OpenShift Container Platform 4.8

Updating clusters

Updating OpenShift Container Platform clusters
Abstract

This document provides instructions for updating, or upgrading, OpenShift Container Platform clusters. Updating your cluster is a simple process that does not require you to take your cluster offline.
9.1.2. Preparing your mirror host

9.1.2.1. Installing the OpenShift CLI by downloading the binary
   Installing the OpenShift CLI on Linux
   Installing the OpenShift CLI on Windows
   Installing the OpenShift CLI on macOS

9.1.3. Configuring credentials that allow images to be mirrored

9.1.4. Mirroring the OpenShift Container Platform image repository

9.1.5. Upgrading the restricted network cluster

9.1.6. Configuring image registry repository mirroring

9.1.7. Widening the scope of the mirror image catalog to reduce the frequency of cluster node reboots

9.1.8. Upgrading the restricted network cluster

9.2. ADDITIONAL RESOURCES

9.3. UPDATING A RESTRICTED NETWORK CLUSTER USING THE OPENSHIFT UPDATE SERVICE

9.3.1. About the OpenShift Update Service

9.3.2. Prerequisites

9.3.3. Configuring access to a secured registry for the OpenShift update service

9.3.4. Updating the global cluster pull secret

9.3.5. Installing the OpenShift Update Service Operator

9.3.6. Installing the OpenShift Update Service Operator by using the web console

9.3.7. Installing the OpenShift Update Service Operator by using the CLI

9.3.8. Creating the OpenShift Update Service graph data container image

9.3.9. Mirroring the OpenShift Container Platform image repository

9.3.10. Creating an OpenShift Update Service application
   9.3.10.1. Creating an OpenShift Update Service application by using the web console
   9.3.10.2. Creating an OpenShift Update Service application by using the CLI
   9.3.10.3. Configuring the Cluster Version Operator (CVO)

9.3.11. Deleting an OpenShift Update Service application
   9.3.11.1. Deleting an OpenShift Update Service application by using the web console
   9.3.11.2. Deleting an OpenShift Update Service application by using the CLI

9.3.12. Uninstalling the OpenShift Update Service Operator
   9.3.12.1. Uninstalling the OpenShift Update Service Operator by using the web console
   9.3.12.2. Uninstalling the OpenShift Update Service Operator by using the CLI
CHAPTER 1. UNDERSTANDING OPENSHIFT CONTAINER PLATFORM UPDATES

With OpenShift Container Platform 4, you can update a OpenShift Container Platform cluster with a single operation by using the web console or the OpenShift CLI (oc). Platform administrators are automatically notified when an update is available for their cluster.

The OpenShift Update Service (OSUS) builds a graph of update possibilities based on release images in the registry. The graph is based on recommended, tested update paths from a specific version. OpenShift Container Platform clusters connect to the Red Hat Hybrid Cloud servers and identify which clusters the user is running, along with the version information. OSUS responds with information about known update targets. Either a cluster administrator or an automatic update controller edits the custom resource (CR) of the Cluster Version Operator (CVO) with the new version to update to. After the CVO receives the update image from the registry, the CVO then applies the changes.

1.1. COMMON TERMS

Control plane

The control plane, which is composed of control plane machines, manages the OpenShift Container Platform cluster. The control plane machines manage workloads on the compute machines, which are also known as worker machines.

Cluster Version Operator

The Cluster Version Operator (CVO) starts the update process for the cluster. It checks with OSUS based on the current cluster version and retrieves the graph which contains available or possible update paths.

Machine Config Operator

The Machine Config Operator (MCO) is a cluster-level Operator that manages the operating system and machine configurations. Through the MCO, platform administrators can configure and update systemd, CRI-O and Kubelet, the kernel, NetworkManager, and other system features on the worker nodes.

OpenShift Update Service

The OpenShift Update Service (OSUS) provides over-the-air updates to OpenShift Container Platform, including to Red Hat Enterprise Linux CoreOS (RHCOS). It provides a graph, or diagram, that contains the vertices of component Operators and the edges that connect them.

Channels

Channels declare an update strategy tied to minor versions of OpenShift Container Platform. The OSUS uses this configured strategy to recommend update edges consistent with that strategy.

Recommended update edge

A recommended update edge is a recommended update between OpenShift Container Platform releases. Whether a given update is recommended can depend on the cluster’s configured channel, current version, known bugs, and other information. OSUS communicates the recommended edges to the CVO, which runs in every cluster.

Extended Update Support

All post-4.7 even-numbered minor releases are labeled as Extended Update Support (EUS) releases. These releases introduce a verified update path between EUS releases, permitting customers to streamline updates of worker worker nodes and formulate update strategies of EUS-to-EUS OpenShift Container Platform releases that will cause fewer reboots of worker nodes. For more information, see Red Hat OpenShift Extended Update Support (EUS) Overview.
Additional resources

- Machine config overview
- About the OpenShift Update Service
- Update channels and releases
CHAPTER 2. UPDATING CLUSTERS OVERVIEW

You can update a OpenShift Container Platform cluster with a single operation by using the web console or the OpenShift CLI (oc) with OpenShift Container Platform 4.

2.1. UNDERSTANDING OPENSIFT CONTAINER PLATFORM UPDATES

About the OpenShift Update Service: For clusters with internet accessibility, Red Hat provides over-the-air updates through an OpenShift Container Platform update service as a hosted service located behind public APIs.

Upgrading channels and releases: Upgrade channels allow you to choose an upgrade strategy. Upgrade channels are connected to a minor version of OpenShift Container Platform. Upgrade channels control only release selection and do not impact the version of the cluster that you install; the openshift-install binary file for a specific version of OpenShift Container Platform always installs that version. For more information, see the following information:

- Upgrading version paths
- Understanding fast and stable channel use and strategies
- Understanding restricted network clusters
- Switching between channels

Preparing to perform an EUS-to-EUS update: Due to fundamental Kubernetes design, all OpenShift Container Platform updates between minor versions must be serialized. You must update from OpenShift Container Platform 4.8 to 4.9, and then to 4.10. You cannot update from OpenShift Container Platform 4.8 to 4.10 directly. However, if you want to update between two Extended Update Support (EUS) versions, you can do so by incurring only a single reboot of non-master hosts. For more information, see the following item:

- Updating EUS-to-EUS

Updating a cluster within a minor version using the web console: You can update an OpenShift Container Platform cluster by using the web console. You can the update a cluster between minor versions. For more information, see the following:

- Performing a canary rollout update
- Updating a cluster by using the web console
- Changing the update server by using the web console

Updating a cluster within a minor version using the CLI: You can update an OpenShift Container Platform cluster by using the oc. You can the update a cluster between minor versions. For more information, see the following:
Performing a canary rollout update: By controlling rollout of an update to the worker nodes, you can ensure that mission-critical applications stay available during the whole update, even if the update process causes your applications to fail. Depending on your organizational needs, you might want to update a small subset of worker nodes, evaluate cluster and workload health over a period of time, and then update the remaining nodes. This is referred to as a canary update. Alternatively, you might also want to fit worker node updates, which often requires a host reboot, into smaller defined maintenance windows when it is not possible to take a large maintenance window to update the entire cluster at one time. You can perform the following actions:

- Creating machine configuration pools to perform a canary rollout update
- Pausing the machine configuration pools
- Performing the cluster update
- Unpausing the machine configuration pools
- Moving a node to the original machine configuration pool

Updating a cluster that includes RHEL compute machines: You can update an OpenShift Container Platform cluster. If your cluster contains Red Hat Enterprise Linux (RHEL) machines, you must update those machines. You can perform the following actions:

- Optional: Adding hooks to perform Ansible tasks on RHEL machines
- Updating RHEL compute machines in your cluster

Updating a restricted network cluster: A restricted network environment is one in which your cluster nodes cannot access the internet. You can update a restricted network OpenShift Container Platform cluster by using the `oc`. You can also update a restricted network OpenShift Container Platform cluster by using the OpenShift Update Service. For more information, see the following:

- Preparing your mirror host
- Configuring credentials that allow images to be mirrored
- Mirroring the OpenShift Container Platform image repository
- Updating the restricted network cluster
- Configuring image registry repository mirroring
- Widening the scope of the mirror image catalog to reduce the frequency of cluster node reboots
- Installing the OpenShift Update Service Operator
Creating an OpenShift Update Service application
Deleting an OpenShift Update Service application
Uninstalling the OpenShift Update Service Operator
</updating-clusters-overview-update-restricted-network-cluster>
CHAPTER 3. UNDERSTANDING UPGRADE CHANNELS AND RELEASES

In OpenShift Container Platform 4.1, Red Hat introduced the concept of channels for recommending the appropriate release versions for cluster updates. By controlling the pace of updates, these upgrade channels allow you to choose an update strategy. Upgrade channels are tied to a minor version of OpenShift Container Platform. For instance, OpenShift Container Platform 4.8 upgrade channels recommend update to 4.8 and updates within 4.8. They also recommend updates within 4.7 and from 4.7 to 4.8, to allow clusters on 4.7 to eventually update to 4.8. They do not recommend updates to 4.9 or later releases. This strategy ensures that administrators explicitly decide to update to the next minor version of OpenShift Container Platform.

Upgrade channels control only release selection and do not impact the version of the cluster that you install; the openshift-install binary file for a specific version of OpenShift Container Platform always installs that version.

OpenShift Container Platform 4.8 offers the following upgrade channels:

- candidate-4.8
- fast-4.8
- stable-4.8
- eus-4.y (only when running an even-numbered 4.y cluster release, like 4.8)

**WARNING**

Red Hat recommends upgrading to versions suggested by Openshift Update Service only. For minor version update, versions must be contiguous. Red Hat does not test updates to noncontiguous versions and cannot guarantee compatibility with earlier versions.

3.1. UPGRADE CHANNELS AND RELEASE PATHS

Cluster administrators can configure the upgrade channel from the web console.

3.1.1. candidate-4.8 channel

The candidate-4.8 channel contains candidate builds for a z-stream (4.8.z) and previous minor version releases. Release candidates contain all the features of the product but are not supported. Use release candidates to test feature acceptance and assist in qualifying the next version of OpenShift Container Platform. A release candidate is any build that is available in the candidate channel, including ones that do not contain a pre-release version such as -rc in their names. After a version is available in the candidate channel, it goes through more quality checks. If it meets the quality standard, it is promoted to the fast-4.8 or stable-4.8 channels. Because of this strategy, if a specific release is available in both the candidate-4.8 channel and in the fast-4.8 or stable-4.8 channels, it is a Red Hat-supported version. The candidate-4.8 channel can include release versions from which there are no recommended updates in any channel.
You can use the **candidate-4.8** channel to update from a previous minor version of OpenShift Container Platform.

**NOTE**

Release candidates differ from the nightly builds. Nightly builds are available for early access to features, but updating to or from nightly builds is neither recommended nor supported. Nightly builds are not available in any upgrade channel. You can reference the OpenShift Container Platform **release statuses** for more build information.

### 3.1.2. fast-4.8 channel

The **fast-4.8** channel is updated with new and previous minor versions of 4.8 as soon as Red Hat declares the given version as a general availability release. As such, these releases are fully supported, are production quality, and have performed well while available as a release candidate in the **candidate-4.8** channel from where they were promoted. Some time after a release appears in the **fast-4.8** channel, it is added to the **stable-4.8** channel. Releases never appear in the **stable-4.8** channel before they appear in the **fast-4.8** channel.

You can use the **fast-4.8** channel to update from a previous minor version of OpenShift Container Platform.

### 3.1.3. stable-4.8 channel

While the **fast-4.8** channel contains releases as soon as their errata are published, releases are added to the **stable-4.8** channel after a delay. During this delay, data is collected from Red Hat SRE teams, Red Hat support services, and pre-production and production environments that participate in connected customer program about the stability of the release.

You can use the **stable-4.8** channel to update from a previous minor version of OpenShift Container Platform.

### 3.1.4. eus-4.y channel

In addition to the stable channel, all even-numbered minor versions of OpenShift Container Platform offer an **Extended Update Support** (EUS). These EUS versions extend the Full and Maintenance support phases for customers with Standard and Premium Subscriptions to 18 months.

Although there is no difference between **stable-4.y** and **eus-4.y** channels until OpenShift Container Platform 4.y transitions to the EUS phase, you can switch to the **eus-4.y** channel as soon as it becomes available.

When updates to the next EUS channel are offered, you can switch to the next EUS channel and update until you have reached the next EUS version.

This update process does not apply for updates coming from the **eus-4.6** channel. Upgrading to the next EUS channel from **eus-4.6** requires a serial set of version updates, from 4.6 to 4.7 to 4.8.

**NOTE**

Both standard and non-EUS subscribers can access all EUS repositories and necessary RPMs (**rhel-*-eus-rpms**) to be able to support critical purposes such as debugging and building drivers.
3.1.5. Upgrade version paths

OpenShift Container Platform maintains an update recommendation service that understands the version of OpenShift Container Platform you have installed as well as the path to take within the channel you choose to get you to the next release.

You can imagine seeing the following in the fast-4.8 channel:

- 4.8.0
- 4.8.1
- 4.8.3
- 4.8.4

The service recommends only updates that have been tested and have no serious issues. It will not suggest updating to a version of OpenShift Container Platform that contains known vulnerabilities. For example, if your cluster is on 4.8.1 and OpenShift Container Platform suggests 4.8.4, then it is safe for you to update from 4.8.1 to 4.8.4. Do not rely on consecutive patch numbers. In this example, 4.8.2 is not and never was available in the channel.

Update stability depends on your channel. The presence of an update recommendation in the candidate-4.8 channel does not imply that the update is supported. It means that no serious issues have been found with the update yet, but there might not be significant traffic through the update to suggest stability. The presence of an update recommendation in the fast-4.8 or stable-4.8 channels at any point is a declaration that the update is supported. While releases will never be removed from a channel, update recommendations that exhibit serious issues will be removed from all channels. Updates initiated after the update recommendation has been removed are still supported.

Red Hat will eventually provide supported update paths from any supported release in the fast-4.8 or stable-4.8 channels to the latest release in 4.8.z, although there can be delays while safe paths away from troubled releases are constructed and verified.

3.1.6. Fast and stable channel use and strategies

The fast-4.8 and stable-4.8 channels present a choice between receiving general availability releases as soon as they are available or allowing Red Hat to control the rollout of those updates. If issues are detected during rollout or at a later time, updates to that version might be blocked in both the fast-4.8 and stable-4.8 channels, and a new version might be introduced that becomes the new preferred update target.

Customers can improve this process by configuring pre-production systems on the fast-4.8 channel, configuring production systems on the stable-4.8 channel, and participating in the Red Hat connected customer program. Red Hat uses this program to observe the impact of updates on your specific hardware and software configurations. Future releases might improve or alter the pace at which updates move from the fast-4.8 to the stable-4.8 channel.

3.1.7. Restricted network clusters

If you manage the container images for your OpenShift Container Platform clusters yourself, you must consult the Red Hat errata that is associated with product releases and note any comments that impact updates. During update, the user interface might warn you about switching between these versions, so you must ensure that you selected an appropriate version before you bypass those warnings.

3.1.8. Switching between channels
A channel can be switched from the web console or through the `patch` command:

```
$ oc patch clusterversion version --type json -p '[["op": "add", "path": "/spec/channel", "value": "<channel>"}]]'
```

The web console will display an alert if you switch to a channel that does not include the current release. The web console does not recommend any updates while on a channel without the current release. You can return to the original channel at any point, however.

Changing your channel might impact the supportability of your cluster. The following conditions might apply:

- Your cluster is still supported if you change from the `stable-4.8` channel to the `fast-4.8` channel.
- You can switch to the `candidate-4.8` channel but, some releases for this channel might be unsupported.
- You can switch from the `candidate-4.8` channel to the `fast-4.8` channel if your current release is a general availability release.
- You can always switch from the `fast-4.8` channel to the `stable-4.8` channel. There is a possible delay of up to a day for the release to be promoted to `stable-4.8` if the current release was recently promoted.
CHAPTER 4. PREPARING TO PERFORM AN EUS-TO-EUS UPDATE

Due to fundamental Kubernetes design, all OpenShift Container Platform updates between minor versions must be serialized. You must update from OpenShift Container Platform 4.8 to 4.9 and then to 4.10. You cannot update from OpenShift Container Platform 4.8 to 4.10 directly. However, beginning with the update from OpenShift Container Platform 4.8 to 4.9 to 4.10, administrators who wish to update between two Extended Update Support (EUS) versions can do so incurring only a single reboot of non-master hosts.

There are a number of caveats to consider when attempting an EUS-to-EUS update.

- EUS-to-EUS updates are only offered after updates between all versions involved have been made available in stable channels.

- If you encounter issues during or after upgrading to the odd-numbered minor version but before upgrading to the next even-numbered version, then remediation of those issues may require that non-master hosts complete the update to the odd-numbered version before moving forward.

- You can complete the update process during multiple maintenance windows by pausing at intermediate steps. However, plan to complete the entire update within 60 days. This is critical to ensure that normal cluster automation processes are completed including those associated with certificate rotation.

- You must be running at least OpenShift Container Platform 4.8.14 before starting the EUS-to-EUS update procedure. If you do not meet this minimum requirement, update to a later 4.8.z before attempting the EUS-to-EUS update.

- Support for RHEL7 workers was removed in OpenShift Container Platform 4.10 and replaced with RHEL8 workers, therefore EUS-to-EUS updates are not available for clusters with RHEL7 workers.

- Node components are not updated to OpenShift Container Platform 4.9. Do not expect all features and bugs fixed in OpenShift Container Platform 4.9 to be made available until you complete the update to OpenShift Container Platform 4.10 and enable all MachineConfigPools to update.

4.1. EUS-TO-EUS UPDATE

The following procedure pauses all non-master MachineConfigPools and performs updates from OpenShift Container Platform 4.8 to 4.9 to 4.10, then unpauses the previously paused MachineConfigPools. Following this procedure reduces the total update duration and the number of times worker nodes are restarted.

Prerequisites

- Review the release notes for OpenShift Container Platform 4.9 and 4.10

- Review the release notes and product lifecycles for any layered products and OLM Operators. Some may require updates either before or during an EUS-to-EUS update.

- Ensure that you are familiar with version-specific prerequisites, such as administrator acknowledgement that is required prior to updating from OpenShift Container Platform 4.8 to 4.9.
• Verify that your cluster is running OpenShift Container Platform version 4.8.14 or later. If your cluster is running a version earlier than OpenShift Container Platform 4.8.14, you must update to a later 4.8.z version before updating to 4.9. The update to 4.8.14 or later is necessary to fulfill the minimum version requirements that must be performed without pausing MachineConfigPools.

• Verify that MachineConfigPools is unpaused.

Procedure

1. Upgrade any OLM Operators to versions that are compatible with both versions you are updating to.

2. Verify that all MachineConfigPools display a status of UPDATED and no MachineConfigPools display a status of UPDATING. View the status of all MachineConfigPools, run the following command:

   ```
   $ oc get mcp
   ```

   **Example output**
   
   Output is trimmed for clarity:

   ```
   NAME       CONFIG                                      UPDATED   UPDATING
   master     rendered-master-ecbb9582781c1091e1c9f19d50c0836c True     False
   worker     rendered-worker-00a3f0c68ae94e747193156b491553d5 True     False
   ```

3. Pause the MachineConfigPools you wish to skip reboots on, run the following commands:

   ```
   $ oc patch mcp/worker --type merge --patch '{"spec":{"paused":true}}'
   ```

4. Change to the **eus-4.10** channel, run the following command:

   ```
   $ oc adm upgrade channel eus-4.10
   ```

5. Update to 4.9, run the following command:

   ```
   $ oc adm upgrade --to-latest
   ```

   **Example output**

   Updating to latest version 4.9.18

6. Ensure the 4.9 updates are completed successfully retrieve the cluster version, run the following command:

   ```
   $ oc get clusterversion
   ```

   **Example output**

   ```
   OpenShift Container Platform 4.8.10 Updating clusters
   14
   ```
7. If necessary, upgrade OLM Operators using the Administrator perspective on the web console.

8. Update to 4.10, run the following command:

   $ oc adm upgrade --to-latest

9. Ensure the 4.10 update is completed successfully retrieve the cluster version, run the following command:

   $ oc get clusterversion

   **Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>AVAILABLE</th>
<th>PROGRESSING</th>
<th>SINCE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4.9.18</td>
<td>True</td>
<td>False</td>
<td>6m29s</td>
<td>Cluster version is 4.9.18</td>
</tr>
</tbody>
</table>

10. Unpause all previously paused MachineConfigPools, run the following command:

    $ oc patch mcp/worker --type merge --patch '{"spec":{"paused":false}}'

   **NOTE**

   If pools are not unparsed, the cluster is not permitted to update to any future minors and maintenance tasks such as certificate rotation are inhibited. This puts the cluster at risk for future degradation.

11. Verify that your previously paused pools have updated and your cluster completed the update to 4.10, run the following command:

    $ oc get mcp

   **Example output**

   Output is trimmed for clarity:

<table>
<thead>
<tr>
<th>NAME</th>
<th>CONFIG</th>
<th>UPDATED</th>
<th>UPDATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>rendered-master-52da4d2760807cb2b96a3402179a9a4c</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>worker</td>
<td>rendered-worker-4756f60eccae96fb9dcb4c392c69d497</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
CHAPTER 5. UPDATING A CLUSTER USING THE WEB CONSOLE

You can update, or upgrade, an OpenShift Container Platform cluster by using the web console. The following steps are updating a cluster within a minor a version. You can use the same instructions for updating a cluster between minor versions.

NOTE

Because of the difficulty of changing update channels by using `oc`, use the web console to change the update channel. It is recommended to complete the update process within the web console. You can follow the steps in Updating a cluster using the CLI to complete the update after you change to a 4.8 channel.

5.1. PREREQUISITES

- Have access to the cluster as a user with admin privileges. See Using RBAC to define and apply permissions.
- Have a recent etcd backup in case your update fails and you must restore your cluster to a previous state.
- Ensure all Operators previously installed through Operator Lifecycle Manager (OLM) are updated to their latest version in their latest channel. Updating the Operators ensures they have a valid update path when the default OperatorHub catalogs switch from the current minor version to the next during a cluster update. See Upgrading installed Operators for more information.
- Ensure that all machine config pools (MCPs) are running and not paused. Nodes associated with a paused MCP are skipped during the update process. You can pause the MCPs if you are performing a canary rollout update strategy.
- If your cluster uses manually maintained credentials, ensure that the Cloud Credential Operator (CCO) is in an upgradeable state. For more information, see Upgrading clusters with manually maintained credentials for AWS, Azure, or GCP.

IMPORTANT

- When an update is failing to complete, the Cluster Version Operator (CVO) reports the status of any blocking components while attempting to reconcile the update. Rolling your cluster back to a previous version is not supported. If your update is failing to complete, contact Red Hat support.
- Using the unsupportedConfigOverrides section to modify the configuration of an Operator is unsupported and might block cluster updates. You must remove this setting before you can update your cluster.

Additional resources

- Support policy for unmanaged Operators

5.2. PERFORMING A CANARY ROLLOUT UPDATE
In some specific use cases, you might want a more controlled update process where you do not want specific nodes updated concurrently with the rest of the cluster. These use cases include, but are not limited to:

- You have mission-critical applications that you do not want unavailable during the update. You can slowly test the applications on your nodes in small batches after the update.
- You have a small maintenance window that does not allow the time for all nodes to be updated, or you have multiple maintenance windows.

The rolling update process is not a typical update workflow. With larger clusters, it can be a time-consuming process that requires you execute multiple commands. This complexity can result in errors that can affect the entire cluster. It is recommended that you carefully consider whether your organization wants to use a rolling update and carefully plan the implementation of the process before you start.

The rolling update process described in this topic involves:

- Creating one or more custom machine config pools (MCPs).
- Labeling each node that you do not want to update immediately to move those nodes to the custom MCPs.
- Pausing those custom MCPs, which prevents updates to those nodes.
- Performing the cluster update.
- Unpausing one custom MCP, which triggers the update on those nodes.
- Testing the applications on those nodes to make sure the applications work as expected on those newly-updated nodes.
- Optionally removing the custom labels from the remaining nodes in small batches and testing the applications on those nodes.

**NOTE**

Pausing an MCP prevents the Machine Config Operator from applying any configuration changes on the associated nodes. Pausing an MCP also prevents any automatically-rotated certificates from being pushed to the associated nodes, including the automatic CA rotation of the `kube-apiserver-to-kubelet-signer` CA certificate. If the MCP is paused when the `kube-apiserver-to-kubelet-signer` CA certificate expires and the MCO attempts to automatically renew the certificate, the new certificate is created but not applied across the nodes in the respective machine config pool. This causes failure in multiple `oc` commands, including but not limited to `oc debug`, `oc logs`, `oc exec`, and `oc attach`. Pausing an MCP should be done with careful consideration about the `kube-apiserver-to-kubelet-signer` CA certificate expiration and for short periods of time only.

If you want to use the canary rollout update process, see [Performing a canary rollout update](#).

### 5.3. UPDATING A CLUSTER BY USING THE WEB CONSOLE

If updates are available, you can update your cluster from the web console.

You can find information about available OpenShift Container Platform advisories and updates in the [errata section](#) of the Customer Portal.
Prerequisites

- Have access to the web console as a user with **admin** privileges.

Procedure

1. From the web console, click **Administration → Cluster Settings** and review the contents of the **Details** tab.

2. For production clusters, ensure that the **Channel** is set to the correct channel for the version that you want to update to, such as **stable-4.8**.

   **IMPORTANT**
   
   For production clusters, you must subscribe to a **stable-*** or **fast-*** channel.

   - If the **Update status** is not **Updates available**, you cannot update your cluster.
   - **Select channel** indicates the cluster version that your cluster is running or is updating to.

3. Select a version to update to, and click **Save**.

   The input channel **Update status** changes to **Update to <product-version> in progress** and you can review the progress of the cluster update by watching the progress bars for the Operators and nodes.

   **NOTE**
   
   If you are upgrading your cluster to the next minor version, like version 4.y to 4. (y+1), it is recommended to confirm your nodes are updated before deploying workloads that rely on a new feature. Any pools with worker nodes that are not yet updated are displayed on the **Cluster Settings** page.

4. After the update completes and the Cluster Version Operator refreshes the available updates, check if more updates are available in your current channel.

   - If updates are available, continue to perform updates in the current channel until you can no longer update.

   - If no updates are available, change the **Channel** to the **stable-*** or **fast-*** channel for the next minor version, and update to the version that you want in that channel.

   You might need to perform several intermediate updates until you reach the version that you want.

### 5.4. Changing the Update Server by Using the Web Console

Changing the update server is optional. If you have an OpenShift Update Service (OSUS) installed and configured locally, you must set the URL for the server as the **upstream** to use the local server during updates.

Procedure

1. Navigate to **Administration → Cluster Settings**, click **version**.
2. Click the YAML tab and then edit the `upstream` parameter value:

   **Example output**

   ```yaml
   ... spec:
   clusterID: db93436d-7b05-42cc-b856-43e11ad2d31a
   upstream: '<update-server-url>'
   ...
   ```

1. The `<update-server-url>` variable specifies the URL for the update server.

   The default `upstream` is `https://api.openshift.com/api/upgrades_info/v1/graph`.

3. Click Save.
CHAPTER 6. UPDATING A CLUSTER USING THE CLI

You can update, or upgrade, an OpenShift Container Platform cluster within a minor version by using the OpenShift CLI (oc).

6.1. PREREQUISITES

- Have access to the cluster as a user with admin privileges. See Using RBAC to define and apply permissions.
- Have a recent etcd backup in case your update fails and you must restore your cluster to a previous state.
- Ensure all Operators previously installed through Operator Lifecycle Manager (OLM) are updated to their latest version in their latest channel. Updating the Operators ensures they have a valid update path when the default OperatorHub catalogs switch from the current minor version to the next during a cluster update. See Upgrading installed Operators for more information.
- Ensure that all machine config pools (MCPs) are running and not paused. Nodes associated with a paused MCP are skipped during the update process. You can pause the MCPs if you are performing a canary rollout update strategy.
- If your cluster uses manually maintained credentials, ensure that the Cloud Credential Operator (CCO) is in an upgradeable state. For more information, see Upgrading clusters with manually maintained credentials for AWS, Azure, or GCP.

IMPORTANT

- When an update is failing to complete, the Cluster Version Operator (CVO) reports the status of any blocking components while attempting to reconcile the update. Rolling your cluster back to a previous version is not supported. If your update is failing to complete, contact Red Hat support.
- Using the unsupportedConfigOverrides section to modify the configuration of an Operator is unsupported and might block cluster updates. You must remove this setting before you can update your cluster.

Additional resources

- Support policy for unmanaged Operators

6.2. UPDATING A CLUSTER BY USING THE CLI

If updates are available, you can update your cluster by using the OpenShift CLI (oc).

You can find information about available OpenShift Container Platform advisories and updates in the errata section of the Customer Portal.

Prerequisites

- Install the OpenShift CLI (oc) that matches the version for your updated version.
- Log in to the cluster as user with cluster-admin privileges.
• Install the `jq` package.

Procedure

1. Ensure that your cluster is available:

   ```
   $ oc get clusterversion
   ``

   **Example output**

   ```
   NAME VERSION AVAILABLE PROGRESSING SINCE STATUS
   version 4.6.9 True False 158m Cluster version is 4.6.9
   ```

2. Review the current update channel information and confirm that your channel is set to `stable-4.8`:

   ```
   $ oc get clusterversion -o json | jq ".items[0].spec"
   ``

   **Example output**

   ```
   {
     "channel": "stable-4.8",
     "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff"
   }
   ```

   **IMPORTANT**

   For production clusters, you must subscribe to a `stable-*` or `fast-*` channel.

3. View the available updates and note the version number of the update that you want to apply:

   ```
   $ oc adm upgrade
   ``

   **Example output**

   Cluster version is 4.1.0

   Updates:

   ```
   VERSION IMAGE
   4.1.2 quay.io/openshift-release-dev/ocp-release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b
   ```

4. Apply an update:

   • To update to the latest version:

     ```
     $ oc adm upgrade --to-latest=true 1
     ```

   • To update to a specific version:
5. Review the status of the Cluster Version Operator:

$ oc get clusterversion -o json | jq ".items[0].spec"

Example output

```json
{
  "channel": "stable-4.8",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
  "desiredUpdate": {
    "force": false,
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",
    "version": "4.8.0"
  },
}
```

If the `version` number in the `desiredUpdate` stanza matches the value that you specified, the update is in progress.

6. Review the cluster version status history to monitor the status of the update. It might take some time for all the objects to finish updating.

$ oc get clusterversion -o json | jq ".items[0].status.history"

Example output

```json
[
  {
    "completionTime": null,
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b7",
    "startedTime": "2021-01-28T20:30:50Z",
    "state": "Partial",
    "verified": true,
    "version": "4.8.0"
  },
  {
    "completionTime": "2021-01-28T20:30:50Z",
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b7",
    "startedTime": "2021-01-28T17:38:10Z",
    "state": "Completed",
    "verified": false,
  }
]
The history contains a list of the most recent versions applied to the cluster. This value is updated when the CVO applies an update. The list is ordered by date, where the newest update is first in the list. Updates in the history have state Completed if the rollout completed and Partial if the update failed or did not complete.

7. After the update completes, you can confirm that the cluster version has updated to the new version:

```bash
$ oc get clusterversion
```

**Example output**

```
NAME      VERSION     AVAILABLE   PROGRESSING   SINCE     STATUS
version   4.8.0       True        False         2m        Cluster version is 4.8.0
```

8. If you are upgrading your cluster to the next minor version, like version 4.y to 4.(y+1), it is recommended to confirm your nodes are upgraded before deploying workloads that rely on a new feature:

```bash
$ oc get nodes
```

**Example output**

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.21.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.21.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.21.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.21.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.21.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.21.0
```

### 6.3. CHANGING THE UPDATE SERVER BY USING THE CLI

Changing the update server is optional. If you have an OpenShift Update Service (OSUS) installed and configured locally, you must set the URL for the server as the upstream to use the local server during updates. The default value for upstream is `https://api.openshift.com/api/upgrades_info/v1/graph`.

**Procedure**

- Change the `upstream` parameter value in the cluster version:

  ```bash
  $ oc patch clusterversion/version --patch '{"spec":{"upstream":"<update-server-url>"}}' --type=merge
  
  ```

  The `<update-server-url>` variable specifies the URL for the update server.

  **Example output**
clusterversion.config.openshift.io/version patched
CHAPTER 7. PERFORMING A CANARY ROLLOUT UPDATE

There might be some scenarios where you want a more controlled rollout of an update to the worker nodes in order to ensure that mission-critical applications stay available during the whole update, even if the update process causes your applications to fail. Depending on your organizational needs, you might want to update a small subset of worker nodes, evaluate cluster and workload health over a period of time, then update the remaining nodes. This is commonly referred to as a canary update. Or, you might also want to fit worker node updates, which often require a host reboot, into smaller defined maintenance windows when it is not possible to take a large maintenance window to update the entire cluster at one time.

In these scenarios, you can create multiple custom machine config pools (MCPs) to prevent certain worker nodes from updating when you update the cluster. After the rest of the cluster is updated, you can update those worker nodes in batches at appropriate times.

For example, if you have a cluster with 100 nodes with 10% excess capacity, maintenance windows that must not exceed 4 hours, and you know that it takes no longer than 8 minutes to drain and reboot a worker node, you can leverage MCPs to meet your goals. For example, you could define four MCPs, named `workerpool-canary`, `workerpool-A`, `workerpool-B`, and `workerpool-C`, with 10, 30, 30, and 30 nodes respectively.

During your first maintenance window, you would pause the MCP for `workerpool-A`, `workerpool-B`, and `workerpool-C`, then initiate the cluster update. This updates components that run on top of OpenShift Container Platform and the 10 nodes which are members of the `workerpool-canary` MCP, because that pool was not paused. The other three MCPs are not updated, because they were paused. If for some reason, you determine that your cluster or workload health was negatively affected by the `workerpool-canary` update, you would then cordon and drain all nodes in that pool while still maintaining sufficient capacity until you have diagnosed the problem. When everything is working as expected, you would then evaluate the cluster and workload health before deciding to un-pause, and thus update, `workerpool-A`, `workerpool-B`, and `workerpool-C` in succession during each additional maintenance window.

While managing worker node updates using custom MCPs provides flexibility, it can be a time-consuming process that requires you execute multiple commands. This complexity can result in errors that can affect the entire cluster. It is recommended that you carefully consider your organizational needs and carefully plan the implementation of the process before you start.

**NOTE**

It is not recommended to update the MCPs to different OpenShift Container Platform versions. For example, do not update one MCP from 4.y.10 to 4.y.11 and another to 4.y.12. This scenario has not been tested and might result in an undefined cluster state.

**IMPORTANT**

Pausing a machine config pool prevents the Machine Config Operator from applying any configuration changes on the associated nodes. Pausing an MCP also prevents any automatically-rotated certificates from being pushed to the associated nodes, including the automatic CA rotation of the `kube-apiserver-to-kubelet-signer` CA certificate. If the MCP is paused when the `kube-apiserver-to-kubelet-signer` CA certificate expires and the MCO attempts to automatically renew the certificate, the new certificate is created but not applied across the nodes in the respective machine config pool. This causes failure in multiple `oc` commands, including but not limited to `oc debug`, `oc logs`, `oc exec`, and `oc attach`. Pausing an MCP should be done with careful consideration about the `kube-apiserver-to-kubelet-signer` CA certificate expiration and for short periods of time only.
7.1. ABOUT THE CANARY ROLLOUT UPDATE PROCESS AND MCPS

In OpenShift Container Platform, nodes are not considered individually. Nodes are grouped into machine config pools (MCP). There are two MCPs in a default OpenShift Container Platform cluster: one for the control plane nodes and one for the worker nodes. An OpenShift Container Platform update affects all MCPs concurrently.

During the update, the Machine Config Operator (MCO) drains and cordons all nodes within a MCP up to the specified maxUnavailable number of nodes (if specified), by default 1. Draining and cordonning a node deschedules all pods on the node and marks the node as unschedulable. After the node is drained, the Machine Config Daemon applies a new machine configuration, which can include updating the operating system (OS). Updating the OS requires the host to reboot.

To prevent specific nodes from being updated, and thus, not drained, cordoned, and updated, you can create custom MCPs. Then, pause those MCPs to ensure that the nodes associated with those MCPs are not updated. The MCO does not update any paused MCPs. You can create one or more custom MCPs, which can give you more control over the sequence in which you update those nodes. After you update the nodes in the first MCP, you can verify the application compatibility, and then update the rest of the nodes gradually to the new version.

**NOTE**

To ensure the stability of the control plane, creating a custom MCP from the control plane nodes (also known as the master nodes) is not supported. The Machine Config Operator (MCO) ignores any custom MCP created for the control plane nodes.

You should give careful consideration to the number of MCPs you create and the number of nodes in each MCP, based on your workload deployment topology. For example, if you need to fit updates into specific maintenance windows, you need to know how many nodes that OpenShift Container Platform can update within a window. This number is dependent on your unique cluster and workload characteristics.

Also, you need to consider how much extra capacity you have available in your cluster. For example, in the case where your applications fail to work as expected on the updated nodes, you can cordon and drain those nodes in the pool, which moves the application pods to other nodes. You need to consider how much extra capacity you have available in order to determine the number of custom MCPs you need and how many nodes are in each MCP. For example, if you use two custom MCPs and 50% of your nodes are in each pool, you need to determine if running 50% of your nodes would provide sufficient quality-of-service (QoS) for your applications.

You can use this update process with all documented OpenShift Container Platform update processes. However, the process does not work with Red Hat Enterprise Linux (RHEL) machines, which are updated using Ansible playbooks.

7.2. ABOUT PERFORMING A CANARY ROLLOUT UPDATE

This topic describes the general workflow of this canary rollout update process. The steps to perform each task in the workflow are described in the following sections.

1. Create MCPs based on the worker pool. The number of nodes in each MCP depends on a few factors, such as your maintenance window duration for each MCP, and the amount of reserve capacity, meaning extra worker nodes, available in your cluster.
NOTE
You can change the maxUnavailable setting in an MCP to specify the percentage or the number of machines that can be updating at any given time. The default is 1.

2. Add a node selector to the custom MCPs. For each node that you do not want to update simultaneously with the rest of the cluster, add a matching label to the nodes. This label associates the node to the MCP.

NOTE
Do not remove the default worker label from the nodes. The nodes must have a role label to function properly in the cluster.

3. Pause the MCPs you do not want to update as part of the update process.

NOTE
Pausing the MCP also pauses the kube-apiserver-to-kubelet-signer automatic CA certificates rotation. New CA certificates are generated at 292 days from the installation date and old certificates are removed 365 days from the installation date. See the Understand CA cert auto renewal in Red Hat OpenShift 4 to find out how much time you have before the next automatic CA certificate rotation. Make sure the pools are unpause when the CA cert rotation happens. If the MCPs are paused, the cert rotation does not happen, which causes the cluster to become degraded and causes failure in multiple oc commands, including but not limited to oc debug, oc logs, oc exec, and oc attach.

4. Perform the cluster update. The update process updates the MCPs that are not paused, including the control plane nodes (also known as the master nodes).

5. Test the applications on the updated nodes to ensure they are working as expected.

6. Unpause the remaining MCPs one-by-one and test the applications on those nodes until all worker nodes are updated. Unpausing an MCP starts the update process for the nodes associated with that MCP. You can check the progress of the update from the web console by clicking Administration → Cluster settings. Or, use the oc get machineconfigpools CLI command.

7. Optionally, remove the custom label from updated nodes and delete the custom MCPs.

7.3. CREATING MACHINE CONFIG POOLS TO PERFORM A CANARY ROLLOUT UPDATE

The first task in performing this canary rollout update is to create one or more machine config pools (MCP).

1. Create an MCP from a worker node.
   a. List the worker nodes in your cluster.

   $ oc get -l 'node-role.kubernetes.io/master!=' -o jsonpath={range .items[*]}{.metadata.name}{"\n"}{end} nodes
Example output

| ci-ln-pwnll6b-f76d1-s8t9n-worker-a-s75z4 |
| ci-ln-pwnll6b-f76d1-s8t9n-worker-b-dglj2 |
| ci-ln-pwnll6b-f76d1-s8t9n-worker-c-lddbm |

b. For the nodes you want to delay, add a custom label to the node:

$ oc label node <node name> node-role.kubernetes.io/<custom-label>=

For example:

$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-role.kubernetes.io/workerpool-canary=

Example output

node/ci-ln-gtrwm8t-f76d1-spbl7-worker-a-xk76k labeled

c. Create the new MCP:

```json
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: workerpool-canary
spec:
  machineConfigSelector:
    matchExpressions:
    - {
      key: machineconfiguration.openshift.io/role,
      operator: In,
      values: [worker,workerpool-canary]
    }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/workerpool-canary: ""
```

1 Specify a name for the MCP.
2 Specify the worker and custom MCP name.
3 Specify the custom label you added to the nodes that you want in this pool.

$ oc create -f <file_name>

Example output

machineconfigpool.machineconfiguration.openshift.io/workerpool-canary created

d. View the list of MCPs in the cluster and their current state:
$ oc get machineconfigpool

Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>CONFIG</th>
<th>UPDATED</th>
<th>UPDATING</th>
<th>DEGRADED</th>
<th>MACHINECOUNT</th>
<th>READYMACHINECOUNT</th>
<th>UPDATEDMACHINECOUNT</th>
<th>DEGRADEDMACHINECOUNT</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>rendered-master-b0bb90c4921860f2a5d8a2f8137c1867</td>
<td>True</td>
<td>False</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>97m</td>
<td></td>
</tr>
<tr>
<td>workerpool-canary</td>
<td>rendered-workerpool-canary-87ba3dec1ad78cb6aeebf7fbb476a36</td>
<td>True</td>
<td>False</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2m42s</td>
</tr>
<tr>
<td>worker</td>
<td>rendered-worker-87ba3dec1ad78cb6aeebf7fbb476a36</td>
<td>True</td>
<td>False</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>97m</td>
<td></td>
</tr>
</tbody>
</table>

The new machine config pool, workerpool-canary, is created and the number of nodes to which you added the custom label are shown in the machine counts. The worker MCP machine counts are reduced by the same number. It can take several minutes to update the machine counts. In this example, one node was moved from the worker MCP to the workerpool-canary MCP.

### 7.4. PAUSING THE MACHINE CONFIG POOLS

In this canary rollout update process, after you label the nodes that you do not want to update with the rest of your OpenShift Container Platform cluster and create the machine config pools (MCPs), you pause those MCPs. Pausing an MCP prevents the Machine Config Operator (MCO) from updating the nodes associated with that MCP.

**NOTE**

Pausing the MCP also pauses the kube-apiserver-to-kubelet-signer automatic CA certificates rotation. New CA certificates are generated at 292 days from the installation date and old certificates are removed 365 days from the installation date. See the [Understand CA cert auto renewal in Red Hat OpenShift 4](#) to find out how much time you have before the next automatic CA certificate rotation. Make sure the pools are unpause when the CA cert rotation happens. If the MCPs are paused, the cert rotation does not happen, which causes the cluster to become degraded and causes failure in multiple oc commands, including but not limited to oc debug, oc logs, oc exec, and oc attach.

To pause an MCP:

1. Patch the MCP that you want paused:

   ```
   $ oc patch mcp/<mcp_name> --patch '{"spec": {"paused": true}}' --type=merge
   ```

   For example:

   ```
   $ oc patch mcp/workerpool-canary --patch '{"spec": {"paused": true}}' --type=merge
   ```

   **Example output**

   ```
   machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
   ```
7.5. PERFORMING THE CLUSTER UPDATE

When the MCPs enter ready state, you can perform the cluster update. See one of the following update methods, as appropriate for your cluster:

- Updating a cluster using the web console
- Updating a cluster using the CLI

After the update is complete, you can start to unpause the MCPs one-by-one.

7.6. UNPAUSING THE MACHINE CONFIG POOLS

In this canary rollout update process, after the OpenShift Container Platform update is complete, unpause your custom MCPs one-by-one. Unpausing an MCP allows the Machine Config Operator (MCO) to update the nodes associated with that MCP.

To unpause an MCP:

1. Patch the MCP that you want to unpause:

   ```
   $ oc patch mcp/<mcp_name> --patch '{"spec":{"paused":false}}' --type=merge
   
   For example:
   
   $ oc patch mcp/workerpool-canary --patch '{"spec":{"paused":false}}' --type=merge
   
   Example output
   
   machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
   
   You can check the progress of the update by using the `oc get machineconfigpools` command.
   
   2. Test your applications on the updated nodes to ensure that they are working as expected.
   
   3. Unpause any other paused MCPs one-by-one and verify that your applications work.

7.6.1. In case of application failure

In case of a failure, such as your applications not working on the updated nodes, you can cordon and drain the nodes in the pool, which moves the application pods to other nodes to help maintain the quality-of-service for the applications. This first MCP should be no larger than the excess capacity.

7.7. MOVING A NODE TO THE ORIGINAL MACHINE CONFIG POOL

In this canary rollout update process, after you have unpaused a custom machine config pool (MCP) and verified that the applications on the nodes associated with that MCP are working as expected, you should move the node back to its original MCP by removing the custom label you added to the node.

**IMPORTANT**

A node must have a role to be properly functioning in the cluster.
To move a node to its original MCP:

1. Remove the custom label from the node.

   $ oc label node <node_name> node-role.kubernetes.io/<custom-label>-

   For example:

   $ oc label node ci-ln-0qv1yp2-176d1-kl2tq-worker-a-j2ssz node-role.kubernetes.io/workerpool-canary-

   **Example output**

   node/ci-ln-0qv1yp2-176d1-kl2tq-worker-a-j2ssz labeled

   The MCO moves the nodes back to the original MCP and reconciles the node to the MCP configuration.

2. View the list of MCPs in the cluster and their current state:

   $oc get mcp

<table>
<thead>
<tr>
<th>NAME</th>
<th>CONFIG</th>
<th>UPDATED</th>
<th>UPDATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>rendered-master-1203f157d053fd987c7cbdd1e3fbc0ed</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>workerpool-canary</td>
<td>rendered-mcp-noupdate-5ad4791166c468f3a35cd16e734c9028</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>worker</td>
<td>rendered-worker-5ad4791166c468f3a35cd16e734c9028</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

   The node is removed from the custom MCP and moved back to the original MCP. It can take several minutes to update the machine counts. In this example, one node was moved from the removed *workerpool-canary* MCP to the `worker` MCP.

3. Optional: Delete the custom MCP:

   $ oc delete mcp <mcp_name>
CHAPTER 8. UPDATING A CLUSTER THAT INCLUDES RHEL COMPUTE MACHINES

You can update, or upgrade, an OpenShift Container Platform cluster. If your cluster contains Red Hat Enterprise Linux (RHEL) machines, you must perform more steps to update those machines.

8.1. PREREQUISITES

- Have access to the cluster as a user with admin privileges. See Using RBAC to define and apply permissions.
- Have a recent etcd backup in case your update fails and you must restore your cluster to a previous state.
- If your cluster uses manually maintained credentials, ensure that the Cloud Credential Operator (CCO) is in an upgradeable state. For more information, see Upgrading clusters with manually maintained credentials for AWS, Azure, or GCP.

Additional resources

- Support policy for unmanaged Operators

8.2. UPDATING A CLUSTER BY USING THE WEB CONSOLE

If updates are available, you can update your cluster from the web console.

You can find information about available OpenShift Container Platform advisories and updates in the errata section of the Customer Portal.

Prerequisites

- Have access to the web console as a user with admin privileges.

Procedure

1. From the web console, click Administration → Cluster Settings and review the contents of the Details tab.

2. For production clusters, ensure that the Channel is set to the correct channel for the version that you want to update to, such as stable-4.8.

   IMPORTANT

   For production clusters, you must subscribe to a stable-* or fast-* channel.

   - If the Update status is not Updates available, you cannot update your cluster.
   - Select channel indicates the cluster version that your cluster is running or is updating to.

3. Select a version to update to, and click Save.
   The Input channel Update status changes to Update to <product-version> in progress and you can review the progress of the cluster update by watching the progress bars for the Operators and nodes.
NOTE

If you are upgrading your cluster to the next minor version, like version 4.y to 4.(y+1), it is recommended to confirm your nodes are updated before deploying workloads that rely on a new feature. Any pools with worker nodes that are not yet updated are displayed on the **Cluster Settings** page.

4. After the update completes and the Cluster Version Operator refreshes the available updates, check if more updates are available in your current channel.
   
   - If updates are available, continue to perform updates in the current channel until you can no longer update.
   
   - If no updates are available, change the **Channel** to the **stable-** or **fast-** channel for the next minor version, and update to the version that you want in that channel.

You might need to perform several intermediate updates until you reach the version that you want.

NOTE

When you update a cluster that contains Red Hat Enterprise Linux (RHEL) worker machines, those workers temporarily become unavailable during the update process. You must run the update playbook against each RHEL machine as it enters the **NotReady** state for the cluster to finish updating.

### 8.3. OPTIONAL: ADDING HOOKS TO PERFORM ANSIBLE TASKS ON RHEL MACHINES

You can use hooks to run Ansible tasks on the RHEL compute machines during the OpenShift Container Platform update.

#### 8.3.1. About Ansible hooks for upgrades

When you update OpenShift Container Platform, you can run custom tasks on your Red Hat Enterprise Linux (RHEL) nodes during specific operations by using hooks. Hooks allow you to provide files that define tasks to run before or after specific update tasks. You can use hooks to validate or modify custom infrastructure when you update the RHEL compute nodes in your OpenShift Container Platform cluster.

Because when a hook fails, the operation fails, you must design hooks that are idempotent, or can run multiple times and provide the same results.

Hooks have the following important limitations:

- Hooks do not have a defined or versioned interface. They can use internal `openshift-ansible` variables, but it is possible that the variables will be modified or removed in future OpenShift Container Platform releases.
- Hooks do not have error handling, so an error in a hook halts the update process. If you get an error, you must address the problem and then start the upgrade again.

#### 8.3.2. Configuring the Ansible inventory file to use hooks

You define the hooks to use when you update the Red Hat Enterprise Linux (RHEL) compute machines, which are also known as worker machines, in the **hosts** inventory file under the **all:vars** section.
## Prerequisites

- You have access to the machine that you used to add the RHEL compute machines cluster. You must have access to the `hosts` Ansible inventory file that defines your RHEL machines.

## Procedure

1. After you design the hook, create a YAML file that defines the Ansible tasks for it. This file must be a set of tasks and cannot be a playbook, as shown in the following example:

   ```yaml
   # Trivial example forcing an operator to acknowledge the start of an upgrade
   # file=/home/user/openshift-ansible/hooks/pre_compute.yml
   - name: note the start of a compute machine update
     debug:
       msg: "Compute machine upgrade of {{ inventory_hostname }} is about to start"
   - name: require the user agree to start an upgrade
     pause:
       prompt: "Press Enter to start the compute machine update"
   
   # Example hook definitions in an inventory file
   [all:vars]
   openshift_node_pre_upgrade_hook=/home/user/openshift-ansible/hooks/pre_node.yml
   openshift_node_post_upgrade_hook=/home/user/openshift-ansible/hooks/post_node.yml
   ```

2. Modify the `hosts` Ansible inventory file to specify the hook files. The hook files are specified as parameter values in the `[all:vars]` section, as shown:

   ```yaml
   [all:vars]
   openshift_node_pre_upgrade_hook=/home/user/openshift-ansible/hooks/pre_node.yml
   openshift_node_post_upgrade_hook=/home/user/openshift-ansible/hooks/post_node.yml
   ```

   To avoid ambiguity in the paths to the hook, use absolute paths instead of a relative paths in their definitions.

## 8.3.3. Available hooks for RHEL compute machines

You can use the following hooks when you update the Red Hat Enterprise Linux (RHEL) compute machines in your OpenShift Container Platform cluster.

<table>
<thead>
<tr>
<th>Hook name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>openshift_node_pre_cordon_hook</code></td>
<td>- Runs before each node is cordoned.</td>
</tr>
<tr>
<td></td>
<td>- This hook runs against each node in serial.</td>
</tr>
<tr>
<td></td>
<td>- If a task must run against a different host, the task must use <code>delegate_to</code> or <code>local_action</code>.</td>
</tr>
<tr>
<td>Hook name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| openshift_node_pre_upgrade_hook               | - Runs after each node is cordoned but before it is updated.  
- This hook runs against each node in serial.  
- If a task must run against a different host, the task must use `delegate_to` or `local_action`.                                                                                                                                 |
| openshift_node_pre_uncordon_hook             | - Runs after each node is updated but before it is uncordoned.  
- This hook runs against each node in serial.  
- If a task must run against a different host, they task must use `delegate_to` or `local_action`.                                                                                                                                 |
| openshift_node_post_upgrade_hook             | - Runs after each node uncordoned. It is the last node update action.  
- This hook runs against each node in serial.  
- If a task must run against a different host, the task must use `delegate_to` or `local_action`.                                                                                                                                 |

8.4. UPDATING RHEL COMPUTE MACHINES IN YOUR CLUSTER

After you update your cluster, you must update the Red Hat Enterprise Linux (RHEL) compute machines in your cluster.

**IMPORTANT**

Because only Red Hat Enterprise Linux (RHEL) version 7.9 or later is supported for worker (compute) machines, you must not upgrade the RHEL worker machines to version 8.

You can also update your compute machines to another minor version of OpenShift Container Platform if you are using RHEL as the operating system. You do not need to exclude any RPM packages from RHEL when performing a minor version update.

**Prerequisites**

- You updated your cluster.
IMPORTANT

Because the RHEL machines require assets that are generated by the cluster to complete the update process, you must update the cluster before you update the RHEL worker machines in it.

- You have access to the local machine that you used to add the RHEL compute machines to your cluster. You must have access to the hosts Ansible inventory file that defines your RHEL machines and the upgrade playbook.

- For updates to a minor version, the RPM repository is using the same version of OpenShift Container Platform that is running on your cluster.

Procedure

1. Stop and disable firewalld on the host:

   # systemctl disable --now firewalld.service

   **NOTE**

   By default, the base OS RHEL with "Minimal" installation option enables firewalld service. Having the firewalld service enabled on your host prevents you from accessing OpenShift Container Platform logs on the worker. Do not enable firewalld later if you wish to continue accessing OpenShift Container Platform logs on the worker.

2. Enable the repositories that are required for OpenShift Container Platform 4.8:

   a. On the machine that you run the Ansible playbooks, update the required repositories:

      # subscription-manager repos --disable=rhel-7-server-ose-4.7-rpms \  
      --enable=rhel-7-server-ansible-2.9-rpms \  
      --enable=rhel-7-server-ose-4.8-rpms

   b. On the machine that you run the Ansible playbooks, update the required packages, including openshift-ansible:

      # yum update openshift-ansible openshift-clients

   c. On each RHEL compute node, update the required repositories:

      # subscription-manager repos --disable=rhel-7-server-ose-4.7-rpms \  
      --enable=rhel-7-server-ose-4.8-rpms \  
      --enable=rhel-7-fast-datapath-rpms \  
      --enable=rhel-7-server-optional-rpms

3. Update a RHEL worker machine:

   a. Review the current node status to determine which RHEL worker to update:

      # oc get node
Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>mycluster-control-plane-0</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-control-plane-1</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-control-plane-2</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-rhel7-0</td>
<td>NotReady,SchedulingDisabled</td>
<td>worker</td>
<td>98m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-rhel7-1</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-rhel7-2</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>mycluster-rhel7-3</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.21.0</td>
</tr>
</tbody>
</table>

Note which machine has the **NotReady,SchedulingDisabled** status.

b. Review your Ansible inventory file at `/<path>/inventory/hosts` and update its contents so that only the machine with the **NotReady,SchedulingDisabled** status is listed in the [workers] section, as shown in the following example:

```yaml
[all:vars]
ansible_user=root
#ansible_become=True
openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel7-0.example.com
```

c. Change to the `openshift-ansible` directory:

```
$ cd /usr/share/ansible/openshift-ansible
```

d. Run the `upgrade` playbook:

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/upgrade.yml
```

For `<path>`, specify the path to the Ansible inventory file that you created.

**NOTE**

The **upgrade** playbook only upgrades the OpenShift Container Platform packages. It does not update the operating system packages.

4. Follow the process in the previous step to update each RHEL worker machine in your cluster.

5. After you update all of the workers, confirm that all of your cluster nodes have updated to the new version:

```
# oc get node
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>mycluster-control-plane-0</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.21.0</td>
</tr>
</tbody>
</table>
6. Optional: Update the operating system packages that were not updated by the `upgrade` playbook. To update packages that are not on 4.8, use the following command:

```
# yum update
```

**NOTE**

You do not need to exclude RPM packages if you are using the same RPM repository that you used when you installed 4.8.
CHAPTER 9. UPDATING A RESTRICTED NETWORK CLUSTER

You can update a restricted network OpenShift Container Platform cluster by using the `oc` command-line interface (CLI) or using the OpenShift Update Service.

9.1. UPDATING A RESTRICTED NETWORK CLUSTER USING THE CLI

You can update a restricted network OpenShift Container Platform cluster by using the `oc` command-line interface (CLI).

A restricted network environment is the one in which your cluster nodes cannot access the internet. For this reason, you must populate a registry with the installation images. If your registry host cannot access both the internet and the cluster, you can mirror the images to a file system that disconnected from that environment and then bring that host or removable media across that gap. If the local container registry and the cluster are connected to the mirror registry’s host, you can directly push the release images to the local registry.

If multiple clusters are present within the restricted network, mirror the required release images to a single container image registry and use that registry to update all the clusters.

9.1.1. Prerequisites

- Have access to the internet to obtain the necessary container images.
- Have write access to a container registry in the restricted-network environment to push and pull images. The container registry must be compatible with Docker registry API v2.
- You must have the `oc` command-line interface (CLI) tool installed.
- Have access to the cluster as a user with `admin` privileges. See Using RBAC to define and apply permissions.
- Have a recent `etcd backup` in case your update fails and you must restore your cluster to a previous state.
- Ensure that all machine config pools (MCPs) are running and not paused. Nodes associated with a paused MCP are skipped during the update process. You can pause the MCPs if you are performing a canary rollout update strategy.
- If your cluster uses manually maintained credentials, ensure that the Cloud Credential Operator (CCO) is in an upgradeable state. For more information, see Upgrading clusters with manually maintained credentials for AWS, Azure, or GCP.

9.1.2. Preparing your mirror host

Before you perform the mirror procedure, you must prepare the host to retrieve content and push it to the remote location.

9.1.2.1. Installing the OpenShift CLI by downloading the binary

You can install the OpenShift CLI (`oc`) to interact with OpenShift Container Platform from a command-line interface. You can install `oc` on Linux, Windows, or macOS.
IMPORTANT

If you installed an earlier version of `oc`, you cannot use it to complete all of the commands in OpenShift Container Platform 4.8. Download and install the new version of `oc`. If you are upgrading a cluster in a restricted network, install the `oc` version that you plan to upgrade to.

Installing the OpenShift CLI on Linux

You can install the OpenShift CLI (`oc`) binary on Linux by using the following procedure.

Procedure

2. Select the appropriate version in the Version drop-down menu.
3. Click **Download Now** next to the **OpenShift v4.8 Linux Client** entry and save the file.
4. Unpack the archive:
   ```
   $ tar xzvf <file>
   ```
5. Place the `oc` binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:
   ```
   $ echo $PATH
   ```

After you install the OpenShift CLI, it is available using the `oc` command:

```
$ oc <command>
```

Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (`oc`) binary on Windows by using the following procedure.

Procedure

2. Select the appropriate version in the Version drop-down menu.
3. Click **Download Now** next to the **OpenShift v4.8 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the `oc` binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:
   ```
   C:\> path
   ```

After you install the OpenShift CLI, it is available using the `oc` command:
Installing the OpenShift CLI on macOS
You can install the OpenShift CLI (oc) binary on macOS by using the following procedure.

Procedure

2. Select the appropriate version in the Version drop-down menu.
3. Click Download Now next to the OpenShift v4.8 MacOSX Client entry and save the file.
4. Unpack and unzip the archive.
5. Move the oc binary to a directory on your PATH. To check your PATH, open a terminal and execute the following command:

   $ echo $PATH

After you install the OpenShift CLI, it is available using the oc command:

   $ oc <command>

9.1.3. Configuring credentials that allow images to be mirrored

Create a container image registry credentials file that allows mirroring images from Red Hat to your mirror.

WARNING
Do not use this image registry credentials file as the pull secret when you install a cluster. If you provide this file when you install cluster, all of the machines in the cluster will have write access to your mirror registry.

WARNING
This process requires that you have write access to a container image registry on the mirror registry and adds the credentials to a registry pull secret.

Prerequisites

- You configured a mirror registry to use in your restricted network.
You identified an image repository location on your mirror registry to mirror images into.

You provisioned a mirror registry account that allows images to be uploaded to that image repository.

Procedure

Complete the following steps on the installation host:

1. Download your registry.redhat.io pull secret from the Red Hat OpenShift Cluster Manager and save it to a .json file.

2. Generate the base64-encoded user name and password or token for your mirror registry:

   $ echo -n '<user_name>:<password>' | base64 -w0
   BGVtbYk3ZHAtqXs=

   For <user_name> and <password>, specify the user name and password that you configured for your registry.

3. Make a copy of your pull secret in JSON format:

   $ cat ./pull-secret.text | jq . > <path>/<pull_secret_file_in_json>

   Specify the path to the folder to store the pull secret in and a name for the JSON file that you create.

   The contents of the file resemble the following example:

   ```json
   {
     "auths": {
       "cloud.openshift.com": {
         "auth": "b3BlbnNo...",
         "email": "you@example.com"
       },
       "quay.io": {
         "auth": "b3BlbnNo...",
         "email": "you@example.com"
       },
       "registry.connect.redhat.com": {
         "auth": "NTE3Njg5Nj...",
         "email": "you@example.com"
       },
       "registry.redhat.io": {
         "auth": "NTE3Njg5Nj...",
         "email": "you@example.com"
       }
     }
   }
   ```

4. Edit the new file and add a section that describes your registry to it:

   ```json
   "auths": {
     "<mirror_registry>": {
   ```
For `<mirror_registry>`, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, `registry.example.com` or `registry.example.com:5000`.

For `<credentials>`, specify the base64-encoded user name and password for the mirror registry.

The file resembles the following example:

```
{
  "auths": {
    "registry.example.com": {
      "auth": "BGVtbYk3ZHAtqXs=",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

### 9.1.4. Mirroring the OpenShift Container Platform image repository

Before you update a cluster on infrastructure that you provision in a restricted network, you must mirror the required container images into that environment. You can also use this procedure in unrestricted networks to ensure your clusters only use container images that have satisfied your organizational controls on external content.

**Procedure**

1. Use the [Red Hat OpenShift Container Platform Upgrade Graph visualizer and update planner](https://examples.redhat.com/openshift-upgrade-graph/) to plan an update from one version to another. The OpenShift Upgrade Graph provides channel graphs and a way to confirm that there is an update path between your current and intended cluster versions.

2. Set the required environment variables:
a. Export the release version:

```bash
$ export OCP_RELEASE={<release_version>}
```

For `<release_version>`, specify the tag that corresponds to the version of OpenShift Container Platform to which you want to update, such as `4.5.4`.

b. Export the local registry name and host port:

```bash
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

For `<local_registry_host_name>`, specify the registry domain name for your mirror repository, and for `<local_registry_host_port>`, specify the port that it serves content on.

c. Export the local repository name:

```bash
$ LOCAL_REPOSITORY='<local_repository_name>'
```

For `<local_repository_name>`, specify the name of the repository to create in your registry, such as `ocp4/openshift4`.

d. Export the name of the repository to mirror:

```bash
$ PRODUCT_REPO='openshift-release-dev'
```

For a production release, you must specify `openshift-release-dev`.

e. Export the path to your registry pull secret:

```bash
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

For `<path_to_pull_secret>`, specify the absolute path to and file name of the pull secret for your mirror registry that you created.

**NOTE**

If your cluster uses an `ImageContentSourcePolicy` object to configure repository mirroring, you can use only global pull secrets for mirrored registries. You cannot add a pull secret to a project.

f. Export the release mirror:

```bash
$ RELEASE_NAME="ocp-release"
```

For a production release, you must specify `ocp-release`.

g. Export the type of architecture for your server, such as `x86_64`:

```bash
$ ARCHITECTURE=<server_architecture>
```

h. Export the path to the directory to host the mirrored images:

```bash
$ REMOVABLE_MEDIA_PATH=<path> ①
```
Specify the full path, including the initial forward slash (/) character.

3. Review the images and configuration manifests to mirror:

   $ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/$(PRODUCT_REPO)/$(RELEASE_NAME):$(OCP_RELEASE)-$(ARCHITECTURE)
--dry-run

4. Mirror the version images to the mirror registry.
   - If your mirror host does not have internet access, take the following actions:
     i. Connect the removable media to a system that is connected to the internet.
     ii. Mirror the images and configuration manifests to a directory on the removable media:

        $ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/$(PRODUCT_REPO)/$(RELEASE_NAME):$(OCP_RELEASE)-$(ARCHITECTURE)

     iii. Take the media to the restricted network environment and upload the images to the
         local container registry.

        $ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:$(OCP_RELEASE)"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}

For **REMOVABLE_MEDIA_PATH**, you must use the same path that you specified
when you mirrored the images.

iv. Use **oc** command-line interface (CLI) to log in to the cluster that you are upgrading.

v. Apply the mirrored release image signature config map to the connected cluster:

   $ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>

   For **<image_signature_file>**, specify the path and name of the file, for example,
   signature-sha256-81154f5c03294534.yaml.

   - If the local container registry and the cluster are connected to the mirror host, directly push
   the release images to the local registry and apply the config map to the cluster by using
   following command:

   $ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} --apply-release-image-signature
9.1.5. Upgrading the restricted network cluster

Update the restricted network cluster to the OpenShift Container Platform version that you downloaded the release images for.

**NOTE**

If you have a local OpenShift Update Service, you can update by using the connected web console or CLI instructions instead of this procedure.

**Prerequisites**

- You mirrored the images for the new release to your registry.
- You applied the release image signature ConfigMap for the new release to your cluster.
- You obtained the sha256 sum value for the release from the image signature ConfigMap.
- Install the OpenShift CLI (`oc`), version 4.4.8 or later.

**Procedure**

- Update the cluster:

  ```
  $ oc adm upgrade --allow-explicit-upgrade --to-image
  ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}@sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92
  ```

  The `@sha256_sum_value` value is the sha256 sum value for the release from the image signature ConfigMap, for example, `@sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92`

  If you use an `ImageContentSourcePolicy` for the mirror registry, you can use the canonical registry name instead of `LOCAL_REGISTRY`.

**NOTE**

You can only configure global pull secrets for clusters that have an `ImageContentSourcePolicy` object. You cannot add a pull secret to a project.

9.1.6. Configuring image registry repository mirroring

Setting up container registry repository mirroring enables you to do the following:

- Configure your OpenShift Container Platform cluster to redirect requests to pull images from a repository on a source image registry and have it resolved by a repository on a mirrored image registry.
- Identify multiple mirrored repositories for each target repository, to make sure that if one mirror is down, another can be used.
The attributes of repository mirroring in OpenShift Container Platform include:

- Image pulls are resilient to registry downtimes.
- Clusters in restricted networks can pull images from critical locations, such as quay.io, and have registries behind a company firewall provide the requested images.
- A particular order of registries is tried when an image pull request is made, with the permanent registry typically being the last one tried.
- The mirror information you enter is added to the `/etc/containers/registries.conf` file on every node in the OpenShift Container Platform cluster.
- When a node makes a request for an image from the source repository, it tries each mirrored repository in turn until it finds the requested content. If all mirrors fail, the cluster tries the source repository. If successful, the image is pulled to the node.

Setting up repository mirroring can be done in the following ways:

- **At OpenShift Container Platform installation:** By pulling container images needed by OpenShift Container Platform and then bringing those images behind your company’s firewall, you can install OpenShift Container Platform into a datacenter that is in a restricted network.
- **After OpenShift Container Platform installation:** Even if you don’t configure mirroring during OpenShift Container Platform installation, you can do so later using the `ImageContentSourcePolicy` object.

The following procedure provides a post-installation mirror configuration, where you create an `ImageContentSourcePolicy` object that identifies:

- The source of the container image repository you want to mirror.
- A separate entry for each mirror repository you want to offer the content requested from the source repository.

**NOTE**

You can only configure global pull secrets for clusters that have an `ImageContentSourcePolicy` object. You cannot add a pull secret to a project.

**Prerequisites**

- Access to the cluster as a user with the `cluster-admin` role.

**Procedure**

1. Configure mirrored repositories, by either:

   - Setting up a mirrored repository with Red Hat Quay, as described in Red Hat Quay Repository Mirroring. Using Red Hat Quay allows you to copy images from one repository to another and also automatically sync those repositories repeatedly over time.

   - Using a tool such as `skopeo` to copy images manually from the source directory to the mirrored repository.
For example, after installing the skopeo RPM package on a Red Hat Enterprise Linux (RHEL) 7 or RHEL 8 system, use the `skopeo` command as shown in this example:

```
$ skopeo copy \\
docker://registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eab6 \\
docker://example.io/example/ubi-minimal
```

In this example, you have a container image registry that is named `example.io` with an image repository named `example` to which you want to copy the `ubi8/ubi-minimal` image from `registry.access.redhat.com`. After you create the registry, you can configure your OpenShift Container Platform cluster to redirect requests made of the source repository to the mirrored repository.

2. Log in to your OpenShift Container Platform cluster.

3. Create an `ImageContentSourcePolicy` file (for example, `registryrepomirror.yaml`), replacing the source and mirrors with your own registry and repository pairs and images:

```
apiVersion: operator.openshift.io/v1alpha1
definition: ImageContentSourcePolicy
metadata:
  name: ubi8repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - example.io/example/ubi-minimal
      source: registry.access.redhat.com/ubi8/ubi-minimal
  - mirrors:
    - example.com/example/ubi-minimal
      source: registry.access.redhat.com/ubi8/ubi-minimal
  - mirrors:
    - mirror.example.com/redhat
      source: registry.redhat.io/openshift4

1 Indicates the name of the image registry and repository.
2 Indicates the registry and repository containing the content that is mirrored.
3 You can configure a namespace inside a registry to use any image in that namespace. If you use a registry domain as a source, the `ImageContentSourcePolicy` resource is applied to all repositories from the registry.

4. Create the new `ImageContentSourcePolicy` object:

```
$ oc create -f registryrepomirror.yaml
```

After the `ImageContentSourcePolicy` object is created, the new settings are deployed to each node and the cluster starts using the mirrored repository for requests to the source repository.

5. To check that the mirrored configuration settings, are applied, do the following on one of the nodes.

a. List your nodes:
$ oc get node

### Example output

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip-10-0-137-44.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>7m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>ip-10-0-138-148.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>ip-10-0-139-122.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>ip-10-0-147-35.ec2.internal</td>
<td>Ready,SchedulingDisabled</td>
<td>worker</td>
<td>7m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>ip-10-0-153-12.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>7m</td>
<td>v1.21.0</td>
</tr>
<tr>
<td>ip-10-0-154-10.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.21.0</td>
</tr>
</tbody>
</table>

You can see that scheduling on each worker node is disabled as the change is being applied.

b. Start the debugging process to access the node:

$ oc debug node/ip-10-0-147-35.ec2.internal

### Example output

Starting pod/ip-10-0-147-35ec2internal-debug ...
To use host binaries, run `chroot /host`

c. Access the node’s files:

```sh
sh-4.2# chroot /host
```

d. Check the `/etc/containers/registries.conf` file to make sure the changes were made:

```sh
sh-4.2# cat /etc/containers/registries.conf
```

### Example output

```bash
unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
[[registry]]
    location = "registry.access.redhat.com/ubi8/"
    insecure = false
    blocked = false
    mirror-by-digest-only = true
    prefix = ""

[[registry.mirror]]
    location = "example.io/example/ubi8-minimal"
    insecure = false

[[registry.mirror]]
    location = "example.com/example/ubi8-minimal"
    insecure = false
```

e. Pull an image digest to the node from the source and check if it is resolved by the mirror. **ImageContentSourcePolicy** objects support image digests only, not image tags.
Troubleshooting repository mirroring

If the repository mirroring procedure does not work as described, use the following information about how repository mirroring works to help troubleshoot the problem.

- The first working mirror is used to supply the pulled image.
- The main registry is only used if no other mirror works.
- From the system context, the Insecure flags are used as fallback.
- The format of the /etc/containers/registries.conf file has changed recently. It is now version 2 and in TOML format.

9.1.7. Widening the scope of the mirror image catalog to reduce the frequency of cluster node reboots

You can scope the mirrored image catalog at the repository level or the wider registry level. A widely scoped ImageContentSourcePolicy resource reduces the number of times the nodes need to reboot in response to changes to the resource.

To widen the scope of the mirror image catalog in the ImageContentSourcePolicy resource, perform the following procedure.

Prerequisites

- Install the OpenShift Container Platform CLI oc.
- Log in as a user with cluster-admin privileges.
- Configure a mirrored image catalog for use in your disconnected cluster.

Procedure

1. Run the following command, specifying values for <local_registry>, <pull_spec>, and <pull_secret_file>:

   $ oc adm catalog mirror <local_registry>/<pull_spec> <local_registry> -a <pull_secret_file> --icsp-scope=registry

   where:

   <local_registry>
   is the local registry you have configured for your disconnected cluster, for example, local.registry:5000.

   <pull_spec>
   is the pull specification as configured in your disconnected registry, for example, redhat/redhat-operator-index:v4.8

   <pull_secret_file>
is the `registry.redhat.io` pull secret in `.json` file format. You can download the pull secret from the Red Hat OpenShift Cluster Manager.

The `oc adm catalog mirror` command creates a `/redhat-operator-index-manifests` directory and generates `imageContentSourcePolicy.yaml`, `catalogSource.yaml`, and `mapping.txt` files.

2. Apply the new `ImageContentSourcePolicy` resource to the cluster:

```bash
$ oc apply -f imageContentSourcePolicy.yaml
```

**Verification**

- Verify that `oc apply` successfully applied the change to `ImageContentSourcePolicy`:

```bash
$ oc get ImageContentSourcePolicy -o yaml
```

**Example output**

```yaml
apiVersion: v1
items:
- apiVersion: operator.openshift.io/v1alpha1
  kind: ImageContentSourcePolicy
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration:
        "apiVersion":"operator.openshift.io/v1alpha1","kind":"ImageContentSourcePolicy","metadata":
        {["annotations":{},"name":"redhat-operator-index"],"spec":{"repositoryDigestMirrors":
        {["mirrors":["local.registry:5000"],"source":"registry.redhat.io"]}}}
```

After you update the `ImageContentSourcePolicy` resource, OpenShift Container Platform deploys the new settings to each node and the cluster starts using the mirrored repository for requests to the source repository.

**9.2. ADDITIONAL RESOURCES**

- Using Operator Lifecycle Manager on restricted networks
- Machine Config Overview

**9.3. UPDATING A RESTRICTED NETWORK CLUSTER USING THE OPENSHIFT UPDATE SERVICE**

**9.3.1. About the OpenShift Update Service**

The OpenShift Update Service (OSUS) provides over-the-air updates to OpenShift Container Platform, including Red Hat Enterprise Linux CoreOS (RHCOS). It provides a graph, or diagram, that contains the vertices of component Operators and the edges that connect them. The edges in the graph show which versions you can safely update to. The vertices are update payloads that specify the intended state of the managed cluster components.

The Cluster Version Operator (CVO) in your cluster checks with the OpenShift Update Service to see
the valid updates and update paths based on current component versions and information in the graph. When you request an update, the CVO uses the release image for that update to update your cluster. The release artifacts are hosted in Quay as container images.

To allow the OpenShift Update Service to provide only compatible updates, a release verification pipeline drives automation. Each release artifact is verified for compatibility with supported cloud platforms and system architectures, as well as other component packages. After the pipeline confirms the suitability of a release, the OpenShift Update Service notifies you that it is available.

**IMPORTANT**

The OpenShift Update Service displays all recommended updates for your current cluster. If an upgrade path is not recommended by the OpenShift Update Service, it might be because of a known issue with the update or the target release.

Two controllers run during continuous update mode. The first controller continuously updates the payload manifests, applies the manifests to the cluster, and outputs the controlled rollout status of the Operators to indicate whether they are available, upgrading, or failed. The second controller polls the OpenShift Update Service to determine if updates are available.

**IMPORTANT**

Only upgrading to a newer version is supported. Reverting or rolling back your cluster to a previous version is not supported. If your update fails, contact Red Hat support.

During the update process, the Machine Config Operator (MCO) applies the new configuration to your cluster machines. The MCO cordons the number of nodes as specified by the `maxUnavailable` field on the machine configuration pool and marks them as unavailable. By default, this value is set to 1. The MCO then applies the new configuration and reboots the machine.

If you use Red Hat Enterprise Linux (RHEL) machines as workers, the MCO does not update the kubelet because you must update the OpenShift API on the machines first.

With the specification for the new version applied to the old kubelet, the RHEL machine cannot return to the *Ready* state. You cannot complete the update until the machines are available. However, the maximum number of unavailable nodes is set to ensure that normal cluster operations can continue with that number of machines out of service.

The OpenShift Update Service is composed of an Operator and one or more application instances.

**NOTE**

During the update process, nodes in the cluster might become temporarily unavailable. The *MachineHealthCheck* might identify such nodes as unhealthy and reboot them. To avoid rebooting such nodes, remove any *MachineHealthCheck* resource that you have deployed before updating the cluster. However, a MachineHealthCheck resource that is deployed by default (such as `machine-api-termination-handler`) cannot be removed and will be recreated.

Additional resources

- Understanding upgrade channels and releases
For clusters with internet accessibility, Red Hat provides over-the-air updates through an OpenShift Container Platform update service as a hosted service located behind public APIs. However, clusters in a restricted network have no way to access public APIs for update information.

To provide a similar update experience in a restricted network, you can install and configure the OpenShift Update Service locally so that it is available within a disconnected environment.

The following sections describe how to provide over-the-air updates for your disconnected cluster and its underlying operating system.

9.3.2. Prerequisites

- For more information on installing Operators, see Installing Operators in your namespace.

9.3.3. Configuring access to a secured registry for the OpenShift update service

If the release images are contained in a secure registry, complete the steps in Configuring additional trust stores for image registry access along with following changes for the update service.

The OpenShift Update Service Operator needs the config map key name updateservice-registry in the registry CA cert.

**Image registry CA config map example for the update service**

```yaml
apiVersion: v1
data:
  updateservice-registry: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com:5000: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----

<1> The OpenShift Update Service Operator requires the config map key name updateservice-registry in the registry CA cert.

<2> If the registry has the port, such as `registry-with-port.example.com:5000`, `:` should be replaced with `\`.`.

9.3.4. Updating the global cluster pull secret

You can update the global pull secret for your cluster by either replacing the current pull secret or appending a new pull secret.

The procedure is required when users use a separate registry to store images than the registry used during installation.
WARNING
Cluster resources must adjust to the new pull secret, which can temporarily limit the usability of the cluster.

Prerequisites
- You have access to the cluster as a user with the `cluster-admin` role.

Procedure
1. Optional: To append a new pull secret to the existing pull secret, complete the following steps:
   a. Enter the following command to download the pull secret:

   ```
   $ oc get secret/pull-secret -n openshift-config --template='{{index .data ".dockerconfigjson" | base64decode}}' > <pull_secret_location>
   
   Provide the path to the pull secret file.
   ```
   b. Enter the following command to add the new pull secret:

   ```
   $ oc registry login --registry="<registry>" \
   --auth-basic="<username>:<password>" \
   --to=<pull_secret_location>
   
   Provide the new registry. You can include multiple repositories within the same registry, for example: `--registry="<registry/my-namespace/my-repository>"`.
   
   Provide the credentials of the new registry.
   
   Provide the path to the pull secret file.
   ```

   Alternatively, you can perform a manual update to the pull secret file.

2. Enter the following command to update the global pull secret for your cluster:

   ```
   $ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson= <pull_secret_location>
   
   Provide the path to the new pull secret file.
   ```

   This update is rolled out to all nodes, which can take some time depending on the size of your cluster.
As of OpenShift Container Platform 4.7.4, changes to the global pull secret no longer trigger a node drain or reboot.

9.3.5. Installing the OpenShift Update Service Operator

To install the OpenShift Update Service, you must first install the OpenShift Update Service Operator by using the OpenShift Container Platform web console or CLI.

NOTE

For clusters that are installed on restricted networks, also known as disconnected clusters, Operator Lifecycle Manager by default cannot access the Red Hat-provided OperatorHub sources hosted on remote registries because those remote sources require full internet connectivity. For more information, see Using Operator Lifecycle Manager on restricted networks.

9.3.6. Installing the OpenShift Update Service Operator by using the web console

You can use the web console to install the OpenShift Update Service Operator.

Procedure

1. In the web console, click Operators → OperatorHub.

   NOTE

   Enter Update Service into the Filter by keyword... field to find the Operator faster.

2. Choose OpenShift Update Service from the list of available Operators, and click Install.
   
   a. Channel v1 is selected as the Update Channel since it is the only channel available in this release.
   
   b. Select A specific namespace on the cluster under Installation Mode.
   
   c. Select a namespace for Installed Namespace or accept the recommended namespace openshift-update-service.
   
   d. Select an Approval Strategy:
   
      ● The Automatic strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
   
      ● The Manual strategy requires a cluster administrator to approve the Operator update.
   
   e. Click Install.

3. Verify that the OpenShift Update Service Operator is installed by switching to the Operators → Installed Operators page.

4. Ensure that OpenShift Update Service is listed in the selected namespace with a Status of Succeeded.
9.3.7. Installing the OpenShift Update Service Operator by using the CLI

You can use the OpenShift CLI (oc) to install the OpenShift Update Service Operator.

**Procedure**

1. Create a namespace for the OpenShift Update Service Operator:
   a. Create a **Namespace** object YAML file, for example, *update-service-namespace.yaml*, for the OpenShift Update Service Operator:

   ```yaml
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-update-service
   annotations:
     openshift.io/node-selector: ""
   labels:
     openshift.io/cluster-monitoring: "true" ①
   ```

   ① Set the **openshift.io/cluster-monitoring** label to enable Operator-recommended cluster monitoring on this namespace.

   b. Create the namespace:

   ```bash
   $ oc create -f <filename>.yaml
   $ oc create -f update-service-namespace.yaml
   ```

2. Install the OpenShift Update Service Operator by creating the following objects:
   a. Create an **OperatorGroup** object YAML file, for example, *update-service-operator-group.yaml*:

   ```yaml
   apiVersion: operators.coreos.com/v1
   kind: OperatorGroup
   metadata:
     name: update-service-operator-group
   spec:
     targetNamespaces:
       - openshift-update-service
   ```

   b. Create an **OperatorGroup** object:

   ```bash
   $ oc -n openshift-update-service create -f <filename>.yaml
   $ oc -n openshift-update-service create -f update-service-operator-group.yaml
   ```

   c. Create a **Subscription** object YAML file, for example, *update-service-subscription.yaml*:
Example Subscription

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: update-service-subscription
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators"
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"
```

1. Specify the name of the catalog source that provides the Operator. For clusters that do not use a custom Operator Lifecycle Manager (OLM), specify `redhat-operators`. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the CatalogSource object created when you configured Operator Lifecycle Manager (OLM).

d. Create the Subscription object:

   ```bash
   $ oc create -f <filename>.yaml
   ```
   
   For example:

   ```bash
   $ oc -n openshift-update-service create -f update-service-subscription.yaml
   ```

   The OpenShift Update Service Operator is installed to the `openshift-update-service` namespace and targets the `openshift-update-service` namespace.

3. Verify the Operator installation:

   ```bash
   $ oc -n openshift-update-service get clusterserviceversions
   ```

   **Example output**

   ```
   NAME                             DISPLAY                    VERSION   REPLACES   PHASE
   update-service-operator.v4.6.0   OpenShift Update Service   4.6.0                Succeeded
   ...
   ```

   If the OpenShift Update Service Operator is listed, the installation was successful. The version number might be different than shown.

9.3.8. Creating the OpenShift Update Service graph data container image

The OpenShift Update Service requires a graph-data container image, from which the OpenShift Update Service retrieves information about channel membership and blocked update edges. Graph data is typically fetched directly from the upgrade graph data repository. In environments where an internet connection is unavailable, loading this information from an init container is another way to make the graph data available to the OpenShift Update Service. The role of the init container is to provide a local copy of the graph data, and during pod initialization, the init container copies the data to a volume that is accessible by the service.
Procedure

1. Create a Dockerfile, for example, `./Dockerfile`, containing the following:

   ```
   FROM registry.access.redhat.com/ubi8/ubi:8.1

   RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-data/archive/master.tar.gz

   CMD exec /bin/bash -c "tar xzvf cincinnati-graph-data.tar.gz -C /var/lib/cincinnati/graph-data/ --strip-components=1"
   ```

2. Use the docker file created in the above step to build a graph-data container image, for example, `registry.example.com/openshift/graph-data:latest`:

   ```
   $ podman build -f ./Dockerfile -t registry.example.com/openshift/graph-data:latest
   ```

3. Push the graph-data container image created in the previous step to a repository that is accessible to the OpenShift Update Service, for example, `registry.example.com/openshift/graph-data:latest`:

   ```
   $ podman push registry.example.com/openshift/graph-data:latest
   ```

   **NOTE**

   To push a graph data image to a local registry in a restricted network, copy the graph-data container image created in the previous step to a repository that is accessible to the OpenShift Update Service. Run `oc image mirror --help` for available options.

9.3.9. Mirroring the OpenShift Container Platform image repository

The OpenShift Update Service requires a locally accessible registry containing update release payloads.

**IMPORTANT**

To avoid excessive memory usage by the OpenShift Update Service application, it is recommended that you mirror release images to a separate repository, as described in the following procedure.

Prerequisites

- You reviewed and completed the steps from "Mirroring images for a disconnected installation" up to but not including the section entitled **Mirroring the OpenShift Container Platform image repository**.
- You configured a mirror registry to use in your restricted network and can access the certificate and credentials that you configured.
- You downloaded the **pull secret from the Red Hat OpenShift Cluster Manager** and modified it to include authentication to your mirror repository.
If you use self-signed certificates that do not set a Subject Alternative Name, you must precede the `oc` commands in this procedure with `GODEBUG=x509ignoreCN=0`. If you do not set this variable, the `oc` commands will fail with the following error:

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0
```

### Procedure

Complete the following steps on the mirror host:

1. Review the [OpenShift Container Platform downloads page](https://docs.openshift.com/containerplatform/4.6/install红旗包.html) to determine the version of OpenShift Container Platform to which you want to update and determine the corresponding tag on the [Repository Tags](https://mirror.openshift.redhat.com) page.

2. Set the required environment variables:
   a. Export the release version:

   ```
   $ OCP_RELEASE=<release_version>
   
   For `<release_version>`, specify the tag that corresponds to the version of OpenShift Container Platform to install, such as `4.6.4`.
   ```

   b. Export the local registry name and host port:

   ```
   $ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
   
   For `<local_registry_host_name>`, specify the registry domain name for your mirror repository, and for `<local_registry_host_port>`, specify the port that it serves content on.
   ```

   c. Export the local repository name:

   ```
   $ LOCAL_REPOSITORY='<local_repository_name>'
   
   For `<local_repository_name>`, specify the name of the repository to create in your registry, such as `ocp4/openshift4`.
   ```

   d. Export an additional local repository name to contain the release images:

   ```
   $ LOCAL_RELEASE_IMAGES_REPOSITORY='<local_release_images_repository_name>'
   
   For `<local_release_images_repository_name>`, specify the name of the repository to create in your registry, such as `ocp4/openshift4-release-images`.
   ```

   e. Export the name of the repository to mirror:

   ```
   $ PRODUCT_REPO='openshift-release-dev'
   
   For a production release, you must specify `openshift-release-dev`.
   ```

   f. Export the path to your registry pull secret:

   ```
   $ LOCAL_SECRET_JSON='<path_to_pull_secret>'
   ```
For `<path_to_pull_secret>`, specify the absolute path to and file name of the pull secret for your mirror registry that you created.

g. Export the release mirror:

```
$ RELEASE_NAME="ocp-release"
```

For a production release, you must specify `ocp-release`.

h. Export the type of architecture for your server, such as `x86_64`:

```
$ ARCHITECTURE=<server_architecture>
```

i. Export the path to the directory to host the mirrored images:

```
$ REMOVABLE_MEDIA_PATH=<path>  
```

Specify the full path, including the initial forward slash (`/`) character.

3. Mirror the version images to the mirror registry:

- If your mirror host does not have internet access, take the following actions:

  i. Connect the removable media to a system that is connected to the internet.

  ii. Review the images and configuration manifests to mirror:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  
--from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}  
--to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}  
--to-release-
image=${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}  
  --dry-run
```

  iii. Mirror the images to a directory on the removable media:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  
--to-dir=${REMOVABLE_MEDIA_PATH}/mirror  
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE}  
```

  iv. Take the media to the restricted network environment and upload the images to the local container registry:

```
$ oc image mirror -a ${LOCAL_SECRET_JSON}  
--from-dir=${REMOVABLE_MEDIA_PATH}/mirror  
"file://openshift/release:${OCP_RELEASE}"*  
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}  
```

For `REMOVABLE_MEDIA_PATH`, you must use the path where you mounted the removable media.
v. Use `oc` command-line interface (CLI) to log in to the cluster that you are upgrading.

vi. Apply the mirrored release image signature config map to the disconnected cluster:

```
$ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>
```

For `<image_signature_file>`, specify the path and name of the file, for example, `signature-sha256-81154f5c03294534.yaml`.

vii. Mirror the release image to a separate repository:

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} \
  ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-\n  ${ARCHITECTURE} \
  ${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_RELEASE}-\n  ${ARCHITECTURE}
```

- If the local container registry is connected to the mirror host, push the release images directly to the local registry:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} \n  --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-\n  ${ARCHITECTURE} \n  --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \n  --to-release-\n  image=${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_R\n  ELEASE}-${ARCHITECTURE}
```

9.3.10. Creating an OpenShift Update Service application

You can create an OpenShift Update Service application by using the OpenShift Container Platform web console or CLI.

9.3.10.1. Creating an OpenShift Update Service application by using the web console

You can use the OpenShift Container Platform web console to create an OpenShift Update Service application by using the OpenShift Update Service Operator.

**Prerequisites**

- The OpenShift Update Service Operator has been installed.
- The OpenShift Update Service graph-data container image has been created and pushed to a repository that is accessible to the OpenShift Update Service.
- The current release and update target releases have been mirrored to a locally accessible registry.

**Procedure**

1. In the web console, click **Operators → Installed Operators**.
2. Choose **OpenShift Update Service** from the list of installed Operators.

3. Click the **Update Service** tab.

4. Click **Create UpdateService**.

5. Enter a name in the **Name** field, for example, **service**.

6. Enter the local pullspec in the **Graph Data Image** field to the graph-data container image created in "Creating the OpenShift Update Service graph data container image", for example, registry.example.com/openshift/graph-data:latest.

7. In the **Releases** field, enter the local registry and repository created to contain the release images in "Mirroring the OpenShift Container Platform image repository", for example, registry.example.com/ocp4/openshift4-release-images.

8. Enter 2 in the **Replicas** field.

9. Click **Create** to create the OpenShift Update Service application.

10. Verify the OpenShift Update Service application:

    - From the **UpdateServices** list in the **Update Service** tab, click the Update Service application just created.

    - Click the **Resources** tab.

    - Verify each application resource has a status of **Created**.

### 9.3.10.2. Creating an OpenShift Update Service application by using the CLI

You can use the OpenShift CLI (**oc**) to create an OpenShift Update Service application.

**Prerequisites**

- The OpenShift Update Service Operator has been installed.

- The OpenShift Update Service graph-data container image has been created and pushed to a repository that is accessible to the OpenShift Update Service.

- The current release and update target releases have been mirrored to a locally accessible registry.

**Procedure**

1. Configure the OpenShift Update Service target namespace, for example, **openshift-update-service**:

   ```bash
   $ NAME=$NAMESPACE=openshift-update-service
   
   The namespace must match the **targetNamespaces** value from the operator group.
   ```

2. Configure the name of the OpenShift Update Service application, for example, **service**:

   ```bash
   $ NAME=service
   ```
3. Configure the local registry and repository for the release images as configured in "Mirroring the OpenShift Container Platform image repository", for example, registry.example.com/ocp4/openshift4-release-images:

   $ RELEASE_IMAGES=registry.example.com/ocp4/openshift4-release-images

4. Set the local pullspec for the graph-data image to the graph-data container image created in "Creating the OpenShift Update Service graph data container image", for example, registry.example.com/openshift/graph-data:latest:

   $ GRAPH_DATA_IMAGE=registry.example.com/openshift/graph-data:latest

5. Create an OpenShift Update Service application object:

   $ oc -n "$NAMESPACE" create -f - <<EOF
apiVersion: updateservice.operator.openshift.io/v1
kind: UpdateService
metadata:
  name: "$NAME"
spec:
  replicas: 2
  releases: "$RELEASE_IMAGES"
  graphDataImage: "$GRAPH_DATA_IMAGE"
EOF

6. Verify the OpenShift Update Service application:

   a. Use the following command to obtain a policy engine route:

      $ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "$NAMESPACE" get -o jsonpath="{.status.policyEngineURI}/api/upgrades_info/v1/graph\"name\"\"updateservice "$NAME\")")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}" = http -o "${SCHEME}" = https; then break; fi; done

      You might need to poll until the command succeeds.

   b. Retrieve a graph from the policy engine. Be sure to specify a valid version for channel. For example, if running in OpenShift Container Platform 4.8, use stable-4.8:

      $ while sleep 10; do HTTP_CODE="$(curl --header Accept:application/json --output /dev/stderr --write-out "\$http_code\" $(${POLICY_ENGINE_GRAPH_URI}?channel=stable-4.6")" ; if test "$HTTP_CODE" -eq 200; then break; fi; echo "$HTTP_CODE"; done

      This polls until the graph request succeeds; however, the resulting graph might be empty depending on which release images you have mirrored.

**NOTE**

The policy engine route name must not be more than 63 characters based on RFC-1123. If you see ReconcileCompleted status as false with the reason CreateRouteFailed caused by host must conform to DNS 1123 naming convention and must be no more than 63 characters, try creating the Update Service with a shorter name.
9.3.10.3. Configuring the Cluster Version Operator (CVO)

After the OpenShift Update Service Operator has been installed and the OpenShift Update Service application has been created, the Cluster Version Operator (CVO) can be updated to pull graph data from the locally installed OpenShift Update Service.

**Prerequisites**

- The OpenShift Update Service Operator has been installed.
- The OpenShift Update Service graph-data container image has been created and pushed to a repository that is accessible to the OpenShift Update Service.
- The current release and update target releases have been mirrored to a locally accessible registry.
- The OpenShift Update Service application has been created.

**Procedure**

1. Set the OpenShift Update Service target namespace, for example, `openshift-update-service`:

   ```
   $ NAMESPACE=openshift-update-service
   ```

2. Set the name of the OpenShift Update Service application, for example, `service`:

   ```
   $ NAME=service
   ```

3. Obtain the policy engine route:

   ```
   $ POLICY_ENGINE_GRAPH_URI="$(oc -n "$NAMESPACE" get -o jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{""}’ updateservice "$NAME")"
   ```

4. Set the patch for the pull graph data:

   ```
   $ PATCH="{"spec":{"upstream":"$POLICY_ENGINE_GRAPH_URI"}}"
   ```

5. Patch the CVO to use the local OpenShift Update Service:

   ```
   $ oc patch clusterversion version -p $PATCH --type merge
   ```

   **NOTE**

   See Enabling the cluster-wide proxy to configure the CA to trust the update server.

9.3.11. Deleting an OpenShift Update Service application

You can delete an OpenShift Update Service application by using the OpenShift Container Platform web console or CLI.

9.3.11.1. Deleting an OpenShift Update Service application by using the web console
You can use the OpenShift Container Platform web console to delete an OpenShift Update Service application by using the OpenShift Update Service Operator.

**Prerequisites**

- The OpenShift Update Service Operator has been installed.

**Procedure**

1. In the web console, click **Operators → Installed Operators**.
2. Choose **OpenShift Update Service** from the list of installed Operators.
3. Click the **Update Service** tab.
4. From the list of installed OpenShift Update Service applications, select the application to be deleted and then click **Delete UpdateService**.
5. From the **Delete UpdateService?** confirmation dialog, click **Delete** to confirm the deletion.

### 9.3.11.2. Deleting an OpenShift Update Service application by using the CLI

You can use the OpenShift CLI (**oc**) to delete an OpenShift Update Service application.

**Procedure**

1. Get the OpenShift Update Service application name using the namespace the OpenShift Update Service application was created in, for example, **openshift-update-service**:

   ```sh
   $ oc get updateservice -n openshift-update-service
   
   Example output
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>6s</td>
</tr>
</tbody>
</table>
   
   2. Delete the OpenShift Update Service application using the **NAME** value from the previous step and the namespace the OpenShift Update Service application was created in, for example, **openshift-update-service**:

   ```sh
   $ oc delete updateservice service -n openshift-update-service
   
   Example output
   
   updateservice.updateservice.operator.openshift.io "service" deleted
   ```

### 9.3.12. Uninstalling the OpenShift Update Service Operator

To uninstall the OpenShift Update Service, you must first delete all OpenShift Update Service applications by using the OpenShift Container Platform web console or CLI.

#### 9.3.12.1. Uninstalling the OpenShift Update Service Operator by using the web console
You can use the OpenShift Container Platform web console to uninstall the OpenShift Update Service Operator.

**Prerequisites**

- All OpenShift Update Service applications have been deleted.

**Procedure**

1. In the web console, click **Operators → Installed Operators**.
2. Select **OpenShift Update Service** from the list of installed Operators and click **Uninstall Operator**.
3. From the **Uninstall Operator?** confirmation dialog, click **Uninstall** to confirm the uninstallation.

**9.3.12.2. Uninstalling the OpenShift Update Service Operator by using the CLI**

You can use the OpenShift CLI (oc) to uninstall the OpenShift Update Service Operator.

**Prerequisites**

- All OpenShift Update Service applications have been deleted.

**Procedure**

1. Change to the project containing the OpenShift Update Service Operator, for example, `openshift-update-service`:

   ```bash
   $ oc project openshift-update-service
   Now using project "openshift-update-service" on server "https://example.com:6443".
   ```

2. Get the name of the OpenShift Update Service Operator operator group:

   ```bash
   $ oc get operatorgroup
   NAME   AGE
   openshift-update-service-fprx2  4m41s
   ```

3. Delete the operator group, for example, `openshift-update-service-fprx2`:

   ```bash
   $ oc delete operatorgroup openshift-update-service-fprx2
   operatorgroup.operators.coreos.com "openshift-update-service-fprx2" deleted
   ```
4. Get the name of the OpenShift Update Service Operator subscription:

```
$ oc get subscription
```

**Example output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>PACKAGE</th>
<th>SOURCE</th>
<th>CHANNEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>update-service-operator</td>
<td>update-service-operator</td>
<td>updateservice-index-catalog</td>
<td>v1</td>
</tr>
</tbody>
</table>

5. Using the **Name** value from the previous step, check the current version of the subscribed OpenShift Update Service Operator in the **currentCSV** field:

```
$ oc get subscription update-service-operator -o yaml | grep " currentCSV"
```

**Example output**

```
currentCSV: update-service-operator.v0.0.1
```

6. Delete the subscription, for example, **update-service-operator**:

```
$ oc delete subscription update-service-operator
```

**Example output**

```
subscription.operators.coreos.com "update-service-operator" deleted
```

7. Delete the CSV for the OpenShift Update Service Operator using the **currentCSV** value from the previous step:

```
$ oc delete clusterserviceversion update-service-operator.v0.0.1
```

**Example output**

```
clusterserviceversion.operators.coreos.com "update-service-operator.v0.0.1" deleted
```