Rancher UI



*Prerequisites:*

- The monitoring application needs to be installed.
- If you configured monitoring with an existing Alertmanager Secret, it must have a format that is supported by Rancher's UI. Otherwise you will only be able to make changes based on modifying the Alertmanager Secret directly. Note: We are continuing to make enhancements to what kinds of Alertmanager Configurations we can support using the Routes and Receivers UI, so please file an issue [https://github.com/rancher/rancher/issues/new] if you have a request for a feature enhancement.

1. Go to the cluster where you want to create receivers. Click **Monitoring -> Alerting -> AlertManagerConfigs**.
2. Click **Create**.
3. Enter a **Name** for the new AlertmanagerConfig.
4. Click **Create**.
5. After creating the AlertManagerConfig, click it to add a receiver.
6. Click **Add Receiver**.
7. Enter a **Name** for the receiver.
8. Configure one or more providers for the receiver. For help filling out the forms, refer to the configuration options below.
9. Click **Create**.

**Result:** Alerts can be configured to send notifications to the receiver(s).

#### Receiver Configuration

The notification integrations are configured with the `receiver`, which is explained in the Prometheus documentation. [https://prometheus.io/docs/alerting/latest/configuration/#receiver]

#### Native vs. Non-native Receivers

By default, AlertManager provides native integration with some receivers, which are listed in this section. [https://prometheus.io/docs/alerting/latest/configuration/#receiver] All natively supported receivers are configurable through the Rancher UI.

For notification mechanisms, such as Telegram, that are not natively supported by AlertManager, integration is achieved using the webhook receiver. [https://prometheus.io/docs/alerting/latest/configuration/#webhook_config] A list of third-party drivers providing such integrations can be found here. [https://prometheus.io/docs/operating/integrations/#alertmanager-webhook-receiver] Access to these drivers, and their associated integrations, is provided through the Alerting Drivers app. Once enabled, configuring non-native receivers can also be

done through the Rancher UI.

Currently the Rancher Alerting Drivers app provides access to the following integrations:

- Microsoft Teams, based on the prom2teams [https://github.com/idealista/prom2teams] driver.
- Telegram, based on the Sachet [https://github.com/messagebird/sachet] driver.

The following types of receivers can be configured in the Rancher UI:

- Slack
- Email
- PagerDuty
- Opsgenie
- Webhook
- Custom
- Teams
- SMS
- Telegram

The custom receiver option can be used to configure any receiver in YAML that cannot be configured by filling out the other forms in the Rancher UI.

**Slack**

| Field | Type | Description |
|---|---|---|
| URL | String | Enter your Slack webhook URL. For instructions to create a Slack webhook, see the Slack documentation. [https://get.slack.help/hc/en-us/articles/115005265063-Incoming-WebHooks-for-Slack] |
| Default Channel | String | Enter the name of the channel that you want to send alert notifications in the following format: `#<channelname>`. |
| Proxy URL | String | Proxy for the webhook notifications. |
| Enable Send Resolved Alerts | Bool | Whether to send a follow-up notification if an alert has been resolved (e.g. [Resolved] High CPU Usage). |

**Email**

| Field | Type | Description |
|---|---|---|
| Default Recipient Address | String | The email address that will receive notifications. |
| Enable Send Resolved Alerts | Bool | Whether to send a follow-up notification if an alert has been resolved (e.g. [Resolved] High CPU Usage). |

SMTP options:

| Field | Type | Description |
|---|---|---|
| Sender | String | Enter an email address available on your SMTP mail server that you want to send the notification from. |
| Host | String | Enter the IP address or hostname for your SMTP server. Example: `smtp.email.com`. |
| Use TLS | Bool | Use TLS for encryption. |
| Username | String | Enter a username to authenticate with the SMTP server. |
| Password | String | Enter a password to authenticate with the SMTP server. |

**PagerDuty**

| Field | Type | Description |
|---|---|---|
| Integration Type | String | `Events API v2` or `Prometheus`. |
| Default Integration Key | String | For instructions to get an integration key, see the PagerDuty documentation. [https://www.pagerduty.com/docs/guides/prometheus-integration-guide/] |
| Proxy URL | String | Proxy for the PagerDuty notifications. |
| Enable Send Resolved Alerts | Bool | Whether to send a follow-up notification if an alert has been resolved (e.g. [Resolved] High CPU Usage). |

**Opsgenie**

| Field | Description |
|---|---|
| API Key | For instructions to get an API key, refer to the Opsgenie documentation. [https://docs.opsgenie.com/docs/api-key-management] |
| Proxy URL | Proxy for the Opsgenie notifications. |
| Enable Send Resolved Alerts | Whether to send a follow-up notification if an alert has been resolved (e.g. [Resolved] High CPU Usage). |

Opsgenie Responders:

| Field | Type | Description |
|---|---|---|
| Type | String | Schedule, Team, User, or Escalation. For more information on alert responders, refer to the Opsgenie documentation. [https://docs.opsgenie.com/docs/alert-recipients-and-teams] |
| Send To | String | Id, Name, or Username of the Opsgenie recipient. |

**Webhook**

| Field | Description |
|---|---|
| URL | Webhook URL for the app of your choice. |
| Proxy URL | Proxy for the webhook notification. |
| Enable Send Resolved Alerts | Whether to send a follow-up notification if an alert has been resolved (e.g. [Resolved] High CPU Usage). |

#### Custom

The YAML provided here will be directly appended to your receiver within the Alertmanager Config Secret.

#### Teams

##### Enabling the Teams Receiver for Rancher Managed Clusters

The Teams receiver is not a native receiver and must be enabled before it can be used. You can enable the Teams receiver for a Rancher managed cluster by going to the Apps page and installing the rancher-alerting-drivers app with the Teams option selected.

1. In the Rancher UI, go to the cluster where you want to install rancher-alerting-drivers and click **Apps**.
2. Click the **Alerting Drivers** app.
3. Click the **Helm Deploy Options** tab.
4. Select the **Teams** option and click **Install**.
5. Take note of the namespace used as it will be required in a later step.

##### Configuring the Teams Receiver

1. To configure the Teams receiver, update its ConfigMap. The following example is a minimal Teams receiver configuration:

```
[Microsoft Teams]
connector: https://your-teams-webhook-url
```

2. After you update the configuration, follow the instructions in Creating Receivers in the Rancher UI to add the receiver. Use the example below to form your URL. Make sure to replace `<namespace>` with the namespace of the `rancher-alerting-drivers` app:

```
 url: http://rancher-alerting-drivers-
prom2teams.<namespace>.svc:8089/v2/connector
```

#### SMS

##### Enabling the SMS Receiver for Rancher Managed Clusters

The SMS receiver is not a native receiver and must be enabled before it can be used. You can enable the SMS receiver for a Rancher managed cluster by going to the Apps page and installing the rancher-alerting-drivers app with the SMS option selected.

1. In the upper left corner, click ☰ **> Cluster Management**.

2. On the **Clusters** page, go to the cluster where you want to install `rancher-alerting-drivers` and click **Explore**.
3. In the left navigation bar, click
4. Click the **Alerting Drivers** app.
5. Click the **Helm Deploy Options** tab
6. Select the **SMS** option and click **Install**.
7. Take note of the namespace used as it will be required in a later step.

#### Configuring the SMS Receiver

The SMS receiver can be configured by updating its ConfigMap. For example, the following is a minimal SMS receiver configuration.

```
providers:
  telegram:
    token: 'your-token-from-telegram'

receivers:
- name: 'telegram-receiver-1'
  provider: 'telegram'
  to:
    - '123456789'
```

When configuration is complete, add the receiver using the steps in this section.

Use the example below as the name and URL, where:

- the name assigned to the receiver, e.g. `telegram-receiver-1`, must match the name in the `receivers.name` field in the ConfigMap, e.g. `telegram-receiver-1`
- `ns-1` in the URL is replaced with the namespace where the `rancher-alerting-drivers` app is installed

```
name: telegram-receiver-1
url http://rancher-alerting-drivers-sachet.ns-1.svc:9876/alert
```

#### Telegram

##### Enabling the Telegram Receiver for Rancher Managed Clusters

The Telegram receiver is not a native receiver. You must enable it before it can be used. You can enable the Telegram receiver for a Rancher-managed cluster by going to the **Apps** page and installing the `rancher-alerting-drivers` app with the **Telegram** option selected:

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to install `rancher-alerting-drivers` and click **Explore**.

3. In the left navigation bar, click on **Apps**.
4. Click the **Alerting Drivers** app.
5. Click on **Install**.
6. In the page that opens next, make sure that **Enable SMS** checkbox is selected. Telegram notifications require you to enable SMS.
7. Take note of the namespace used as it will be required in a later step.

#### Test the Configuration by Configuring a PrometheusRule

To test your Telegram setup, create a **PrometheusRule** that continuously raises alerts.



*NOTE*

This rule is intended only to test if Telegram alerts work as expected. Do not leave it on after testing is completed.

1. In the left navigation menu, click **Monitoring**.
2. Click **Advanced**.
3. Click **PrometheusRules › Create**.
4. Select a namespace to place the rule in and name the rule appropriately.
5. Set the group name to `test`. Use this value later when you create a **Route** in the **AlertManagerConfig**.
6. Under **Alerting Rules** click **Add**.
7. Set an appropriate **Alert Name**.
8. To trigger the alert immediately and continuously, enter the following PromQL Expression: `vector(1)`.
9. Under **Labels**, click **Add Label**. Enter the key `test` and value `alert`. This key-value pair will also be used later.

#### Configure an AlertManagerConfig

Configure an **AlertManagerConfig** to contain the **Receiver** and **Route** configuration for the **PrometheusRule** created above:

1. Click **Monitoring › Alerting**, and open **AlertManagerConfigs**.
2. Click **Create**

#### Create a Receiver in AlertManagerConfig

1. Choose a namespace from the dropdown and set an appropriate name.
2. Click **Create**.
3. Open the newly created **AlertManagerConfig** and click ⬔ **> Edit Config**.
4. Click **Add Receiver**.
5. Select **Webhook** from the list on the **Create Receiver in AlertmanagerConfig** page.
6. Name the webhook, and click **Add Webhook**.
7. In the **Select Webhook Type** dropdown, select **SMS**. This will automatically populate the **Target** field as `http://rancher-alerting-drivers-sachet.cattle-monitoring-`

`system.svc:9876/alert`. If you installed the **Alerting Drivers** in a namespace other than `cattle-monitoring-system`, the target URL will reflect that.

8. Click **Create**.

#### Create a Route in AlertManagerConfig

1. Click ⋮ **> Edit Config**.
2. Click **Route**.
3. In the dropdown, select the **Receiver** you just created.
4. In the **Labels to Group Alerts By** field, type `test`.
5. Under **Waiting and Intervals**, set **Group Wait** to `1s` and **Group Interval** to `10s`. This triggers frequent alerts. Change the values as appropriate.
6. Under **Matchers** click **Add Matcher**. Enter `test` in the **Name** field and `alert` in the **Value** field. From the **Match Type** dropdown, select `MatchEqual`.
7. Click **Save**.

#### Configuring the Telegram Receiver

You can configure the Telegram receiver by updating the `rancher-alerting-drivers-sachet` ConfigMap in the `cattle-monitoring-system` namespace. For example, the following is a minimal Telegram receiver configuration:

```
providers:
  telegram:
    token: <your-token-from-telegram>

receivers:
- name: 'cattle-monitoring-system/test-amc/prom2tel'
  provider: 'telegram'
  to:
    - '123456789'
```

To obtain a Telegram token, setup a Telegram bot. Refer to the official Telegram guide [https://core.telegram.org/bots/tutorial] for details. After you finish configuring the receiver, add it.

Name the receiver `<namespace>/<alertmanagerconfig-name>/<receiver-name>`. Enter `123456789` as a placeholder for the Telegram user ID to send the notifications to. To find your Telegram ID, check the Telegram userinfo bot [https://telegram.me/userinfobot].

You should now receive Telegram notifications to the user ID. If you don't receive notifications, please check if there are any errors reported in the Pod for the Deployment `rancher-alerting-drivers-sachet` under the `cattle-monitoring-system` namespace.

#### Configuring Multiple Receivers

By editing the forms in the Rancher UI, you can set up a Receiver resource with all the information Alertmanager needs to send alerts to your notification system.

It is also possible to send alerts to multiple notification systems. One way is to configure the Receiver using custom YAML, in which case you can add the configuration for multiple notification systems, as long as you are sure that both systems should receive the same messages.

You can also set up multiple receivers by using the `continue` option for a route, so that the alerts sent to a receiver continue being evaluated in the next level of the routing tree, which could contain another receiver.

#### Example Alertmanager Configs

##### Slack

To set up notifications via Slack, the following Alertmanager Config YAML can be placed into the `alertmanager.yaml` key of the Alertmanager Config Secret, where the `api_url` should be updated to use your Webhook URL from Slack:

```
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: 'slack-notifications'
receivers:
- name: 'slack-notifications'
  slack_configs:
  - send_resolved: true
    text: '{{ template "slack.rancher.text" . }}'
    api_url: <user-provided slack webhook url here>
templates:
- /etc/alertmanager/config/*.tmpl
```

##### PagerDuty

To set up notifications via PagerDuty, use the example below from the PagerDuty documentation [https://www.pagerduty.com/docs/guides/prometheus-integration-guide/] as a guideline. This example sets up a route that captures alerts for a database service and sends them to a receiver linked to a service that will directly notify the DBAs in PagerDuty, while all other alerts will be directed to a default receiver with a different PagerDuty integration key.

The following Alertmanager Config YAML can be placed into the `alertmanager.yaml` key of the Alertmanager Config Secret. The `service_key` should be updated to use your PagerDuty integration key and can be found as per the "Integrating with Global Event Routing" section of the PagerDuty documentation. For the full list of configuration options, refer to the Prometheus documentation [https://prometheus.io/docs/alerting/latest/configuration/#pagerduty_config].

```
route:
 group_by: [cluster]
 receiver: 'pagerduty-notifications'
 group_interval: 5m
```

```
 routes:
  - match:
      service: database
    receiver: 'database-notifcations'

receivers:
- name: 'pagerduty-notifications'
  pagerduty_configs:
  - service_key: 'primary-integration-key'

- name: 'database-notifcations'
  pagerduty_configs:
  - service_key: 'database-integration-key'
```

#### Example Route Config for Compliance Scan Alerts

While configuring the routes for `rancher-compliance` alerts, you can specify the matching using the key-value pair `job: rancher-compliance-scan`.

For example, the following example route configuration could be used with a Slack receiver named `test-compliance`:

```
spec:
  receiver: test-compliance
  group_by:
#   - string
  group_wait: 30s
  group_interval: 30s
  repeat_interval: 30s
  match:
    job: rancher-compliance-scan
#   key: string
  match_re:
    {}
#   key: string
```

For more information on enabling alerting for `rancher-compliance`, see this section.

#### Trusted CA for Notifiers

If you need to add a trusted CA to your notifier, follow the steps in this section.

### Route Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives

updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

The route configuration is the section of the Alertmanager custom resource that controls how the alerts fired by Prometheus are grouped and filtered before they reach the receiver.

When a Route is changed, the Prometheus Operator regenerates the Alertmanager custom resource to reflect the changes.

For more information about configuring routes, refer to the official Alertmanager documentation. [https://www.prometheus.io/docs/alerting/latest/configuration/#route]



This section assumes familiarity with how monitoring components work together. For more information, see this section.

#### Route Restrictions

Alertmanager proxies alerts for Prometheus based on its receivers and a routing tree that filters alerts to certain receivers based on labels.

Alerting drivers proxy alerts for Alertmanager to non-native receivers, such as Microsoft Teams and SMS.

In the Rancher UI for configuring routes and receivers, you can configure routing trees with one root and then a depth of one more level, for a tree with a depth of two. But if you use a `continue` route when configuring Alertmanager directly, you can make the tree deeper.

Each receiver is for one or more notification providers. So if you know that every alert for Slack should also go to PagerDuty, you can configure both in the same receiver.

#### Route Configuration

##### Note on Labels and Annotations

Labels should be used for identifying information that can affect the routing of notifications. Identifying information about the alert could consist of a container name, or the name of the team that should be notified.

Annotations should be used for information that does not affect who receives the alert, such as a runbook url or error message.

#### Receiver

The route needs to refer to a receiver that has already been configured.

#### Grouping

| Field | Default | Description |
|---|---|---|
| Group By | N/a | List of labels to group by. Labels must be unique. Special label "..." (aggregate by all possible labels), if provided, must be the only element in the list. Accepts a list of strings. See the upstream documentation [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#route] for details. |
| Group Wait | 30s | How long to wait to buffer alerts of the same group before sending initially. |
| Group Interval | 5m | How long to wait before sending an alert that has been added to a group of alerts for which an initial notification has already been sent. |
| Repeat Interval | 4h | How long to wait before re-sending a given alert that has already been sent. |

#### Matching

The **Match** field refers to a set of equality matchers used to identify which alerts to send to a given Route based on labels defined on that alert. When you add key-value pairs to the Rancher UI, they correspond to the YAML in this format:

```
match:
  [ <labelname>: <labelvalue>, ... ]
```

The **Match Regex** field refers to a set of regex-matchers used to identify which alerts to send to a given Route based on labels defined on that alert. When you add key-value pairs in the Rancher UI, they correspond to the YAML in this format:

```
match_re:
  [ <labelname>: <regex>, ... ]
```

### ServiceMonitor and PodMonitor Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

ServiceMonitors and PodMonitors are both pseudo-CRDs that map the scrape configuration of the Prometheus custom resource.

These configuration objects declaratively specify the endpoints that Prometheus will scrape metrics from.

ServiceMonitors are more commonly used than PodMonitors, and we recommend them for most use cases.



This section assumes familiarity with how monitoring components work together. For more information about Alertmanager, see this section.

#### ServiceMonitors

This pseudo-CRD maps to a section of the Prometheus custom resource configuration. It declaratively specifies how groups of Kubernetes services should be monitored.

When a ServiceMonitor is created, the Prometheus Operator updates the Prometheus scrape configuration to include the ServiceMonitor configuration. Then Prometheus begins scraping metrics from the endpoint defined in the ServiceMonitor.

Any Services in your cluster that match the labels located within the ServiceMonitor `selector` field will be monitored based on the `endpoints` specified on the ServiceMonitor. For more information on what fields can be specified, please look at the spec [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#servicemonitor] provided by Prometheus Operator.

For more information about how ServiceMonitors work, refer to the Prometheus Operator documentation. [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/user-guides/running-exporters.md]

#### PodMonitors

This pseudo-CRD maps to a section of the Prometheus custom resource configuration. It declaratively specifies how group of pods should be monitored.

When a PodMonitor is created, the Prometheus Operator updates the Prometheus scrape configuration to include the PodMonitor configuration. Then Prometheus begins scraping metrics from the endpoint defined in the PodMonitor.

Any Pods in your cluster that match the labels located within the PodMonitor `selector` field will be monitored based on the `podMetricsEndpoints` specified on the PodMonitor. For more information on what fields can be specified, please look at the spec [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#podmonitor] provided by Prometheus Operator.

### Advanced Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

#### Alertmanager

For information on configuring the Alertmanager custom resource, see this page.

#### Prometheus

For information on configuring the Prometheus custom resource, see this page.

#### PrometheusRules

For information on configuring the Prometheus custom resource, see this page.

#### Alertmanager Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

It is usually not necessary to directly edit the Alertmanager custom resource. For most use cases, you will only need to edit the Receivers and Routes to configure notifications.

When Receivers and Routes are updated, the monitoring application will automatically update the Alertmanager custom resource to be consistent with those changes.



This section assumes familiarity with how monitoring components work together. For more information about Alertmanager, see this section.

#### About the Alertmanager Custom Resource

By default, Rancher Monitoring deploys a single Alertmanager onto a cluster that uses a default Alertmanager Config Secret.

You may want to edit the Alertmanager custom resource if you would like to take advantage of advanced options that are not exposed in the Rancher UI forms, such as the ability to create a routing tree structure that is more than two levels deep.

It is also possible to create more than one Alertmanager in a cluster, which may be useful if you want to implement namespace-scoped monitoring. In this case, you should manage the Alertmanager custom resources using the same underlying Alertmanager Config Secret.

#### Deeply Nested Routes

While the Rancher UI only supports a routing tree that is two levels deep, you can configure more deeply nested routing structures by editing the Alertmanager YAML.

#### Multiple Alertmanager Replicas

As part of the chart deployment options, you can opt to increase the number of replicas of the Alertmanager deployed onto your cluster. The replicas can all be managed using the same underlying Alertmanager Config Secret.

This Secret should be updated or modified any time you want to:

- Add in new notifiers or receivers

- Change the alerts that should be sent to specific notifiers or receivers
- Change the group of alerts that are sent out

By default, you can either choose to supply an existing Alertmanager Config Secret (i.e. any Secret in the `cattle-monitoring-system` namespace) or allow Rancher Monitoring to deploy a default Alertmanager Config Secret onto your cluster.

By default, the Alertmanager Config Secret created by Rancher will never be modified or deleted on an upgrade or uninstall of the `rancher-monitoring` chart. This restriction prevents users from losing or overwriting their alerting configuration when executing operations on the chart.

For more information on what fields can be specified in the Alertmanager Config Secret, please look at the Prometheus Alertmanager docs. [https://prometheus.io/docs/alerting/latest/alertmanager/]

The full spec for the Alertmanager configuration file and what it takes in can be found here. [https://prometheus.io/docs/alerting/latest/configuration/#configuration-file]

#### Prometheus Configuration



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

It is usually not necessary to directly edit the Prometheus custom resource because the monitoring application automatically updates it based on changes to ServiceMonitors and PodMonitors.



This section assumes familiarity with how monitoring components work together. For more information, see this section.

#### About the Prometheus Custom Resource

The Prometheus CR defines a desired Prometheus deployment. The Prometheus Operator observes the Prometheus CR. When the CR changes, the Prometheus Operator creates `prometheus-rancher-monitoring-prometheus`, a Prometheus deployment based on the CR configuration.

The Prometheus CR specifies details such as rules and what Alertmanagers are connected to Prometheus. Rancher builds this CR for you.

Monitoring V2 only supports one Prometheus per cluster. However, you might want to edit the Prometheus CR if you want to limit monitoring to certain namespaces.

#### Configuring PrometheusRules



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from

`rancher-monitoring` to `rancher-monitoring-dashboards`.

A PrometheusRule defines a group of Prometheus alerting and/or recording rules.



This section assumes familiarity with how monitoring components work together. For more information, see this section.

#### Creating PrometheusRules in the Rancher UI



*Prerequisite:*

The monitoring application needs to be installed.

To create rule groups in the Rancher UI,

1. Go to the cluster where you want to create rule groups. Click **Monitoring** › **Advanced** and click **Prometheus Rules**.
2. Click **Create**.
3. Enter a **Group Name**.
4. Configure the rules. In Rancher's UI, we expect a rule group to contain either alert rules or recording rules, but not both. For help filling out the forms, refer to the configuration options below.
5. Click **Create**.

**Result:** Alerts can be configured to send notifications to the receiver(s).

#### About the PrometheusRule Custom Resource

When you define a Rule (which is declared within a RuleGroup in a PrometheusRule resource), the spec of the Rule itself [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#rule] contains labels that are used by Alertmanager to figure out which Route should receive this Alert. For example, an Alert with the label `team: front-end` will be sent to all Routes that match on that label.

Prometheus rule files are held in PrometheusRule custom resources. A PrometheusRule allows you to define one or more RuleGroups. Each RuleGroup consists of a set of Rule objects that can each represent either an alerting or a recording rule with the following fields:

- The name of the new alert or record
- A PromQL expression for the new alert or record
- Labels that should be attached to the alert or record that identify it (e.g. cluster name or severity)
- Annotations that encode any additional important pieces of information that need to be displayed on the notification for an alert (e.g. summary, description, message, runbook URL, etc.). This field is not required for recording rules.

For more information on what fields can be specified, please look at the Prometheus Operator spec. [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/api-reference/api.md#prometheusrulespec]

Use the label selector field `ruleSelector` in the Prometheus object to define the rule files that you want to be mounted into Prometheus.

For examples, refer to the Prometheus documentation on recording rules [https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/] and alerting rules. [https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/]

#### Configuration

#### Rule Group

| Field | Description |
|---|---|
| Group Name | The name of the group. Must be unique within a rules file. |
| Override Group Interval | Duration in seconds for how often rules in the group are evaluated. |

#### Alerting Rules

Alerting rules [https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/] allow you to define alert conditions based on PromQL (Prometheus Query Language) expressions and to send notifications about firing alerts to an external service.

| Field | Description |
|---|---|
| Alert Name | The name of the alert. Must be a valid label value. |
| Wait To Fire For | Duration in seconds. Alerts are considered firing once they have been returned for this long. Alerts which have not yet fired for long enough are considered pending. |
| PromQL Expression | The PromQL expression to evaluate. Prometheus will evaluate the current value of this PromQL expression on every evaluation cycle and all resultant time series will become pending/firing alerts. For more information, refer to the Prometheus documentation [https://prometheus.io/docs/prometheus/latest/querying/basics/] or our example PromQL expressions. |
| Labels | Labels to add or overwrite for each alert. |
| Severity | When enabled, labels are attached to the alert or record that identify it by the severity level. |
| Severity Label Value | Critical, warning, or none |
| Annotations | Annotations are a set of informational labels that can be used to store longer additional information, such as alert descriptions or runbook links. A runbook [https://en.wikipedia.org/wiki/Runbook] is a set of documentation about how to handle alerts. The annotation values can be templated. [https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/#templating] |

#### Recording Rules

Recording rules [https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/#recording-rules] allow you to precompute frequently needed or computationally expensive PromQL (Prometheus Query Language) expressions and save their result as a new set of time series.

| Field | Description |
|---|---|
| Time Series Name | The name of the time series to output to. Must be a valid metric name. |

| Field | Description |
|---|---|
| PromQL Expression | The PromQL expression to evaluate. Prometheus will evaluate the current value of this PromQL expression on every evaluation cycle and the result will be recorded as a new set of time series with the metric name as given by 'record'. For more information about expressions, refer to the Prometheus documentation [https://prometheus.io/docs/prometheus/latest/querying/basics/] or our example PromQL expressions. |
| Labels | Labels to add or overwrite before storing the result. |

## Monitoring Configuration Guides



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

This page captures some of the most important options for configuring Monitoring v2 in the Rancher UI.

For information on configuring custom scrape targets and rules for Prometheus, please refer to the upstream documentation for the Prometheus Operator. [https://github.com/prometheus-operator/prometheus-operator] Some of the most important custom resources are explained in the Prometheus Operator design documentation. [https://github.com/prometheus-operator/prometheus-operator/blob/main/Documentation/getting-started/design.md] The Prometheus Operator documentation can help also you set up RBAC, Thanos, or custom configuration.

### Setting Resource Limits and Requests

The resource requests and limits for the monitoring application can be configured when installing `rancher-monitoring`. For more information about the default limits, see this page.



On an idle cluster, Monitoring may have high CPU usage. To improve performance, turn off the Prometheus adapter.

### Prometheus Configuration

It is usually not necessary to directly edit the Prometheus custom resource.

Instead, to configure Prometheus to scrape custom metrics, you will only need to create a new ServiceMonitor or PodMonitor to configure Prometheus to scrape additional metrics.

#### ServiceMonitor and PodMonitor Configuration

For details, see this page.

#### Advanced Prometheus Configuration

For more information about directly editing the Prometheus custom resource, which may be helpful in advanced use cases, see this page.

### Alertmanager Configuration

The Alertmanager custom resource usually doesn't need to be edited directly. For most common use cases, you can manage alerts by updating Routes and Receivers.

Routes and receivers are part of the configuration of the alertmanager custom resource. In the Rancher UI, Routes and Receivers are not true custom resources, but pseudo-custom resources that the Prometheus Operator uses to synchronize your configuration with the Alertmanager custom resource. When routes and receivers are updated, the monitoring application will automatically update Alertmanager to reflect those changes.

For some advanced use cases, you may want to configure alertmanager directly. For more information, refer to this page.

#### Receivers

Receivers are used to set up notifications. For details on how to configure receivers, see this page.

#### Routes

Routes filter notifications before they reach receivers. Each route needs to refer to a receiver that has already been configured. For details on how to configure routes, see this page.

#### Advanced

For more information about directly editing the Alertmanager custom resource, which may be helpful in advanced use cases, see this page.

### Debugging High Memory Usage



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Every time series in Prometheus is uniquely identified by its metric name [https://prometheus.io/docs/practices/naming/#metric-names] and optional key-value pairs called labels. [https://prometheus.io/docs/practices/naming/#labels]

The labels allow the ability to filter and aggregate the time series data, but they also multiply the amount of data that Prometheus collects.

Each time series has a defined set of labels, and Prometheus generates a new time series for all unique combinations of labels. If a metric has two labels attached, two time series are generated for that metric. Changing any label value, including adding or removing a label, will create a new time series.

Prometheus is optimized to store data that is index-based on series. It is designed for a relatively consistent number of time series and a relatively large number of samples that need to be collected from the exporters over time.

Inversely, Prometheus is not optimized to accommodate a rapidly changing number of time series. For that

reason, large bursts of memory usage can occur when monitoring is installed on clusters where many resources are being created and destroyed, especially on multi-tenant clusters.

#### Reducing Memory Bursts

To reduce memory consumption, Prometheus can be configured to store fewer time series, by scraping fewer metrics or by attaching fewer labels to the time series. To see which series use the most memory, you can check the TSDB (time series database) status page in the Prometheus UI.

Distributed Prometheus solutions such as Thanos [https://thanos.io/] and Cortex [https://cortexmetrics.io/] use an alternate architecture in which multiple small Prometheus instances are deployed. In the case of Thanos, the metrics from each Prometheus are aggregated into the common Thanos deployment, and then those metrics are exported to a persistent store, such as S3. This more robust architecture avoids burdening any single Prometheus instance with too many time series, while also preserving the ability to query metrics on a global level.

### Persistent Grafana Dashboards



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

To allow the Grafana dashboard to persist after the Grafana instance restarts, add the dashboard configuration JSON into a ConfigMap. ConfigMaps also allow the dashboards to be deployed with a GitOps or CD based approach. This allows the dashboard to be put under version control.

#### Creating a Persistent Grafana Dashboard



*Prerequisites:*

- The monitoring application needs to be installed.
- To create the persistent dashboard, you must have at least the **Manage Config Maps** Rancher RBAC permissions assigned to you in the project or namespace that contains the Grafana Dashboards. This correlates to the `monitoring-dashboard-edit` or `monitoring-dashboard-admin` Kubernetes native RBAC Roles exposed by the Monitoring chart.
- To see the links to the external monitoring UIs, including Grafana dashboards, you will need at least a project-member role.

##### 1. Get the JSON model of the dashboard that you want to persist

To create a persistent dashboard, you will need to get the JSON model of the dashboard you want to persist. You can use a premade dashboard or build your own.

To use a premade dashboard, go to https://grafana.com/grafana/dashboards, open up its detail page, and click on the **Download JSON** button to get the JSON model for the next step.

To use your own dashboard:

1. Click on the link to open Grafana. On the cluster detail page, click **Monitoring**.
2. Log in to Grafana. Note: The default Admin username and password for the Grafana instance is `admin` and `prom-operator`. Alternative credentials can also be supplied on deploying or upgrading the chart.



Regardless of who has the password, in order to access the Grafana instance, you still need at least the **Manage Services** or **View Monitoring** permissions in the project that Rancher Monitoring is deployed into. Alternative credentials can also be supplied on deploying or upgrading the chart.

3. Create a dashboard using Grafana's UI. Once complete, go to the dashboard's settings by clicking on the gear icon in the top navigation menu. In the left navigation menu, click **JSON Model**.
4. Copy the JSON data structure that appears.

#### 2. Create a ConfigMap using the Grafana JSON model

Create a ConfigMap in the namespace that contains your Grafana Dashboards (e.g. `cattle-dashboards` by default).

The ConfigMap should look like this:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    grafana_dashboard: "1"
  name: <dashboard-name>
  namespace: cattle-dashboards # Change if using a non-default namespace
data:
  <dashboard-name>.json: |-
    <copied-json>
```

By default, Grafana is configured to watch all ConfigMaps with the `grafana_dashboard` label within the `cattle-dashboards` namespace.

To specify that you would like Grafana to watch for ConfigMaps across all namespaces, refer to this section.

To create the ConfigMap through the Rancher UI, first make sure that you are currently logged in to the Grafana UI, to ensure that dashboards import without encountering permissions issues. Then, return to the Rancher UI and perform the following steps:

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the visualizations and click **Explore**.
3. Click menu:More Resources[Core > ConfigMaps].
4. Click **Create**.
5. On the **Data** tab, set up the key-value pairs similar to the example above. When entering the value for `<dashboard-name>.json`, click **Read from File** to upload the JSON data model as the value.
6. On the **Labels & Annotations** tab, click **Add Label** and enter `grafana_dashboard` as the key, and `1` as

the value.

7. Click **Create**.

**Result:** After the ConfigMap is created, it should show up on the Grafana UI and be persisted even if the Grafana pod is restarted.



The actual key-value pair may differ if you have modified the Helm chart to watch a different dashboard label and value.

Dashboards that are persisted using ConfigMaps cannot be deleted or edited from the Grafana UI.

If you attempt to delete the dashboard in the Grafana UI, you will see the error message "Dashboard cannot be deleted because it was provisioned." To delete the dashboard, you will need to delete the ConfigMap.

#### Configuring Namespaces for the Grafana Dashboard ConfigMap

To specify that you would like Grafana to watch for ConfigMaps across all namespaces, set this value in the `rancher-monitoring` Helm chart:

```
grafana.sidecar.dashboards.searchNamespace=ALL
```

Note that the RBAC roles exposed by the Monitoring chart to add Grafana Dashboards are still restricted to giving permissions for users to add dashboards in the namespace defined in `grafana.dashboards.namespace`, which defaults to `cattle-dashboards`.

### Customizing Grafana Dashboards



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

In this section, you'll learn how to customize the Grafana dashboard to show metrics that apply to a certain container.

#### Prerequisites

Before you can customize a Grafana dashboard, the `rancher-monitoring` application must be installed.

To see the links to the external monitoring UIs, including Grafana dashboards, you will need at least a project-member role.

#### Signing in to Grafana

1. In the Rancher UI, go to the cluster that has the dashboard you want to customize.
2. In the left navigation menu, click **Monitoring.**
3. Click **Grafana.** The Grafana dashboard should open in a new tab.

4. Go to the log in icon in the lower left corner and click **Sign In.**
5. Log in to Grafana. The default Admin username and password for the Grafana instance is `admin` and `prom-operator`. (Regardless of who has the password, cluster administrator permission in Rancher is still required access the Grafana instance.) Alternative credentials can also be supplied on deploying or upgrading the chart.

#### Getting the PromQL Query Powering a Grafana Panel

For any panel, you can click the title and click **Explore** to get the PromQL queries powering the graphic.

For this example, we would like to get the CPU usage for the Alertmanager container, so we click **CPU Utilization › Inspect.**

The **Data** tab shows the underlying data as a time series, with the time in first column and the PromQL query result in the second column. Copy the PromQL query.

```
(1 - (avg(irate({__name__=
~"node_cpu_seconds_total|windows_cpu_time_total",mode="idle"}[5m])))) *
100
```

You can then modify the query in the Grafana panel or create a new Grafana panel using the query.

See also:

- Grafana docs on editing a panel [https://grafana.com/docs/grafana/latest/panels-visualizations/configure-panel-options/#edit-a-panel]
- Grafana docs on adding a panel to a dashboard [https://grafana.com/docs/grafana/latest/panels-visualizations/panel-editor-overview]

## Prometheus Federator



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Prometheus Federator, also referred to as Project Monitoring v2, deploys a Helm Project Operator (based on the rancher/helm-project-operator [https://github.com/rancher/helm-project-operator]), an operator that manages deploying Helm charts each containing a Project Monitoring Stack, where each stack contains:

- Prometheus [https://prometheus.io/] (managed externally by Prometheus Operator [https://github.com/prometheus-operator/prometheus-operator])
- Alertmanager [https://prometheus.io/docs/alerting/latest/alertmanager/] (managed externally by Prometheus Operator [https://github.com/prometheus-operator/prometheus-operator])
- Grafana [https://github.com/helm/charts/tree/master/stable/grafana] (deployed via an embedded Helm chart)
- Default PrometheusRules and Grafana dashboards based on the collection of community-curated

resources from kube-prometheus [https://github.com/prometheus-operator/kube-prometheus/]

- Default ServiceMonitors that watch the deployed resources



Prometheus Federator is designed to be deployed alongside an existing Prometheus Operator deployment in a cluster that has already installed the Prometheus Operator CRDs.

**How does the operator work?**

1. On deploying this chart, users can create ProjectHelmCharts CRs with `spec.helmApiVersion` set to `monitoring.cattle.io/vlalphal` (also known as "Project Monitors" in the Rancher UI) in a **Project Registration Namespace (`cattle-project-<id>`)**.
2. On seeing each ProjectHelmChartCR, the operator will automatically deploy a Project Prometheus stack on the Project Owner's behalf in the **Project Release Namespace (`cattle-project-<id>-monitoring`)** based on a HelmChart CR and a HelmRelease CR automatically created by the ProjectHelmChart controller in the **Operator / System Namespace**.
3. RBAC will automatically be assigned in the Project Release Namespace to allow users to view the Prometheus, Alertmanager, and Grafana UIs of the Project Monitoring Stack deployed; this will be based on RBAC defined on the Project Registration Namespace against the default Kubernetes user-facing roles [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#user-facing-roles]. For more information, see the section on configuring RBAC.

**What is a Project?**

In Prometheus Federator, a Project is a group of namespaces that can be identified by a `metav1.LabelSelector`. By default, the label used to identify projects is `field.cattle.io/projectId`, the label used to identify namespaces that are contained within a given Rancher Project.

**Configuring the Helm release created by a ProjectHelmChart**

The `spec.values` of this ProjectHelmChart's resources will correspond to the `values.yaml` override to be supplied to the underlying Helm chart deployed by the operator on the user's behalf; to see the underlying chart's `values.yaml` spec, either:

- View the chart's definition located at `rancher/prometheus-federator` under `charts/rancher-project-monitoring` [https://github.com/rancher/prometheus-federator/blob/main/charts/rancher-project-monitoring] (where the chart version will be tied to the version of this operator).
- Look for the ConfigMap named `monitoring.cattle.io.vlalphal` that is automatically created in each Project Registration Namespace, which will contain both the `values.yaml` and `questions.yaml` that was used to configure the chart (which was embedded directly into the `prometheus-federator` binary).

**Namespaces**

As a Project Operator based on rancher/helm-project-operator [https://github.com/rancher/helm-project-operator], Prometheus Federator has three different classifications of namespaces that the operator looks out for:

1. **Operator / System Namespace**: The namespace that the operator is deployed into (e.g., `cattle-`

`monitoring-system`). This namespace will contain all HelmCharts and HelmReleases for all ProjectHelmCharts watched by this operator. **Only Cluster Admins should have access to this namespace.**

2. **Project Registration Namespace (`cattle-project-<id>`)**: The set of namespaces that the operator watches for ProjectHelmCharts within. The RoleBindings and ClusterRoleBindings that apply to this namespace will also be the source of truth for the auto-assigned RBAC created in the Project Release Namespace. For details, refer to the RBAC page. **Project Owners (admin), Project Members (edit), and Read-Only Members (view) should have access to this namespace.**



- Project Registration Namespaces will be auto-generated by the operator and imported into the Project it is tied to if `.Values.global.cattle.projectLabel` is provided, which is set to `field.cattle.io/projectId` by default. This indicates that a Project Registration Namespace should be created by the operator if at least one namespace is observed with that label. The operator will not let these namespaces be deleted unless either all namespaces with that label are gone (e.g., this is the last namespace in that project, in which case the namespace will be marked with the label `"helm.cattle.io/helm-project-operator-orphaned": "true"`, which signals that it can be deleted), or it is no longer watching that project because the project ID was provided under `.Values.helmProjectOperator.otherSystemProjectLabelValues`, which serves as a denylist for Projects. These namespaces will also never be auto-deleted to avoid destroying user data; it is recommended that users clean up these namespaces manually if desired on creating or deleting a project.
- If `.Values.global.cattle.projectLabel` is not provided, the Operator / System Namespace will also be the Project Registration Namespace.

3. **Project Release Namespace (`cattle-project-<id>-monitoring`):** The set of namespaces that the operator deploys Project Monitoring Stacks within on behalf of a ProjectHelmChart; the operator will also automatically assign RBAC to Roles created in this namespace by the Project Monitoring Stack based on bindings found in the Project Registration Namespace. **Only Cluster Admins should have access to this namespace; Project Owners (admin), Project Members (edit), and Read-Only Members (view) will be assigned limited access to this namespace by the deployed Helm Chart and Prometheus Federator.**



- Project Release Namespaces are automatically deployed and imported into the project whose ID is specified under `.Values.helmProjectOperator.projectReleaseNamespaces.labelValue`, which defaults to the value of `.Values.global.cattle.systemProjectId` if not specified, whenever a ProjectHelmChart is specified in a Project Registration Namespace.
- Project Release Namespaces follow the same orphaning conventions as Project Registration Namespaces (see note above).
- If `.Values.projectReleaseNamespaces.enabled` is false, the Project Release Namespace will be the same as the Project Registration Namespace.

#### Helm Resources (HelmChart, HelmRelease)

On deploying a ProjectHelmChart, the Prometheus Federator will automatically create and manage two child

custom resources that manage the underlying Helm resources in turn:

- A HelmChart CR (managed via an embedded k3s-io/helm-contoller [https://github.com/k3s-io/helm-controller] in the operator): This custom resource automatically creates a Job in the same namespace that triggers a `helm install`, `helm upgrade`, or `helm uninstall` depending on the change applied to the HelmChart CR. This CR is automatically updated on changes to the ProjectHelmChart (e.g., modifying the values.yaml) or changes to the underlying Project definition (e.g., adding or removing namespaces from a project).



If a ProjectHelmChart is not deploying or updating the underlying Project Monitoring Stack for some reason, the Job created by this resource in the Operator / System namespace should be the first place you check to see if there's something wrong with the Helm operation. However, this is generally only accessible by a **Cluster Admin.**

- A HelmRelease CR (managed via an embedded rancher/helm-locker [https://github.com/rancher/helm-locker] in the operator): This custom resource automatically locks a deployed Helm release in place and automatically overwrites updates to underlying resources unless the change happens via a Helm operation (`helm install`, `helm upgrade`, or `helm uninstall` performed by the HelmChart CR).



HelmRelease CRs emit Kubernetes Events that detect when an underlying Helm release is being modified and locks it back to place. To view these events, you can use `kubectl describe helmrelease <helm-release-name> -n <operator/system-namespace>`; you can also view the logs on this operator to see when changes are detected and which resources modifications were attempted on.

Both of these resources are created for all Helm charts in the Operator / System namespaces to avoid escalation of privileges to underprivileged users.

#### Advanced Helm Project Operator Configuration

For more information on advanced configurations, refer to this page [https://github.com/rancher/prometheus-federator/blob/main/charts/prometheus-federator/README.md#advanced-helm-project-operator-configuration].

### Prometheus Federator on the Local Cluster

Prometheus Federator is a resource intensive application. Installing it to the local cluster is possible, but **not recommended**.

### Role-Based Access Control



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

This section describes the expectations for Role-Based Access Control (RBAC) for Prometheus Federator.

As described in the section on namespaces, Prometheus Federator expects that Project Owners, Project Members, and other users in the cluster with Project-level permissions (e.g. permissions in a certain set of

namespaces identified by a single label selector) have minimal permissions in any namespaces except the Project Registration Namespace (which is imported into the project by default) and those that already comprise their projects. Therefore, in order to allow Project Owners to assign specific chart permissions to other users in their Project namespaces, the Helm Project Operator will automatically watch the following bindings:

- ClusterRoleBindings
- RoleBindings in the Project Release Namespace

On observing a change to one of those types of bindings, the Helm Project Operator will check whether the `roleRef` that the binding points to matches a ClusterRole with the name provided under:

- `helmProjectOperator.releaseRoleBindings.clusterRoleRefs.admin`
- `helmProjectOperator.releaseRoleBindings.clusterRoleRefs.edit`
- `helmProjectOperator.releaseRoleBindings.clusterRoleRefs.view`

By default, these roleRefs will correspond to `admin`, `edit`, and `view` respectively, which are the default Kubernetes user-facing roles [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#user-facing-roles].



For Rancher RBAC users, these default Kubernetes user-facing roles [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#user-facing-roles] directly correlate to the `Project Owner`, `Project Member`, and `Read-Only` default Project Role Templates.

If the `roleRef` matches, the Helm Project Operator will filter the `subjects` of the binding for all Users and Groups and use that to automatically construct a RoleBinding for each Role in the Project Release Namespace with the same name as the role and the following labels:

- `helm.cattle.io/project-helm-chart-role: {{ .Release.Name }}`
- `helm.cattle.io/project-helm-chart-role-aggregate-from: <admin|edit|view>`

By default, `rancher-project-monitoring`, the underlying chart deployed by Prometheus Federator, creates three default Roles per Project Release Namespace that provide `admin`, `edit`, and `view` users to permissions to view the Prometheus, Alertmanager, and Grafana UIs of the Project Monitoring Stack to provide least privilege. However, if a Cluster Admin would like to assign additional permissions to certain users, they can either directly assign RoleBindings in the Project Release Namespace to certain users or create Roles with the above two labels on them to allow Project Owners to control assigning those RBAC roles to users in their Project Registration namespaces.

### Enable Prometheus Federator



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

#### Requirements

By default, Prometheus Federator is configured and intended to be deployed alongside rancher-monitoring,

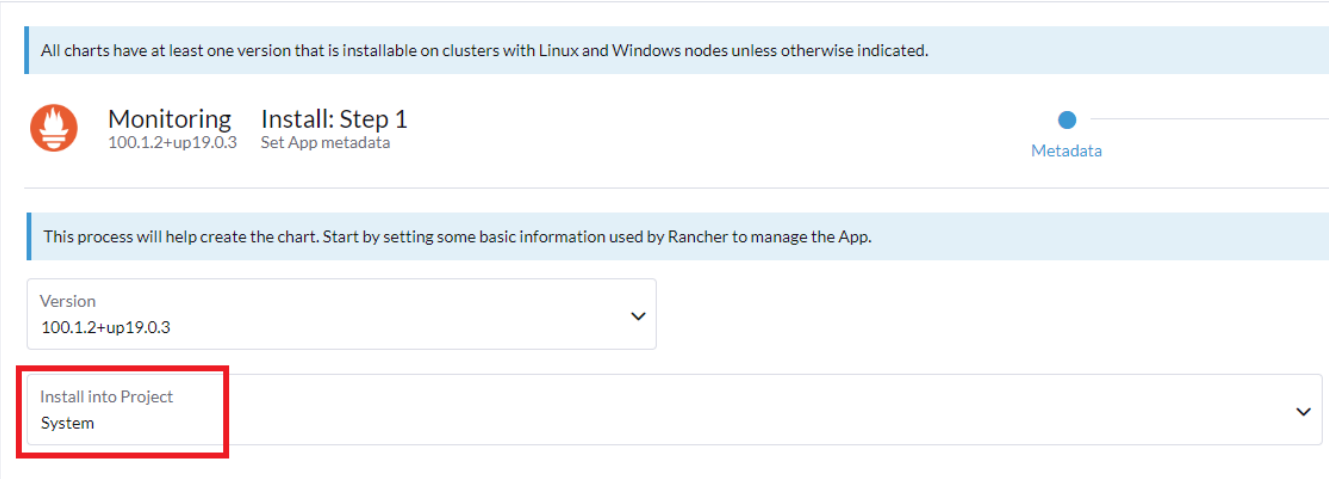
which deploys Prometheus Operator alongside a Cluster Prometheus that each Project Monitoring Stack is configured to federate namespace-scoped metrics from by default.

For instructions on installing rancher-monitoring, refer to this page.

The default configuration should already be compatible with your rancher-monitoring stack. However, to optimize the security and usability of Prometheus Federator in your cluster, we recommend making these additional configurations to rancher-monitoring:

- Ensure the cattle-monitoring-system namespace is placed into the System Project.
- Configure rancher-monitoring to only watch for resources created by the Helm chart itself.
- Increase the CPU / memory limits of the Cluster Prometheus.

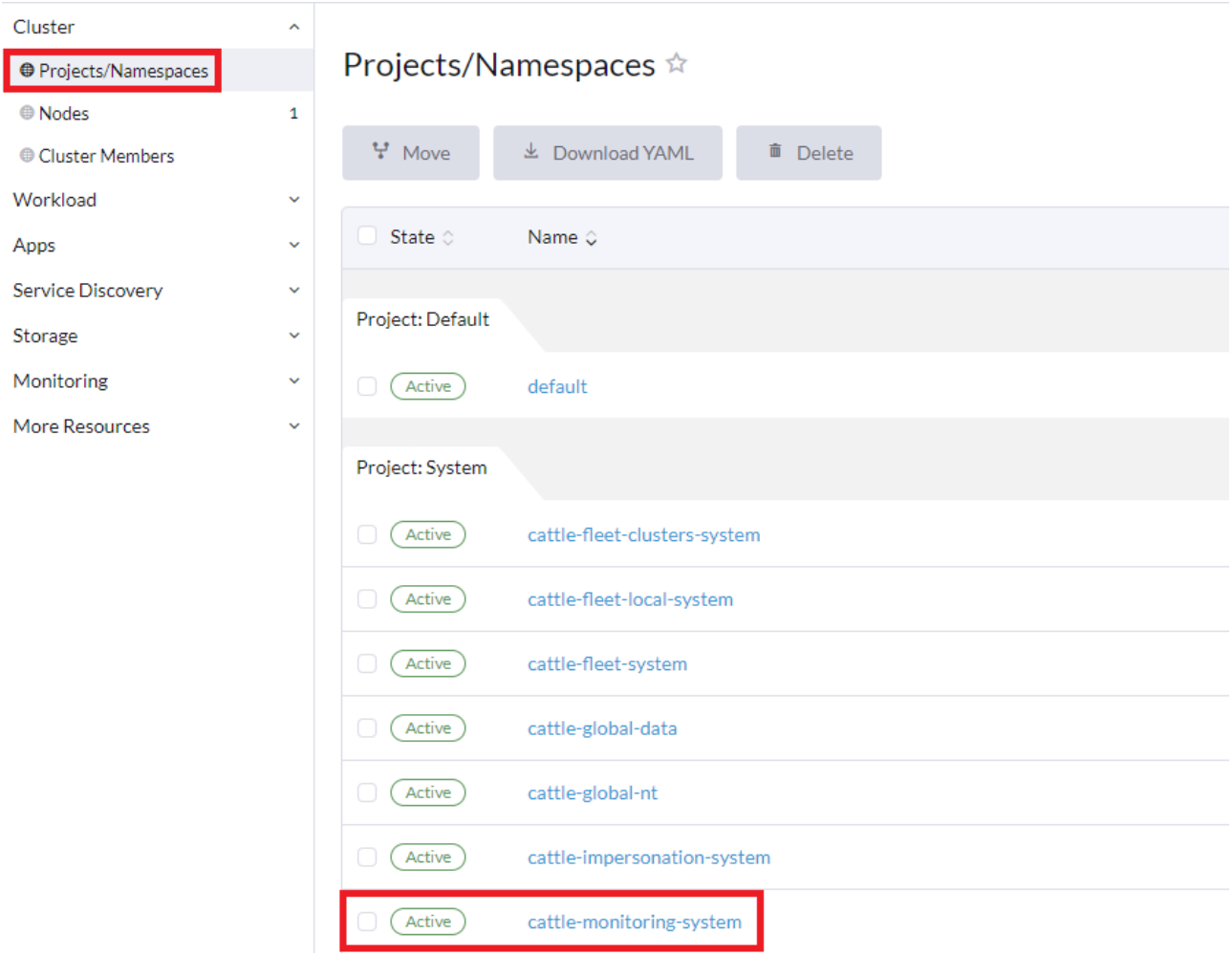
**Ensure the cattle-monitoring-system namespace is placed into the System Project (or a similarly locked down Project that has access to other Projects in the cluster)**




Prometheus Operator's security model expects that the namespace it is deployed into (e.g., `cattle-monitoring-system`) has limited access for anyone except Cluster Admins to avoid privilege escalation via execing into Pods (such as the Jobs executing Helm operations). In addition, deploying Prometheus Federator and all Project Prometheus stacks into the System Project ensures that each Project Prometheus is able to reach out to scrape workloads across all Projects, even if Network Policies are defined via Project Network Isolation. It also provides limited access for Project Owners, Project Members, and other users so that they're unable to access data that they shouldn't have access to (i.e., being allowed to exec into pods, set up the ability to scrape namespaces outside of a given Project, etc.).

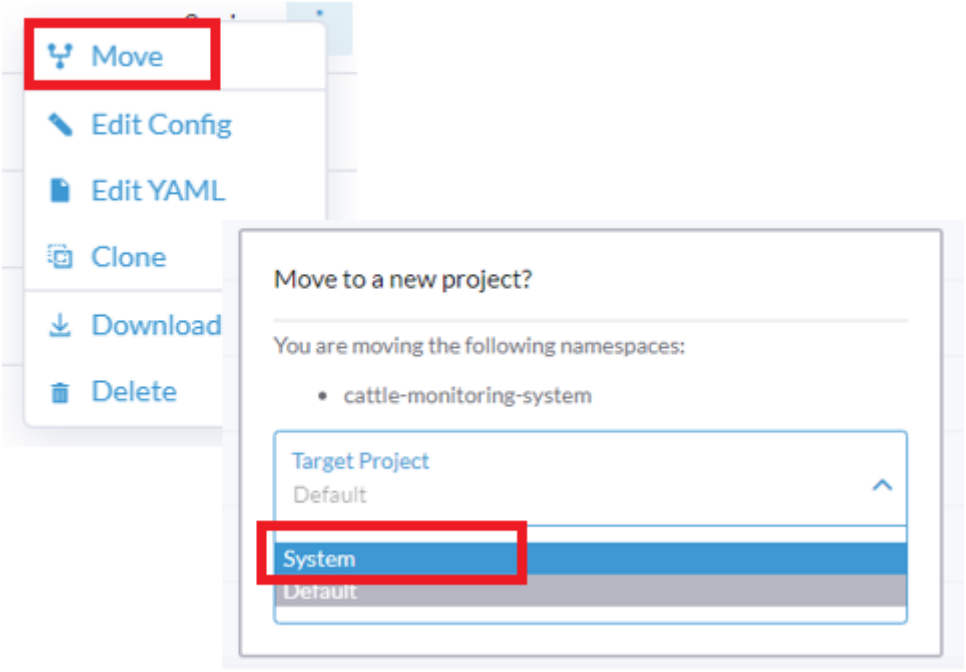
1. Open the `System` project to check your namespaces:

   Click **Cluster › Projects/Namespaces** in the Rancher UI. This will display all of the namespaces in the `System` project:




2. If you have an existing Monitoring V2 installation within the `cattle-monitoring-system` namespace, but that namespace is not in the `System` project, you may move the `cattle-monitoring-system` namespace into the `System` project or into another project of limited access. To do so, you may either:

   - Drag and drop the namespace into the `System` project or
   - Select ⋮ to the right of the namespace, click **Move**, then choose `System` from the **Target Project** dropdown




**Configure rancher-monitoring to only watch for resources created by the Helm chart itself**

Since each Project Monitoring Stack will watch the other namespaces and collect additional custom workload metrics or dashboards already, it's recommended to configure the following settings on all selectors to ensure that the Cluster Prometheus Stack only monitors resources created by the Helm Chart itself:

```
matchLabels:
  release: "rancher-monitoring"
```

The following selector fields are recommended to have this value:

- `.Values.alertmanager.alertmanagerSpec.alertmanagerConfigSelector`
- `.Values.prometheus.prometheusSpec.serviceMonitorSelector`
- `.Values.prometheus.prometheusSpec.podMonitorSelector`
- `.Values.prometheus.prometheusSpec.ruleSelector`
- `.Values.prometheus.prometheusSpec.probeSelector`

Once this setting is turned on, you can always create ServiceMonitors or PodMonitors that are picked up by the Cluster Prometheus by adding the label `release: "rancher-monitoring"` to them, in which case they will be ignored by Project Monitoring Stacks automatically by default, even if the namespace in which those ServiceMonitors or PodMonitors reside in are not system namespaces.



If you don't want to allow users to be able to create ServiceMonitors and PodMonitors that aggregate into the Cluster Prometheus in Project namespaces, you can additionally set the namespaceSelectors on the chart to only target system namespaces (which must contain `cattle-monitoring-system` and `cattle-dashboards`, where resources are deployed into by default by rancher-monitoring; you will also need to monitor the `default` namespace to get apiserver metrics or create a custom ServiceMonitor to scrape apiserver metrics from the Service residing in the default namespace) to limit your Cluster

Prometheus from picking up other Prometheus Operator CRs. In that case, it would be recommended to turn `.Values.prometheus.prometheusSpec.ignoreNamespaceSelectors=true` to allow you to define ServiceMonitors that can monitor non-system namespaces from within a system namespace.

#### Increase the CPU / memory limits of the Cluster Prometheus

Depending on a cluster's setup, it's generally recommended to give a large amount of dedicated memory to the Cluster Prometheus to avoid restarts due to out-of-memory errors (OOMKilled) usually caused by churn created in the cluster that causes a large number of high cardinality metrics to be generated and ingested by Prometheus within one block of time. This is one of the reasons why the default Rancher Monitoring stack expects around 4GB of RAM to be able to operate in a normal-sized cluster. However, when introducing Project Monitoring Stacks that are all sending `/federate` requests to the same Cluster Prometheus and are reliant on the Cluster Prometheus being "up" to federate that system data on their namespaces, it's even more important that the Cluster Prometheus has an ample amount of CPU / memory assigned to it to prevent an outage that can cause data gaps across all Project Prometheis in the cluster.



There are no specific recommendations on how much memory the Cluster Prometheus should be configured with since it depends entirely on the user's setup (namely the likelihood of encountering a high churn rate and the scale of metrics that could be generated at that time); it generally varies per setup.

#### Install the Prometheus Federator Application

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you want to install Prometheus Federator and click **Explore**.
3. Click **Apps -> Charts**.
4. Click the **Prometheus Federator** chart.
5. Click **Install**.
6. On the **Metadata** page, click **Next**.
7. In the **Namespaces** > **Project Release Namespace Project ID** field, the `System Project` is used as the default but can be overridden with another project with similarly limited access. Project IDs can be found with the following command run in the local upstream cluster:

```
 kubectl get projects -A -o custom-
columns="NAMESPACE":.metadata.namespace,"ID":.metadata.name,"NAME":.spec
.displayName
```

1. Click **Install**.

**Result:** The Prometheus Federator app is deployed in the `cattle-monitoring-system` namespace.

#### Uninstall Prometheus Federator



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives

updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click **Apps**.
4. Click **Installed Apps**.
5. Go to the `cattle-monitoring-system` namespace and check the boxes for `rancher-monitoring-crd` and `rancher-monitoring`.
6. Click **Delete**.
7. Confirm **Delete**.

**Result:** `prometheus-federator` is uninstalled.

#### Setting up Prometheus Federator for a Workload



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

##### Display CPU and Memory Metrics for a Workload

Displaying CPU and memory metrics with Prometheus Federator is done the same way as with rancher-monitoring. For instructions, refer here.

##### Setting up Metrics Beyond CPU and Memory

Setting up metrics beyond CPU and memory with Prometheus Federator is done the same way as with rancher-monitoring. For instructions, refer here.

#### Customizing Grafana Dashboards



*Deprecation Notice*

Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Grafana dashboards are customized the same way whether it's for rancher-monitoring or for Prometheus Federator.

For instructions, refer to this page.

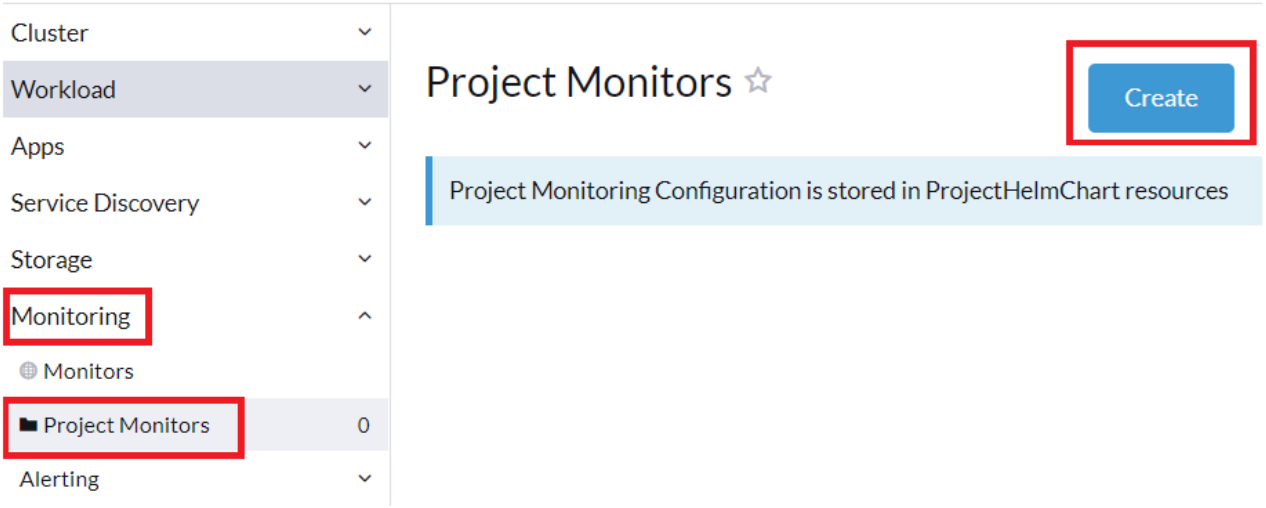
### Installing Project Monitors

*Deprecation Notice*

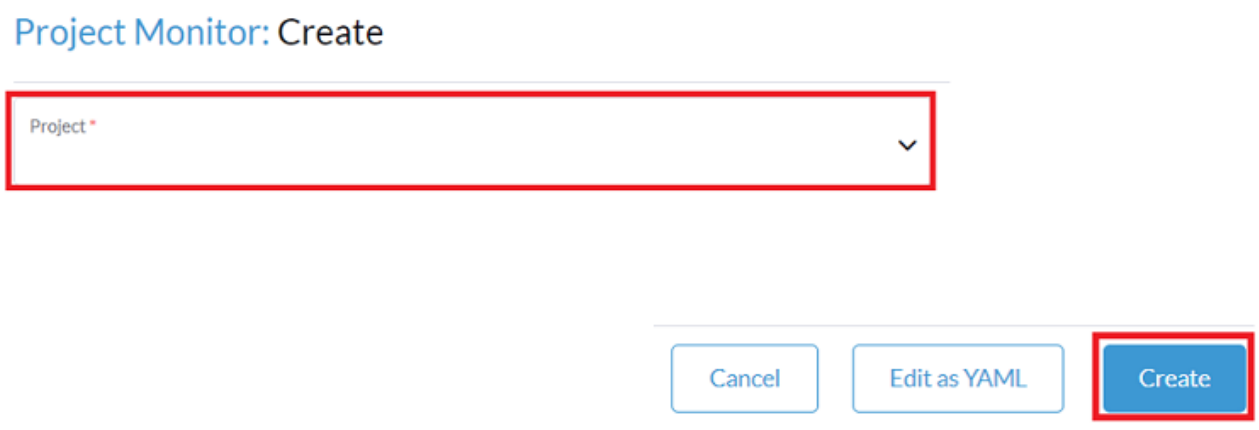
Rancher Monitoring v2 (`rancher-monitoring`) is deprecated and no longer receives updates. Rancher maintains the legacy chart only to support existing deployments. For new installations, ongoing updates, and support, use the decoupled `rancher-monitoring-dashboards` chart. For instructions on migrating, see Migrating from `rancher-monitoring` to `rancher-monitoring-dashboards`.

Install **Project Monitors** in each project where you want to enable project monitoring.

1. Click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to enable monitoring and click **Explore**.
3. Click **Monitoring** › **Project Monitors** on the left nav bar. Then click **Create** in the upper right.




4. Select your project from the drop-down menu, then click **Create** again.




## Istio

Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement

[https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

Istio [https://istio.io/] is an open-source tool that makes it easier for DevOps teams to observe, secure, control, and troubleshoot the traffic within a complex network of microservices.

As a network of microservices changes and grows, the interactions between them can become increasingly difficult to manage and understand. In such a situation, it is useful to have a service mesh as a separate infrastructure layer. Istio's service mesh lets you manipulate traffic between microservices without changing the microservices directly.

Our integration of Istio is designed so that a Rancher operator, such as an administrator or cluster owner, can deliver Istio to a team of developers. Then developers can use Istio to enforce security policies, troubleshoot problems, or manage traffic for green/blue deployments, canary deployments, or A/B testing.

This core service mesh provides features that include but are not limited to the following:

- **Traffic Management** such as ingress and egress routing, circuit breaking, mirroring.
- **Security** with resources to authenticate and authorize traffic and users, mTLS included.
- **Observability** of logs, metrics, and distributed traffic flows.

After setting up istio you can leverage Istio's control plane functionality through the Rancher UI, `kubectl`, or `istioctl`.

Istio needs to be set up by a `cluster-admin` before it can be used in a project.

## What's New in Rancher v2.5

The overall architecture of Istio has been simplified. A single component, Istiod, has been created by combining Pilot, Citadel, Galley and the sidecar injector. Node Agent functionality has also been merged into istio-agent.

Addons that were previously installed by Istio (cert-manager, Grafana, Jaeger, Kiali, Prometheus, Zipkin) will now need to be installed separately. Istio will support installation of integrations that are from the Istio Project and will maintain compatibility with those that are not.

A Prometheus integration will still be available through an installation of Rancher Monitoring, or by installing your own Prometheus operator. Rancher's Istio chart will also install Kiali by default to ensure you can get a full picture of your microservices out of the box.

Istio has migrated away from Helm as a way to install Istio and now provides installation through the istioctl binary or Istio Operator. To ensure the easiest interaction with Istio, Rancher's Istio will maintain a Helm chart that utilizes the istioctl binary to manage your Istio installation.

This Helm chart will be available via the Apps and Marketplace in the UI. A user that has access to the Rancher Chart's catalog will need to set up Istio before it can be used in the project.

## Tools Bundled with Istio

Our Istio [https://istio.io/] installer wraps the istioctl binary commands in a handy Helm chart, including an overlay file option to allow complex customization.

It also includes the following:

#### Kiali

Kiali [https://kiali.io/] is a comprehensive visualization aid used for graphing traffic flow throughout the service mesh. It allows you to see how they are connected, including the traffic rates and latencies between them.

You can check the health of the service mesh, or drill down to see the incoming and outgoing requests to a single component.



For Istio installations `103.1.0+upl.19.6` and later, Kiali uses a token value for its authentication strategy. The name of the Kiali service account in Rancher is `kiali`. Use this name if you are writing commands that require you to enter the name of the Kiali service account (for example, if you are trying to generate or retrieve a session token). For more information, refer to the Kiali token authentication FAQ [https://kiali.io/docs/faq/authentication/].

#### Jaeger

Our Istio installer includes a quick-start, all-in-one installation of Jaeger, [https://www.jaegertracing.io/] a tool used for tracing distributed systems.

Note that this is not a production-qualified deployment of Jaeger. This deployment uses an in-memory storage component, while a persistent storage component is recommended for production. For more information on which deployment strategy you may need, refer to the Jaeger documentation. [https://www.jaegertracing.io/docs/1.65/operator/#production-strategy]

### Prerequisites

Before enabling Istio, we recommend that you confirm that your Rancher worker nodes have enough CPU and memory to run all of the components of Istio.

If you are installing Istio on RKE2 cluster, some additional steps are required. For details, see this section.

### Setup Guide

Refer to the setup guide for instructions on how to set up Istio and use it in a project.

### Remove Istio

To remove Istio components from a cluster, namespace, or workload, refer to the section on uninstalling Istio.

### Accessing Visualizations

> By default, only cluster-admins have access to Kiali. For instructions on how to allow admin, edit or views roles to access them, see this section.



For Istio installations version `103.1.0+upl.19.6` and later, Kiali uses a token value for its authentication strategy. The name of the Kiali service account in Rancher is `kiali`. Use this name if you are writing commands that require you to enter the name of the Kiali service account (for example, if you are trying to generate or retrieve a session token). For more information, refer to the Kiali token authentication FAQ [https://kiali.io/docs/faq/

authentication/].

After Istio is set up in a cluster, Grafana, Prometheus, and Kiali are available in the Rancher UI.

To access the Grafana and Prometheus visualizations,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the visualizations and click **Explore**.
3. In the left navigation bar, click **Monitoring**.
4. Click **Grafana** or any of the other dashboards.

To access the Kiali visualization,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see Kiali and click **Explore**.
3. In the left navigation bar, click **Istio**.
4. Click **Kiali**. From here you can access the **Traffic Graph** tab or the **Traffic Metrics** tab to see network visualizations and metrics.

By default, all namespace will picked up by prometheus and make data available for Kiali graphs. Refer to selector/scrape config setup if you would like to use a different configuration for prometheus data scraping.

Your access to the visualizations depend on your role. Grafana and Prometheus are only available for `cluster-admin` roles. The Kiali UI is available only to `cluster-admin` by default, but `cluster-admin` can allow other roles to access them by editing the Istio values.yaml.

## Architecture

Istio installs a service mesh that uses Envoy [https://www.envoyproxy.io] sidecar proxies to intercept traffic to each workload. These sidecars intercept and manage service-to-service communication, allowing fine-grained observation and control over traffic within the cluster.

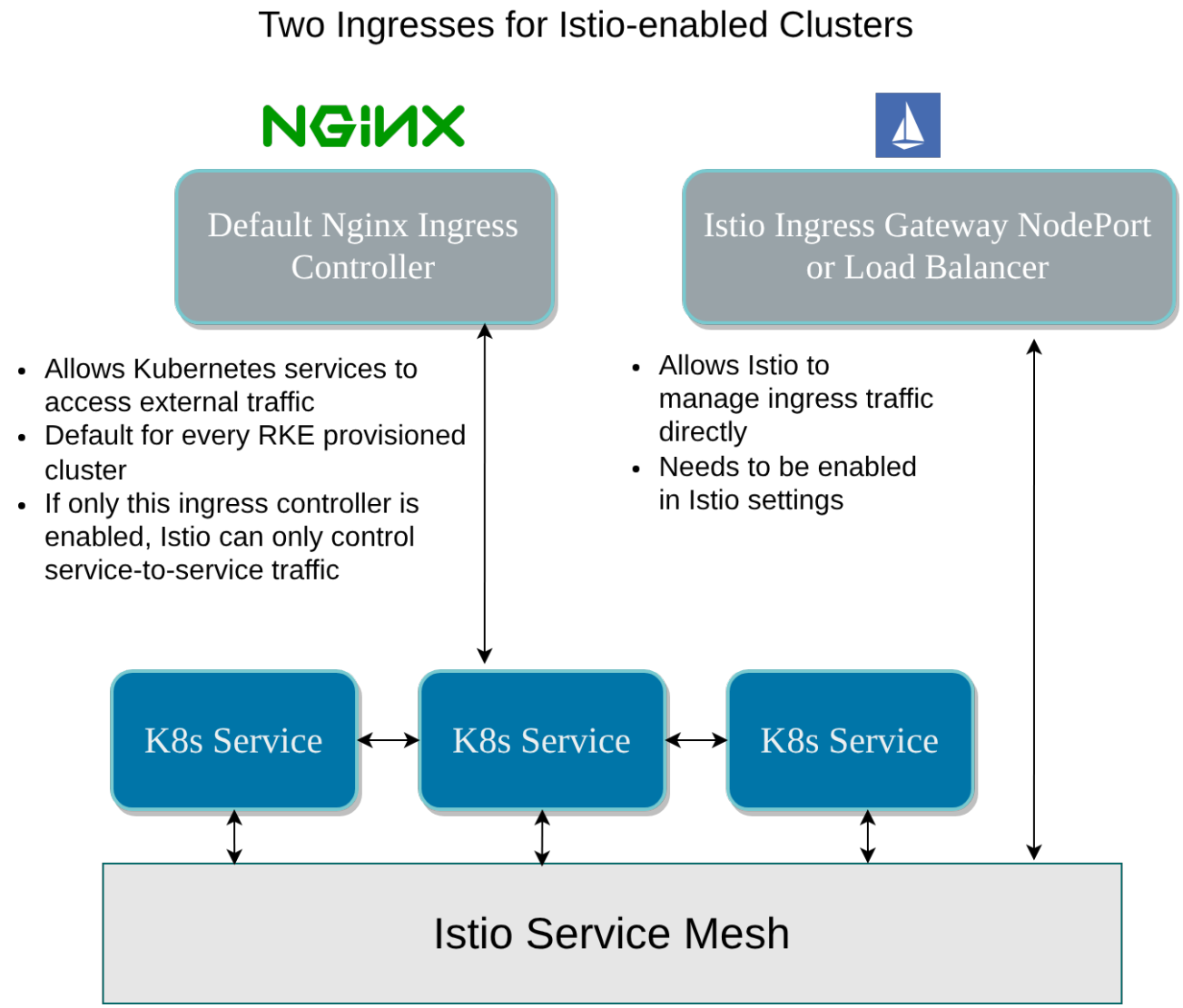
Only workloads that have the Istio sidecar injected can be tracked and controlled by Istio.

When a namespace has Istio enabled, new workloads deployed in the namespace will automatically have the Istio sidecar. You need to manually enable Istio in preexisting workloads.

For more information on the Istio sidecar, refer to the Istio sidecare-injection docs [https://istio.io/docs/setup/kubernetes/additional-setup/sidecar-injection/] and for more information on Istio's architecture, refer to the Istio Architecture docs [https://istio.io/latest/docs/ops/deployment/architecture/]

### Multiple Ingresses

By default, each Rancher-provisioned cluster has one NGINX ingress controller allowing traffic into the cluster. Istio also installs an ingress gateway by default into the `istio-system` namespace. The result is that your cluster will have two ingresses in your cluster.




Additional Istio Ingress gateways can be enabled via the overlay file.

#### Egress Support

By default the Egress gateway is disabled, but can be enabled on install or upgrade through the values.yaml or via the overlay file.

## Additional Steps for Installing Istio on an SUSE® Rancher Prime: RKE2 Cluster

To install Istio on an RKE2 cluster, follow the steps in this section.

## Upgrading Istio in an Air-Gapped Environment

The Istio pod security policy is now enabled by default. A new value, `installer.releaseMirror.enabled`, has been added to the rancher-istio chart to enable and disable the server that supports air-gapped upgrades. Note that `installer.releaseMirror.enabled` is set to `false` by default. You can set this value as needed when you install or upgrade. Follow the steps below:

1. Provision an air-gapped Rancher instance and an air-gapped custom cluster in the Rancher UI.
2. Install Monitoring in the cluster: **Cluster Explorer -> Apps & Marketplace -> Charts -> Monitoring**.
3. Pull all required images for Istio into the private registry you will use in the air-gapped environment.

4. Install Istio in the cluster: **Cluster Explorer -> Apps & Marketplace -> Charts -> Istio**.



You can enable Jaeger [https://www.jaegertracing.io/] and Kiali [https://kiali.io/] on a fresh Istio install. To ensure that Jaeger and Kiali work, set `installer.releaseMirror.enabled` to `true` in `values.yaml` during installation.

1. Upgrade the Istio installation.



If you haven't already, set `installer.releaseMirror.enabled=true` to upgrade Istio.

## Role-based Access Control



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

This section describes the permissions required to access Istio features.

The rancher istio chart installs three `ClusterRoles`

### Cluster-Admin Access

By default, only those with the `cluster-admin` `ClusterRole` can:

- Install istio app in a cluster
- Configure resource allocations for Istio

### Admin and Edit access

By default, only Admin and Edit roles can:

- Enable and disable Istio sidecar auto-injection for namespaces
- Add the Istio sidecar to workloads
- View the traffic metrics and traffic graph for the cluster
- Configure Istio's resources (such as the gateway, destination rules, or virtual services)

### Summary of Default Permissions for Kubernetes Default roles

Istio creates three `ClusterRoles` and adds Istio CRD access to the following default K8s `ClusterRole`:

| ClusterRole create by chart | Default K8s ClusterRole | Rancher Role |
|---|---|---|
| `istio-admin` | admin | Project Owner |
| `istio-edit` | edit | Project Member |
| `istio-view` | view | Read-only |

Rancher will continue to use cluster-owner, cluster-member, project-owner, project-member, etc as role

names, but will utilize default roles to determine access. For each default K8s `ClusterRole` there are different Istio CRD permissions and K8s actions (Create ( C ), Get ( G ), List ( L ), Watch ( W ), Update ( U ), Patch ( P ), Delete( D ), All ( * )) that can be performed.

| CRDs | Admin | Edit | View |
|---|---|---|---|
| • config.istio.io<br>adapters<br>attributemanifests<br>handlers<br>httpapispecbindings<br>httpapispecs<br>instances<br>quotaspecbindings<br>quotaspecs<br>rules<br>templates | GLW | GLW | GLW |
| • networking.istio.io<br>destinationrules<br>envoyfilters<br>gateways<br>serviceentries<br>sidecars<br>virtualservices<br>workloadentries | * | * | GLW |
| • security.istio.io<br>authorizationpolicies<br>peerauthentications<br>requestauthentications | * | * | GLW |

## CPU and Memory Allocations



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

This section describes the minimum recommended computing resources for the Istio components in a cluster.

The CPU and memory allocations for each component are configurable.

Before enabling Istio, we recommend that you confirm that your Rancher worker nodes have enough CPU and memory to run all of the components of Istio.



In larger deployments, it is strongly advised that the infrastructure be placed on dedicated nodes in the cluster by adding a node selector for each Istio component.

The table below shows a summary of the minimum recommended resource requests and limits for the CPU and memory of each core Istio component.

In Kubernetes, the resource request indicates that the workload will not deployed on a node unless the node has at least the specified amount of memory and CPU available. If the workload surpasses the limit for CPU or memory, it can be terminated or evicted from the node. For more information on managing resource limits for containers, refer to the Kubernetes documentation. [https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/]

| Workload | CPU - Request | Memory - Request | CPU - Limit | Memory - Limit |
|---|---|---|---|---|
| ingress gateway | 100m | 128mi | 2000m | 1024mi |
| egress gateway | 100m | 128mi | 2000m | 1024mi |
| istiod | 500m | 2048mi | No limit | No limit |
| proxy | 10m | 10mi | 2000m | 1024mi |
| **Totals:** | **710m** | **2314Mi** | **6000m** | **3072Mi** |

**Configuring Resource Allocations**

You can individually configure the resource allocation for each type of Istio component. This section includes the default resource allocations for each component.

To make it easier to schedule the workloads to a node, a cluster-admin can reduce the CPU and memory resource requests for the component. However, the default CPU and memory allocations are the minimum that we recommend.

You can find more information about Istio configuration in the official Istio documentation [https://istio.io/].

To configure the resources allocated to an Istio component,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click **Apps**.
4. Click **Installed Apps**.
5. Go to the `istio-system` namespace. In one of the Istio workloads, such as `rancher-istio`, click ☰ > **Edit/Upgrade**.
6. Click **Upgrade** to edit the base components via changes to the values.yaml or add an overlay file. For more information about editing the overlay file, see this section.
7. Change the CPU or memory allocations, the nodes where each component will be scheduled to, or the node tolerations.
8. Click **Upgrade**. to rollout changes

**Result:** The resource allocations for the Istio components are updated.

#### Editing the Overlay File

The overlay file can contain any of the values in the Istio Operator spec. [https://istio.io/latest/docs/reference/config/istio.operator.v1alpha1/#IstioOperatorSpec] The overlay file included with the Istio application is just one example of a potential configuration of the overlay file.

As long as the file contains `kind: IstioOperator` and the YAML options are valid, the file can be used as an overlay.

In the example overlay file provided with the Istio application, the following section allows you to change Kubernetes resources:

```
#     k8s:
#       resources:
#         requests:
#           cpu: 200m
```

## Disabling Istio



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

This section describes how to uninstall Istio in a cluster or disable a namespace, or workload.

#### Uninstall Istio in a Cluster

To uninstall Istio,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click menu:Apps[Installed Apps].
4. In the `istio-system` namespace, go to `rancher-istio` and click ☰ > **Delete**.
5. After `rancher-istio` is deleted, you can then select all the remaining apps in the `istio-system` namespace and click **Delete**.

**Result:** The `rancher-istio` app in the cluster gets removed. The Istio sidecar cannot be deployed on any workloads in the cluster.



You can no longer disable and re-enable your Istio installation. If you would like to save your settings for a future install, view and save individual YAMLs to refer back to / reuse for future installations.

**Troubleshooting Uninstall:** If you didn't follow the uninstall steps, you may encounter a warning during uninstall:

```
Error: uninstallation completed with 1 error(s): unable to build kubernetes
```

`objects for delete: unable to recognize "": no matches for kind "MonitoringDashboard" in version "monitoring.kiali.io/vlalphal"`

This could mean a few things. You either selected all the apps in the `istio-system` namespace and deleted them at the same time, or you deleted `rancher-istio` chart dependencies prior to deleting the `rancher-istio` chart. Since the uninstall did not complete properly, you will have resources remaining in the `istio-system` namespace that you will need to manually clean up. Another option to avoid manual clean up is to install `rancher-istio` again, then uninstall it in the correct order.

#### Disable Istio in a Namespace

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click menu:Cluster[Projects/Namespaces].
4. Go to the namespace where you want to enable Istio and click ☰ > **Enable Istio Auto Injection**. Alternately, click the namespace, and then on the namespace detail page, click ☰ > **Enable Istio Auto Injection**.

**Result:** When workloads are deployed in this namespace, they will not have the Istio sidecar.

#### Remove the Istio Sidecar from a Workload

Disable Istio in the namespace, then redeploy the workloads with in it. They will be deployed without the Istio sidecar.

## Configuration Options



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

#### Egress Support

By default the Egress gateway is disabled, but can be enabled on install or upgrade through the values.yaml or via the overlay file.

#### Enabling Automatic Sidecar Injection

Automatic sidecar injection is disabled by default. To enable this, set the `sidecarInjectorWebhook.enableNamespacesByDefault=true` in the values.yaml on install or upgrade. This automatically enables Istio sidecar injection into all new namespaces that are deployed.

#### Overlay File

An Overlay File is designed to support extensive configuration of your Istio installation. It allows you to make changes to any values available in the IstioOperator API [https://istio.io/latest/docs/reference/config/istio.operator.v1alpha1/]. This will ensure you can customize the default installation to fit any scenario.

The Overlay File will add configuration on top of the default installation that is provided from the Istio chart

installation. This means you do not need to redefine the components that already defined for installation.

For more information on Overlay Files, refer to the Istio documentation. [https://istio.io/latest/docs/setup/install/istioctl/#configure-component-settings]

#### Selectors and Scrape Configs

The Monitoring app sets `prometheus.prometheusSpec.ignoreNamespaceSelectors=false` which enables monitoring across all namespaces by default. This ensures you can view traffic, metrics and graphs for resources deployed in a namespace with `istio-injection=enabled` label.

If you would like to limit Prometheus to specific namespaces, set `prometheus.prometheusSpec.ignoreNamespaceSelectors=true`. Once you do this, you must perform some additional configuration to continue to monitor your resources.

For details, refer to this section.

#### Additional Steps for Installing Istio on an SUSE® Rancher Prime: RKE2 Cluster

Refer to this section.

#### Additional Steps for Project Network Isolation

Refer to this section.

#### Selectors and Scrape Configs



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

The Monitoring app sets `prometheus.prometheusSpec.ignoreNamespaceSelectors=false`, which enables monitoring across all namespaces by default.

This ensures you can view traffic, metrics and graphs for resources deployed in a namespace with `istio-injection=enabled` label.

If you would like to limit Prometheus to specific namespaces, set `prometheus.prometheusSpec.ignoreNamespaceSelectors=true`. Once you do this, you must perform some additional configuration to continue to monitor your resources.

##### Limiting Monitoring to Specific Namespaces by Setting ignoreNamespaceSelectors to True

To limit monitoring to specific namespaces, you will edit the `ignoreNamespaceSelectors` Helm chart option. You will configure this option when installing or upgrading the Monitoring Helm chart:

1. When installing or upgrading the Monitoring Helm chart, edit the values.yml and set`prometheus.prometheusSpec.ignoreNamespaceSelectors=true`.
2. Complete the install or upgrade.

**Result:** Prometheus will be limited to specific namespaces which means one of the following configurations

will need to be set up to continue to view data in various dashboards

#### Enabling Prometheus to Detect Resources in Other Namespaces

There are two different ways to enable Prometheus to detect resources in other namespaces when `prometheus.prometheusSpec.ignoreNamespaceSelectors=true`:

- **Monitoring specific namespaces:** Add a Service Monitor or Pod Monitor in the namespace with the targets you want to scrape.
- **Monitoring across namespaces:** Add an `additionalScrapeConfig` to your rancher-monitoring instance to scrape all targets in all namespaces.

#### Monitoring Specific Namespaces: Create a Service Monitor or Pod Monitor

This option allows you to define which specific services or pods you would like monitored in a specific namespace.

The usability tradeoff is that you have to create the service monitor or pod monitor per namespace since you cannot monitor across namespaces.



*Prerequisite:*

Define a ServiceMonitor or PodMonitor for `<your namespace>`. An example ServiceMonitor is provided below.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the top navigation bar, open the kubectl shell.
4. If the ServiceMonitor or PodMonitor file is stored locally in your cluster, in `kubectl create -f <name of service/pod monitor file>.yaml`.
5. If the ServiceMonitor or PodMonitor is not stored locally, run `cat<< EOF | kubectl apply -f -`, paste the file contents into the terminal, then run `EOF` to complete the command.
6. Run `kubectl label namespace <your namespace> istio-injection=enabled` to enable the envoy sidecar injection.

**Result:** `<your namespace>` can be scraped by prometheus.

*Example Service Monitor for Istio Proxies*

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: envoy-stats-monitor
  namespace: istio-system
  labels:
    monitoring: istio-proxies
spec:
  selector:
    matchExpressions:
```

```
    - {key: istio-prometheus-ignore, operator: DoesNotExist}
  namespaceSelector:
    any: true
  jobLabel: envoy-stats
  endpoints:
  - path: /stats/prometheus
    targetPort: 15090
    interval: 15s
    relabelings:
    - sourceLabels: [__meta_kubernetes_pod_container_port_name]
      action: keep
      regex: '.*-envoy-prom'
    - action: labeldrop
      regex: "__meta_kubernetes_pod_label_(.+)"
    - sourceLabels: [__meta_kubernetes_namespace]
      action: replace
      targetLabel: namespace
    - sourceLabels: [__meta_kubernetes_pod_name]
      action: replace
      targetLabel: pod_name
```

**Monitoring across namespaces: Set ignoreNamespaceSelectors to False**

This enables monitoring across namespaces by giving Prometheus additional scrape configurations.

The usability tradeoff is that all of Prometheus' `additionalScrapeConfigs` are maintained in a single Secret. This could make upgrading difficult if monitoring is already deployed with additionalScrapeConfigs before installing Istio.

1. When installing or upgrading the Monitoring Helm chart, edit the values.yml and set the `prometheus.prometheusSpec.additionalScrapeConfigs` array to the **Additional Scrape Config** provided below.
2. Complete the install or upgrade.

**Result:** All namespaces with the `istio-injection=enabled` label will be scraped by prometheus.

*Additional Scrape Config*

```
- job_name: 'istio/envoy-stats'
  scrape_interval: 15s
  metrics_path: /stats/prometheus
  kubernetes_sd_configs:
    - role: pod
  relabel_configs:
    - source_labels: [__meta_kubernetes_pod_container_port_name]
      action: keep
      regex: '.*-envoy-prom'
```

```
    - source_labels: [__address__,
__meta_kubernetes_pod_annotation_prometheus_io_port]
      action: replace
      regex: ([^:]+)(?::\d+)?;(\d+)
      replacement: $1:15090
      target_label: __address__
    - action: labelmap
      regex: __meta_kubernetes_pod_label_(.+)
    - source_labels: [__meta_kubernetes_namespace]
      action: replace
      target_label: namespace
    - source_labels: [__meta_kubernetes_pod_name]
      action: replace
      target_label: pod_name
```

**Additional Steps for Project Network Isolation**



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

In clusters where:

- You are using Rancher v2.5.8+ with any RKE2 network plug-in that supports the enforcement of Kubernetes network policies, such as Canal
- The Project Network Isolation option is enabled
- You install the Istio Ingress module

The Istio Ingress Gateway pod won't be able to redirect ingress traffic to the workloads by default. This is because all the namespaces will be inaccessible from the namespace where Istio is installed. You have two options.

The first option is to add a new Network Policy in each of the namespaces where you intend to have ingress controlled by Istio. Your policy should include the following lines:

```
- podSelector:
    matchLabels:
      app: istio-ingressgateway
```

The second option is to move the `istio-system` namespace to the `system` project, which by default is excluded from the network isolation.

**Additional Steps for Installing Istio on SUSE® Rancher Prime: RKE2 and SUSE® Rancher Prime: K3s Clusters**



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

When installing or upgrading the Istio Helm chart through **Apps,**

1. If you are installing the chart, click **Customize Helm options before install** and click **Next**.
2. You will see options for configuring the Istio Helm chart. On the **Components** tab, check the box next to **Enabled CNI**.
3. Add a custom overlay file specifying `cniBinDir` and `cniConfDir`. For more information on these options, refer to the Istio documentation. [https://istio.io/latest/docs/setup/additional-setup/cni/#helm-chart-parameters] An example is below:

**RKE2**

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
    cni:
      enabled: true
      k8s:
        overlays:
        - apiVersion: "apps/v1"
          kind: "DaemonSet"
          name: "istio-cni-node"
          patches:
          - path: spec.template.spec.containers.[name:install-
cni].securityContext.privileged
            value: true
  values:
    cni:
      cniBinDir: /opt/cni/bin
      cniConfDir: /etc/cni/net.d
```

**K3s**

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  components:
```

```
    cni:
      enabled: true
      k8s:
        overlays:
        - apiVersion: "apps/v1"
          kind: "DaemonSet"
          name: "istio-cni-node"
          patches:
          - path: spec.template.spec.containers.[name:install-
cni].securityContext.privileged
            value: true
  values:
    cni:
      cniBinDir: /var/lib/rancher/k3s/data/current/bin
      cniConfDir: /var/lib/rancher/k3s/agent/etc/cni/net.d
```

**Result:** Now you should be able to utilize Istio as desired, including sidecar injection and monitoring via Kiali.

#### Enable Istio in the Cluster



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].



*Prerequisites:*

- Only a user with the `cluster-admin` Kubernetes default role [https://kubernetes.io/docs/reference/access-authn-authz/rbac/#user-facing-roles] assigned can configure and install Istio in a Kubernetes cluster.
- To install Istio on an RKE2 cluster, additional steps are required. For details, see this section.
- To install Istio in a cluster where project network isolation is enabled, additional steps are required. For details, see this section.

1. Click ☰ > **Cluster Management**.
2. Go to the where you want to enable Istio and click **Explore**.
3. Click **Apps**.
4. Click **Charts**.
5. Click **Istio**.
6. If you have not already installed your own monitoring app, you will be prompted to install the rancher-monitoring app. Optional: Set your Selector or Scrape config options on rancher-monitoring app install.
7. Optional: Configure member access and resource limits for the Istio components. Ensure you have

enough resources on your worker nodes to enable Istio.

8. Optional: Make additional configuration changes to values.yaml if needed.
9. Optional: Add further resources or configuration via the overlay file.
10. Click **Install**.

**Result:** Istio is installed at the cluster level.

#### Additional Config Options

For more information on configuring Istio, refer to the configuration reference.

### Enable Istio in a Namespace



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

You will need to manually enable Istio in each namespace that you want to be tracked or controlled by Istio. When Istio is enabled in a namespace, the Envoy sidecar proxy will be automatically injected into all new workloads that are deployed in the namespace.

This namespace setting will only affect new workloads in the namespace. Any preexisting workloads will need to be re-deployed to leverage the sidecar auto injection.



*Prerequisite:*

To enable Istio in a namespace, the cluster must have Istio installed.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click menu:Cluster[Projects/Namespaces].
4. Go to the namespace where you want to enable Istio and click ☰ > **Enable Istio Auto Injection**. Alternately, click the namespace, and then on the namespace detail page, click ☰ > **Enable Istio Auto Injection**.

**Result:** The namespace now has the label `istio-injection=enabled`. All new workloads deployed in this namespace will have the Istio sidecar injected by default.

#### Verifying that Automatic Istio Sidecar Injection is Enabled

To verify that Istio is enabled, deploy a hello-world workload in the namespace. Go to the workload and click the pod name. In the **Containers** section, you should see the `istio-proxy` container.

#### Excluding Workloads from Being Injected with the Istio Sidecar

If you need to exclude a workload from getting injected with the Istio sidecar, use the following annotation on the workload:

```
sidecar.istio.io/inject: “false”
```

To add the annotation to a workload,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click **Workload**.
4. Go to the workload that should not have the sidecar and edit as yaml
5. Add the following key, value `sidecar.istio.io/inject: false` as an annotation on the workload
6. Click **Save**.

**Result:** The Istio sidecar will not be injected into the workload.



If you are having issues with a Job you deployed not completing, you will need to add this annotation to your pod using the provided steps. Since Istio Sidecars run indefinitely, a Job cannot be considered complete even after its task has completed.

**Next: Add Deployments with the Istio Sidecar**

### Set up Istio’s Components for Traffic Management



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

A central advantage of traffic management in Istio is that it allows dynamic request routing. Some common applications for dynamic request routing include canary deployments and blue/green deployments. The two key resources in Istio traffic management are *virtual services* and *destination rules*.

- Virtual services [https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/] intercept and direct traffic to your Kubernetes services, allowing you to divide percentages of traffic from a request to different services. You can use them to define a set of routing rules to apply when a host is addressed.
- Destination rules [https://istio.io/docs/reference/config/networking/v1alpha3/destination-rule/] serve as the single source of truth about which service versions are available to receive traffic from virtual services. You can use these resources to define policies that apply to traffic that is intended for a service after routing has occurred.

This section describes how to add an example virtual service that corresponds to the `reviews` microservice in the sample BookInfo app. The purpose of this service is to divide traffic between two versions of the `reviews` service.

In this example, we take the traffic to the `reviews` service and intercept it so that 50 percent of it goes to `v1` of the service and 50 percent goes to `v2`.

After this virtual service is deployed, we will generate traffic and see from the Kiali visualization that traffic is being routed evenly between the two versions of the service.

To deploy the virtual service and destination rules for the `reviews` service,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster where Istio is installed and click **Explore**.
3. In the cluster where Istio is installed, click menu:Istio[DestinationRules] in the left navigation bar.
4. Click **Create**.
5. Copy and paste the DestinationRule yaml provided below.
6. Click **Create**.
7. Click **Edit as YAML** and use this configuration:

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
```

8. Click **Create**.

Then to deploy the VirtualService that provides the traffic routing that utilizes the DestinationRule:

1. Click **VirtualService** in the side nav bar.
2. Click **Create from Yaml**.
3. Copy and paste the VirtualService yaml provided below.
4. Click **Create**.

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
```

```
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
---
```

**Result:** When you generate traffic to this service (for example, by refreshing the ingress gateway URL), the Kiali traffic graph will reflect that traffic to the `reviews` service is divided evenly between `v1` and `v3`.

Next: Generate and View Traffic

## Istio Setup Guides



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

This section describes how to enable Istio and start using it in your projects.

If you use Istio for traffic management, you will need to allow external traffic to the cluster. In that case, you will need to follow all of the steps below.

### Prerequisites

This guide assumes you have already installed Rancher, and you have already provisioned a separate Kubernetes cluster on which you will install Istio.

The nodes in your cluster must meet the CPU and memory requirements.

The workloads and services that you want to be controlled by Istio must meet Istio's requirements. [https://istio.io/docs/setup/additional-setup/requirements/]

### Install



*Quick Setup Tip:*

If you don't need external traffic to reach Istio, and you just want to set up Istio for monitoring and tracing traffic within the cluster, skip the steps for setting up the Istio gateway and setting up Istio's components for traffic management.

1. Enable Istio in the cluster.
2. Enable Istio in all the namespaces where you want to use it.

3. Add deployments and services that have the Istio sidecar injected.
4. Set up the Istio gateway.
5. Set up Istio's components for traffic management.
6. Generate traffic and see Istio in action.

### Generate and View Traffic from Istio



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

#### The Kiali Traffic Graph

The Istio overview page provides a link to the Kiali dashboard. From the Kiali dashboard, you can view graphs for each namespace. The Kiali graph provides a powerful way to visualize the topology of your Istio service mesh. It shows you which services communicate with each other.

#### Prerequisites

To enable traffic to show up in the graph, ensure that you have Prometheus installed in the cluster. `Rancher-istio` installs Kiali, and configures it by default to work with the `rancher-monitoring` chart. You can use `rancher-monitoring` or install your own monitoring solution.

Additionally, for Istio installations version `103.1.0+upl.19.6` and later, Kiali uses a token value for its authentication strategy. If you are trying to generate or retrieve the token (e.g. for login), note that the name of the Kiali service account in Rancher is `kiali`. For more information, refer to the Kiali token authentication FAQ [https://kiali.io/docs/faq/authentication/].

Optional: You can configure which namespaces data scraping occurs in by setting the Helm chart options described in Selectors & Scrape Configs.

#### Traffic Visualization

To see the traffic graph follow the steps below:

1. In the cluster where Istio is installed, click **Istio** in the left navigation bar.
2. Click the **Kiali** link.
3. Click on **Graph** in the side nav.
4. Change the namespace in the **Namespace** dropdown to view the traffic for each namespace.

If you refresh the URL to the BookInfo app several times, you should be able to see green arrows on the Kiali graph showing traffic to `v1` and `v3` of the `reviews` service. The control panel on the right side of the graph lets you configure details including how many minutes of the most recent traffic should be shown on the graph.

For additional tools and visualizations, you can go to Grafana, and Prometheus dashboards from the **Monitoring Overview** page

### Set up the Istio Gateway



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].

The gateway to each cluster can have its own port or load balancer, which is unrelated to a service mesh. By default, each Rancher-provisioned cluster has one NGINX ingress controller allowing traffic into the cluster.

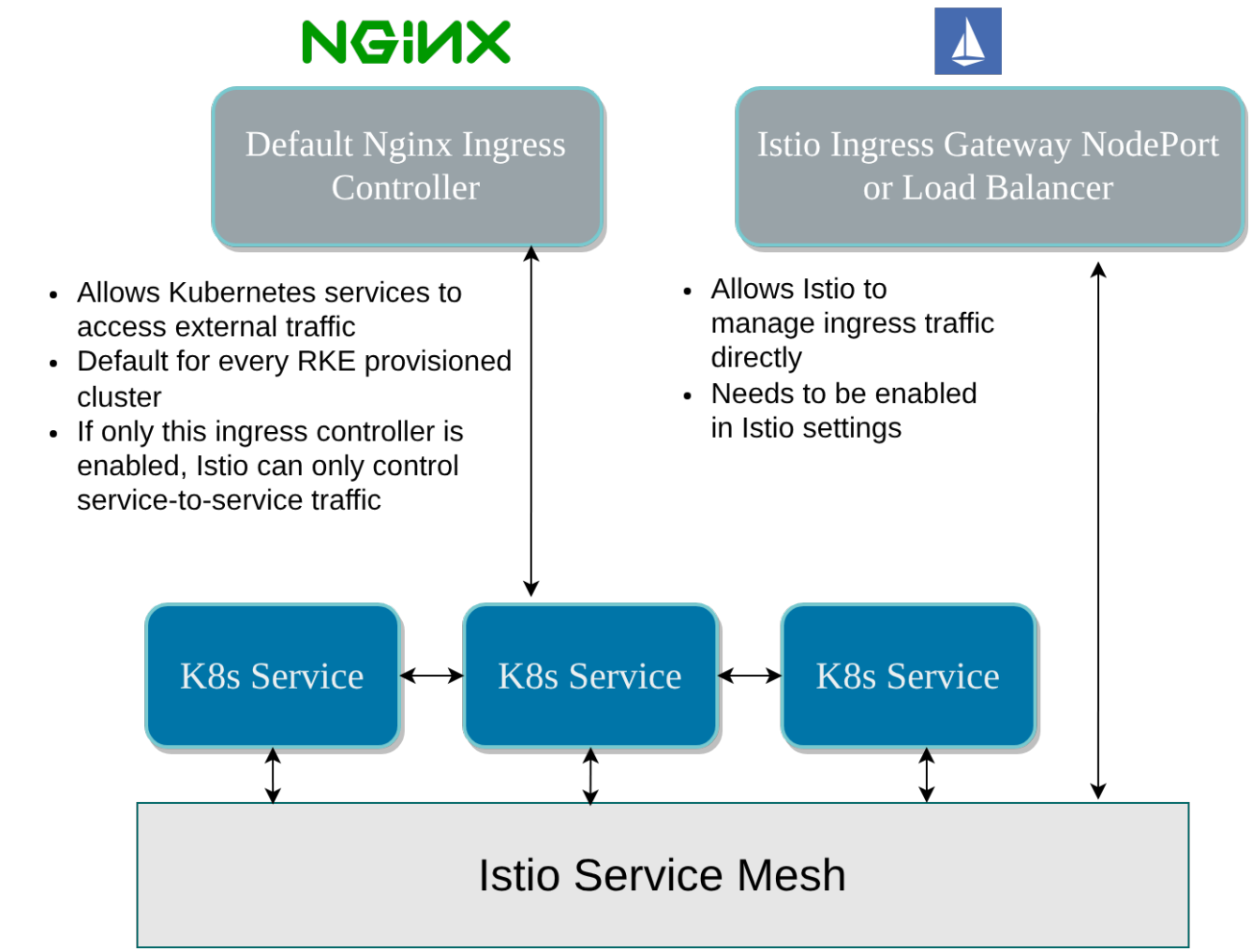
You can use the Nginx Ingress controller with or without Istio installed. If this is the only gateway to your cluster, Istio will be able to route traffic from service to service, but Istio will not be able to receive traffic from outside the cluster.

To allow Istio to receive external traffic, you need to enable Istio's gateway, which works as a north-south proxy for external traffic. When you enable the Istio gateway, the result is that your cluster will have two Ingresses.

You will also need to set up a Kubernetes gateway for your services. This Kubernetes resource points to Istio's implementation of the ingress gateway to the cluster.

You can route traffic into the service mesh with a load balancer or use Istio's NodePort gateway. This section describes how to set up the NodePort gateway.

For more information on the Istio gateway, refer to the Istio documentation. [https://istio.io/docs/reference/config/networking/v1alpha3/gateway/]




#### Enable an Istio Gateway

The ingress gateway is a Kubernetes service that will be deployed in your cluster. The Istio Gateway allows for more extensive customization and flexibility.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click menu:Istio[Gateways].
4. Click **Create from Yaml**.
5. Paste your Istio Gateway yaml, or **Read from File**.
6. Click **Create**.

**Result:** The gateway is deployed, and will now route traffic with applied rules.

#### Example Istio Gateway

We add the BookInfo app deployments in services when going through the Workloads example. Next we add an Istio Gateway so that the app is accessible from outside your cluster.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.

3. In the left navigation bar, click menu:Istio[Gateways].
4. Click **Create from Yaml**.
5. Copy and paste the Gateway yaml provided below.
6. Click **Create**.

```
apiVersion: networking.istio.io/vlalpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
---
```

Then to deploy the VirtualService that provides the traffic routing for the Gateway:

1. Click ☰ **> Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click menu:Istio[VirtualServices].
4. Copy and paste the VirtualService yaml provided below.
5. Click **Create**.

```
apiVersion: networking.istio.io/vlalpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "*"
  gateways:
  - bookinfo-gateway
  http:
  - match:
    - uri:
        exact: /productpage
    - uri:
```

```
        prefix: /static
    - uri:
        exact: /login
    - uri:
        exact: /logout
    - uri:
        prefix: /api/v1/products
    route:
    - destination:
        host: productpage
        port:
          number: 9080
```

**Result:** You have configured your gateway resource so that Istio can receive traffic from outside the cluster.

Confirm that the resource exists by running:

```
kubectl get gateway -A
```

The result should be something like this:

```
NAME               AGE
bookinfo-gateway   64m
```

#### Access the ProductPage Service from a Web Browser

To test and see if the BookInfo app deployed correctly, the app can be viewed a web browser using the Istio controller IP and port, combined with the request name specified in your Kubernetes gateway resource:

`http://<IP of Istio controller>:<Port of istio controller>/productpage`

To get the ingress gateway URL and port,

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Scroll down to the `istio-system` namespace.
5. Within `istio-system`, there is a workload named `istio-ingressgateway`. Under the name of this workload, you should see links, such as `80/tcp`.
6. Click one of those links. This should show you the URL of the ingress gateway in your web browser. Append `/productpage` to the URL.

**Result:** You should see the BookInfo app in the web browser.

For help inspecting the Istio controller URL and ports, try the commands the Istio documentation. [https://istio.io/docs/tasks/traffic-management/ingress/ingress-control/#determining-the-ingress-ip-and-ports]

#### Troubleshooting

The official Istio documentation [https://istio.io/docs/tasks/traffic-management/ingress/ingress-control/#troubleshooting] suggests `kubectl` commands to inspect the correct ingress host and ingress port for external requests.

##### Confirming that the Kubernetes Gateway Matches Istio's Ingress Controller

You can try the steps in this section to make sure the Kubernetes gateway is configured properly.

In the gateway resource, the selector refers to Istio's default ingress controller by its label, in which the key of the label is `istio` and the value is `ingressgateway`. To make sure the label is appropriate for the gateway, do the following:

1. Click ☰ **> Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the left navigation bar, click **Workload**.
4. Scroll down to the `istio-system` namespace.
5. Within `istio-system`, there is a workload named `istio-ingressgateway`. Click the name of this workload and go to the **Labels and Annotations** section. You should see that it has the key `istio` and the value `ingressgateway`. This confirms that the selector in the Gateway resource matches Istio's default ingress controller.

**Next: Set up Istio's Components for Traffic Management**

### Add Deployments and Services with the Istio Sidecar



Rancher-Istio [https://github.com/rancher/charts/tree/release-v2.11/charts/rancher-istio] has been deprecated since Rancher v2.12.0; turn to the SUSE Application Collection [https://apps.rancher.io] build of Istio for enhanced security (included in SUSE Rancher Prime subscriptions). Detailed information can be found in this announcement [https://forums.suse.com/t/deprecation-of-rancher-istio/45043].



*Prerequisite:*

To enable Istio for a workload, the cluster and namespace must have the Istio app installed.

Enabling Istio in a namespace only enables automatic sidecar injection for new workloads. To enable the Envoy sidecar for existing workloads, you need to enable it manually for each workload.

To inject the Istio sidecar on an existing workload in the namespace,

1. In the upper left corner, click ☰ **> Cluster Management**.
2. On the **Clusters** page, go to the cluster where you want to see the visualizations and click **Explore**.
3. Click **Workload**.
4. Go to the workload where you want to inject the Istio sidecar and click ☰ **> Redeploy**. When the workload is redeployed, it will have the Envoy sidecar automatically injected.

Wait a few minutes for the workload to upgrade to have the istio sidecar. Click it and go to the Containers

section. You should be able to see `istio-proxy` alongside your original workload. This means the Istio sidecar is enabled for the workload. Istio is doing all the wiring for the sidecar envoy. Now Istio can do all the features automatically if you enable them in the yaml.

#### Add Deployments and Services

There are a few ways to add new **Deployments** in your namespace:

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click **Workload**.
4. Click **Create**.
5. Click **Deployment**.
6. Fill out the form, or **Edit as Yaml**.
7. Click **Create**.

To add a **Service** to your namespace:

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. Click menu:Service Discovery[Services].
4. Click **Create**.
5. Select the type of service that you want.
6. Fill out the form, or **Edit as Yaml**.
7. Click **Create**

You can also create deployments and services using the kubectl **shell**

1. Run `kubectl create -f <name of service/deployment file>.yaml` if your file is stored locally in the cluster
2. Or run `cat<< EOF | kubectl apply -f -`, paste the file contents into the terminal, then run `EOF` to complete the command.

#### Example Deployments and Services

Next we add the Kubernetes resources for the sample deployments and services for the BookInfo app in Istio's documentation.

1. Click ☰ > **Cluster Management**.
2. Go to the cluster that you created and click **Explore**.
3. In the top navigation bar, open the kubectl shell.
4. Run `cat<< EOF | kubectl apply -f -`
5. Copy the below resources into the shell.
6. Run `EOF`

This will set up the following sample resources from Istio's example BookInfo app:

Details service and deployment:

- A `details` Service
- A ServiceAccount for `bookinfo-details`
- A `details-v1` Deployment

Ratings service and deployment:

- A `ratings` Service
- A ServiceAccount for `bookinfo-ratings`
- A `ratings-v1` Deployment

Reviews service and deployments (three versions):

- A `reviews` Service
- A ServiceAccount for `bookinfo-reviews`
- A `reviews-v1` Deployment
- A `reviews-v2` Deployment
- A `reviews-v3` Deployment

Productpage service and deployment:

This is the main page of the app, which will be visible from a web browser. The other services will be called from this page.

- A `productpage` service
- A ServiceAccount for `bookinfo-productpage`
- A `productpage-v1` Deployment

**Resource YAML**

```
# Copyright 2017 Istio Authors
#
#   Licensed under the Apache License, Version 2.0 (the "License");
#   you may not use this file except in compliance with the License.
#   You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.
```

```
##################################################################################
##########################
# Details service
##################################################################################
##########################
apiVersion: v1
kind: Service
metadata:
  name: details
  labels:
    app: details
    service: details
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: details
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-details
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: details-v1
  labels:
    app: details
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: details
      version: v1
  template:
    metadata:
      labels:
        app: details
        version: v1
    spec:
      serviceAccountName: bookinfo-details
      containers:
```

```
      - name: details
        image: docker.io/istio/examples-bookinfo-details-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
##################################################################################
##########################
# Ratings service
##################################################################################
##########################
apiVersion: v1
kind: Service
metadata:
  name: ratings
  labels:
    app: ratings
    service: ratings
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: ratings
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-ratings
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-v1
  labels:
    app: ratings
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ratings
      version: v1
  template:
    metadata:
```

```
      labels:
        app: ratings
        version: v1
    spec:
      serviceAccountName: bookinfo-ratings
      containers:
      - name: ratings
        image: docker.io/istio/examples-bookinfo-ratings-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
##################################################################################
##########################
# Reviews service
##################################################################################
##########################
apiVersion: v1
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: reviews
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v1
  labels:
    app: reviews
    version: v1
spec:
  replicas: 1
```

```
  selector:
    matchLabels:
      app: reviews
      version: v1
  template:
    metadata:
      labels:
        app: reviews
        version: v1
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
      - name: reviews
        image: docker.io/istio/examples-bookinfo-reviews-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v2
  labels:
    app: reviews
    version: v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v2
  template:
    metadata:
      labels:
        app: reviews
        version: v2
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
      - name: reviews
        image: docker.io/istio/examples-bookinfo-reviews-v2:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v3
  labels:
    app: reviews
    version: v3
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v3
  template:
    metadata:
      labels:
        app: reviews
        version: v3
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
      - name: reviews
        image: docker.io/istio/examples-bookinfo-reviews-v3:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
##################################################################################
############################
# Productpage services
##################################################################################
############################
apiVersion: v1
kind: Service
metadata:
  name: productpage
  labels:
    app: productpage
    service: productpage
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: productpage
```

```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-productpage
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: productpage-v1
  labels:
    app: productpage
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: productpage
      version: v1
  template:
    metadata:
      labels:
        app: productpage
        version: v1
    spec:
      serviceAccountName: bookinfo-productpage
      containers:
      - name: productpage
        image: docker.io/istio/examples-bookinfo-productpage-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
```

**Next: Set up the Istio Gateway**

# API

## RK-API Quick Start Guide

You can access Rancher's resources through the Kubernetes API. This guide helps you get started on using this API as a Rancher user.

1. In the upper left corner, click ☰ > **Global Settings**.
2. Find and copy the address in the `server-url` field.
3. Create a Rancher API key with no scope.



A Rancher API key with no scope grants unrestricted access to all resources that the user can access. To prevent unauthorized use, this key should be stored securely and rotated frequently.

4. Create a `kubeconfig.yaml` file. Replace `$SERVER_URL` with the server url and `$API_KEY` with your Rancher API key:

```
apiVersion: v1
kind: Config
clusters:
- name: "rancher"
  cluster:
    server: "$SERVER_URL"

users:
- name: "rancher"
  user:
    token: "$API_KEY"

contexts:
- name: "rancher"
  context:
    user: "rancher"
    cluster: "rancher"

current-context: "rancher"
```

You can use this file with any compatible tool, such as kubectl or client-go [https://github.com/kubernetes/client-go]. For a quick demo, see the kubectl example.

For more information on handling more complex certificate setups, see Specifying CA Certs.

For more information on available kubeconfig options, see the upstream documentation [https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/].

### API kubectl Example

In this example, we'll show how to use kubectl to create a project, followed by deleting it. For a list of other Rancher resources available, refer to the API Reference page.



At this time, not all Rancher resources are available through the Rancher Kubernetes API.

1. Set your KUBECONFIG environment variable to the kubeconfig file you just created:

```
export KUBECONFIG=$(pwd)/kubeconfig.yaml
```

2. Use `kubectl explain` to view the available fields for projects, or complex sub-fields of resources:

```
kubectl explain projects
kubectl explain projects.spec
```

   Not all resources may have detailed output.

3. Add the following content to a file named `project.yaml`:

```
apiVersion: management.cattle.io/v3
kind: Project
metadata:
  # name should be unique across all projects in every cluster
  name: p-abcl23
  # generateName can be used instead of `name` to randomly generate a
name.
  # generateName: p-
  # namespace should match spec.ClusterName.
  namespace: local
spec:
  # clusterName should match `metadata.Name` of the target cluster.
  clusterName: local
  description: Example Project
  # displayName is the human-readable name and is visible from the
UI.
  displayName: Example
```

4. Create the project:

```
kubectl create -f project.yaml
```

5. Delete the project:

How you delete the project depends on how you created the project name.

- If you used `name` when creating the project*:

```
kubectl delete -f project.yaml
```

- If you used `generateName`*:

  Replace `$PROJECT_NAME` with the randomly generated name of the project displayed by Kubectl after you created the project.

```
kubectl delete project $PROJECT_NAME -n local
```

### Specifying CA Certs

To ensure that your tools can recognize Rancher's CA certificates, most setups require additional modifications to the above template.

1. In the upper left corner, click ☰ > **Global Settings**.
2. Find and copy the value in the `ca-certs` field.
3. Save the value in a file named `rancher.crt`.



If your Rancher instance is proxied by another service, you must extract the certificate that the service is using, and add it to the kubeconfig file, as demonstrated in step 5.

4. The following commands convert `rancher.crt` to base64 output, trim all new-lines, and update the cluster in the kubeconfig with the certificate, then finish by removing the `rancher.crt` file:

```
export KUBECONFIG=$PATH_TO_RANCHER_KUBECONFIG
kubectl config set clusters.rancher.certificate-authority-data $(cat
rancher.crt | base64 -i - | tr -d '\n')
rm rancher.crt
```

5. (Optional) If you use self-signed certificatess that aren't trusted by your system, you can set the insecure option in your kubeconfig with kubectl:



This option shouldn't be used in production as it is a security risk.

```
export KUBECONFIG=$PATH_TO_RANCHER_KUBECONFIG
kubectl config set clusters.rancher.insecure-skip-tls-verify true
```

If your Rancher instance is proxied by another service, you must extract the certificate that the service is using, and add it to the kubeconfig file, as demonstrated above.

# Workflows

## Kubeconfigs

### Kubeconfig Resource

Kubeconfig is a Rancher resource `kubeconfigs.ext.cattle.io` that allows generating `v1.Config` kubeconfig files for interacting with Rancher and clusters managed by Rancher.

```
kubectl api-resources --api-group=ext.cattle.io
```

To get a description of the fields and structure of the Kubeconfig resource, run:

```
kubectl explain kubeconfigs.ext.cattle.io
```

### Creating a Kubeconfig

Since Rancher v2.13, the `status.bearerToken` contains a fully formed and ready-to-use Bearer token that can be used to authenticate to the Rancher API.

Only a **valid and active** Rancher user can create a Kubeconfig. For example, trying to create a Kubeconfig using a `system:admin` service account leads to an error:

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Kubeconfig
EOF
Error from server (Forbidden): error when creating "STDIN":
kubeconfigs.ext.cattle.io is forbidden: user system:admin is not a
Rancher user
```



The kubeconfig content is generated and returned in the `.status.value` field **only once** when the Kubeconfig is successfully created because it contains secret values for created tokens. Therefore it has to be captured by using an appropriate output option, such as `-o jsonpath='{.status.value}'`, or `-o yaml`.

A kubeconfig can be created for more than one cluster at a time by specifying a list of cluster names in the `spec.clusters` field. You can look up cluster names by listing `clusters.management.cattle.io` resources.

```
kubectl get clusters.management.cattle.io -o=jsonpath
="{.items[*]['metadata.name', 'spec.displayName']}{'\n'}"
local local
c-m-p66cdvlj downstream1
```

The `metadata.name` and `metadata.generateName` fields are ignored, and the name of the new Kubeconfig is automatically generated using the prefix `kubeconfig-`.

You can use the `spec.currentContext` field to set the cluster name, and it is used to set the current context in the kubeconfig. If you do not set the `spec.currentContext` field, then the first cluster in the `spec.clusters` list is used as the current context. For ACE-enabled clusters that don't have an FQDN set, the first control plane node is used as the current context.

For ACE-enabled clusters, if the FQDN is set, then that is used as a cluster entry in the kubeconfig; otherwise, entries for all control plane nodes are created.

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Kubeconfig
spec:
  clusters: [c-m-p66cdvlj, c-m-fcd3g5h]
  description: My Kubeconfig
  currentContext: c-m-p66cdvlj
EOF
```

If `"*"` is specified as the first item in the `spec.clusters` field, the kubeconfig is created for all clusters that the user has access to, if any.

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Kubeconfig
spec:
  clusters: ["*"]
  description: My Kubeconfig
EOF
```

If `spec.ttl` is not specified, the Kubeconfig's tokens is created with the expiration time defined in the `kubeconfig-default-token-ttl-minutes` setting, which is 30 days by default. If `spec.ttl` is specified, it should be greater than 0 and less than or equal to the value of the `kubeconfig-default-token-ttl-minutes` setting expressed in seconds.

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Kubeconfig
spec:
  clusters: [c-m-p66cdvlj] # Downstream cluster
  ttl: 7200 # 2 hours
EOF
```

#### Listing Kubeconfigs

Listing previously generated Kubeconfigs can be useful for cleaning up backing tokens if the Kubeconfig is no longer needed (e.g., it was issued temporarily). Admins can list all Kubeconfigs, while regular users can only view their own.

```
kubectl get kubeconfig
NAME               TTL   TOKENS   STATUS     AGE
kubeconfig-zp786   30d   2/2      Complete   18d
kubeconfig-7zvzp   30d   1/1      Complete   12d
kubeconfig-jznml   30d   1/1      Complete   12d
```

Use `-o wide` to get more details:

```
kubectl get kubeconfig -o wide
NAME               TTL   TOKENS   STATUS     AGE     USER
CLUSTERS       DESCRIPTION
kubeconfig-zp786   30d   2/2      Complete   18d     user-w5gcf   *
all clusters
kubeconfig-7zvzp   30d   1/1      Complete   12d     u-w7drc      *
kubeconfig-jznml   30d   1/1      Complete   12d     u-w7drc      *
```

#### Viewing a Kubeconfig

Admins can get any Kubeconfig, while regular users can only get their own.

```
kubectl get kubeconfig kubeconfig-zp786
NAME               TTL   TOKENS   STATUS     AGE
kubeconfig-zp786   30d   2/2      Complete   18d
```

Use `-o wide` to get more details:

```
kubectl get kubeconfig kubeconfig-zp786 -o wide
NAME               TTL   TOKENS   STATUS     AGE     USER
CLUSTERS       DESCRIPTION
kubeconfig-zp786   30d   2/2      Complete   18d     user-w5gcf   *
all clusters
```

#### Deleting a Kubeconfig

Admins can delete any Kubeconfig, while regular users can only delete their own. When a Kubeconfig is deleted, the kubeconfig tokens are also deleted.

```
kubectl delete kubeconfig kubeconfig-zp786
```

```
kubeconfig.ext.cattle.io "kubeconfig-zp786" deleted
```

To delete a Kubeconfig using preconditions:

```
cat <<EOF | k delete --raw
/apis/ext.cattle.io/v1/kubeconfigs/kubeconfig-zp786 -f -
{
  "apiVersion": "v1",
  "kind": "DeleteOptions",
  "preconditions": {
    "uid": "52183e05-d382-47d2-b4b9-d0735823ce90",
    "resourceVersion": "31331505"
  }
}
EOF
```

#### Deleting a Collection of Kubeconfigs

Admins can delete any Kubeconfig, while regular users can only delete their own.

To delete all Kubeconfigs:

```
kubectl delete --raw /apis/ext.cattle.io/v1/kubeconfigs
```

To delete a collection of Kubeconfigs by label:

```
kubectl delete --raw /apis/ext.cattle.io/v1/kubeconfigs?labelSelector
=foo%3Dbar
```

#### Updating a Kubeconfig

Only the `metadata`, e.g. adding a label or an annotation, and the `spec.description` field can be updated. All other `spec` fields are immutable.

To edit a Kubeconfig:

```
kubectl edit kubeconfig kubeconfig-zp786
```

To patch a Kubeconfig and update its description:

```
kubectl patch kubeconfig kubeconfig-zp786 -type merge -p
'{"spec":{"description":"Updated description"}}'
kubeconfig.ext.cattle.io/kubeconfig-zp786 patched
```

```
kubectl get kubeconfig kubeconfig-fdcpl -o jsonpath
='{.spec.description}'
Updated description
```

To patch a Kubeconfig and add a label:

```
kubectl patch kubeconfig kubeconfig-zp786 -type merge -p
'{"metadata":{"labels":{"foo":"bar"}}}'
kubeconfig.ext.cattle.io/kubeconfig-zp786 patched

kubectl get kubeconfig kubeconfig-zp786 -o jsonpath
='{.metadata.labels.foo}'
bar
```

## Projects

### Creating a Project

Project resources may only be created on the management cluster. See below for creating namespaces under projects in a managed cluster.

#### Creating a Basic Project

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
kind: Project
metadata:
  generateName: p-
  namespace: c-m-abcde
spec:
  clusterName: c-m-abcde
  displayName: myproject
EOF
```

When creating a new project, you have two primary options for setting the name:

- **Automatic Generation:** Use `metadata.generateName` to ensure a unique project ID. However, note that you must use `kubectl create` (instead of `kubectl apply`) with this option, as `kubectl apply` does not support it.
- **Manual Naming:** You can explicitly set the project ID using `metadata.name`. If a project with that exact name already exists, the name request is denied. The display name seen in the UI is set by `spec.displayName`. If `spec.displayName` is not provided, the field `metadata.name` is used instead.

Set `metadata.namespace` and `spec.clusterName` to the ID for the cluster the project belongs to.

If you create a project through a cluster member account and want that account to be able to access the project, you must include the annotation `field.cattle.io/creatorId`, and set it to the cluster member account's user ID.

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
kind: Project
metadata:
  annotations:
    field.cattle.io/creatorId: user-id
  generateName: p-
  namespace: c-m-abcde
spec:
  clusterName: c-m-abcde
  displayName: myproject
EOF
```

Setting the `field.cattle.io/creatorId` field creates a `ProjectRoleTemplateBinding` that grants the specified user the ability to see project resources with the `get` command and view the project in the Rancher UI. Cluster owner and admin accounts don't need to set this annotation to perform these tasks.

Setting the `field.cattle.io/creator-principal-name` annotation to the user's principal preserves it in a projectroletemplatebinding automatically created for the project owner.

If you don't want the creator to be added as the owner member (e.g., if the creator is a cluster administrator) to the project you may set the `field.cattle.io/no-creator-rbac` annotation to `true`, which will prevent the corresponding projectroletemplatebinding from being created.

#### Creating a Project With a Resource Quota

Refer to Kubernetes Resource Quota [https://kubernetes.io/docs/concepts/policy/resource-quotas/].

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
kind: Project
metadata:
  generateName: p-
  namespace: c-m-abcde
spec:
  clusterName: c-m-abcde
  displayName: myproject
  resourceQuota:
    limit:
      limitsCpu: 1000m
  namespaceDefaultResourceQuota:
    limit:
      limitsCpu: 50m
```

```
EOF
```

#### Creating a Project With Container Limit Ranges

Refer to Kubernetes Limit Ranges [https://kubernetes.io/docs/concepts/policy/limit-range/].

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
kind: Project
metadata:
  generateName: p-
  namespace: c-m-abcde
spec:
  clusterName: c-m-abcde
  displayName: myproject
  containerDefaultResourceLimit:
    limitsCpu:    100m
    limitsMemory: 100Mi
    requestsCpu:  50m
    requestsMemory: 50Mi
EOF
```

#### Backing Namespace

After creating the project, the field `status.backingNamespace` gets populated. This represents the namespace in the management cluster that is created to manage project related resources. Examples of resources stored in the backing namespace are project scoped secrets and project role template bindings.

### Adding a Member to a Project

Look up the project's backing namespace to specify the `metadata.namespace` field value and look up the project's ID to specify the `projectName` field value.

```
kubectl --namespace c-m-abcde get projects
```

Look up the role template ID to specify the `roleTemplateName` field value (e.g. `project-member` or `project-owner`).

```
kubectl get roletemplates
```

When adding a user member specify the `userPrincipalName` field:

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
```

```
kind: ProjectRoleTemplateBinding
metadata:
  generateName: prtb-
  namespace: c-m-abcde-p-vwxyz
projectName: c-m-abcde:p-vwxyz
roleTemplateName: project-member
userPrincipalName: keycloak_user://user
EOF
```

When adding a group member specify the `groupPrincipalName` field instead:

```
kubectl create -f - <<EOF
apiVersion: management.cattle.io/v3
kind: ProjectRoleTemplateBinding
metadata:
  generateName: prtb-
  namespace: p-vwxyz
projectName: c-m-abcde:p-vwxyz
roleTemplateName: project-member
groupPrincipalName: keycloak_group://group
EOF
```

Create a projectroletemplatebinding for each role you want to assign to the project member.

#### Listing Project Members

Look up the project backing namespace:

```
kubectl --namespace c-m-abcde get projects
```

To list projectroletemplatebindings in the project's backing namespace:

```
kubectl --namespace c-m-abcde-p-vwxyz get projectroletemplatebindings
```

#### Deleting a Member From a Project

Lookup the projectroletemplatebinding IDs containing the member in the project's namespace as described in the Listing Project Members section.

Delete the projectroletemplatebinding from the project's namespace:

```
kubectl --namespace c-m-abcde-p-vwxyz delete projectroletemplatebindings
prtb-qx874 prtb-7zw7s
```

#### Creating a Namespace in a Project

The Project resource resides in the management cluster, even if the Project is for a managed cluster. The namespaces under the project reside in the managed cluster.

On the management cluster, look up the project ID for the cluster you are administrating if generated using `metadata.generateName`:

```
kubectl --namespace c-m-abcde get projects
```

On the managed cluster, create a namespace with a project annotation:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespace
  annotations:
    field.cattle.io/projectId: c-m-abcde:p-vwxyz
EOF
```

Note the format, `<cluster ID>:<project ID>`.

#### Deleting a Project

Look up the project to delete in the cluster namespace:

```
kubectl --namespace c-m-abcde get projects
```

Delete the project under the cluster namespace:

```
kubectl --namespace c-m-abcde delete project p-vwxyz
```

Note that this command doesn't delete the namespaces and resources that formerly belonged to the project.

It does delete all project role template bindings for the projects, so recreating the project will not restore members added to the project, and you have to add users as members again.

### Tokens



*Deprecation of v3 API tokens*

Rancher v2.13 introduced a new token resource in the `ext.cattle.io` API group to serve as the public API for tokens, which was only accessible through `kubectl`.

Starting with Rancher v2.14.1:

- Legacy v3 API tokens (`tokens.management.cattle.io`) are being phased out and scheduled for removal in a future release. If you currently rely on legacy tokens for automation or API access, transition to the new `tokens.ext.cattle.io` tokens to ensure continued compatibility.
- The Rancher dashboard provides basic support for `tokens.ext.cattle.io` (create, view or list, and delete).

#### Token Resource

Rancher has an imperative API resource `tokens.ext.cattle.io` that allows you to generate tokens for authenticating with Rancher.

```
kubectl api-resources --api-group=ext.cattle.io
```

To get a description of the fields and structure of the Token resource, run:

```
kubectl explain tokens.ext.cattle.io
```

#### Creating a Token



The Token value is only returned once in the `status.value` field.

Only a **valid and active** Rancher user can create a Token. Otherwise, you will get an error displayed (`Error from server (Forbidden)...`) when attempting to create a Token.

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Token
EOF
Error from server (Forbidden): error when creating "STDIN":
tokens.ext.cattle.io is forbidden: user system:admin is not a Rancher
user
```

A Token is always created for the user making the request. Attempting to create a Token for a different user, by specifying a different `spec.userID`, is forbidden and will fail.

- The `spec.description` field can be set to an arbitrary human-readable description of the Token's purpose. The default value is empty.
- The `spec.kind` field can be set to the kind of Token. The value `session` indicates a login Token. All other values, including the default empty string, indicate a kind of derived Token.
- The `metadata.name` and `metadata.generateName` fields are ignored, and the name of the new Token is automatically generated using the prefix `token-`.

  ```
  kubectl create -o jsonpath='{.status.value}' -f -<<EOF
  ```

```
apiVersion: ext.cattle.io/v1
kind: Token
spec:
  description: My Token
EOF
```

- If the `spec.ttl` is not specified, the Token is created with the expiration time defined in the `auth-token-max-ttl-minutes` setting. The default expiration time is 90 days. If `spec.ttl` is specified, it should be greater than 0 and less than or equal to the value of the `auth-token-max-ttl-minutes` setting expressed in milliseconds.

```
kubectl create -o jsonpath='{.status.value}' -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: Token
spec:
  ttl: 7200000 # 2 hours
EOF
```

#### Listing Tokens

Listing previously generated Tokens can help clean up tokens that are no longer needed (e.g., they were issued temporarily). Admins can list all Tokens, while regular users can only see their own.

```
kubectl get tokens.ext.cattle.io
NAME          KIND   TTL   AGE
token-chjc9          90d   18s
token-6fzgj          90d   16s
token-8nbrm          90d   14s
```

Use `-o wide` to get more details:

```
kubectl get tokens.ext.cattle.io -o wide
NAME          USER         KIND   TTL   AGE   DESCRIPTION
token-chjc9   user-jtghh          90d   24s   example
token-6fzgj   user-jtghh          90d   22s   box
token-8nbrm   user-jtghh          90d   20s   jinx
```

#### Viewing a Token

Admins can get any Token, while regular users can only get their own.

```
kubectl get tokens.ext.cattle.io token-chjc9
NAME          KIND   TTL   AGE
```

```
token-chjc9          90d   18s
```

Use `-o wide` to get more details:

```
kubectl get tokens.ext.cattle.io token-chjc9 -o wide
NAME          USER         KIND   TTL   AGE   DESCRIPTION
token-chjc9   user-jtghh          90d   24s   example
```

#### Deleting a Token

Admins can delete any Token, while regular users can only delete their own.

```
kubectl delete tokens.ext.cattle.io token-chjc9
token.ext.cattle.io "token-chjc9" deleted
```

#### Updating a Token

Only the metadata fields `spec.description`, `spec.ttl`, and `spec.enabled` can be updated. All other `spec` fields are immutable. Admins can extend the `spec.ttl` field, while regular users can only reduce the value.

An example `kubectl` command to edit a Token:

```
kubectl edit tokens.ext.cattle.io token-zp786
```

### Users

#### User Resource

The `User` resource (users.management.cattle.io) represents a user account in Rancher.

To get a description of the fields and structure of the `User` resource, run:

```
kubectl explain users.management.cattle.io
```

#### Creating a User

Creating a local user is a two-step process: you must create the `User` resource, then provide a password via a Kubernetes `Secret`.

Only a user with sufficient permissions can create a `User` resource.

```
kubectl create -f -<<EOF
apiVersion: management.cattle.io/v3
kind: User
```

```
metadata:
  name: testuser
displayName: "Test User"
username: "testuser"
EOF
```

The user's password must be provided in a `Secret` object within the `cattle-local-user-passwords` namespace. The Rancher webhook automatically hashes the password and updates the `Secret`.



Important: The `Secret` must have the same name as the metadata.name (and username) of the `User` resource.

```
kubectl create -f -<<EOF
apiVersion: v1
kind: Secret
metadata:
  name: testuser
  namespace: cattle-local-user-passwords
type: Opaque
stringData:
  password: Pass1234567!
EOF
```

After the plaintext password is submitted, the Rancher-Webhook automatically hashes it, replacing the content of the `Secret`, ensuring that the plaintext password is never stored:

```
apiVersion: v1
data:
  password: 1clY4CdjlehGWFz26F4l4x2qoj4gch5L5OXsx35MAa8=
  salt: m8Co+CfMDo5XwVl0FqYzGcRIOTgRrwFSqW8yurh5DcE=
kind: Secret
metadata:
  annotations:
    cattle.io/password-hash: pbkdf2sha3512
  name: testuser
  namespace: cattle-local-user-passwords
  ownerReferences:
  - apiVersion: management.cattle.io/v3
    kind: User
    name: testuser
    uid: 663ffb4f-8178-46c8-85a3-337f4d5cbc2e
  uid: bade9f0a-b06f-4a77-9a39-4284dc2349c5
type: Opaque
```

#### Updating User's Password

To change a user's password, use the `PasswordChangeRequest` resource, which handles secure password updates.

```
kubectl create  -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: PasswordChangeRequest
spec:
  userID: "testuser"
  currentPassword: "Pass1234567!"
  newPassword: "NewPass1234567!"
EOF
```

#### Listing Users

List all `User` resources in the cluster:

```
kubectl get users
NAME           AGE
testuser       3m54s
user-4n5ws     12m
```

#### Viewing a User

View a specific `User` resource by name:

```
kubectl get user testuser
NAME           AGE
testuser       3m54s
```

#### Deleting a User

Deleting a user automatically deletes the corresponding password `Secret`.

```
kubectl delete user testuser
user.management.cattle.io "testuser" deleted
```

#### Get a Current User's Information

A client uses the `SelfUser` resource to retrieve information about the currently authenticated user without knowing their ID. The user ID is returned in the `.status.userID` field.

```
kubectl create -o jsonpath='{.status.userID}'  -f -<<EOF
```

```
apiVersion: ext.cattle.io/v1
kind: SelfUser
EOF
testuser
```

### Refreshing a User's Group Membership

Updates to user group memberships are triggered by the `GroupMembershipRefreshRequest` resource.



Group membership is only supported for external authentication providers.

#### For a Single User

```
kubectl create -o jsonpath='{.status}'  -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: GroupMembershipRefreshRequest
spec:
  userId: testuser
EOF
{
    "conditions": [
        {
            "lastTransitionTime": "2025-11-10T12:01:03Z",
            "message": "",
            "reason": "",
            "status": "True",
            "type": "UserRefreshInitiated"
        }
    ],
    "summary": "Completed"
}
```

#### For All Users

```
kubectl create -o jsonpath='{.status}'  -f -<<EOF
apiVersion: ext.cattle.io/v1
kind: GroupMembershipRefreshRequest
spec:
   userId: "*"
EOF
{
    "conditions": [
        {
            "lastTransitionTime": "2025-11-10T12:01:59Z",
```

```
            "message": "",
            "reason": "",
            "status": "True",
            "type": "UserRefreshInitiated"
        }
    ],
    "summary": "Completed"
}
```


## API Reference



At this time, not all Rancher resources are available through the Rancher Kubernetes API.

```
<div class="api-doc">
    <redoc id='redoc-container'></redoc>
    <script
src="https://cdn.redoc.ly/redoc/v2.x/bundles/redoc.standalone.js"></script>
    <script>
        Redoc.init('../_attachments/swagger.json',
        {scrollYOffset: '.toolbar'},
        document.getElementById('redoc-container'))
    </script>
</div>
```

## Using API Tokens



*Deprecation of v3 API tokens*

Rancher v2.13 introduced a new token resource in the `ext.cattle.io` API group to serve as the public API for tokens, which was only accessible through `kubectl`.

Starting with Rancher v2.14.1:

- Legacy v3 API tokens (`tokens.management.cattle.io`) are being phased out and scheduled for removal in a future release. If you currently rely on legacy tokens for automation or API access, transition to the new `tokens.ext.cattle.io` tokens to ensure continued compatibility.
- The Rancher dashboard provides basic support for `tokens.ext.cattle.io` (create, view or list, and delete).

Rancher v2.8.0 introduced the Rancher Kubernetes API which can be used to manage Rancher resources through `kubectl`. This page covers information on API tokens used with the Rancher CLI, kubeconfig files, Terraform and the v3 API browser.

By default, some cluster-level API tokens are generated with infinite time-to-live (`ttl=0`). In other words, API tokens with `ttl=0` never expire unless you invalidate them. Tokens are not invalidated by changing a password.

You can deactivate API tokens by deleting them or by deactivating the user account.

## Deleting Tokens

To delete a token:

1. Go to the list of all tokens in the Rancher API view at `https://<Rancher-Server-IP>/v3/tokens`.
2. Access the token you want to delete by its ID. For example, `https://<Rancher-Server-IP>/v3/tokens/kubectl-shell-user-vqkqt`
3. Click **Delete**.

The following is a complete list of tokens generated with `ttl=0`:

| Token | Description |
|---|---|
| `kubectl-shell-*` | Access to `kubectl` shell in the browser |
| `agent-*` | Token for agent deployment |
| `compose-token-*` | Token for compose |
| `helm-token-*` | Token for Helm chart deployment |
| `drain-node-*` | Token for drain (Rancher uses `kubectl` for drain because there is no native Kubernetes API). |

## Setting TTL on Kubeconfig Tokens

Admins can set a global time-to-live (TTL) on Kubeconfig tokens. Changing the default kubeconfig TTL can be done by navigating to global settings and setting `kubeconfig-default-token-ttl-minutes` to the desired duration in minutes. As of Rancher v2.8, the default value of `kubeconfig-default-token-ttl-minutes` is `43200`, which means that tokens expire in 30 days.



This setting is used by all kubeconfig tokens except those created by the CLI to generate kubeconfig tokens.

## Disable Tokens in Generated Kubeconfigs

Set the `kubeconfig-generate-token` setting to `false`. This setting instructs Rancher to no longer automatically generate a token when a user clicks on download a kubeconfig file. When this setting is deactivated, a generated kubeconfig references the Rancher CLI to retrieve a short-lived token for the cluster. When this kubeconfig is used in a client, such as `kubectl`, the Rancher CLI needs to be installed to complete the log in request.

## Token Hashing

You can enable token hashing, where tokens undergo a one-way hash using the SHA256 algorithm. This is a non-reversible process: once enabled, this feature cannot be disabled. You should first evaluate this setting in a test environment, and/or take backups before enabling.

This feature affects all tokens which include, but are not limited to, the following:

- Kubeconfig tokens
- Bearer tokens API keys/calls

- Tokens used by internal operations

## Token Settings

These global settings affect Rancher token behavior.

| Setting | Description |
|---|---|
| `auth-user-session-ttl-minutes` | TTL in minutes on a user auth session token. |
| `auth-user-session-idle-ttl-minutes` | TTL in minutes on a user auth session token, without user activity. |
| `kubeconfig-default-token-ttl-minutes` | Default TTL applied to all kubeconfig tokens except for tokens generated by Rancher CLI. |
| `auth-token-max-ttl-minutes` | Max TTL for all tokens except those controlled by `auth-user-session-ttl-minutes`. |
| `kubeconfig-generate-token` | If true, automatically generate tokens when a user downloads a kubeconfig. |

#### auth-user-session-ttl-minutes

Time to live (TTL) duration in minutes, used to determine when a user auth session token expires. When expired, the user must log in and obtain a new token. This setting is not affected by `auth-token-max-ttl-minutes`. Session tokens are created when a user logs into Rancher.

#### auth-user-session-idle-ttl-minutes

Time to live (TTL) without user activity for login sessions tokens, in minutes. By default, `auth-user-session-idle-ttl-minutes` is set to the same value as `auth-user-session-ttl-minutes` (for backward compatibility). It must never exceed the value of `auth-user-session-ttl-minutes`.

#### kubeconfig-default-token-ttl-minutes

Time to live (TTL) duration in minutes, used to determine when a kubeconfig token expires. When the token is expired, the API rejects the token. This setting can't be larger than `auth-token-max-ttl-minutes`. This setting applies to tokens generated in a requested kubeconfig file, except for tokens generated by Rancher CLI. As of Rancher v2.8, the default duration is `43200`, which means that tokens expire in 30 days.

#### auth-token-max-ttl-minutes

Maximum Time to Live (TTL) in minutes allowed for auth tokens. If a user attempts to create a token with a TTL greater than `auth-token-max-ttl-minutes`, Rancher sets the token TTL to the value of `auth-token-max-ttl-minutes`. Applies to all kubeconfig tokens and API tokens. As of Rancher v2.8, the default duration is `129600`, which means that tokens expire in 90 days.

#### kubeconfig-generate-token

When true, kubeconfigs requested through the UI contain a valid token. When false, kubeconfigs contain a command that uses the Rancher CLI to prompt the user to log in. The CLI then retrieves and caches a token for the user. cli/kubectl.adoc#_authentication_with_kubectl_and_kubeconfig_tokens_with_ttl[The CLI then retrieves and caches a token for the user].

# Extension API Server

Rancher extends Kubernetes with additional APIs by registering an extension API server using the Kubernetes API Aggregation Layer [https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/apiserver-aggregation/].

## Aggregation Layer is Required

The API aggregation layer must be configured on the local Kubernetes cluster for the `v1.ext.cattle.io APIService` to work correctly. If the `APIService` does not receive a registration request after the Rancher server starts, the pod crashes with a log entry indicating the error. If your pods are consistently failing to detect registration despite having a correctly configured cluster, you can increase the timeout by setting the `.Values.aggregationRegistrationTimeout` in Helm.

In this version of Rancher, the Kubernetes versions supported by RKE2/K3s have the aggregation layer configured and enabled by default. However, if you suspect that your cluster configuration is incorrect, refer to the Kubernetes Aggregation Layer documentation [https://kubernetes.io/docs/tasks/extend-kubernetes/configure-aggregation-layer/] for information on configuring the aggregation layer.



If the underlying Kubernetes distribution does not support the aggregation layer, you must migrate to a Kubernetes distribution that does before upgrading.

# Previous v3 SUSE Rancher Prime API Guide



The v3 API is not supported and backwards compatibility is not guaranteed.

Rancher v2.8.0 introduced the Rancher Kubernetes API (RK-API), which is Rancher's supported API. This page describes the v3 API. For more information on RK-API, see the RK-API quickstart and reference guide.

## How to Use the API

The previous v3 API has its own user interface accessible from a web browser. This is an easy way to see resources, perform actions, and see the equivalent `curl` or HTTP request & response. To access it:

1. Click your user avatar in the upper right corner.
2. Click **Account & API Keys**.
3. Under the **API Keys** section, find the **API Endpoint** field and click the link. The link looks something like `https://<rancher_fqdn>/v3`, where `<RANCHER_FQDN>` is the fully qualified domain name of your Rancher deployment.

## Authentication

API requests must include authentication information. Authentication is done with HTTP basic authentication using API keys. API keys can create new clusters and have access to multiple clusters via `/v3/clusters/`. Cluster and project roles apply to these keys and restrict what clusters and projects the account can see and what actions they can take.

By default, certain cluster-level API tokens are generated with infinite time-to-live (`ttl=0`). In other words, API tokens with `ttl=0` never expire unless you invalidate them. For details on how to invalidate them, refer

to the API tokens page.

## Making Requests

The API is generally RESTful but has several features to make the definition of everything discoverable by a client so that generic clients can be written instead of having to write specific code for every type of resource. For detailed info about the generic API spec, see further documentation [https://github.com/rancher/api-spec/blob/master/specification.md].

- Every type has a Schema which describes:
  - The URL to get to the collection of this type of resource.
  - Every field the resource can have, along with their type, basic validation rules, whether they are required or optional, etc.
  - Every action that is possible on this type of resource, with their inputs and outputs (also as schemas).
  - Every field that allows filtering.
  - What HTTP verb methods are available for the collection itself, or for individual resources in the collection.

The design allows you to load just the list of schemas and access everything about the API. The UI for the API contains no code specific to Rancher itself. The URL to get Schemas is sent in every HTTP response as a `X-Api-Schemas` header. From there you can follow the `collection` link on each schema to know where to list resources, and follow other `links` inside of the returned resources to get any other information.

In practice, you may just want to construct URL strings. We highly suggest limiting this to the top-level to list a collection (`/v3/<type>`) or get a specific resource (`/v3/<type>/<id>`). Anything deeper than that is subject to change in future releases.

Resources have relationships between each other called links. Each resource includes a map of `links` with the name of the link and the URL where you can retrieve that information. Again, you should `GET` the resource and then follow the URL in the `links` map, not construct these strings yourself.

Most resources have actions, which do something or change the state of the resource. To use them, send a HTTP `POST` to the URL in the `actions` map of the action you want. Certain actions require input or produce output. See the individual documentation for each type or the schemas for specific information.

To edit a resource, send a HTTP `PUT` to the `links.update` link on the resource with the fields that you want to change. If the link is missing then you don't have permission to update the resource. Unknown fields and ones that are not editable are ignored.

To delete a resource, send a HTTP `DELETE` to the `links.remove` link on the resource. If the link is missing then you don't have permission to update the resource.

To create a new resource, HTTP `POST` to the collection URL in the schema (which is `/v3/<type>`).

## Filtering

Most collections can be filtered on the server-side by common fields using HTTP query parameters. The `filters` map shows you what fields can be filtered on and what the filtered values were for the request you made. The API UI has controls to setup filtering and show you the appropriate request. For simple "equals" matches it's just `field=value`. Modifiers can be added to the field name, for example, `field_gt=42` for

"field is greater than 42." See the API spec [https://github.com/rancher/api-spec/blob/master/specification.md#filtering] for full details.

## Sorting

Most collections can be sorted on the server-side by common fields using HTTP query parameters. The `sortLinks` map shows you what sorts are available, along with the URL to get the collection sorted by that. It also includes info about what the current response was sorted by, if specified.

## Pagination

API responses are paginated with a limit of 100 resources per page by default. This can be changed with the `limit` query parameter, up to a maximum of 1000, for example, `/v3/pods?limit=1000`. The `pagination` map in collection responses tells you whether or not you have the full result set and has a link to the next page if you do not.

## Capturing v3 API Calls

You can use browser developer tools to capture how the v3 API is called. For example, you could follow these steps to use the Chrome developer tools to get the API call for provisioning a Rancher Kubernetes distribution cluster:

1. In the Rancher UI, go to **Cluster Management** and click **Create.**
2. Click one of the cluster types. This example uses Digital Ocean.
3. Fill out the form with a cluster name and node template, but don't click **Create**.
4. You need to open the developer tools before the cluster creation to see the API call being recorded. To open the tools, right-click the Rancher UI and click **Inspect.**
5. In the developer tools, click the **Network** tab.
6. On the **Network** tab, make sure **Fetch/XHR** is selected.
7. In the Rancher UI, click **Create**. In the developer tools, you should see a new network request with the name `cluster?_replace=true`.
8. Right-click `cluster?_replace=true` and click menu:Copy[Copy as cURL.]
9. Paste the result into any text editor. You can see the POST request, including the URL it was sent to, all headers, and the full body of the request. This command can be used to create a cluster from the command line. Note: the request should be stored in a safe place because it contains credentials.

#### Enable View in API

You can also view captured v3 API calls for your respective clusters and resources. This feature is not enabled by default. To enable it:

1. Click your **User Tile** in the top right corner of the UI and select **Preferences** from the drop-down menu.
2. Under the **Advanced Features** section, click **Enable "View in API"**

Once checked, the **View in API** link is displayed under the ⬔ sub-menu on resource pages in the UI.

# Troubleshooting

## General Troubleshooting

This section contains information to help you troubleshoot issues when using Rancher.

- Kubernetes components

  If you need help troubleshooting core Kubernetes cluster components like:

  - `etcd`
  - `kube-apiserver`
  - `kube-controller-manager`
  - `kube-scheduler`
  - `kubelet`
  - `kube-proxy`
  - `nginx-proxy`

- Kubernetes resources

  Options for troubleshooting Kubernetes resources like Nodes, Ingress Controller and Rancher Agents are described in this section.

- Networking

  Steps to troubleshoot networking issues can be found here.

- DNS

  When you experience name resolution issues in your cluster.

- Troubleshooting Rancher installed on Kubernetes

  If you experience issues with your Rancher server installed on Kubernetes

- Logging

  Read more about what log levels can be configured and how to configure a log level.

- User ID Tracking in Audit Logs

  Read more about how a Rancher Admin can trace an event from the Rancher audit logs and into the Kubernetes audit logs using the external Identity Provider username.

- Expired Webhook Certificates

  Read more about how to rotate a Rancher webhook certificate secret after it expires on an annual basis.

# Kubernetes Components

The commands and steps listed in this section apply to the core Kubernetes components on Rancher Launched Kubernetes clusters.

This section includes troubleshooting tips in the following categories:

- Troubleshooting etcd Nodes
- Troubleshooting Controlplane Nodes
- Troubleshooting nginx-proxy Nodes
- Troubleshooting Worker Nodes and Generic Components

## Kubernetes Component Diagram

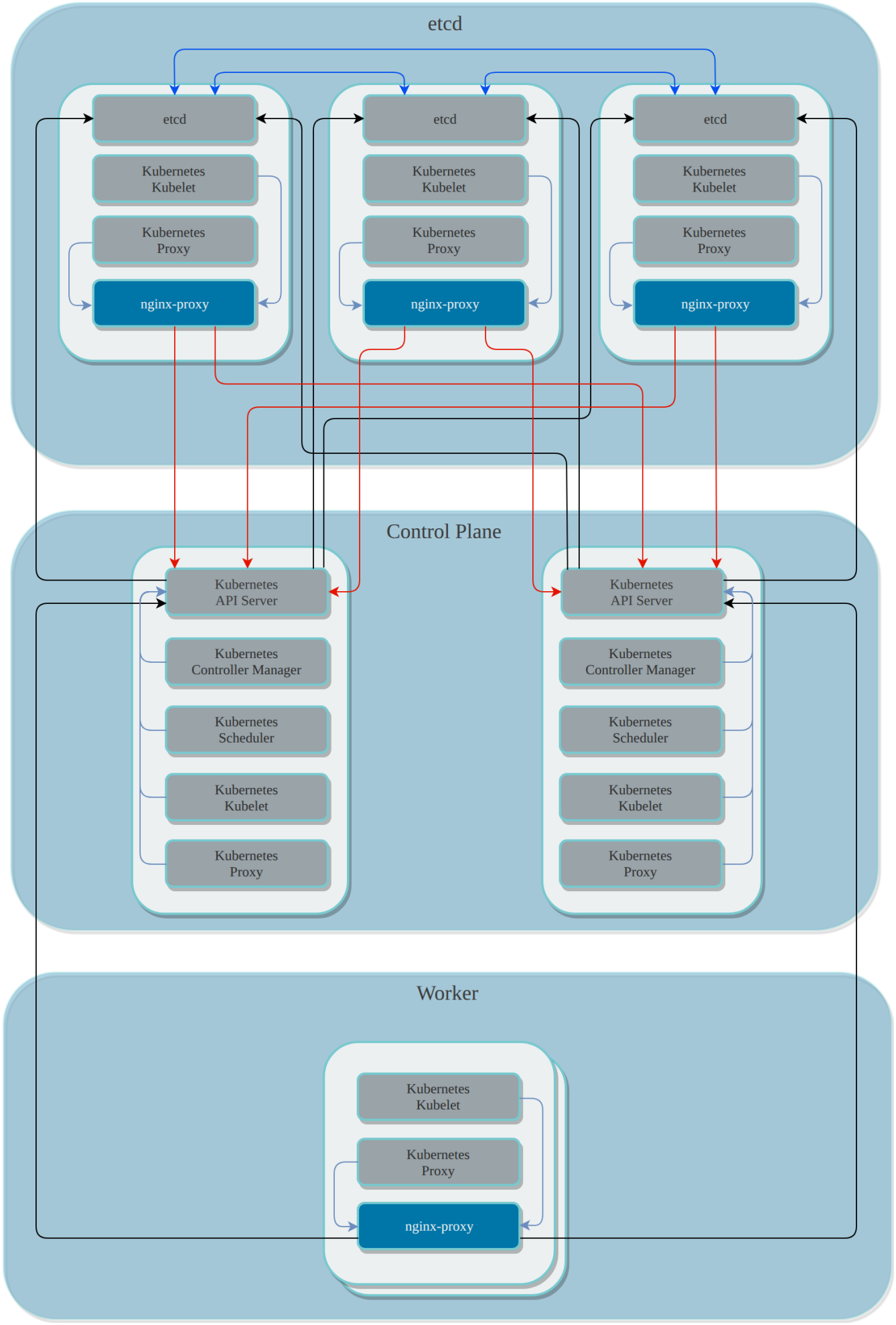



*Figure 27. Lines show the traffic flow between components. Colors are used purely for visual aid*

## Troubleshooting Controlplane Nodes

### Prerequisites

As RKE2 and K3s rely on `containerd` as the container runtime, `crictl` replaces Docker for container management. Before proceeding with the troubleshooting commands, configure your environment by exporting the following variables:

#### RKE2

```
export PATH=$PATH:/var/lib/rancher/rke2/bin/
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
```

#### K3s

```
export PATH=$PATH:/usr/local/bin
export CRI_CONFIG_FILE=/var/lib/rancher/k3s/agent/etc/crictl.yaml
```

### Check if the Controlplane Components are Running

**RKE2**: There are three specific containers launched on nodes with the `controlplane` role:

- `kube-apiserver`
- `kube-controller-manager`
- `kube-scheduler`

The containers should have state **Running**. You can check this using `crictl`:

```
crictl ps | grep -E 'kube-apiserver|kube-controller-manager|kube-
scheduler'
```

Example output:

```
CONTAINER           IMAGE               CREATED             STATE
NAME                                ATTEMPT             POD ID
POD                                 NAMESPACE
deb8a96948594       138b1e685e151       11 days ago         Running
kube-controller-manager         0                   0996426295dc5
kube-controller-manager                 kube-system
f5abb4c7846e4       138b1e685e151       11 days ago         Running
kube-scheduler                  0                   80cd9f30af0be
kube-scheduler                          kube-system
ecd8a6991c22a       138b1e685e151       11 days ago         Running
kube-apiserver                  0                   58e042fabe78c
```

```
kube-apiserver                          kube-system
```

**K3s**: These components run as embedded processes within the K3s service. They do not run as separate containers, so their status is tied to the `k3s` systemd service:

```
systemctl status k3s
```

#### Controlplane Logging



If you added multiple nodes with the `controlplane` role, both `kube-controller-manager` and `kube-scheduler` use a leader election process to determine the leader. Only the current leader will log the performed actions. See Kubernetes leader election how to retrieve the current leader.

The logs can contain information on what the problem could be.

**RKE2**:

```
crictl logs $(crictl ps --name kube-apiserver -q)
crictl logs $(crictl ps --name kube-controller-manager -q)
crictl logs $(crictl ps --name kube-scheduler -q)
```

**K3s**:

```
journalctl -u k3s | grep -i "kube-apiserver"
journalctl -u k3s | grep -i "kube-controller-manager"
journalctl -u k3s | grep -i "kube-scheduler"
```

#### RKE2/K3s Server Logging

If Rancher provisions an RKE2 or K3s cluster that can't communicate with Rancher, you can run this command on a server node in the downstream cluster to get the server logs:

**RKE2**:

```
journalctl -u rke2-server -f
```

**K3s**:

```
journalctl -u k3s -f
```

## Troubleshooting etcd Nodes

This section contains commands and tips for troubleshooting nodes with the `etcd` role in RKE2 and K3s clusters.

### Prerequisites

As RKE2 and K3s rely on `containerd` as the container runtime, `crictl` replaces Docker for container management. Before proceeding with the troubleshooting commands, configure your environment by exporting the following variables:

#### RKE2

```
export PATH=$PATH:/var/lib/rancher/rke2/bin/
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
etcdcontainer=$(crictl ps --name etcd --quiet)
```

#### K3s



K3s does not include `etcdctl` in the system PATH. If you need to perform etcd troubleshooting on a K3s cluster, you may need to install it or locate it within the K3s data directory.

```
export PATH=$PATH:/usr/local/bin
export CRI_CONFIG_FILE=/var/lib/rancher/k3s/agent/etc/crictl.yaml
```

### Checking if the etcd Container is Running

**RKE2**: The container for etcd should be in the **Running** state.

```
crictl ps --name etcd
```

Example output:

```
CONTAINER           IMAGE               CREATED             STATE
NAME                ATTEMPT             POD ID              POD
NAMESPACE
fle289d202ed0       lladl6872a9cf       58 minutes ago      Running
etcd                0                   7b56aab8204ea       etcd-
clusterl     kube-system
```

**K3s**: Etcd runs as an embedded process in the K3s service. Check the service status:

```
systemctl status k3s
```

#### etcd Logging

The logs can contain information on what the problem could be.

**RKE2**:

```
crictl logs $etcdcontainer
```

**K3s**:

```
journalctl -u k3s | grep -i etcd
```

| Log | Explanation |
|---|---|
| `health check for peer xxx could not connect: dial tcp IP:2380: getsockopt: connection refused` | A connection to the address shown on port 2380 cannot be established. Check if the etcd container is running on the host with the address shown. |
| `xxx is starting a new election at term x` | The etcd cluster has lost its quorum and is trying to establish a new leader. This can happen when the majority of the nodes running etcd go down/unreachable. |
| `connection error: desc = "transport: Error while dialing dial tcp 0.0.0.0:2379: i/o timeout"; Reconnecting to {0.0.0.0:2379 0 <nil>}` | The host firewall is preventing network communication. |
| `rafthttp: request cluster ID mismatch` | The node with the etcd instance logging `rafthttp: request cluster ID mismatch` is trying to join a cluster that has already been formed with another peer. The node should be removed from the cluster, and re-added. |
| `rafthttp: failed to find member` | The cluster state (`/var/lib/etcd`) contains wrong information to join the cluster. The node should be removed from the cluster, the state directory should be cleaned and the node should be re-added. |

#### etcd Cluster and Connectivity Checks

The address where etcd is listening depends on the address configuration of the host etcd is running on. If an internal address is configured for the host etcd is running on, the endpoint for `etcdctl` needs to be specified explicitly. If any of the commands respond with `Error: context deadline exceeded`, the etcd instance is unhealthy (either quorum is lost or the instance is not correctly joined in the cluster)

##### Check etcd Members on all Nodes

Output should contain all the nodes with the `etcd` role and the output should be identical on all nodes.

**RKE2**:

Run the command inside the etcd container.

```
crictl exec $etcdcontainer etcdctl member list \
  --cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt \
```

```
  --key /var/lib/rancher/rke2/server/tls/etcd/server-client.key \
  --cacert /var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl member list \
  --cert /var/lib/rancher/k3s/server/tls/etcd/server-client.crt \
  --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key \
  --cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output:

```
1c424074df86e854, started, cluster-node1-f289ac71, https://IP:2380,
https://IP:2379, false
45c68c44c5a792ff, started, cluster-node2-67e3cf6f, https://IP:2380,
https://IP:2379, false
7c584f77c5180258, started, cluster-node3-e976bc00, https://IP:2380,
https://IP:2379, false
```

#### Check Endpoint Status

The values for `RAFT TERM` should be equal and `RAFT INDEX` should be not be too far apart from each other.

**RKE2**:

```
crictl exec $etcdcontainer etcdctl endpoint status --write-out table
--endpoints=$(crictl exec $etcdcontainer etcdctl member list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl endpoint status --write-out table --endpoints=$(etcdctl member
list --cert /var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
```

```
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output:

```
+----------------+------------------+---------+---------+-----------
+-----------+------------+
| ENDPOINT        |        ID        | VERSION | DB SIZE | IS LEADER |
RAFT TERM | RAFT INDEX |
+----------------+------------------+---------+---------+-----------
+-----------+------------+
| https://IP:2379 | 333ef673fc4add56 |  3.6.7  |   24 MB |     false |
72 |      66887 |
| https://IP:2379 | 5feed52d940ce4cf |  3.6.7  |   24 MB |      true |
72 |      66887 |
| https://IP:2379 | db6b3bdb559a848d |  3.6.7  |   25 MB |     false |
72 |      66887 |
+----------------+------------------+---------+---------+-----------
+-----------+------------+
```

##### Check Endpoint Health

**RKE2**:

```
crictl exec $etcdcontainer etcdctl endpoint health --endpoints=$(crictl
exec $etcdcontainer etcdctl member list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl endpoint health --endpoints=$(etcdctl member list --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
```

```
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output:

```
https://IP:2379 is healthy: successfully committed proposal: took =
2.113189ms
https://IP:2379 is healthy: successfully committed proposal: took =
2.649963ms
https://IP:2379 is healthy: successfully committed proposal: took =
2.451201ms
```

#### Check Connectivity on etcd Ports

In modern versions of Kubernetes, the etcd database (versions 3.5 and newer) introduced significant architectural changes regarding network traffic handling. Previously, etcd permitted standard HTTP REST requests on its primary client port (2379). However, to enhance performance and security, etcd 3.5+ strictly enforces the gRPC protocol on this port.

If you attempt to use standard HTTP tools like `curl` to test connectivity on port 2379, the etcd server will automatically terminate the connection or return an error. This behavior often leads administrators to misinterpret the result as a closed port or a node failure.

Since standard HTTP clients can no longer probe the primary etcd ports, the transport layer must be utilized for network troubleshooting. Using `openssl s_client` instead of `curl` bypasses the gRPC application requirement, allowing the raw TCP and TLS handshake to be tested directly.

These script isolate the network and security infrastructure from the database application. A successful `Verify return code: 0 (ok)` explicitly confirms four critical infrastructure components:

- **Network Path:** Routing is functional, and firewalls permit traffic on TCP port 2379 or 2380.
- **Process Availability:** The etcd service is running and actively listening on the designated port.
- **Certificate Validity:** The TLS certificates are active, correctly formatted, and have not expired.
- **Mutual Authentication (mTLS):** The node successfully authenticates against the cluster's specific Certificate Authority (CA).

**How these tests differ from the `etcdctl endpoint health` test**:

If `etcdctl endpoint health` test is failing, run these Connectivity Ports test scripts. If the scripts succeed, your network and certificates are intact, and the issue is likely confined to the etcd database itself. If these scripts fail, the issue is related to a firewall/network restriction, or certificate expiration.

#### Port TCP/2379

**RKE2**:

```
for endpoint in $(crictl exec $etcdcontainer etcdctl member list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
```

```
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | cut -d, -f5); do
  echo "Validating connection to ${endpoint} (Client)";
  echo | openssl s_client -connect ${endpoint#https://} \
    -CAfile /var/lib/rancher/rke2/server/tls/etcd/server-ca.crt \
    -cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt \
    -key /var/lib/rancher/rke2/server/tls/etcd/server-client.key
2>/dev/null | grep -E 'Verify return code' || echo "Connection
Failed/Timeout"
done
```

**K3s**:

```
for endpoint in $(etcdctl member list --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | cut -d, -f5); do
  echo "Validating connection to ${endpoint} (Client)";
  echo | openssl s_client -connect ${endpoint#https://} \
    -CAfile /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt \
    -cert /var/lib/rancher/k3s/server/tls/etcd/server-client.crt \
    -key /var/lib/rancher/k3s/server/tls/etcd/server-client.key
2>/dev/null | grep -E 'Verify return code' || echo "Connection
Failed/Timeout"
done
```

Example output:

```
Validating connection to https://IP:2379/health (Client)
Verify return code: 0 (ok)
Validating connection to https://IP:2379/health (Client)
Verify return code: 0 (ok)
Validating connection to https://IP:2379/health (Client)
Verify return code: 0 (ok)
```

#### Port TCP/2380

**RKE2**:

```
for endpoint in $(crictl exec $etcdcontainer etcdctl member list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | cut -d, -f4); do
  echo "Validating connection to ${endpoint} (Peer)";
  echo | openssl s_client -connect ${endpoint#https://} \
```

```
    -CAfile /var/lib/rancher/rke2/server/tls/etcd/peer-ca.crt \
    -cert /var/lib/rancher/rke2/server/tls/etcd/peer-server-client.crt \
    -key /var/lib/rancher/rke2/server/tls/etcd/peer-server-client.key
2>/dev/null | grep -E 'Verify return code' || echo "Connection
Failed/Timeout"
done
```

**K3s**:

```
for endpoint in $(etcdctl member list --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | cut -d, -f4); do
  echo "Validating connection to ${endpoint} (Peer)";
  echo | openssl s_client -connect ${endpoint#https://} \
    -CAfile /var/lib/rancher/k3s/server/tls/etcd/peer-ca.crt \
    -cert /var/lib/rancher/k3s/server/tls/etcd/peer-server-client.crt \
    -key /var/lib/rancher/k3s/server/tls/etcd/peer-server-client.key
2>/dev/null | grep -E 'Verify return code' || echo "Connection
Failed/Timeout"
done
```

Example output:

```
Validating connection to https://IP:2380/version (Peer)
Verify return code: 0 (ok)
Validating connection to https://IP:2380/version (Peer)
Verify return code: 0 (ok)
Validating connection to https://IP:2380/version (Peer)
Verify return code: 0 (ok)
```

#### etcd Alarms

etcd will trigger alarms, for instance when it runs out of space.

**RKE2**:

```
crictl exec $etcdcontainer etcdctl alarm list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl alarm list --cert /var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output when NOSPACE alarm is triggered:

```
memberID:x alarm:NOSPACE
memberID:x alarm:NOSPACE
memberID:x alarm:NOSPACE
```

#### etcd Space Errors

Related error messages are `etcdserver: mvcc: database space exceeded` or `applying raft message exceeded backend quota`. Alarm `NOSPACE` will be triggered.

Resolutions:

- Compact the Keyspace
- Defrag All etcd Members
- Check Endpoint Status
- Disarm Alarm

##### Compact the Keyspace

**RKE2**:

```
rev=$(crictl exec $etcdcontainer etcdctl endpoint status --write-out json --cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key /var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert /var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | egrep -o '"revision":[0-9]*' | egrep -o '[0-9]*' | head -1)
crictl exec $etcdcontainer etcdctl compact "$rev" --cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key /var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert /var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
rev=$(etcdctl endpoint status --write-out json --cert /var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | egrep -o '"revision":[0-9]*' | egrep -o '[0-9]*' | head -1)
etcdctl compact "$rev" --cert
```

```
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output:

```
compacted revision xxx
```

#### Defrag All etcd Members

**RKE2**:

```
crictl exec $etcdcontainer etcdctl defrag --endpoints=$(crictl exec
$etcdcontainer etcdctl member list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl defrag --endpoints=$(etcdctl member list --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | cut -d, -f5 | sed
-e 's/ //g' | paste -sd ',') --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

Example output:

```
Finished defragmenting etcd member[https://IP:2379]. took xx.xxxxxxms
Finished defragmenting etcd member[https://IP:2379]. took xx.xxxxxxms
Finished defragmenting etcd member[https://IP:2379]. took xx.xxxxxxms
```

#### Disarm Alarm

After verifying that the DB size went down after compaction and defragmenting, the alarm needs to be disarmed for etcd to allow writes again.

**RKE2**:

```
crictl exec $etcdcontainer etcdctl alarm list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
crictl exec $etcdcontainer etcdctl alarm disarm --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
crictl exec $etcdcontainer etcdctl alarm list --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl alarm list --cert /var/lib/rancher/k3s/server/tls/etcd/server-
client.crt --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key
--cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
etcdctl alarm disarm --cert /var/lib/rancher/k3s/server/tls/etcd/server-
client.crt --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key
--cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
etcdctl alarm list --cert /var/lib/rancher/k3s/server/tls/etcd/server-
client.crt --key /var/lib/rancher/k3s/server/tls/etcd/server-client.key
--cacert /var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

### Configure Log Level



You can no longer dynamically change the log level in etcd v3.5 or later.

#### etcd v3.5 And Later

To configure the log level for etcd, edit the cluster configuration YAML:

```
services:
  etcd:
    extra_args:
      log-level: "debug"
```

After modifying the configuration, restart the service (`systemctl restart rke2-server` or `systemctl restart k3s`) if you are configuring a stand-alone cluster.

#### etcd Content

If you want to investigate the contents of your etcd, you can either watch streaming events or you can query etcd directly, see below for examples.

##### Watch Streaming Events

**RKE2**:

```
crictl exec $etcdcontainer etcdctl watch --prefix /registry --cert
/var/lib/rancher/rke2/server/tls/etcd/server-client.crt --key
/var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl watch --prefix /registry --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

If you only want to see the affected keys (and not the binary data), you can append `| grep -a ^/registry` to the command to filter for keys only.

##### Query etcd Directly

**RKE2**:

```
crictl exec $etcdcontainer etcdctl get /registry --prefix=true --keys
-only --cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt
--key /var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt
```

**K3s**:

```
etcdctl get /registry --prefix=true --keys-only --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt
```

You can process the data to get a summary of count per key, using the command below:

**RKE2**:

```
crictl exec $etcdcontainer etcdctl get /registry --prefix=true --keys
```

```
-only --cert /var/lib/rancher/rke2/server/tls/etcd/server-client.crt
--key /var/lib/rancher/rke2/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/rke2/server/tls/etcd/server-ca.crt | grep -v ^$ | awk
-F'/' '{ if ($3 ~ /cattle.io/) {h[$3"/"$4]++} else { h[$3]++ }} END {
for(k in h) print h[k], k }' | sort -nr
```

**K3s**:

```
etcdctl get /registry --prefix=true --keys-only --cert
/var/lib/rancher/k3s/server/tls/etcd/server-client.crt --key
/var/lib/rancher/k3s/server/tls/etcd/server-client.key --cacert
/var/lib/rancher/k3s/server/tls/etcd/server-ca.crt | grep -v ^$ | awk -
F'/' '{ if ($3 ~ /cattle.io/) {h[$3"/"$4]++} else { h[$3]++ }} END {
for(k in h) print h[k], k }' | sort -nr
```

#### Replacing Unhealthy etcd Nodes

When a node in your etcd cluster becomes unhealthy, the recommended approach is to fix or remove the failed or unhealthy node before adding a new etcd node to the cluster.

## Troubleshooting nginx-proxy



The `nginx-proxy` container is an RKE1-specific component. If you are using RKE2 or K3s, this container is not deployed, as load balancing to the API servers is handled internally by a client-side load balancer within the agent process itself.

Additionally, please note that RKE1 has reached its End of Life (EOL) [https://support.scc.suse.com/s/kb/RKE-EOL-what-when-why?]. Therefore, the information on this page is considered deprecated.

The `nginx-proxy` container is deployed on every node that does not have the `controlplane` role. It provides access to all the nodes with the `controlplane` role by dynamically generating the NGINX configuration based on available nodes with the `controlplane` role.

#### Check if the Container is Running

The container is called `nginx-proxy` and should have status `Up`. The duration shown after `Up` is the time the container has been running.

```
docker ps -a -f=name=nginx-proxy
```

Example output:

```
docker ps -a -f=name=nginx-proxy
CONTAINER ID        IMAGE                         COMMAND
CREATED             STATUS              PORTS                NAMES
```

```
c3e933687c0e        rancher/rke-tools:v0.1.15   "nginx-proxy CP_HO..."
3 hours ago         Up 3 hours                                  nginx-proxy
```

#### Check Generated NGINX Configuration

The generated configuration should include the IP addresses of the nodes with the `controlplane` role. The configuration can be checked using the following command:

```
docker exec nginx-proxy cat /etc/nginx/nginx.conf
```

Example output:

```
error_log stderr notice;

worker_processes auto;
events {
  multi_accept on;
  use epoll;
  worker_connections 1024;
}

stream {
        upstream kube_apiserver {

            server ip_of_controlplane_node1:6443;

            server ip_of_controlplane_node2:6443;

        }

        server {
            listen        6443;
            proxy_pass    kube_apiserver;
            proxy_timeout 30;
            proxy_connect_timeout 2s;

        }

}
```

#### nginx-proxy Container Logging

The logging of the containers can contain information on what the problem could be.

```
docker logs nginx-proxy
```

## Troubleshooting Worker Nodes and Generic Components

This section applies to every node as it includes components that run on nodes with any role.

### Prerequisites

Since RKE2 and K3s utilize `containerd` as the container runtime, `crictl` serves as the primary tool for container management, replacing the Docker CLI. To allow `crictl` to communicate with `containerd`, you must configure your environment by exporting the following variables:

#### RKE2

```
export PATH=$PATH:/var/lib/rancher/rke2/bin/
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
```

#### K3s

```
export PATH=$PATH:/usr/local/bin
export CRI_CONFIG_FILE=/var/lib/rancher/k3s/agent/etc/crictl.yaml
```

### Check if the Components are Running

There are two specific components launched on nodes with the `worker` role:

- `kubelet`
- `kube-proxy`

#### RKE2

The `kubelet` runs natively as part of the `rke2-agent` (or `rke2-server`) systemd process, while `kube-proxy` runs as a Static Pod managed by `containerd`.

Check the status of the `kubelet` via the agent service:

```
systemctl status rke2-agent
```



If you are checking a controlplane node, use `systemctl status rke2-server` instead.

Check the status of `kube-proxy` using `crictl`:

```
crictl ps --name kube-proxy
```

Example output:

```
CONTAINER           IMAGE
CREATED             STATE               NAME                ATTEMPT
POD ID
26c7159abbcc        rancher/hardened-kubernetes:v1.28.8-rke2r1-
build20240404   3 hours ago         Running             kube-proxy
0                   1a2b3c4d5e6f7
```

#### K3s

Both `kubelet` and `kube-proxy` run as embedded processes inside the `k3s-agent` (or `k3s` server) systemd service. There are no separate containers for them.

Check their status by checking the K3s service:

```
systemctl status k3s-agent
```



If you are checking a controlplane node, use `systemctl status k3s` instead.

#### Component Logging

The logging of the components can contain information on what the problem could be.

#### RKE2

```
# kubelet logs are part of the systemd service
journalctl -u rke2-agent -f | grep -i "kubelet"

# kube-proxy logs are retrieved from containerd
crictl logs $(crictl ps --name kube-proxy -q)
```

#### K3s

```
# Both components log to the systemd service
journalctl -u k3s-agent -f | grep -iE "kubelet|kube-proxy"
```

### User ID Tracking in Audit Logs

The following audit logs are used in Rancher to track events occurring on the local and downstream clusters:

- Kubernetes Audit Logs [https://rancher.com/docs/rke/latest/en/config-options/audit-log/]
- Rancher API Audit Logs

Audit logs in Rancher v2.6 have been enhanced to include the external Identity Provider name (common

name of the user in the external Auth provider) in both the Rancher and downstream Kubernetes audit logs.

Before v2.6, a Rancher Admin could not trace an event from the Rancher audit logs and into the Kubernetes audit logs without knowing the mapping of the external Identity Provider username to the userId (`u-xXXX`) used in Rancher. To know this mapping, the cluster admins needed to have access to Rancher API, UI, and the local management cluster.

Now with this feature, a downstream cluster admin should be able to look at the Kubernetes audit logs and know which specific external Identity Provider (IDP) user performed an action without needing to view anything in Rancher. If the audit logs are shipped off of the cluster, a user of the logging system should be able to identify the user in the external Identity Provider system. A Rancher Admin should now be able to view Rancher audit logs and follow through to the Kubernetes audit log by using the external Identity Provider username.

#### Feature Description

- When Kubernetes Audit logs are enabled on the downstream cluster, in each event that is logged, the external Identity Provider's username is now logged for each request, at the "metadata" level.
- When you enable Rancher API Audit logs for a Rancher installation, the external Identity Provider's username is also logged now at the `auditLog.level=0` for each request that hits the Rancher API server, including login requests.

## Networking

The commands/steps listed on this page can be used to check networking related issues in your cluster.

Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kube_config_cluster.yml` for Rancher HA) or are using the embedded kubectl via the UI.

#### Double Check if All the Required Ports are Opened in Your (Host) Firewall

Double check if all the required ports are opened in your (host) firewall. The overlay network uses UDP in comparison to all other required ports which are TCP.

#### Check if your downstream node can communicate to Rancher Manager

Rancher components with HTTP endpoints generally contain a `ping` liveness probe which you can use to test connectivity. Replace the `$RANCHER_URL` as appropriate and run the following from a node to check that it has connectivity to Rancher Manager's servers in the `local` cluster. If successful, it should return `pong`.

```
curl -k https://$RANCHER_URL/ping
```

#### Check if Overlay Network is Functioning Correctly

The pod can be scheduled to any of the hosts you used for your cluster, but that means that the NGINX ingress controller needs to be able to route the request from `NODE_1` to `NODE_2`. This happens over the overlay network. If the overlay network is not functioning, you will experience intermittent TCP/HTTP connection failures due to the NGINX ingress controller not being able to route to the pod.

To test the overlay network, you can launch the following `DaemonSet` definition. This will run a `swiss-army-knife` container on every host (image was developed by Rancher engineers and can be found here: https://github.com/rancherlabs/swiss-army-knife), which we will use to run a `ping` test between containers on all hosts.



The `swiss-army-knife` container does not support Windows nodes. It also does not support ARM nodes [https://github.com/leodotcloud/swiss-army-knife/issues/18], such as a Raspberry Pi. When the test encounters incompatible nodes, this is recorded in the pod logs as an error message, such as `exec user process caused: exec format error` for ARM nodes, or `ImagePullBackOff (Back-off pulling image "rancherlabs/swiss-army-knife)` for Windows nodes.

1. Save the following file as `overlaytest.yml`

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: overlaytest
spec:
  selector:
      matchLabels:
        name: overlaytest
  template:
    metadata:
      labels:
        name: overlaytest
    spec:
      tolerations:
      - operator: Exists
      containers:
      - image: rancherlabs/swiss-army-knife
        imagePullPolicy: Always
        name: overlaytest
        command: ["sleep", "infinity"]
        terminationMessagePath: /dev/termination-log
```

2. Launch it using `kubectl create -f overlaytest.yml`
3. Wait until `kubectl rollout status ds/overlaytest -w` returns: `daemon set "overlaytest" successfully rolled out`.
4. Run the following script, from the same location. It will have each `overlaytest` container on every host ping each other:

```
#!/bin/bash
echo "=> Start network overlay test"
  kubectl get pods -l name=overlaytest -o jsonpath='{range
```

```
.items[*]}{@.metadata.name}{" "}{@.spec.nodeName}{"\n"}{end}' |
   while read spod shost
     do kubectl get pods -l name=overlaytest -o jsonpath='{range
.items[*]}{@.status.podIP}{" "}{@.spec.nodeName}{"\n"}{end}' |
     while read tip thost
       do kubectl --request-timeout='10s' exec $spod -c overlaytest
-- /bin/sh -c "ping -c2 $tip > /dev/null 2>&1"
         RC=$?
         if [ $RC -ne 0 ]
           then echo FAIL: $spod on $shost cannot reach pod IP $tip
on $thost
           else echo $shost can reach $thost
         fi
     done
   done
 echo "=> End network overlay test"
```

5. When this command has finished running, it will output the state of each route:

```
=> Start network overlay test
Error from server (NotFound): pods "wk2" not found
FAIL: overlaytest-5bglp on wk2 cannot reach pod IP 10.42.7.3 on wk2
Error from server (NotFound): pods "wk2" not found
FAIL: overlaytest-5bglp on wk2 cannot reach pod IP 10.42.0.5 on cp1
Error from server (NotFound): pods "wk2" not found
FAIL: overlaytest-5bglp on wk2 cannot reach pod IP 10.42.2.12 on wk1
command terminated with exit code 1
FAIL: overlaytest-v4qkl on cp1 cannot reach pod IP 10.42.7.3 on wk2
cp1 can reach cp1
cp1 can reach wk1
command terminated with exit code 1
FAIL: overlaytest-xpxwp on wk1 cannot reach pod IP 10.42.7.3 on wk2
wk1 can reach cp1
wk1 can reach wk1
=> End network overlay test
```

   If you see error in the output, there is some issue with the route between the pods on the two hosts. In the above output the node `wk2` has no connectivity over the overlay network. This could be because the required ports for overlay networking are not opened for `wk2`.

6. You can now clean up the DaemonSet by running `kubectl delete ds/overlaytest`.

#### Check if MTU is Correctly Configured on Hosts and on Peering/Tunnel Appliances/Devices

When the MTU is incorrectly configured (either on hosts running Rancher, nodes in created/imported clusters or on appliances/devices in between), error messages will be logged in Rancher and in the agents,

similar to:

- `websocket: bad handshake`
- `Failed to connect to proxy`
- `read tcp: i/o timeout`

See Google Cloud VPN: MTU Considerations [https://cloud.google.com/vpn/docs/concepts/mtu-considerations#gateway_mtu_vs_system_mtu] for an example how to configure MTU correctly when using Google Cloud VPN between Rancher and cluster nodes.

## Kubernetes Resources

The commands/steps listed on this page can be used to check the most important Kubernetes resources and apply to Rancher Launched Kubernetes clusters.

Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kube_config_cluster.yml` for Rancher HA) or are using the embedded kubectl via the UI.

### Nodes

#### Get nodes

Run the command below and check the following:

- All nodes in your cluster should be listed, make sure there is not one missing.
- All nodes should have the **Ready** status (if not in **Ready** state, check the `kubelet` container logs on that node using `docker logs kubelet`)
- Check if all nodes report the correct version.
- Check if OS/Kernel/Docker values are shown as expected (possibly you can relate issues due to upgraded OS/Kernel/Docker)

```
kubectl get nodes -o wide
```

Example output:

```
NAME             STATUS   ROLES          AGE   VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
controlplane-0   Ready    controlplane   31m   v1.13.5   138.68.188.91
<none>        Ubuntu 18.04.2 LTS   4.15.0-47-generic   docker://18.9.5
etcd-0           Ready    etcd           31m   v1.13.5   138.68.180.33
<none>        Ubuntu 18.04.2 LTS   4.15.0-47-generic   docker://18.9.5
worker-0         Ready    worker         30m   v1.13.5   139.59.179.88
<none>        Ubuntu 18.04.2 LTS   4.15.0-47-generic   docker://18.9.5
```

#### Get node conditions

Run the command below to list nodes with Node Conditions [https://kubernetes.io/docs/concepts/architecture/nodes/#condition]

```
kubectl get nodes -o go-template='{{range .items}}{{$node := .}}{{range
.status.conditions}}{{$node.metadata.name}}{{":
"}}{{.type}}{{":"}}{{.status}}{{"\n"}}{{end}}{{end}}'
```

Run the command below to list nodes with Node Conditions [https://kubernetes.io/docs/concepts/architecture/nodes/#condition] that are active that could prevent normal operation.

```
kubectl get nodes -o go-template='{{range .items}}{{$node := .}}{{range
.status.conditions}}{{if ne .type "Ready"}}{{if eq .status
"True"}}{{$node.metadata.name}}{{":
"}}{{.type}}{{":"}}{{.status}}{{"\n"}}{{end}}{{else}}{{if ne .status
"True"}}{{$node.metadata.name}}{{": "}}{{.type}}{{":
"}}{{.status}}{{"\n"}}{{end}}{{end}}{{end}}{{end}}'
```

Example output:

```
worker-0: DiskPressure:True
```

### Kubernetes leader election

#### Kubernetes Controller Manager leader

The leader is determined by a leader election process. After the leader has been determined, the leader (`holderIdentity`) is saved in the `kube-controller-manager` endpoint (in this example, `controlplane-0`).

```
kubectl -n kube-system get endpoints kube-controller-manager -o
jsonpath='{.metadata.annotations.control-
plane\.alpha\.kubernetes\.io/leader}'
{"holderIdentity":"controlplane-0_xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
,"leaseDurationSeconds":15,"acquireTime":"2018-12-27T08:59:45Z"
,"renewTime":"2018-12-27T09:44:57Z","leaderTransitions":0}>
```

#### Kubernetes Scheduler leader

The leader is determined by a leader election process. After the leader has been determined, the leader (`holderIdentity`) is saved in the `kube-scheduler` endpoint (in this example, `controlplane-0`).

```
kubectl -n kube-system get endpoints kube-scheduler -o jsonpath
='{.metadata.annotations.control-plane\.alpha\.kubernetes\.io/leader}'
```

```
{"holderIdentity":"controlplane-0_xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
,"leaseDurationSeconds":15,"acquireTime":"2018-12-27T08:59:45Z"
,"renewTime":"2018-12-27T09:44:57Z","leaderTransitions":0}>
```

#### Ingress Controller

The default Ingress Controller is Traefik and is deployed as a DaemonSet in the `kube-system` namespace. The pods are only scheduled to nodes with the `worker` role.

Check if the pods are running on all nodes:

```
kubectl -n kube-system get pods -o wide
```

Example RKE2 output:

```
kubectl -n kube-system get pods -o wide
NAME                                     READY     STATUS    RESTARTS
AGE       IP              NODE
local-path-provisioner-xxxxxxxxxx-xxxxx 1/1       Running   0
17m       x.x.x.x         worker-1
rke2-traefik-xxxxxxxxxx-xxxxx            1/1       Running   0
14m       x.x.x.x         worker-1
svclb-rke2-traefik-xxxxxxxx-xxxxx        1/1       Running   0
13m       x.x.x.x         worker-0
...
```

Example K3s output:

```
kubectl -n kube-system get pods -o wide
NAME                                     READY     STATUS    RESTARTS
AGE       IP              NODE
local-path-provisioner-xxxxxxxxxx-xxxxx 1/1       Running   0
17m       x.x.x.x         worker-1
traefik-xxxxxxxxxx-xxxxx                 1/1       Running   0
14m       x.x.x.x         worker-1
svclb-traefik-xxxxxxxx-xxxxx             1/1       Running   0
13m       x.x.x.x         worker-0
...
```

If a pod is unable to run (Status is not **Running**, Ready status is not showing `1/1` or you see a high count of Restarts), check the pod details, logs and namespace events.

#### Pod details

RKE2 example:

```
kubectl -n kube-system describe pods -l app.kubernetes.io/name=rke2-
traefik
```

K3s example:

```
kubectl -n kube-system describe pods -l app.kubernetes.io/name=traefik
```

#### Pod container logs

The below command can show the logs of all the pods labeled "app.kubernetes.io/name=rke2-traefik" if using RKE2 or "app.kubernetes.io/name=traefik" if using K3s, but it will display only 10 lines of log because of the restrictions of the `kubectl logs` command. Refer to `--tail` of `kubectl logs -h` for more information.

RKE2 example:

```
kubectl -n kube-system logs -l app.kubernetes.io/name=rke2-traefik
```

K3s example:

```
kubectl -n kube-system logs -l app.kubernetes.io/name=traefik
```

If the full log is needed, specify the pod name in the trailing command:

```
kubectl -n kube-system logs <pod name>
```

#### Namespace events

```
kubectl -n kube-system get events
```

#### Debug logging

To enable debug logging:

RKE2 example:

```
cat <<EOF | kubectl apply -f -
apiVersion: helm.cattle.io/v1
kind: HelmChartConfig
metadata:
```

```
  name: rke2-traefik
  namespace: kube-system
spec:
  valuesContent: |-
    additionalArguments:
      - "--log.level=DEBUG"
EOF
```

K3s example:

```
cat <<EOF | kubectl apply -f -
apiVersion: helm.cattle.io/v1
kind: HelmChartConfig
metadata:
  name: traefik
  namespace: kube-system
spec:
  valuesContent: |-
    logs:
      general:
        level: "DEBUG"
EOF
```

#### Check configuration

Retrieve generated configuration in each pod:

RKE2 example for manual Traefik configuration file check:

```
kubectl exec -n kube-system pod/traefik-xxxxxxxxx-xxxxx -- cat
/var/lib/rancher/rke2/server/manifests/rke2-traefik-config.yaml
```

K3s example for manual Traefik configuration file check:

```
kubectl exec -n kube-system pod/traefik-xxxxxxxxx-xxxxx -- cat
/var/lib/rancher/k3s/server/manifests/k3s-traefik-config.yaml
```

RKE2/K3s example for Traefik CLI argument configuration check:

```
kubectl get pod traefik-xxxxxxxxx-xxxxx -n kube-system -o jsonpath
='{.spec.containers[0].args}'
```

### Rancher agents

Communication to the cluster (Kubernetes API via `cattle-cluster-agent`) and communication to the nodes (cluster provisioning via `cattle-node-agent`) is done through Rancher agents.

#### cattle-node-agent

Check if the cattle-node-agent pods are present on each node, have status **Running** and don't have a high count of Restarts:

```
kubectl -n cattle-system get pods -l app=cattle-agent -o wide
```

Example output:

```
NAME                       READY     STATUS    RESTARTS   AGE       IP
NODE
cattle-node-agent-4gc2p    1/1       Running   0          2h
x.x.x.x           worker-1
cattle-node-agent-8cxkk    1/1       Running   0          2h
x.x.x.x           etcd-1
cattle-node-agent-kzrlg    1/1       Running   0          2h
x.x.x.x           etcd-0
cattle-node-agent-nclz9    1/1       Running   0          2h
x.x.x.x           controlplane-0
cattle-node-agent-pwxp7    1/1       Running   0          2h
x.x.x.x           worker-0
cattle-node-agent-t5484    1/1       Running   0          2h
x.x.x.x           controlplane-1
cattle-node-agent-t8mtz    1/1       Running   0          2h
x.x.x.x           etcd-2
```

Check logging of a specific cattle-node-agent pod or all cattle-node-agent pods:

```
kubectl -n cattle-system logs -l app=cattle-agent
```

#### cattle-cluster-agent

Check if the cattle-cluster-agent pod is present in the cluster, has status **Running** and doesn't have a high count of Restarts:

```
kubectl -n cattle-system get pods -l app=cattle-cluster-agent -o wide
```

Example output:

```
NAME                                     READY     STATUS    RESTARTS
AGE       IP           NODE
cattle-cluster-agent-54d7c6c54d-ht9h4    1/1       Running   0
2h        x.x.x.x      worker-1
```

Check logging of cattle-cluster-agent pod:

```
kubectl -n cattle-system logs -l app=cattle-cluster-agent
```

### Jobs and Pods

#### Check that pods or jobs have status Running/Completed

To check, run the command:

```
kubectl get pods --all-namespaces
```

If a pod is not in **Running** state, you can dig into the root cause by running:

#### Describe pod

```
kubectl describe pod POD_NAME -n NAMESPACE
```

#### Pod container logs

```
kubectl logs POD_NAME -n NAMESPACE
```

If a job is not in **Completed** state, you can dig into the root cause by running:

#### Describe job

```
kubectl describe job JOB_NAME -n NAMESPACE
```

#### Logs from the containers of pods of the job

```
kubectl logs -l job-name=JOB_NAME -n NAMESPACE
```

#### Evicted pods

Pods can be evicted based on eviction signals [https://kubernetes.io/docs/tasks/administer-cluster/out-of-resource/#eviction-policy].

Retrieve a list of evicted pods (podname and namespace):

```
kubectl get pods --all-namespaces -o go-template='{{range .items}}{{if
eq .status.phase "Failed"}}{{if eq .status.reason
"Evicted"}}{{.metadata.name}}{{"
"}}{{.metadata.namespace}}{{"\n"}}{{end}}{{end}}{{end}}'
```

To delete all evicted pods:

```
kubectl get pods --all-namespaces -o go-template='{{range .items}}{{if
eq .status.phase "Failed"}}{{if eq .status.reason
"Evicted"}}{{.metadata.name}}{{"
"}}{{.metadata.namespace}}{{"\n"}}{{end}}{{end}}{{end}}' | while read
epod enamespace; do kubectl -n $enamespace delete pod $epod; done
```

Retrieve a list of evicted pods, scheduled node and the reason:

```
kubectl get pods --all-namespaces -o go-template='{{range .items}}{{if
eq .status.phase "Failed"}}{{if eq .status.reason
"Evicted"}}{{.metadata.name}}{{"
"}}{{.metadata.namespace}}{{"\n"}}{{end}}{{end}}{{end}}' | while read
epod enamespace; do kubectl -n $enamespace get pod $epod -o=custom-
columns=NAME:.metadata.name,NODE:.spec.nodeName,MSG:.status.message;
done
```

#### Job does not complete

If you have enabled Istio, and you are having issues with a Job you deployed not completing, you will need to add an annotation to your pod using these steps.

Since Istio Sidecars run indefinitely, a Job cannot be considered complete even after its task has completed. This is a temporary workaround and will disable Istio for any traffic to/from the annotated Pod. Keep in mind this may not allow you to continue to use a Job for integration testing, as the Job will not have access to the service mesh.

## DNS

The commands/steps listed on this page can be used to check name resolution issues in your cluster.

Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kube_config_cluster.yml` for Rancher HA) or are using the embedded kubectl via the UI.

Before running the DNS checks, make sure that the overlay network is functioning correctly for your DNS provider as this can also be the reason why DNS resolution (partly) fails.

**Check if DNS pods are running**

```
kubectl -n kube-system get pods -l k8s-app=kube-dns
```

Example output when using CoreDNS:

```
NAME                       READY   STATUS    RESTARTS   AGE
coredns-799dffd9c4-6jhlz   1/1     Running   0          76m
```

Example output when using kube-dns:

```
NAME                        READY   STATUS    RESTARTS   AGE
kube-dns-5fd74c7488-h6f7n   3/3     Running   0          4ml3s
```

**Check if the DNS service is present with the correct cluster-ip**

```
kubectl -n kube-system get svc -l k8s-app=kube-dns
```

```
NAME               TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
service/kube-dns   ClusterIP   10.43.0.10   <none>        53/UDP,53/TCP
4ml3s
```

**Check if domain names are resolving**

Check if internal cluster names are resolving (in this example, `kubernetes.default`), the IP shown after `Server:` should be the same as the `CLUSTER-IP` from the `kube-dns` service.

```
kubectl run -it --rm --restart=Never busybox --image=busybox:1.28 --
nslookup kubernetes.default
```

Example output:

```
Server:    10.43.0.10
Address 1: 10.43.0.10 kube-dns.kube-system.svc.cluster.local

Name:      kubernetes.default
Address 1: 10.43.0.1 kubernetes.default.svc.cluster.local
pod "busybox" deleted
```

Check if external names are resolving (in this example, `www.google.com`)

```
kubectl run -it --rm --restart=Never busybox --image=busybox:1.28 --
nslookup www.google.com
```

Example output:

```
Server:    10.43.0.10
Address 1: 10.43.0.10 kube-dns.kube-system.svc.cluster.local

Name:      www.google.com
Address 1: 2a00:1450:4009:80b::2004 lhr35s04-in-x04.1e100.net
Address 2: 216.58.211.100 ams15s32-in-f4.1e100.net
pod "busybox" deleted
```

If you want to check resolving of domain names on all of the hosts, execute the following steps:

1. Save the following file as `ds-dnstest.yml`

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: dnstest
spec:
  selector:
      matchLabels:
        name: dnstest
  template:
    metadata:
      labels:
        name: dnstest
    spec:
      tolerations:
      - operator: Exists
      containers:
      - image: busybox:1.28
        imagePullPolicy: Always
        name: alpine
        command: ["sleep", "infinity"]
        terminationMessagePath: /dev/termination-log
```

2. Launch it using `kubectl create -f ds-dnstest.yml`
3. Wait until `kubectl rollout status ds/dnstest -w` returns: `daemon set "dnstest" successfully rolled out`.
4. Configure the environment variable `DOMAIN` to a fully qualified domain name (FQDN) that the host

should be able to resolve (`www.google.com` is used as an example) and run the following command to let each container on every host resolve the configured domain name (it's a single line command).

```
 export DOMAIN=www.google.com; echo "=> Start DNS resolve test";
kubectl get pods -l name=dnstest --no-headers -o custom-columns
=NAME:.metadata.name,HOSTIP:.status.hostIP | while read pod host; do
kubectl exec $pod -- /bin/sh -c "nslookup $DOMAIN > /dev/null 2>&1";
RC=$?; if [ $RC -ne 0 ]; then echo $host cannot resolve $DOMAIN; fi;
done; echo "=> End DNS resolve test"
```

5. When this command has finished running, the output indicating everything is correct is:

```
=> Start DNS resolve test
=> End DNS resolve test
```

If you see error in the output, that means that the mentioned host(s) is/are not able to resolve the given FQDN.

Example error output of a situation where host with IP 209.97.182.150 had the UDP ports blocked.

```
=> Start DNS resolve test
command terminated with exit code 1
209.97.182.150 cannot resolve www.google.com
=> End DNS resolve test
```

Cleanup the alpine DaemonSet by running `kubectl delete ds/dnstest`.

### CoreDNS specific

#### Check CoreDNS logging

```
kubectl -n kube-system logs -l k8s-app=kube-dns
```

#### Check configuration

CoreDNS configuration is stored in the configmap `coredns` in the `kube-system` namespace.

```
kubectl -n kube-system get configmap coredns -o go-template
={{.data.Corefile}}
```

#### Check upstream nameservers in resolv.conf

By default, the configured nameservers on the host (in `/etc/resolv.conf`) will be used as upstream nameservers for CoreDNS. You can check this file on the host or run the following Pod with `dnsPolicy` set

to `Default`, which will inherit the `/etc/resolv.conf` from the host it is running on.

```
kubectl run -i --restart=Never --rm test-${RANDOM} --image=ubuntu
--overrides='{"kind":"Pod", "apiVersion":"v1", "spec":
{"dnsPolicy":"Default"}}' -- sh -c 'cat /etc/resolv.conf'
```

#### Enable query logging

Enabling query logging can be done by enabling the log plugin [https://coredns.io/plugins/log/] in the Corefile configuration in the configmap `coredns`. You can do so by using `kubectl -n kube-system edit configmap coredns` or use the command below to replace the configuration in place:

```
kubectl get configmap -n kube-system coredns -o json | sed -e
's_loadbalance_log\\n    loadbalance_g' | kubectl apply -f -
```

All queries will now be logged and can be checked using the command in Check CoreDNS logging.

### kube-dns specific

#### Check upstream nameservers in kubedns container

By default, the configured nameservers on the host (in `/etc/resolv.conf`) will be used as upstream nameservers for kube-dns. Sometimes the host will run a local caching DNS nameserver, which means the address in `/etc/resolv.conf` will point to an address in the loopback range (`127.0.0.0/8`) which will be unreachable by the container. In case of Ubuntu 18.04, this is done by `systemd-resolved`. We detect if `systemd-resolved` is running, and will automatically use the `/etc/resolv.conf` file with the correct upstream nameservers (which is located at `/run/systemd/resolve/resolv.conf`).

Use the following command to check the upstream nameservers used by the kubedns container:

```
kubectl -n kube-system get pods -l k8s-app=kube-dns --no-headers -o
custom-columns=NAME:.metadata.name,HOSTIP:.status.hostIP | while read
pod host; do echo "Pod ${pod} on host ${host}"; kubectl -n kube-system
exec $pod -c kubedns cat /etc/resolv.conf; done
```

Example output:

```
Pod kube-dns-667c7cb9dd-z4dsf on host x.x.x.x
nameserver 1.1.1.1
nameserver 8.8.4.4
```

If the output shows an address in the loopback range (`127.0.0.0/8`), you can correct this in two ways:

- Make sure the correct nameservers are listed in `/etc/resolv.conf` on your nodes in the cluster, please consult your operating system documentation on how to do this. Make sure you execute this before provisioning a cluster, or reboot the nodes after making the modification.

- Configure the `kubelet` to use a different file for resolving names, by using `extra_args` as shown below (where `/run/resolvconf/resolv.conf` is the file with the correct nameservers):

```
services:
  kubelet:
    extra_args:
      resolv-conf: "/run/resolvconf/resolv.conf"
```



As the `kubelet` is running inside a container, the path for files located in `/etc` and `/usr` are in `/host/etc` and `/host/usr` inside the `kubelet` container.

See Editing Cluster as YAML how to apply this change. When the provisioning of the cluster has finished, you have to remove the kube-dns pod to activate the new setting in the pod:

```
kubectl delete pods -n kube-system -l k8s-app=kube-dns
pod "kube-dns-5fd74c7488-6pwsf" deleted
```

Try to resolve name again using Check if domain names are resolving.

If you want to check the kube-dns configuration in your cluster (for example, to check if there are different upstream nameservers configured), you can run the following command to list the kube-dns configuration:

```
kubectl -n kube-system get configmap kube-dns -o go-template='{{range
$key, $value := .data}}{{ $key }}{{":"}}{{ $value }}{{"\n"}}{{end}}'
```

Example output:

```
upstreamNameservers:["1.1.1.1"]
```

# FAQ

## General FAQ

This FAQ is a work in progress designed to answer the questions most frequently asked about Rancher v2.x.

See the Technical FAQ for frequently asked technical questions.

### Is it possible to manage Azure Kubernetes Services with Rancher v2.x?

Yes. See our Cluster Administration guide for what Rancher features are available on AKS, as well as our documentation on AKS.

### Does Rancher support Windows?

Yes. Rancher supports Windows Server 1809 containers. For details on how to set up a cluster with Windows worker nodes, refer to the section on configuring custom clusters for Windows.

### Does Rancher support Istio?

Yes. Rancher supports Istio.

### Will Rancher v2.x support Hashicorp's Vault for storing secrets?

Secrets management is on our roadmap but we haven't assigned it to a specific release yet.

### Does Rancher v2.x support RKT containers as well?

At this time, we only support Docker.

### Does Rancher v2.x support Calico, Contiv, Contrail, Flannel, Weave net, etc., for embedded and registered Kubernetes?

Out-of-the-box, Rancher provides the following CNI network providers for Kubernetes clusters: Canal, Flannel, and Calico. Always refer to the Rancher Support Matrix [https://rancher.com/support-maintenance-terms/] for details about what is officially supported.

### Are you planning on supporting Traefik for existing setups?

We don't currently plan on providing embedded Traefik support, but we're still exploring load-balancing approaches.

### Can I import OpenShift Kubernetes clusters into v2.x?

Our goal is to run any Kubernetes clusters. Therefore, Rancher v2.x should work with OpenShift, but we haven't tested it yet.

### Is SUSE Storage integrated with Rancher?

Yes. Longhorn is integrated with Rancher v2.5 and later.

# Deprecated Features in SUSE Rancher Prime

### What is Rancher's deprecation policy?

The official deprecation policy is documented in the Rancher Prime Deprecation Policy [https://www.suse.com/support/rancher-prime/#Rancher-Prime-Deprecation-Policy].

### Where can I find out which features have been deprecated in Rancher?

Rancher will publish deprecated features as part of the release notes [https://github.com/rancher/rancher/releases] for Rancher found on GitHub. Please consult the following patch releases for deprecated features:

| Patch Version | Release Date |
|---|---|
| 2.15.3 [https://github.com/rancher/rancher/releases/tag/v2.15.3] | 2026-10-29 |
| 2.15.2 [https://github.com/rancher/rancher/releases/tag/v2.15.2] | 2026-09-23 |
| 2.15.1 [https://github.com/rancher/rancher/releases/tag/v2.15.1] | 2026-08-27 |
| 2.15.0 [https://github.com/rancher/rancher/releases/tag/v2.15.0] | 2026-07-30 |

### What can I expect when a feature is marked for deprecation?

In the release where functionality is marked as "Deprecated", it will still be available and supported allowing upgrades to follow the usual procedure. Once upgraded, users/admins should start planning to move away from the deprecated functionality before upgrading to the release it marked as removed. The recommendation for new deployments is to not use the deprecated feature.

# Installing and Configuring kubectl

`kubectl` is a CLI utility for running commands against Kubernetes clusters. It's required for many maintenance and administrative tasks in Rancher 2.x.

### Installation

See kubectl Installation [https://kubernetes.io/docs/tasks/tools/install-kubectl/] for installation on your operating system.

### Configuration

When you create a Kubernetes cluster with RKE2/K3s, the Kubeconfig file is stored at `/etc/rancher/rke2/rke2.yaml` or `/etc/rancher/k3s/k3s.yaml` depending on your chosen distribution. These files are used to configure access to the Kubernetes cluster.

Test your connectivity with `kubectl` and see if you can get the list of nodes back.

```
kubectl get nodes
 NAME                          STATUS    ROLES                       AGE
VERSION
165.227.114.63                Ready     controlplane,etcd,worker    11m
v1.10.1
```

```
165.227.116.167              Ready     controlplane,etcd,worker   11m
v1.10.1
165.227.127.226              Ready     controlplane,etcd,worker   11m
v1.10.1
```

# Technical FAQ

### How can I reset the administrator password?

Docker install:

```
$ docker exec -ti <container_id> reset-password
New password for default administrator (user-xxxxx):
<new_password>
```

Kubernetes install (Helm):

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl --kubeconfig $KUBECONFIG -n cattle-system exec $(kubectl
--kubeconfig $KUBECONFIG -n cattle-system get pods -l app=rancher --no
-headers | head -1 | awk '{ print $1 }') -c rancher -- reset-password
New password for default administrator (user-xxxxx):
<new_password>
```

Kubernetes install (Helm - **Applicable to Rancher Prime v2.13.1**):

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl --kubeconfig $KUBECONFIG -n cattle-system exec $(kubectl
--kubeconfig $KUBECONFIG -n cattle-system get pods -l app=rancher-prime
--no-headers | head -1 | awk '{ print $1 }') -c rancher-prime -- reset-
password
New password for default administrator (user-xxxxx):
<new_password>
```

### I deleted/deactivated the last admin, how can I fix it?

Docker install:

```
$ docker exec -ti <container_id> ensure-default-admin
New default administrator (user-xxxxx)
New password for default administrator (user-xxxxx):
<new_password>
```

Kubernetes install (Helm):

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl --kubeconfig $KUBECONFIG -n cattle-system exec $(kubectl
--kubeconfig $KUBECONFIG -n cattle-system get pods -l app=rancher | grep
'1/1' | head -1 | awk '{ print $1 }') -- ensure-default-admin
New password for default administrator (user-xxxxx):
<new_password>
```

Kubernetes install (Helm - **Applicable to Rancher Prime v2.13.1**):

```
$ KUBECONFIG=./kube_config_cluster.yml
$ kubectl --kubeconfig $KUBECONFIG -n cattle-system exec $(kubectl
--kubeconfig $KUBECONFIG -n cattle-system get pods -l app=rancher-prime
| grep '1/1' | head -1 | awk '{ print $1 }') -- ensure-default-admin
New password for default administrator (user-xxxxx):
<new_password>
```

### How can I enable debug logging?

See Troubleshooting: Logging

### My ClusterIP does not respond to ping

ClusterIP is a virtual IP, which will not respond to ping. Best way to test if the ClusterIP is configured correctly, is by using `curl` to access the IP and port to see if it responds.

### Where can I manage Node Templates?

Node Templates can be accessed by opening your account menu (top right) and selecting `Node Templates`.

### Why is my Layer-4 Load Balancer in `Pending` state?

The Layer-4 Load Balancer is created as `type: LoadBalancer`. In Kubernetes, this needs a cloud provider or controller that can satisfy these requests, otherwise these will be in `Pending` state forever. More information can be found on Cloud Providers or Create External Load Balancer [https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/]

### Where is the state of Rancher stored?

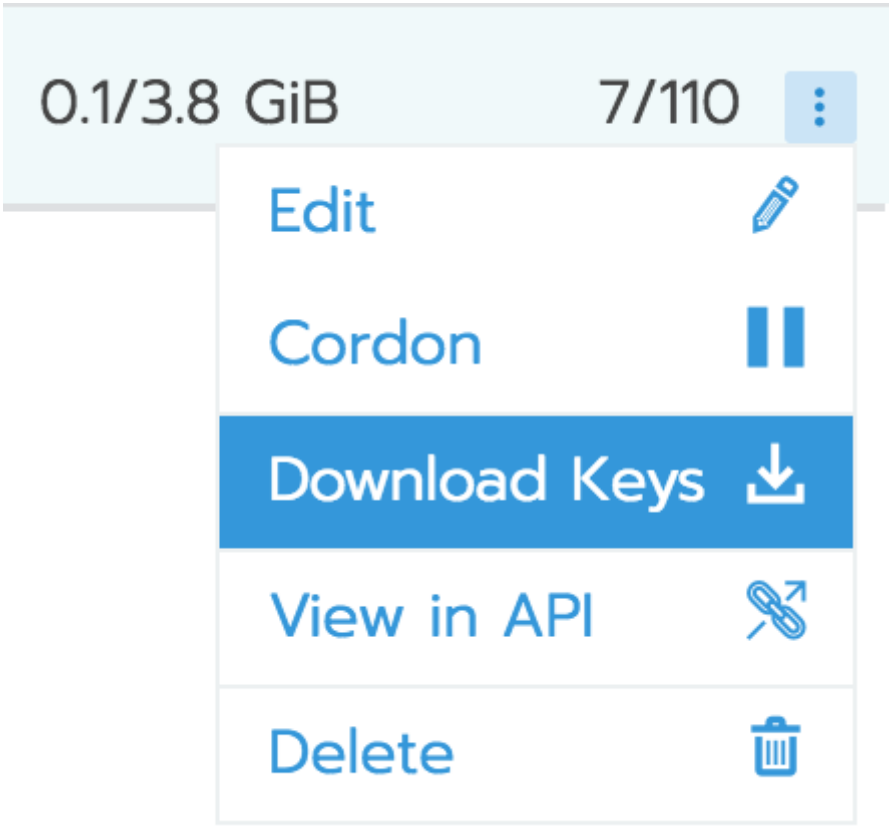
- Docker Install: in the embedded etcd of the `rancher/rancher` container, located at `/var/lib/rancher`.
- Kubernetes install: default location is in the `/var/lib/rancher/rke2` or `/var/lib/rancher/k3s` directories of the respective RKE2/K3s cluster created to run Rancher.

### How are the supported Docker versions determined?

We follow the validated Docker versions for upstream Kubernetes releases. The validated versions can be found under External Dependencies [https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.10.md#external-dependencies] in the Kubernetes release CHANGELOG.md.

### How can I access nodes created by Rancher?

SSH keys to access the nodes created by Rancher can be downloaded via the **Nodes** view. Choose the node which you want to access and click on the vertical ⋮ button at the end of the row, and choose **Download Keys** as shown in the picture below.




Unzip the downloaded zip file, and use the file `id_rsa` to connect to you host. Be sure to use the correct username (`rancher` or `docker` for RancherOS, `ubuntu` for Ubuntu, `ec2-user` for Amazon Linux)

```
$ ssh -i id_rsa user@ip_of_node
```

### How can I automate task X in Rancher?

The UI consists of static files, and works based on responses of the API. That means every action/task that you can execute in the UI, can be automated via the API. There are 2 ways to do this:

- Visit `https://your_rancher_ip/v3` and browse the API options.
- Capture the API calls when using the UI (Most commonly used for this is Chrome Developer Tools [https://developers.google.com/web/tools/chrome-devtools/#network] but you can use anything you like)

### The IP address of a node changed, how can I recover?

A node is required to have a static IP configured (or a reserved IP via DHCP). If the IP of a node has changed, you will have to remove it from the cluster and re-add it. After it is removed, Rancher will update the cluster to the correct state. If the cluster is no longer in `Provisioning` state, the node is removed from the cluster.

When the IP address of the node changed, Rancher lost connection to the node, so it will be unable to clean the node properly. See Cleaning cluster nodes to clean the node.

When the node is removed from the cluster, and the node is cleaned, you can re-add the node to the cluster.

### How can I add more arguments/binds/environment variables to Kubernetes components in a Rancher Launched Kubernetes cluster?

You can add more arguments/binds/environment variables via the respective RKE2 Config File or K3s Config File.

### How do I check if my certificate chain is valid?

Use the `openssl verify` command to validate your certificate chain:



Configure `SSL_CERT_DIR` and `SSL_CERT_FILE` to a dummy location to make sure the OS-installed certificates are not used when verifying manually.

```
SSL_CERT_DIR=/dummy SSL_CERT_FILE=/dummy openssl verify -CAfile ca.pem
rancher.yourdomain.com.pem
rancher.yourdomain.com.pem: OK
```

If you receive the error `unable to get local issuer certificate`, the chain is incomplete. This usually means that there is an intermediate CA certificate that issued your server certificate. If you already have this certificate, you can use it in the verification of the certificate like shown below:

```
SSL_CERT_DIR=/dummy SSL_CERT_FILE=/dummy openssl verify -CAfile ca.pem
-untrusted intermediate.pem rancher.yourdomain.com.pem
rancher.yourdomain.com.pem: OK
```

If you have successfully verified your certificate chain, you should include needed intermediate CA certificates in the server certificate to complete the certificate chain for any connection made to Rancher (for example, by the Rancher agent). The order of the certificates in the server certificate file should be first the server certificate itself (contents of `rancher.yourdomain.com.pem`), followed by intermediate CA certificate(s) (contents of `intermediate.pem`).

```
-----BEGIN CERTIFICATE-----
%YOUR_CERTIFICATE%
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
%YOUR_INTERMEDIATE_CERTIFICATE%
```

```
-----END CERTIFICATE-----
```

If you still get errors during verification, you can retrieve the subject and the issuer of the server certificate using the following command:

```
openssl x509 -noout -subject -issuer -in rancher.yourdomain.com.pem
subject= /C=GB/ST=England/O=Alice Ltd/CN=rancher.yourdomain.com
issuer= /C=GB/ST=England/O=Alice Ltd/CN=Alice Intermediate CA
```

### How do I check `Common Name` and `Subject Alternative Names` in my server certificate?

Although technically an entry in `Subject Alternative Names` is required, having the hostname in both `Common Name` and as entry in `Subject Alternative Names` gives you maximum compatibility with older browser/applications.

Check `Common Name`:

```
openssl x509 -noout -subject -in cert.pem
subject= /CN=rancher.my.org
```

Check `Subject Alternative Names`:

```
openssl x509 -noout -in cert.pem -text | grep DNS
                DNS:rancher.my.org
```

### Why does it take 5+ minutes for a pod to be rescheduled when a node has failed?

This is due to a combination of the following default Kubernetes settings:

- kubelet
  - `node-status-update-frequency`: Specifies how often kubelet posts node status to master (default 10s)
- kube-controller-manager
  - `node-monitor-period`: The period for syncing NodeStatus in NodeController (default 5s)
  - `node-monitor-grace-period`: Amount of time which we allow running Node to be unresponsive before marking it unhealthy (default 40s)
  - `pod-eviction-timeout`: The grace period for deleting pods on failed nodes (default 5m0s)

See Kubernetes: kubelet [https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/] and Kubernetes: kube-controller-manager [https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/] for more information on these settings.

In Kubernetes v1.13, the `TaintBasedEvictions` feature is enabled by default. See Kubernetes: Taint

based Evictions [https://kubernetes.io/docs/concepts/configuration/taint-and-toleration/#taint-based-evictions] for more information.

- kube-apiserver (Kubernetes v1.13 and up)
  - `default-not-ready-toleration-seconds`: Indicates the tolerationSeconds of the toleration for notReady:NoExecute that is added by default to every pod that does not already have such a toleration.
  - `default-unreachable-toleration-seconds`: Indicates the tolerationSeconds of the toleration for unreachable:NoExecute that is added by default to every pod that does not already have such a toleration.

# Security FAQ

### Is there a Hardening Guide?

The Hardening Guide is located in the main Security section.

### Have hardened Rancher Kubernetes clusters been evaluated by the CIS Kubernetes Benchmark? Where can I find the results?

We have run the CIS Kubernetes benchmark against a hardened Rancher Kubernetes cluster. The results of that assessment can be found in the main Security section.

### How does Rancher verify communication with downstream clusters, and what are some associated security concerns?

Communication between the Rancher server and downstream clusters is performed through agents. Rancher uses either a registered certificate authority (CA) bundle or the local trust store to verify communication between Rancher agents and the Rancher server. Using a CA bundle for verification is more strict, as only the certificates based on that bundle are trusted. If TLS verification for a explicit CA bundle fails, Rancher may fall back to using the local trust store for verifying future communication. Any CA within the local trust store can then be used to generate a valid certificate.

As described in Rancher Security Update CVE-2024-22030 [https://www.suse.com/c/rancher-security-update/], under a narrow set of circumstances, malicious actors can take over Rancher nodes by exploiting the behavior of Rancher CAs. For the attack to succeed, the malicious actor must generate a valid certificate from either a valid CA in the targeted Rancher server, or from a valid registered CA. The attacker also needs to either hijack or spoof the Rancher server-url as a preliminary step. Rancher is currently evaluating Rancher CA behavior to mitigate against this and any similar avenues of attack.

# Container Network Interface (CNI) Providers

### What is CNI?

CNI (Container Network Interface), a Cloud Native Computing Foundation project [https://cncf.io/], consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted.

Kubernetes uses CNI as an interface between network providers and Kubernetes pod networking.




For more information visit CNI GitHub project [https://github.com/containernetworking/cni].

## What Network Models are Used in CNI?

CNI network providers implement their network fabric using either an encapsulated network model such as Virtual Extensible Lan (VXLAN [https://github.com/flannel-io/flannel/blob/master/Documentation/backends.md#vxlan]) or an unencapsulated network model such as Border Gateway Protocol (BGP [https://en.wikipedia.org/wiki/Border_Gateway_Protocol]).

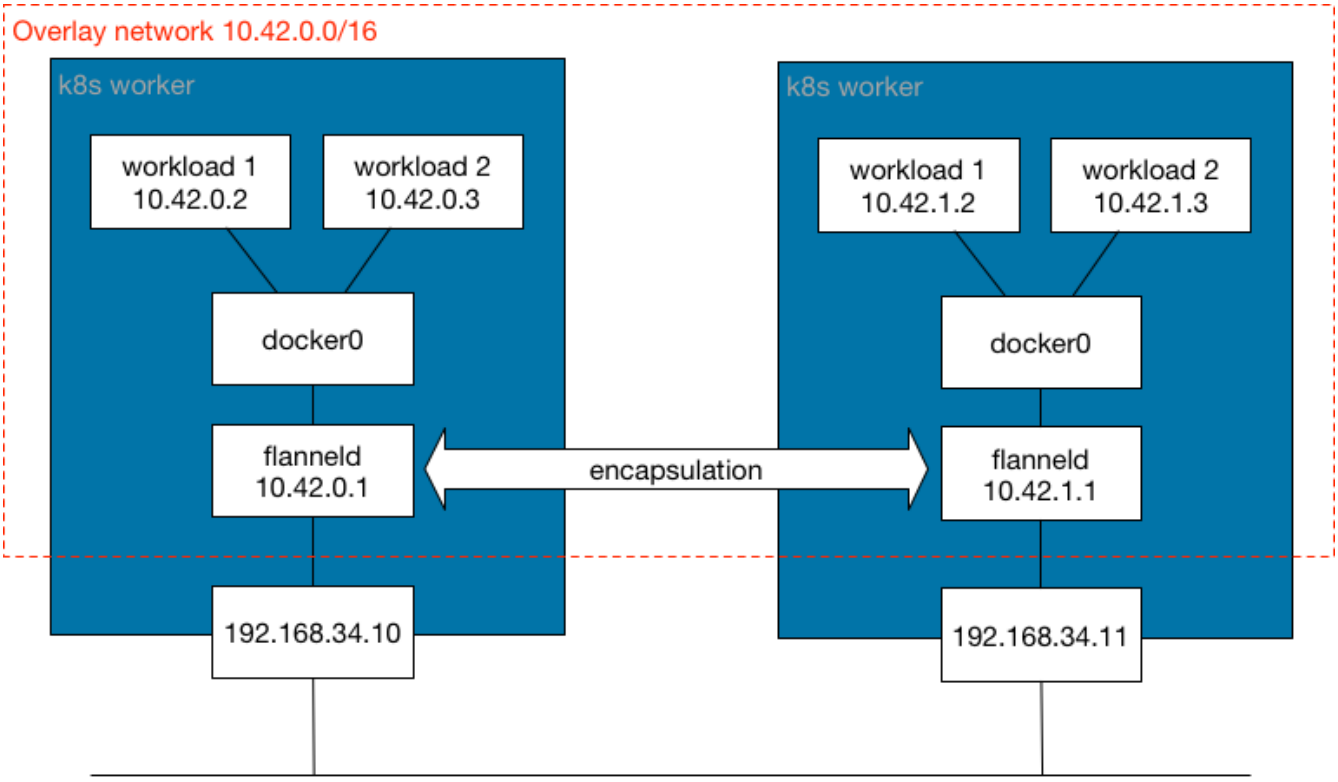
### What is an Encapsulated Network?

This network model provides a logical Layer 2 (L2) network encapsulated over the existing Layer 3 (L3) network topology that spans the Kubernetes cluster nodes. With this model you have an isolated L2 network for containers without needing routing distribution, all at the cost of minimal overhead in terms of processing and increased IP package size, which comes from an IP header generated by overlay encapsulation. Encapsulation information is distributed by UDP ports between Kubernetes workers, interchanging network control plane information about how MAC addresses can be reached. Common encapsulation used in this kind of network model is VXLAN, Internet Protocol Security (IPSec), and IP-in-IP.

In simple terms, this network model generates a kind of network bridge extended between Kubernetes workers, where pods are connected.

This network model is used when an extended L2 bridge is preferred. This network model is sensitive to L3 network latencies of the Kubernetes workers. If datacenters are in distinct geolocations, be sure to have low latencies between them to avoid eventual network segmentation.

CNI network providers using this network model include Flannel, Canal, and Cilium. By default, Calico is not using this model, but it can be configured to do so.



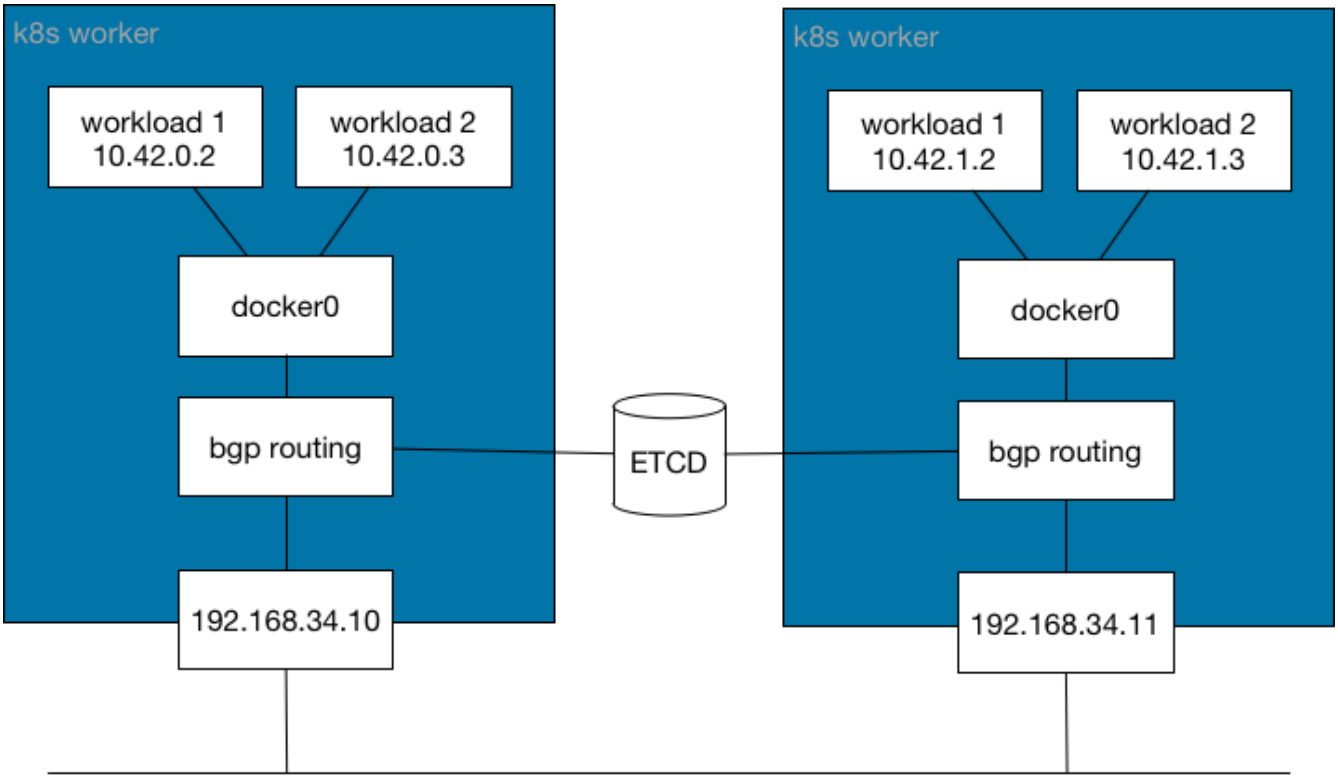

**What is an Unencapsulated Network?**

This network model provides an L3 network to route packets between containers. This model doesn't generate an isolated l2 network, nor generates overhead. These benefits come at the cost of Kubernetes workers having to manage any route distribution that's needed. Instead of using IP headers for encapsulation, this network model uses a network protocol between Kubernetes workers to distribute routing information to reach pods, such as BGP [https://en.wikipedia.org/wiki/Border_Gateway_Protocol].

In simple terms, this network model generates a kind of network router extended between Kubernetes workers, which provides information about how to reach pods.

This network model is used when a routed L3 network is preferred. This mode dynamically updates routes at the OS level for Kubernetes workers. It's less sensitive to latency.

CNI network providers using this network model include Calico and Cilium. Cilium may be configured with this model although it is not the default mode.




## What CNI Providers are Provided by Rancher?

#### SUSE® Rancher Prime: RKE2 Kubernetes clusters

Out-of-the-box, Rancher provides the following CNI network providers for RKE2 Kubernetes clusters: Calico, Canal, Cilium, and Flannel.

You can choose your CNI network provider when you create new Kubernetes clusters from Rancher.

#### Calico

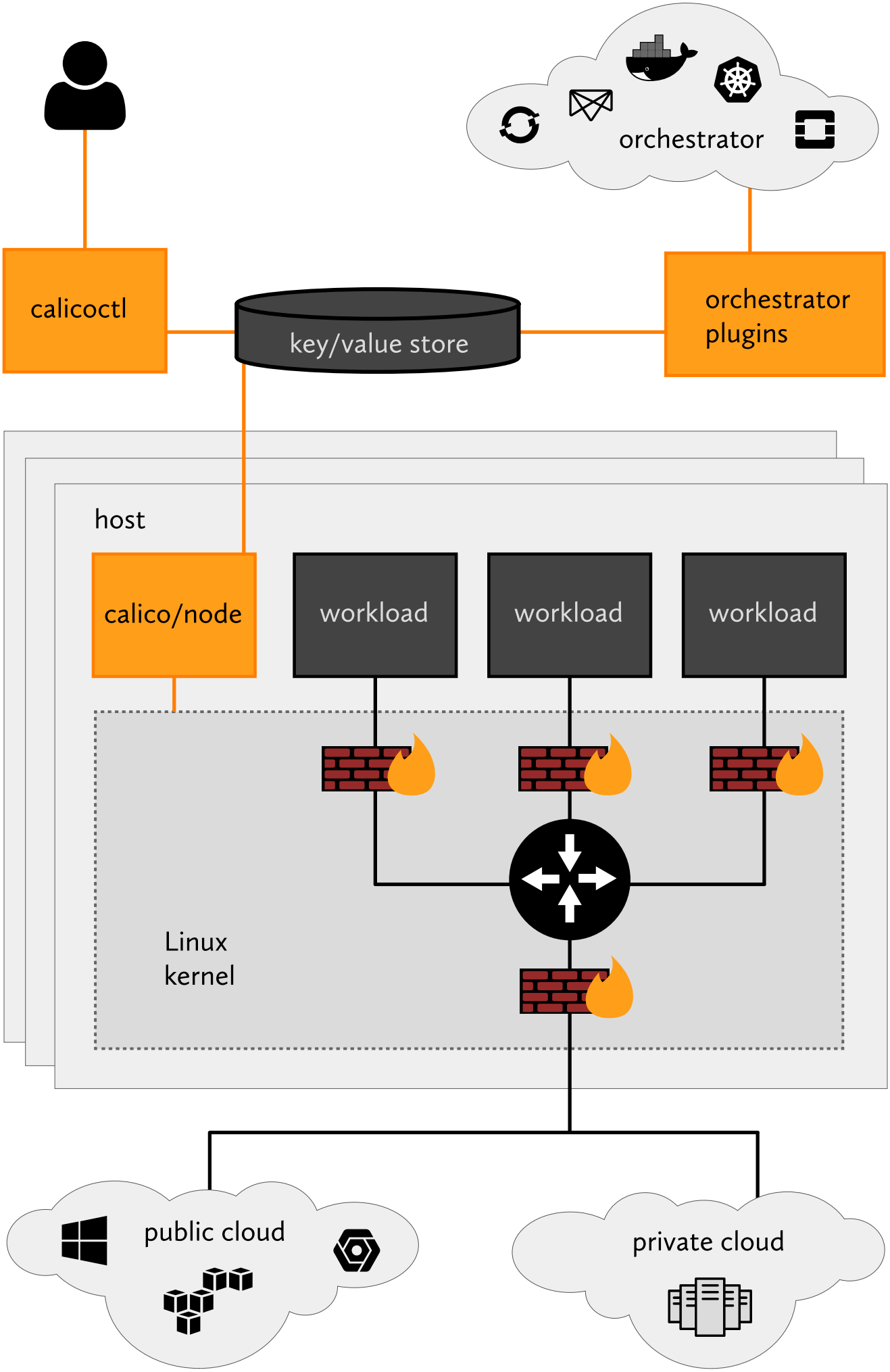


Calico enables networking and network policy in Kubernetes clusters across the cloud. By default, Calico uses a pure, unencapsulated IP network fabric and policy engine to provide networking for your Kubernetes workloads. Workloads are able to communicate over both cloud infrastructure and on-prem using BGP.

Calico also provides a stateless IP-in-IP or VXLAN encapsulation mode that can be used, if necessary. Calico also offers policy isolation, allowing you to secure and govern your Kubernetes workloads using advanced

ingress and egress policies.

Kubernetes workers should open TCP port 179 if using BGP or UDP port 4789 if using VXLAN encapsulation. In addition, TCP port 5473 is needed when using Typha. See the port requirements for user clusters for more details.


orchestrator
calicoctl
key/value store
orchestrator plugins
host
calico/node
workload
workload
workload
Linux kernel
public cloud
private cloud

For more information, see the following pages:

- Project Calico Official Site [https://www.projectcalico.org/]
- Project Calico GitHub Page [https://github.com/projectcalico/calico]

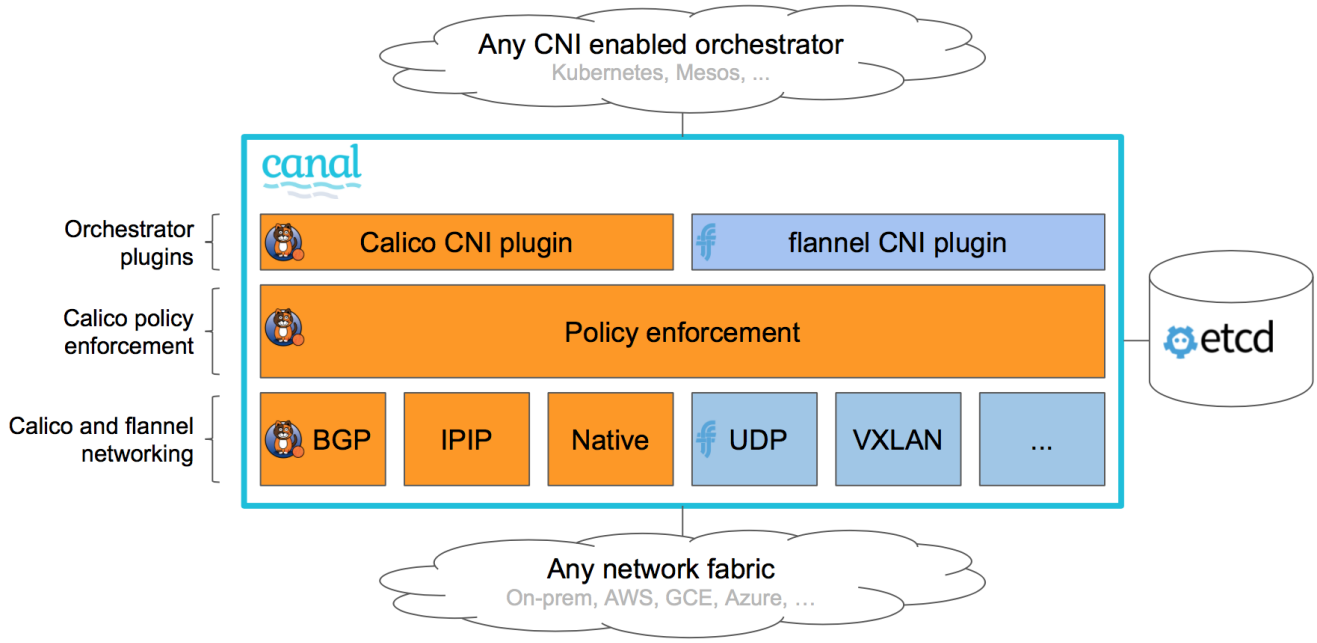
**Canal**




Canal is a CNI network provider that gives you the best of Flannel and Calico. It allows users to easily deploy Calico and Flannel networking together as a unified networking solution, combining Calico's network policy enforcement with the rich superset of Calico (unencapsulated) and/or Flannel (encapsulated) network connectivity options.

In Rancher, Canal is the default CNI network provider combined with Flannel and VXLAN encapsulation.

Kubernetes workers should open UDP port 8472 (VXLAN) and TCP port 9099 (health checks). If using Wireguard, you should open UDP ports 51820 and 51821. For more details, refer to the port requirements for user clusters.




For more information, refer to the Rancher maintained Canal source [https://github.com/rancher/rke2-charts/tree/main-source/packages/rke2-canal] and the Canal GitHub Page [https://github.com/projectcalico/canal].

**Cilium**




Cilium enables networking and network policies (L3, L4, and L7) in Kubernetes. By default, Cilium uses eBPF technologies to route packets inside the node and VXLAN to send packets to other nodes. Unencapsulated techniques can also be configured.

Cilium recommends kernel versions greater than 5.2 to be able to leverage the full potential of eBPF. Kubernetes workers should open TCP port 8472 for VXLAN and TCP port 4240 for health checks. In addition, ICMP 8/0 must be enabled for health checks. For more information, check Cilium System Requirements [https://docs.cilium.io/en/latest/operations/system_requirements/#firewall-requirements].

**Flannel**

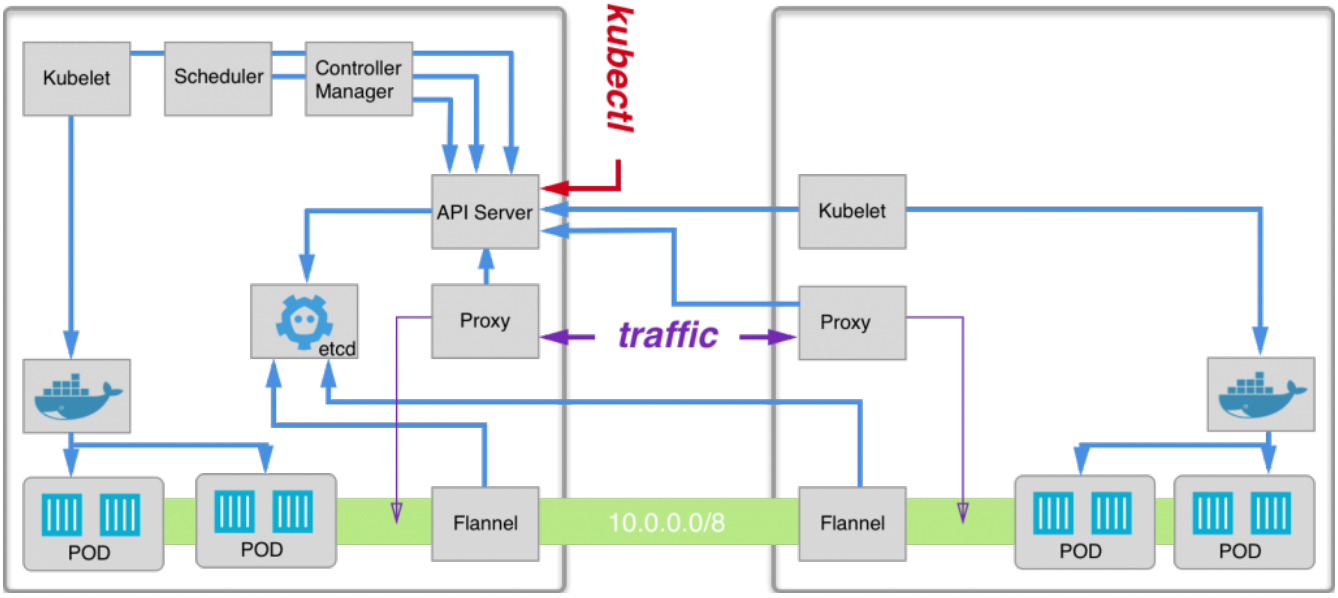



Flannel is a simple and easy way to configure L3 network fabric designed for Kubernetes. Flannel runs a single binary agent named flanneld on each host, which is responsible for allocating a subnet lease to each host out of a larger, preconfigured address space. Flannel uses either the Kubernetes API or etcd directly to store the network configuration, the allocated subnets, and any auxiliary data (such as the host's public IP). Packets are forwarded using one of several backend mechanisms, with the default encapsulation being VXLAN [https://github.com/flannel-io/flannel/blob/master/Documentation/backends.md#vxlan].

Encapsulated traffic is unencrypted by default. Flannel provides two solutions for encryption:

- IPSec [https://github.com/flannel-io/flannel/blob/master/Documentation/backends.md#ipsec], which makes use of strongSwan [https://www.strongswan.org/] to establish encrypted IPSec tunnels between Kubernetes workers. It is an experimental backend for encryption.
- WireGuard [https://github.com/flannel-io/flannel/blob/master/Documentation/backends.md#wireguard], which is a more faster-performing alternative to strongSwan.

Kubernetes workers should open UDP port 8472 (VXLAN). See the port requirements for user clusters for more details.




For more information, see the Flannel GitHub Page [https://github.com/flannel-io/flannel].

#### Ingress Routing Across Nodes in Cilium

By default, Cilium does not allow pods to contact pods on other nodes. To work around this, enable the ingress controller to route requests across nodes with a `CiliumNetworkPolicy`.

After selecting the Cilium CNI and enabling Project Network Isolation for your new cluster, configure as follows:

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: hn-nodes
  namespace: default
spec:
  endpointSelector: {}
  ingress:
    - fromEntities:
      - remote-node
```

## CNI Features by Provider

The following table summarizes the different features available for each CNI network provider provided by Rancher.

| Provider | Network Model | Route Distribution | Network Policies | Mesh | External Datastore | Encryption | Ingress/Egress Policies |
|---|---|---|---|---|---|---|---|
| Canal | Encapsulated (VXLAN) | No | Yes | No | K8s API | Yes | Yes |
| Flannel | Encapsulated (VXLAN) | No | No | No | K8s API | Yes | No |

| Provider | Network Model | Route Distribution | Network Policies | Mesh | External Datastore | Encryption | Ingress/Egress Policies |
|---|---|---|---|---|---|---|---|
| Calico | Encapsulated (VXLAN,IPIP) OR Unencapsulated | Yes | Yes | Yes | Etcd and K8s API | Yes | Yes |
| Cilium | Encapsulated (VXLAN) | Yes | Yes | Yes | Etcd and K8s API | Yes | Yes |

- Network Model: Encapsulated or unencapsulated. For more information, see What Network Models are Used in CNI?
- Route Distribution: An exterior gateway protocol designed to exchange routing and reachability information on the Internet. BGP can assist with pod-to-pod networking between clusters. This feature is a must on unencapsulated CNI network providers, and it is typically done by BGP. If you plan to build clusters split across network segments, route distribution is a feature that's nice-to-have.
- Network Policies: Kubernetes offers functionality to enforce rules about which services can communicate with each other using network policies. This feature is stable as of Kubernetes v1.7 and is ready to use with certain networking plugins.
- Mesh: This feature allows service-to-service networking communication between distinct Kubernetes clusters.
- External Datastore: CNI network providers with this feature need an external datastore for its data.
- Encryption: This feature allows cyphered and secure network control and data planes.
- Ingress/Egress Policies: This feature allows you to manage routing control for both Kubernetes and non-Kubernetes communications.

## CNI Community Popularity

The following table summarizes different GitHub metrics to give you an idea of each project's popularity and activity levels. This data was collected in December 2025.

| Provider | Project | Stars | Forks | Contributors |
|---|---|---|---|---|
| Canal | https://github.com/projectcalico/canal | 721 | 96 | 20 |
| Flannel | https://github.com/flannel-io/flannel | 9.5k | 2.9k | 252 |
| Calico | https://github.com/projectcalico/calico | 7.3k | 1.6k | 418 |
| Weave | https://github.com/weaveworks/weave/ | 6.6k | 673 | 82 |
| Cilium | https://github.com/cilium/cilium | 24.8k | 3.9k | 1126 |

## Which CNI Provider Should I Use?

It depends on your project needs. There are many different providers, which each have various features and options. There isn't one provider that meets everyone's needs.

Canal is the default CNI network provider. We recommend it for most use cases. It provides encapsulated networking for containers with Flannel, while adding Calico network policies that can provide project/namespace isolation in terms of networking.

### How can I configure a CNI network provider?

Please see Cluster Options on how to configure a network provider for your cluster. For more advanced configuration options, please see how to configure your cluster using a Config File.

## SUSE Rancher Prime is No Longer Needed

This page is intended to answer questions about what happens if you don't want Rancher anymore, if you don't want a cluster to be managed by Rancher anymore, or if the Rancher server is deleted.

### If the Rancher server is deleted, what happens to the workloads in my downstream clusters?

If Rancher is ever deleted or unrecoverable, all workloads in the downstream Kubernetes clusters managed by Rancher will continue to function as normal.

### If the Rancher server is deleted, how do I access my downstream clusters?

The capability to access a downstream cluster without Rancher depends on the type of cluster and the way that the cluster was created. To summarize:

- **Registered/Imported clusters:** The cluster will be unaffected and you can access the cluster using the same methods that you did before the cluster was registered into Rancher.
- **Hosted Kubernetes clusters:** If you created the cluster in a cloud-hosted Kubernetes provider such as EKS, GKE, or AKS, you can continue to manage the cluster using your provider's cloud credentials.
- **Rancher provisioned clusters:** To access an RKE2/K3s cluster, the cluster must have the authorized cluster endpoint enabled, and you must have already downloaded the cluster's kubeconfig file from the Rancher UI. (The authorized cluster endpoint is enabled by default for RKE clusters.) With this endpoint, you can access your cluster with kubectl directly instead of communicating through the Rancher server's authentication proxy. For instructions on how to configure kubectl to use the authorized cluster endpoint, refer to the section about directly accessing clusters with kubectl and the kubeconfig file. These clusters will use a snapshot of the authentication as it was configured when Rancher was removed.

### What if I don't want Rancher anymore?

If you installed Rancher on a Kubernetes cluster, remove Rancher by using the Rancher Cleanup [https://github.com/rancher/rancher-cleanup] tool.

Uninstalling Rancher in high-availability (HA) mode will also remove all `helm-operation-*` pods and the following apps:

- fleet
- fleet-agent
- rancher-operator
- rancher-webhook

Custom resources (CRDs) and custom namespaces will still need to be manually removed.

If you installed Rancher with Docker, you can uninstall Rancher by removing the single Docker container that it runs in.

Imported clusters will not be affected by Rancher being removed. For other types of clusters, refer to the section on accessing downstream clusters when Rancher is removed.

## What if I don't want my registered cluster managed by Rancher?

If a registered cluster is deleted from the Rancher UI, the cluster is detached from Rancher, leaving it intact and accessible by the same methods that were used to access it before it was registered in Rancher.

To detach the cluster,

1. In the upper left corner, click ☰ > **Cluster Management**.
2. Go to the registered cluster that should be detached from Rancher and click ☰ > **Delete**.
3. Click **Delete**.

**Result:** The registered cluster is detached from Rancher and functions normally outside of Rancher.

## What if I don't want my hosted Kubernetes cluster managed by Rancher?

At this time, there is no functionality to detach these clusters from Rancher. In this context, "detach" is defined as the ability to remove Rancher components from the cluster and manage access to the cluster independently of Rancher.

The capability to manage these clusters without Rancher is being tracked in this issue. [https://github.com/rancher/rancher/issues/25234]

For information about how to access clusters if the Rancher server is deleted, refer to this section.

# Contributing to SUSE Rancher Prime

Learn about the repositories used for Rancher and Rancher docs, how to build Rancher repositories, and what information to include when you file an issue.

For more detailed information on how to contribute to the development of Rancher projects, refer to the Rancher Developer Wiki [https://github.com/rancher/rancher/wiki]. The wiki has resources on many topics, including the following:

- How to set up the Rancher development environment and run tests
- The typical flow of an issue through the development lifecycle
- Coding guidelines and development best practices
- Debugging and troubleshooting
- Developing the Rancher API

On the Rancher Users Slack, the channel for developers is **#developer**.

## Rancher Docs

If you have suggestions for the documentation on this website, please open an issue [https://github.com/rancher/rancher-product-docs/issues/new/choose] on the repository.

Refer to the Rancher docs README [https://github.com/rancher/rancher-product-docs#readme] for more details on contributing to and building the Rancher docs repo.

## Rancher Repositories

All of repositories are located within our main GitHub organization. There are many repositories used for Rancher, but we'll provide descriptions of some of the main ones used in Rancher.

| Repository | URL | Description |
|---|---|---|
| Rancher | https://github.com/rancher/rancher | This repository is the main source code for Rancher. |
| Types | https://github.com/rancher/types | This repository is the repository that has all the API types for Rancher. |
| API Framework | https://github.com/rancher/norman | This repository is an API framework for building Rancher style APIs backed by Kubernetes Custom Resources. |
| User Interface | https://github.com/rancher/dashboard/ | This repository is the source of the Dashboard UI. |
| (Rancher) Docker Machine | https://github.com/rancher/machine | This repository is the source of the Docker Machine binary used when using Node Drivers. This is a fork of the `docker/machine` repository. |
| machine-package | https://github.com/rancher/machine-package | This repository is used to build the Rancher Docker Machine binary. |

| Repository | URL | Description |
|---|---|---|
| kontainer-engine | https://github.com/rancher/kontainer-engine | This repository is the source of kontainer-engine, the tool to provision hosted Kubernetes clusters. |
| CLI | https://github.com/rancher/cli | This repository is the source code for the Rancher CLI used in Rancher. |
| (Rancher) Helm repository | https://github.com/rancher/helm | This repository is the source of the packaged Helm binary. This is a fork of the `helm/helm` repository. |
| loglevel repository | https://github.com/rancher/loglevel | This repository is the source of the loglevel binary, used to dynamically change log levels. |

To see all libraries/projects used in Rancher, see the `go.mod` file [https://github.com/rancher/rancher/blob/master/go.mod] in the `rancher/rancher` repository.

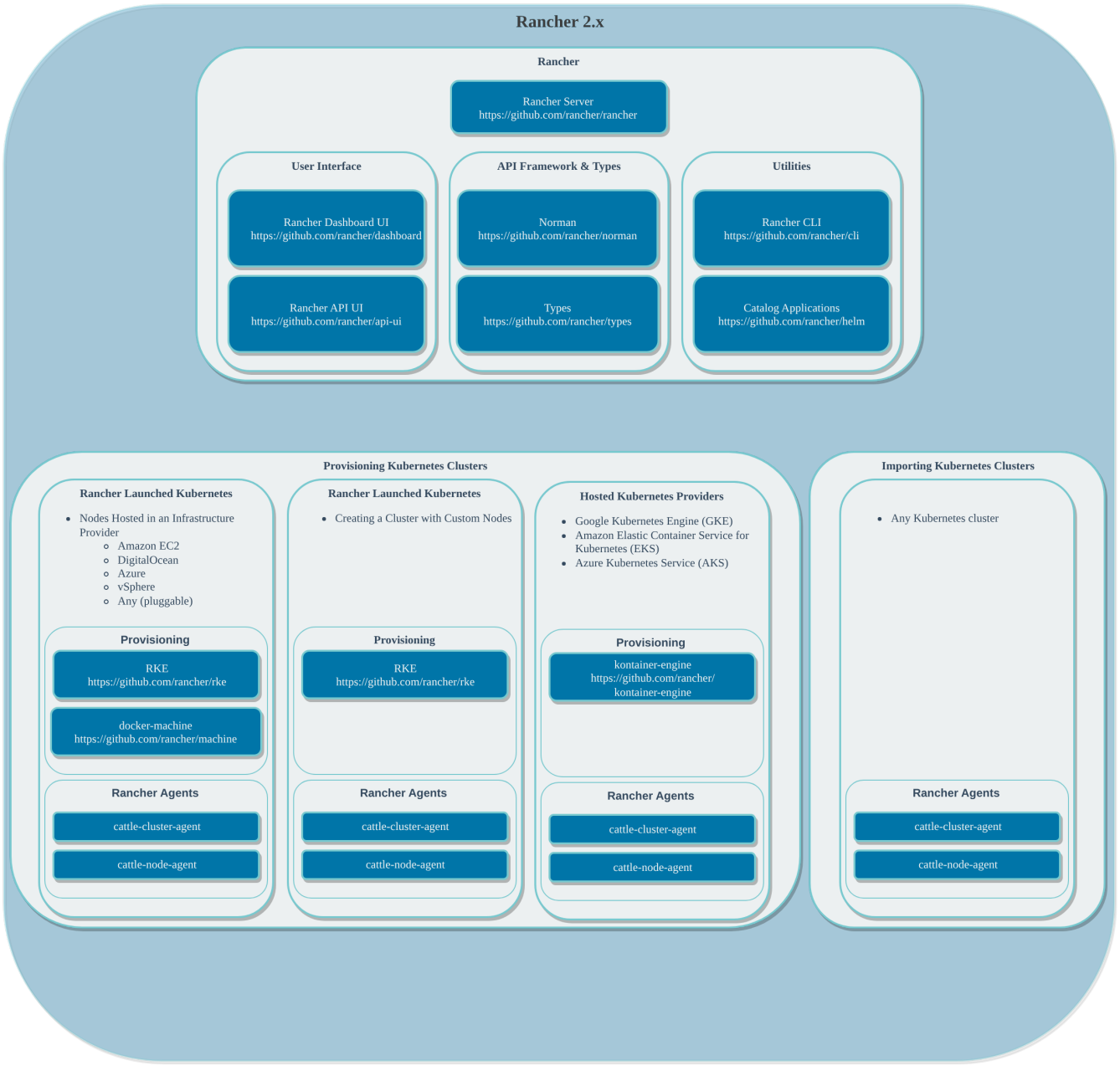



*Figure 28. Rancher components used for provisioning/managing Kubernetes clusters*

## Rancher Bugs, Issues or Questions

If you find any bugs or are having any trouble, please search the reported issue [https://github.com/rancher/rancher/issues] as someone may have experienced the same issue or we are actively working on a solution.

If you can't find anything related to your issue, contact us by filing an issue [https://github.com/rancher/rancher/issues/new]. Though we have many repositories related to Rancher, we want the bugs filed in the Rancher repository so we won't miss them! If you want to ask a question or ask fellow users about an use case, we suggest creating a post on the Rancher Forums [https://forums.rancher.com].

### Checklist for Filing Issues

Please follow this checklist when filing an issue which will helps us investigate and fix the issue. More info means more data we can use to determine what is causing the issue or what might be related to the issue.



For large amounts of data, please use GitHub Gist [https://gist.github.com/] or similar and link the created resource in the issue.



Please remove any sensitive data as it will be publicly viewable.

- **Resources:** Provide as much as detail as possible on the used resources. As the source of the issue can be many things, including as much of detail as possible helps to determine the root cause. See some examples below:
  - **Hosts:** What specifications does the host have, like CPU/memory/disk, what cloud does it happen on, what Amazon Machine Image are you using, what DigitalOcean droplet are you using, what image are you provisioning that we can rebuild or use when we try to reproduce
  - **Operating System:** What operating system are you using? Providing specifics helps here like the output of `cat /etc/os-release` for exact OS release and `uname -r` for exact kernel used
  - **Environment:** Are you in a proxy environment, are you using recognized CA/self signed certificates, are you using an external loadbalancer
  - **Rancher:** What version of Rancher are you using, this can be found on the bottom left of the UI or be retrieved from the image tag you are running on the host
  - **Clusters:** What kind of cluster did you create, how did you create it, what did you specify when you were creating it
- **Steps to reproduce the issue:** Provide as much detail on how you got into the reported situation. This helps the person to reproduce the situation you are in.
  - Provide manual steps or automation scripts used to get from a newly created setup to the situation you reported.
- **Logs:** Provide data/logs from the used resources.
  - Rancher
    - Kubernetes install using `kubectl`



Make sure you configured the correct kubeconfig (for example, `export KUBECONFIG=$PWD/kube_config_cluster.yml` if Rancher is installed on a Kubernetes cluster) or are using the embedded kubectl via the UI.

```
kubectl -n cattle-system \
logs \
-l app=rancher \
--timestamps=true
```

- System logging (these might not all exist, depending on operating system)
  - `/var/log/messages`
  - `/var/log/syslog`
  - `/var/log/kern.log`

- **Metrics:** If you are experiencing performance issues, please provide as much of data (files or screenshots) of metrics which can help determining what is going on. If you have an issue related to a machine, it helps to supply output of `top`, `free -m`, `df` which shows processes/memory/disk usage.