



OpenShift Container Platform 4.18

Updating clusters

Updating OpenShift Container Platform clusters

OpenShift Container Platform 4.18 Updating clusters

Updating OpenShift Container Platform clusters

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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. UNDERSTANDING OPENSIFT UPDATES	5
1.1. INTRODUCTION TO OPENSIFT UPDATES	5
1.1.1. Common questions about update availability	5
1.1.2. About the OpenShift Update Service	7
1.1.3. Understanding cluster Operator condition types	8
1.1.4. Understanding cluster version condition types	9
1.1.5. Common terms	9
1.1.6. Additional resources	10
1.2. HOW CLUSTER UPDATES WORK	10
1.2.1. The Cluster Version Operator	10
1.2.1.1. The ClusterVersion object	10
Update availability data	11
1.2.1.2. Evaluation of update availability	13
1.2.2. Release images	14
1.2.3. Update process workflow	15
1.2.4. Understanding how manifests are applied during an update	16
1.2.5. Understanding how the Machine Config Operator updates nodes	17
1.3. UNDERSTANDING UPDATE CHANNELS AND RELEASES	19
1.3.1. Update channels	20
1.3.1.1. fast-4.18 channel	20
1.3.1.2. stable-4.18 channel	20
1.3.1.3. eus-4.y channel	21
1.3.1.4. candidate-4.18 channel	21
1.3.1.5. Update recommendations in the channel	21
1.3.1.6. Update recommendations and Conditional Updates	21
1.3.1.7. Choosing the correct channel for your cluster	22
1.3.1.8. Restricted network clusters	23
1.3.1.9. Switching between channels	23
1.4. UNDERSTANDING OPENSIFT CONTAINER PLATFORM UPDATE DURATION	23
1.4.1. Factors affecting update duration	23
1.4.2. Cluster update phases	24
1.4.2.1. Cluster Version Operator target update payload deployment	24
1.4.2.2. Machine Config Operator node updates	24
1.4.2.3. Example update duration of cluster Operators	25
1.4.3. Estimating cluster update time	27
1.4.4. Red Hat Enterprise Linux (RHEL) compute nodes	28
1.4.5. Additional resources	28
CHAPTER 2. PREPARING TO UPDATE A CLUSTER	29
2.1. PREPARING TO UPDATE TO OPENSIFT CONTAINER PLATFORM 4.18	29
2.1.1. Kubernetes API removals	29
2.1.2. Assessing the risk of conditional updates	29
2.1.3. etcd backups before cluster updates	29
2.1.4. Best practices for cluster updates	30
2.1.4.1. Choose versions recommended by the OpenShift Update Service	30
2.1.4.2. Address all critical alerts on the cluster	30
2.1.4.3. Ensure that the cluster is in an Upgradable state	30
2.1.4.3.1. SDN support removal	31
2.1.4.4. Ensure that enough spare nodes are available	31
2.1.4.5. Ensure that the cluster's PodDisruptionBudget is properly configured	31
2.2. PREPARING TO UPDATE A CLUSTER WITH MANUALLY MAINTAINED CREDENTIALS	32

2.2.1. Update requirements for clusters with manually maintained credentials	32
2.2.1.1. Cloud credential configuration options and update requirements by platform type	32
2.2.1.2. Determining the Cloud Credential Operator mode by using the web console	34
2.2.1.3. Determining the Cloud Credential Operator mode by using the CLI	37
2.2.2. Extracting and preparing credentials request resources	39
2.2.3. Configuring the Cloud Credential Operator utility for a cluster update	41
2.2.4. Updating cloud provider resources with the Cloud Credential Operator utility	42
2.2.5. Manually updating cloud provider resources	46
2.2.6. Indicating that the cluster is ready to upgrade	49
2.3. PREFLIGHT VALIDATION FOR KERNEL MODULE MANAGEMENT (KMM) MODULES	50
2.3.1. Validation kickoff	50
2.3.2. Validation lifecycle	50
2.3.3. Validation status	50
2.3.4. Preflight validation stages per Module	51
2.3.4.1. Image validation stage	51
2.3.4.2. Build validation stage	51
2.3.4.3. Sign validation stage	52
2.3.5. Example PreflightValidationOCP resource	52
2.4. PREPARING TO UPDATE FROM OPENSIFT CONTAINER PLATFORM 4.18 TO A NEWER VERSION	52
2.4.1. Migrating workloads off of package-based RHEL worker nodes	53
2.4.2. Identifying and removing RHEL worker nodes	54
2.4.3. Provisioning new RHCOS worker nodes	54
2.4.4. Preparing for Gateway API management succession by the Ingress Operator	55
CHAPTER 3. PERFORMING A CLUSTER UPDATE	57
3.1. UPDATING A CLUSTER USING THE CLI	57
3.1.1. Prerequisites	57
3.1.2. Pausing a MachineHealthCheck resource	58
3.1.3. About updating single node OpenShift Container Platform	59
3.1.4. Updating a cluster by using the CLI	60
3.1.5. Finding recommended update paths with oc adm upgrade recommend (Technology Preview)	62
3.1.6. Gathering cluster update status using oc adm upgrade status (Technology Preview)	64
3.1.7. Updating along a conditional update path	66
3.1.8. Changing the update server by using the CLI	66
3.2. UPDATING A CLUSTER USING THE WEB CONSOLE	67
3.2.1. Before updating the OpenShift Container Platform cluster	67
3.2.2. Changing the update server by using the web console	68
3.2.3. Pausing a MachineHealthCheck resource by using the web console	68
3.2.4. Updating a cluster by using the web console	69
3.2.5. Viewing conditional updates in the web console	71
3.2.6. Performing a canary rollout update	71
3.2.7. About updating single node OpenShift Container Platform	72
3.3. PERFORMING A CONTROL PLANE ONLY UPDATE	73
3.3.1. Performing a Control Plane Only update	73
3.3.1.1. Control Plane Only update using the web console	74
3.3.1.2. Control Plane Only update using the CLI	76
3.3.1.3. Control Plane Only updates for layered products and Operators installed through Operator Lifecycle Manager	78
3.4. PERFORMING A CANARY ROLLOUT UPDATE	79
3.4.1. Example Canary update strategy	79
Defining custom machine config pools	80
Updating the canary worker pool	80
Determining whether to proceed with the remaining worker pool updates	80

3.4.2. About the canary rollout update process and MCPs	81
Using custom machine config pools	81
Considerations when using custom machine config pools	81
3.4.3. About performing a canary rollout update	82
3.4.4. Creating machine config pools to perform a canary rollout update	83
3.4.5. Managing machine configuration inheritance for a worker pool canary	84
3.4.6. Pausing the machine config pools	88
3.4.7. Performing the cluster update	88
3.4.8. Unpausing the machine config pools	88
3.4.9. Moving a node to the original machine config pool	89
3.5. UPDATING A CLUSTER THAT INCLUDES RHEL COMPUTE MACHINES	90
3.5.1. Prerequisites	90
3.5.2. Updating a cluster by using the web console	91
3.5.3. Optional: Adding hooks to perform Ansible tasks on RHEL machines	93
3.5.3.1. About Ansible hooks for updates	93
3.5.3.2. Configuring the Ansible inventory file to use hooks	93
3.5.3.3. Available hooks for RHEL compute machines	94
3.5.4. Updating RHEL compute machines in your cluster	95
3.6. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT	97
3.7. UPDATING HARDWARE ON NODES RUNNING ON VSPHERE	97
3.7.1. Updating virtual hardware on vSphere	98
3.7.1.1. Updating the virtual hardware for control plane nodes on vSphere	98
3.7.1.2. Updating the virtual hardware for compute nodes on vSphere	99
3.7.1.3. Updating the virtual hardware for template on vSphere	100
3.7.2. Scheduling an update for virtual hardware on vSphere	101
3.8. MIGRATING TO A CLUSTER WITH MULTI-ARCHITECTURE COMPUTE MACHINES	101
3.8.1. Migrating to a cluster with multi-architecture compute machines using the CLI	101
3.8.2. Migrating the x86 control plane to arm64 architecture on Amazon Web Services	103
3.9. UPDATING THE BOOT LOADER ON RHCOS NODES USING BOOTUPD	106
3.9.1. Updating the boot loader manually	106
3.9.2. Updating the bootloader automatically via a machine config	107
CHAPTER 4. TROUBLESHOOTING A CLUSTER UPDATE	109
4.1. GATHERING DATA ABOUT YOUR CLUSTER UPDATE	109
4.1.1. Gathering log data for a support case	109
4.1.2. Gathering cluster update status using oc adm upgrade status (Technology Preview)	109
4.1.3. Gathering ClusterVersion history	110
4.1.3.1. Gathering ClusterVersion history in the OpenShift Container Platform web console	110
4.1.3.2. Gathering ClusterVersion history using the OpenShift CLI (oc)	111

CHAPTER 1. UNDERSTANDING OPENSIFT UPDATES

1.1. INTRODUCTION TO OPENSIFT UPDATES

With OpenShift Container Platform 4, you can update an OpenShift Container Platform cluster with a single operation by using the web console or the OpenShift CLI (**oc**). Platform administrators can view new update options either by going to **Administration** → **Cluster Settings** in the web console or by looking at the output of the **oc adm upgrade** command.

Red Hat hosts a public OpenShift Update Service (OSUS), which serves a graph of update possibilities based on the OpenShift Container Platform release images in the official registry. The graph contains update information for any public OCP release. OpenShift Container Platform clusters are configured to connect to the OSUS by default, and the OSUS responds to clusters with information about known update targets.

An update begins when either a cluster administrator or an automatic update controller edits the custom resource (CR) of the Cluster Version Operator (CVO) with a new version. To reconcile the cluster with the newly specified version, the CVO retrieves the target release image from an image registry and begins to apply changes to the cluster.



NOTE

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

The target release image contains manifest files for all cluster components that form a specific OCP version. When updating the cluster to a new version, the CVO applies manifests in separate stages called Runlevels. Most, but not all, manifests support one of the cluster Operators. As the CVO applies a manifest to a cluster Operator, the Operator might perform update tasks to reconcile itself with its new specified version.

The CVO monitors the state of each applied resource and the states reported by all cluster Operators. The CVO only proceeds with the update when all manifests and cluster Operators in the active Runlevel reach a stable condition. After the CVO updates the entire control plane through this process, the Machine Config Operator (MCO) updates the operating system and configuration of every node in the cluster.

1.1.1. Common questions about update availability

There are several factors that affect if and when an update is made available to an OpenShift Container Platform cluster. The following list provides common questions regarding the availability of an update:

What are the differences between each of the update channels?

- A new release is initially added to the **candidate** channel.
- After successful final testing, a release on the **candidate** channel is promoted to the **fast** channel, an errata is published, and the release is now fully supported.
- After a delay, a release on the **fast** channel is finally promoted to the **stable** channel. This delay represents the only difference between the **fast** and **stable** channels.

**NOTE**

For the latest z-stream releases, this delay may generally be a week or two. However, the delay for initial updates to the latest minor version may take much longer, generally 45-90 days.

- Releases promoted to the **stable** channel are simultaneously promoted to the **eus** channel. The primary purpose of the **eus** channel is to serve as a convenience for clusters performing a Control Plane Only update.

Is a release on the **stable channel safer or more supported than a release on the **fast** channel?**

- If a regression is identified for a release on a **fast** channel, it will be resolved and managed to the same extent as if that regression was identified for a release on the **stable** channel.
- The only difference between releases on the **fast** and **stable** channels is that a release only appears on the **stable** channel after it has been on the **fast** channel for some time, which provides more time for new update risks to be discovered.
- A release that is available on the **fast** channel always becomes available on the **stable** channel after this delay.

What does it mean if an update has known issues?

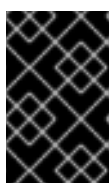
- Red Hat continuously evaluates data from multiple sources to determine whether updates from one version to another have any declared issues. Identified issues are typically documented in the version's release notes. Even if the update path has known issues, customers are still supported if they perform the update.
- Red Hat does not block users from updating to a certain version. Red Hat may declare conditional update risks, which may or may not apply to a particular cluster.
 - Declared risks provide cluster administrators more context about a supported update. Cluster administrators can still accept the risk and update to that particular target version.

What if I see that an update to a particular release is no longer recommended?

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

How long until the next z-stream release is made available on the fast and stable channels?

- While the specific cadence can vary based on a number of factors, new z-stream releases for the latest minor version are typically made available about every week. Older minor versions, which have become more stable over time, may take much longer for new z-stream releases to be made available.

**IMPORTANT**

These are only estimates based on past data about z-stream releases. Red Hat reserves the right to change the release frequency as needed. Any number of issues could cause irregularities and delays in this release cadence.

- Once a z-stream release is published, it also appears in the **fast** channel for that minor version. After a delay, the z-stream release may then appear in that minor version's **stable** channel.

Additional resources

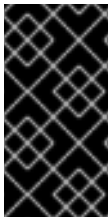
- [Understanding update channels and releases](#)

1.1.2. About the OpenShift Update Service

The OpenShift Update Service (OSUS) provides update recommendations 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 corresponding release image to update your cluster. The release artifacts are hosted in Quay as container images.

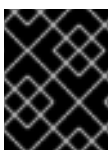
To allow the OpenShift Update Service to provide only compatible updates, a release verification pipeline drives automation. Each release artifact is verified for compatibility with supported cloud platforms and system architectures, as well as other component packages. After the pipeline confirms the 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 related to the update path, such as incompatibility or availability.

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 updating 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 specified by the **maxUnavailable** field on the machine configuration pool and marks them 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 updated 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.

**WARNING**

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

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.

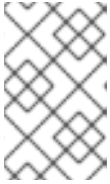
The OpenShift Update Service is composed of an Operator and one or more application instances.

1.1.3. Understanding cluster Operator condition types

The status of cluster Operators includes their condition type, which informs you of the current state of your Operator's health. The following definitions cover a list of some common ClusterOperator condition types. Operators that have additional condition types and use Operator-specific language have been omitted.

The Cluster Version Operator (CVO) is responsible for collecting the status conditions from cluster Operators so that cluster administrators can better understand the state of the OpenShift Container Platform cluster.

- Available: The condition type **Available** indicates that an Operator is functional and available in the cluster. If the status is **False**, at least one part of the operand is non-functional and the condition requires an administrator to intervene.
- Progressing: The condition type **Progressing** indicates that an Operator is actively rolling out new code, propagating configuration changes, or otherwise moving from one steady state to another. Operators do not report the condition type **Progressing** as **True** when they are reconciling a previous known state. If the observed cluster state has changed and the Operator is reacting to it, then the status reports back as **True**, since it is moving from one steady state to another.
- Degraded: The condition type **Degraded** indicates that an Operator has a current state that does not match its required state over a period of time. The period of time can vary by component, but a **Degraded** status represents persistent observation of an Operator's condition. As a result, an Operator does not fluctuate in and out of the **Degraded** state. There might be a different condition type if the transition from one state to another does not persist over a long enough period to report **Degraded**. An Operator does not report **Degraded** during the course of a normal update. An Operator may report **Degraded** in response to a persistent infrastructure failure that requires eventual administrator intervention.



NOTE

This condition type is only an indication that something may need investigation and adjustment. As long as the Operator is available, the **Degraded** condition does not cause user workload failure or application downtime.

- Upgradeable: The condition type **Upgradeable** indicates whether the Operator is safe to update based on the current cluster state. The message field contains a human-readable description of what the administrator needs to do for the cluster to successfully update. The CVO allows updates when this condition is **True**, **Unknown** or missing. When the **Upgradeable** status is **False**, only minor updates are impacted, and the CVO prevents the cluster from performing impacted updates unless forced.

1.1.4. Understanding cluster version condition types

The Cluster Version Operator (CVO) monitors cluster Operators and other components, and is responsible for collecting the status of both the cluster version and its Operators. This status includes the condition type, which informs you of the health and current state of the OpenShift Container Platform cluster.

In addition to **Available**, **Progressing**, and **Upgradeable**, there are condition types that affect cluster versions and Operators.

- Failing: The cluster version condition type **Failing** indicates that a cluster cannot reach its desired state, is unhealthy, and requires an administrator to intervene.
- Invalid: The cluster version condition type **Invalid** indicates that the cluster version has an error that prevents the server from taking action. The CVO only reconciles the current state as long as this condition is set.
- RetrievedUpdates: The cluster version condition type **RetrievedUpdates** indicates whether or not available updates have been retrieved from the upstream update server. The condition is **Unknown** before retrieval, **False** if the updates either recently failed or could not be retrieved, or **True** if the **availableUpdates** field is both recent and accurate.
- ReleaseAccepted: The cluster version condition type **ReleaseAccepted** with a **True** status indicates that the requested release payload was successfully loaded without failure during image verification and precondition checking.
- ImplicitlyEnabledCapabilities: The cluster version condition type **ImplicitlyEnabledCapabilities** with a **True** status indicates that there are enabled capabilities that the user is not currently requesting through **spec.capabilities**. The CVO does not support disabling capabilities if any associated resources were previously managed by the CVO.

1.1.5. 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.

Additional resources

- [Machine Config Overview](#)
- [Using the OpenShift Update Service in a disconnected environment](#)
- [Update channels](#)

1.1.6. Additional resources

- [How cluster updates work.](#)

1.2. HOW CLUSTER UPDATES WORK

The following sections describe each major aspect of the OpenShift Container Platform (OCP) update process in detail. For a general overview of how updates work, see the [Introduction to OpenShift updates](#).

1.2.1. The Cluster Version Operator

The Cluster Version Operator (CVO) is the primary component that orchestrates and facilitates the OpenShift Container Platform update process. During installation and standard cluster operation, the CVO is constantly comparing the manifests of managed cluster Operators to in-cluster resources, and reconciling discrepancies to ensure that the actual state of these resources match their desired state.

1.2.1.1. The ClusterVersion object

One of the resources that the Cluster Version Operator (CVO) monitors is the **ClusterVersion** resource.

Administrators and OpenShift components can communicate or interact with the CVO through the **ClusterVersion** object. The desired CVO state is declared through the **ClusterVersion** object and the current CVO state is reflected in the object's status.

**NOTE**

Do not directly modify the **ClusterVersion** object. Instead, use interfaces such as the **oc** CLI or the web console to declare your update target.

The CVO continually reconciles the cluster with the target state declared in the **spec** property of the **ClusterVersion** resource. When the desired release differs from the actual release, that reconciliation updates the cluster.

Update availability data

The **ClusterVersion** resource also contains information about updates that are available to the cluster. This includes updates that are available, but not recommended due to a known risk that applies to the cluster. These updates are known as conditional updates. To learn how the CVO maintains this information about available updates in the **ClusterVersion** resource, see the "Evaluation of update availability" section.

- You can inspect all available updates with the following command:

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

**NOTE**

The additional **--include-not-recommended** parameter includes updates that are available with known issues that apply to the cluster.

Example output

Cluster version is 4.13.40

Upstream is unset, so the cluster will use an appropriate default.

Channel: stable-4.14 (available channels: candidate-4.13, candidate-4.14, eus-4.14, fast-4.13, fast-4.14, stable-4.13, stable-4.14)

Recommended updates:

VERSION	IMAGE
4.14.27	quay.io/openshift-release-dev/ocp-release@sha256:4d30b359aa6600a89ed49ce6a9a5fdab54092bcb821a25480dfbc47e66af9ec
4.14.26	quay.io/openshift-release-dev/ocp-release@sha256:4fe7d4ccf4d967a309f83118f1a380a656a733d7fcee1dbaf4d51752a6372890
4.14.25	quay.io/openshift-release-dev/ocp-release@sha256:a0ef946ef8ae75aef726af1d9bbaad278559ad8cab2c1ed1088928a0087990b6
4.14.24	quay.io/openshift-release-dev/ocp-release@sha256:0a34eac4b834e67f1bca94493c237e307be2c0eae7b8956d4d8ef1c0c462c7b0
4.14.23	quay.io/openshift-release-dev/ocp-release@sha256:f8465817382128ec7c0bc676174bad0fb43204c353e49c146ddd83a5b3d58d92
4.13.42	quay.io/openshift-release-dev/ocp-release@sha256:dcf5c3ad7384f8bee3c275da8f886b0bc9aea7611d166d695d0cf0fff40a0b55
4.13.41	quay.io/openshift-release-dev/ocp-

```
release@sha256:dbb8aa0cf53dc5ac663514e259ad2768d8c82fd1fe7181a4cfb484e3ffdbd3ba
```

Updates with known issues:

Version: 4.14.22

Image: quay.io/openshift-release-dev/ocp-release@sha256:7093fa606debe63820671cc92a1384e14d0b70058d4b4719d666571e1fc62190

Reason: MultipleReasons

Message: Exposure to AzureRegistryImageMigrationUserProvisioned is unknown due to an evaluation failure: client-side throttling: only 18.061µs has elapsed since the last match call completed for this cluster condition backend; this cached cluster condition request has been queued for later execution

In Azure clusters with the user-provisioned registry storage, the in-cluster image registry component may struggle to complete the cluster update.

<https://issues.redhat.com/browse/IR-468>

Incoming HTTP requests to services exposed by Routes may fail while routers reload their configuration, especially when made with Apache HTTPClient versions before 5.0. The problem is more likely to occur in clusters with higher number of Routes and corresponding endpoints. <https://issues.redhat.com/browse/NE-1689>

Version: 4.14.21

Image: quay.io/openshift-release-dev/ocp-release@sha256:6e3fba19a1453e61f8846c6b0ad3abf41436a3550092cbfd364ad4ce194582b7

Reason: MultipleReasons

Message: Exposure to AzureRegistryImageMigrationUserProvisioned is unknown due to an evaluation failure: client-side throttling: only 33.991µs has elapsed since the last match call completed for this cluster condition backend; this cached cluster condition request has been queued for later execution

In Azure clusters with the user-provisioned registry storage, the in-cluster image registry component may struggle to complete the cluster update.

<https://issues.redhat.com/browse/IR-468>

Incoming HTTP requests to services exposed by Routes may fail while routers reload their configuration, especially when made with Apache HTTPClient versions before 5.0. The problem is more likely to occur in clusters with higher number of Routes and corresponding endpoints. <https://issues.redhat.com/browse/NE-1689>

The **oc adm upgrade** command queries the **ClusterVersion** resource for information about available updates and presents it in a human-readable format.

- One way to directly inspect the underlying availability data created by the CVO is by querying the **ClusterVersion** resource with the following command:

```
$ oc get clusterversion version -o json | jq '.status.availableUpdates'
```

Example output

```
[
  {
    "channels": [
      "candidate-4.11",
```



```

    "candidate-4.12",
    "fast-4.11",
    "fast-4.12"
  ],
  "image": "quay.io/openshift-release-dev/ocp-
release@sha256:400267c7f4e61c6bfa0a59571467e8bd85c9188e442cbd820cc8263809be377
5",
  "url": "https://access.redhat.com/errata/RHBA-2023:3213",
  "version": "4.11.41"
},
...
]

```

- A similar command can be used to check conditional updates:

```
$ oc get clusterversion version -o json | jq '.status.conditionalUpdates'
```

Example output

```

[
  {
    "conditions": [
      {
        "lastTransitionTime": "2023-05-30T16:28:59Z",
        "message": "The 4.11.36 release only resolves an installation issue
https://issues.redhat.com/browse/OCBUGS-11663 , which does not affect already running
clusters. 4.11.36 does not include fixes delivered in recent 4.11.z releases and therefore
upgrading from these versions would cause fixed bugs to reappear. Red Hat does not
recommend upgrading clusters to 4.11.36 version for this reason.
https://access.redhat.com/solutions/7007136",
        "reason": "PatchesOlderRelease",
        "status": "False",
        "type": "Recommended"
      }
    ],
    "release": {
      "channels": [...],
      "image": "quay.io/openshift-release-dev/ocp-
release@sha256:8c04176b771a62abd801fcda3e952633566c8b5ff177b93592e8e8d2d1f8471d
",
      "url": "https://access.redhat.com/errata/RHBA-2023:1733",
      "version": "4.11.36"
    },
    "risks": [...]
  },
  ...
]

```

1.2.1.2. Evaluation of update availability

The Cluster Version Operator (CVO) periodically queries the OpenShift Update Service (OSUS) for the most recent data about update possibilities. This data is based on the cluster's subscribed channel. The CVO then saves information about update recommendations into either the **availableUpdates** or **conditionalUpdates** field of its **ClusterVersion** resource.

The CVO periodically checks the conditional updates for update risks. These risks are conveyed through the data served by the OSUS, which contains information for each version about known issues that might affect a cluster updated to that version. Most risks are limited to clusters with specific characteristics, such as clusters with a certain size or clusters that are deployed in a particular cloud platform.

The CVO continuously evaluates its cluster characteristics against the conditional risk information for each conditional update. If the CVO finds that the cluster matches the criteria, the CVO stores this information in the **conditionalUpdates** field of its **ClusterVersion** resource. If the CVO finds that the cluster does not match the risks of an update, or that there are no risks associated with the update, it stores the target version in the **availableUpdates** field of its **ClusterVersion** resource.

The user interface, either the web console or the OpenShift CLI (**oc**), presents this information in sectioned headings to the administrator. Each known issue associated with the update path contains a link to further resources about the risk so that the administrator can make an informed decision about the update.

Additional resources

- [Update recommendation removals and Conditional Updates](#)

1.2.2. Release images

A release image is the delivery mechanism for a specific OpenShift Container Platform (OCP) version. It contains the release metadata, a Cluster Version Operator (CVO) binary matching the release version, every manifest needed to deploy individual OpenShift cluster Operators, and a list of SHA digest-versioned references to all container images that make up this OpenShift version.

You can inspect the content of a specific release image by running the following command:

```
$ oc adm release extract <release image>
```

Example output

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.12.6-x86_64
Extracted release payload from digest
sha256:800d1e39d145664975a3bb7cbc6e674fbf78e3c45b5dde9ff2c5a11a8690c87b created at
2023-03-01T12:46:29Z

$ ls
0000_03_authorization-openshift_01_rolebindingrestriction.crd.yaml
0000_03_config-operator_01_proxy.crd.yaml
0000_03_marketplace-operator_01_operatorhub.crd.yaml
0000_03_marketplace-operator_02_operatorhub.cr.yaml
0000_03_quota-openshift_01_clusterresourcequota.crd.yaml 1
...
0000_90_service-ca-operator_02_prometheusrolebinding.yaml 2
0000_90_service-ca-operator_03_servicemonitor.yaml
0000_99_machine-api-operator_00_tombstones.yaml
image-references 3
release-metadata
```

- 1 Manifest for **ClusterResourceQuota** CRD, to be applied on Runlevel 03

- 2 Manifest for **PrometheusRoleBinding** resource for the **service-ca-operator**, to be applied on Runlevel 90
- 3 List of SHA digest-versioned references to all required images

1.2.3. Update process workflow

The following steps represent a detailed workflow of the OpenShift Container Platform (OCP) update process:

1. The target version is stored in the **spec.desiredUpdate.version** field of the **ClusterVersion** resource, which may be managed through the web console or the CLI.
2. The Cluster Version Operator (CVO) detects that the **desiredUpdate** in the **ClusterVersion** resource differs from the current cluster version. Using graph data from the OpenShift Update Service, the CVO resolves the desired cluster version to a pull spec for the release image.
3. The CVO validates the integrity and authenticity of the release image. Red Hat publishes cryptographically-signed statements about published release images at predefined locations by using image SHA digests as unique and immutable release image identifiers. The CVO utilizes a list of built-in public keys to validate the presence and signatures of the statement matching the checked release image.
4. The CVO creates a job named **version-\$version-\$hash** in the **openshift-cluster-version** namespace. This job uses containers that are executing the release image, so the cluster downloads the image through the container runtime. The job then extracts the manifests and metadata from the release image to a shared volume that is accessible to the CVO.
5. The CVO validates the extracted manifests and metadata.
6. The CVO checks some preconditions to ensure that no problematic condition is detected in the cluster. Certain conditions can prevent updates from proceeding. These conditions are either determined by the CVO itself, or reported by individual cluster Operators that detect some details about the cluster that the Operator considers problematic for the update.
7. The CVO records the accepted release in **status.desired** and creates a **status.history** entry about the new update.
8. The CVO begins reconciling the manifests from the release image. Cluster Operators are updated in separate stages called Runlevels, and the CVO ensures that all Operators in a Runlevel finish updating before it proceeds to the next level.
9. Manifests for the CVO itself are applied early in the process. When the CVO deployment is applied, the current CVO pod stops, and a CVO pod that uses the new version starts. The new CVO proceeds to reconcile the remaining manifests.
10. The update proceeds until the entire control plane is updated to the new version. Individual cluster Operators might perform update tasks on their domain of the cluster, and while they do so, they report their state through the **Progressing=True** condition.
11. The Machine Config Operator (MCO) manifests are applied towards the end of the process. The updated MCO then begins updating the system configuration and operating system of every node. Each node might be drained, updated, and rebooted before it starts to accept workloads again.

The cluster reports as updated after the control plane update is finished, usually before all nodes are updated. After the update, the CVO maintains all cluster resources to match the state delivered in the release image.

1.2.4. Understanding how manifests are applied during an update

Some manifests supplied in a release image must be applied in a certain order because of the dependencies between them. For example, the **CustomResourceDefinition** resource must be created before the matching custom resources. Additionally, there is a logical order in which the individual cluster Operators must be updated to minimize disruption in the cluster. The Cluster Version Operator (CVO) implements this logical order through the concept of Runlevels.

These dependencies are encoded in the filenames of the manifests in the release image:

```
0000_<runlevel>_<component>_<manifest-name>.yaml
```

For example:

```
0000_03_config-operator_01_proxy.crd.yaml
```

The CVO internally builds a dependency graph for the manifests, where the CVO obeys the following rules:

- During an update, manifests at a lower Runlevel are applied before those at a higher Runlevel.
- Within one Runlevel, manifests for different components can be applied in parallel.
- Within one Runlevel, manifests for a single component are applied in lexicographic order.

The CVO then applies manifests following the generated dependency graph.






NOTE

For some resource types, the CVO monitors the resource after its manifest is applied, and considers it to be successfully updated only after the resource reaches a stable state. Achieving this state can take some time. This is especially true for **ClusterOperator** resources, while the CVO waits for a cluster Operator to update itself and then update its **ClusterOperator** status.

The CVO waits until all cluster Operators in the Runlevel meet the following conditions before it proceeds to the next Runlevel:

- The cluster Operators have an **Available=True** condition.
- The cluster Operators have a **Degraded=False** condition.
- The cluster Operators declare they have achieved the desired version in their ClusterOperator resource.

Some actions can take significant time to finish. The CVO waits for the actions to complete in order to ensure the subsequent Runlevels can proceed safely. Initially reconciling the new release's manifests is expected to take 60 to 120 minutes in total; see **Understanding OpenShift Container Platform update duration** for more information about factors that influence update duration.

 Completed
  In progress
  Waiting



341_OpenShift_0623

In the previous example diagram, the CVO is waiting until all work is completed at Runlevel 20. The CVO has applied all manifests to the Operators in the Runlevel, but the **kube-apiserver-operator ClusterOperator** performs some actions after its new version was deployed. The **kube-apiserver-operator ClusterOperator** declares this progress through the **Progressing=True** condition and by not declaring the new version as reconciled in its **status.versions**. The CVO waits until the ClusterOperator reports an acceptable status, and then it will start reconciling manifests at Runlevel 25.

Additional resources

- [Understanding OpenShift Container Platform update duration](#)

1.2.5. Understanding how the Machine Config Operator updates nodes

The Machine Config Operator (MCO) applies a new machine configuration to each control plane node and compute node. During the machine configuration update, control plane nodes and compute nodes are organized into their own machine config pools, where the pools of machines are updated in parallel. The **.spec.maxUnavailable** parameter, which has a default value of **1**, determines how many nodes in a machine config pool can simultaneously undergo the update process.

**WARNING**

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

When the machine configuration update process begins, the MCO checks the amount of currently unavailable nodes in a pool. If there are fewer unavailable nodes than the value of **.spec.maxUnavailable**, the MCO initiates the following sequence of actions on available nodes in the pool:

1. Cordon and drain the node

**NOTE**

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

2. Update the system configuration and operating system (OS) of the node
3. Reboot the node
4. Uncordon the node

A node undergoing this process is unavailable until it is uncordoned and workloads can be scheduled to it again. The MCO begins updating nodes until the number of unavailable nodes is equal to the value of **.spec.maxUnavailable**.

As a node completes its update and becomes available, the number of unavailable nodes in the machine config pool is once again fewer than **.spec.maxUnavailable**. If there are remaining nodes that need to be updated, the MCO initiates the update process on a node until the **.spec.maxUnavailable** limit is once again reached. This process repeats until each control plane node and compute node has been updated.

The following example workflow describes how this process might occur in a machine config pool with 5 nodes, where **.spec.maxUnavailable** is 3 and all nodes are initially available:

1. The MCO cordons nodes 1, 2, and 3, and begins to drain them.
2. Node 2 finishes draining, reboots, and becomes available again. The MCO cordons node 4 and begins draining it.
3. Node 1 finishes draining, reboots, and becomes available again. The MCO cordons node 5 and begins draining it.
4. Node 3 finishes draining, reboots, and becomes available again.
5. Node 5 finishes draining, reboots, and becomes available again.
6. Node 4 finishes draining, reboots, and becomes available again.

Because the update process for each node is independent of other nodes, some nodes in the example above finish their update out of the order in which they were cordoned by the MCO.

You can check the status of the machine configuration update by running the following command:

```
$ oc get mcp
```

Example output

NAME	CONFIG	UPDATED	UPDATING	DEGRADED	
MACHINECOUNT	READYMACHINECOUNT	UPDATEDMACHINECOUNT			
DEGRADEDMACHINECOUNT	AGE				
master	rendered-master-acd1358917e9f98cbdb599aea622d78b		True	False	False
3	3	0	22h		
worker	rendered-worker-1d871ac76e1951d32b2fe92369879826		False	True	False
1	1	0	22h		

Additional resources

- [Machine Config Overview](#)

1.3. UNDERSTANDING UPDATE CHANNELS AND RELEASES

Update channels are the mechanism by which users declare the OpenShift Container Platform minor version they intend to update their clusters to. They also allow users to choose the timing and level of support their updates will have through the **fast**, **stable**, **candidate**, and **eus** channel options. The Cluster Version Operator uses an update graph based on the channel declaration, along with other conditional information, to provide a list of recommended and conditional updates available to the cluster.

Update channels correspond to a minor version of OpenShift Container Platform. The version number in the channel represents the target minor version that the cluster will eventually be updated to, even if it is higher than the cluster's current minor version.

For instance, OpenShift Container Platform 4.10 update channels provide the following recommendations:

- Updates within 4.10.
- Updates within 4.9.
- Updates from 4.9 to 4.10, allowing all 4.9 clusters to eventually update to 4.10, even if they do not immediately meet the minimum z-stream version requirements.
- **eus-4.10** only: updates within 4.8.
- **eus-4.10** only: updates from 4.8 to 4.9 to 4.10, allowing all 4.8 clusters to eventually update to 4.10.

4.10 update channels do not recommend updates to 4.11 or later releases. This strategy ensures that administrators must 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 4.18 offers the following update channels:

- **stable-4.18**
- **eus-4.y** (only offered for EUS versions and meant to facilitate updates between EUS versions)
- **fast-4.18**
- **candidate-4.18**

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 updating 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.

1.3.1. Update channels

1.3.1.1. fast-4.18 channel

The **fast-4.18** channel is updated with new versions of OpenShift Container Platform 4.18 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.

1.3.1.2. stable-4.18 channel

While the **fast-4.18** channel contains releases as soon as their errata are published, releases are added to the **stable-4.18** 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, 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.

1.3.1.3. 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 a Control Plane Only 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.

1.3.1.4. candidate-4.18 channel

The **candidate-4.18** 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.

1.3.1.5. 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:

- 4.18.0
- 4.18.1
- 4.18.3
- 4.18.4

The service recommends only updates that have been tested and have no known serious regressions. For example, if your cluster is on 4.18.1 and OpenShift Container Platform suggests 4.18.4, then it is recommended to update from 4.18.1 to 4.18.4.



IMPORTANT

Do not rely on consecutive patch numbers. In this example, 4.18.2 is not and never was available in the channel, therefore updates to 4.18.2 are not recommended or supported.

1.3.1.6. Update recommendations 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 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 risks are confirmed, they are declared as Conditional Update risks for the relevant updates. 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 update paths are labeled as *updates with known issues*, and a reference link to the known issues is provided. The reference link helps the cluster admin decide if they want to accept the risk and continue to update their cluster.

When Red Hat chooses to declare Conditional Update risks, that action is taken in all relevant channels simultaneously. Declaration of a Conditional Update risk may happen either before or after the update has been promoted to supported channels.

1.3.1.7. Choosing the correct channel for your cluster

Choosing the appropriate channel involves two decisions.

First, select the minor version you want for your cluster update. 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 updates between versions many minors apart, channels that assist in planning updates beyond a single Control Plane Only 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-4.18** and **stable-4.18** are both fully supported and benefit equally from ongoing data analysis. The promotion delay before promoting a 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 factored into our rollout, so issues relevant to you can be fixed faster.

1.3.1.8. 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.

1.3.1.9. 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-4.18** channel to the **fast-4.18** channel.
- You can switch to the **candidate-4.18** channel at any time, but some releases for this channel might be unsupported.
- You can switch from the **candidate-4.18** channel to the **fast-4.18** channel if your current release is a general availability release.
- You can always switch from the **fast-4.18** channel to the **stable-4.18** channel. There is a possible delay of up to a day for the release to be promoted to **stable-4.18** if the current release was recently promoted.

Additional resources

- [Updating along a conditional upgrade path](#)
- [Choosing the correct channel for your cluster](#)

1.4. 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.

1.4.1. 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

**WARNING**

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane 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

1.4.2. 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

1.4.2.1. 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.

1.4.2.2. 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.
- Before you begin the update, you must ensure that all the nodes are available. Any unavailable nodes can significantly impact the update duration because the node unavailability affects the **maxUnavailable** and pod disruption budgets.

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

```
$ oc get node
```

Example Output

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-137-31.us-east-2.compute.internal	Ready,SchedulingDisabled	worker	12d	v1.23.5+3afdacb
ip-10-0-151-208.us-east-2.compute.internal	Ready	master	12d	v1.23.5+3afdacb
ip-10-0-176-138.us-east-2.compute.internal	Ready	master	12d	v1.23.5+3afdacb
ip-10-0-183-194.us-east-2.compute.internal	Ready	worker	12d	v1.23.5+3afdacb
ip-10-0-204-102.us-east-2.compute.internal	Ready	master	12d	v1.23.5+3afdacb
ip-10-0-207-224.us-east-2.compute.internal	Ready	worker	12d	v1.23.5+3afdacb

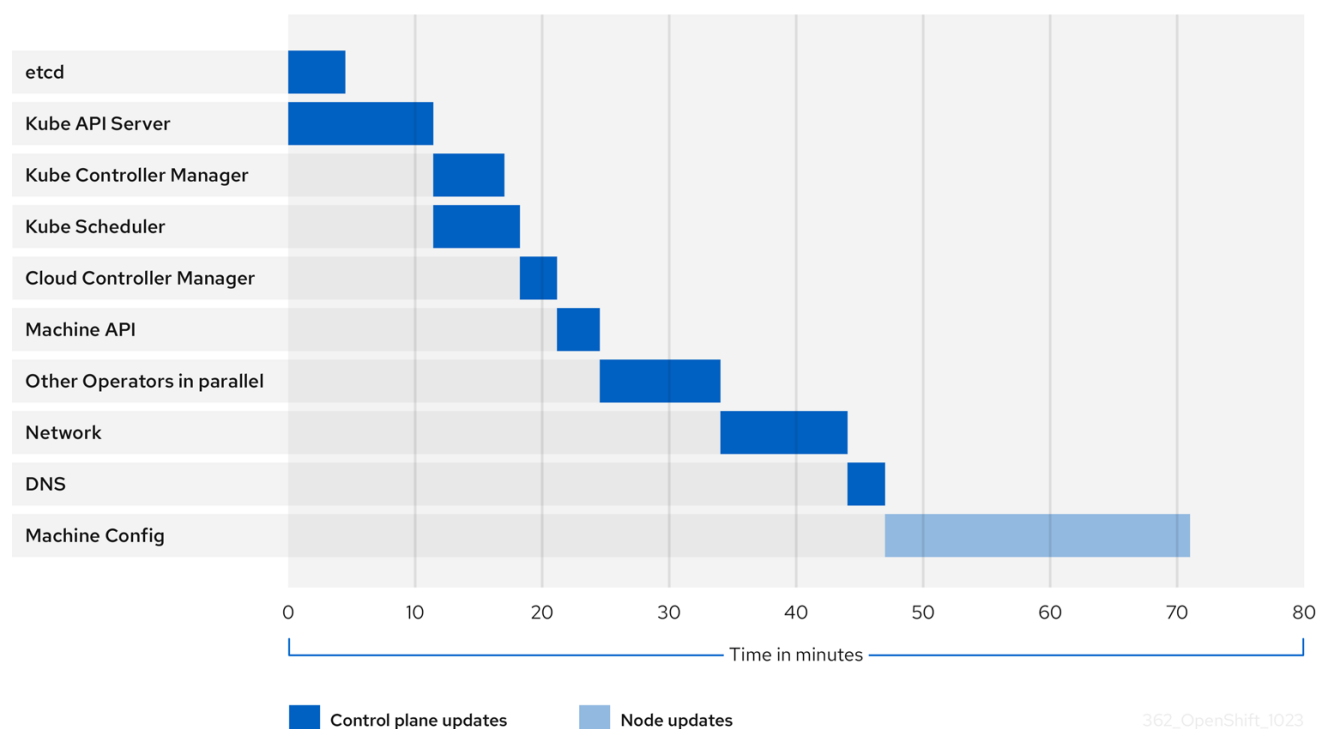
If the status of the node is **NotReady** or **SchedulingDisabled**, then the node is not available and this impacts the update duration.

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

Additional resources

- [Machine Config Overview](#)
- [Pod disruption budget](#)

1.4.2.3. Example update duration of cluster Operators



The previous diagram shows an example of the time that cluster Operators might take to update to their new versions. The example is based on a three-node AWS OVN cluster, which has a healthy compute **MachineConfigPool** and no workloads that take long to drain, updating from 4.13 to 4.14.



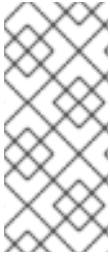
NOTE

- The specific update duration of a cluster and its Operators can vary based on several cluster characteristics, such as the target version, the amount of nodes, and the types of workloads scheduled to the nodes.
- Some Operators, such as the Cluster Version Operator, update themselves in a short amount of time. These Operators have either been omitted from the diagram or are included in the broader group of Operators labeled "Other Operators in parallel".

Each cluster Operator has characteristics that affect the time it takes to update itself. For instance, the Kube API Server Operator in this example took more than eleven minutes to update because **kube-apiserver** provides graceful termination support, meaning that existing, in-flight requests are allowed to complete gracefully. This might result in a longer shutdown of the **kube-apiserver**. In the case of this Operator, update speed is sacrificed to help prevent and limit disruptions to cluster functionality during an update.

Another characteristic that affects the update duration of an Operator is whether the Operator utilizes DaemonSets. The Network and DNS Operators utilize full-cluster DaemonSets, which can take time to roll out their version changes, and this is one of several reasons why these Operators might take longer to update themselves.

The update duration for some Operators is heavily dependent on characteristics of the cluster itself. For instance, the Machine Config Operator update applies machine configuration changes to each node in the cluster. A cluster with many nodes has a longer update duration for the Machine Config Operator compared to a cluster with fewer nodes.

**NOTE**

Each cluster Operator is assigned a stage during which it can be updated. Operators within the same stage can update simultaneously, and Operators in a given stage cannot begin updating until all previous stages have been completed. For more information, see "Understanding how manifests are applied during an update" in the "Additional resources" section.

Additional resources

- [Introduction to OpenShift updates](#)
- [Understanding how manifests are applied during an update](#)

1.4.3. 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:

Cluster update time = CVO target update payload deployment time + (# node update iterations x 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.

**WARNING**

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

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:

Cluster update time = 60 + (6 x 5) = 90 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.

Cluster update time = 60 + (3 x 5) = 75 minutes



IMPORTANT

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.

1.4.4. 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](#)

1.4.5. Additional resources

- [OpenShift Container Platform architecture](#)
- [OpenShift Container Platform updates](#)

CHAPTER 2. PREPARING TO UPDATE A CLUSTER

2.1. PREPARING TO UPDATE TO OPENSIFT CONTAINER PLATFORM 4.18

Learn more about administrative tasks that cluster admins must perform to successfully initialize an update, as well as optional guidelines for ensuring a successful update.

2.1.1. Kubernetes API removals

There are no Kubernetes API removals in OpenShift Container Platform 4.18.

2.1.2. Assessing the risk of conditional updates

A *conditional update* is an update target that is available but not recommended due to a known risk that applies to your cluster. The Cluster Version Operator (CVO) periodically queries the OpenShift Update Service (OSUS) for the most recent data about update recommendations, and some potential update targets might have risks associated with them.

The CVO evaluates the conditional risks, and if the risks are not applicable to the cluster, then the target version is available as a recommended update path for the cluster. If the risk is determined to be applicable, or if for some reason CVO cannot evaluate the risk, then the update target is available to the cluster as a conditional update.

When you encounter a conditional update while you are trying to update to a target version, you must assess the risk of updating your cluster to that version. Generally, if you do not have a specific need to update to that target version, it is best to wait for a recommended update path from Red Hat.

However, if you have a strong reason to update to that version, for example, if you need to fix an important CVE, then the benefit of fixing the CVE might outweigh the risk of the update being problematic for your cluster. You can complete the following tasks to determine whether you agree with the Red Hat assessment of the update risk:

- Complete extensive testing in a non-production environment to the extent that you are comfortable completing the update in your production environment.
- Follow the links provided in the conditional update description, investigate the bug, and determine if it is likely to cause issues for your cluster. If you need help understanding the risk, contact Red Hat Support.

Additional resources

- [Evaluation of update availability](#)

2.1.3. etcd backups before cluster updates

etcd backups record the state of your cluster and all of its resource objects. You can use backups to attempt restoring the state of a cluster in disaster scenarios where you cannot recover a cluster in its currently dysfunctional state.

In the context of updates, you can attempt an etcd restoration of the cluster if an update introduced catastrophic conditions that cannot be fixed without reverting to the previous cluster version. etcd restorations might be destructive and destabilizing to a running cluster, use them only as a last resort.

**WARNING**

Due to their high consequences, etcd restorations are not intended to be used as a rollback solution. Rolling your cluster back to a previous version is not supported. If your update is failing to complete, contact Red Hat support.

There are several factors that affect the viability of an etcd restoration. For more information, see "Backing up etcd data" and "Restoring to a previous cluster state".

Additional resources

- [Backing up etcd](#)
- [Restoring to a previous cluster state](#)

2.1.4. Best practices for cluster updates

OpenShift Container Platform provides a robust update experience that minimizes workload disruptions during an update. Updates will not begin unless the cluster is in an upgradeable state at the time of the update request.

This design enforces some key conditions before initiating an update, but there are a number of actions you can take to increase your chances of a successful cluster update.

2.1.4.1. Choose versions recommended by the OpenShift Update Service

The OpenShift Update Service (OSUS) provides update recommendations based on cluster characteristics such as the cluster's subscribed channel. The Cluster Version Operator saves these recommendations as either recommended or conditional updates. While it is possible to attempt an update to a version that is not recommended by OSUS, following a recommended update path protects users from encountering known issues or unintended consequences on the cluster.

Choose only update targets that are recommended by OSUS to ensure a successful update.

2.1.4.2. Address all critical alerts on the cluster

Critical alerts must always be addressed as soon as possible, but it is especially important to address these alerts and resolve any problems before initiating a cluster update. Failing to address critical alerts before beginning an update can cause problematic conditions for the cluster.

In the **Administrator** perspective of the web console, navigate to **Observe → Alerting** to find critical alerts.

2.1.4.3. Ensure that the cluster is in an Upgradable state

When one or more Operators have not reported their **Upgradeable** condition as **True** for more than an hour, the **ClusterNotUpgradeable** warning alert is triggered in the cluster. In most cases this alert does not block patch updates, but you cannot perform a minor version update until you resolve this alert and all Operators report **Upgradeable** as **True**.

For more information about the **Upgradeable** condition, see "Understanding cluster Operator condition types" in the additional resources section.

2.1.4.3.1. SDN support removal

OpenShift SDN network plugin was deprecated in versions 4.15 and 4.16. Starting with OpenShift Container Platform 4.17, the SDN network plugin is no longer supported and the content has been removed from the documentation.

If your OpenShift Container Platform cluster is still using the OpenShift SDN CNI, see [Migrating from the OpenShift SDN network plugin](#).



IMPORTANT

It is not possible to update a cluster to OpenShift Container Platform 4.17 and later if it is using the OpenShift SDN network plugin. You must migrate to the OVN-Kubernetes plugin before upgrading to OpenShift Container Platform 4.17 and later.

2.1.4.4. Ensure that enough spare nodes are available

A cluster should not be running with little to no spare node capacity, especially when initiating a cluster update. Nodes that are not running and available may limit a cluster's ability to perform an update with minimal disruption to cluster workloads.

Depending on the configured value of the cluster's **maxUnavailable** spec, the cluster might not be able to apply machine configuration changes to nodes if there is an unavailable node. Additionally, if compute nodes do not have enough spare capacity, workloads might not be able to temporarily shift to another node while the first node is taken offline for an update.

Make sure that you have enough available nodes in each worker pool, as well as enough spare capacity on your compute nodes, to increase the chance of successful node updates.



WARNING

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

2.1.4.5. Ensure that the cluster's PodDisruptionBudget is properly configured

You can use the **PodDisruptionBudget** object to define the minimum number or percentage of pod replicas that must be available at any given time. This configuration protects workloads from disruptions during maintenance tasks such as cluster updates.

However, it is possible to configure the **PodDisruptionBudget** for a given topology in a way that prevents nodes from being drained and updated during a cluster update.

When planning a cluster update, check the configuration of the **PodDisruptionBudget** object for the following factors:

- For highly available workloads, make sure there are replicas that can be temporarily taken offline without being prohibited by the **PodDisruptionBudget**.
- For workloads that are not highly available, make sure they are either not protected by a **PodDisruptionBudget** or have some alternative mechanism for draining these workloads eventually, such as periodic restart or guaranteed eventual termination.

Additional resources

- [Understanding cluster Operator condition types](#)

2.2. PREPARING TO UPDATE A CLUSTER 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 4.12 to 4.13, this status prevents you from updating 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 4.13.0 to 4.13.1, no permissions are added or changed, so the update is not blocked.

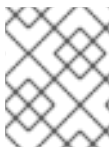
Before updating a cluster with manually maintained credentials, you must accommodate any new or changed credentials in the release image for the version of OpenShift Container Platform you are updating to.

2.2.1. Update requirements for clusters with manually maintained credentials

Before you update a cluster that uses manually maintained credentials with the Cloud Credential Operator (CCO), you must update the cloud provider resources for the new release.

If the cloud credential management for your cluster was configured using the CCO utility (**ccctl**), use the **ccctl** utility to update the resources. Clusters that were configured to use manual mode without the **ccctl** utility require manual updates for the resources.

After updating the cloud provider resources, you must update the **upgradeable-to** annotation for the cluster to indicate that it is ready to update.



NOTE

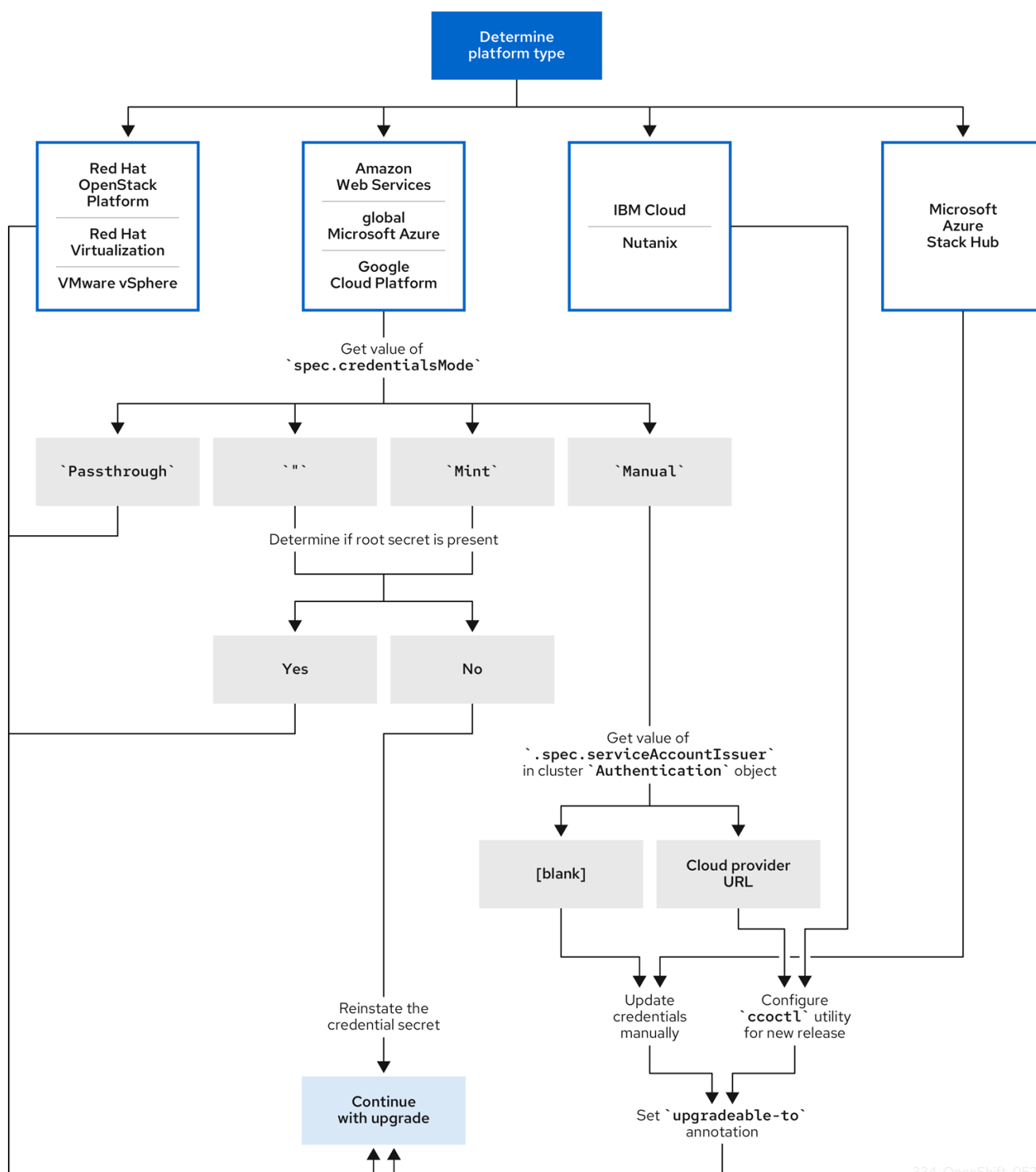
The process to update the cloud provider resources and the **upgradeable-to** annotation can only be completed by using command-line tools.

2.2.1.1. Cloud credential configuration options and update requirements by platform type

Some platforms only support using the CCO in one mode. For clusters that are installed on those platforms, the platform type determines the credentials update requirements.

For platforms that support using the CCO in multiple modes, you must determine which mode the cluster is configured to use and take the required actions for that configuration.

Figure 2.1. Credentials update requirements by platform type



334_OpenShift_0523

Red Hat OpenStack Platform (RHOSP) and VMware vSphere

These platforms do not support using the CCO in manual mode. Clusters on these platforms handle changes in cloud provider resources automatically and do not require an update to the **upgradeable-to** annotation.

Administrators of clusters on these platforms should skip the manually maintained credentials section of the update process.

IBM Cloud and Nutanix

Clusters installed on these platforms are configured using the **ccoctl** utility.

Administrators of clusters on these platforms must take the following actions:

1. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
2. Configure the **ccctl** utility for the new release and use it to update the cloud provider resources.
3. Indicate that the cluster is ready to update with the **upgradeable-to** annotation.

Microsoft Azure Stack Hub

These clusters use manual mode with long-term credentials and do not use the **ccctl** utility. Administrators of clusters on these platforms must take the following actions:

1. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
2. Manually update the cloud provider resources for the new release.
3. Indicate that the cluster is ready to update with the **upgradeable-to** annotation.

Amazon Web Services (AWS), global Microsoft Azure, and Google Cloud Platform (GCP)

Clusters installed on these platforms support multiple CCO modes.

The required update process depends on the mode that the cluster is configured to use. If you are not sure what mode the CCO is configured to use on your cluster, you can use the web console or the CLI to determine this information.

Additional resources

- [Determining the Cloud Credential Operator mode by using the web console](#)
- [Determining the Cloud Credential Operator mode by using the CLI](#)
- [Extracting and preparing credentials request resources](#)
- [About the Cloud Credential Operator](#)

2.2.1.2. Determining the Cloud Credential Operator mode by using the web console

You can determine what mode the Cloud Credential Operator (CCO) is configured to use by using the web console.



NOTE

Only Amazon Web Services (AWS), global Microsoft Azure, and Google Cloud Platform (GCP) clusters support multiple CCO modes.

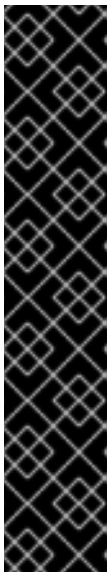
Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator permissions.

Procedure

1. Log in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
2. Navigate to **Administration** → **Cluster Settings**.

3. On the **Cluster Settings** page, select the **Configuration** tab.
4. Under **Configuration resource**, select **CloudCredential**.
5. On the **CloudCredential details** page, select the **YAML** tab.
6. In the YAML block, check the value of **spec.credentialsMode**. The following values are possible, though not all are supported on all platforms:
 - **"**: The CCO is operating in the default mode. In this configuration, the CCO operates in mint or passthrough mode, depending on the credentials provided during installation.
 - **Mint**: The CCO is operating in mint mode.
 - **Passthrough**: The CCO is operating in passthrough mode.
 - **Manual**: The CCO is operating in manual mode.



IMPORTANT

To determine the specific configuration of an AWS, GCP, or global Microsoft Azure cluster that has a **spec.credentialsMode** of **"**, **Mint**, or **Manual**, you must investigate further.

AWS and GCP clusters support using mint mode with the root secret deleted. If the cluster is specifically configured to use mint mode or uses mint mode by default, you must determine if the root secret is present on the cluster before updating.

An AWS, GCP, or global Microsoft Azure cluster that uses manual mode might be configured to create and manage cloud credentials from outside of the cluster with AWS STS, GCP Workload Identity, or Microsoft Entra Workload ID. You can determine whether your cluster uses this strategy by examining the cluster **Authentication** object.

7. AWS or GCP clusters that use mint mode only: To determine whether the cluster is operating without the root secret, navigate to **Workloads** → **Secrets** and look for the root secret for your cloud provider.



NOTE

Ensure that the **Project** dropdown is set to **All Projects**.

Platform	Secret name
AWS	aws-creds
GCP	gcp-credentials

- If you see one of these values, your cluster is using mint or passthrough mode with the root secret present.

- If you do not see these values, your cluster is using the CCO in mint mode with the root secret removed.
8. AWS, GCP, or global Microsoft Azure clusters that use manual mode only: To determine whether the cluster is configured to create and manage cloud credentials from outside of the cluster, you must check the cluster **Authentication** object YAML values.
- a. Navigate to **Administration** → **Cluster Settings**.
 - b. On the **Cluster Settings** page, select the **Configuration** tab.
 - c. Under **Configuration resource**, select **Authentication**.
 - d. On the **Authentication details** page, select the **YAML** tab.
 - e. In the YAML block, check the value of the **.spec.serviceAccountIssuer** parameter.
 - A value that contains a URL that is associated with your cloud provider indicates that the CCO is using manual mode with short-term credentials for components. These clusters are configured using the **ccctl** utility to create and manage cloud credentials from outside of the cluster.
 - An empty value ("") indicates that the cluster is using the CCO in manual mode but was not configured using the **ccctl** utility.

Next steps

- If you are updating a cluster that has the CCO operating in mint or passthrough mode and the root secret is present, you do not need to update any cloud provider resources and can continue to the next part of the update process.
- If your cluster is using the CCO in mint mode with the root secret removed, you must reinstate the credential secret with the administrator-level credential before continuing to the next part of the update process.
- If your cluster was configured using the CCO utility (**ccctl**), you must take the following actions:
 - a. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
 - b. Configure the **ccctl** utility for the new release and use it to update the cloud provider resources.
 - c. Update the **upgradeable-to** annotation to indicate that the cluster is ready to update.
- If your cluster is using the CCO in manual mode but was not configured using the **ccctl** utility, you must take the following actions:
 - a. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
 - b. Manually update the cloud provider resources for the new release.
 - c. Update the **upgradeable-to** annotation to indicate that the cluster is ready to update.

Additional resources

- [Extracting and preparing credentials request resources](#)

2.2.1.3. Determining the Cloud Credential Operator mode by using the CLI

You can determine what mode the Cloud Credential Operator (CCO) is configured to use by using the CLI.



NOTE

Only Amazon Web Services (AWS), global Microsoft Azure, and Google Cloud Platform (GCP) clusters support multiple CCO modes.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator permissions.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Log in to **oc** on the cluster as a user with the **cluster-admin** role.
2. To determine the mode that the CCO is configured to use, enter the following command:

```
$ oc get cloudcredentials cluster \
  -o=jsonpath={.spec.credentialsMode}
```

The following output values are possible, though not all are supported on all platforms:

- **"**: The CCO is operating in the default mode. In this configuration, the CCO operates in mint or passthrough mode, depending on the credentials provided during installation.
- **Mint**: The CCO is operating in mint mode.
- **Passthrough**: The CCO is operating in passthrough mode.
- **Manual**: The CCO is operating in manual mode.



IMPORTANT

To determine the specific configuration of an AWS, GCP, or global Microsoft Azure cluster that has a **spec.credentialsMode** of **"**, **Mint**, or **Manual**, you must investigate further.

AWS and GCP clusters support using mint mode with the root secret deleted. If the cluster is specifically configured to use mint mode or uses mint mode by default, you must determine if the root secret is present on the cluster before updating.

An AWS, GCP, or global Microsoft Azure cluster that uses manual mode might be configured to create and manage cloud credentials from outside of the cluster with AWS STS, GCP Workload Identity, or Microsoft Entra Workload ID. You can determine whether your cluster uses this strategy by examining the cluster **Authentication** object.

3. AWS or GCP clusters that use mint mode only: To determine whether the cluster is operating without the root secret, run the following command:

```
$ oc get secret <secret_name> \
-n=kube-system
```

where **<secret_name>** is **aws-creds** for AWS or **gcp-credentials** for GCP.

If the root secret is present, the output of this command returns information about the secret. An error indicates that the root secret is not present on the cluster.

4. AWS, GCP, or global Microsoft Azure clusters that use manual mode only: To determine whether the cluster is configured to create and manage cloud credentials from outside of the cluster, run the following command:

```
$ oc get authentication cluster \
-o jsonpath \
--template='{ .spec.serviceAccountIssuer }'
```

This command displays the value of the **.spec.serviceAccountIssuer** parameter in the cluster **Authentication** object.

- An output of a URL that is associated with your cloud provider indicates that the CCO is using manual mode with short-term credentials for components. These clusters are configured using the **ccctl** utility to create and manage cloud credentials from outside of the cluster.
- An empty output indicates that the cluster is using the CCO in manual mode but was not configured using the **ccctl** utility.

Next steps

- If you are updating a cluster that has the CCO operating in mint or passthrough mode and the root secret is present, you do not need to update any cloud provider resources and can continue to the next part of the update process.
- If your cluster is using the CCO in mint mode with the root secret removed, you must reinstate the credential secret with the administrator-level credential before continuing to the next part of the update process.
- If your cluster was configured using the CCO utility (**ccctl**), you must take the following actions:
 - a. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
 - b. Configure the **ccctl** utility for the new release and use it to update the cloud provider resources.
 - c. Update the **upgradeable-to** annotation to indicate that the cluster is ready to update.
- If your cluster is using the CCO in manual mode but was not configured using the **ccctl** utility, you must take the following actions:
 - a. Extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.
 - b. Manually update the cloud provider resources for the new release.

- c. Update the **upgradeable-to** annotation to indicate that the cluster is ready to update.

Additional resources

- [Extracting and preparing credentials request resources](#)

2.2.2. Extracting and preparing credentials request resources

Before updating a cluster that uses the Cloud Credential Operator (CCO) in manual mode, you must extract and prepare the **CredentialsRequest** custom resources (CRs) for the new release.

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.

Procedure

1. Obtain the pull spec for the update that you want to apply by running the following command:

```
$ oc adm upgrade
```

The output of this command includes pull specs for the available updates similar to the following:

Partial example output

```
...
Recommended updates:

VERSION IMAGE
4.18.0 quay.io/openshift-release-dev/ocp-
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d01803
2
...
```

2. Set a **\$RELEASE_IMAGE** variable with the release image that you want to use by running the following command:

```
$ RELEASE_IMAGE=<update_pull_spec>
```

where **<update_pull_spec>** is the pull spec for the release image that you want to use. For example:

```
quay.io/openshift-release-dev/ocp-
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d01803
2
```

3. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
```

```
--from=$RELEASE_IMAGE \
--credentials-requests \
--included \ 1
--to=<path_to_directory_for_credentials_requests> 2
```

- 1 The **--included** parameter includes only the manifests that your specific cluster configuration requires for the target release.
- 2 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

4. For each **CredentialsRequest** CR in the release image, ensure that a namespace that matches the text in the **spec.secretRef.namespace** field exists in the cluster. This field is where the generated secrets that hold the credentials configuration are stored.

Sample AWS CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: cloud-credential-operator-iam-ro
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - effect: Allow
        action:
          - iam:GetUser
          - iam:GetUserPolicy
          - iam:ListAccessKeys
        resource: "*"
  secretRef:
    name: cloud-credential-operator-iam-ro-creds
    namespace: openshift-cloud-credential-operator 1
```

- 1 This field indicates the namespace which must exist to hold the generated secret.

The **CredentialsRequest** CRs for other platforms have a similar format with different platform-specific values.

5. For any **CredentialsRequest** CR for which the cluster does not already have a namespace with the name specified in **spec.secretRef.namespace**, create the namespace by running the following command:

```
$ oc create namespace <component_namespace>
```

Next steps

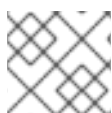
- If the cloud credential management for your cluster was configured using the CCO utility (**ccctl**), configure the **ccctl** utility for a cluster update and use it to update your cloud provider resources.
- If your cluster was not configured with the **ccctl** utility, manually update your cloud provider resources.

Additional resources

- [Configuring the Cloud Credential Operator utility for a cluster update](#)
- [Manually updating cloud provider resources](#)

2.2.3. Configuring the Cloud Credential Operator utility for a cluster update

To upgrade a cluster that uses the Cloud Credential Operator (CCO) in manual mode to create and manage cloud credentials from outside of the cluster, extract and prepare the CCO utility (**ccctl**) binary.



NOTE

The **ccctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- Your cluster was configured using the **ccctl** utility to create and manage cloud credentials from outside of the cluster.
- You have extracted the **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image and ensured that a namespace that matches the text in the **spec.secretRef.namespace** field exists in the cluster.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(oc get clusterversion -o jsonpath={..desired.image})
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'  
$RELEASE_IMAGE -a ~/.pull-secret)
```



NOTE

Ensure that the architecture of the **\$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \
  -a ~/.pull-secret
```

- 1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl.<rhel_version>
```

Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl.rhel9
```

Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws      Manage credentials objects for AWS cloud
  azure    Manage credentials objects for Azure
  gcp      Manage credentials objects for Google cloud
  help     Help about any command
  ibmcloud Manage credentials objects for {ibm-cloud-title}
  nutanix  Manage credentials objects for Nutanix

Flags:
  -h, --help  help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

2.2.4. Updating cloud provider resources with the Cloud Credential Operator utility

The process for upgrading an OpenShift Container Platform cluster that was configured using the CCO utility (**ccoctl**) is similar to creating the cloud provider resources during installation.



NOTE

On AWS clusters, some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

Prerequisites

- You have extracted the **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image and ensured that a namespace that matches the text in the **spec.secretRef.namespace** field exists in the cluster.
- You have extracted and configured the **ccoctl** binary from the release image.

Procedure

1. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the command for your cloud provider. The following commands process **CredentialsRequest** objects:

Example 2.1. Amazon Web Services (AWS)

```
$ ccoctl aws create-all \ 1
--name=<name> \ 2
--region=<aws_region> \ 3
--credentials-requests-dir=<path_to_credentials_requests_directory> \ 4
--output-dir=<path_to_ccoctl_output_dir> \ 5
--create-private-s3-bucket 6
```

- 1 To create the AWS resources individually, use the "Creating AWS resources individually" procedure in the "Installing a cluster on AWS with customizations" content. This option might be useful if you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization.
- 2 Specify the name used to tag any cloud resources that are created for tracking.
- 3 Specify the AWS region in which cloud resources will be created.
- 4 Specify the directory containing the files for the component **CredentialsRequest** objects.
- 5 Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
- 6 Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

Example 2.2. Google Cloud Platform (GCP)

```
$ ccoctl gcp create-all \
  --name=<name> \ ❶
  --region=<gcp_region> \ ❷
  --project=<gcp_project_id> \ ❸
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ ❹
  --output-dir=<path_to_ccoctl_output_dir> ❺
```

- ❶ Specify the user-defined name for all created GCP resources used for tracking.
- ❷ Specify the GCP region in which cloud resources will be created.
- ❸ Specify the GCP project ID in which cloud resources will be created.
- ❹ Specify the directory containing the files of **CredentialsRequest** manifests to create GCP service accounts.
- ❺ Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.

Example 2.3. IBM Cloud

```
$ ccoctl ibmcloud create-service-id \
  --credentials-requests-dir=<path_to_credential_requests_directory> \ ❶
  --name=<cluster_name> \ ❷
  --output-dir=<installation_directory> \ ❸
  --resource-group-name=<resource_group_name> ❹
```

- ❶ Specify the directory containing the files for the component **CredentialsRequest** objects.
- ❷ Specify the name of the OpenShift Container Platform cluster.
- ❸ Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
- ❹ Optional: Specify the name of the resource group used for scoping the access policies.

Example 2.4. Microsoft Azure

```
$ ccoctl azure create-managed-identities \
  --name <azure_infra_name> \ ❶
  --output-dir ./output_dir \
  --region <azure_region> \ ❷
  --subscription-id <azure_subscription_id> \ ❸
  --credentials-requests-dir <path_to_directory_for_credentials_requests> \
  --issuer-url "${OIDC_ISSUER_URL}" \ ❹
  --dnszone-resource-group-name <azure_dns_zone_resourcegroup_name> \ ❺
  --installation-resource-group-name "${AZURE_INSTALL_RG}" ❻
```


- 1 The value of the **name** parameter is used to create an Azure resource group. To use an existing Azure resource group instead of creating a new one, specify the **--oidc-resource-group-name** argument with the existing group name as its value.
- 2 Specify the region of the existing cluster.
- 3 Specify the subscription ID of the existing cluster.
- 4 Specify the OIDC issuer URL from the existing cluster. You can obtain this value by running the following command:

```
$ oc get authentication cluster \
  -o jsonpath \
  --template='{ .spec.serviceAccountIssuer }'
```

- 5 Specify the name of the resource group that contains the DNS zone.
- 6 Specify the Azure resource group name. You can obtain this value by running the following command:

```
$ oc get infrastructure cluster \
  -o jsonpath \
  --template '{ .status.platformStatus.azure.resourceGroupName }'
```

Example 2.5. Nutanix

```
$ ccoctl nutanix create-shared-secrets \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 1
  --output-dir=<ccoctl_output_dir> \ 2
  --credentials-source-filepath=<path_to_credentials_file> \ 3
```

- 1 Specify the path to the directory that contains the files for the component **CredentialsRequests** objects.
- 2 Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
- 3 Optional: Specify the directory that contains the credentials data YAML file. By default, **ccoctl** expects this file to be in **<home_directory>/nutanix/credentials**.

For each **CredentialsRequest** object, **ccoctl** creates the required provider resources and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

2. Apply the secrets to your cluster by running the following command:

```
$ ls <path_to_ccoctl_output_dir>/manifests/*-credentials.yaml | xargs -l{} oc apply -f {}
```

Verification

You can verify that the required provider resources and permissions policies are created by querying the cloud provider. For more information, refer to your cloud provider documentation on listing roles or service accounts.

Next steps

- Update the **upgradeable-to** annotation to indicate that the cluster is ready to upgrade.

Additional resources

- [Indicating that the cluster is ready to upgrade](#)

2.2.5. Manually updating cloud provider resources

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

Prerequisites

- You have extracted the **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image and ensured that a namespace that matches the text in the **spec.secretRef.namespace** field exists in the cluster.

Procedure

1. Create YAML files with secrets for any **CredentialsRequest** custom resources that the new release image adds. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

Example 2.6. Sample AWS YAML files

Sample AWS CredentialsRequest object with secrets

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - effect: Allow
        action:
          - s3:CreateBucket
          - s3>DeleteBucket
        resource: "*"
    ...
  secretRef:
```

```

name: <component_secret>
namespace: <component_namespace>
...

```

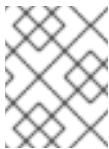
Sample AWS Secret object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>

```

Example 2.7. Sample Azure YAML files



NOTE

Global Azure and Azure Stack Hub use the same **CredentialsRequest** object and secret formats.

Sample Azure **CredentialsRequest** object with secrets

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
      - role: Contributor
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
...

```

Sample Azure Secret object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  azure_subscription_id: <base64_encoded_azure_subscription_id>
  azure_client_id: <base64_encoded_azure_client_id>

```

```

azure_client_secret: <base64_encoded_azure_client_secret>
azure_tenant_id: <base64_encoded_azure_tenant_id>
azure_resource_prefix: <base64_encoded_azure_resource_prefix>
azure_resourcegroup: <base64_encoded_azure_resourcegroup>
azure_region: <base64_encoded_azure_region>

```

Example 2.8. Sample GCP YAML files

Sample GCP CredentialsRequest object with secrets

```

apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
      - roles/iam.securityReviewer
      - roles/iam.roleViewer
    skipServiceCheck: true
    ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...

```

Sample GCP Secret object

```

apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  service_account.json: <base64_encoded_gcp_service_account_file>

```

2. If the **CredentialsRequest** custom resources for any existing credentials that are stored in secrets have changed permissions requirements, update the permissions as required.

Next steps

- Update the **upgradeable-to** annotation to indicate that the cluster is ready to upgrade.

Additional resources

- [Manually creating long-term credentials for AWS](#)

- [Manually creating long-term credentials for Azure](#)
- [Manually creating long-term credentials for Azure Stack Hub](#)
- [Manually creating long-term credentials for GCP](#)
- [Indicating that the cluster is ready to upgrade](#)

2.2.6. Indicating that the cluster is ready to upgrade

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

Prerequisites

- For the release image that you are upgrading to, you have processed any new credentials manually or by using the Cloud Credential Operator utility (**ccctl**).
- You have installed the OpenShift CLI (**oc**).

Procedure

1. Log in to **oc** on the cluster as a user with the **cluster-admin** role.
2. Edit the **CloudCredential** resource to add an **upgradeable-to** annotation within the **metadata** field by running the following command:

```
$ oc edit cloudcredential cluster
```

Text to add

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

Where **<version_number>** is the version that you are upgrading to, in the format **x.y.z**. For example, use **4.12.2** for OpenShift Container Platform 4.12.2.

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

2.3. PREFLIGHT VALIDATION FOR KERNEL MODULE MANAGEMENT (KMM) MODULES

Before performing an upgrade on the cluster with applied KMM modules, you must verify that kernel modules installed using KMM are able to be installed on the nodes after the cluster upgrade and possible kernel upgrade. Preflight attempts to validate every **Module** loaded in the cluster, in parallel. Preflight does not wait for validation of one **Module** to complete before starting validation of another **Module**.

2.3.1. Validation kickoff

Preflight validation is triggered by creating a **PreflightValidationOCP** resource in the cluster. This spec contains two fields:

releaseImage

Mandatory field that provides the name of the release image for the OpenShift Container Platform version the cluster is upgraded to.

pushBuiltImage

If **true**, then the images created during the Build and Sign validation are pushed to their repositories. This field is **false** by default.

2.3.2. Validation lifecycle

Preflight validation attempts to validate every module loaded in the cluster. Preflight stops running validation on a **Module** resource after the validation is successful. If module validation fails, you can change the module definitions and Preflight tries to validate the module again in the next loop.

If you want to run Preflight validation for an additional kernel, then you should create another **PreflightValidationOCP** resource for that kernel. After all the modules have been validated, it is recommended to delete the **PreflightValidationOCP** resource.

2.3.3. Validation status

A **PreflightValidationOCP** resource reports the status and progress of each module in the cluster that it attempts or has attempted to validate in its **.status.modules** list. Elements of that list contain the following fields:

lastTransitionTime

The last time the **Module** resource status transitioned from one status to another. This should be when the underlying status has changed. If that is not known, then using the time when the API field changed is acceptable.

name

The name of the **Module** resource.

namespace

The namespace of the **Module** resource.

statusReason

Verbal explanation regarding the status.

verificationStage

Describes the validation stage being executed:

- **image**: Image existence verification

- **build**: Build process verification
- **sign**: Sign process verification

verificationStatus

The status of the Module verification:

- **true**: Verified
- **false**: Verification failed
- **error**: Error during the verification process
- **unknown**: Verification has not started

2.3.4. Preflight validation stages per Module

Preflight runs the following validations on every KMM Module present in the cluster:

1. Image validation stage
2. Build validation stage
3. Sign validation stage

2.3.4.1. Image validation stage

Image validation is always the first stage of the preflight validation to be executed. If image validation is successful, no other validations are run on that specific module.

Image validation consists of two stages:

1. Image existence and accessibility. The code tries to access the image defined for the upgraded kernel in the module and get its manifests.
2. Verify the presence of the kernel module defined in the **Module** in the correct path for future **modprobe** execution. If this validation is successful, it probably means that the kernel module was compiled with the correct Linux headers. The correct path is `<dirname>/lib/modules/<upgraded_kernel>/`.

2.3.4.2. Build validation stage

Build validation is executed only when image validation has failed and there is a **build** section in the **Module** that is relevant for the upgraded kernel. Build validation attempts to run the build job and validate that it finishes successfully.



NOTE

You must specify the kernel version when running **depmod**, as shown here:

```
$ RUN depmod -b /opt ${KERNEL_VERSION}
```

If the **PushBuiltImage** flag is defined in the **PreflightValidationOCP** custom resource (CR), it also tries to push the resulting image into its repository. The resulting image name is taken from the definition of the **containerImage** field of the **Module** CR.



NOTE

If the **sign** section is defined for the upgraded kernel, then the resulting image will not be the **containerImage** field of the **Module** CR, but a temporary image name, because the resulting image should be the product of Sign flow.

2.3.4.3. Sign validation stage

Sign validation is executed only when image validation has failed. There is a **sign** section in the **Module** resource that is relevant for the upgrade kernel, and build validation finishes successfully in case there was a **build** section in the **Module** relevant for the upgraded kernel. Sign validation attempts to run the sign job and validate that it finishes successfully.

If the **PushBuiltImage** flag is defined in the **PreflightValidationOCP** CR, sign validation also tries to push the resulting image to its registry. The resulting image is always the image defined in the **ContainerImage** field of the **Module**. The input image is either the output of the Build stage, or an image defined in the **UnsignedImage** field.



NOTE

If a **build** section exists, the **sign** section input image is the **build** section's output image. Therefore, in order for the input image to be available for the **sign** section, the **PushBuiltImage** flag must be defined in the **PreflightValidationOCP** CR.

2.3.5. Example PreflightValidationOCP resource

This section shows an example of the **PreflightValidationOCP** resource in the YAML format.

The example verifies all of the currently present modules against the upcoming kernel version included in the OpenShift Container Platform release 4.11.18, which the following release image points to:

```
quay.io/openshift-release-dev/ocp-
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
```

Because **.spec.pushBuiltImage** is set to **true**, KMM pushes the resulting images of Build/Sign in to the defined repositories.

```
apiVersion: kmm.sigs.x-k8s.io/v1beta2
kind: PreflightValidationOCP
metadata:
  name: preflight
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
  pushBuiltImage: true
```

2.4. PREPARING TO UPDATE FROM OPENSIFT CONTAINER PLATFORM 4.18 TO A NEWER VERSION

Before you update from OpenShift Container Platform 4.18 to a newer version, learn about some of the specific concerns around Red Hat Enterprise Linux (RHEL) compute machines and Gateway API networking resources.

2.4.1. Migrating workloads off of package-based RHEL worker nodes

With the introduction of OpenShift Container Platform 4.19, package-based RHEL worker nodes are no longer supported. If you try to update your cluster while those nodes are up and running, the update will fail.

You can reschedule pods running on RHEL compute nodes to run on your RHCOS nodes instead by using node selectors.

For example, the following **Node** object has a label for its operating system information, in this case RHCOS:

Sample Node object with RHCOS label

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-131-14.ec2.internal
  selfLink: /api/v1/nodes/ip-10-0-131-14.ec2.internal
  uid: 7bc2580a-8b8e-11e9-8e01-021ab4174c74
  resourceVersion: '478704'
  creationTimestamp: '2019-06-10T14:46:08Z'
  labels:
    kubernetes.io/os: linux
    failure-domain.beta.kubernetes.io/zone: us-east-1a
    node.openshift.io/os_version: '4.18'
    node-role.kubernetes.io/worker: ""
    failure-domain.beta.kubernetes.io/region: us-east-1
    node.openshift.io/os_id: rhcos ❶
    beta.kubernetes.io/instance-type: m4.large
    kubernetes.io/hostname: ip-10-0-131-14
    beta.kubernetes.io/arch: amd64
#...
```

- ❶ The label identifying the operating system that runs on the node, to match the pod node selector.

Any pods that you want to schedule to new RHCOS nodes must contain a matching label in its **nodeSelector** field. The following procedure describes how to add the label.

Procedure

1. Deschedule the RHEL node currently running your existing pods by entering the following command:

```
$ oc adm cordon <rhel-node>
```

2. Add an **rhcos** node selector to a pod:
 - To add the node selector to existing and future pods, add the node selector to the controller object for the pods by entering the following command:

Example Deployment object with rhcos label

```
$ oc patch dc <my-app> -p '{"spec":{"template":{"spec":{"nodeSelector":{"node.openshift.io/os_id":"rhcos"}}}}}'
```

Any existing pods under your **Deployment** controlling object will be re-created on your RHCOS nodes.

- To add the node selector to a specific, new pod, add the selector to the **Pod** object directly:

Example Pod object with rhcos label

```
apiVersion: v1
kind: Pod
metadata:
  name: <my-app>
#...
spec:
  nodeSelector:
    node.openshift.io/os_id: rhcos
#...
```

The new pod will be created on RHCOS nodes, assuming the pod also has a controlling object.

2.4.2. Identifying and removing RHEL worker nodes

With the introduction of OpenShift Container Platform 4.19, package-based RHEL worker nodes are no longer supported. The following procedure describes how to identify RHEL nodes for cluster removal on bare-metal installations. You must complete the following steps to successfully update your cluster.

Procedure

1. Identify nodes in your cluster that are running RHEL by entering the following command:

```
$ oc get -l node.openshift.io/os_id=rhel
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
rhel-node1.example.com	Ready	worker	7h	v1.31.7
rhel-node2.example.com	Ready	worker	7h	v1.31.7
rhel-node3.example.com	Ready	worker	7h	v1.31.7

2. [Continue with the node removal process](#). RHEL nodes are not managed by the Machine API and have no compute machine sets associated with them. You must unschedule and drain the node before you manually delete it from the cluster.

For more information on this process, see [How to remove a worker node from Red Hat OpenShift Container Platform 4 UPI](#).

2.4.3. Provisioning new RHCOS worker nodes

If you need additional compute nodes for your workloads, you can provision new ones either before or after you update your cluster. For more information, see the following [machine management](#) documentation:

- [Manually scaling a compute machine set](#)
- [Applying autoscaling to an OpenShift Container Platform cluster](#)
- [Adding compute machines to clusters with user-provisioned infrastructure manually](#)

For installer-provisioned infrastructure installations, automatic scaling adds RHCOS nodes by default. For user-provisioned infrastructure installations on bare metal platforms, you can manually [add RHCOS compute nodes to your cluster](#).

2.4.4. Preparing for Gateway API management succession by the Ingress Operator

Starting in OpenShift Container Platform 4.19, the Ingress Operator manages the lifecycle of any Gateway API custom resource definitions (CRDs). Updating from a version before 4.19 of OpenShift Container Platform where this management was not present requires you to replace or remove any Gateway API CRDs that already exist in the cluster so that they conform to the specific OpenShift Container Platform specification required by the Ingress Operator. OpenShift Container Platform version 4.19 requires Gateway API Standard version 1.2.0 CRDs.



WARNING

Updating or deleting Gateway API resources can result in downtime and loss of service or data. Be sure you understand how this will affect your cluster before performing the steps in this procedure. If necessary, back up any Gateway API objects in YAML format in order to restore it later.

Prerequisites

- You have installed the OpenShift CLI (**oc**).
- You have access to an OpenShift Container Platform account with cluster administrator access.
- Optional: You have backed up any necessary Gateway API objects.

Procedure

1. List all the Gateway API CRDs that you need to remove by running the following command:

```
$ oc get crd | grep "gateway.networking"
```

Example output

```
gatewayclasses.gateway.networking.k8s.io
gateways.gateway.networking.k8s.io
grpcroutes.gateway.networking.k8s.io
```

```
httproutes.gateway.networking.k8s.io  
referencegrants.gateway.networking.k8s.io
```

2. Delete the Gateway API CRDs from the previous step by running the following command:

```
$ oc delete crd gatewayclasses.networking.k8s.io && \  
oc delete crd gateways.networking.k8s.io && \  
oc delete crd grpcroutes.gateway.networking.k8s.io && \  
oc delete crd httproutes.gateway.networking.k8s.io && \  
oc delete crd referencegrants.gateway.networking.k8s.io
```



IMPORTANT

Any controller that was previously managing the lifecycle of the Gateway API CRDs will fail to operate properly. Attempting to force its use in conjunction with the Ingress Operator to manage Gateway API CRDs might prevent the cluster update from succeeding.

3. Get the supported Gateway API CRDs by running the following command:

```
$ oc apply -f https://github.com/kubernetes-sigs/gateway-  
api/releases/download/v1.2.0/standard-install.yaml
```

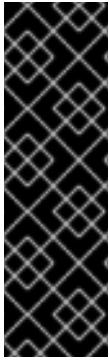
CHAPTER 3. PERFORMING A CLUSTER UPDATE

3.1. UPDATING A CLUSTER USING THE CLI

You can perform minor version and patch updates on an OpenShift Container Platform cluster by using the OpenShift CLI (**oc**).

3.1.1. 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.
- Have a recent [Container Storage Interface \(CSI\) volume snapshot](#) in case you need to restore persistent volumes due to a pod failure.
- Your RHEL7 workers are replaced with RHEL8 or RHCOS workers. 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.
- You have updated all Operators previously installed through Operator Lifecycle Manager (OLM) to a version that is compatible with your target release. 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 on how to check compatibility and, if necessary, update the installed Operators.
- 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, update the cloud provider resources for the new release. For more information, including how to determine if this is a requirement for your cluster, see [Preparing to update a cluster with manually maintained credentials](#).
- 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 updated. 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 update 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](#)

3.1.2. Pausing a MachineHealthCheck resource

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



NOTE

Some **MachineHealthCheck** resources might not need to be paused. If your **MachineHealthCheck** resource relies on unrecoverable conditions, pausing that MHC is unnecessary.

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:

```
$ 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

```

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-
```

3.1.3. 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 update 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 [About the Machine Config Operator](#).

3.1.4. Updating a cluster by using the CLI

You can use the OpenShift CLI (**oc**) to review and request cluster updates.

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.
- Pause all **MachineHealthCheck** resources.

Procedure

1. 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 4.13.10
Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.13 (available channels: candidate-4.13, candidate-4.14, fast-4.13, stable-4.13)
Recommended updates:
VERSION    IMAGE
4.13.14    quay.io/openshift-release-dev/ocp-release@sha256:406fcc160c097f61080412afcf7fd65284ac8741ac7ad5b480e304aba73674b
4.13.13    quay.io/openshift-release-dev/ocp-release@sha256:d62495768e335c79a215ba56771ff5ae97e3cbb2bf49ed8fb3f6cefabc0f17
4.13.12    quay.io/openshift-release-dev/ocp-release@sha256:73946971c03b43a0dc6f7b0946b26a177c2f3c9d37105441315b4e3359373a55
4.13.11    quay.io/openshift-release-dev/ocp-release@sha256:e1c2377fdae1d063aaddc753b99acf25972b6997ab9a0b7e80cfef627b9ef3dd
```




NOTE

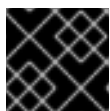
- If there are no recommended updates, updates that have known issues might still be available. See *Updating along a conditional update path* for more information.
- For details and information on how to perform a **Control Plane Only** update, please refer to the *Preparing to perform a Control Plane Only update* page, listed in the Additional resources section.

2. Based on your organization requirements, set the appropriate update channel. For example, you can set your channel to **stable-4.13** or **fast-4.13**. For more information about channels, refer to *Understanding update channels and releases* listed in the Additional resources section.

```
$ oc adm upgrade channel <channel>
```

For example, to set the channel to **stable-4.18**:

```
$ oc adm upgrade channel stable-4.18
```



IMPORTANT

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



NOTE

When you are ready to move to the next minor version, choose the channel that corresponds to that minor version. The sooner the update channel is declared, the more effectively the cluster can recommend update paths to your target version. The cluster might take some time to evaluate all the possible updates that are available and offer the best update recommendations to choose from. Update recommendations can change over time, as they are based on what update options are available at the time.

If you cannot see an update path to your target minor version, keep updating your cluster to the latest patch release for your current version until the next minor version is available in the path.

3. Apply an update:

- To update to the latest version:

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

- To update to a specific version:

```
$ oc adm upgrade --to=<version> 1
```

1 **1** **<version>** is the update version that you obtained from the output of the **oc adm upgrade** command.



IMPORTANT

When using **oc adm upgrade --help**, there is a listed option for **--force**. This is **heavily discouraged**, as using the **--force** option bypasses cluster-side guards, including release verification and precondition checks. Using **--force** does not guarantee a successful update. Bypassing guards put the cluster at risk.

4. Review the status of the Cluster Version Operator:

```
$ oc adm upgrade
```

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

```
$ oc adm upgrade
```

Example output

```
Cluster version is <version>
```

```
Upstream is unset, so the cluster will use an appropriate default.
```

```
Channel: stable-<version> (available channels: candidate-<version>, eus-<version>, fast-
<version>, stable-<version>)
```

```
No updates available. You may force an update to a specific release image, but doing so
might not be supported and might result in downtime or data loss.
```

6. If you are updating your cluster to the next minor version, such as version X.y to X.(y+1), it is recommended to confirm that your nodes are updated before deploying workloads that rely on a new feature:

```
$ oc get nodes
```

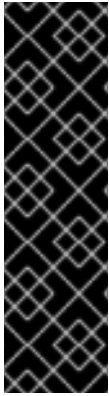
Example output

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-168-251.ec2.internal	Ready	master	82m	v1.31.3
ip-10-0-170-223.ec2.internal	Ready	master	82m	v1.31.3
ip-10-0-179-95.ec2.internal	Ready	worker	70m	v1.31.3
ip-10-0-182-134.ec2.internal	Ready	worker	70m	v1.31.3
ip-10-0-211-16.ec2.internal	Ready	master	82m	v1.31.3
ip-10-0-250-100.ec2.internal	Ready	worker	69m	v1.31.3

3.1.5. Finding recommended update paths with oc adm upgrade recommend (Technology Preview)

When updating your cluster, the **oc adm upgrade** command returns a list of the next available versions. You can use the **oc adm upgrade recommend** command to narrow down your suggestions and recommend a new target release before you launch your update.

The **oc adm upgrade recommend** command is read-only and cannot alter the state of your cluster.



IMPORTANT

The **oc adm upgrade recommend** command is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).



NOTE

Your cluster does not need to be a Technology Preview-enabled cluster in order for you to use the **oc adm upgrade recommend** command.

Procedure

1. Set the **OC_ENABLE_CMD_UPGRADE_RECOMMEND** environment variable to **true** by running the following command:

```
$ export OC_ENABLE_CMD_UPGRADE_RECOMMEND=true
```

2. Run one of the following commands:

- To list the recommended versions for your update, run the **oc adm upgrade recommend** command:

```
$ oc adm upgrade recommend
```

Example output

```
Upstream is unset, so the cluster will use an appropriate default.
Channel: stable-4.13 (available channels: candidate-4.12, candidate-4.13, eus-4.12, eus-4.14, fast-4.12, fast-4.13, stable-4.12, stable-4.13)
```

```
Updates to 4.13:
```

```
Version: 4.13.50
```

```
Image: quay.io/openshift-release-dev/ocp-release@sha256:6afb11e1cac46fd26476ca134072937115256b9c6360f7a1cd1812992c065f02
```

```
Reason: AdminAckRequired
```

```
Message: Kubernetes 1.26 and therefore OpenShift 4.13 remove several APIs which require admin consideration. Please see the knowledge article https://access.redhat.com/articles/6958394 for details and instructions.
```

```
Version: 4.13.49
```

```
Image: quay.io/openshift-release-dev/ocp-release@sha256:ab6f574665395d809511db9dc57764358278538eaae248c6d199208b3c30ab7d
```

```
Reason: AdminAckRequired
```

```
Message: Kubernetes 1.26 and therefore OpenShift 4.13 remove several APIs which require admin consideration. Please see the knowledge article
```

<https://access.redhat.com/articles/6958394> for details and instructions.

Updates to 4.12:

VERSION ISSUES

4.12.64 no known issues

4.12.63 no known issues

And 43 older 4.12 updates you can see with '--show-outdated-releases' or '--version VERSION'.

- To determine whether a specific version is recommended for your update, run the **oc adm upgrade recommend --version** command:

```
$ oc adm upgrade recommend --version 4.12.51
```

Example output

Upstream is unset, so the cluster will use an appropriate default.

Channel: stable-4.13 (available channels: candidate-4.12, candidate-4.13, eus-4.12, eus-4.14, fast-4.12, fast-4.13, stable-4.12, stable-4.13)

Update to 4.12.51 Recommended=False:

Image: quay.io/openshift-release-dev/ocp-release@sha256:158ced797e49f6caf7862acccef58484be63b642fdd2f66e6416295fa7958ab0

Release URL: <https://access.redhat.com/errata/RHSA-2024:1052>

Reason: MultipleReasons

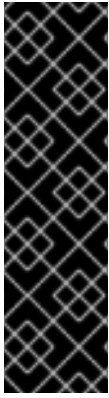
Message: An unintended reversion to the default kubelet nodeStatusReportFrequency can cause significant load on the control plane. <https://issues.redhat.com/browse/MCO-1094>

After rebooting into kernel-4.18.0-372.88.1.el8_6 or later, kernel nodes experience high load average and io_wait times. The nodes might fail to start or stop pods and probes may fail. Workload and host processes may become unresponsive and workload may be disrupted. <https://issues.redhat.com/browse/COS-2705>

3.1.6. Gathering cluster update status using oc adm upgrade status (Technology Preview)

When updating your cluster, the **oc adm upgrade** command returns limited information about the status of your update. You can use the **oc adm upgrade status** command to decouple status information from the **oc adm upgrade** command and return specific information regarding a cluster update, including the status of the control plane and worker node updates.

The **oc adm upgrade status** command is read-only and does not alter any state in your cluster.



IMPORTANT

The **oc adm upgrade status** command is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

The **oc adm upgrade status** command can be used for clusters from version 4.12 up to the latest supported release.



NOTE

Your cluster does not need to be a Technology Preview-enabled cluster in order for you to use the **oc adm upgrade status** command.

Procedure

1. Set the **OC_ENABLE_CMD_UPGRADE_STATUS** environment variable to **true** by running the following command:

```
$ export OC_ENABLE_CMD_UPGRADE_STATUS=true
```

2. Run the **oc adm upgrade status** command:

```
$ oc adm upgrade status
```

Example output for an update progressing successfully

```
= Control Plane =
Assessment:   Progressing
Target Version: 4.17.1 (from 4.17.0)
Updating:     machine-config
Completion:    97% (32 operators updated, 1 updating, 0 waiting)
Duration:     54m (Est. Time Remaining: <10m)
Operator Status: 32 Healthy, 1 Unavailable

Control Plane Nodes
NAME                                     ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-53-40.us-east-2.compute.internal Progressing  Draining 4.17.0   +10m
ip-10-0-30-217.us-east-2.compute.internal Outdated    Pending 4.17.0   ?
ip-10-0-92-180.us-east-2.compute.internal Outdated    Pending 4.17.0   ?

= Worker Upgrade =

WORKER POOL  ASSESSMENT  COMPLETION  STATUS
worker       Progressing  0% (0/2)    1 Available, 1 Progressing, 1 Draining
infra        Progressing  50% (1/2)    1 Available, 1 Progressing, 1 Draining

Worker Pool Nodes: Worker
```

```

NAME                                ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-4-159.us-east-2.compute.internal  Progressing Draining  4.17.0  +10m
ip-10-0-99-40.us-east-2.compute.internal  Outdated    Pending  4.17.0  ?

Worker Pool Nodes: infra
NAME                                ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-4-159-infra.us-east-2.compute.internal  Progressing Draining  4.17.0  +10m
ip-10-0-20-162.us-east-2.compute.internal      Completed   Updated   4.17.1  -

= Update Health =
SINCE  LEVEL  IMPACT  MESSAGE
54m4s  Info    None    Update is proceeding well

```

Additional resources

- [Performing a Control Plane Only update](#)
- [Updating along a conditional update path](#)
- [Understanding update channels and releases](#)

3.1.7. Updating along a conditional update path

You can update along a recommended conditional update 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 update 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 update version that you obtained from the output of the previous command, which is supported but also has known issues or risks.

Additional resources

- [Understanding update channels and releases](#)

3.1.8. Changing the update server by using the CLI

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

Procedure

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

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

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

Example output

```
clusterversion.config.openshift.io/version patched
```

3.2. UPDATING A CLUSTER USING THE WEB CONSOLE

You can perform minor version and patch updates on an OpenShift Container Platform cluster by using the web console.



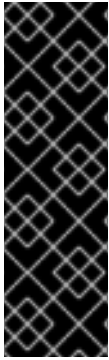
NOTE

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 4.18 channel.

3.2.1. Before updating the OpenShift Container Platform cluster

Before updating, consider the following:

- You have recently backed up etcd.
- In **PodDisruptionBudget**, if **minAvailable** is set to **1**, the nodes are drained to apply pending machine configs that 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.
- You might need to update the cloud provider resources for the new release if your cluster uses manually maintained credentials.
- You must review administrator acknowledgement requests, take any recommended actions, and provide the acknowledgement when you are ready.
- You can perform a partial update by updating the worker or custom pool nodes to accommodate the time it takes to update. You can pause and resume within the progress bar of each pool.



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.

3.2.2. 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.

Prerequisites

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

Procedure

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

Example output

```
...
spec:
  clusterID: db93436d-7b05-42cc-b856-43e11ad2d31a
  upstream: '<update-server-url>' 1
...
```

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

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

3. Click **Save**.

Additional resources

- [Understanding update channels and releases](#)

3.2.3. Pausing a MachineHealthCheck resource by using the web console


During the update 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.
2. Navigate to **Compute → MachineHealthChecks**.
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**.

3.2.4. 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 **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- Pause all **MachineHealthCheck** resources.
- You have updated all Operators previously installed through Operator Lifecycle Manager (OLM) to a version that is compatible with your target release. 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" in the "Additional resources" section for more information on how to check compatibility and, if necessary, update the installed Operators.
- Your 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.
- Your RHEL7 workers are replaced with RHEL8 or RHCOS workers. 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.

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-4.18**.



IMPORTANT

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



NOTE

When you are ready to move to the next minor version, choose the channel that corresponds to that minor version. The sooner the update channel is declared, the more effectively the cluster can recommend update paths to your target version. The cluster might take some time to evaluate all the possible updates that are available and offer the best update recommendations to choose from. Update recommendations can change over time, as they are based on what update options are available at the time.

If you cannot see an update path to your target minor version, keep updating your cluster to the latest patch release for your current version until the next minor version is available in the path.

- 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 updating your cluster to the next minor version, for example from version 4.10 to 4.11, confirm that 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-***, **eus-*** 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.

Additional resources

- [Updating installed Operators](#)

3.2.5. Viewing conditional updates in the web console

You can view and assess the risks associated with particular updates with conditional updates.

Prerequisites

- You have access to the cluster with **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- Pause all **MachineHealthCheck** resources.
- You have updated all Operators previously installed through Operator Lifecycle Manager (OLM) to a version that is compatible with your target release. 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" in the "Additional resources" section for more information on how to check compatibility and, if necessary, update the installed Operators.
- Your 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 an advanced update strategy, such as a canary rollout, an EUS update, or a control-plane update.

Procedure

1. From the web console, click **Administration** → **Cluster settings** page and review the contents of the **Details** tab.
2. You can enable the **Include versions with known issues** feature in the **Select new version** dropdown of the **Update cluster** modal to populate the dropdown list with conditional updates.



NOTE

If a version with known issues is selected, more information is provided with potential risks that are associated with the version.

3. Review the notification detailing the potential risks to updating.

Additional resources

- [Updating installed Operators](#)
- [Update recommendations and Conditional Updates](#)

3.2.6. 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 should be done with careful consideration and for short periods of time only.

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

3.2.7. 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 update 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

- [About the Machine Config Operator](#).

3.3. PERFORMING A CONTROL PLANE ONLY 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.y> to <4.y+1>, and then to <4.y+2>. You cannot update from OpenShift Container Platform <4.y> to <4.y+2> directly. However, administrators who want to update between two even-numbered minor versions can do so incurring only a single reboot of non-control plane hosts.



IMPORTANT

This update was previously known as an **EUS-to-EUS** update and is now referred to as a **Control Plane Only** update. These updates are only viable between **even-numbered minor versions** of OpenShift Container Platform.

There are several caveats to consider when attempting a Control Plane Only update.

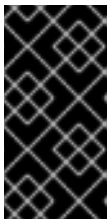
- Control Plane Only updates are only offered after updates between all versions involved have been made available in **stable** channels.
- If you encounter issues during or after updating to the odd-numbered minor version but before updating to the next even-numbered version, then remediation of those 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.
- Until the machine config pools are unpaused and the update is complete, some features and bugs fixes in <4.y+1> and <4.y+2> of OpenShift Container Platform are not available.
- 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 Control Plane Only updates with pools paused.

3.3.1. Performing a Control Plane Only update

The following procedure pauses all non-**master** machine config pools and performs updates from OpenShift Container Platform <4.y> to <4.y+1> to <4.y+2>, then unpauses the machine config pools. 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.y+1> and <4.y+2>.
- Review the release notes and product lifecycles for any layered products and Operator Lifecycle Manager (OLM) Operators. Some products and OLM Operators might require updates either before or during a Control Plane Only update.
- Ensure that you are familiar with version-specific prerequisites, such as the removal of deprecated APIs, that are required before updating from OpenShift Container Platform <4.y+1> to <4.y+2>.
- If your cluster uses in-tree vSphere volumes, update vSphere to version 7.0u3L+ or 8.0u2+.



IMPORTANT

If you do not update vSphere to 7.0u3L+ or 8.0u2+ before initiating an OpenShift Container Platform update, known issues might occur with your cluster after the update. For more information, see [Known Issues with OpenShift 4.12 to 4.13 or 4.13 to 4.14 vSphere CSI Storage Migration](#).

3.3.1.1. Control Plane Only update using the web console

Prerequisites

- Verify that machine config pools are unpaused.
- Have access to the web console as a user with **admin** privileges.

Procedure

1. Using the Administrator perspective on the web console, update any Operator Lifecycle Manager (OLM) Operators to the versions that are compatible with your intended updated version. You can find more information on how to perform this action in "Updating installed Operators"; see "Additional resources".
2. Verify that all machine config pools display a status of **Up to date** and that no machine config pool displays a status of **UPDATING**.
To view the status of all machine config pools, click **Compute** → **MachineConfigPools** and review the contents of the **Update status** column.



NOTE

If your machine config pools have an **Updating** status, please wait for this status to change to **Up to date**. This process could take several minutes.

3. Set your channel to **eus-<4.y+2>**.
To set your channel, click **Administration** → **Cluster Settings** → **Channel**. You can edit your channel by clicking on the current hyperlinked channel.

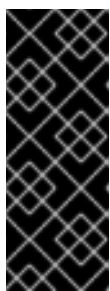
4. Pause all worker machine pools except for the master pool. You can perform this action on the **MachineConfigPools** tab under the **Compute** page. Select the vertical ellipses next to the machine config pool you'd like to pause and click **Pause updates**.
5. Update to version <4.y+1> and complete up to the **Save** step. You can find more information on how to perform these actions in "Updating a cluster by using the web console"; see "Additional resources".
6. Ensure that the <4.y+1> updates are complete by viewing the **Last completed version** of your cluster. You can find this information on the **Cluster Settings** page under the **Details** tab.
7. If necessary, update your OLM Operators by using the Administrator perspective on the web console. You can find more information on how to perform these actions in "Updating installed Operators"; see "Additional resources".
8. Update to version <4.y+2> and complete up to the **Save** step. You can find more information on how to perform these actions in "Updating a cluster by using the web console"; see "Additional resources".
9. Ensure that the <4.y+2> update is complete by viewing the **Last completed version** of your cluster. You can find this information on the **Cluster Settings** page under the **Details** tab.
10. Unpause all previously paused machine config pools. You can perform this action on the **MachineConfigPools** tab under the **Compute** page. Select the vertical ellipses next to the machine config pool you'd like to unpause and click **Unpause updates**.



IMPORTANT

If pools are paused, the cluster is not permitted to upgrade to any future minor versions, and some maintenance tasks are inhibited. This puts the cluster at risk for future degradation.

11. Verify that your previously paused pools are updated and that your cluster has completed the update to version <4.y+2>.
You can verify that your pools have updated on the **MachineConfigPools** tab under the **Compute** page by confirming that the **Update status** has a value of **Up to date**.



IMPORTANT

When you update a cluster that contains Red Hat Enterprise Linux (RHEL) compute machines, those machines 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. For more information, see "Updating a cluster that includes RHEL compute machines" in the additional resources section.

You can verify that your cluster has completed the update by viewing the **Last completed version** of your cluster. You can find this information on the **Cluster Settings** page under the **Details** tab.

Additional resources

- [Updating installed Operators](#)
- [Updating a cluster by using the web console](#)

- [Updating a cluster that includes RHEL compute machines](#)

3.3.1.2. Control Plane Only update using the CLI

Prerequisites

- Verify that machine config pools are unpaused.
- Update the OpenShift CLI (**oc**) to the target version before each update.



IMPORTANT

It is highly discouraged to skip this prerequisite. If the OpenShift CLI (**oc**) is not updated to the target version before your update, unexpected issues may occur.

Procedure

1. Using the Administrator perspective on the web console, update any Operator Lifecycle Manager (OLM) Operators to the versions that are compatible with your intended updated version. You can find more information on how to perform this action in "Updating installed Operators"; see "Additional resources".
2. Verify that all machine config pools display a status of **UPDATED** and that no machine config pool displays a status of **UPDATING**. To view the status of all machine config pools, run the following command:

```
$ oc get mcp
```

Example output

	NAME	CONFIG	UPDATED	UPDATING
	master	rendered-master-ecbb9582781c1091e1c9f19d50cf836c	True	False
	worker	rendered-worker-00a3f0c68ae94e747193156b491553d5	True	False

3. Your current version is <4.y>, and your intended version to update is <4.y+2>. Change to the **eus-<4.y+2>** channel by running the following command:

```
$ oc adm upgrade channel eus-<4.y+2>
```



NOTE

If you receive an error message indicating that **eus-<4.y+2>** is not one of the available channels, this indicates that Red Hat is still rolling out EUS version updates. This rollout process generally takes 45–90 days starting at the GA date.

4. Pause all worker machine pools except for the master pool by running the following command:

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



NOTE

You cannot pause the master pool.

- Update to the latest version by running the following command:

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

Example output

```
Updating to latest version <4.y+1.z>
```

- Review the cluster version to ensure that the updates are complete by running the following command:

```
$ oc adm upgrade
```

Example output

```
Cluster version is <4.y+1.z>
...
```

- Update to version <4.y+2> by running the following command:

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

- Retrieve the cluster version to ensure that the <4.y+2> updates are complete by running the following command:

```
$ oc adm upgrade
```

Example output

```
Cluster version is <4.y+2.z>
...
```

- To update your worker nodes to <4.y+2>, unpause all previously paused machine config pools by running the following command:

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



IMPORTANT

If pools are not unpaused, the cluster is not permitted to update to any future minor versions, and some maintenance tasks are inhibited. This puts the cluster at risk for future degradation.

- Verify that your previously paused pools are updated and that the update to version <4.y+2> is complete by running the following command:

```
$ oc get mcp
```



IMPORTANT

When you update a cluster that contains Red Hat Enterprise Linux (RHEL) compute machines, those machines 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. For more information, see "Updating a cluster that includes RHEL compute machines" in the additional resources section.

Example output

NAME	CONFIG	UPDATED	UPDATING
master	rendered-master-52da4d2760807cb2b96a3402179a9a4c	True	False
worker	rendered-worker-4756f60eccae96fb9dcb4c392c69d497	True	False

Additional resources

- [Updating installed Operators](#)
- [Updating a cluster that includes RHEL compute machines](#)

3.3.1.3. Control Plane Only updates for layered products and Operators installed through Operator Lifecycle Manager

In addition to the Control Plane Only update steps mentioned for the web console and CLI, there are additional steps to consider when performing Control Plane Only updates for clusters with the following:

- Layered products
- Operators installed through Operator Lifecycle Manager (OLM)

What is a layered product?

Layered products refer to products that are made of multiple underlying products that are intended to be used together and cannot be broken into individual subscriptions. For examples of layered OpenShift Container Platform products, see [Layered Offering On OpenShift](#).

As you perform a Control Plane Only update for the clusters of layered products and those of Operators that have been installed through OLM, you must complete the following:

1. You have updated all Operators previously installed through Operator Lifecycle Manager (OLM) to a version that is compatible with your target release. 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" in the "Additional resources" section for more information on how to check compatibility and, if necessary, update the installed Operators.
2. Confirm the cluster version compatibility between the current and intended Operator versions. You can verify which versions your OLM Operators are compatible with by using the [Red Hat OpenShift Container Platform Operator Update Information Checker](#).

As an example, here are the steps to perform a Control Plane Only update from <4.y> to <4.y+2> for 'OpenShift Data Foundation'. This can be done through the CLI or web console. For information about how to update clusters through your desired interface, see *Control Plane Only update using the web console* and "Control Plane Only update using the CLI" in "Additional resources".

Example workflow

1. Pause the worker machine pools.
2. Update OpenShift <4.y> → OpenShift <4.y+1>.
3. Update ODF <4.y> → ODF <4.y+1>.
4. Update OpenShift <4.y+1> → OpenShift <4.y+2>.
5. Update to ODF <4.y+2>.
6. Unpause the worker machine pools.

**NOTE**

The update to ODF <4.y+2> can happen before or after worker machine pools have been unpause.

Additional resources

- [Updating installed Operators](#)
- [Performing a Control Plane Only update using the web console](#)
- [Performing a Control Plane Only update using the CLI](#)
- [Preventing workload updates during a Control Plane Only update](#)

3.4. PERFORMING A CANARY ROLLOUT UPDATE

A *canary update* is an update strategy where worker node updates are performed in discrete, sequential stages instead of updating all worker nodes at the same time. This strategy can be useful in the following scenarios:

- You want a more controlled rollout of worker node updates to ensure that mission-critical applications stay available during the whole update, even if the update process causes your applications to fail.
- You 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.
- You 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.

3.4.1. Example Canary update strategy

The following example describes a canary update strategy where you have a cluster with 100 nodes with 10% excess capacity, you have 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.



NOTE

The previous values are an example only. The time it takes to drain a node might vary depending on factors such as workloads.

Defining custom machine config pools

In order to organize the worker node updates into separate stages, you can begin by defining the following MCPs:

- **workerpool-canary** with 10 nodes
- **workerpool-A** with 30 nodes
- **workerpool-B** with 30 nodes
- **workerpool-C** with 30 nodes

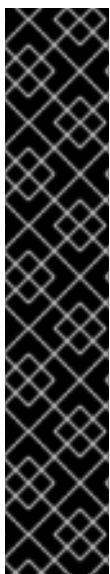
Updating the canary worker pool

During your first maintenance window, you pause the MCPs for **workerpool-A**, **workerpool-B**, and **workerpool-C**, and then initiate the cluster update. This updates components that run on top of OpenShift Container Platform and the 10 nodes that are part of the unpaused **workerpool-canary** MCP. The other three MCPs are not updated because they were paused.

Determining whether to proceed with the remaining worker pool updates

If for some reason you determine that your cluster or workload health was negatively affected by the **workerpool-canary** update, you then cordon and drain all nodes in that pool while still maintaining sufficient capacity until you have diagnosed and resolved the problem. When everything is working as expected, you evaluate the cluster and workload health before deciding to unpaue, and thus update, **workerpool-A**, **workerpool-B**, and **workerpool-C** in succession during each additional maintenance window.

Managing worker node updates using custom MCPs provides flexibility, however it can be a time-consuming process that requires you execute multiple commands. This complexity can result in errors that might 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.



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.

**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.

3.4.2. About the canary rollout update process and MCPs

In OpenShift Container Platform, nodes are not considered individually. Instead, they are grouped into machine config pools (MCPs). By default, nodes in an OpenShift Container Platform cluster are grouped into two MCPs: 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 an MCP up to the specified **maxUnavailable** number of nodes, if a max number is specified. By default, **maxUnavailable** is set to **1**. Draining and cordoning 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.

Using custom machine config pools

To prevent specific nodes from being updated, you can create custom MCPs. Because the MCO does not update nodes within paused MCPs, you can pause the MCPs containing nodes that you do not want to update before initiating a cluster update.

Using one or more custom MCPs can give you more control over the sequence in which you update your worker nodes. For example, 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.

**WARNING**

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

**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.

Considerations when using custom machine config pools

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

You must also consider how much extra capacity is available in your cluster to determine the number of custom MCPs and the amount of nodes within each MCP. In a case where your applications fail to work

as expected on newly updated nodes, you can cordon and drain those nodes in the pool, which moves the application pods to other nodes. However, you must determine whether the available nodes in the remaining MCPs can provide sufficient quality-of-service (QoS) for your applications.



NOTE

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.

3.4.3. About performing a canary rollout update

The following steps outline the high-level workflow of the canary rollout update process:

1. Create custom machine config pools (MCP) based on the worker pool.



NOTE

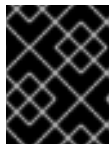
You can change the **maxUnavailable** setting in an MCP to specify the percentage or the number of machines that can be updating at any given time. The default is **1**.



WARNING

The default setting for **maxUnavailable** is **1** for all the machine config pools in OpenShift Container Platform. It is recommended to not change this value and update one control plane node at a time. Do not change this value to **3** for the control plane pool.

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.



IMPORTANT

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.
4. Perform the cluster update. The update process updates the MCPs that are not paused, including the control plane nodes.
5. Test your applications on the updated nodes to ensure they are working as expected.
6. Unpause one of the remaining MCPs, wait for the nodes in that pool to finish updating, and test the applications on those nodes. Repeat this process until all worker nodes are updated.
7. Optional: Remove the custom label from updated nodes and delete the custom MCPs.

3.4.4. Creating machine config pools to perform a canary rollout update

To perform a canary rollout update, you must first create one or more custom machine config pools (MCP).

Procedure

1. List the worker nodes in your cluster by running the following command:

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

Example output

```
ci-ln-pwnll6b-f76d1-s8t9n-worker-a-s75z4
ci-ln-pwnll6b-f76d1-s8t9n-worker-b-dglj2
ci-ln-pwnll6b-f76d1-s8t9n-worker-c-lldbm
```

2. For each node that you want to delay, add a custom label to the node by running the following command:

```
$ 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
```

3. Create the new MCP:

- a. Create an MCP YAML file:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: workerpool-canary 1
spec:
  machineConfigSelector:
    matchExpressions:
    - {
      key: machineconfiguration.openshift.io/role,
      operator: In,
      values: [worker,workerpool-canary] 2
    }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/workerpool-canary: "" 3
```

- 1** Specify a name for the MCP.

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

b. Create the **MachineConfigPool** object by running the following command:

```
$ oc create -f <file_name>
```

Example output

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

4. View the list of MCPs in the cluster and their current state by running the following command:

```
$ oc get machineconfigpool
```

Example output

NAME	CONFIG	UPDATED	UPDATING
DEGRADED	MACHINECOUNT	READYMACHINECOUNT	UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT	AGE		
master	rendered-master-b0bb90c4921860f2a5d8a2f8137c1867	True	False
False	3	3	3
		0	97m
workerpool-canary	rendered-workerpool-canary-87ba3dec1ad78cb6aecebf7fbb476a36		
True	False	False	1
		1	1
		1	0
			2m42s
worker	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.

3.4.5. Managing machine configuration inheritance for a worker pool canary

You can configure a machine config pool (MCP) canary to inherit any **MachineConfig** assigned to an existing MCP. This configuration is useful when you want to use an MCP canary to test as you update nodes one at a time for an existing MCP.

Prerequisites

- You have created one or more MCPs.

Procedure

1. Create a secondary MCP as described in the following two steps:
 - a. Save the following configuration file as **machineConfigPool.yaml**.

Example machineConfigPool YAML

```
apiVersion: machineconfiguration.openshift.io/v1
```



```

kind: MachineConfigPool
metadata:
  name: worker-perf
spec:
  machineConfigSelector:
    matchExpressions:
      - {
        key: machineconfiguration.openshift.io/role,
        operator: In,
        values: [worker,worker-perf]
      }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/worker-perf: ""
# ...

```

- b. Create the new machine config pool by running the following command:

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

Example output

```
machineconfigpool.machineconfiguration.openshift.io/worker-perf created
```

2. Add some machines to the secondary MCP. The following example labels the worker nodes **worker-a**, **worker-b**, and **worker-c** to the MCP **worker-perf**:

```
$ oc label node worker-a node-role.kubernetes.io/worker-perf="
```

```
$ oc label node worker-b node-role.kubernetes.io/worker-perf="
```

```
$ oc label node worker-c node-role.kubernetes.io/worker-perf="
```

3. Create a new **MachineConfig** for the MCP **worker-perf** as described in the following two steps:
 - a. Save the following **MachineConfig** example as a file called **new-machineconfig.yaml**:

Example MachineConfig YAML

```

apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker-perf
  name: 06-kdump-enable-worker-perf
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - enabled: true
          name: kdump.service

```

```
kernelArguments:
  - crashkernel=512M
# ...
```

- b. Apply the **MachineConfig** by running the following command:

```
$ oc create -f new-machineconfig.yaml
```

4. Create the new canary MCP and add machines from the MCP you created in the previous steps. The following example creates an MCP called **worker-perf-canary**, and adds machines from the **worker-perf** MCP that you previously created.

- a. Label the canary worker node **worker-a** by running the following command:

```
$ oc label node worker-a node-role.kubernetes.io/worker-perf-canary=""
```

- b. Remove the canary worker node **worker-a** from the original MCP by running the following command:

```
$ oc label node worker-a node-role.kubernetes.io/worker-perf-
```

- c. Save the following file as **machineConfigPool-Canary.yaml**.

Example machineConfigPool-Canary.yaml file

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
metadata:
  name: worker-perf-canary
spec:
  machineConfigSelector:
    matchExpressions:
      - {
          key: machineconfiguration.openshift.io/role,
          operator: In,
          values: [worker,worker-perf,worker-perf-canary] 1
        }
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/worker-perf-canary: ""
```

- 1 Optional value. This example includes **worker-perf-canary** as an additional value. You can use a value in this way to configure members of an additional **MachineConfig**.

- d. Create the new **worker-perf-canary** by running the following command:

```
$ oc create -f machineConfigPool-Canary.yaml
```

Example output

```
machineconfigpool.machineconfiguration.openshift.io/worker-perf-canary created
```

5. Check if the **MachineConfig** is inherited in **worker-perf-canary**.

a. Verify that no MCP is degraded by running the following command:

```
$ oc get mcp
```

Example output

NAME	CONFIG	UPDATED	UPDATING
DEGRADED	MACHINECOUNT	READYMACHINECOUNT	
UPDATEDMACHINECOUNT	DEGRADEDMACHINECOUNT	AGE	
master	rendered-master-2bf1379b39e22bae858ea1a3ff54b2ac		True
False	False	3 3 3 0 5d16h	
worker	rendered-worker-b9576d51e030413cfab12eb5b9841f34		True
False	False	0 0 0 0 5d16h	
worker-perf	rendered-worker-perf-b98a1f62485fa702c4329d17d9364f6a		True
False	False	2 2 2 0 56m	
worker-perf-canary	rendered-worker-perf-canary-b98a1f62485fa702c4329d17d9364f6a		
True	False	False 1 1 1 0 44m	

b. Verify that the machines are inherited from **worker-perf** into **worker-perf-canary**.

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
...				
worker-a	Ready	worker,worker-perf-canary	5d15h	v1.27.13+e709aa5
worker-b	Ready	worker,worker-perf	5d15h	v1.27.13+e709aa5
worker-c	Ready	worker,worker-perf	5d15h	v1.27.13+e709aa5

c. Verify that **kdump** service is enabled on **worker-a** by running the following command:

```
$ systemctl status kdump.service
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
...				
kdump.service - Crash recovery kernel arming				
Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; preset: disabled)				
Active: active (exited) since Tue 2024-09-03 12:44:43 UTC; 10s ago				
Process: 4151139 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCESS)				
Main PID: 4151139 (code=exited, status=0/SUCCESS)				

d. Verify that the MCP has updated the **crashkernel** by running the following command:

```
$ cat /proc/cmdline
```

The output should include the updated **crashekernel** value, for example:

Example output

```
crashkernel=512M
```

6. Optional: If you are satisfied with the upgrade, you can return **worker-a** to **worker-perf**.

- a. Return **worker-a** to **worker-perf** by running the following command:

```
$ oc label node worker-a node-role.kubernetes.io/worker-perf=
```

- b. Remove **worker-a** from the canary MCP by running the following command:

```
$ oc label node worker-a node-role.kubernetes.io/worker-perf-canary-
```

3.4.6. Pausing the machine config pools

After you create your custom machine config pools (MCPs), you then pause those MCPs. Pausing an MCP prevents the Machine Config Operator (MCO) from updating the nodes associated with that MCP.

Procedure

1. Patch the MCP that you want paused by running the following command:

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

For example:

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

Example output

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

3.4.7. Performing the cluster update

After the machine config pools (MCP) enter a 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 cluster update is complete, you can begin to unpause the MCPs one at a time.

3.4.8. Unpausing the machine config pools

After the OpenShift Container Platform update is complete, unpause your custom machine config pools (MCP) one at a time. Unpausing an MCP allows the Machine Config Operator (MCO) to update the nodes associated with that MCP.

Procedure

1. Patch the MCP that you want to unpause:

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

For example:

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

Example output

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

2. Optional: Check the progress of the update by using one of the following options:
 - a. Check the progress from the web console by clicking **Administration** → **Cluster settings**.
 - b. Check the progress by running the following command:
3. Test your applications on the updated nodes to ensure that they are working as expected.
4. Repeat this process for any other paused MCPs, one at a time.



NOTE

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.

3.4.9. Moving a node to the original machine config pool

After you update and verify applications on nodes in a custom machine config pool (MCP), move the nodes back to their original MCP by removing the custom label that you added to the nodes.



IMPORTANT

A node must have a role to be properly functioning in the cluster.

Procedure

1. For each node in a custom MCP, remove the custom label from the node by running the following command:

```
$ 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-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz labeled
```

The Machine Config Operator moves the nodes back to the original MCP and reconciles the node to the MCP configuration.

- To ensure that node has been removed from the custom MCP, view the list of MCPs in the cluster and their current state by running the following command:

```
$ oc get mcp
```

Example output

NAME	CONFIG	UPDATED	UPDATING
DEGRADED	MACHINECOUNT	READYMACHINECOUNT	UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT	AGE		
master	rendered-master-1203f157d053fd987c7cbd91e3fbc0ed	True	False
False	3 3 3 0	61m	
workerpool-canary	rendered-mcp-noupdate-5ad4791166c468f3a35cd16e734c9028	True	False
False	False 0 0 0 0	21m	
worker	rendered-worker-5ad4791166c468f3a35cd16e734c9028	True	False
False	3 3 3 0	61m	

When 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.

- Optional: Delete the custom MCP by running the following command:

```
$ oc delete mcp <mcp_name>
```

3.5. UPDATING A CLUSTER THAT INCLUDES RHEL COMPUTE MACHINES

You can perform minor version and patch updates on an OpenShift Container Platform cluster. If your cluster contains Red Hat Enterprise Linux (RHEL) machines, you must take additional steps to update those machines.



IMPORTANT

The use of RHEL compute machines on OpenShift Container Platform clusters has been deprecated and will be removed in a future release.

3.5.1. Prerequisites

- Have access to the cluster as a user with **admin** privileges. See [Using RBAC to define and apply permissions](#).
- Have a recent [etcd backup](#) in case your update fails and you must restore your cluster to a previous state.

- Your RHEL7 workers are replaced with RHEL8 or RHCOS workers. 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, update the cloud provider resources for the new release. For more information, including how to determine if this is a requirement for your cluster, see [Preparing to update a cluster with manually maintained credentials](#).
- If you run an Operator or you have configured any application with the pod disruption budget, you might experience an interruption during the update 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](#)

3.5.2. 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 **cluster-admin** privileges.
- You have access to the OpenShift Container Platform web console.
- Pause all **MachineHealthCheck** resources.
- You have updated all Operators previously installed through Operator Lifecycle Manager (OLM) to a version that is compatible with your target release. 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" in the "Additional resources" section for more information on how to check compatibility and, if necessary, update the installed Operators.
- Your 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.
- Your RHEL7 workers are replaced with RHEL8 or RHCOS workers. 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.

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-4.18**.



IMPORTANT

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



NOTE

When you are ready to move to the next minor version, choose the channel that corresponds to that minor version. The sooner the update channel is declared, the more effectively the cluster can recommend update paths to your target version. The cluster might take some time to evaluate all the possible updates that are available and offer the best update recommendations to choose from. Update recommendations can change over time, as they are based on what update options are available at the time.

If you cannot see an update path to your target minor version, keep updating your cluster to the latest patch release for your current version until the next minor version is available in the path.

- 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 updating your cluster to the next minor version, for example from version 4.10 to 4.11, confirm that 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-***, **eus-*** 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.



IMPORTANT

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

Additional resources

- [Updating installed Operators](#)

3.5.3. 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.

3.5.3.1. About Ansible hooks for updates

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 update again.

3.5.3.2. 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:

```
---
# 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:

Example hook definitions in an inventory file

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

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

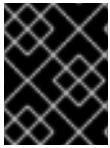
3.5.3.3. Available hooks for RHEL compute machines

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

Hook name	Description
openshift_node_pre_cordon_hook	<ul style="list-style-type: none"> Runs before each node is cordoned. This hook runs against each node in serial. If a task must run against a different host, the task must use delegate_to or local_action.
openshift_node_pre_upgrade_hook	<ul style="list-style-type: none"> Runs after each node is cordoned but before it is updated. This hook runs against each node in serial. If a task must run against a different host, the task must use delegate_to or local_action.
openshift_node_pre_uncordon_hook	<ul style="list-style-type: none"> Runs after each node is updated but before it is uncordoned. This hook runs against each node in serial. If a task must run against a different host, they task must use delegate_to or local_action.
openshift_node_post_upgrade_hook	<ul style="list-style-type: none"> Runs after each node uncordoned. It is the last node update action. This hook runs against each node in serial. If a task must run against a different host, the task must use delegate_to or local_action.

3.5.4. 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 8.6 and later 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 update 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:

```
# systemctl disable --now firewalld.service
```



NOTE

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

2. Enable the repositories that are required for OpenShift Container Platform 4.18:
 - a. On the machine that you run the Ansible playbooks, update the required repositories:

```
# subscription-manager repos --disable=rhocp-4.17-for-rhel-8-x86_64-rpms \
--enable=rhocp-4.18-for-rhel-8-x86_64-rpms
```



IMPORTANT

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 update 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:

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

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

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

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

```
# subscription-manager repos --disable=rhocp-4.17-for-rhel-8-x86_64-rpms \
--enable=rhocp-4.18-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:

```
[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:

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

- c. Run the **upgrade** playbook:

```
$ ansible-playbook -i /<path>/inventory/hosts playbooks/upgrade.yml 1
```

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

**NOTE**

The **upgrade** playbook only updates 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

NAME	STATUS	ROLES	AGE	VERSION
mycluster-control-plane-0	Ready	master	145m	v1.31.3
mycluster-control-plane-1	Ready	master	145m	v1.31.3
mycluster-control-plane-2	Ready	master	145m	v1.31.3
mycluster-rhel8-0	Ready	worker	98m	v1.31.3
mycluster-rhel8-1	Ready	worker	98m	v1.31.3
mycluster-rhel8-2	Ready	worker	98m	v1.31.3
mycluster-rhel8-3	Ready	worker	98m	v1.31.3

5. Optional: Update the operating system packages that were not updated by the **upgrade** playbook. To update packages that are not on 4.18, 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 4.18.

3.6. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT

You can update a cluster in an environment without access to the internet by taking additional steps to prepare your environment.

For information about updating a cluster in a disconnected environment, see [About cluster updates in a disconnected environment](#).

3.7. 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 15 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**

- Version 4.18 of OpenShift Container Platform requires VMware virtual hardware version 15 or later.
- Before upgrading OpenShift 4.12 to OpenShift 4.13, you must update vSphere to **v7.0.2 or later**, otherwise, the OpenShift 4.12 cluster is marked **un-upgradeable**.

3.7.1. 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.

**IMPORTANT**

As of OpenShift Container Platform 4.13, VMware virtual hardware version 13 is no longer supported. You need to update to VMware version 15 or later for supporting functionality.

3.7.1.1. 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 7.0U2 or later.

Procedure

1. List the control plane nodes in your cluster.

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

Example output

NAME	STATUS	ROLES	AGE	VERSION
control-plane-node-0	Ready	master	75m	v1.31.3
control-plane-node-1	Ready	master	75m	v1.31.3
control-plane-node-2	Ready	master	75m	v1.31.3

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 [Upgrade the Compatibility of a Virtual Machine Manually](#) 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**:

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

7. Mark the control plane node as schedulable again:

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

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

3.7.1.2. 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 7.0U2 or later.

Procedure

1. List the compute nodes in your cluster.

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

Example output

```
NAME           STATUS  ROLES  AGE  VERSION
compute-node-0 Ready   worker  30m  v1.31.3
compute-node-1 Ready   worker  30m  v1.31.3
compute-node-2 Ready   worker  30m  v1.31.3
```

Note the names of your compute nodes.

2. Mark the compute node as unschedulable:

```
$ oc adm cordon <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 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 [Upgrade the Compatibility of a Virtual Machine Manually](#) 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.

3.7.1.3. 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 7.0U2 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 virtual machine (VM) in the VMware vSphere client. Complete the steps outlined in [Upgrade the Compatibility of a Virtual Machine Manually](#) (VMware vSphere documentation).

**IMPORTANT**

If you modified the VM settings, those changes might reset after moving to a newer virtual hardware. Please review that all your configured settings are still in place after your upgrade before proceeding to the next step.

3. Convert the VM in the vSphere client to a template by right-clicking on the VM and then selecting **Template → Convert to Template**

**IMPORTANT**

The steps for converting a VM to a template might change in future vSphere documentation versions.

Additional resources

- [Understanding how to evacuate pods on nodes](#)

3.7.2. 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 update prior to performing an update of OpenShift Container Platform, the virtual hardware update occurs when the nodes are rebooted during the course of the OpenShift Container Platform update.

3.8. MIGRATING TO A CLUSTER WITH MULTI-ARCHITECTURE COMPUTE MACHINES

You can migrate your current cluster with single-architecture compute machines to a cluster with multi-architecture compute machines by updating to a multi-architecture, manifest-listed payload. This allows you to add mixed architecture compute nodes to your cluster.

For information about configuring your multi-architecture compute machines, see "Configuring multi-architecture compute machines on an OpenShift Container Platform cluster".

Before migrating your single-architecture cluster to a cluster with multi-architecture compute machines, it is recommended to install the Multiarch Tuning Operator, and deploy a **ClusterPodPlacementConfig** custom resource. For more information, see [Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator](#).

**IMPORTANT**

Migration from a multi-architecture payload to a single-architecture payload is not supported. Once a cluster has transitioned to using a multi-architecture payload, it can no longer accept a single-architecture update payload.

3.8.1. Migrating to a cluster with multi-architecture compute machines using the CLI**Prerequisites**

- You have access to the cluster as a user with the **cluster-admin** role.
- Your OpenShift Container Platform version is up to date to at least version 4.13.0.
For more information on how to update your cluster version, see *Updating a cluster using the web console* or *Updating a cluster using the CLI*.
- You have installed the OpenShift CLI (**oc**) that matches the version for your current cluster.
- Your **oc** client is updated to at least version 4.13.0.
- Your OpenShift Container Platform cluster is installed on AWS, Azure, GCP, bare metal or IBM P/Z platforms.
For more information on selecting a supported platform for your cluster installation, see *Selecting a cluster installation type*.

Procedure

1. Verify that the **RetrievedUpdates** condition is **True** in the Cluster Version Operator (CVO) by running the following command:

```
$ oc get clusterversion/version -o=jsonpath="{.status.conditions[?
(.type=='RetrievedUpdates')].status}"
```

If the **RetrievedUpdates** condition is **False**, you can find supplemental information regarding the failure by using the following command:

```
$ oc adm upgrade
```

For more information about cluster version condition types, see *Understanding cluster version condition types*.

2. If the condition **RetrievedUpdates** is **False**, change the channel to **stable-*<4.y>*** or **fast-*<4.y>*** with the following command:

```
$ oc adm upgrade channel <channel>
```

After setting the channel, verify if **RetrievedUpdates** is **True**.

For more information about channels, see *Understanding update channels and releases*.

3. Migrate to the multi-architecture payload with following command:

```
$ oc adm upgrade --to-multi-arch
```

Verification

- You can monitor the migration by running the following command:

```
$ oc adm upgrade
```

Example output

```
working towards ${VERSION}: 106 of 841 done (12% complete), waiting on machine-config
```



IMPORTANT

Machine launches may fail as the cluster settles into the new state. To notice and recover when machines fail to launch, we recommend deploying machine health checks. For more information about machine health checks and how to deploy them, see *About machine health checks*.

1. Optional: To retrieve more detailed information about the status of your update, monitor the migration by running the following command:

```
$ oc adm upgrade status
```

For more information about how to use the **oc adm upgrade status** command, see *Gathering cluster update status using oc adm upgrade status (Technology Preview)*.

The migrations must be complete and all the cluster operators must be stable before you can add compute machine sets with different architectures to your cluster.

Additional resources

- [Configuring multi-architecture compute machines on an OpenShift Container Platform cluster](#)
- [Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator](#).
- [Updating a cluster using the web console](#)
- [Updating a cluster using the CLI](#)
- [Understanding cluster version condition types](#)
- [Understanding update channels and releases](#)
- [Selecting a cluster installation type](#)
- [About machine health checks](#)
- [Gathering cluster update status using oc adm upgrade status \(Technology Preview\)](#)

3.8.2. Migrating the x86 control plane to arm64 architecture on Amazon Web Services

You can migrate the control plane in your cluster from **x86** to **arm64** architecture on Amazon Web Services (AWS).

Prerequisites

- You have installed the OpenShift CLI (**oc**).
- You logged in to **oc** as a user with **cluster-admin** privileges.

Procedure

1. Check the architecture of the control plane nodes by running the following command:

```
$ oc get nodes -o wide
```

Example output

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION
worker-001.example.com	Ready	worker	100d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								
worker-002.example.com	Ready	worker	98d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								
worker-003.example.com	Ready	worker	98d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								
master-001.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								
master-002.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								
master-003.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x	<none>	Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0	5.x.x-xxx.x.x.el9_xx.x86_64
cri-o://1.30.x								

The **KERNEL-VERSION** field in the output indicates the architecture of the nodes.

- Check that your cluster uses the multi payload by running the following command:

```
$ oc adm release info -o jsonpath='{ .metadata.metadata}'
```

If you see the following output, the cluster is multi-architecture compatible.

```
{
  "release.openshift.io/architecture": "multi",
  "url": "https://access.redhat.com/errata/<errata_version>"
}
```

If the cluster is not using the multi payload, migrate the cluster to a multi-architecture cluster. For more information, see "Migrating to a cluster with multi-architecture compute machines".

- Update your image stream from single-architecture to multi-architecture by running the following command:

```
$ oc import-image <multiarch_image_stream_tag> --from=
<registry>/<project_name>/<image_name> \
--import-mode='PreserveOriginal'
```

- Get the **arm64** compatible Amazon Machine Image (AMI) for configuring the control plane machine set by running the following command:

```
$ oc get configmap/coreos-bootimages -n openshift-machine-config-operator -o
jsonpath='{.data.stream}' | jq -r '.architectures.aarch64.images.aws.regions.'
<aws_region>".image' 1
```

- 1 Replace **<aws_region>** with the AWS region where the current cluster is installed. You can get the AWS region for the installed cluster by running the following command:

```
$ oc get infrastructure cluster -o jsonpath='{.status.platformStatus.aws.region}'
```

Example output

```
ami-xxxxxxx
```

5. Update the control plane machine set to support the **arm64** architecture by running the following command:

```
$ oc edit controlplanemachineset.machine.openshift.io cluster -n openshift-machine-api
```

Update the **instanceType** field to a type that supports the **arm64** architecture, and set the **ami.id** field to an AMI that is compatible with the **arm64** architecture. For information about supported instance types, see "Tested instance types for AWS on 64-bit ARM infrastructures".

For more information about configuring the control plane machine set for AWS, see "Control plane configuration options for Amazon Web Services".

Verification

- Verify that the control plane nodes are now running on the **arm64** architecture:

```
$ oc get nodes -o wide
```

Example output

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP
EXTERNAL-IP	OS-IMAGE			KERNEL-VERSION	
CONTAINER-RUNTIME					
worker-001.example.com	Ready	worker	100d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.x86_64 cri-o://1.30.x					
worker-002.example.com	Ready	worker	98d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.x86_64 cri-o://1.30.x					
worker-003.example.com	Ready	worker	98d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.x86_64 cri-o://1.30.x					
master-001.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.aarch64 cri-o://1.30.x					
master-002.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.aarch64 cri-o://1.30.x					
master-003.example.com	Ready	control-plane,master	120d	v1.30.7	10.x.x.x <none>
Red Hat Enterprise Linux CoreOS 4xx.xx.xxxxx-0 5.x.x-xxx.x.x.el9_xx.aarch64 cri-o://1.30.x					

Additional resources

- [Control plane configuration options for Amazon Web Services](#)
- [Tested instance types for AWS on 64-bit ARM infrastructures](#)
- [Migrating to a cluster with multi-architecture compute machines using the CLI](#)

3.9. UPDATING THE BOOT LOADER ON RHCOS NODES USING BOOTUPD

To update the boot loader on RHCOS nodes using **bootupd**, you must either run the **bootupctl update** command on RHCOS machines manually or provide a machine config with a **systemd** unit.

Unlike **grubby** or other boot loader tools, **bootupd** does not manage kernel space configuration such as passing kernel arguments. To configure kernel arguments, see [Adding kernel arguments to nodes](#).



NOTE

You can use **bootupd** to update the boot loader to protect against the BootHole vulnerability.

3.9.1. Updating the boot loader manually

You can manually inspect the status of the system and update the boot loader by using the **bootupctl** command-line tool.

1. Inspect the system status:

```
# bootupctl status
```

Example output for x86_64

```
Component EFI
Installed: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
Update: At latest version
```

Example output for aarch64

```
Component EFI
Installed: grub2-efi-aa64-1:2.02-99.el8_4.1.aarch64,shim-aa64-15.4-2.el8_1.aarch64
Update: At latest version
```

2. OpenShift Container Platform clusters initially installed on version 4.4 and older require an explicit adoption phase.

If the system status is **Adoptable**, perform the adoption:

```
# bootupctl adopt-and-update
```

Example output

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

3. If an update is available, apply the update so that the changes take effect on the next reboot:

```
# bootupctl update
```

Example output

```
Updated: grub2-efi-x64-1:2.04-31.el8_4.1.x86_64,shim-x64-15-8.el8_1.x86_64
```

3.9.2. Updating the bootloader automatically via a machine config

Another way to automatically update the boot loader with **bootupd** is to create a systemd service unit that will update the boot loader as needed on every boot. This unit will run the **bootupctl update** command during the boot process and will be installed on the nodes via a machine config.



NOTE

This configuration is not enabled by default as unexpected interruptions of the update operation may lead to unbootable nodes. If you enable this configuration, make sure to avoid interrupting nodes during the boot process while the bootloader update is in progress. The boot loader update operation generally completes quickly thus the risk is low.

1. Create a Butane config file, **99-worker-bootupctl-update.bu**, including the contents of the **bootupctl-update.service** systemd unit.



NOTE

The [Butane version](#) you specify in the config file should match the OpenShift Container Platform version and always ends in **0**. For example, **4.18.0**. See "Creating machine configs with Butane" for information about Butane.

Example output

```
variant: openshift
version: 4.18.0
metadata:
  name: 99-worker-chrony 1
  labels:
    machineconfiguration.openshift.io/role: worker 2
systemd:
  units:
    - name: bootupctl-update.service
      enabled: true
      contents: |
        [Unit]
        Description=Bootupd automatic update

        [Service]
        ExecStart=/usr/bin/bootupctl update
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
```

1 2 On control plane nodes, substitute **master** for **worker** in both of these locations.

2. Use Butane to generate a **MachineConfig** object file, **99-worker-bootupctl-update.yaml**, containing the configuration to be delivered to the nodes:

```
$ butane 99-worker-bootupctl-update.bu -o 99-worker-bootupctl-update.yaml
```

3. Apply the configurations in one of two ways:

- If the cluster is not running yet, after you generate manifest files, add the **MachineConfig** object file to the **<installation_directory>/openshift** directory, and then continue to create the cluster.
- If the cluster is already running, apply the file:

```
$ oc apply -f ./99-worker-bootupctl-update.yaml
```


CHAPTER 4. TROUBLESHOOTING A CLUSTER UPDATE

4.1. GATHERING DATA ABOUT YOUR CLUSTER UPDATE

When reaching out to Red Hat support for issues with an update, it is important to provide data for the support team to use for troubleshooting your failed cluster update.

4.1.1. Gathering log data for a support case

To gather data from your cluster, including log data, use the **oc adm must-gather** command. See *Gathering data about your cluster*.

4.1.2. Gathering cluster update status using **oc adm upgrade status** (Technology Preview)

When updating your cluster, the **oc adm upgrade** command returns limited information about the status of your update. You can use the **oc adm upgrade status** command to decouple status information from the **oc adm upgrade** command and return specific information regarding a cluster update, including the status of the control plane and worker node updates.

The **oc adm upgrade status** command is read-only and does not alter any state in your cluster.



IMPORTANT

The **oc adm upgrade status** command is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see [Technology Preview Features Support Scope](#).

The **oc adm upgrade status** command can be used for clusters from version 4.12 up to the latest supported release.



NOTE

Your cluster does not need to be a Technology Preview-enabled cluster in order for you to use the **oc adm upgrade status** command.

Procedure

1. Set the **OC_ENABLE_CMD_UPGRADE_STATUS** environment variable to **true** by running the following command:

```
$ export OC_ENABLE_CMD_UPGRADE_STATUS=true
```

2. Run the **oc adm upgrade status** command:

```
$ oc adm upgrade status
```

Example output for an update progressing successfully

```

= Control Plane =
Assessment:   Progressing
Target Version: 4.17.1 (from 4.17.0)
Updating:     machine-config
Completion:   97% (32 operators updated, 1 updating, 0 waiting)
Duration:     54m (Est. Time Remaining: <10m)
Operator Status: 32 Healthy, 1 Unavailable

Control Plane Nodes
NAME                                     ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-53-40.us-east-2.compute.internal Progressing  Draining  4.17.0  +10m
ip-10-0-30-217.us-east-2.compute.internal Outdated   Pending   4.17.0  ?
ip-10-0-92-180.us-east-2.compute.internal Outdated   Pending   4.17.0  ?

= Worker Upgrade =

WORKER POOL  ASSESSMENT  COMPLETION  STATUS
worker       Progressing  0% (0/2)    1 Available, 1 Progressing, 1 Draining
infra        Progressing  50% (1/2)    1 Available, 1 Progressing, 1 Draining

Worker Pool Nodes: Worker
NAME                                     ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-4-159.us-east-2.compute.internal Progressing  Draining  4.17.0  +10m
ip-10-0-99-40.us-east-2.compute.internal Outdated    Pending    4.17.0  ?

Worker Pool Nodes: infra
NAME                                     ASSESSMENT  PHASE    VERSION  EST  MESSAGE
ip-10-0-4-159-infra.us-east-2.compute.internal Progressing  Draining  4.17.0  +10m
ip-10-0-20-162.us-east-2.compute.internal   Completed   Updated    4.17.1  -

= Update Health =
SINCE  LEVEL  IMPACT  MESSAGE
54m4s  Info   None    Update is proceeding well

```

4.1.3. Gathering ClusterVersion history

The Cluster Version Operator (CVO) records updates made to a cluster, known as the ClusterVersion history. The entries can reveal correlation between changes in cluster behavior with potential triggers, although correlation does not imply causation.

**NOTE**

The initial, minor, and z-stream version updates are stored by the ClusterVersion history. However, the ClusterVersion history has a size limit. If the limit is reached, the oldest z-stream updates in previous minor versions are pruned to accommodate the limit.

You can view the ClusterVersion history by using the OpenShift Container Platform web console or by using the OpenShift CLI (**oc**).

4.1.3.1. Gathering ClusterVersion history in the OpenShift Container Platform web console

You can view the ClusterVersion history in the OpenShift Container Platform web console.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have access to the OpenShift Container Platform web console.

Procedure

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

4.1.3.2. Gathering ClusterVersion history using the OpenShift CLI (oc)

You can view the ClusterVersion history using the OpenShift CLI (**oc**).

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. View the cluster update history by entering the following command:

```
$ oc describe clusterversions/version
```

Example output

```
Desired:
Channels:
  candidate-4.13
  candidate-4.14
  fast-4.13
  fast-4.14
  stable-4.13
Image:   quay.io/openshift-release-dev/ocp-
release@sha256:a148b19231e4634196717c3597001b7d0af91bf3a887c03c444f59d9582864f4

URL:     https://access.redhat.com/errata/RHSA-2023:6130
Version: 4.13.19
History:
  Completion Time: 2023-11-07T20:26:04Z
  Image:          quay.io/openshift-release-dev/ocp-
release@sha256:a148b19231e4634196717c3597001b7d0af91bf3a887c03c444f59d9582864f4

  Started Time:    2023-11-07T19:11:36Z
  State:           Completed
  Verified:        true
  Version:         4.13.19
  Completion Time: 2023-10-04T18:53:29Z
  Image:          quay.io/openshift-release-dev/ocp-
```

```
release@sha256:eac141144d2ecd6cf27d24efe9209358ba516da22becc5f0abc199d25a9cfcec
```

```
Started Time: 2023-10-04T17:26:31Z
```

```
State: Completed
```

```
Verified: true
```

```
Version: 4.13.13
```

```
Completion Time: 2023-09-26T14:21:43Z
```

```
Image: quay.io/openshift-release-dev/ocp-
```

```
release@sha256:371328736411972e9640a9b24a07be0af16880863e1c1ab8b013f9984b4ef727
```

```
Started Time: 2023-09-26T14:02:33Z
```

```
State: Completed
```

```
Verified: false
```

```
Version: 4.13.12
```

```
Observed Generation: 4
```

```
Version Hash: CMLI3sLq-EA=
```

```
Events: <none>
```

Additional resources

- [Gathering data about your cluster](#)