OpenShift Container Platform 4.11

Updating clusters

Last Updated: 2022-12-15
OpenShift Container Platform 4.11 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.
# Table of Contents

## CHAPTER 1. UPDATING CLUSTERS OVERVIEW ............................................. 5
  1.1. UNDERSTANDING OPENSHIFT CONTAINER PLATFORM UPDATES 5  
  1.2. UNDERSTANDING UPGRADE CHANNELS AND RELEASES 5  
  1.3. PREPARING TO PERFORM AN EUS-TO-EUS UPDATE 5  
  1.4. UPDATING A CLUSTER USING THE WEB CONSOLE 5  
  1.5. UPDATING A CLUSTER USING THE CLI 6  
  1.6. PERFORMING A CANARY ROLLOUT UPDATE 6  
  1.7. UPDATING A CLUSTER THAT INCLUDES RHEL COMPUTE MACHINES 6  
  1.8. UPDATING A RESTRICTED NETWORK CLUSTER 6  
  1.9. UPDATING HARDWARE ON NODES RUNNING IN VSphere 7  
  1.10. UPDATING A CLUSTER THAT INCLUDES THE SPECIAL RESOURCE OPERATOR 7

## CHAPTER 2. UNDERSTANDING OPENSHIFT CONTAINER PLATFORM UPDATES ............. 8
  2.1. COMMON TERMS 8

## CHAPTER 3. UNDERSTANDING UPDATE CHANNELS AND RELEASES .......................... 10
  3.1. UPDATE CHANNELS 10  
     3.1.1. fast-4.11 channel 10  
     3.1.2. stable-4.11 channel 10  
     3.1.3. eus-4.y channel 11  
     3.1.4. candidate-4.11 channel 11  
     3.1.5. Update recommendations in the channel 11  
     3.1.6. Update recommendation removals and Conditional Updates 12  
     3.1.7. Choosing the correct channel for your cluster 12  
     3.1.8. Restricted network clusters 13  
     3.1.9. Switching between channels 13

## CHAPTER 4. UNDERSTANDING OPENSHIFT CONTAINER PLATFORM UPDATE DURATION .......... 14
  4.1. PREREQUISITES 14  
  4.2. FACTORS AFFECTING UPDATE DURATION 14  
  4.3. CLUSTER UPDATE PHASES 14  
     4.3.1. Cluster Version Operator target update payload deployment 14  
     4.3.2. Machine Config Operator node updates 14  
  4.4. ESTIMATING CLUSTER UPDATE TIME 16  
  4.5. RED HAT ENTERPRISE LINUX (RHEL) COMPUTE NODES 17

## CHAPTER 5. PREPARING TO PERFORM AN EUS-TO-EUS UPDATE ............................. 18
  5.1. EUS-TO-EUS UPDATE 18

## CHAPTER 6. UPDATING A CLUSTER USING THE WEB CONSOLE ............................ 21
  6.1. PREREQUISITES 21  
  6.2. PERFORMING A CANARY ROLLOUT UPDATE 22  
  6.3. UPGRADING CLUSTERS WITH MANUALLY MAINTAINED CREDENTIALS 23  
  6.4. PAUSING A MACHINEHEALTHCHECK RESOURCE BY USING THE WEB CONSOLE 24  
  6.5. ABOUT UPDATING SINGLE NODE OPENSHIFT CONTAINER PLATFORM 25  
  6.6. UPGRADING A CLUSTER BY USING THE WEB CONSOLE 26  
  6.7. CHANGING THE UPDATE SERVER BY USING THE WEB CONSOLE 27

## CHAPTER 7. UPDATING A CLUSTER USING THE CLI ........................................ 28
  7.1. PREREQUISITES 28  
  7.2. UPGRADING CLUSTERS WITH MANUALLY MAINTAINED CREDENTIALS 29  
  7.3. PAUSING A MACHINEHEALTHCHECK RESOURCE 30  
  7.4. ABOUT UPDATING SINGLE NODE OPENSHIFT CONTAINER PLATFORM 31
### Chapter 7. Update a cluster

7.5. Updating a Cluster by Using the CLI
7.6. Updating Along a Conditional Upgrade Path
7.7. Changing the Update Server by Using the CLI

### Chapter 8. Performing a Canary Rollout Update

8.1. About the Canary Rollout Update Process and MCPS
8.2. About Performing a Canary Rollout Update
8.3. Creating Machine Config Pools to Perform a Canary Rollout Update
8.4. Pausing the Machine Config Pools
8.5. Performing the Cluster Update
8.6. Unpausing the Machine Config Pools
8.6.1. In case of application failure
8.7. Moving a Node to the Original Machine Config Pool

### Chapter 9. Updating a Cluster That Includes RHEL Compute Machines

9.1. Prerequisites
9.2. Updating a Cluster by Using the Web Console
9.3. Optional: Adding Hooks to Perform Ansible Tasks on RHEL Machines
   9.3.1. About Ansible hooks for upgrades
   9.3.2. Configuring the Ansible inventory file to use hooks
   9.3.3. Available hooks for RHEL compute machines
9.4. Updating RHEL Compute Machines in Your Cluster

### Chapter 10. Updating a Restricted Network Cluster

10.1. Updating a Restricted Network Cluster Using the CLI
    10.1.1. Prerequisites
    10.1.2. Preparing your mirror host
           10.1.2.1. Installing the OpenShift CLI by downloading the binary
           10.1.2.2. Configuring credentials that allow images to be mirrored
           10.1.3. Mirroring the OpenShift Container Platform image repository
           10.1.3.1. Mirroring resources using the oc-mirror plug-in
           10.1.3.2. Mirroring images using the oc adm release mirror command
           10.1.4. Pausing a MachineHealthCheck resource
           10.1.5. Upgrading the restricted network cluster
           10.1.6. Configuring image registry repository mirroring
           10.1.7. Widening the scope of the mirror image catalog to reduce the frequency of cluster node reboots
10.2. Additional Resources
10.3. Updating a Restricted Network Cluster Using the OpenShift Update Service
    10.3.1. About the OpenShift Update Service
    10.3.2. Prerequisites
    10.3.3. Configuring access to a secured registry for the OpenShift update service
    10.3.4. Updating the global cluster pull secret
    10.3.5. Installing the OpenShift Update Service Operator
    10.3.6. Installing the OpenShift Update Service Operator by using the web console
    10.3.7. Installing the OpenShift Update Service Operator by using the CLI
    10.3.8. Creating the OpenShift Update Service graph data container image
    10.3.9. Mirroring the OpenShift Container Platform image repository
           10.3.9.1. Mirroring resources using the oc-mirror plug-in
           10.3.9.2. Mirroring images using the oc adm release mirror command
    10.3.10. Creating an OpenShift Update Service application
           10.3.10.1. Creating an OpenShift Update Service application by using the web console
CHAPTER 11. UPDATING HARDWARE ON NODES RUNNING ON VSPHERE .................. 85
11.1. UPDATING VIRTUAL HARDWARE ON VSPHERE .......................... 85
   11.1.1. Updating the virtual hardware for control plane nodes on vSphere 85
   11.1.2. Updating the virtual hardware for compute nodes on vSphere 86
   11.1.3. Updating the virtual hardware for template on vSphere 87
11.2. SCHEDULING AN UPDATE FOR VIRTUAL HARDWARE ON VSPHERE .... 88

CHAPTER 12. UPDATING A CLUSTER THAT INCLUDES THE SPECIAL RESOURCE OPERATOR ...... 89
12.1. CUSTOM RESOURCE AND VERIFICATION STATUS INFORMATION .......... 89
12.2. RUNNING A PREFLIGHT CHECK FOR THE SPECIAL RESOURCE OPERATOR 89
12.3. ADDITIONAL RESOURCES ............................................... 91

10.3.10.2. Creating an OpenShift Update Service application by using the CLI 79
10.3.10.3. Configuring the Cluster Version Operator (CVO) 81
10.3.11. Deleting an OpenShift Update Service application 82
10.3.11.1. Deleting an OpenShift Update Service application by using the web console 82
10.3.11.2. Deleting an OpenShift Update Service application by using the CLI 82
10.3.12. Uninstalling the OpenShift Update Service Operator 83
10.3.12.1. Uninstalling the OpenShift Update Service Operator by using the web console 83
10.3.12.2. Uninstalling the OpenShift Update Service Operator by using the CLI 83
CHAPTER 1. UPDATING CLUSTERS OVERVIEW

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

1.1. UNDERSTANDING OPENSHIFT CONTAINER PLATFORM UPDATES

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

1.2. UNDERSTANDING UPGRADE CHANNELS AND RELEASES

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

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

1.3. PREPARING TO PERFORM AN EUS-TO-EUS UPDATE

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 4.9 to 4.10, and then to 4.11. 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-control plane hosts. For more information, see the following:

- Updating EUS-to-EUS

1.4. UPDATING A CLUSTER USING THE WEB CONSOLE

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

- Performing a canary rollout update
- Pausing a MachineHealthCheck resource
- About updating OpenShift Container Platform on a single-node cluster
- Updating a cluster by using the web console
1.5. UPDATING A CLUSTER USING THE CLI

Updating a cluster using the CLI: You can update an OpenShift Container Platform cluster within a minor version by using the OpenShift CLI (oc). The following steps update a cluster within a minor version. You can use the same instructions for updating a cluster between minor versions.

- Pausing a MachineHealthCheck resource
- About updating OpenShift Container Platform on a single-node cluster
- Updating a cluster by using the CLI
- Changing the update server by using the CLI

1.6. PERFORMING A CANARY ROLLOUT UPDATE

Performing a canary rollout update: By controlling the 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 procedures:

- 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

1.7. UPDATING A CLUSTER THAT INCLUDES RHEL COMPUTE MACHINES

Updating a cluster that includes RHEL compute machines: If your cluster contains Red Hat Enterprise Linux (RHEL) machines, you must perform additional steps to update those machines. You can perform the following procedures:

- Updating a cluster by using the web console
- Optional: Adding hooks to perform Ansible tasks on RHEL machines
- Updating RHEL compute machines in your cluster

1.8. UPDATING A RESTRICTED NETWORK CLUSTER

Updating a restricted network cluster: If your mirror host cannot access both the internet and the cluster, you can mirror the images to a file system that is disconnected from that environment. You can
then bring that host or removable media across that gap. If the local container registry and the cluster are connected to the mirror host of a registry, you can directly push the release images to the local registry.

- 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

1.9. UPDATING HARDWARE ON NODES RUNNING IN VSphere

Updating hardware on vSphere: You must ensure that your nodes running in vSphere are running on the hardware version supported by OpenShift Container Platform. Currently, hardware version 13 or later is supported for vSphere virtual machines in a cluster. For more information, see the following:

- Updating virtual hardware on vSphere
- Scheduling an update for virtual hardware on vSphere

1.10. UPDATING A CLUSTER THAT INCLUDES THE SPECIAL RESOURCE OPERATOR

Updating a cluster that includes the Special Resource Operator: When updating a cluster that includes the Special Resource Operator (SRO), it is important to consider whether the new kernel module version is compatible with the kernel modules currently loaded by the SRO. You can run a preflight check to confirm if the SRO will be able to upgrade the kernel modules.

IMPORTANT

Using hardware version 13 for your cluster nodes running on vSphere is now deprecated. This version is still fully supported, but support will be removed in a future version of OpenShift Container Platform. Hardware version 15 is now the default for vSphere virtual machines in OpenShift Container Platform.
CHAPTER 2. 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.

NOTE
Operators previously installed through Operator Lifecycle Manager (OLM) follow a different process for updates. See Updating installed Operators for more information.

2.1. 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
Update channels are tied to a minor version of OpenShift Container Platform. For instance, OpenShift Container Platform 4.10 update channels recommend updates to 4.10 and updates within 4.10. They also recommend updates within 4.9 and from 4.9 to 4.10 allowing all 4.9 to eventually update to 4.10, even if they do not immediately meet the minimum z-stream version requirements. They do not recommend updates to 4.11 or later releases. This strategy ensures that administrators explicitly decide to update to the next minor version of OpenShift Container Platform.

Update 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 4.11 offers the following update channels:

- **stable-4.11**
- **eus-4.y** (only offered for EUS versions and meant to facilitate upgrades between EUS versions)
- **fast-4.11**
- **candidate-4.11**

If you do not want the Cluster Version Operator to fetch available updates from the update recommendation service, you can use the `oc adm upgrade channel` command in the OpenShift CLI to configure an empty channel. This configuration can be helpful if, for example, a cluster has restricted network access and there is no local, reachable update recommendation service.

**WARNING**

Red Hat recommends upgrading to versions suggested by OpenShift Update Service only. For a 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. UPDATE CHANNELS

#### 3.1.1. fast-4.11 channel

The **fast-4.11** channel is updated with new versions of OpenShift Container Platform 4.11 as soon as Red Hat declares the version as a general availability (GA) release. As such, these releases are fully supported and purposed to be used in production environments.

#### 3.1.2. stable-4.11 channel

While the **fast-4.11** channel contains releases as soon as their errata are published, releases are added to the **stable-4.11** channel after a delay. During this delay, data is collected from multiple sources and analyzed for indications of product regressions. Once a significant number of data points have been
collected, and absent negative signals, these releases are added to the stable channel.

NOTE

Since the time required to obtain a significant number of data points varies based on many factors, Service Level Objective (SLO) is not offered for the delay duration between fast and stable channels. For more information, please see "Choosing the correct channel for your cluster"

Newly installed clusters default to using stable channels.

3.1.3. eus-4.y channel

In addition to the stable channel, all even-numbered minor versions of OpenShift Container Platform offer Extended Update Support (EUS). Releases promoted to the stable channel are also simultaneously promoted to the EUS channels. The primary purpose of the EUS channels is to serve as a convenience for clusters performing an EUS-to-EUS update.

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.

3.1.4. candidate-4.11 channel

The candidate-4.11 channel offers unsupported early access to releases as soon as they are built. Releases present only in candidate channels may not contain the full feature set of eventual GA releases or features may be removed prior to GA. Additionally, these releases have not been subject to full Red Hat Quality Assurance and may not offer update paths to later GA releases. Given these caveats, the candidate channel is only suitable for testing purposes where destroying and recreating a cluster is acceptable.

3.1.5. Update recommendations in the channel

OpenShift Container Platform maintains an update recommendation service that knows your installed OpenShift Container Platform version and the path to take within the channel to get you to the next release. Update paths are also limited to versions relevant to your currently selected channel and its promotion characteristics.

You can imagine seeing the following releases in your channel:

- 4.11.0
- 4.11.1
- 4.11.3
- 4.11.4

The service recommends only updates that have been tested and have no known serious regressions. For example, if your cluster is on 4.11.1 and OpenShift Container Platform suggests 4.11.4, then it is recommended to update from 4.11.1 to 4.11.4.
IMPORTANT

Do not rely on consecutive patch numbers. In this example, 4.11.2 is not and never was available in the channel, therefore updates to 4.11.2 are not recommended or supported.

3.1.6. Update recommendation removals and Conditional Updates

Red Hat monitors newly released versions and update paths associated with those versions before and after they are added to supported channels. If a serious regression is identified, Red Hat may remove affected update recommendations. When Red Hat chooses to remove update recommendations, that action is taken in all relevant channels simultaneously. The removal of recommended updates may happen either before or after updates have been promoted to supported channels.

If Red Hat removes update recommendations from any supported release, a superseding update recommendation will be provided to a future version that corrects the regression. There may however be a delay while the defect is corrected, tested, and promoted to your selected channel.

Beginning in OpenShift Container Platform 4.10, when update recommendations are removed from supported channels, they are replaced with Conditional Updates that declare one or more known risks. Each known risk may apply to all clusters or only clusters matching certain conditions. Some examples include having the Platform set to None or the CNI provider set to OpenShiftSDN. The Cluster Version Operator (CVO) continually evaluates known risks against the current cluster state. If no risks match, the update is recommended. If the risk matches, those updates are listed as a Supported But Not Recommended update and a reference link is provided. The reference link helps the cluster admin decide if they would like to accept the risk and update anyway.

3.1.7. Choosing the correct channel for your cluster

Choosing the appropriate channel involves two decisions.

First, select the minor version you want for your cluster upgrade. Selecting a channel which matches your current version ensures that you only apply z-stream updates and do not receive feature updates. Selecting an available channel which has a version greater than your current version will ensure that after one or more updates your cluster will have updated to that version. Your cluster will only be offered channels which match its current version, the next version, or the next EUS version.

NOTE

Due to the complexity involved in planning upgrades between versions many minors apart, channels that assist in planning upgrades beyond a single EUS-to-EUS update are not offered.

Second, you should choose your desired rollout strategy. You may choose to update as soon as Red Hat declares a release GA by selecting from fast channels or you may want to wait for Red Hat to promote releases to the stable channel. Update recommendations offered in the fast-4.11 and stable-4.11 are both fully supported and benefit equally from ongoing data analysis. The promotion delay before promoting release to the stable channel represents the only difference between the two channels. Updates to the latest z-streams are generally promoted to the stable channel within a week or two, however the delay when initially rolling out updates to the latest minor is much longer, generally 45-90 days. Please consider the promotion delay when choosing your desired channel as waiting for promotion to the stable channel may affect your scheduling plans.

Additionally, there are several factors which may lead an organization to move clusters to the fast channel either permanently or temporarily including:
The desire to apply a specific fix known to affect your environment without delay.

Application of CVE fixes without delay. CVE fixes may introduce regressions, so promotion delays still apply to z-streams with CVE fixes.

Internal testing processes. If it takes your organization several weeks to qualify releases it is best test concurrently with our promotion process rather than waiting. This also assures that any telemetry signal provided to Red Hat is a factored into our rollout, so issues relevant to you can be fixed faster.

3.1.8. 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 an 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.

3.1.9. Switching between channels

A channel can be switched from the web console or through the `adm upgrade channel` command:

```
$ oc adm upgrade channel <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-4.11` channel to the `fast-4.11` channel.
- You can switch to the `candidate-4.11` channel at any time, but some releases for this channel might be unsupported.
- You can switch from the `candidate-4.11` channel to the `fast-4.11` channel if your current release is a general availability release.
- You can always switch from the `fast-4.11` channel to the `stable-4.11` channel. There is a possible delay of up to a day for the release to be promoted to `stable-4.11` if the current release was recently promoted.

Additional resources

- Updating along a conditional upgrade path
- Choosing the correct channel for your cluster
CHAPTER 4. UNDERSTANDING OPENSIFT CONTAINER PLATFORM UPDATE DURATION

OpenShift Container Platform update duration varies based on the deployment topology. This page helps you understand the factors that affect update duration and estimate how long the cluster update takes in your environment.

4.1. PREREQUISITES

- You are familiar with OpenShift Container Platform architecture and OpenShift Container Platform updates.

4.2. FACTORS AFFECTING UPDATE DURATION

The following factors can affect your cluster update duration:

- The reboot of compute nodes to the new machine configuration by Machine Config Operator (MCO)
  - The value of MaxUnavailable in the machine config pool
  - The minimum number or percentages of replicas set in pod disruption budget (PDB)
- The number of nodes in the cluster
- The health of the cluster nodes

4.3. CLUSTER UPDATE PHASES

In OpenShift Container Platform, the cluster update happens in two phases:

- Cluster Version Operator (CVO) target update payload deployment
- Machine Config Operator (MCO) node updates

4.3.1. Cluster Version Operator target update payload deployment

The Cluster Version Operator (CVO) retrieves the target update release image and applies to the cluster. All components which run as pods are updated during this phase, whereas the host components are updated by the Machine Config Operator (MCO). This process might take 60 to 120 minutes.

**NOTE**

The CVO phase of the update does not restart the nodes.

Additional resources

- Cluster Version Operator (CVO) overview

4.3.2. Machine Config Operator node updates
The Machine Config Operator (MCO) applies a new machine configuration to each control plane and compute node. During this process, the MCO performs the following sequential actions on each node of the cluster:

1. Cordon and drain all the nodes
2. Update the operating system (OS)
3. Reboot the nodes
4. Uncordon all nodes and schedule workloads on the node

**NOTE**

When a node is cordoned, workloads cannot be scheduled to it.

The time to complete this process depends on several factors including the node and infrastructure configuration. This process might take 5 or more minutes to complete per node.

In addition to MCO, you should consider the impact of the following parameters:

- The control plane node update duration is predictable and oftentimes shorter than compute nodes, because the control plane workloads are tuned for graceful updates and quick drains.

- You can update the compute nodes in parallel by setting the `maxUnavailable` field to greater than 1 in the Machine Config Pool (MCP). The MCO cordons the number of nodes specified in `maxUnavailable` and marks them unavailable for update.

- When you increase `maxUnavailable` on the MCP, it can help the pool to update more quickly. However, if `maxUnavailable` is set too high, and several nodes are cordoned simultaneously, the pod disruption budget (PDB) guarded workloads could fail to drain because a schedulable node cannot be found to run the replicas. If you increase `maxUnavailable` for the MCP, ensure that you still have sufficient schedulable nodes to allow PDB guarded workloads to drain.

To check the status of nodes from the terminal, run the following command:

```
$ oc get node
```

**Example Output**

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES, AGE, VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ip-10-0-137-31.us-east-2.compute.internal</td>
<td>Ready,SchedulingDisabled</td>
<td>worker 12d</td>
</tr>
<tr>
<td></td>
<td>v1.23.5+3afdacb</td>
<td></td>
</tr>
<tr>
<td>ip-10-0-151-208.us-east-2.compute.internal</td>
<td>Ready</td>
<td>master 12d</td>
</tr>
<tr>
<td></td>
<td>v1.23.5+3afdacb</td>
<td></td>
</tr>
<tr>
<td>ip-10-0-176-138.us-east-2.compute.internal</td>
<td>Ready</td>
<td>master 12d</td>
</tr>
<tr>
<td></td>
<td>v1.23.5+3afdacb</td>
<td></td>
</tr>
<tr>
<td>ip-10-0-183-194.us-east-2.compute.internal</td>
<td>Ready</td>
<td>worker 12d</td>
</tr>
<tr>
<td></td>
<td>v1.23.5+3afdacb</td>
<td></td>
</tr>
<tr>
<td>ip-10-0-204-102.us-east-2.compute.internal</td>
<td>Ready</td>
<td>master 12d</td>
</tr>
</tbody>
</table>
If the status of the node is **NotReady** or **SchedulingDisabled**, then the node is not available and this impacts the update duration.

You can check the status of nodes from the **Administrator** perspective in the web console by expanding **Compute → Node**.

### 4.4. ESTIMATING CLUSTER UPDATE TIME

Historical update duration of similar clusters provides you the best estimate for the future cluster updates. However, if the historical data is not available, you can use the following convention to estimate your cluster update time:

\[
\text{Cluster update time} = \text{CVO target update payload deployment time} + (\# \text{ node update iterations} \times \text{MCO node update time})
\]

A node update iteration consists of one or more nodes updated in parallel. The control plane nodes are always updated in parallel with the compute nodes. In addition, one or more compute nodes can be updated in parallel based on the **maxUnavailable** value.

For example, to estimate the update time, consider an OpenShift Container Platform cluster with three control plane nodes and six compute nodes and each host takes about 5 minutes to reboot.

**NOTE**

The time it takes to reboot a particular node varies significantly. In cloud instances, the reboot might take about 1 to 2 minutes, whereas in physical bare metal hosts the reboot might take more than 15 minutes.

**Scenario-1**

When you set **maxUnavailable** to 1 for both the control plane and compute nodes Machine Config Pool (MCP), then all the six compute nodes will update one after another in each iteration:

\[
\text{Cluster update time} = 60 + (6 \times 5) = 90 \text{ minutes}
\]

**Scenario-2**

When you set **maxUnavailable** to 2 for the compute node MCP, then two compute nodes will update in parallel in each iteration. Therefore it takes total three iterations to update all the nodes.

\[
\text{Cluster update time} = 60 + (3 \times 5) = 75 \text{ minutes}
\]
The default setting for `maxUnavailable` is 1 for all the MCPs in OpenShift Container Platform. It is recommended that you do not change the `maxUnavailable` in the control plane MCP.

4.5. RED HAT ENTERPRISE LINUX (RHEL) COMPUTE NODES

Red Hat Enterprise Linux (RHEL) compute nodes require an additional usage of `openshift-ansible` to update node binary components. The actual time spent updating RHEL compute nodes should not be significantly different from Red Hat Enterprise Linux CoreOS (RHCOS) compute nodes.

Additional resources

- Updating RHEL compute machines
Due to fundamental Kubernetes design, all OpenShift Container Platform updates between minor versions must be serialized. You must update from OpenShift Container Platform 4.9 to 4.10 and then to 4.11. You cannot update from OpenShift Container Platform 4.9 to 4.11 directly. However, beginning with the update from OpenShift Container Platform 4.9 to 4.10 to 4.11, administrators who wish to update between two Extended Update Support (EUS) versions can do so incurring only a single reboot of non-control plane 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-control plane hosts complete the update to the odd-numbered version before moving forward.
- You can do a partial update by updating the worker or custom pool nodes to accommodate the time it takes for maintenance.
- 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.
- All the clusters might update using EUS channels for a conventional update without pools paused, but only clusters with non control-plane MachineConfigPools objects can do EUS-to-EUS update with pools paused.

5.1. 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-ecbb9582781c1091e1c9f19d50cf836c       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}}'
   ```

   **NOTE**

   You cannot pause the master pool.

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:

```bash
$ 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>

7. If necessary, upgrade OLM Operators using the Administrator perspective on the web console.

8. Update to 4.10, run the following command:

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

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>4.10.1</td>
<td>True</td>
<td>False</td>
<td>6m29s</td>
<td>Cluster version is 4.10.1</td>
</tr>
</tbody>
</table>

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

```bash
$ 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>4.10.1</td>
<td>True</td>
<td>False</td>
<td>6m29s</td>
<td>Cluster version is 4.10.1</td>
</tr>
</tbody>
</table>

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

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

**NOTE**

If pools are not unpaused, 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:

```bash
$ oc get mcp
```

Example output

<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>

OpenShift Container Platform 4.11 Updating clusters
CHAPTER 6. 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 update a cluster within a minor version. You can use the same instructions for updating a cluster between minor versions.

NOTE

Use the web console or `oc adm upgrade channel <channel>` to change the update channel. You can follow the steps in Updating a cluster using the CLI to complete the update after you change to a 4.11 channel.

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.
- Support for RHEL7 workers is removed in OpenShift Container Platform 4.11. You must replace RHEL7 workers with RHEL8 or RHCOS workers before upgrading to OpenShift Container Platform 4.11. Red Hat does not support in-place RHEL7 to RHEL8 updates for RHEL workers; those hosts must be replaced with a clean operating system install.
- 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 Updating 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.
- To accommodate the time it takes to update, you are able to do a partial update by updating the worker or custom pool nodes. You can pause and resume within the progress bar of each pool.
- 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.
- If your cluster uses manually maintained credentials with the AWS Secure Token Service (STS), obtain a copy of the `ccoctl` utility from the release image being updated to and use it to process any updated credentials. For more information, see Upgrading an OpenShift Container Platform cluster configured for manual mode with STS.
- If you run an Operator or you have configured any application with the pod disruption budget, you might experience an interruption during the upgrade process. If `minAvailable` is set to 1 in `PodDisruptionBudget`, the nodes are drained to apply pending machine configs which might block the eviction process. If several nodes are rebooted, all the pods might run on only one node, and the `PodDisruptionBudget` field can prevent the node drain.
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. 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 `oc debug`, `oc logs`, `oc exec`, and `oc attach`. You receive alerts in the Alerting UI of the OpenShift Container Platform web console if an MCP is paused when the certificates are rotated.

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.

6.3. UPGRADING CLUSTERS WITH MANUALLY MAINTAINED CREDENTIALS

The Cloud Credential Operator (CCO) **Upgradable** status for a cluster with manually maintained credentials is **False** by default.

- For minor releases, for example, from 4.10 to 4.11, this status prevents you from upgrading until you have addressed any updated permissions and annotated the `CloudCredential` resource to indicate that the permissions are updated as needed for the next version. This annotation changes the **Upgradable** status to **True**.
- For z-stream releases, for example, from 4.11.0 to 4.11.1, no permissions are added or changed, so the upgrade is not blocked.

Before upgrading a cluster with manually maintained credentials, you must create any new credentials for the release image that you are upgrading to. Additionally, you must review the required permissions for existing credentials and accommodate any new permissions requirements in the new release for those components.

**Procedure**

1. Extract and examine the `CredentialsRequest` custom resource for the new release. The "Manually creating IAM" section of the installation content for your cloud provider explains how to obtain and use the credentials required for your cloud.

2. Update the manually maintained credentials on your cluster:
   - Create new secrets for any `CredentialsRequest` custom resources that are added by the new release image.
   - If the `CredentialsRequest` custom resources for any existing credentials that are stored in secrets have changed their permissions requirements, update the permissions as required.

3. When all of the secrets are correct for the new release, indicate that the cluster is ready to upgrade:
   - Log in to the OpenShift Container Platform CLI as a user with the `cluster-admin` role.
b. Edit the **CloudCredential** resource to add an **upgradeable-to** annotation within the **metadata** field:

   ```bash
   $ oc edit cloudcredential cluster
   ```

   **Text to add**

   ```yaml
   metadata:
   annotations:
     cloudcredential.openshift.io/upgradeable-to: <version_number>
   ```

   Where `<version_number>` is the version you are upgrading to, in the format `x.y.z`. For example, **4.8.2** for OpenShift Container Platform 4.8.2.

   It may take several minutes after adding the annotation for the upgradeable status to change.

### Verification

1. In the **Administrator** perspective of the web console, navigate to **Administration → Cluster Settings**.

2. To view the CCO status details, click **cloud-credential** in the **Cluster Operators** list.

   a. If the **Upgradeable** status in the **Conditions** section is **False**, verify that the **upgradeable-to** annotation is free of typographical errors. When the **Upgradeable** status in the **Conditions** section is **True**, you can begin the OpenShift Container Platform upgrade.

### Additional resources

- Manually creating IAM for AWS
- Manually creating IAM for Azure
- Manually creating IAM for GCP

### 6.4. PAUSING A MACHINEHEALTHCHECK RESOURCE BY USING THE WEB CONSOLE

During the upgrade process, nodes in the cluster might become temporarily unavailable. In the case of worker nodes, the machine health check might identify such nodes as unhealthy and reboot them. To avoid rebooting such nodes, pause all the **MachineHealthCheck** resources before updating the cluster.

#### Prerequisites

- You have access to the cluster with **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.

#### Procedure

1. Log in to the OpenShift Container Platform web console.

3. To pause the machine health checks, add the cluster.x-k8s.io/paused="" annotation to each MachineHealthCheck resource. For example, to add the annotation to the machine-api-termination-handler resource, complete the following steps:
   a. Click the Options menu next to the machine-api-termination-handler and click Edit annotations.
   b. In the Edit annotations dialog, click Add more.
   c. In the Key and Value fields, add cluster.x-k8s.io/paused and "" values, respectively, and click Save.

6.5. ABOUT UPDATING SINGLE NODE OPENSIFT CONTAINER PLATFORM

You can update, or upgrade, a single-node OpenShift Container Platform cluster by using either the console or CLI.

However, note the following limitations:

- The prerequisite to pause the MachineHealthCheck resources is not required because there is no other node to perform the health check.

- Restoring a single-node OpenShift Container Platform cluster using an etcd backup is not officially supported. However, it is good practice to perform the etcd backup in case your upgrade fails. If your control plane is healthy, you might be able to restore your cluster to a previous state by using the backup.

- Updating a single-node OpenShift Container Platform cluster requires downtime and can include an automatic reboot. The amount of downtime depends on the update payload, as described in the following scenarios:
  - If the update payload contains an operating system update, which requires a reboot, the downtime is significant and impacts cluster management and user workloads.
  - If the update contains machine configuration changes that do not require a reboot, the downtime is less, and the impact on the cluster management and user workloads is lessened. In this case, the node draining step is skipped with single-node OpenShift Container Platform because there is no other node in the cluster to reschedule the workloads to.
  - If the update payload does not contain an operating system update or machine configuration changes, a short API outage occurs and resolves quickly.

**IMPORTANT**

There are conditions, such as bugs in an updated package, that can cause the single node to not restart after a reboot. In this case, the update does not rollback automatically.

Additional resources
For information on which machine configuration changes require a reboot, see the note in Understanding the Machine Config Operator.

### 6.6. 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.
- Pause all **MachineHealthCheck** resources.

**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-4.11**.

   ![IMPORTANT](image)

   **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](image)

   **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.

6.7. 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
db93436d-7b05-42cc-b856-43e11ad2d31a
upstream: '<update-server-url>'
```

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

3. Click Save.

Additional resources

- Understanding upgrade channels and releases
CHAPTER 7. 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). You can also update a cluster between minor versions by following the same instructions.

7.1. 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.

- Support for RHEL7 workers is removed in OpenShift Container Platform 4.11. You must replace RHEL7 workers with RHEL8 or RHCOS workers before upgrading to OpenShift Container Platform 4.11. Red Hat does not support in-place RHEL7 to RHEL8 updates for RHEL workers; those hosts must be replaced with a clean operating system install.

- 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 Updating 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.

- If your cluster uses manually maintained credentials with the AWS Secure Token Service (STS), obtain a copy of the ccoctl utility from the release image being updated to and use it to process any updated credentials. For more information, see Upgrading an OpenShift Container Platform cluster configured for manual mode with STS.

- Ensure that you address all Upgradeable=False conditions so the cluster allows an update to the next minor version. An alert displays at the top of the Cluster Settings page when you have one or more cluster Operators that cannot be upgraded. You can still update to the next available patch update for the minor release you are currently on.

- If you run an Operator or you have configured any application with the pod disruption budget, you might experience an interruption during the upgrade process. If minAvailable is set to 1 in PodDisruptionBudget, the nodes are drained to apply pending machine configs which might block the eviction process. If several nodes are rebooted, all the pods might run on only one node, and the PodDisruptionBudget field can prevent the node drain.
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

## 7.2. UPGRADING CLUSTERS WITH MANUALLY MAINTAINED CREDENTIALS

The Cloud Credential Operator (CCO) **Upgradable** status for a cluster with manually maintained credentials is **False** by default.

- For minor releases, for example, from 4.10 to 4.11, this status prevents you from upgrading until you have addressed any updated permissions and annotated the CloudCredential resource to indicate that the permissions are updated as needed for the next version. This annotation changes the Upgradable status to **True**.

- For z-stream releases, for example, from 4.11.0 to 4.11.1, no permissions are added or changed, so the upgrade is not blocked.

Before upgrading a cluster with manually maintained credentials, you must create any new credentials for the release image that you are upgrading to. Additionally, you must review the required permissions for existing credentials and accommodate any new permissions requirements in the new release for those components.

### Procedure

1. Extract and examine the **CredentialsRequest** custom resource for the new release. The "Manually creating IAM" section of the installation content for your cloud provider explains how to obtain and use the credentials required for your cloud.

2. Update the manually maintained credentials on your cluster:
   - Create new secrets for any **CredentialsRequest** custom resources that are added by the new release image.
   - If the **CredentialsRequest** custom resources for any existing credentials that are stored in secrets have changed their permissions requirements, update the permissions as required.

3. When all of the secrets are correct for the new release, indicate that the cluster is ready to upgrade:
   a. Log in to the OpenShift Container Platform CLI as a user with the **cluster-admin** role.
   b. Edit the **CloudCredential** resource to add an **upgradeable-to** annotation within the **metadata** field:
Where `<version_number>` is the version you are upgrading to, in the format `x.y.z`. For example, 4.8.2 for OpenShift Container Platform 4.8.2.

It may take several minutes after adding the annotation for the upgradeable status to change.

Verification

1. In the **Administrator** perspective of the web console, navigate to **Administration → Cluster Settings**.

2. To view the CCO status details, click **cloud-credential** in the **Cluster Operators** list.
   a. If the **Upgradeable** status in the **Conditions** section is **False**, verify that the **upgradeable-to** annotation is free of typographical errors. When the **Upgradeable** status in the **Conditions** section is **True**, you can begin the OpenShift Container Platform upgrade.

Additional resources

- Manually creating IAM for AWS
- Manually creating IAM for Azure
- Manually creating IAM for GCP

### 7.3. PAUSING A MACHINEHEALTHCHECK RESOURCE

During the upgrade process, nodes in the cluster might become temporarily unavailable. In the case of worker nodes, the machine health check might identify such nodes as unhealthy and reboot them. To avoid rebooting such nodes, pause all the **MachineHealthCheck** resources before updating the cluster.

**Prerequisites**

- Install the OpenShift CLI (**oc**).

**Procedure**

1. To list all the available **MachineHealthCheck** resources that you want to pause, run the following command:

   ```
   $ oc get machinehealthcheck -n openshift-machine-api
   ```

2. To pause the machine health checks, add the **cluster.x-k8s.io/paused=**"" annotation to the **MachineHealthCheck** resource. Run the following command:
The annotated MachineHealthCheck resource resembles the following YAML file:

```yaml
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
  annotations:
    cluster.x-k8s.io/paused: ""
spec:
  selector:
    matchLabels:
      role: worker
  unhealthyConditions:
  - type: "Ready"
    status: "Unknown"
    timeout: "300s"
  - type: "Ready"
    status: "False"
    timeout: "300s"
  maxUnhealthy: "40%"
  status:
    currentHealthy: 5
    expectedMachines: 5
```

$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""

### 7.4. ABOUT UPDATING SINGLE NODE OPENSHIFT CONTAINER PLATFORM

You can update, or upgrade, a single-node OpenShift Container Platform cluster by using either the console or CLI.

However, note the following limitations:

- The prerequisite to pause the MachineHealthCheck resources is not required because there is no other node to perform the health check.

- Restoring a single-node OpenShift Container Platform cluster using an etcd backup is not officially supported. However, it is good practice to perform the etcd backup in case your upgrade fails. If your control plane is healthy, you might be able to restore your cluster to a previous state by using the backup.

IMPORTANT

Resume the machine health checks after updating the cluster. To resume the check, remove the pause annotation from the MachineHealthCheck resource by running the following command:

```bash
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```
Updating a single-node OpenShift Container Platform cluster requires downtime and can include an automatic reboot. The amount of downtime depends on the update payload, as described in the following scenarios:

- If the update payload contains an operating system update, which requires a reboot, the downtime is significant and impacts cluster management and user workloads.
- If the update contains machine configuration changes that do not require a reboot, the downtime is less, and the impact on the cluster management and user workloads is lessened. In this case, the node draining step is skipped with single-node OpenShift Container Platform because there is no other node in the cluster to reschedule the workloads to.
- If the update payload does not contain an operating system update or machine configuration changes, a short API outage occurs and resolves quickly.

**IMPORTANT**

There are conditions, such as bugs in an updated package, that can cause the single node to not restart after a reboot. In this case, the update does not rollback automatically.

**Additional resources**

- For information on which machine configuration changes require a reboot, see the note in [Understanding the Machine Config Operator](#).

### 7.5. 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.
- Pause all **MachineHealthCheck** resources.

**Procedure**

1. Ensure that your cluster is available:

   ```
   $ oc get clusterversion
   ```

   **Example output**

   ```
   NAME     VERSION AVAILABLE PROGRESSING SINCE   STATUS
   version  4.9.23    True     False     158m    Cluster version is 4.9.23
   ```
2. 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 4.9.23

Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.10 (available channels: candidate-4.10, candidate-4.9, fast-4.10, fast-4.9, stable-4.10, stable-4.9)

Recommended updates:

<table>
<thead>
<tr>
<th>VERSION</th>
<th>IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9.24</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056d6d018032</td>
</tr>
<tr>
<td>4.9.25</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:2eafde815e543b92f70839972f585cc52aa7c37aa72d5f3c8bc886b0fd45707a</td>
</tr>
<tr>
<td>4.9.26</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:3c8d09dd08c30f27a54355f1787d09b83979cd31cf0b4c6ff56cd68814ef6c8</td>
</tr>
<tr>
<td>4.9.27</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:1c7db78e8cf05df2ceadd44f69c0e4b2c3234d5635c884a1e1b922c3bedae16</td>
</tr>
<tr>
<td>4.9.28</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:4084d94969b186e20189649b5affba7da59f79f3ef78b981eaf8b7a8</td>
</tr>
<tr>
<td>4.9.29</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:b04ca01d116f0134a102a57f86c67e5b1a3b5da1c4a580af91d521b8fa0aa6ec</td>
</tr>
<tr>
<td>4.9.31</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:2a28b8e8bb53d67dd8059421c39e36d9896b1e65cb54af81fb86ea9ac3bf2d7</td>
</tr>
<tr>
<td>4.9.32</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:ecdb60df547b857eaf0edcb55743dd64ca69aff1a61f1ac6f6d41203bedfa</td>
</tr>
<tr>
<td>4.10.3</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:7fe4cdd12be27e355a6e0e5ee5c5d8f923c1400d969df590806e8daabc56</td>
</tr>
<tr>
<td>4.10.4</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:9f9c3aaaca64f62af992bae5de1e984571c8b812f989b74c84dc630b064389fb7</td>
</tr>
<tr>
<td>4.10.5</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:ee6a9c7a11f883e90489229f6c76bc4da4af12f6546f4f9411d73a9896f02a</td>
</tr>
<tr>
<td>4.10.6</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:88b3946633e09dc23aa1f1a61edded8e52478edf34b51a7dbbb21d9abde251a</td>
</tr>
<tr>
<td>4.10.8</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:0696e249622b4d07d8f4501504b6c568ed6ba92416176a01a12b7f1827071</td>
</tr>
<tr>
<td>4.10.9</td>
<td>quay.io/openshift-release-dev/ocp-release@sha256:39f3600029b9b5c730d1167879ad6437352d51e72acc9fe80add3ec2a0d20400d</td>
</tr>
</tbody>
</table>
3. Based on your organization requirements, set the upgrade channel to **stable-4.10**, **fast-4.10**, or **eus-4.10**:

   $ oc adm upgrade channel <channel>

For example, to set the channel to **stable-4.10**:

   $ oc adm upgrade channel stable-4.10

**IMPORTANT**

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

4. Apply an update:

   - To update to the latest version:
     
     $$ oc adm upgrade --to-latest=true$$

   - To update to a specific version:
     
     $$ oc adm upgrade --to=<version>$$

   \(<version>\) is the update version that you obtained from the output of the `oc adm upgrade` command.

5. Review the status of the Cluster Version Operator:

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

**Example output**

```
{
  "channel": "stable-4.10",
  "clusterID": "990f7ab8-109b-4c95-8480-2bd1deec55ff",
  "desiredUpdate": {
    "force": false,
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:9c5f0df8b192a0d7b46cd5f6a4da2289c155fd5302dec7954f8f06c878160b8b",
```
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

```
[  
  {  
    "completionTime": "null",
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b7",
    "startTime": "2021-01-28T20:30:50Z",
    "state": "Partial",
    "verified": true,
    "version": "4.10.13"
  },
  
  {  
    "completionTime": "2021-01-28T20:30:50Z",
    "image": "quay.io/openshift-release-dev/ocp-release@sha256:b8fa13e09d869089fc5957c32b02b7d3792a0b6f36693432acc0409615ab23b7",
    "startTime": "2021-01-28T17:38:10Z",
    "state": "Completed",
    "verified": false,
    "version": "4.9.23"
  }
]
```

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:

```
$ oc get clusterversion
```

Example output

```
NAME  VERSION  AVAILABLE  PROGRESSING  SINCE  STATUS
version 4.10.13  True  False  2m  Cluster version is 4.10.13
```
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 updated before deploying workloads that rely on a new feature:

```
$ oc get nodes
```

**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-10-0-168-251.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>82m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-170-223.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>82m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-179-95.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>70m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-182-134.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>70m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-211-16.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>82m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-250-100.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>69m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

**Additional resources**

- Understanding upgrade channels and releases

### 7.6. UPDATING ALONG A CONDITIONAL UPGRADE PATH

You can update along a recommended conditional upgrade path using the web console or the OpenShift CLI (`oc`). When a conditional update is not recommended for your cluster, you can update along a conditional upgrade path using the OpenShift CLI (`oc`) 4.10 or later.

**Procedure**

1. To view the description of the update when it is not recommended because a risk might apply, run the following command:

   ```
   $ oc adm upgrade --include-not-recommended
   ```

2. If the cluster administrator evaluates the potential known risks and decides it is acceptable for the current cluster, then the administrator can waive the safety guards and proceed the update by running the following command:

   ```
   $ oc adm upgrade --allow-not-recommended --to <version> <.>
   ```

   `<.>` `<version>` is the supported but not recommended update version that you obtained from the output of the previous command.

**Additional resources**

- Upgrade channels and release paths

### 7.7. 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](https://api.openshift.com/api/upgrades_info/v1/graph).
Procedure

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

  ```
  $ oc patch clusterversion/version --patch '{"spec":{"upstream":"<update-server-url>"}}' --type=merge
  
  clusterversion.config.openshift.io/version patched
  ```

The `<update-server-url>` variable specifies the URL for the update server.
CHAPTER 8. 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 unpause, 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 MCO cannot push the newly rotated certificates to those nodes. This causes failure in multiple `oc` commands, including `oc debug`, `oc logs`, `oc exec`, and `oc attach`. You receive alerts in the Alerting UI of the OpenShift Container Platform web console if an MCP is paused when the certificates are rotated.

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.

8.1. 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 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.

8.2. 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 unpaused when the CA certificate rotation happens. If the MCPs are paused, the MCO cannot push the newly rotated certificates to those nodes. This causes the cluster to become degraded and causes failure in multiple `oc` commands, including `oc debug`, `oc logs`, `oc exec`, and `oc attach`. You receive alerts in the Alerting UI of the OpenShift Container Platform web console if an MCP is paused when the certificates are rotated.

4. Perform the cluster update. The update process updates the MCPs that are not paused, including the control plane 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.

### 8.3. 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
   
   ci-ln-pwnll6b-f76d1-s8t9n-worker-a-s75z4
   ci-ln-pwnll6b-f76d1-s8t9n-worker-b-dglj2
   ci-ln-pwnll6b-f76d1-s8t9n-worker-c-ldbmr
   
   Example output
   ```

   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:

   ```
   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] }
   ```
Specify a name for the MCP.

Specify the worker and custom MCP name.

Specify the custom label you added to the nodes that you want in this pool.

Example output

```
$ oc create -f <file_name>
machineconfigpool.machineconfiguration.openshift.io/workerpool-canary created
```

Example output

```
NAME                CONFIG                                      UPDATED   UPDATING
master              rendered-master-b0bb90c4921860f2a5d8a2f8137c1867          True
                   False      False      3              3                   3                     0                      97m
workerpool-canary  rendered-workerpool-canary-87ba3dec1ad78cb6aecebf7fbb476a36 True      False      False      1              1                   1                     0                      2m42s
                   rendered-worker-87ba3dec1ad78cb6aecebf7fbb476a36              True
                   False      False      2              2                   2                     0                      97m
```

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.

## 8.4. 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.
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 certificate rotation happens. If the MCPs are paused, the MCO cannot push the newly rotated certificates to those nodes. This causes the cluster to become degraded and causes failure in multiple `oc` commands, including `oc debug`, `oc logs`, `oc exec`, and `oc attach`. You receive alerts in the Alerting UI of the OpenShift Container Platform web console if an MCP is paused when the certificates are rotated.

To pause an MCP:

1. Patch the MCP that you want paused:

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

   For example:

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

   Example output

   ```
   machineconfigpool.machineconfiguration.openshift.io/workerpool-canary patched
   ```

**8.5. 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.

**8.6. 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:

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

   For example:
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.

### 8.6.1. 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.

### 8.7. 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-f76d1-kl2tq-worker-a-j2ssz node-role.kubernetes.io/workerpool-canary-
   ```

   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>OpenShift Container Platform 4.11 Updating clusters</td>
<td></td>
<td></td>
<td></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 9. 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.

9.1. 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.

- Support for RHEL7 workers is removed in OpenShift Container Platform 4.11. You must replace RHEL7 workers with RHEL8 or RHCOS workers before upgrading to OpenShift Container Platform 4.11. Red Hat does not support in-place RHEL7 to RHEL8 updates for RHEL workers; those hosts must be replaced with a clean operating system install.

- 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.

- If your cluster uses manually maintained credentials with the AWS Secure Token Service (STS), obtain a copy of the ccoctl utility from the release image being updated to and use it to process any updated credentials. For more information, see Upgrading an OpenShift Container Platform cluster configured for manual mode with STS.

- If you run an Operator or you have configured any application with the pod disruption budget, you might experience an interruption during the upgrade process. If minAvailable is set to 1 in PodDisruptionBudget, the nodes are drained to apply pending machine configs which might block the eviction process. If several nodes are rebooted, all the pods might run on only one node, and the PodDisruptionBudget field can prevent the node drain.

Additional resources

- Support policy for unmanaged Operators

9.2. 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.

- Pause all MachineHealthCheck resources.

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-4.11.

   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 upgrade playbook against each RHEL machine as it enters the NotReady state for the cluster to finish updating.

9.3. 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.

9.3.1. 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.

9.3.2. 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"
```

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.

**9.3.3. 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><strong>openshift_node_pre_cordon_hook</strong></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><strong>openshift_node_pre_upgrade_hook</strong></td>
<td>● Runs after each node is cordoned but before it is updated.</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><strong>openshift_node_pre_uncordon_hook</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Runs after each node is updated but before it is uncordoned.</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><strong>openshift_node_post_upgrade_hook</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Runs after each node uncordoned. It is the last node update action.</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>
</tbody>
</table>

**9.4. 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**

Red Hat Enterprise Linux (RHEL) versions 8.4 and 8.5 are supported for RHEL compute machines.
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.

**IMPORTANT**

You cannot upgrade RHEL 7 compute machines to RHEL 8. You must deploy new RHEL 8 hosts, and the old RHEL 7 hosts should be removed.

**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:

   ```bash
   # systemctl disable --now firewalld.service
   # subscription-manager repos --disable=rhocp-4.10-for-rhel-8-x86_64-rpms
   --disable=ansible-2.9-for-rhel-8-x86_64-rpms
   --enable=rhocp-4.11-for-rhel-8-x86_64-rpms
   ```

   **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 4.11:

   a. On the machine that you run the Ansible playbooks, update the required repositories:
As of OpenShift Container Platform 4.11, the Ansible playbooks are provided only for RHEL 8. If a RHEL 7 system was used as a host for the OpenShift Container Platform 4.10 Ansible playbooks, you must either upgrade the Ansible host to RHEL 8, or create a new Ansible host on a RHEL 8 system and copy over the inventories from the old Ansible host.

b. On the machine that you run the Ansible playbooks, update the Ansible package:

```yaml
# yum swap ansible ansible-core
```

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

```yaml
# yum update openshift-ansible openshift-clients
```

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

```bash
# subscription-manager repos --disable=rhocp-4.10-for-rhel-8-x86_64-rpms
   --enable=rhocp-4.11-for-rhel-8-x86_64-rpms
```

3. Update a RHEL worker machine:

a. Review your Ansible inventory file at `/<path>/inventory/hosts` and update its contents so that the RHEL 8 machines are 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-rhel8-0.example.com
mycluster-rhel8-1.example.com
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

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

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

c. Run the `upgrade` playbook:

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

For `<path>`, specify the path to the Ansible inventory file that you created.
The upgrade playbook only upgrades the OpenShift Container Platform packages. It does not update the operating system packages.

4. 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>v1.24.0</td>
</tr>
<tr>
<td>mycluster-control-plane-1</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>mycluster-control-plane-2</td>
<td>Ready</td>
<td>master</td>
<td>145m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>mycluster-rhel8-0</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>mycluster-rhel8-1</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>mycluster-rhel8-2</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>mycluster-rhel8-3</td>
<td>Ready</td>
<td>worker</td>
<td>98m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

5. Optional: Update the operating system packages that were not updated by the upgrade playbook. To update packages that are not on 4.11, 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 4.11.
CHAPTER 10. 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.

10.1. 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.

10.1.1. 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.
- If you run an Operator or you have configured any application with the pod disruption budget, you might experience an interruption during the upgrade process. If minAvailable is set to 1 in PodDisruptionBudget, the nodes are drained to apply pending machine configs which might block the eviction process. If several nodes are rebooted, all the pods might run on only one node, and the PodDisruptionBudget field can prevent the node drain.

10.1.2. 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.

10.1.2.1. 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 4.11. 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 architecture in the Product Variant drop-down menu.
3. Select the appropriate version in the Version drop-down menu.
4. Click Download Now next to the OpenShift v4.11 Linux Client entry and save the file.
5. Unpack the archive:
   
   $ tar xvzf <file>

6. 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 v4.11 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:
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 v4.11 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:

```bash
C:\> oc <command>
```

10.1.2.2. 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.

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

   ```
   $ cat ./pull-secret | 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"
   }
   }
   }
   ```

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

   ```
   $ echo -n '<user_name>:<password>' | base64 -w0
   ```
   
   For `<user_name>` and `<password>`, specify the user name and password that you configured for your registry.

4. Edit the JSON file and add a section that describes your registry to it.
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:8443`.

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

The file resembles the following example:

```json
{
  "auths": { 
    "<mirror_registry>": { 
      "auth": "<credentials>", 
      "email": "you@example.com"
    }, 
  
  "<mirror_registry>": { 
    "auth": "<credentials>", 
    "email": "you@example.com"
  }, 
  
  "<mirror_registry>": { 
    "auth": "<credentials>", 
    "email": "you@example.com"
  }, 
  
  "<mirror_registry>": { 
    "auth": "<credentials>", 
    "email": "you@example.com"
  }, 
  
  "<mirror_registry>": { 
    "auth": "<credentials>", 
    "email": "you@example.com"
  }
}
```

10.1.3. Mirroring the OpenShift Container Platform image repository

You must mirror container images onto a mirror registry before you can update a cluster in a restricted network 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.

There are two supported methods for mirroring images onto a mirror registry:

- Using the `oc-mirror` OpenShift CLI (`oc`) plug-in
- Using the `oc adm release mirror` command

Choose one of the following supported options.
10.1.3.1. Mirroring resources using the oc-mirror plug-in

Use the oc-mirror OpenShift CLI (oc) plug-in to mirror images onto a mirror registry. Compared to using `oc adm release mirror`, the oc-mirror plug-in has the following advantages:

- It is simpler to use.
- It can mirror content other than container images.
- After mirroring images for the first time, it is easier to update images in the registry.

Procedure

1. Navigate to the Mirroring images for a disconnected installation using the oc-mirror plug-in page of the documentation.

2. Follow the instructions on that page to mirror resources onto a mirror registry.
   - If you are using oc-mirror for the first time, follow the instructions on that page up until and including the section titled Installing the ImageContentSourcePolicy and CatalogSource resources into the cluster.
   - If you have already used oc-mirror to mirror resources onto a registry, follow the instructions in the section titled Keeping your mirror registry content updated.

Additional resources

- Mirroring images for a disconnected installation using the oc-mirror plug-in

10.1.3.2. Mirroring images using the oc adm release mirror command

Procedure

1. Use the Red Hat OpenShift Container Platform Upgrade Graph visualizer and update planner 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:

   ```
   $ 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:

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

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

Export the path to your registry 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.

Export the release mirror:

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

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

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

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

3. Review the images and configuration manifests to mirror:

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.

```
$ LOCAL_REPOSITORY='<local_repository_name>'

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

$ PRODUCT_REPO='openshift-release-dev'

Export the name of the repository to mirror:

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

$ LOCAL_SECRET_JSON='<path_to_pull_secret>'

Export the path to your registry 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.

Export the release mirror:

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

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

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

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

3. Review the images and configuration manifests to mirror:

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.

```
$ LOCAL_REPOSITORY='<local_repository_name>'

$ PRODUCT_REPO='openshift-release-dev'

$ LOCAL_SECRET_JSON='<path_to_pull_secret>'

$ RELEASE_NAME="ocp-release"

$ ARCHITECTURE=<server_architecture>

$ REMOVABLE_MEDIA_PATH=<path> ¹

¹ Specify the full path, including the initial forward slash (/) character.

CHAPTER 10. UPDATING A RESTRICTED NETWORK CLUSTER

59
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}
```

1 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>
```

1 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 ${LOCAL_SECRET_JSON} --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} --apply-release-image-signature
```

**NOTE**

If you include the `--apply-release-image-signature` option, do not create the config map for image signature verification.

### 10.1.4. Pausing a MachineHealthCheck resource

During the upgrade process, nodes in the cluster might become temporarily unavailable. In the case of worker nodes, the machine health check might identify such nodes as unhealthy and reboot them. To avoid rebooting such nodes, pause all the **MachineHealthCheck** resources before updating the cluster.

#### Prerequisites

- Install the OpenShift CLI (**oc**).
Procedure

1. To list all the available MachineHealthCheck resources that you want to pause, run the following command:

   
   $ oc get machinehealthcheck -n openshift-machine-api

2. To pause the machine health checks, add the \texttt{cluster.x-k8s.io/paused=""} annotation to the MachineHealthCheck resource. Run the following command:

   
   $ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""

The annotated MachineHealthCheck resource resembles the following YAML file:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: example
  namespace: openshift-machine-api
  annotations:
    cluster.x-k8s.io/paused: ""
spec:
  selector:
    matchLabels:
      role: worker
  unhealthyConditions:
  - type: "Ready"
    status: "Unknown"
    timeout: "300s"
  - type: "Ready"
    status: "False"
    timeout: "300s"
  maxUnhealthy: "40%"
  status:
    currentHealthy: 5
    expectedMachines: 5
```

\textbf{IMPORTANT}

Resume the machine health checks after updating the cluster. To resume the check, remove the pause annotation from the MachineHealthCheck resource by running the following command:

   
   $ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused=""  

10.1.5. Upgrading the restricted network cluster

Update the restricted network cluster to the OpenShift Container Platform version that you downloaded the release images for.
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`).
- Pause all `MachineHealthCheck` resources.

Procedure

- Update the cluster:

```
$ oc adm upgrade --allow-explicit-upgrade --to-image
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}<sha256_sum_value>
```

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.

10.1.6. 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.

### 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:

   ```bash
   $ skopeo copy \
   docker://registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba6 \
   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:

```yaml
apiVersion: operator.openshift.io/v1alpha1
kind: 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
  - mirrors:
    - mirror.example.net/image
    source: registry.example.com/example/myimage
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example
  - mirrors:
    - mirror.example.net/registry-example-com
    source: registry.example.com
```

1. Indicates the name of the image registry and repository.
2. Indicates multiple mirror repositories for each target repository. If one mirror is down, the target repository can use another mirror.
3. Indicates the registry and repository containing the content that is mirrored.
4. 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.
5. If you configure the registry name, the ImageContentSourcePolicy resource is applied to all repositories from a source registry to a mirror registry.
6. Pulls the image mirror.example.net/image@sha256:....
7. Pulls the image myimage in the source registry namespace from the mirror mirror.example.net/myimage@sha256:....
Pulls the image `registry.example.com/example/myimage` from the mirror registry `mirror.example.net/registry-example-com/example/myimage@sha256:...`. The

4. Create the new **ImageContentSourcePolicy** object:

   ```bash
   $ 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>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-138-148.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-139-122.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-147-35.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>7m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-153-12.ec2.internal</td>
<td>Ready</td>
<td>worker</td>
<td>7m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>ip-10-0-154-10.ec2.internal</td>
<td>Ready</td>
<td>master</td>
<td>11m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

   The **Imagecontentsourcepolicy** resource does not restart the nodes.

   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. Change your root directory to `/host`:

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

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

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

   Example output

   ```
   unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
   short-name-mode = ""
   ```

   ```
   [
   ```

   [registry]
   ```

   ```
   ```
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.

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455eba6
```

OpenShift Container Platform 4.11 Updating clusters
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.

10.1.7. 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>`:

   ```bash
   $ 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:v4.11`

   - `<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.

10.2. ADDITIONAL RESOURCES

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

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

10.3.1. 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 update 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 updates the affected nodes alphabetically by zone, based on the `topology.kubernetes.io/zone` label. If a zone has more than one node, the oldest nodes are updated first. For nodes that do not use zones, such as in bare metal deployments, the nodes are upgraded by age, with the oldest nodes updated first. The MCO updates the number of nodes as specified by the `maxUnavailable` field on the machine configuration pool at a time. 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.

**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.
10.3.2. Prerequisites

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

10.3.3. 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
group: core
kind: ConfigMap
metadata:
  name: my-registry-ca
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 `..`

10.3.4. 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.

**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:
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>
```

1. Provide the new registry. You can include multiple repositories within the same registry, for example: `$ registry="<registry/my-namespace/my-repository>"`.

2. Provide the credentials of the new registry.

3. 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>
```

1. 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.

**NOTE**

As of OpenShift Container Platform 4.7.4, changes to the global pull secret no longer trigger a node drain or reboot.

10.3.5. 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](#).

10.3.6. 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**.

10.3.7. 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"
      ```

      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
```

For example:

```bash
$ 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`:

```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"
```

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:
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
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>DISPLAY</th>
<th>VERSION</th>
<th>REPLACES</th>
<th>PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>update-service-operator.v4.6.0</td>
<td>OpenShift Update Service</td>
<td>4.6.0</td>
<td></td>
<td>Succeeded</td>
</tr>
</tbody>
</table>
   ...                             |
   
   If the OpenShift Update Service Operator is listed, the installation was successful. The version number might be different than shown.

### 10.3.8. 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 xvzf 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`:

   ```bash
   $ 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`: 

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

   For example:

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

10.3.9. Mirroring the OpenShift Container Platform image repository

You must mirror container images onto a mirror registry before you can update a cluster in a restricted network 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.

There are two supported methods for mirroring images onto a mirror registry:

- Using the `oc-mirror` OpenShift CLI (`oc`) plug-in
- Using the `oc adm release mirror` command

Choose one of the following supported options.

10.3.9.1. Mirroring resources using the `oc-mirror` plug-in

Use the `oc-mirror` OpenShift CLI (`oc`) plug-in to mirror images onto a mirror registry. Compared to using `oc adm release mirror`, the `oc-mirror` plug-in has the following advantages:

- It is simpler to use.
- It can mirror content other than container images.
- After mirroring images for the first time, it is easier to update images in the registry.

Procedure

1. Navigate to the Mirroring images for a disconnected installation using the `oc-mirror` plug-in page of the documentation.
2. Follow the instructions on that page to mirror resources onto a mirror registry.
   - If you are using `oc-mirror` for the first time, follow the instructions on that page up until and including the section titled Installing the ImageContentSourcePolicy and CatalogSource resources into the cluster.
   - If you have already used `oc-mirror` to mirror resources onto a registry, follow the instructions in the section titled Keeping your mirror registry content updated.

Additional resources

- Mirroring images for a disconnected installation using the `oc-mirror` plug-in

10.3.9.2. Mirroring images using the `oc adm release mirror` command

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

```bash
$ podman push registry.example.com/openshift/graph-data:latest
```
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, you have specified a Subject Alternative Name in the certificates.

Procedure

Complete the following steps on the mirror host:

1. Review the OpenShift Container Platform downloads page to determine the version of OpenShift Container Platform to which you want to update and determine the corresponding tag on the Repository Tags 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:

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

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

f. 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.

g. Export the release mirror:

```bash
$ 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`:

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

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

```bash
$ REMOVABLE_MEDIA_PATH=<path> ¹
```

³ 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:

```bash
$ 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}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OC
  P_RELEASE}-${ARCHITECTURE} --dry-run
```

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

```bash
$ 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}:${OCP_RELEASE}-${ARCHITECTURE}
  ${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_RELEASE}-${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 ${LOCAL_SECRET_JSON} \  
  --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} \  
  --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \  
  --to-release-image=${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}
```

### 10.3.10. Creating an OpenShift Update Service application

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

#### 10.3.10.1. 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**.

**10.3.10.2. 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`:

   ```
   $ 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`:

   ```
   $ 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\("n\")\' updateservice "$NAME")]; SCHEME=$(HTTP_CODE) = http -o "$SCHEME") = https; if test "$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 4.11, use `stable-4.11`:

   ```
   $ 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.

10.3.10.3. 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:

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

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

   ```bash
   $ NAME=service
   ```

3. Obtain the policy engine route:

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

4. Set the patch for the pull graph data:

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

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

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

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

10.3.11. Deleting an OpenShift Update Service application

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

10.3.11.1. 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.

10.3.11.2. 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`:

   ```shell
   $ oc get updateservice -n openshift-update-service
   
   NAME   AGE
   service 6s
   
   $ oc delete updateservice service -n openshift-update-service
   ```
10.3.12. 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.

10.3.12.1. 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.

10.3.12.2. 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`:

   ```
   $ oc project openshift-update-service
   ```

   **Example output**

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

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

   ```
   $ oc get operatorgroup
   ```

   **Example output**
3. Delete the operator group, for example, **openshift-update-service-fprx2**:  
   
   $ oc delete operatorgroup openshift-update-service-fprx2

   **Example output**

   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**

   NAME   PACKAGE               SOURCE                        CHANNEL
   update-service-operator   update-service-operator   updateservice-index-catalog   v1

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
CHAPTER 11. UPDATING HARDWARE ON NODES RUNNING ON VSPHERE

You must ensure that your nodes running in vSphere are running on the hardware version supported by OpenShift Container Platform. Currently, hardware version 13 or later is supported for vSphere virtual machines in a cluster.

You can update your virtual hardware immediately or schedule an update in vCenter.

IMPORTANT

Using hardware version 13 for your cluster nodes running on vSphere is now deprecated. This version is still fully supported, but support will be removed in a future version of OpenShift Container Platform. Hardware version 15 is now the default for vSphere virtual machines in OpenShift Container Platform.

11.1. UPDATING VIRTUAL HARDWARE ON VSPHERE

To update the hardware of your virtual machines (VMs) on VMware vSphere, update your virtual machines separately to reduce the risk of downtime for your cluster.

11.1.1. Updating the virtual hardware for control plane nodes on vSphere

To reduce the risk of downtime, it is recommended that control plane nodes be updated serially. This ensures that the Kubernetes API remains available and etcd retains quorum.

Prerequisites

- You have cluster administrator permissions to execute the required permissions in the vCenter instance hosting your OpenShift Container Platform cluster.
- Your vSphere ESXi hosts are version 6.7U3 or later.

Procedure

1. List the control plane nodes in your cluster.

   $ oc get nodes -l node-role.kubernetes.io/master

   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>control-plane-node-0</td>
<td>Ready</td>
<td>master</td>
<td>75m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>control-plane-node-1</td>
<td>Ready</td>
<td>master</td>
<td>75m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>control-plane-node-2</td>
<td>Ready</td>
<td>master</td>
<td>75m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

   Note the names of your control plane nodes.

2. Mark the control plane node as unschedulable.

   $ oc adm cordon <control_plane_node>
3. Shut down the virtual machine (VM) associated with the control plane node. Do this in the vSphere client by right-clicking the VM and selecting **Power → Shut Down Guest OS**. Do not shut down the VM using **Power Off** because it might not shut down safely.

4. Update the VM in the vSphere client. Follow [Upgrading a virtual machine to the latest hardware version](https://vmware.com) in the VMware documentation for more information.

5. Power on the VM associated with the control plane node. Do this in the vSphere client by right-clicking the VM and selecting **Power On**.

6. Wait for the node to report as **Ready**:

   ```bash
   $ oc wait --for=condition=Ready node/<control_plane_node>
   ```

7. Mark the control plane node as schedulable again:

   ```bash
   $ oc adm uncordon <control_plane_node>
   ```

8. Repeat this procedure for each control plane node in your cluster.

### 11.1.2. Updating the virtual hardware for compute nodes on vSphere

To reduce the risk of downtime, it is recommended that compute nodes be updated serially.

**NOTE**

Multiple compute nodes can be updated in parallel given workloads are tolerant of having multiple nodes in a **NotReady** state. It is the responsibility of the administrator to ensure that the required compute nodes are available.

#### Prerequisites

- You have cluster administrator permissions to execute the required permissions in the vCenter instance hosting your OpenShift Container Platform cluster.
- Your vSphere ESXi hosts are version 6.7U3 or later.

#### Procedure

1. List the compute nodes in your cluster.

   ```bash
   $ oc get nodes -l node-role.kubernetes.io/worker
   ```

   **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>compute-node-0</td>
<td>Ready</td>
<td>worker</td>
<td>30m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>compute-node-1</td>
<td>Ready</td>
<td>worker</td>
<td>30m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>compute-node-2</td>
<td>Ready</td>
<td>worker</td>
<td>30m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>

   Note the names of your compute nodes.

2. Mark the compute node as unschedulable:

   ```bash
   $ oc wait --for=condition=Ready node/<compute_node>
   $ oc adm uncordon <compute_node>
   ```
3. Evacuate the pods from the compute node. There are several ways to do this. For example, you can evacuate all or selected pods on a node:

   $ oc adm cordon <compute_node>

   $ oc adm drain <compute_node> [--pod-selector=<pod_selector>]

   See the ”Understanding how to evacuate pods on nodes” section for other options to evacuate pods from a node.

4. Shut down the virtual machine (VM) associated with the compute node. Do this in the vSphere client by right-clicking the VM and selecting **Power → Shut Down Guest OS** Do not shut down the VM using **Power Off** because it might not shut down safely.

5. Update the VM in the vSphere client. Follow [Upgrading a virtual machine to the latest hardware version](#) in the VMware documentation for more information.

6. Power on the VM associated with the compute node. Do this in the vSphere client by right-clicking the VM and selecting **Power On**.

7. Wait for the node to report as **Ready**:

   $ oc wait --for=condition=Ready node/<compute_node>

8. Mark the compute node as schedulable again:

   $ oc adm uncordon <compute_node>

9. Repeat this procedure for each compute node in your cluster.

### 11.1.3. Updating the virtual hardware for template on vSphere

**Prerequisites**

- You have cluster administrator permissions to execute the required permissions in the vCenter instance hosting your OpenShift Container Platform cluster.

- Your vSphere ESXi hosts are version 6.7U3 or later.

**Procedure**

1. If the RHCOS template is configured as a vSphere template follow [Convert a Template to a Virtual Machine](#) in the VMware documentation prior to the next step.

   **NOTE**

   Once converted from a template, do not power on the virtual machine.

2. Update the VM in the vSphere client. Follow [Upgrading a virtual machine to the latest hardware version](#) in the VMware documentation for more information.

3. Convert the VM in the vSphere client from a VM to template. Follow [Convert a Virtual Machine to a Template in the vSphere Client](#) in the VMware documentation for more information.
11.2. SCHEDULING AN UPDATE FOR VIRTUAL HARDWARE ON VSPHERE

Virtual hardware updates can be scheduled to occur when a virtual machine is powered on or rebooted. You can schedule your virtual hardware updates exclusively in vCenter by following Schedule a Compatibility Upgrade for a Virtual Machine in the VMware documentation.

When scheduling an upgrade prior to performing an upgrade of OpenShift Container Platform, the virtual hardware update occurs when the nodes are rebooted during the course of the OpenShift Container Platform upgrade.
CHAPTER 12. UPDATING A CLUSTER THAT INCLUDES THE SPECIAL RESOURCE OPERATOR

When updating a cluster that includes the Special Resource Operator (SRO), it is important to consider whether the new kernel module version is compatible with the kernel modules currently loaded by the SRO. You can run a preflight check to confirm if the SRO will be able to upgrade the kernel modules.

12.1. CUSTOM RESOURCE AND VERIFICATION STATUS INFORMATION

The preflight check provides information on the status of the custom resource (CR) and the status of the verification.

Possible CR statuses

The possible CR statuses are as follows:

True
The Special Resource Operator (SRO) CR will upgrade without kernel compatibility issues.

False
There is an issue with kernel compatibility for the CR. The **Status Reason** field provides additional information on this.

Error
The status check was not completed due to an internal error. The **Status Reason** field provides additional information on debugging this.

Unknown
The check has not yet verified the status of the CR. This might be because the process has not yet reached the specific CR or not enough time has elapsed for the check to complete.

Possible verification statuses

The possible verification statuses are as follows:

True
The image exists and is compatible, or the image does not exist but there is a **BuildConfig** resource in place.

False
The image does not exist and there is no **BuildConfig** resource in place, or an image exists but it is not compatible with the new kernel version.

If the verification status is **False**, you can take one of the following steps:

- Create a prebuilt image with the correct name and check the status field again.
- Change the CR to include a **BuildConfig** resource.

12.2. RUNNING A PREFLIGHT CHECK FOR THE SPECIAL RESOURCE OPERATOR

You can use the following example procedure to check the compatibility of a kernel module version before updating a cluster that includes the Special Resource Operator (SRO).
Prerequisites

- You have a running OpenShift Container Platform cluster.
- You installed the OpenShift CLI (`oc`).
- You are logged in to the OpenShift CLI as a user with `cluster-admin` privileges.
- You installed the SRO.

Procedure

1. Create the following preflight validation custom resource definition (CRD) and save the YAML as `prevalidation.yaml`.

    ```yaml
    apiVersion: sro.openshift.io/v1beta1
    kind: PreflightValidation
    metadata:
        name: preflight
        namespace: preflight
    spec:
        updateImage: quay.io/openshift-release-dev/ocp-release@sha256:f7f252c39b64601c8ac3de737a584ba4f6016b1f4b17801d726ca2fd15492878
    ```

   Specify the name of the update image here.

2. Start the validation check by running the following command:

    ```bash
    $ oc apply -f prevalidation.yaml
    $ oc describe preflightvalidations.sro.openshift.io/v1beta1 preflight
    ```

Verification

- Check the status of the custom resource (CR) by running the following command:

    ```bash
    $ oc describe preflightvalidations.sro.openshift.io/v1beta1 preflight
    ```

Example output

The following is an example output in which `simple-oot` is a `SpecialResource` CR that is deployed on the cluster.

```
Status:
  Cr Statuses:
    Last Transition Time: 2022-08-02T08:48:45Z
    Name: simple-oot
    Status Reason: Verification successful, all driver-containers for the next kernel version are present
    Verification Stage: Image
    Verification Status: True
  Events: <none>
```

The preflight check continues to run until all CRs are verified. You can repeat the preceding command to check the status. After all CRs are verified, you should delete the preflight CR.
12.3. ADDITIONAL RESOURCES

- Special Resource Operator