OpenShift Container Platform 4.18

Disconnected environments

Managing OpenShift Container Platform clusters in a disconnected environment

Last Updated: 2025-12-18

OpenShift Container Platform 4.18 Disconnected environments

Managing OpenShift Container Platform clusters in a disconnected environment

Legal Notice

Copyright © Red Hat.

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, 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 Software Collections 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 information about installing, managing, and configuring OpenShift Container Platform clusters in environments without internet access.

Table of Contents

CHAPTER 1. ABOUT DISCONNECTED ENVIRONMENTS 7
1.1. GLOSSARY OF DISCONNECTED ENVIRONMENT TERMS 7
1.2. PREFERRED METHODS FOR WORKING WITH DISCONNECTED ENVIRONMENTS 7

CHAPTER 2. CONVERTING A CONNECTED CLUSTER TO A DISCONNECTED CLUSTER 9
2.1. ABOUT THE MIRROR REGISTRY 9
2.2. PREREQUISITES 10
2.3. PREPARING THE CLUSTER FOR MIRRORING 10
2.4. MIRRORING THE IMAGES 11
2.5. CONFIGURING THE CLUSTER FOR THE MIRROR REGISTRY 14
2.6. ENSURE APPLICATIONS CONTINUE TO WORK 17
2.7. DISCONNECT THE CLUSTER FROM THE NETWORK 17
2.8. RESTORING A DEGRADED INSIGHTS OPERATOR 18
2.9. RESTORING THE NETWORK 18

CHAPTER 3. ABOUT DISCONNECTED INSTALLATION MIRRORING 20
3.1. MIRROR REGISTRIES FOR INSTALLING CLUSTERS IN DISCONNECTED ENVIRONMENTS 20

CHAPTER 4. CREATING A MIRROR REGISTRY WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT 21
4.1. PREREQUISITES 21
4.2. MIRROR REGISTRY FOR RED HAT OPENSHIFT INTRODUCTION 21
4.2.1. Mirror registry for Red Hat OpenShift limitations 22
4.3. MIRRORING ON A LOCAL HOST WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT 22
4.4. UPDATING MIRROR REGISTRY FOR RED HAT OPENSHIFT FROM A LOCAL HOST 24
4.5. MIRRORING ON A REMOTE HOST WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT 25
4.6. UPDATING MIRROR REGISTRY FOR RED HAT OPENSHIFT FROM A REMOTE HOST 26
4.7. REPLACING MIRROR REGISTRY FOR RED HAT OPENSHIFT SSL/TLS CERTIFICATES 28
4.8. UNINSTALLING THE MIRROR REGISTRY FOR RED HAT OPENSHIFT 29
4.9. MIRROR REGISTRY FOR RED HAT OPENSHIFT FLAGS 29
4.10. MIRROR REGISTRY FOR RED HAT OPENSHIFT RELEASE NOTES 31
4.10.1. Mirror registry for Red Hat OpenShift 2.0 release notes 31
4.10.1.1. Mirror registry for Red Hat OpenShift 2.0.8 31
4.10.1.2. Mirror registry for Red Hat OpenShift 2.0.7 31
4.10.1.3. Mirror registry for Red Hat OpenShift 2.0.6 31
4.10.1.4. Mirror registry for Red Hat OpenShift 2.0.5 31
4.10.1.5. Mirror registry for Red Hat OpenShift 2.0.4 31
4.10.1.6. Mirror registry for Red Hat OpenShift 2.0.3 32
4.10.1.7. Mirror registry for Red Hat OpenShift 2.0.2 32
4.10.1.8. Mirror registry for Red Hat OpenShift 2.0.1 32
4.10.1.9. Mirror registry for Red Hat OpenShift 2.0.0 32
4.10.2. Mirror registry for Red Hat OpenShift 1.3 release notes 33
4.10.3. Mirror registry for Red Hat OpenShift 1.2 release notes 33
4.10.4. Mirror registry for Red Hat OpenShift 1.1 release notes 33
4.11. TROUBLESHOOTING MIRROR REGISTRY FOR RED HAT OPENSHIFT 33
4.12. ADDITIONAL RESOURCES 33

CHAPTER 5. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION BY USING THE OC-MIRROR PLUGIN V2 35
5.1. ABOUT OC-MIRROR PLUGIN V2 35
5.1.1. High level workflow 35
5.1.2. oc-mirror plugin v2 compatibility and support 36
5.2. PREREQUISITES 36

5.3. PREPARING YOUR MIRROR HOSTS 36
5.3.1. Installing the oc-mirror OpenShift CLI plugin 36
5.3.2. Configuring credentials that allow images to be mirrored 37
5.4. MIRRORING AN IMAGE SET TO A MIRROR REGISTRY 40
5.4.1. Creating the image set configuration 40
5.4.2. Comparison of oc-mirror workflows 41
5.4.3. Mirroring an image set in a partially disconnected environment 42
5.4.4. Mirroring an image set in a fully disconnected environment 43
5.4.4.1. Mirroring from mirror to disk 43
5.4.4.2. Mirroring from disk to mirror 43
5.5. ABOUT CUSTOM RESOURCES GENERATED BY OC-MIRROR PLUGIN V2 44
5.5.1. Restrictions on modifying resources that are generated by the oc-mirror plugin 45
5.5.2. Configuring your cluster to use the resources generated by oc-mirror plugin v2 45
5.6. DELETION OF IMAGES FROM YOUR DISCONNECTED ENVIRONMENT 47
5.6.1. Resolving storage cleanup issues in the distribution registry 48
5.6.2. Deleting images from a disconnected environment 49
5.7. VERIFYING YOUR SELECTED IMAGES FOR MIRRORING 51
5.7.1. Performing a dry run for oc-mirror plugin v2 51
5.7.2. Troubleshooting oc-mirror plugin v2 errors 52
5.8. BENEFITS OF ENCLAVE SUPPORT 54
5.8.1. Enclave mirroring workflow 54
5.8.2. Mirroring to an enclave 55
5.9. OC-MIRROR PLUGIN V2 SUPPORT PROXY SETTING 58
5.10. HOW FILTERING WORKS IN THE OPERATOR CATALOG 58
5.11. IMAGESET CONFIGURATION PARAMETERS FOR OC-MIRROR PLUGIN V2 63
5.11.1. DeleteImageSetConfiguration parameters 70
5.12. COMMAND REFERENCE FOR OC-MIRROR PLUGIN V2 74
5.12.1. Command reference for deleting images 75
5.13. NEXT STEPS 77

CHAPTER 6. MIGRATING FROM OC-MIRROR PLUGIN V1 TO V2 78
6.1. CHANGES FROM OC-MIRROR PLUGIN V1 TO V2 78
6.2. MIGRATING TO OC-MIRROR PLUGIN V2 79
6.3. ADDITIONAL RESOURCES 81

CHAPTER 7. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION USING THE OC-MIRROR PLUGIN 83
7.1. ABOUT THE OC-MIRROR PLUGIN 83
7.1.1. High level workflow 83
7.2. OC-MIRROR PLUGIN COMPATIBILITY AND SUPPORT 84
7.3. ABOUT THE MIRROR REGISTRY 84
7.4. PREREQUISITES 85
7.5. PREPARING YOUR MIRROR HOSTS 85
7.5.1. Installing the oc-mirror OpenShift CLI plugin 86
7.5.2. Configuring credentials that allow images to be mirrored 87
7.6. CREATING THE IMAGE SET CONFIGURATION 89
7.7. MIRRORING AN IMAGE SET TO A MIRROR REGISTRY 91
7.7.1. Mirroring an image set in a partially disconnected environment 91
7.7.1.1. Mirroring from mirror to mirror 91
7.7.2. Mirroring an image set in a fully disconnected environment 93
7.7.2.1. Mirroring from mirror to disk 93
7.7.2.2. Mirroring from disk to mirror 94
7.8. CONFIGURING YOUR CLUSTER TO USE THE RESOURCES GENERATED BY OC-MIRROR 95

7.9. UPDATING YOUR MIRROR REGISTRY CONTENT 96
7.9.1. Mirror registry update examples 97
7.9.1.1. Mirroring a specific OpenShift Container Platform version by pruning the existing images 97
7.9.1.2. Updating to the latest version of an Operator by pruning the existing images 98
7.9.1.3. Mirroring a new Operator by pruning the existing Operator 98
7.9.1.4. Pruning all the OpenShift Container Platform images 99
7.10. PERFORMING A DRY RUN 99
7.11. INCLUDING LOCAL OCI OPERATOR CATALOGS 100
7.12. IMAGE SET CONFIGURATION PARAMETERS 103
7.13. IMAGE SET CONFIGURATION EXAMPLES 110
7.13.1. Use case: Including the shortest OpenShift Container Platform update path 110
7.13.2. Use case: Including all versions of OpenShift Container Platform from a minimum to the latest version for multi-architecture releases 110
7.13.3. Use case: Including Operator versions from a minimum to the latest 111
7.13.4. Use case: Including the Nutanix CSI Operator 112
7.13.5. Use case: Including the default Operator channel 112
7.13.6. Use case: Including an entire catalog (all versions) 113
7.13.7. Use case: Including an entire catalog (channel heads only) 113
7.13.8. Use case: Including arbitrary images and helm charts 114
7.13.9. Use case: Including the upgrade path for EUS releases 114
7.13.10. Use case: Including the multi-arch OpenShift Container Platform images and catalog for multicluster engine Operator 115
7.14. COMMAND REFERENCE FOR OC-MIRROR 115
7.15. ADDITIONAL RESOURCES 117

CHAPTER 8. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION BY USING THE OC ADM COMMAND 118
8.1. PREREQUISITES 118
8.2. ABOUT THE MIRROR REGISTRY 118
8.3. PREPARING YOUR MIRROR HOST 119
8.3.1. Installing the OpenShift CLI on Linux 119
8.3.2. Installing the OpenShift CLI on Windows 120
8.3.3. Installing the OpenShift CLI on macOS 121
8.4. CONFIGURING CREDENTIALS THAT ALLOW IMAGES TO BE MIRRORED 121
8.5. MIRRORING THE OPENSHIFT CONTAINER PLATFORM IMAGE REPOSITORY 124
8.6. THE CLUSTER SAMPLES OPERATOR IN A DISCONNECTED ENVIRONMENT 127
8.6.1. Cluster Samples Operator assistance for mirroring 127
8.7. MIRRORING OPERATOR CATALOGS FOR USE WITH DISCONNECTED CLUSTERS 128
8.7.1. Prerequisites 128
8.7.2. Extracting and mirroring catalog contents 129
8.7.2.1. Mirroring catalog contents to registries on the same network 129
8.7.2.2. Mirroring catalog contents to airgapped registries 131
8.7.3. Generated manifests 133
8.7.4. Postinstallation requirements 134
8.8. NEXT STEPS 135
8.9. ADDITIONAL RESOURCES 135

CHAPTER 9. INSTALLING A CLUSTER IN A DISCONNECTED ENVIRONMENT 136
9.1. INSTALLING A CLUSTER WITH THE AGENT-BASED INSTALLER 136
9.2. INSTALLING A CLUSTER ON AMAZON WEB SERVICES 136
9.3. INSTALLING A CLUSTER ON MICROSOFT AZURE 136
9.4. INSTALLING A CLUSTER ON GOOGLE CLOUD 136
9.5. INSTALLING A CLUSTER ON IBM CLOUD 137
9.6. INSTALLING A CLUSTER ON NUTANIX 137

9.7. INSTALLING A BARE-METAL CLUSTER 137
9.8. INSTALLING A CLUSTER ON IBM Z(R) OR IBM(R) LINUXONE 137
9.9. INSTALLING A CLUSTER ON IBM POWER 137
9.10. INSTALLING A CLUSTER ON OPENSTACK 137
9.11. INSTALLING A CLUSTER ON VSPHERE 137

CHAPTER 10. USING OPERATOR LIFECYCLE MANAGER IN DISCONNECTED ENVIRONMENTS 139
10.1. PREREQUISITES 140
10.2. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES 140
10.3. MIRRORING AN OPERATOR CATALOG 141
10.4. ADDING A CATALOG SOURCE TO A CLUSTER 141
10.5. NEXT STEPS 143

CHAPTER 11. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT 144
11.1. ABOUT CLUSTER UPDATES IN A DISCONNECTED ENVIRONMENT 144
11.1.1. Mirroring OpenShift Container Platform images 144
11.1.2. Performing a cluster update in a disconnected environment 144
11.1.3. Uninstalling the OpenShift Update Service from a cluster 144
11.2. MIRRORING OPENSHIFT CONTAINER PLATFORM IMAGES 144
11.2.1. Mirroring resources using the oc-mirror plugin 145
11.2.2. Mirroring images using the oc adm release mirror command 145
11.2.2.1. Prerequisites 145
11.2.2.2. Preparing your mirror host 146
11.2.2.2.1. Installing the OpenShift CLI on Linux 146
11.2.2.2.2. Installing the OpenShift CLI on Windows 147
11.2.2.2.3. Installing the OpenShift CLI on macOS 147
11.2.2.2.4. Configuring credentials that allow images to be mirrored 148
11.2.2.3. Mirroring images to a mirror registry 150
11.3. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT USING THE OPENSHIFT UPDATE SERVICE 154
11.3.1. Using the OpenShift Update Service in a disconnected environment 154
11.3.2. Prerequisites 155
11.3.3. Configuring access to a secured registry for the OpenShift Update Service 155
11.3.4. Updating the global cluster pull secret 155
11.3.5. Installing the OpenShift Update Service Operator 156
11.3.5.1. Installing the OpenShift Update Service Operator by using the web console 157
11.3.5.2. Installing the OpenShift Update Service Operator by using the CLI 157
11.3.6. Creating the OpenShift Update Service graph data container image 159
11.3.7. Creating an OpenShift Update Service application 160
11.3.7.1. Creating an OpenShift Update Service application by using the web console 160
11.3.7.2. Creating an OpenShift Update Service application by using the CLI 161
11.3.8. Configuring the Cluster Version Operator (CVO) 163
11.3.9. Next steps 164
11.4. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT WITHOUT THE OPENSHIFT UPDATE SERVICE 164
11.4.1. Prerequisites 165
11.4.2. Pausing a MachineHealthCheck resource 165
11.4.3. Retrieving a release image digest 166
11.4.4. Updating the disconnected cluster 167
11.4.5. Understanding image registry repository mirroring 168
11.4.5.1. Configuring image registry repository mirroring 170
11.4.5.2. Converting ImageContentSourcePolicy (ICSP) files for image registry repository mirroring 175
11.4.6. Widening the scope of the mirror image catalog to reduce the frequency of cluster node reboots 176

11.4.7. Additional resources 177
11.5. UNINSTALLING THE OPENSHIFT UPDATE SERVICE FROM A CLUSTER 177
11.5.1. Deleting an OpenShift Update Service application 178
11.5.1.1. Deleting an OpenShift Update Service application by using the web console 178
11.5.1.2. Deleting an OpenShift Update Service application by using the CLI 178
11.5.2. Uninstalling the OpenShift Update Service Operator 179
11.5.2.1. Uninstalling the OpenShift Update Service Operator by using the web console 179
11.5.2.2. Uninstalling the OpenShift Update Service Operator by using the CLI 179

CHAPTER 1. ABOUT DISCONNECTED ENVIRONMENTS

A disconnected environment is an environment that does not have full access to the internet.

OpenShift Container Platform is designed to perform many automatic functions that depend on an internet connection, such as retrieving release images from a registry or retrieving update paths and recommendations for the cluster. Without a direct internet connection, you must perform additional setup and configuration for your cluster to maintain full functionality in the disconnected environment.

1.1. GLOSSARY OF DISCONNECTED ENVIRONMENT TERMS

Although it is used throughout the OpenShift Container Platform documentation, *disconnected environment* is a broad term that can refer to environments with various levels of internet connectivity. Other terms are sometimes used to refer to a specific level of internet connectivity, and these environments might require additional unique configurations.

The following table describes the different terms used to refer to environments without a full internet connection:

Table 1.1. Disconnected environment terms

Term	Description
Air-gapped network	An environment or network that is completely isolated from an external network. This isolation depends on a physical separation, or an "air gap", between machines on the internal network and any other part of an external network. Air-gapped environments are often used in industries with strict security or regulatory requirements.
Disconnected environment	An environment or network that has some level of isolation from an external network. This isolation could be enabled by physical or logical separation between machines on the internal network and an external network. Regardless of the level of isolation from the external network, a cluster in a disconnected environment does not have access to public services hosted by Red Hat and requires additional setup to maintain full cluster functionality.
Restricted Network	An environment or network with limited connection to an external network. A physical connection may exist between machines on the internal network and an external network, but network traffic is limited by additional configurations, such as with firewalls and proxies.

1.2. PREFERRED METHODS FOR WORKING WITH DISCONNECTED ENVIRONMENTS

You can choose between multiple options for most aspects of managing a cluster in a disconnected environment. For example, when mirroring images you can choose between using the oc-mirror OpenShift CLI (**oc**) plugin or using the **oc adm** command.

However, some options provide a simpler and more convenient user experience for disconnected environments, and are the preferred method over their alternatives.

Unless your organizational needs require you to choose another option, use the following methods for mirroring images, installing your cluster, and updating your cluster:

- Mirror your images using the oc-mirror plugin v2.
- Install your cluster using the Agent-based Installer.
- Update your cluster using a local OpenShift Update Service instance.

CHAPTER 2. CONVERTING A CONNECTED CLUSTER TO A DISCONNECTED CLUSTER

There might be some scenarios where you need to convert your OpenShift Container Platform cluster from a connected cluster to a disconnected cluster.

A disconnected cluster, also known as a restricted cluster, does not have an active connection to the internet. As such, you must mirror the contents of your registries and installation media. You can create this mirror registry on a host that can access both the internet and your closed network, or copy images to a device that you can move across network boundaries.

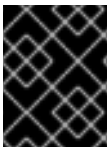
This topic describes the general process for converting an existing, connected cluster into a disconnected cluster.

2.1. ABOUT THE MIRROR REGISTRY

You must have access to the internet to obtain the necessary container images. Using an alternative registry means that you place the mirror registry on a mirror host that has access to both your network and the internet.

You can mirror the images that are required for OpenShift Container Platform installation and subsequent product updates to a container mirror registry such as Red Hat Quay, JFrog Artifactory, Sonatype Nexus Repository, or Harbor. If you do not have access to a large-scale container registry, you can use the *mirror registry for Red Hat OpenShift*, a small-scale container registry included with OpenShift Container Platform subscriptions.

You can use any container registry that supports Docker v2-2, such as Red Hat Quay, the *mirror registry for Red Hat OpenShift*, Artifactory, Sonatype Nexus Repository, or Harbor. Regardless of your chosen registry, the procedure to mirror content from Red Hat hosted sites on the internet to an isolated image registry is the same. After you mirror the content, you configure each cluster to retrieve this content from your mirror registry.



IMPORTANT

The OpenShift image registry cannot be used as the target registry because it does not support pushing without a tag, which is required during the mirroring process.

If choosing a container registry that is not the *mirror registry for Red Hat OpenShift*, it must be reachable by every machine in the clusters that you provision. If the registry is unreachable, installation, updating, or normal operations such as workload relocation might fail. For that reason, you must run mirror registries in a highly available way, and the mirror registries must at least match the production availability of your OpenShift Container Platform clusters.

When you populate your mirror registry with OpenShift Container Platform images, you can follow two scenarios. If you have a host that can access both the internet and your mirror registry, but not your cluster nodes, you can directly mirror the content from that machine. This process is referred to as *connected mirroring*. If you have no such host, you must mirror the images to a file system and then bring that host or removable media into your restricted environment. This process is referred to as *disconnected mirroring*.

For mirrored registries, to view the source of pulled images, you must review the **Trying to access** log entry in the CRI-O logs. Other methods to view the image pull source, such as using the **crictl images** command on a node, show the non-mirrored image name, even though the image is pulled from the mirrored location.

NOTE

Red Hat does not test third party registries with OpenShift Container Platform.

2.2. PREREQUISITES

- The **oc** client is installed.
- A running cluster.
- An installed mirror registry, which is a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as one of the following registries:
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus Repository
  - Harbor

  If you have a subscription to Red Hat Quay, see the documentation on deploying Red Hat Quay for proof-of-concept purposes or by using the Quay Operator.
- The mirror repository must be configured to share images. For example, a Red Hat Quay repository requires Organizations in order to share images.
- Access to the internet to obtain the necessary container images.

2.3. PREPARING THE CLUSTER FOR MIRRORING

Before disconnecting your cluster, you must mirror, or copy, the images to a mirror registry that is reachable by every node in your disconnected cluster. In order to mirror the images, you must prepare your cluster by:

- Adding the mirror registry certificates to the list of trusted CAs on your host.
- Creating a **.dockerconfigjson** file that contains your image pull secret, which is from the **cloud.openshift.com** token.

Procedure

1. Configuring credentials that allow image mirroring:
   a. Add the CA certificate for the mirror registry, in the simple PEM or DER file formats, to the list of trusted CAs. For example:

   ```
   $ cp </path/to/cert.crt> /usr/share/pki/ca-trust-source/anchors/
   ```

   where, </path/to/cert.crt>

   Specifies the path to the certificate on your local file system.

   b. Update the CA trust. For example, in Linux:

```
$ update-ca-trust
```

c. Extract the **.dockerconfigjson** file from the global pull secret:

```
$ oc extract secret/pull-secret -n openshift-config --confirm --to=.
```

Example output

```
.dockerconfigjson
```

d. Edit the **.dockerconfigjson** file to add your mirror registry and authentication credentials and save it as a new file:

```
{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}},"
<registry>:<port>/<namespace>/":{"auth":"<token>"}}}
```

where:

<local_registry>

Specifies the registry domain name, and optionally the port, that your mirror registry uses to serve content.

auth

Specifies the base64-encoded user name and password for your mirror registry.

<registry>:<port>/<namespace>

Specifies the mirror registry details.

<token>

Specifies the base64-encoded **username:password** for your mirror registry.

For example:

```
$ {"auths":{"cloud.openshift.com":
{"auth":"b3BlbnNoaWZ0Y3UjhGOVZPT0lOMEFaUjdPUzRGTA==","email":"user@exa
mple.com"},
"quay.io":
{"auth":"b3BlbnNoaWZ0LXJlbGVhc2UtZGOVZPT0lOMEFaUGSTd4VGVGVUjdPUzR
GTA==","email":"user@example.com"},
"registry.connect.redhat.com"
{"auth":"NTE3MTMwNDB8dWhjLTFEZlN3VHkxOSTd4VGVGVU1MdTpleUpoYkdjaUail
A==","email":"user@example.com"},
"registry.redhat.io":
{"auth":"NTE3MTMwNDB8dWhjLTFEZlN3VH3BGSTd4VGVGVU1MdTpleUpoYkdjaU9
fZw==","email":"user@example.com"},
"registry.svc.ci.openshift.org":
{"auth":"dXNlcjpyWjAwWVFjSEJiT2RKVW1pSmg4dW92dGp1SXRxQ3RGN1pwajJhN1
ZXeTRV"},"my-registry:5000/my-namespace/":
{"auth":"dXNlcm5hbWU6cGFzc3dvcmQ="}}}
```

2.4. MIRRORING THE IMAGES

After the cluster is properly configured, you can mirror the images from your external repositories to the mirror repository.

Procedure

1. Mirror the Operator Lifecycle Manager (OLM) images:

```
$ oc adm catalog mirror registry.redhat.io/redhat/redhat-operator-index:v{product-version}
<mirror_registry>:<port>/olm -a <reg_creds>
```

where:

product-version

Specifies the tag that corresponds to the version of OpenShift Container Platform to install, such as **4.8**.

mirror_registry

Specifies the fully qualified domain name (FQDN) for the target registry and namespace to mirror the Operator content to, where **<namespace>** is any existing namespace on the registry.

reg_creds

Specifies the location of your modified **.dockerconfigjson** file.

For example:

```
$ oc adm catalog mirror registry.redhat.io/redhat/redhat-operator-index:v4.8
mirror.registry.com:443/olm -a ./.dockerconfigjson  --index-filter-by-os='.*'
```

2. Mirror the content for any other Red Hat-provided Operator:

```
$ oc adm catalog mirror <index_image> <mirror_registry>:<port>/<namespace> -a
<reg_creds>
```

where:

index_image

Specifies the index image for the catalog that you want to mirror.

mirror_registry

Specifies the FQDN for the target registry and namespace to mirror the Operator content to, where **<namespace>** is any existing namespace on the registry.

reg_creds

Optional: Specifies the location of your registry credentials file, if required.

For example:

```
$ oc adm catalog mirror registry.redhat.io/redhat/community-operator-index:v4.8
mirror.registry.com:443/olm -a ./.dockerconfigjson  --index-filter-by-os='.*'
```

3. Mirror the OpenShift Container Platform image repository:

```
$ oc adm release mirror -a .dockerconfigjson --from=quay.io/openshift-release-dev/ocp-release:v<product-version>-<architecture> --to=<local_registry>/<local_repository> --to-release-image=<local_registry>/<local_repository>:v<product-version>-<architecture>
```

where:

product-version

Specifies the tag that corresponds to the version of OpenShift Container Platform to install, such as **4.8.15-x86_64**.

architecture

Specifies the type of architecture for your server, such as **x86_64**.

local_registry

Specifies the registry domain name for your mirror repository.

local_repository

Specifies the name of the repository to create in your registry, such as **ocp4/openshift4**.

For example:

```
$ oc adm release mirror -a .dockerconfigjson --from=quay.io/openshift-release-dev/ocp-release:4.8.15-x86_64 --to=mirror.registry.com:443/ocp/release --to-release-image=mirror.registry.com:443/ocp/release:4.8.15-x86_64
```

Example output

```
info: Mirroring 109 images to mirror.registry.com/ocp/release ...
mirror.registry.com:443/
  ocp/release
 manifests:
   sha256:086224cadce475029065a0efc5244923f43fb9bb3bb47637e0aaf1f32b9cad47 -> 4.8.15-x86_64-thanos
   sha256:0a214f12737cb1cfbec473cc301aa2c289d4837224c9603e99d1e90fc00328db -> 4.8.15-x86_64-kuryr-controller
   sha256:0cf5fd36ac4b95f9de506623b902118a90ff17a07b663aad5d57c425ca44038c -> 4.8.15-x86_64-pod
   sha256:0d1c356c26d6e5945a488ab2b050b75a8b838fc948a75c0fa13a9084974680cb -> 4.8.15-x86_64-kube-client-agent

…..
sha256:66e37d2532607e6c91eedf23b9600b4db904ce68e92b43c43d5b417ca6c8e63c mirror.registry.com:443/ocp/release:4.5.41-multus-admission-controller
sha256:d36efdbf8d5b2cbc4dcdbd64297107d88a31ef6b0ec4a39695915c10db4973f1 mirror.registry.com:443/ocp/release:4.5.41-cluster-kube-scheduler-operator
sha256:bd1baa5c8239b23ecdf76819ddb63cd1cd6091119fecdbf1a0db1fb3760321a2 mirror.registry.com:443/ocp/release:4.5.41-aws-machine-controllers
info: Mirroring completed in 2.02s (0B/s)

Success
Update image:  mirror.registry.com:443/ocp/release:4.5.41-x86_64
Mirror prefix: mirror.registry.com:443/ocp/release
```

4. Mirror any other registries, as needed:

```
$ oc image mirror <online_registry>/my/image:latest <mirror_registry>
```

Additional resources

- For more information about mirroring Operator catalogs, see Mirroring an Operator catalog.
- For more information about the **oc adm catalog mirror** command, see the OpenShift CLI administrator command reference.

2.5. CONFIGURING THE CLUSTER FOR THE MIRROR REGISTRY

After creating and mirroring the images to the mirror registry, you must modify your cluster so that pods can pull images from the mirror registry.

You must:

- Add the mirror registry credentials to the global pull secret.
- Add the mirror registry server certificate to the cluster.
- Create an **ImageContentSourcePolicy** custom resource (ICSP), which associates the mirror registry with the source registry.

1. Add mirror registry credential to the cluster global pull-secret:

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=
<pull_secret_location> 1
```

1 Provide the path to the new pull secret file.

For example:

```
$ oc set data secret/pull-secret -n openshift-config --from-
file=.dockerconfigjson=.mirrorsecretconfigjson
```

2. Add the CA-signed mirror registry server certificate to the nodes in the cluster:

a. Create a config map that includes the server certificate for the mirror registry

```
$ oc create configmap <config_map_name> --from-file=<mirror_address_host>..
<port>=$path/ca.crt -n openshift-config
```

For example:

```
S oc create configmap registry-config --from-
file=mirror.registry.com..443=/root/certs/ca-chain.cert.pem -n openshift-config
```

b. Use the config map to update the **image.config.openshift.io/cluster** custom resource (CR). OpenShift Container Platform applies the changes to this CR to all nodes in the cluster:

```
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"<config_map_name>"}}}' --type=merge
```

For example:

```
$ oc patch image.config.openshift.io/cluster --patch '{"spec":{"additionalTrustedCA":
{"name":"registry-config"}}}' --type=merge
```

3. Create an ICSP to redirect container pull requests from the online registries to the mirror registry:

   a. Create the **ImageContentSourcePolicy** custom resource:

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mirror-ocp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:443/ocp/release (1)
    source: quay.io/openshift-release-dev/ocp-release (2)
  - mirrors:
    - mirror.registry.com:443/ocp/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

   (1) Specifies the name of the mirror image registry and repository.

   (2) Specifies the online registry and repository containing the content that is mirrored.

   b. Create the ICSP object:

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

   Example output

```
imagecontentsourcepolicy.operator.openshift.io/mirror-ocp created
```

   OpenShift Container Platform applies the changes to this CR to all nodes in the cluster.

4. Verify that the credentials, CA, and ICSP for mirror registry were added:

   a. Log into a node:

```
$ oc debug node/<node_name>
```

   b. Set /**host** as the root directory within the debug shell:

```
sh-4.4# chroot /host
```

   c. Check the **config.json** file for the credentials:

```
sh-4.4# cat /var/lib/kubelet/config.json
```

   Example output

```
{"auths":{"brew.registry.redhat.io":{"xx=="},"brewregistry.stage.redhat.io":
{"auth":"xxx=="},"mirror.registry.com:443":{"auth":"xx="}}} 1
```

1 Ensure that the mirror registry and credentials are present.

d. Change to the **certs.d** directory

```
sh-4.4# cd /etc/docker/certs.d/
```

e. List the certificates in the **certs.d** directory:

```
sh-4.4# ls
```

Example output

```
image-registry.openshift-image-registry.svc.cluster.local:5000
image-registry.openshift-image-registry.svc:5000
mirror.registry.com:443 1
```

1 Ensure that the mirror registry is in the list.

f. Check that the ICSP added the mirror registry to the **registries.conf** file:

```
sh-4.4# cat /etc/containers/registries.conf
```

Example output

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

[[registry]]
  prefix = ""
  location = "quay.io/openshift-release-dev/ocp-release"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.registry.com:443/ocp/release"

[[registry]]
  prefix = ""
  location = "quay.io/openshift-release-dev/ocp-v4.0-art-dev"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.registry.com:443/ocp/release"
```

The **registry.mirror** parameters indicate that the mirror registry is searched before the original registry.

g. Exit the node.

```
sh-4.4# exit
```

2.6. ENSURE APPLICATIONS CONTINUE TO WORK

Before disconnecting the cluster from the network, ensure that your cluster is working as expected and all of your applications are working as expected.

Procedure

Use the following commands to check the status of your cluster:

- Ensure your pods are running:

```
$ oc get pods --all-namespaces
```

Example output

```
NAMESPACE                                NAME                                    READY
STATUS      RESTARTS   AGE
kube-system                              apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-0
1/1     Running     0          39m
kube-system                              apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-1
1/1     Running     0          39m
kube-system                              apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-2
1/1     Running     0          39m
openshift-apiserver-operator                 openshift-apiserver-operator-79c7c646fd-5rvr5
1/1     Running     3          45m
openshift-apiserver                      apiserver-b944c4645-q694g                    2/2
Running     0          29m
openshift-apiserver                      apiserver-b944c4645-shdxb                    2/2
Running     0          31m
openshift-apiserver                      apiserver-b944c4645-x7rf2                    2/2
Running     0          33m
 ...
```

- Ensure your nodes are in the READY status:

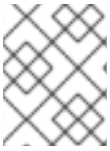
```
$ oc get nodes
```

Example output

```
NAME                                  STATUS   ROLES    AGE   VERSION
ci-ln-47ltxtb-f76d1-mrffg-master-0        Ready    master   42m   v1.31.3
ci-ln-47ltxtb-f76d1-mrffg-master-1        Ready    master   42m   v1.31.3
ci-ln-47ltxtb-f76d1-mrffg-master-2        Ready    master   42m   v1.31.3
ci-ln-47ltxtb-f76d1-mrffg-worker-a-gsxbz   Ready    worker   35m   v1.31.3
ci-ln-47ltxtb-f76d1-mrffg-worker-b-5qqdx   Ready    worker   35m   v1.31.3
ci-ln-47ltxtb-f76d1-mrffg-worker-c-rjkpq   Ready    worker   34m   v1.31.3
```

2.7. DISCONNECT THE CLUSTER FROM THE NETWORK

After mirroring all the required repositories and configuring your cluster to work as a disconnected cluster, you can disconnect the cluster from the network.



NOTE

The Insights Operator is degraded when the cluster loses its Internet connection. You can avoid this problem by temporarily disabling the Insights Operator until you can restore it.

2.8. RESTORING A DEGRADED INSIGHTS OPERATOR

Disconnecting the cluster from the network necessarily causes the cluster to lose the Internet connection. The Insights Operator becomes degraded because it requires access to Red Hat Insights.

This topic describes how to recover from a degraded Insights Operator.

Procedure

1. Edit your **.dockerconfigjson** file to remove the **cloud.openshift.com** entry, for example:

```
"cloud.openshift.com":{"auth":"<hash>","email":"user@example.com"}
```

2. Save the file.
3. Update the cluster secret with the edited **.dockerconfigjson** file:

```
$ oc set data secret/pull-secret -n openshift-config --from-
file=.dockerconfigjson=./.dockerconfigjson
```

4. Verify that the Insights Operator is no longer degraded:

```
$ oc get co insights
```

Example output

```
NAME       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
insights   4.5.41    True        False         False      3d
```

2.9. RESTORING THE NETWORK

If you want to reconnect a disconnected cluster and pull images from online registries, delete the cluster's ImageContentSourcePolicy (ICSP) objects. Without the ICSP, pull requests to external registries are no longer redirected to the mirror registry.

Procedure

1. View the ICSP objects in your cluster:

```
$ oc get imagecontentsourcepolicy
```

Example output

```
NAME              AGE
mirror-ocp        6d20h
ocp4-index-0      6d18h
qe45-index-0      6d15h
```

2. Delete all the ICSP objects you created when disconnecting your cluster:

```
$ oc delete imagecontentsourcepolicy <icsp_name> <icsp_name> <icsp_name>
```

For example:

```
$ oc delete imagecontentsourcepolicy mirror-ocp ocp4-index-0 qe45-index-0
```

Example output

```
imagecontentsourcepolicy.operator.openshift.io "mirror-ocp" deleted
imagecontentsourcepolicy.operator.openshift.io "ocp4-index-0" deleted
imagecontentsourcepolicy.operator.openshift.io "qe45-index-0" deleted
```

3. Wait for all the nodes to restart and return to the READY status and verify that the **registries.conf** file is pointing to the original registries and not the mirror registries:

   a. Log into a node:

```
$ oc debug node/<node_name>
```

   b. Set **/host** as the root directory within the debug shell:

```
sh-4.4# chroot /host
```

   c. Examine the **registries.conf** file:

```
sh-4.4# cat /etc/containers/registries.conf
```

Example output

```
unqualified-search-registries = ["registry.access.redhat.com", "docker.io"] 1
```

1 The **registry** and **registry.mirror** entries created by the ICSPs you deleted are removed.

CHAPTER 3. ABOUT DISCONNECTED INSTALLATION MIRRORING

As a cluster administrator, you can use a mirror registry to ensure that your clusters only use container images that satisfy your organizational controls on external content.

3.1. MIRROR REGISTRIES FOR INSTALLING CLUSTERS IN DISCONNECTED ENVIRONMENTS

You must mirror required container images into a disconnected environment before you can install and provision a cluster there. To mirror those container images, you must have a mirror registry. Consider the following options for creating and using a mirror registry:

- If you already have a container image registry, such as Red Hat Quay, you can use it as your mirror registry. If you do not already have a registry, you must create one.
- After you establish your registry, you need mirroring tools. To mirror your OpenShift Container Platform image repository to the mirror registry in your disconnected environment, you can use the oc-mirror OpenShift CLI (**oc**) plugin. The oc-mirror plugin is a single tool that mirrors all required OpenShift Container Platform content and other images to your mirror registry. The oc-mirror plugin is the preferred method for mirroring.
- Alternately, you can use the **oc adm** command to mirror just release and catalog images for OpenShift Container Platform.

CHAPTER 4. CREATING A MIRROR REGISTRY WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT

The *mirror registry for Red Hat OpenShift* is a small and streamlined container registry that you can use as a target for mirroring the required container images of OpenShift Container Platform for disconnected installations.

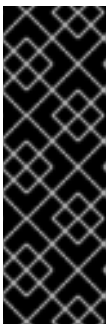
If you already have a container image registry, such as Red Hat Quay, you can skip this section and go straight to Mirroring the OpenShift Container Platform image repository .

IMPORTANT

The *mirror registry for Red Hat OpenShift* is not intended to be a substitute for a production deployment of Red Hat Quay.

4.1. PREREQUISITES

- An OpenShift Container Platform subscription.
- Red Hat Enterprise Linux (RHEL) 8 and 9 with Podman 3.4.2 or later and OpenSSL installed.
- Fully qualified domain name for the Red Hat Quay service, which must resolve through a DNS server.
- Key-based SSH connectivity on the target host. SSH keys are automatically generated for local installs. For remote hosts, you must generate your own SSH keys.
- 2 or more vCPUs.
- 8 GB of RAM.
- About 12 GB for OpenShift Container Platform 4.18 release images, or about 358 GB for OpenShift Container Platform 4.18 release images and OpenShift Container Platform 4.18 Red Hat Operator images.



IMPORTANT

- Up to 1 TB per stream or more is suggested.
- These requirements are based on local testing results with only release images and Operator images. Storage requirements can vary based on your organization's needs. You might require more space, for example, when you mirror multiple z-streams. You can use standard Red Hat Quay functionality or the proper API callout to remove unnecessary images and free up space.

4.2. MIRROR REGISTRY FOR RED HAT OPENSHIFT INTRODUCTION

For disconnected deployments of OpenShift Container Platform, a container registry is required to carry out the installation of the clusters. To run a production-grade registry service on such a cluster, you must create a separate registry deployment to install the first cluster. The *mirror registry for Red Hat OpenShift* addresses this need and is included in every OpenShift Container Platform subscription. It is available for download on the OpenShift console **Downloads** page.

The *mirror registry for Red Hat OpenShift* allows users to install a small-scale version of Red Hat Quay

and its required components by using the **mirror-registry** command-line interface (CLI) tool. The *mirror registry for Red Hat OpenShift* is deployed automatically with pre-configured local storage and a local database. It also includes auto-generated user credentials and access permissions with a single set of inputs and no additional configuration choices to get started.

The *mirror registry for Red Hat OpenShift* provides a pre-determined network configuration and reports deployed component credentials and access URLs upon success. A limited set of optional configuration inputs such as fully qualified domain name (FQDN) services, superuser name and password, and custom TLS certificates are also provided. This provides users with a container registry so that they can easily create an offline mirror of all OpenShift Container Platform release content when running OpenShift Container Platform in restricted network environments.

Use of the *mirror registry for Red Hat OpenShift* is optional if another container registry is already available in the install environment.

4.2.1. Mirror registry for Red Hat OpenShift limitations

The following limitations apply to the *mirror registry for Red Hat OpenShift*:

- The *mirror registry for Red Hat OpenShift* is not a highly-available registry and only local file system storage is supported. It is not intended to replace Red Hat Quay or the internal image registry for OpenShift Container Platform.
- The *mirror registry for Red Hat OpenShift* is not intended to be a substitute for a production deployment of Red Hat Quay.
- The *mirror registry for Red Hat OpenShift* is only supported for hosting images that are required to install a disconnected OpenShift Container Platform cluster, such as Release images or Red Hat Operator images. It uses local storage on your Red Hat Enterprise Linux (RHEL) machine, and storage supported by RHEL is supported by the *mirror registry for Red Hat OpenShift*.

NOTE

Because the *mirror registry for Red Hat OpenShift* uses local storage, you should remain aware of the storage usage consumed when mirroring images and use Red Hat Quay's garbage collection feature to mitigate potential issues. For more information about this feature, see "Red Hat Quay garbage collection".

- Support for Red Hat product images that are pushed to the *mirror registry for Red Hat OpenShift* for bootstrapping purposes are covered by valid subscriptions for each respective product. A list of exceptions to further enable the bootstrap experience can be found on the Self-managed Red Hat OpenShift sizing and subscription guide.
- Content built by customers should not be hosted by the *mirror registry for Red Hat OpenShift*.
- Using the *mirror registry for Red Hat OpenShift* with more than one cluster is discouraged because multiple clusters can create a single point of failure when updating your cluster fleet. Instead, use the *mirror registry for Red Hat OpenShift* to install a cluster that can host a production-grade, highly-available registry such as Red Hat Quay, which can serve OpenShift Container Platform content to other clusters.

4.3. MIRRORING ON A LOCAL HOST WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT

This procedure explains how to install the *mirror registry for Red Hat OpenShift* on a local host by using the **mirror-registry** installer tool. By doing so, users can create a local host registry running on port 443 for the purpose of storing a mirror of OpenShift Container Platform images.

NOTE

Installing the *mirror registry for Red Hat OpenShift* using the **mirror-registry** CLI tool makes several changes to your machine. After installation, a **$HOME/quay-install** directory is created, which has installation files, local storage, and the configuration bundle. Trusted SSH keys are generated in case the deployment target is the local host, and systemd files on the host machine are set up to ensure that container runtimes are persistent. Additionally, an initial user named **init** is created with an automatically generated password. All access credentials are printed at the end of the install routine.

Procedure

1. Download the **mirror-registry.tar.gz** package for the latest version of the *mirror registry for Red Hat OpenShift* found on the OpenShift console **Downloads** page.

2. Install the *mirror registry for Red Hat OpenShift* on your local host with your current user account by using the **mirror-registry** tool. For a full list of available flags, see "mirror registry for Red Hat OpenShift flags".

```
$ ./mirror-registry install \
  --quayHostname <host_example_com> \
  --quayRoot <example_directory_name>
```

3. Use the username and password generated during installation to log in to the registry by running the following command:

```
$ podman login -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 You can avoid running **--tls-verify=false** by configuring your system to trust the generated rootCA certificates. See "Securing Red Hat Quay" and "Configuring the system to trust the certificate authority" for more information.

NOTE

You can also log in by accessing the UI at **https://<host.example.com>:8443** after installation.

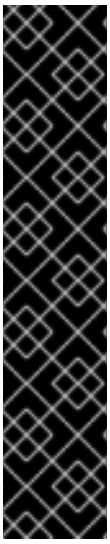
4. You can mirror OpenShift Container Platform images after logging in. Depending on your needs, see either the "Mirroring the OpenShift Container Platform image repository" or the "Mirroring Operator catalogs for use with disconnected clusters" sections of this document.

NOTE

If there are issues with images stored by the *mirror registry for Red Hat OpenShift* due to storage layer problems, you can re-mirror the OpenShift Container Platform images, or reinstall mirror registry on more stable storage.

4.4. UPDATING MIRROR REGISTRY FOR RED HAT OPENSHIFT FROM A LOCAL HOST

This procedure explains how to update the *mirror registry for Red Hat OpenShift* from a local host by using the **upgrade** command. Updating to the latest version ensures new features, bug fixes, and security vulnerability fixes.



IMPORTANT

When upgrading from version 1 to version 2, be aware of the following constraints:

- The worker count is set to **1** because multiple writes are not allowed in SQLite.
- You must not use the *mirror registry for Red Hat OpenShift* user interface (UP).
- Do not access the **sqlite-storage** Podman volume during the upgrade.
- There is intermittent downtime of your mirror registry because it is restarted during the upgrade process.
- PostgreSQL data is backed up under the **/$HOME/quay-install/quay-postgres-backup/** directory for recovery.

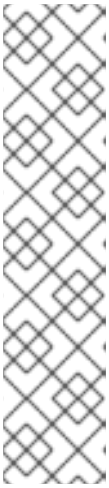
Prerequisites

- You have installed the *mirror registry for Red Hat OpenShift* on a local host.

Procedure

- If you are upgrading the *mirror registry for Red Hat OpenShift* from 1.3 → 2.y, and your installation directory is the default at **/etc/quay-install**, you can enter the following command:

```
$ sudo ./mirror-registry upgrade -v
```



NOTE

- *mirror registry for Red Hat OpenShift* migrates Podman volumes for Red Hat Quay storage, Postgres data, and **/etc/quay-install** data to the new **$HOME/quay-install** location. This allows you to use *mirror registry for Red Hat OpenShift* without the **--quayRoot** flag during future upgrades.
- Users who upgrade *mirror registry for Red Hat OpenShift* with the **./mirror-registry upgrade -v** flag must include the same credentials used when creating their mirror registry. For example, if you installed the *mirror registry for Red Hat OpenShift* with **--quayHostname <host_example_com>** and **--quayRoot <example_directory_name>**, you must include that string to properly upgrade the mirror registry.

- If you are upgrading *the mirror registry for Red Hat OpenShift* from 1.3 → 2.y and you used a custom quay configuration and storage directory in your 1.y deployment, you must pass in the **--quayRoot** and **--quayStorage** flags. For example:

```
$ sudo ./mirror-registry upgrade --quayHostname <host_example_com> --quayRoot
<example_directory_name>  --quayStorage <example_directory_name>/quay-storage -v
```

- If you are upgrading the *mirror registry for Red Hat OpenShift* from 1.3 → 2.y and want to specify a custom SQLite storage path, you must pass in the **--sqliteStorage** flag, for example:

```
$ sudo ./mirror-registry upgrade --sqliteStorage <example_directory_name>/sqlite-storage -v
```

Verification

1. Ensure that *mirror registry for Red Hat OpenShift* has been updated by running the following command:

```
$ podman ps
```

Example output

```
registry.redhat.io/quay/quay-rhel8:v3.12.10
```

4.5. MIRRORING ON A REMOTE HOST WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT

This procedure explains how to install the *mirror registry for Red Hat OpenShift* on a remote host by using the **mirror-registry** tool. By doing so, users can create a registry to hold a mirror of OpenShift Container Platform images.

NOTE

Installing the *mirror registry for Red Hat OpenShift* using the **mirror-registry** CLI tool makes several changes to your machine. After installation, a **$HOME/quay-install** directory is created, which has installation files, local storage, and the configuration bundle. Trusted SSH keys are generated in case the deployment target is the local host, and systemd files on the host machine are set up to ensure that container runtimes are persistent. Additionally, an initial user named **init** is created with an automatically generated password. All access credentials are printed at the end of the install routine.

Procedure

1. Download the **mirror-registry.tar.gz** package for the latest version of the *mirror registry for Red Hat OpenShift* found on the OpenShift console **Downloads** page.

2. Install the *mirror registry for Red Hat OpenShift* on your local host with your current user account by using the **mirror-registry** tool. For a full list of available flags, see "mirror registry for Red Hat OpenShift flags".

```
$ ./mirror-registry install -v \
  --targetHostname <host_example_com> \
  --targetUsername <example_user> \
```

```
-k ~/.ssh/my_ssh_key \
--quayHostname <host_example_com> \
--quayRoot <example_directory_name>
```

3. Use the username and password generated during installation to log in to the mirror registry by running the following command:

```
$ podman login -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 You can avoid running **--tls-verify=false** by configuring your system to trust the generated rootCA certificates. See "Securing Red Hat Quay" and "Configuring the system to trust the certificate authority" for more information.

NOTE

You can also log in by accessing the UI at **https://<host.example.com>:8443** after installation.

4. You can mirror OpenShift Container Platform images after logging in. Depending on your needs, see either the "Mirroring the OpenShift Container Platform image repository" or the "Mirroring Operator catalogs for use with disconnected clusters" sections of this document.

NOTE

If there are issues with images stored by the *mirror registry for Red Hat OpenShift* due to storage layer problems, you can re-mirror the OpenShift Container Platform images, or reinstall mirror registry on more stable storage.

4.6. UPDATING MIRROR REGISTRY FOR RED HAT OPENSHIFT FROM A REMOTE HOST

This procedure explains how to update the *mirror registry for Red Hat OpenShift* from a remote host by using the **upgrade** command. Updating to the latest version ensures bug fixes and security vulnerability fixes.

IMPORTANT

When upgrading from version 1 to version 2, be aware of the following constraints:

- The worker count is set to **1** because multiple writes are not allowed in SQLite.
- You must not use the *mirror registry for Red Hat OpenShift* user interface (UP).
- Do not access the **sqlite-storage** Podman volume during the upgrade.
- There is intermittent downtime of your mirror registry because it is restarted during the upgrade process.
- PostgreSQL data is backed up under the **/$HOME/quay-install/quay-postgres-backup/** directory for recovery.

Prerequisites

- You have installed the *mirror registry for Red Hat OpenShift* on a remote host.

Procedure

- To upgrade the *mirror registry for Red Hat OpenShift* from a remote host, enter the following command:

```
$ ./mirror-registry upgrade -v --targetHostname <remote_host_url> --targetUsername <user_name> -k ~/.ssh/my_ssh_key
```

NOTE

Users who upgrade the *mirror registry for Red Hat OpenShift* with the **./mirror-registry upgrade -v** flag must include the same credentials used when creating their mirror registry. For example, if you installed the *mirror registry for Red Hat OpenShift* with **--quayHostname <host_example_com>** and **--quayRoot <example_directory_name>**, you must include that string to properly upgrade the mirror registry.

- If you are upgrading the *mirror registry for Red Hat OpenShift* from 1.3 → 2.y and want to specify a custom SQLite storage path, you must pass in the **--sqliteStorage** flag, for example:

```
$ ./mirror-registry upgrade -v --targetHostname <remote_host_url> --targetUsername <user_name> -k ~/.ssh/my_ssh_key --sqliteStorage <example_directory_name>/quay-storage
```

Verification

1. Ensure that *mirror registry for Red Hat OpenShift* has been updated by running the following command:

```
$ podman ps
```

Example output

```
registry.redhat.io/quay/quay-rhel8:v3.12.10
```

4.7. REPLACING MIRROR REGISTRY FOR RED HAT OPENSHIFT SSL/TLS CERTIFICATES

In some cases, you might want to update your SSL/TLS certificates for the *mirror registry for Red Hat OpenShift*. This is useful in the following scenarios:

- If you are replacing the current *mirror registry for Red Hat OpenShift* certificate.
- If you are using the same certificate as the previous *mirror registry for Red Hat OpenShift* installation.
- If you are periodically updating the *mirror registry for Red Hat OpenShift* certificate.

Use the following procedure to replace *mirror registry for Red Hat OpenShift* SSL/TLS certificates.

Prerequisites

- You have downloaded and installed the **./mirror-registry** binary from the OpenShift console **Downloads** page.

Procedure

1. Enter the following command to install the *mirror registry for Red Hat OpenShift*:

```
$ ./mirror-registry install \
--quayHostname <host_example_com> \
--quayRoot <example_directory_name>
```

   This installs the *mirror registry for Red Hat OpenShift* to the **$HOME/quay-install** directory.

2. Prepare a new certificate authority (CA) bundle and generate new **ssl.key** and **ssl.crt** key files. For more information, see Configuring SSL and TLS for Red Hat Quay.

3. Assign **/$HOME/quay-install** an environment variable, for example, **QUAY**, by entering the following command:

```
$ export QUAY=/$HOME/quay-install
```

4. Copy the new **ssl.crt** file to the **/$HOME/quay-install** directory by entering the following command:

```
$ cp ~/ssl.crt $QUAY/quay-config
```

5. Copy the new **ssl.key** file to the **/$HOME/quay-install** directory by entering the following command:

```
$ cp ~/ssl.key $QUAY/quay-config
```

6. Restart the **quay-app** application pod by entering the following command:

```
$ systemctl --user restart quay-app
```

4.8. UNINSTALLING THE MIRROR REGISTRY FOR RED HAT OPENSHIFT

Use the following procedure to uninstall the *mirror registry for Red Hat OpenShift* from your local host.

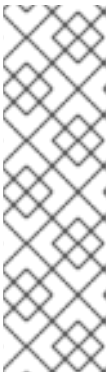
Prerequisites

- You have installed *mirror registry for Red Hat OpenShift* on a local host.

Procedure

- Uninstall the *mirror registry for Red Hat OpenShift* from your local host by running the following command:

```
$ ./mirror-registry uninstall -v \
  --quayRoot <example_directory_name>
```



NOTE

- Deleting the *mirror registry for Red Hat OpenShift* prompts the user before deletion. You can use **--autoApprove** to skip this prompt.
- Users who install the *mirror registry for Red Hat OpenShift* with the **--quayRoot** flag must include the **--quayRoot** flag when uninstalling. For example, if you installed the *mirror registry for Red Hat OpenShift* with **--quayRoot example_directory_name**, you must include that string to properly uninstall the mirror registry.

4.9. MIRROR REGISTRY FOR RED HAT OPENSHIFT FLAGS

The following flags are available for the *mirror registry for Red Hat OpenShift*:

Flags	Description
--autoApprove	A boolean value that disables interactive prompts. If set to **true**, the **quayRoot** directory is automatically deleted when uninstalling the mirror registry. Defaults to **false** if left unspecified.
--initPassword	The password of the init user created during Quay installation. Must be at least eight characters and contain no whitespace.
--initUser string	Shows the username of the initial user. Defaults to **init** if left unspecified.
--no-color, **-c**	Allows users to disable color sequences and propagate that to Ansible when running install, uninstall, and upgrade commands.

Flags	Description
--quayHostname	The fully-qualified domain name of the mirror registry that clients will use to contact the registry. Equivalent to **SERVER_HOSTNAME** in the Quay **config.yaml**. Must resolve by DNS. Defaults to**<targetHostname>:8443** if left unspecified. [1]
--quayStorage	The folder where Quay persistent storage data is saved. Defaults to the **quay-storage** Podman volume. Root privileges are required to uninstall.
--quayRoot, **-r**	The directory where container image layer and configuration data is saved, including **rootCA.key**, **rootCA.pem**, and **rootCA.srl** certificates. Defaults to **$HOME/quay-install** if left unspecified.
--sqliteStorage	The folder where SQLite database data is saved. Defaults to **sqlite-storage** Podman volume if not specified. Root is required to uninstall.
--ssh-key, **-k**	The path of your SSH identity key. Defaults to **~/.ssh/quay_installer** if left unspecified.
--sslCert	The path to the SSL/TLS public key / certificate. Defaults to **{quayRoot}/quay-config** and is auto-generated if left unspecified.
--sslCheckSkip	Skips the check for the certificate hostname against the **SERVER_HOSTNAME** in the **config.yaml** file. [2]
--sslKey	The path to the SSL/TLS private key used for HTTPS communication. Defaults to **{quayRoot}/quay-config** and is auto-generated if left unspecified.
--targetHostname, **-H**	The hostname of the target you want to install Quay to. Defaults to **$HOST**, for example, a local host, if left unspecified.
--targetUsername, **-u**	The user on the target host which will be used for SSH. Defaults to **$USER**, for example, the current user if left unspecified.
--verbose, **-v**	Shows debug logs and Ansible Playbook outputs.
--version	Shows the version for the *mirror registry for Red Hat OpenShift*.

1. **--quayHostname** must be modified if the public DNS name of your system is different from the local hostname. Additionally, the **--quayHostname** flag does not support installation with an IP address. Installation with a hostname is required.

2. **--sslCheckSkip** is used in cases when the mirror registry is set behind a proxy and the exposed hostname is different from the internal Quay hostname. It can also be used when users do not want the certificates to be validated against the provided Quay hostname during installation.

4.10. MIRROR REGISTRY FOR RED HAT OPENSHIFT RELEASE NOTES

The *mirror registry for Red Hat OpenShift* is a small and streamlined container registry that you can use as a target for mirroring the required container images of OpenShift Container Platform for disconnected installations.

These release notes track the development of the *mirror registry for Red Hat OpenShift* in OpenShift Container Platform.

4.10.1. Mirror registry for Red Hat OpenShift 2.0 release notes

The following sections provide details for each 2.0 release of the mirror registry for Red Hat OpenShift.

4.10.1.1. Mirror registry for Red Hat OpenShift 2.0.8

Issued: 16 October 2025

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.12.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2025:17062 - mirror registry for Red Hat OpenShift 2.0.8

4.10.1.2. Mirror registry for Red Hat OpenShift 2.0.7

Issued: 14 July 2025

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.10.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2025:9645 - mirror registry for Red Hat OpenShift 2.0.7

4.10.1.3. Mirror registry for Red Hat OpenShift 2.0.6

Issued: 28 April 2025

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.8.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2025:4251 - mirror registry for Red Hat OpenShift 2.0.6

4.10.1.4. Mirror registry for Red Hat OpenShift 2.0.5

Issued: 13 January 2025

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.5.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2025:0298 - mirror registry for Red Hat OpenShift 2.0.5

4.10.1.5. Mirror registry for Red Hat OpenShift 2.0.4

Issued: 06 January 2025

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.4.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2025:0033 - mirror registry for Red Hat OpenShift 2.0.4

4.10.1.6. Mirror registry for Red Hat OpenShift 2.0.3

Issued: 25 November 2024

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.3.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2024:10181 - mirror registry for Red Hat OpenShift 2.0.3

4.10.1.7. Mirror registry for Red Hat OpenShift 2.0.2

Issued: 31 October 2024

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.2.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2024:8370 - mirror registry for Red Hat OpenShift 2.0.2

4.10.1.8. Mirror registry for Red Hat OpenShift 2.0.1

Issued: 26 September 2024

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.1.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2024:7070 - mirror registry for Red Hat OpenShift 2.0.1

4.10.1.9. Mirror registry for Red Hat OpenShift 2.0.0

Issued: 03 September 2024

Mirror registry for Red Hat OpenShift is now available with Red Hat Quay 3.12.0.

The following advisory is available for the *mirror registry for Red Hat OpenShift*:

- RHBA-2024:5277 - mirror registry for Red Hat OpenShift 2.0.0

The following new features are available with *mirror registry for Red Hat OpenShift* 2.0.0:

- With the release of *mirror registry for Red Hat OpenShift*, the internal database has been upgraded from PostgreSQL to SQLite. As a result, data is now stored on the **sqlite-storage** Podman volume by default, and the overall tarball size is reduced by 300 MB.
  New installations use SQLite by default. Before upgrading to version 2.0, see "Updating mirror registry for Red Hat OpenShift from a local host" or "Updating mirror registry for Red Hat OpenShift from a remote host" depending on your environment.

- A new feature flag, **--sqliteStorage** has been added. With this flag, you can manually set the location where SQLite database data is saved.
- *Mirror registry for Red Hat OpenShift* is now available on IBM Power and IBM Z architectures (**s390x** and **ppc64le**).

4.10.2. Mirror registry for Red Hat OpenShift 1.3 release notes

To view the *mirror registry for Red Hat OpenShift* 1.3 release notes, see Mirror registry for Red Hat OpenShift 1.3 release notes.

4.10.3. Mirror registry for Red Hat OpenShift 1.2 release notes

To view the *mirror registry for Red Hat OpenShift* 1.2 release notes, see Mirror registry for Red Hat OpenShift 1.2 release notes.

4.10.4. Mirror registry for Red Hat OpenShift 1.1 release notes

To view the *mirror registry for Red Hat OpenShift* 1.1 release notes, see Mirror registry for Red Hat OpenShift 1.1 release notes.

4.11. TROUBLESHOOTING MIRROR REGISTRY FOR RED HAT OPENSHIFT

To assist in troubleshooting *mirror registry for Red Hat OpenShift*, you can gather logs of systemd services installed by the mirror registry. The following services are installed:

- quay-app.service
- quay-redis.service
- quay-pod.service

Prerequisites

- You have installed *mirror registry for Red Hat OpenShift*.

Procedure

- If you installed *mirror registry for Red Hat OpenShift* with root privileges, you can get the status information of its systemd services by entering the following command:

```
$ sudo systemctl status <service>
```

- If you installed *mirror registry for Red Hat OpenShift* as a standard user, you can get the status information of its systemd services by entering the following command:

```
$ systemctl --user status <service>
```

4.12. ADDITIONAL RESOURCES

- Red Hat Quay garbage collection

- Securing Red Hat Quay
- Configuring the system to trust the certificate authority
- Mirroring the OpenShift Container Platform image repository
- Mirroring Operator catalogs for use with disconnected clusters

CHAPTER 5. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION BY USING THE OC-MIRROR PLUGIN V2

You can run your cluster in a disconnected environment if you install the cluster from a mirrored set of OpenShift Container Platform container images in a private registry. This registry must be running whenever your cluster is running.

You can use oc-mirror plugin v2 to mirror images to a mirror registry in your fully or partially disconnected environments. To download the required images from the official Red Hat registries, you must run oc-mirror plugin v2 from a system with internet connectivity.

5.1. ABOUT OC-MIRROR PLUGIN V2

The oc-mirror OpenShift CLI (**oc**) plugin is a single tool that mirrors all required OpenShift Container Platform content and other images to your mirror registry.

To use the new version of oc-mirror, add the **--v2** flag to the oc-mirror plugin v2 command line.

oc-mirror plugin v2 has the following features:

- Provides a centralized method to mirror OpenShift Container Platform releases, Operators, helm charts, and other images.
- Verifies that the complete image set specified in the image set configuration file is mirrored to the mirrored registry, regardless of whether the images were previously mirrored or not.
- Uses a cache system instead of metadata, which prevents the need to start the mirroring process over in the case of a failure for a single step of the process.
- Maintains minimal archive sizes by incorporating only new images into the archive.
- Generates mirroring archives with content selected by mirroring date.
- Can generate **ImageDigestMirrorSet** (IDMS) and **ImageTagMirrorSet** (ITMS) resources, which cover the full image set, instead of an **ImageContentSourcePolicy** (ICSP) resource, which only covered incremental changes to the image set for each mirroring operation with v1.
- Does not perform automatic pruning. v2 now uses a **Delete** feature, which grants users more control over deleting images.
- Introduces support for **registries.conf** files. This change facilitates mirroring to multiple enclaves while using the same cache.

5.1.1. High level workflow

The following steps outline the high-level workflow on how to use the oc-mirror plugin v2 to mirror images to a mirror registry:

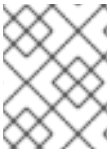
1. Create an image set configuration file.
2. Mirror the image set to the target mirror registry by using one of the following workflows:
    - Mirror an image set directly to the target mirror registry (mirror to mirror).

- Mirror an image set to disk (mirror to disk), transfer the **tar** file to the target environment, then mirror the image set to the target mirror registry (disk to mirror).

3. Configure your cluster to use the resources generated by the oc-mirror plugin v2.
4. Repeat these steps to update your target mirror registry as necessary.

5.1.2. oc-mirror plugin v2 compatibility and support

The oc-mirror plugin v2 is supported for OpenShift Container Platform.



NOTE

On **aarch64**, **ppc64le**, and **s390x** architectures the oc-mirror plugin v2 is supported only for OpenShift Container Platform 4.14 and later.

Use the latest available version of the oc-mirror plugin v2 regardless of which versions of OpenShift Container Platform you need to mirror.

5.2. PREREQUISITES

- You must have a container image registry that supports Docker V2-2 in the location that hosts the OpenShift Container Platform cluster, such as Red Hat Quay.

NOTE

- If you use Red Hat Quay, use version 3.6 or later with the oc-mirror plugin. See Deploying the Red Hat Quay Operator on OpenShift Container Platform (Red Hat Quay documentation). If you need additional assistance selecting and installing a registry, contact your sales representative or Red Hat Support.
- If you do not have an existing solution for a container image registry, OpenShift Container Platform subscribers receive a mirror registry for Red Hat OpenShift. This mirror registry is included with your subscription and serves as a small-scale container registry. You can use this registry to mirror the necessary container images of OpenShift Container Platform for disconnected installations.

- Every machine in the provisioned clusters must have access to the mirror registry. If the registry is unreachable, tasks like installation, updating, or routine operations such as workload relocation, might fail. Mirror registries must be operated in a highly available manner, ensuring their availability aligns with the production availability of your OpenShift Container Platform clusters.

5.3. PREPARING YOUR MIRROR HOSTS

To use the oc-mirror plugin v2 for image mirroring, you must install the plugin and create a file with credentials for container images, enabling you to mirror from Red Hat to your mirror.

5.3.1. Installing the oc-mirror OpenShift CLI plugin

Install the oc-mirror OpenShift CLI plugin to manage image sets in disconnected environments.

Prerequisites

- You have installed the OpenShift CLI (**oc**). If you are mirroring image sets in a fully disconnected environment, ensure the following:
  - You have installed the oc-mirror plugin on the host that has internet access.
  - The host in the disconnected environment has access to the target mirror registry.
- You have set the **umask** parameter to **0022** on the operating system that uses oc-mirror.
- You have installed the correct binary for the RHEL version that you are using.

Procedure

1. Download the oc-mirror CLI plugin:
   a. Navigate to the Downloads page of the Red Hat Hybrid Cloud Console.
   b. In the **OpenShift disconnected installation tools** section, select the **OS type** and **Architecture type** of the **OpenShift Client (oc) mirror plugin** from the dropdown menus.
   c. Click **Download** to save the file.
2. Extract the archive by running the following command:

```
$ tar xvzf oc-mirror.tar.gz
```

3. If necessary, update the plugin file to be executable by running the following command:

```
$ chmod +x oc-mirror
```

NOTE

Do not rename the **oc-mirror** file.

4. Install the oc-mirror CLI plugin by placing the file in your **PATH**, for example **/usr/local/bin**, by running the following command:

```
$ sudo mv oc-mirror /usr/local/bin/.
```

Verification

- Verify that the oc-mirror plugin v2 is successfully installed by running the following command:

```
$ oc mirror --v2 --help
```

5.3.2. Configuring credentials that allow images to be mirrored

Create a container image registry credentials file so that you can mirror images from Red Hat to your mirror. Complete the following steps on the installation host.

WARNING

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

Prerequisites

- You configured a mirror registry to use in your disconnected environment.
- You identified an image repository location on your mirror registry to mirror images into.
- You provisioned a mirror registry account that allows images to be uploaded to that image repository.
- You have write access to the mirror registry.

Procedure

1. Download your **registry.redhat.io** pull secret from Red Hat OpenShift Cluster Manager .
2. Make a copy of your pull secret in JSON format by running the following command:

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

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

Example pull secret

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

1. If the **$XDG_RUNTIME_DIR/containers** directory does not exist, create one by entering the following command:

```
$ mkdir -p $XDG_RUNTIME_DIR/containers
```

2. Save the pull secret file as **$XDG_RUNTIME_DIR/containers/auth.json**.

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

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

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

Example output

```
BGVtbYk3ZHAtqXs=
```

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

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

- For the **<mirror_registry>** value, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:8443**.

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

Example modified pull secret

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

```
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

5.4. MIRRORING AN IMAGE SET TO A MIRROR REGISTRY

Mirroring an image set to a mirror registry ensures that the required images are available in a secure and controlled environment, facilitating smoother deployments, updates, and maintenance tasks.

5.4.1. Creating the image set configuration

Before you can use the oc-mirror plugin v2 to mirror images, you must create an image set configuration file. This image set configuration file defines which OpenShift Container Platform releases, Operators, and other images to mirror, along with other configuration settings for the oc-mirror plugin v2.

Prerequisites

- You have created a container image registry credentials file. For instructions, see *Configuring credentials that allow images to be mirrored*.

Procedure

- Create an **ImageSetConfiguration** YAML file and modify it to include your required images.

  Example ImageSetConfiguration YAML file

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    channels:
    - name: stable-4.18 (1)
      minVersion: 4.18.2
      maxVersion: 4.18.2
    graph: true (2)
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18 (3)
      packages: (4)
       - name: aws-load-balancer-operator
       - name: 3scale-operator
       - name: node-observability-operator
  additionalImages: (5)
   - name: registry.redhat.io/ubi8/ubi:latest
   - name:
registry.redhat.io/ubi9/ubi@sha256:20f695d2a91352d4eaa25107535126727b5945bff38ed36a
3e59590f495046f0
```

(1) Set the channel to retrieve OpenShift Container Platform images from.

2 Add **graph: true** to build and push the graph-data image to the mirror registry. The graph-data image is required to create OpenShift Update Service (OSUS). The **graph: true** field also generates the **UpdateService** custom resource manifest. The **oc** command-line interface (CLI) can use the **UpdateService** custom resource manifest to create OSUS. For more information, see *About the OpenShift Update Service* .

3 Set the Operator catalog to retrieve the OpenShift Container Platform images from.

4 Specify only certain Operator packages to include in the image set. Remove this field to retrieve all packages in the catalog.

5 Specify any additional images to include in image set.

NOTE

In oc-mirror plugin v2, you must use explicit registry hostnames for all images listed under **additionalImages**. Otherwise, images are mirrored to incorrect target paths.

Additional resources

- About the OpenShift Update Service

5.4.2. Comparison of oc-mirror workflows

Use the following table to compare the supported use cases for the mirror-to-disk (m2d), disk-to-mirror (d2m), and mirror-to-mirror (m2m) workflows.

Use Case	Mirror To Disk (m2d) and Disk To Mirror (d2m)	Mirror To Mirror (m2m)
The target registry exists in an environment with no internet access and no external access.	✓	
The target registry exists in an environment with no internet access but is accessible from another machine. For example, the target registry resides in a bastion host.		✓
You must move content to the disconnected environment by using a physical method, such as USB devices.	✓	
The workflow moves content directly to the target registry without generating an intermediate tar file.		✓
The workflow uses an internal cache to resume after failures but requires additional disk space.	✓	

Use Case	Mirror To Disk (m2d) and Disk To Mirror (d2m)	Mirror To Mirror (m2m)
The workflow does not use a cache, restarts from the beginning after a failure, and requires no additional disk space.		✓

5.4.3. Mirroring an image set in a partially disconnected environment

You can mirror image sets to a registry using the oc-mirror plugin v2 in environments with restricted internet access.

Prerequisites

- You have access to the internet and the mirror registry in the environment where you are running the oc-mirror plugin v2.

Procedure

- Mirror the images from the specified image set configuration to a specified registry by running the following command:

```
$ oc mirror -c <image_set_configuration> --workspace file://<file_path>
docker://<mirror_registry_url> --v2 1
```

where:

<image_set_configuration>
: Specifies the name of the image set configuration file.

<file_path>
: Specifies the directory where cluster resources will be generated in.

<mirror_registry_url>
: Specifies the URL or address of the mirror registry where the images are stored and from which they need to be deleted.

Verification

1. Navigate to the **working-dir/cluster-resources** directory that was generated in the **<file_path>** directory.
2. Verify that the YAML files are present for the **ImageDigestMirrorSet**, **ImageTagMirrorSet**, and **CatalogSource** resources.

Next steps

- Configure your cluster to use the resources generated by oc-mirror plugin v2.

5.4.4. Mirroring an image set in a fully disconnected environment

You can mirror image sets in a fully disconnected environment where the OpenShift Container Platform cluster cannot access the public internet. The following high-level workflow describes the mirroring process:

1. **Mirror to disk**: Mirror the image set to an archive.
2. **Disk transfer**: Manually transfer the archive to the network of the disconnected mirror registry.
3. **Disk to mirror**: Mirror the image set from the archive to the target disconnected registry.

5.4.4.1. Mirroring from mirror to disk

You can use the oc-mirror plugin v2 to generate an image set and save the content to disk. Afterwards, you can transfer the generated image set to the disconnected environment and mirror it to the target registry.

oc-mirror plugin v2 retrieves the container images from the source specified in the image set configuration file and packs them into a tar archive in a local directory.

Procedure

- Mirror the images from the specified image set configuration to the disk by running the following command:

```
$ oc mirror -c <image_set_configuration> file://<file_path> --v2
```

where:

<image_set_configuration>
: Specifies the name of the image set configuration file.

<file_path>
: Specifies the directory where the archives containing the image sets will be generated in.

Verification

1. Navigate to the **<file_path>** directory that was generated.
2. Verify that the archive files have been generated.

Next steps

- Mirroring from disk to mirror

5.4.4.2. Mirroring from disk to mirror

You can use the oc-mirror plugin v2 to mirror image sets from a disk to a target mirror registry.

The oc-mirror plugin v2 retrieves container images from a local disk and transfers them to the specified mirror registry.

Procedure

1. Transfer the disk containing mirrored image sets to the environment that contains the target mirror registry.

2. Process the image set file on the disk and mirror the contents to a target mirror registry by running the following command:

```
$ oc mirror -c <image_set_configuration> --from file://<file_path>
docker://<mirror_registry_url> --v2
```

where:

<image_set_configuration>

Specifies the name of the image set configuration file.

<file_path>

Specifies the directory on the disk containing the archives. This folder will also contain cluster resources generated for you apply to the cluster, for example the ImageDigestMirrorSet (IDMS) or ImageTagMirrorSet (ITMS) resources.

<mirror_registry_url>

Specifies the URL or address of the mirror registry where the images are stored.

Verification

1. Navigate to the **cluster-resources** directory within the **working-dir** directory that was generated in the **<file_path>** directory.

2. Verify that the YAML files are present for the **ImageDigestMirrorSet**, **ImageTagMirrorSet**, and **CatalogSource** resources.

Next steps

- Configure your cluster to use the resources generated by oc-mirror plugin v2.

5.5. ABOUT CUSTOM RESOURCES GENERATED BY OC-MIRROR PLUGIN V2

The oc-mirror plugin v2 automatically generates the following custom resources:

ImageDigestMirrorSet (IDMS)

Handles registry mirror rules when using image digest pull specifications. Generated if at least one image of the image set is mirrored by digest.

ImageTagMirrorSet (ITMS)

Handles registry mirror rules when using image tag pull specifications. Generated if at least one image from the image set is mirrored by tag.

CatalogSource

Retrieves information about the available Operators in the mirror registry. Used by Operator Lifecycle Manager (OLM) Classic.

ClusterCatalog

Retrieves information about the available cluster extensions (which includes Operators) in the mirror registry. Used by OLM v1.

UpdateService

Provides update graph data to the disconnected environment. Used by the OpenShift Update Service.

Additional resources

- CatalogSource
- ImageDigestMirrorSet
- ImageTagMirrorSet
- About catalogs in OLM v1

5.5.1. Restrictions on modifying resources that are generated by the oc-mirror plugin

When using resources that are generated by the oc-mirror plugin v2 to configure your cluster, you must not change certain fields. Modifying these fields can cause errors and is not supported.

The following table lists the resources and their fields that must remain unchanged:

Table 5.1. Fields that must not be modified in the resources generated by oc-mirror

Resource	Fields that must not be changed
CatalogSource	**apiVersion**, **kind**, **spec.image**
ClusterCatalog	**apiVersion**, **kind**, **spec.source.image.ref**
ImageDigestMirrorSet	**apiVersion**, **kind**, **spec.imageDigestMirrors**
ImageTagMirrorSet	**apiVersion**, **kind**, **spec.imageTagMirrors**
Signature **ConfigMap**	**apiVersion**, **kind**, **metadata.namespace**, **binaryData**
UpdateService	**apiVersion**, **kind**, **spec.graphDataImage**, **spec.releases**

For more information about these resources, see the OpenShift API documentation for **CatalogSource**, **ImageDigestMirrorSet**, and **ImageTagMirrorSet**.

5.5.2. Configuring your cluster to use the resources generated by oc-mirror plugin v2

After you have mirrored your image set to the mirror registry, you must apply the generated **ImageDigestMirrorSet** (IDMS), **ImageTagMirrorSet** (ITMS), **CatalogSource**, and **UpdateService** resources to the cluster.

IMPORTANT

In oc-mirror plugin v2, the IDMS and ITMS files cover the entire image set, unlike the **ImageContentSourcePolicy** (ICSP) files in oc-mirror plugin v1. Therefore, the IDMS and ITMS files contain all images of the set even if you only add new images during incremental mirroring.

Prerequisites

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

Procedure

1. Log in to the OpenShift CLI as a user with the **cluster-admin** role.
2. Apply the YAML files from the results directory to the cluster by running the following command:

```
$ oc apply -f <path_to_oc_mirror_workspace>/working-dir/cluster-resources
```

3. If you mirrored release images, apply the release image signatures to the cluster by running the following command:

```
$ oc apply -f working-dir/cluster-resources/signature-configmap.json
```

IMPORTANT

If you are mirroring Operators instead of clusters, do not run the preceding command. Running the command results in an error because there are no release image signatures to apply.

Additionally, a YAML file is available in the same directory **working-dir/cluster-resources/**. You can use either the JSON or YAML format.

Verification

1. Verify that the **ImageDigestMirrorSet** resources are successfully installed by running the following command:

```
$ oc get imagedigestmirrorset
```

2. Verify that the **ImageTagMirrorSet** resources are successfully installed by running the following command:

```
$ oc get imagetagmirrorset
```

3. Verify that the **CatalogSource** resources are successfully installed by running the following command:

```
$ oc get catalogsource -n openshift-marketplace
```

4. Verify that the **ClusterCatalog** resources are successfully installed by running the following command:

```
$ oc get clustercatalog
```

After your cluster is configured to use the resources generated by oc-mirror plugin v2, see Next steps for information about tasks that you can perform using your mirrored images.

Additional resources

- Disconnected environment support in OLM v1

5.6. DELETION OF IMAGES FROM YOUR DISCONNECTED ENVIRONMENT

If you have previously deployed images using the oc-mirror plugin v2, you can delete these images to free up space in your mirror registry. oc-mirror plugin v2 does not automatically prune images that are not included in the **ImageSetConfiguration** file. This prevents accidentally deleting necessary or deployed images when making changes to the **ImageSetConfig.yaml** file.

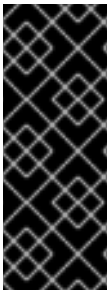
You must create a **DeleteImageSetConfiguration** file to specify which images to delete.

In the following example, the **DeleteImageSetConfiguration** file removes the following images:

- All release images for OpenShift Container Platform 4.13.3.
- The **aws-load-balancer-operator** v0.0.1 bundle and all its related images.
- The additional images for **ubi** and **ubi-minimal**, referenced by their corresponding digests.

Example DeleteImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v2alpha1
kind: DeleteImageSetConfiguration
delete:
  platform:
    channels:
      - name: stable-4.13
        minVersion: 4.13.3
        maxVersion: 4.13.3
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
      packages:
      - name: aws-load-balancer-operator
          minVersion: 0.0.1
          maxVersion: 0.0.1
  additionalImages:
    - name:
registry.redhat.io/ubi8/ubi@sha256:bce7e9f69fb7d4533447232478fd825811c760288f87a35699f9c8f03
0f2c1a6
    - name: registry.redhat.io/ubi8/ubi-
minimal@sha256:8bedbe742f140108897fb3532068e8316900d9814f399d676ac78b46e740e34e
```



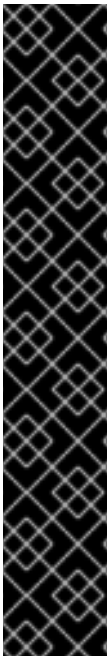
IMPORTANT

- Consider using the mirror-to-disk and disk-to-mirror workflows to reduce deletion issues.
- In oc-mirror plugin v2, you must use explicit registry hostnames for all images listed under **additionalImages**. Otherwise, images are mirrored to incorrect target paths.

oc-mirror plugin v2 deletes only the manifests of the images, which does not reduce the storage occupied in the registry.

To free up storage space from unnecessary images, such as those with deleted manifests, you must enable the garbage collector on your container registry. With the garbage collector enabled, the registry will delete the image blobs that no longer have references to any manifests, reducing the storage previously occupied by the deleted blobs. The process for enabling the garbage collector differs depending on your container registry.

For more information, see "Resolving storage cleanup issues in the distribution registry".



IMPORTANT

- To skip deleting the Operator catalog image while you are deleting Operator images, you must list the specific Operators under the Operator catalog image in the **DeleteImageSetConfiguration** file. This ensures that only the specified Operators are deleted, not the catalog image.
  If only the Operator catalog image is specified, all Operators within that catalog, as well as the catalog image itself, will be deleted.
- oc-mirror plugin v2 does not delete Operator catalog images automatically because other Operators may still be deployed and depend on these images. If you are certain that no Operators from a catalog remain in the registry or cluster, you can explicitly add the catalog image to **additionalImages** in **DeleteImageSetConfiguration** to remove it.
- Garbage collection behavior depends on the registry. Some registries do not automatically remove deleted images, requiring a system administrator to manually trigger garbage collection to free up space.

Additional resources

- Resolving storage cleanup issues in the distribution registry

5.6.1. Resolving storage cleanup issues in the distribution registry

A known issue in the distribution registry prevents the garbage collector from freeing up storage as expected. This issue does not occur when you use Red Hat Quay.

Procedure

- Choose the appropriate method to work around the known issue in the distribution registry:
  - To restart the container registry, run the following command:

```
$ podman restart <registry_container>
```

- To disable caching in the registry configuration, perform the following steps:

  i. To disable the **blobdescriptor** cache, modify the **/etc/docker/registry/config.yml** file:

```
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: ""
  filesystem:
    rootdirectory: /var/lib/registry
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

  ii. To apply the changes, restart the container registry by running the following command:

```
$ podman restart <registry_container>
```

5.6.2. Deleting images from a disconnected environment

To delete images from a disconnected environment using the oc-mirror plugin v2, follow the procedure.

Prerequisites

- You have enabled garbage collection in your environment to delete images that no longer reference manifests.

Procedure

1. Create a **delete-image-set-config.yaml** file and include the following content:

   DeleteImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v2alpha1
kind: DeleteImageSetConfiguration
delete:
  platform:
    channels:
      - name: <channel_name> 1
        minVersion: <channel_min_version> 2
        maxVersion: <channel_max_version> 3
  operators:
    - catalog: <operator_catalog_name> 4
      packages:
```

```
    - name: <operator_name> 5
      minVersion: <operator_max_version> 6
      maxVersion: <operator_min_version> 7
additionalImages:
 - name: <additional_images>
```

1 1 Specify the name of the OpenShift Container Platform channel to delete, for example **stable-4.15**.

2 3 Specify a version range of the images to delete within the channel, for example **4.15.0** for the minimum version and **4.15.1** for the maximum version. To delete only one version's images, use that version number for both the **minVersion** and **maxVersion** fields.

4 Specify an Operator catalog image containing the Operators to delete, for example **registry.redhat.io/redhat/redhat-operator-index:v4.14**. The Operator catalog image will not get deleted. Its presence in the registry might be necessary for other Operators still remaining on the cluster.

5 Specify a specific Operator to delete, for example **aws-load-balancer-operator**.

6 7 Specify a version range of the images to delete for the Operator, for example **0.0.1** for the minimum version and **0.0.2** for the maximum version.

2. Create a **delete-images.yaml** file by running the following command:

```
$ oc-mirror delete --config delete-image-set-config.yaml --workspace
file://<previously_mirrored_work_folder> --v2 --generate docker://<remote_registry>
```

where:

<previously_mirrored_work_folder>

Specifies the directory where images were previously mirrored to or stored during the mirroring process.

<remote_registry>

Specifies the URL or address of the remote container registry from which images will be deleted.

IMPORTANT

When deleting images, specify the correct workspace directory. Modify or delete the cache directory only when starting mirroring from scratch, such as setting up a new cluster. Incorrect changes to the cache directory might disrupt further mirroring operations.

3. Go to the **<previously_mirrored_work_folder>/delete** directory that was created.

4. Verify that the **delete-images.yaml** file has been generated.

5. Manually ensure that each image listed in the file is no longer needed by the cluster and can be safely removed from the registry.

6. After you generate the **delete-images** YAML file, delete the images from the remote registry by running the following command:

```
$ oc-mirror delete --v2 --delete-yaml-file <previously_mirrored_work_folder>/working-dir/delete/delete-images.yaml docker://<remote_registry>
```

where:

<previously_mirrored_work_folder>

Specifies the directory where images were previously mirrored or stored during the mirroring process.

<remote_registry>

Specifies the URL or address of the remote container registry from which images will be deleted.

IMPORTANT

When using the mirror-to-mirror method to mirror images, images are not cached locally, so you cannot delete images from a local cache.

5.7. VERIFYING YOUR SELECTED IMAGES FOR MIRRORING

You can use oc-mirror plugin v2 to perform a test run (dry run) that does not actually mirror any images. This enables you to review the list of images that would be mirrored. You can also use a dry run to catch any errors with your image set configuration early. When running a dry run on a mirror-to-disk workflow, the oc-mirror plugin v2 checks if all the images within the image set are available in its cache. Any missing images are listed in the **missing.txt** file. When a dry run is performed before mirroring, both **missing.txt** and **mapping.txt** files contain the same list of images.

5.7.1. Performing a dry run for oc-mirror plugin v2

Verify your image set configuration by performing a dry run without mirroring any images. This ensures your setup is correct and prevents unintended changes.

Procedure

- To perform a test run, run the **oc mirror** command and append the **--dry-run** argument to the command:

```
$ oc mirror -c <image_set_config_yaml> file://<oc_mirror_workspace_path> --dry-run --v2
```

where:

<image_set_config_yaml>

Specifies the image set configuration file that you created.

<oc_mirror_workspace_path>

Insert the address of the workspace path.

<mirror_registry_url>

Insert the URL or address of the remote container registry from which images will be mirrored or deleted.

Example output

```
[INFO]   : :wave: Hello, welcome to oc-mirror
[INFO]   : :gear:  setting up the environment for you...
[INFO]   : :twisted_rightwards_arrows: workflow mode: mirrorToDisk
[INFO]   : :sleuth_or_spy:  going to discover the necessary images...
[INFO]   : :mag: collecting release images...
[INFO]   : :mag: collecting operator images...
[INFO]   : :mag: collecting additional images...
[WARN]   : :warning:  54/54 images necessary for mirroring are not available in the cache.
[WARN]   : List of missing images in : CLID-19/working-dir/dry-run/missing.txt.
please re-run the mirror to disk process
[INFO]   : :page_facing_up: list of all images for mirroring in : CLID-19/working-dir/dry-run/mapping.txt
[INFO]   : mirror time     : 9.641091076s
[INFO]   : :wave: Goodbye, thank you for using oc-mirror
```

Verification

1. Navigate to the workspace directory that was generated:

```
$ cd <oc_mirror_workspace_path>
```

2. Review the **mapping.txt** and **missing.txt** files that were generated. These files contain a list of all images that would be mirrored.

5.7.2. Troubleshooting oc-mirror plugin v2 errors

oc-mirror plugin v2 now logs all image mirroring errors in a separate file, making it easier to track and diagnose failures.

IMPORTANT

When errors occur while mirroring release or release component images, they are critical. This stops the mirroring process immediately.

Errors with mirroring Operators, Operator-related images, or additional images do not stop the mirroring process. Mirroring continues, and oc-mirror plugin v2 saves a file under the **working-dir/logs** directory describing which Operator failed to mirror.

When an image fails to mirror, and that image is mirrored as part of one or more Operator bundles, oc-mirror plugin v2 notifies the user which Operators are incomplete, providing clarity on the Operator bundles affected by the error.

Procedure

1. Check for server-related issues:

Example error

```
[ERROR]  : [Worker] error mirroring image localhost:55000/openshift/graph-image:latest
error: copying image 1/4 from manifest list: trying to reuse blob
sha256:edab65b863aead24e3ed77cea194b6562143049a9307cd48f86b542db9eecb6e at
destination: pinging container registry localhost:5000: Get "https://localhost:5000/v2/": http:
server gave HTTP response to HTTPS client
```

a. Open the **mirroring_error_date_time.log** file in the **working-dir/logs** folder located in the oc-mirror plugin v2 output directory.

b. Look for error messages that typically indicate server-side issues, such as **HTTP 500** errors, expired tokens, or timeouts.

c. Retry the mirroring process or contact support if the issue persists.

2. Check for incomplete mirroring of Operators:

Example error

```
error mirroring image docker://registry.redhat.io/3scale-amp2/zync-
rhel9@sha256:8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d
(Operator bundles: [3scale-operator.v0.11.12] - Operators: [3scale-operator]) error: initializing
source docker://localhost:55000/3scale-amp2/zync-
rhel9:8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d: reading
manifest 8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d in
localhost:55000/3scale-amp2/zync-rhel9: manifest unknown
error mirroring image docker://registry.redhat.io/3scale-amp2/3scale-rhel7-operator-
metadata@sha256:de0a70d1263a6a596d28bf376158056631afd0b6159865008a7263a8e9bf0
c7d error: skipping operator bundle docker://registry.redhat.io/3scale-amp2/3scale-rhel7-
operator-
metadata@sha256:de0a70d1263a6a596d28bf376158056631afd0b6159865008a7263a8e9bf0
c7d because one of its related images failed to mirror
error mirroring image docker://registry.redhat.io/3scale-amp2/system-
rhel7@sha256:fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e
(Operator bundles: [3scale-operator.v0.11.12] - Operators: [3scale-operator]) error: initializing
source docker://localhost:55000/3scale-amp2/system-
rhel7:fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e: reading
manifest fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e in
localhost:55000/3scale-amp2/system-rhel7: manifest unknown
```

a. Check for warnings in the console or log file indicating which Operators are incomplete. If an Operator is flagged as incomplete, the image related to that Operator likely failed to mirror.

b. Manually mirror the missing image or retry the mirroring process.

3. Check for errors related to generated cluster resources. Even if some images fail to mirror, oc-mirror v2 will still generate cluster resources such as **IDMS.yaml** and **ITMS.yaml** files for the successfully mirrored images.

a. Check the output directory for the generated files.

b. If these files are missing for specific images, ensure that no critical errors occurred for those images during the mirroring process.

By following these steps, you can better diagnose issues and ensure smoother mirroring.

5.8. BENEFITS OF ENCLAVE SUPPORT

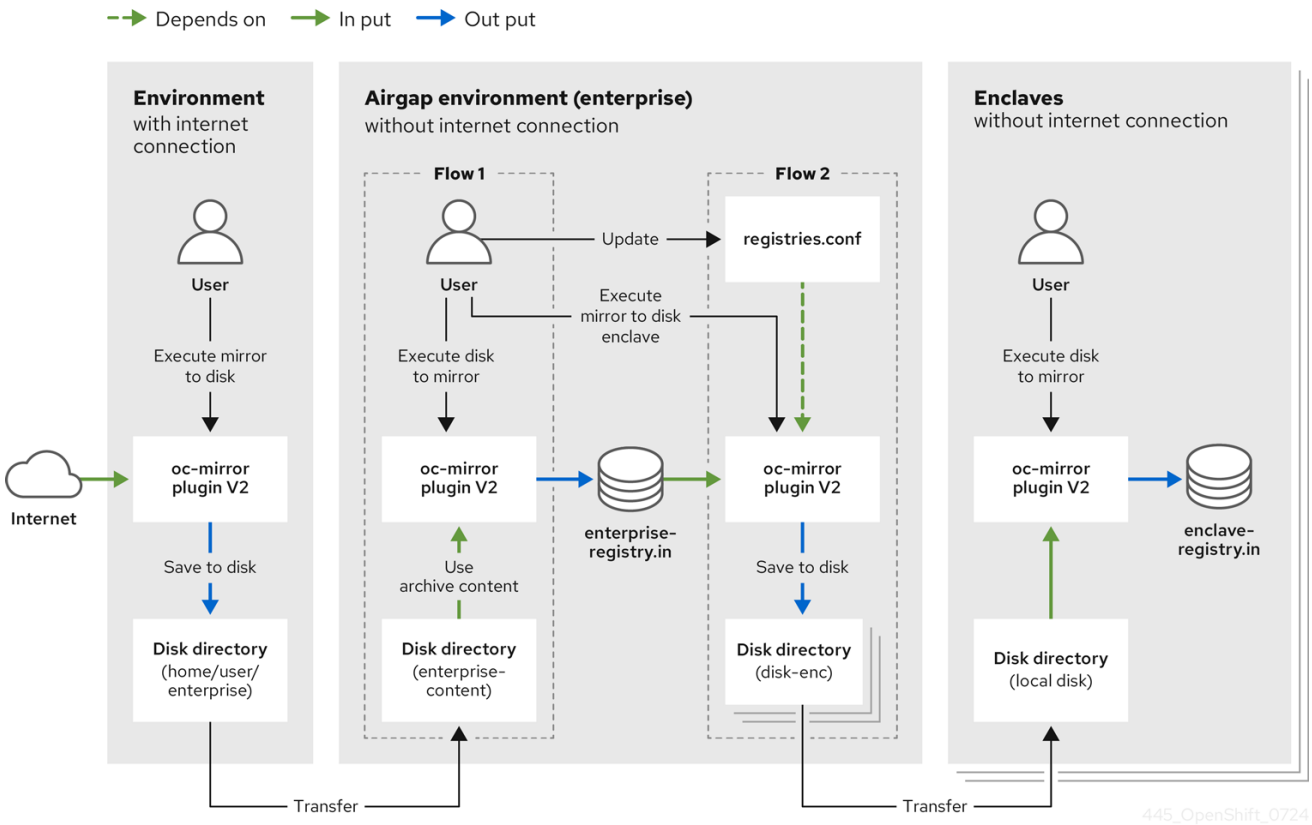
Enclave support restricts internal access to a specific part of a network. Unlike a demilitarized zone (DMZ) network, which allows inbound and outbound traffic access through firewall boundaries, enclaves do not cross firewall boundaries.

The new enclave support functionality is for scenarios where mirroring is needed for multiple enclaves that are secured behind at least one intermediate disconnected network.

Enclave support has the following benefits:

- You can mirror content for multiple enclaves and centralize it in a single internal registry. Because some customers want to run security checks on the mirrored content, with this setup they can run these checks all at once. The content is then vetted before being mirrored to downstream enclaves.
- You can mirror content directly from the centralized internal registry to enclaves without restarting the mirroring process from the internet for each enclave.
- You can minimize data transfer between network stages, so to ensure that a blob or image is transferred only once from one stage to another.

5.8.1. Enclave mirroring workflow



The previous image outlines the flow for using the oc-mirror plugin in different environments, including environments with and without an internet connection.

Environment with Internet Connection:

1. The user executes oc-mirror plugin v2 to mirror content from an online registry to a local disk directory.

2. The mirrored content is saved to the disk for transfer to offline environments.

Disconnected Enterprise Environment (No Internet)

- Flow 1:
  - The user runs oc-mirror plugin v2 to load the mirrored content from the disk directory, which was transferred from the online environment, into the **enterprise-registry.in** registry.
- Flow 2:
  a. After updating the **registries.conf** file, the user executes the oc-mirror plugin v2 to mirror content from the **enterprise-registry.in** registry to an enclave environment.
  b. The content is saved to a disk directory for transfer to the enclave.

Enclave Environment (No Internet)

- The user runs oc-mirror plugin v2 to load content from the disk directory into the **enclave-registry.in** registry.

The image visually represents the data flow across these environments and emphasizes the use of oc-mirror to handle disconnected and enclave environments without an internet connection.

5.8.2. Mirroring to an enclave

When you mirror to an enclave, you must first transfer the necessary images from one or more enclaves into the enterprise central registry.

The central registry is situated within a secure network, specifically a disconnected environment, and is not directly linked to the public internet. But the user must execute **oc mirror** in an environment with access to the public internet.

Procedure

1. Before running oc-mirror plugin v2 in the disconnected environment, create a **registries.conf** file. The TOML format of the file is described in this specification:

NOTE

It is recommended to store the file under **$HOME/.config/containers/registries.conf** or **/etc/containers/registries.conf**.

Example registries.conf

```
[[registry]]
location="registry.redhat.io"
[[registry.mirror]]
location="<enterprise-registry.in>"

[[registry]]
location="quay.io"
[[registry.mirror]]
location="<enterprise-registry.in>"
```

2. Generate a mirror archive.

   a. To collect all the OpenShift Container Platform content into an archive on the disk under **<file_path>/enterprise-content**, run the following command:

   ```
   $ oc mirror --v2 -c isc.yaml file://<file_path>/enterprise-content
   ```

   Example of isc.yaml

   ```
   apiVersion: mirror.openshift.io/v2alpha1
   kind: ImageSetConfiguration
   mirror:
     platform:
       architectures:
         - "amd64"
       channels:
         - name: stable-4.15
           minVersion: 4.15.0
           maxVersion: 4.15.3
   ```

   After the archive is generated, it is transferred to the disconnected environment. The transport mechanism is not part of oc-mirror plugin v2. The enterprise network administrators determine the transfer strategy.

   In some cases, the transfer is done manually, in that the disk is physically unplugged from one location, and plugged to another computer in the disconnected environment. In other cases, the Secure File Transfer Protocol (SFTP) or other protocols are used.

3. After the transfer of the archive is done, you can execute oc-mirror plugin v2 again in order to mirror the relevant archive contents to the registry (**entrerpise_registry.in** in the example) as demonstrated in the following example:

   ```
   $ oc mirror --v2 -c isc.yaml --from file://<disconnected_environment_file_path>/enterprise-content docker://<enterprise_registry.in>/
   ```

   Where:

   - **--from** points to the folder containing the archive. It starts with the **file:**//.
   - **docker:**// is the destination of the mirroring is the final argument. Because it is a docker registry.
   - **-c** (**--config**) is a mandatory argument. It enables oc-mirror plugin v2 to eventually mirror only sub-parts of the archive to the registry. One archive might contain several OpenShift Container Platform releases, but the disconnected environment or an enclave might mirror only a few.

4. Prepare the **imageSetConfig** YAML file, which describes the content to mirror to the enclave:

   Example isc-enclave.yaml

   ```
   apiVersion: mirror.openshift.io/v2alpha1
   kind: ImageSetConfiguration
   mirror:
     platform:
   ```

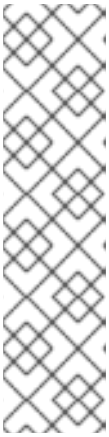
```
    architectures:
      - "amd64"
    channels:
      - name: stable-4.15
        minVersion: 4.15.2
        maxVersion: 4.15.2
```

You must run oc-mirror plugin v2 on a machine with access to the disconnected registry. In the previous example, the disconnected environment, **enterprise-registry.in**, is accessible.

5. Update the graph URL
   If you are using **graph:true**, oc-mirror plugin v2 attempts to reach the **cincinnati** API endpoint. Because this environment is disconnected, be sure to export the environment variable **UPDATE_URL_OVERRIDE** to refer to the URL for the OpenShift Update Service (OSUS):

```
$ export UPDATE_URL_OVERRIDE=https://<osus.enterprise.in>/graph
```

For more information on setting up OSUS on an OpenShift cluster, see "Updating a cluster in a disconnected environment using the OpenShift Update Service".



NOTE

When updating between OpenShift Container Platform Extended Update Support (EUS) versions, you must also include images for an intermediate minor version between the current and target versions. The oc-mirror plugin v2 might not always detect this requirement automatically, so check the Red Hat OpenShift Container Platform Update Graph page to confirm any required intermediate versions.

Use the Update Graph page to find the intermediate minor versions suggested by the application, and include any of these versions in the **ImageSetConfiguration** file when using the oc-mirror plugin v2.

6. Generate a mirror archive from the enterprise registry for the enclave.
   To prepare an archive for the **enclave1**, the user executes oc-mirror plugin v2 in the enterprise disconnected environment by using the **imageSetConfiguration** specific for that enclave. This ensures that only images needed by that enclave are mirrored:

```
$ oc mirror --v2 -c isc-enclave.yaml
file:///disk-enc1/
```

This action collects all the OpenShift Container Platform content into an archive and generates an archive on disk.

7. After the archive is generated, it will be transferred to the **enclave1** network. The transport mechanism is not the responsibility of oc-mirror plugin v2.

8. Mirror contents to the enclave registry
   After the transfer of the archive is done, the user can execute oc-mirror plugin v2 again in order to mirror the relevant archive contents to the registry.

```
$ oc mirror --v2 -c isc-enclave.yaml --from file://local-disk docker://registry.enc1.in
```

The administrators of the OpenShift Container Platform cluster in **enclave1** are now ready to install or upgrade that cluster.

5.9. OC-MIRROR PLUGIN V2 SUPPORT PROXY SETTING

The oc-mirror plugin v2 can operate in a proxy-configured environment. The plugin can use system proxy settings to retrieve images for OpenShift Container Platform, Operator catalog, and the **additionalImages** registry.

Additional resources

- Updating a cluster in a disconnected environment using the OpenShift Update Service
- Resolving storage cleanup issues in the distribution registry

5.10. HOW FILTERING WORKS IN THE OPERATOR CATALOG

oc-mirror plugin v2 selects the list of bundles for mirroring by processing the information in **imageSetConfig**.

When oc-mirror plugin v2 selects bundles for mirroring, it does not infer Group Version Kind (GVK) or bundle dependencies, omitting them from the mirroring set. Instead, it strictly adheres to the user instructions. You must explicitly specify any required dependent packages and their versions.

Table 5.2. Use the following table to see what bundle versions are included in different scenarios

ImageSetConfig operator filtering	Expected bundle versions

Scenario 1
`mirror:`
`operators:`
`- catalog:`
`registry.redhat.io/redhat/redhat-operator-index:v4.10`
	For each package in the catalog, one bundle, corresponding to the head version for each channel of that package.

Scenario 2
`mirror:`
`operators:`
`- catalog:`
`registry.redhat.io/redhat/redhat-operator-index:v4.10`
`full: true`
	All bundles of all channels of the specified catalog.

ImageSetConfig operator filtering	Expected bundle versions
Scenario 3 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 packages: - name: compliance-operator	One bundle, corresponding to the head version for each channel of that package.
Scenario 4 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator	All bundles of all channels for the packages specified.
Scenario 5 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator minVersion: 5.6.0	All bundles in all channels, from the **minVersion**, up to the channel head for that package.
Scenario 6 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator maxVersion: 6.0.0	All bundles in all channels that are lower than the **maxVersion** for that package.

ImageSetConfig operator filtering	Expected bundle versions
Scenario 7 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator minVersion: 5.6.0 maxVersion: 6.0.0	All bundles in all channels, between the **minVersion** and **maxVersion** for that package. The head of the channel is not included, even if multiple channels are included in the filtering.
Scenario 8 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable	The head bundle for the selected channel of that package. You must use the **defaultChannel** field in case the filtered channels are not the default.
Scenario 9 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator channels: - name: 'stable-v0'	All bundles for the packages and channels specified. The **defaultChannel** should be used in case the filtered channels are not the default.

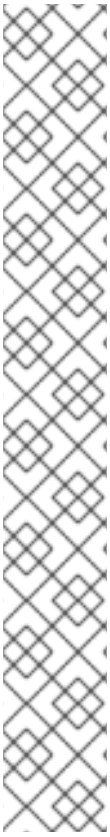
ImageSetConfig operator filtering	Expected bundle versions
Scenario 10 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable - name: stable-5.5	The head bundle for each selected channel of that package.
Scenario 11 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0	Within the selected channel of that package, all versions starting with the **minVersion** up to the channel head. You must use the **defaultChannel** field in case the filtered channels are not the default.
Scenario 12 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable maxVersion: 6.0.0	Within the selected channel of that package, all versions up to **maxVersion**. Head of channel is not included, even if multiple channels are included in the filtering. You might see errors if this filtering leads to a channel with multiple heads. You must use the **defaultChannel** field in case the filtered channels are not the default.

ImageSetConfig operator filtering	Expected bundle versions
Scenario 13 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	Within the selected channel of that package, all versions between the **minVersion** and **maxVersion**. The head of channel is not included, even if multiple channels are included in the filtering. You might see errors if this filtering leads to a channel with multiple heads. You must use the **defaultChannel** field in case the filtered channels are not the default.
Scenario 14 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	Do not use this scenario. filtering by channel and by package with a **minVersion** or **maxVersion** is not allowed.
Scenario 15 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	Do not use this scenario. You cannot filter using **full:true** and the **minVersion** or **maxVersion**.

ImageSetConfig operator filtering	Expected bundle versions
Scenario 16 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 full: true packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	Do not use this scenario. You cannot filter using **full:true** and the **minVersion** or **maxVersion**.

5.11. IMAGESET CONFIGURATION PARAMETERS FOR OC-MIRROR PLUGIN V2

The oc-mirror plugin v2 requires an image set configuration file that defines what images to mirror. The following table lists the available parameters for the **ImageSetConfiguration** resource.



NOTE

- When selecting bundles for mirroring, the oc-mirror plugin v2 does not automatically detect group/version/kind (GVK) and bundle dependencies. You must explicitly specify the required Operators, their channels, and the Operator versions in the **ImageSetConfiguration** file. For more information, see "opm CLI reference".
- Using the **minVersion** and **maxVersion** properties to filter for a specific Operator version range can result in a multiple channel heads error. The error message states that there are **multiple channel heads**. This is because when the filter is applied, the update graph of the Operator is truncated.
- OLM requires that every Operator channel contains versions that form an update graph with exactly one end point, that is, the latest version of the Operator. When the filter range is applied, that graph can turn into two or more separate graphs or a graph that has more than one end point.
- To avoid this error, do not filter out the latest version of an Operator. If you still run into the error, depending on the Operator, either the **maxVersion** property must be increased or the **minVersion** property must be decreased. Because every Operator graph can be different, you might need to adjust these values until the error resolves.

Table 5.3. **ImageSetConfiguration** parameters

Parameter	Description	Values
apiVersion	The API version of the **ImageSetConfiguration** content.	String Example: **mirror.openshift.io/v2alpha1**
archiveSize	The maximum size, in GiB, of each archive file within the image set.	Integer Example: **4**
kubeVirtContainer	When set to **true**, includes images from the HyperShift KubeVirt CoreOS container.	Boolean Example **ImageSetConfiguration** file: apiVersion: mirror.openshift.io/v2alpha1 kind: ImageSetConfiguration mirror: platform: channels: - name: stable-4.16 minVersion: 4.16.0 maxVersion: 4.16.0 kubeVirtContainer: true
mirror	The configuration of the image set.	Object
mirror.additionalImages	The additional images configuration of the image set.	Array of objects Example: additionalImages: - name: registry.redhat.io/ubi8/ubi:latest
mirror.additionalImages.name	The tag or digest of the image to mirror.	String Example: **registry.redhat.io/ubi8/ubi:latest**

Parameter	Description	Values
mirror.blockedImages	List of images with a tag or digest (SHA) to block from mirroring.	Array of strings Example: **docker.io/library/alpine**
mirror.helm	The helm configuration of the image set. The oc-mirror plugin does not support helm charts with manually modified **values.yaml** files.	Object

mirror.helm.local	The local helm charts to mirror.	Array of objects. For example:
		`local:`
		`  - name: podinfo`
		`  path: /test/podinfo-5.0.0.tar.gz`

mirror.helm.local.name	The name of the local helm chart to mirror.	String. For example: **podinfo**.
mirror.helm.local.path	The path of the local helm chart to mirror.	String. For example: **/test/podinfo-5.0.0.tar.gz**.

mirror.helm.repositories	The remote helm repositories to mirror from.	Array of objects. For example:
		`repositories:`
		`  - name: podinfo`
		`  url: https://example.github.io/podinfo`
		`  charts:`
		`  - name: podinfo`
		`  version: 5.0.0`
		
		`imagePaths:`
		`  - "{.spec.template.spec.custom[*].image}"`

Parameter	Description	Values
mirror.helm.repositories.name	The name of the helm repository to mirror from.	String. For example: **podinfo**.
mirror.helm.repositories.url	The URL of the helm repository to mirror from.	String. For example: **https://example.github.io/podinfo**.
mirror.helm.repositories.charts	The remote helm charts to mirror.	Array of objects.
mirror.helm.repositories.charts.name	The name of the helm chart to mirror.	String. For example: **podinfo**.
mirror.helm.repositories.charts.imagePaths	The custom path of a container image inside of the helm chart. + NOTE **oc-mirror** detects and mirrors container images from the helm chart by searching well-known paths. You can also specify custom paths using this field.	Array of string. For example: **"-{.spec.template.spec.custom[*].image}"**.
mirror.operators	The Operators configuration of the image set.	Array of objects Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.18 packages: - name: elasticsearch-operator minVersion: '2.4.0'

Parameter	Description	Values
mirror.operators.catalog	The Operator catalog to include in the image set.	String Example: **registry.redhat.io/redhat/redhat-operator-index:v4.15**
mirror.operators.full	When **true**, downloads the full catalog, Operator package, or Operator channel.	Boolean The default value is **false**.
mirror.operators.packages	The Operator packages configuration.	Array of objects Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.18 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	The Operator package name to include in the image set.	String Example: **elasticsearch-operator**
mirror.operators.packages.channels	Operator package channel configuration	Object
mirror.operators.packages.channels.name	The Operator channel name, unique within a package, to include in the image set.	String Eample: **fast** or **stable-v4.15**
mirror.operators.packages.channels.maxVersion	The highest version of the Operator mirror across all channels in which it exists.	String Example: **5.2.3-31**
mirror.operators.packages.channels.minVersion	The lowest version of the Operator to mirror across all channels in which it exists	String Example: **5.2.3-31**
mirror.operators.packages.maxVersion	The highest version of the Operator to mirror across all channels in which it exists.	String Example: **5.2.3-31**

Parameter	Description	Values
mirror.operators.packages.minVersion	The lowest version of the Operator to mirror across all channels in which it exists.	String Example: **5.2.3-31**
mirror.operators.targetCatalog	An alternative name and optional namespace hierarchy to mirror the referenced catalog as	String Example: **my-namespace/my-operator-catalog**

mirror.operators.targetCatalogSourceTemplate	Path on disk for a template to use to complete catalogSource custom resource generated by oc-mirror plugin v2.	String Example: **/tmp/catalog-source_template.yaml** Example of a template file:
		`apiVersion: operators.coreos.com/v1alpha1`
		`kind: CatalogSource`
		`metadata:`
		`  name: discarded`
		`  namespace: openshift-marketplace`
		`spec:`
		`  image: discarded`
		`  sourceType: grpc`
		
		`updateStrategy:`
		
		`registryPoll:`
		`  interval: 30m0s`

mirror.operators.targetTag	An alternative tag to append to the **targetName** or **targetCatalog**.	String Example: **v1**
mirror.platform	The platform configuration of the image set.	Object

Parameter	Description	Values

mirror.platform.architectures	The architecture of the platform release payload to mirror.	Array of strings Example:
		`architectures:`
		`- amd64`
		`- arm64`
		`- multi`
		`- ppc64le`
		`- s390x`
		The default value is **amd64**. The value **multi** ensures that the mirroring is supported for all available architectures, eliminating the need to specify individual architectures

mirror.platform.channels	The platform channel configuration of the image set.	Array of objects Example:
		`channels:`
		`- name: stable-4.12`
		`- name: stable-4.18`

mirror.platform.channels.full	When **true**, sets the **minVersion** to the first release in the channel and the **maxVersion** to the last release in the channel.	Boolean The default value is **false**
mirror.platform.channels.name	Name of the release channel	String Example: **stable-4.15**
mirror.platform.channels.minVersion	The minimum version of the referenced platform to be mirrored.	String Example: **4.12.6**
mirror.platform.channels.maxVersion	The highest version of the referenced platform to be mirrored.	String Example: **4.15.1**

Parameter	Description	Values
mirror.platform.channels.shortestPath	Toggles shortest path mirroring or full range mirroring.	Boolean The default value is **false**
mirror.platform.channels.type	Type of the platform to be mirrored	String Example: **ocp** or **okd**. The default is **ocp**.
mirror.platform.graph	Indicates whether the OSUS graph is added to the image set and subsequently published to the mirror.	Boolean The default value is **false**

mirror.operators.packages.defaultChannel	Must be defined when excluding the default channel from the filtering.	Array of objects. For example:
		`mirror:`
		`operators:`
		`- catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19`
		`packages:`
		`- name: rhods-operator`
		`defaultChannel: fast`
		`channels:`
		`- name: fast`

5.11.1. DeleteImageSetConfiguration parameters

To use remove images with the oc-mirror plugin v2, you must use a **DeleteImageSetConfiguration.yaml** configuration file that defines which images to delete from the mirror registry. The following table lists the available parameters for the **DeleteImageSetConfiguration** resource.

Table 5.4. **DeleteImageSetConfiguration** parameters

Parameter	Description	Values
apiVersion	The API version for the **DeleteImageSetConfiguration** content.	String Example: **mirror.openshift.io/v2alpha1**

Parameter	Description	Values
delete	The configuration of the image set to delete.	Object

delete.additionalImages	The additional images configuration of the delete image set.	Array of objects
		Example:
		`additionalImages:`
		`  - name: registry.redhat.io/ubi8/ubi:latest`

delete.additionalImages.name	The tag or digest of the image to delete.	String Example: **registry.redhat.io/ubi8/ubi:latest**

delete.operators	The Operators configuration of the delete image set.	Array of objects
		Example:
		`operators:`
		`  - catalog: registry.redhat.io/redhat/redhat-operator-index: {product-version}`
		`    packages:`
		`     - name: elasticsearch-operator`
		`     minVersion: '2.4.0'`

delete.operators.catalog	The Operator catalog to include in the delete image set.	String Example: **registry.redhat.io/redhat/redhat-operator-index:v4.15**
delete.operators.full	When true, deletes the full catalog, Operator package, or Operator channel.	Boolean The default value is **false**

Parameter	Description	Values

delete.operators.packages	Operator packages configuration	Array of objects
		Example:
		`operators:`
		`  - catalog: registry.redhat.io/redhat/redhat-operator-index:{product-version}`
		`  packages:`
		`  - name: elasticsearch-operator`
		`  minVersion: '5.2.3-31'`

delete.operators.packages.name	The Operator package name to include in the delete image set.	String Example: **elasticsearch-operator**
delete.operators.packages.channels	Operator package channel configuration	Object
delete.operators.packages.channels.name	The Operator channel name, unique within a package, to include in the delete image set.	String Example: **fast** or **stable-v4.15**
delete.operators.packages.channels.maxVersion	The highest version of the Operator to delete within the selected channel.	String Example: **5.2.3-31**
delete.operators.packages.channels.minVersion	The lowest version of the Operator to delete within the selection in which it exists.	String Example: **5.2.3-31**
delete.operators.packages.maxVersion	The highest version of the Operator to delete across all channels in which it exists.	String Example: **5.2.3-31**
delete.operators.packages.minVersion	The lowest version of the Operator to delete across all channels in which it exists.	String Example: **5.2.3-31**
delete.platform	The platform configuration of the image set	Object

Parameter	Description	Values

delete.platform.architectures	The architecture of the platform release payload to delete.	Array of strings
		Example:
		`architectures:`
		`- amd64`
		`- arm64`
		`- multi`
		`- ppc64le`
		`- s390x`
		The default value is **amd64**

delete.platform.channels	The platform channel configuration of the image set.	Array of objects
		Example:
		`channels:`
		`- name: stable-4.12`
		`- name: stable-4.18`

delete.platform.channels.full	When **true**, sets the **minVersion** to the first release in the channel and the **maxVersion** to the last release in the channel.	Boolean The default value is **false**
delete.platform.channels.name	Name of the release channel	String Example: **stable-4.15**
delete.platform.channels.minVersion	The minimum version of the referenced platform to be deleted.	String Example: **4.12.6**
delete.platform.channels.maxVersion	The highest version of the referenced platform to be deleted.	String Example: **4.15.1**
delete.platform.channels.shortestPath	Toggles between deleting the shortest path and deleting the full range.	Boolean The default value is **false**
delete.platform.channels.type	Type of the platform to be deleted	String Example: **ocp** or **okd** The default is **ocp**

Parameter	Description	Values
delete.platform.graph	Determines whether the OSUS graph is deleted as well on the mirror registry as well.	Boolean The default value is **false**

Additional resources

- opm CLI reference

5.12. COMMAND REFERENCE FOR OC-MIRROR PLUGIN V2

The following tables describe the **oc mirror** subcommands and flags for oc-mirror plugin v2:

Table 5.5. Subcommands and flags for the oc-mirror plugin v2

Subcommand	Description
help	Show help about any subcommand
version	Output the oc-mirror version
delete	Deletes images in remote registry and local cache.

Table 5.6. oc mirror flags

Flag	Description
--authfile	Displays the string path of the authentication file. Default is **${XDG_RUNTIME_DIR}/containers/auth.json**.
-c, **--config <string>**	Specifies the path to an image set configuration file.
--cache-dir <string>	oc-mirror cache directory location. The default value is **$HOME**.
--dest-tls-verify	Requires HTTPS and verifies certificates when accessing the container registry or daemon. The default value is **true**.
--dry-run	Prints actions without mirroring images.
--from <string>	Specifies the path to an image set archive that was generated by executing oc-mirror plugin v2 to load a target registry.
-h, **--help**	Displays help
--image-timeout duration	Timeout for mirroring an image. The default is 10m0s. Valid time units are **ns**, **us** or **µs**, **ms**, **s**, **m**, and **h**.

Flag	Description
--log-level <string>	Displays string log levels. Supported values include info, debug, trace, error. The default value is **info**.
-p, **--port**	Determines the HTTP port used by oc-mirror plugin v2 local storage instance. The default value is **55000**.
--parallel-images <unit>	Specifies the number of images mirrored in parallel. The default value is **4**.
--parallel-layers <unit>	Specifies the number of image layers mirrored in parallel. The default value is **5**.
--max-nested-paths <int>	Specifies the maximum number of nested paths for destination registries that limit nested paths. The default value is **0**.
--secure-policy	The default value is **false**. If you set a non-default value, the command enables signature verification, which is the secure policy for signature verification.
--since	Includes all new content since a specified date (format: **yyyy-mm-dd**). When not provided, new content since previous mirroring is mirrored.
--src-tls-verify	Requires HTTPS and verifies certificates when accessing the container registry or daemon.
--strict-archive	The default value is **false**. If you set a value, the command generates archives that are strictly less than the **archiveSize** that was set in the **imageSetConfig** custom resource (CR). Mirroring exist in error if a file being archived exceeds **archiveSize** (GB).
-v, **--version**	Displays the version for oc-mirror plugin v2.
--workspace	Determines string oc-mirror plugin v2 workspace where resources and internal artifacts are generated.
--retry-delay duration	Delay between 2 retries. The default value is **1s**.
--retry-times <int>	The number of times to retry. The default value is **2**.
--rootless-storage-path <string>	Overrides the default container rootless storage path (usually in **etc/containers/storage.conf**).

5.12.1. Command reference for deleting images

The following tables describe the **oc mirror** subcommands and flags for deleting images:

Table 5.7. Subcommands and flags for the deleting images

Subcommand	Description
--authfile <string>	Path of the authentication file. The default value is **${XDG_RUNTIME_DIR}/containers/auth.json**.
--cache-dir <string>	oc-mirror cache directory location. The default is **$HOME**.
-c <string>, **--config <string>**	Path to the delete imageset configuration file.
--delete-id <string>	Used to differentiate between versions for files created by the delete functionality.
--delete-v1-images	Used during the migration, along with **--generate**, in order to target images previously mirrored with oc-mirror plugin v1.
--delete-yaml-file <string>	If set, uses the generated or updated yaml file to delete contents.
--dest-tls-verify	Require HTTPS and verify certificates when talking to the container registry or daemon. The default value is **true**.
--force-cache-delete	Used to force delete the local cache manifests and blobs.
--generate	Used to generate the delete yaml for the list of manifests and blobs, used when deleting from local cache and remote registry.
-h, **--help**	Displays help.
--log-level <string>	Log level one of **info**, **debug**, **trace**, and **error**. The default value is **info**.
--parallel-images <unit>	Indicates the number of images deleted in parallel. The default value is **4**.
--parallel-layers <unit>	Indicates the number of image layers mirrored in parallel. The default value is **5**.
-p <unit>, **--port <unit>**	HTTP port used by oc-mirror's local storage instance. The default value is **55000**.
--retry-delay	Duration delay between 2 retries. The default value is **1s**.
--retry-times <int>	The number of times to retry. The default value is **2**.
--src-tls-verify	Require HTTPS and verify certificates when talking to the container registry or daemon. The default value is **true**.

Subcommand	Description
--workspace <string>	oc-mirror workspace where resources and internal artifacts are generated.

Additional resources

- Configuring your cluster to use the resources generated by oc-mirror

5.13. NEXT STEPS

After you mirror images to your disconnected environment using oc-mirror plugin v2, you can perform any of the following actions:

- Installing a cluster in a disconnected environment
- Using Operator Lifecycle Manager in disconnected environments
- Updating a cluster in a disconnected environment using the OpenShift Update Service

CHAPTER 6. MIGRATING FROM OC-MIRROR PLUGIN V1 TO V2

The oc-mirror v2 plugin introduces major changes to image mirroring workflows. This guide provides step-by-step instructions for migration while ensuring compatibility with oc-mirror plugin v2.

IMPORTANT

You must manually update the configurations by modifying the API version and removing deprecated fields. For more information, see "Changes from oc-mirror plugin v1 to v2".

6.1. CHANGES FROM OC-MIRROR PLUGIN V1 TO V2

Before migrating from oc-mirror plugin v1 to v2, see the following differences between oc-mirror plugin v1 and v2:

- Explicit version selection: Users must explicitly specify **--v2** when using **oc-mirror**. If no version is specified, v1 is executed by default. This behavior is expected to change in future releases, where **--v2** will be the default.
- Updated commands: Commands for mirroring workflows have changed to align with oc-mirror plugin v2's new workflow.
  - For mirror-to-disk, run the following command:

    ```
    $ oc-mirror --config isc.yaml file://<directory_name> --v2
    ```

  - For disk-to-mirror, run the following command:

    ```
    $ oc-mirror --config isc.yaml --from file://<directory_name> docker://<remote_registry> --v2
    ```

  - For mirror-to-mirror, run the following command:

    ```
    $ oc-mirror --config isc.yaml --workspace file://<directory_name> docker://<remote_registry> --v2
    ```

    NOTE

    --workspace is now required for mirror-to-mirror operation.

- API version update: The **ImageSetConfiguration** API version changes from **v1alpha2** (v1) to **v2alpha1** (v2). You must manually update the configuration files before migration.
- Configuration changes:
  - **storageConfig** must be removed in oc-mirror plugin v2.
  - Incremental mirroring is now handled automatically through the working directory or local cache.
- Changes in results directory: All custom resources to be applied to the disconnected cluster are generated in the **<workspace_path>/working-dir/cluster-resources** directory after the migration.

- Outputs in oc-mirror plugin v2 are not stored in the same location as oc-mirror plugin v1.
- You must check the **cluster-resources** folder under the working directory for the following resources:
  - **ImageDigestMirrorSet** (IDMS)
  - **ImageTagMirrorSet** (ITMS)
  - **CatalogSource**
  - **ClusterCatalog**
  - **UpdateService**

- Workspace and directory naming: Follow the new oc-mirror v2 convention to prevent any potential data inconsistencies when transitioning between versions.
  - The oc-mirror plugin v1 **oc-mirror-workspace** directory is no longer needed.
  - Use a new directory for oc-mirror plugin v2 to avoid conflicts.
- Replacing **ImageContentSourcePolicy** (ICSP) resources with IDMS/ITMS:

IMPORTANT

Deleting all **ImageContentSourcePolicy** (ICSP) resources might remove configurations unrelated to oc-mirror.

To avoid unintended deletions, identify ICSP resources generated by oc-mirror before removing them. If you are unsure, check with your cluster administrator. For more information, see "Mirroring images for a disconnected installation by using the oc-mirror plugin v2".

  - In oc-mirror plugin v2, the ICSP resource is replaced by **ImageDigestMirrorSet** (IDMS) and **ImageTagMirrorSet** (ITMS) resources.

6.2. MIGRATING TO OC-MIRROR PLUGIN V2

To migrate from oc-mirror plugin v1 to v2, you must manually update the **ImageSetConfiguration** file, modify mirroring commands, and clean up v1 artifacts. Follow these steps to complete the migration.

Procedure

1. Modify the API version and remove deprecated fields in your **ImageSetConfiguration**.

Example ImageSetConfiguration file with oc-mirror plugin v1 configuration

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
  platform:
    channels:
      - name: stable-4.17
```

```
    graph: true
  helm:
    repositories:
      - name: sbo
        url: https://redhat-developer.github.io/service-binding-operator-helm-chart/
  additionalImages:
    - name: registry.redhat.io/ubi8/ubi:latest
    - name: quay.io/openshifttest/hello-openshift@sha256:example_hash
  operators:
    - catalog: oci:///test/redhat-operator-index
      packages:
        - name: aws-load-balancer-operator
storageConfig:  # REMOVE this field in v2
  local:
    path: /var/lib/oc-mirror
```

Example ImageSetConfiguration file with oc-mirror plugin v2 configuration

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    channels:
      - name: stable-4.17
    graph: true
  helm:
    repositories:
      - name: sbo
        url: https://redhat-developer.github.io/service-binding-operator-helm-chart/
  additionalImages:
    - name: registry.redhat.io/ubi8/ubi:latest
    - name: quay.io/openshifttest/hello-openshift@sha256:example_hash
  operators:
    - catalog: oci:///test/redhat-operator-index
      packages:
        - name: aws-load-balancer-operator
```

NOTE

When migrating from oc-mirror plugin v1 to v2, you must use explicit registry hostnames for all images listed under **additionalImages**. Otherwise, images are mirrored to incorrect target paths.

2. Check the **cluster-resources** directory inside the working directory for IDMS, ITMS, **CatalogSource**, and **ClusterCatalog** resources by running the following command:

```
$ ls <v2_workspace>/working-dir/cluster-resources/
```

3. Once the migration is complete, verify that mirrored images and catalogs are available:

- Ensure that no errors or warnings occurred during mirroring.
- Ensure that no error file was generated (**working-dir/logs/mirroring_errors_YYYYMMdd_HHmmss.txt**).

4. Verify that mirrored images and catalogs are available using the following the commands:

```
$ oc get catalogsource -n openshift-marketplace
```

```
$ oc get imagedigestmirrorset,imagetagmirrorset
```

For more information, refer to "Mirroring images for a disconnected installation using oc-mirror plugin v2".

5. Optional: Remove images mirrored using oc-mirror plugin v1:

   a. Mirror the images using oc-mirror plugin v1.

   b. Update the API version in the **ImageSetConfiguration** file from **v1alpha2** (v1) to **v2alpha1** (v2), then run the following command:

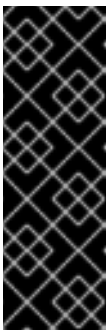
```
$ oc-mirror -c isc.yaml file://some-dir --v2
```

NOTE

storageConfig is not a valid field in the **ImageSetConfiguration** and **DeleteImageSetConfiguration** files. Remove this field when updating to oc-mirror plugin v2.

   c. Generate a delete manifest and delete v1 images by running the following command:

```
$ oc-mirror delete --config=delete-isc.yaml --generate --delete-v1-images --workspace
file://some-dir docker://registry.example:5000  --v2
```



IMPORTANT

oc-mirror plugin v2 does not automatically prune the destination registry, unlike oc-mirror plugin v1. To clean up images that are no longer needed, use the delete functionality in v2 with the **--delete-v1-images** command flag.

Once all images mirrored with oc-mirror plugin v1 are removed, you no longer need to use this flag. If you need to delete images mirrored with oc-mirror plugin v2, do not set **--delete-v1-images**.

For more information about deleting images, see "Deletion of images from your disconnected environment".

   d. Delete images based on the generated manifest by running the following command:

```
$ oc-mirror delete --delete-yaml-file some-dir/working-dir/delete/delete-images.yaml
docker://registry.example:5000 --v2
```

6.3. ADDITIONAL RESOURCES

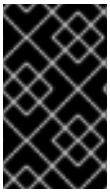
- Mirroring an image set in a partially disconnected environment
- Mirroring an image set in a fully disconnected environment

- For details regarding configuration changes, see Changes from oc-mirror plugin v1 to v2 .
- For more information about deleting images, see Deletion of images from your disconnected environment.

CHAPTER 7. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION USING THE OC-MIRROR PLUGIN

Running your cluster in a restricted network without direct internet connectivity is possible by installing the cluster from a mirrored set of OpenShift Container Platform container images in a private registry. This registry must be running at all times as long as the cluster is running. See the Prerequisites section for more information.

You can use the oc-mirror OpenShift CLI (**oc**) plugin to mirror images to a mirror registry in your fully or partially disconnected environments. You must run oc-mirror from a system with internet connectivity in order to download the required images from the official Red Hat registries.



IMPORTANT

The oc-mirror v1 plugin is deprecated. To prevent failures in a future release, specify the **--v1** flag to continue using the v1 plugin, or migrate to the supported v2 plugin and use the **--v2** flag. Transition to the oc-mirror v2 plugin for continued support and improvements.

7.1. ABOUT THE OC-MIRROR PLUGIN

You can use the oc-mirror OpenShift CLI (**oc**) plugin to mirror all required OpenShift Container Platform content and other images to your mirror registry by using a single tool. It provides the following features:

- Provides a centralized method to mirror OpenShift Container Platform releases, Operators, helm charts, and other images.
- Maintains update paths for OpenShift Container Platform and Operators.
- Uses a declarative image set configuration file to include only the OpenShift Container Platform releases, Operators, and images that your cluster needs.
- Performs incremental mirroring, which reduces the size of future image sets.
- Prunes images from the target mirror registry that were excluded from the image set configuration since the previous execution.
- Optionally generates supporting artifacts for OpenShift Update Service (OSUS) usage.

When using the oc-mirror plugin, you specify which content to mirror in an image set configuration file. In this YAML file, you can fine-tune the configuration to only include the OpenShift Container Platform releases and Operators that your cluster needs. This reduces the amount of data that you need to download and transfer. The oc-mirror plugin can also mirror arbitrary helm charts and additional container images to assist users in seamlessly synchronizing their workloads onto mirror registries.

The first time you run the oc-mirror plugin, it populates your mirror registry with the required content to perform your disconnected cluster installation or update. In order for your disconnected cluster to continue receiving updates, you must keep your mirror registry updated. To update your mirror registry, you run the oc-mirror plugin using the same configuration as the first time you ran it. The oc-mirror plugin references the metadata from the storage backend and only downloads what has been released since the last time you ran the tool. This provides update paths for OpenShift Container Platform and Operators and performs dependency resolution as required.

7.1.1. High level workflow

The following steps outline the high-level workflow on how to use the oc-mirror plugin to mirror images to a mirror registry:

1. Create an image set configuration file.
2. Mirror the image set to the target mirror registry by using one of the following methods:
   - Mirror an image set directly to the target mirror registry.
   - Mirror an image set to disk, transfer the image set to the target environment, then upload the image set to the target mirror registry.
3. Configure your cluster to use the resources generated by the oc-mirror plugin.
4. Repeat these steps to update your target mirror registry as necessary.

IMPORTANT

When using the oc-mirror CLI plugin to populate a mirror registry, any further updates to the target mirror registry must be made by using the oc-mirror plugin.

7.2. OC-MIRROR PLUGIN COMPATIBILITY AND SUPPORT

The oc-mirror plugin supports mirroring OpenShift Container Platform payload images and Operator catalogs for OpenShift Container Platform versions 4.12 and later.

NOTE

On **aarch64**, **ppc64le**, and **s390x** architectures the oc-mirror plugin is only supported for OpenShift Container Platform versions 4.14 and later.

Use the latest available version of the oc-mirror plugin regardless of which versions of OpenShift Container Platform you need to mirror.

Additional resources

- For information on updating oc-mirror, see Viewing the image pull source.

7.3. ABOUT THE MIRROR REGISTRY

You must have access to the internet to obtain the necessary container images. Using an alternative registry means that you place the mirror registry on a mirror host that has access to both your network and the internet.

You can mirror the images that are required for OpenShift Container Platform installation and subsequent product updates to a container mirror registry that supports Docker v2-2, such as Red Hat Quay. If you do not have access to a large-scale container registry, you can use the *mirror registry for Red Hat OpenShift*, which is a small-scale container registry included with OpenShift Container Platform subscriptions.

Regardless of your chosen registry, the procedure to mirror content from Red Hat hosted sites on the internet to an isolated image registry is the same. After you mirror the content, you configure each cluster to retrieve this content from your mirror registry.

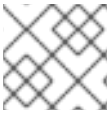
IMPORTANT

The OpenShift image registry cannot be used as the target registry because it does not support pushing without a tag, which is required during the mirroring process.

If choosing a container registry that is not the *mirror registry for Red Hat OpenShift*, it must be reachable by every machine in the clusters that you provision. If the registry is unreachable, installation, updating, or normal operations such as workload relocation might fail. For that reason, you must run mirror registries in a highly available way, and the mirror registries must at least match the production availability of your OpenShift Container Platform clusters.

When you populate your mirror registry with OpenShift Container Platform images, you can follow two scenarios. If you have a host that can access both the internet and your mirror registry, but not your cluster nodes, you can directly mirror the content from that machine. This process is referred to as *connected mirroring*. If you have no such host, you must mirror the images to a file system and then bring that host or removable media into your restricted environment. This process is referred to as *disconnected mirroring*.

For mirrored registries, to view the source of pulled images, you must review the **Trying to access** log entry in the CRI-O logs. Other methods to view the image pull source, such as using the **crictl images** command on a node, show the non-mirrored image name, even though the image is pulled from the mirrored location.



NOTE

Red Hat does not test third party registries with OpenShift Container Platform.

Additional resources

- Viewing the image pull source.

7.4. PREREQUISITES

- You must have a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as Red Hat Quay.

NOTE

If you use Red Hat Quay, you must use version 3.6 or later with the oc-mirror plugin. If you have an entitlement to Red Hat Quay, see the documentation on deploying Red Hat Quay for proof-of-concept purposes or by using the Red Hat Quay Operator. If you need additional assistance selecting and installing a registry, contact your sales representative or Red Hat Support.

If you do not already have an existing solution for a container image registry, subscribers of OpenShift Container Platform are provided a mirror registry for Red Hat OpenShift . The *mirror registry for Red Hat OpenShift* is included with your subscription and is a small-scale container registry that can be used to mirror the required container images of OpenShift Container Platform in disconnected installations.

7.5. PREPARING YOUR MIRROR HOSTS

Before you can use the oc-mirror plugin to mirror images, you must install the plugin and create a container image registry credentials file to allow the mirroring from Red Hat to your mirror.

7.5.1. Installing the oc-mirror OpenShift CLI plugin

Install the oc-mirror OpenShift CLI plugin to manage image sets in disconnected environments.

Prerequisites

- You have installed the OpenShift CLI (**oc**). If you are mirroring image sets in a fully disconnected environment, ensure the following:
  - You have installed the oc-mirror plugin on the host that has internet access.
  - The host in the disconnected environment has access to the target mirror registry.
- You have set the **umask** parameter to **0022** on the operating system that uses oc-mirror.
- You have installed the correct binary for the RHEL version that you are using.

Procedure

1. Download the oc-mirror CLI plugin:
   a. Navigate to the Downloads page of the Red Hat Hybrid Cloud Console.
   b. In the **OpenShift disconnected installation tools** section, select the **OS type** and **Architecture type** of the **OpenShift Client (oc) mirror plugin** from the dropdown menus.
   c. Click **Download** to save the file.
2. Extract the archive by running the following command:

```
$ tar xvzf oc-mirror.tar.gz
```

3. If necessary, update the plugin file to be executable by running the following command:

```
$ chmod +x oc-mirror
```

NOTE

Do not rename the **oc-mirror** file.

4. Install the oc-mirror CLI plugin by placing the file in your **PATH**, for example **/usr/local/bin**, by running the following command:

```
$ sudo mv oc-mirror /usr/local/bin/.
```

Verification

- Verify that the oc-mirror plugin v1 is successfully installed by running the following command:

```
$ oc mirror help
```

Additional resources

- Installing and using CLI plugins

7.5.2. Configuring credentials that allow images to be mirrored

Create a container image registry credentials file so that you can mirror images from Red Hat to your mirror. Complete the following steps on the installation host.

WARNING

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

Prerequisites

- You configured a mirror registry to use in your disconnected environment.
- You identified an image repository location on your mirror registry to mirror images into.
- You provisioned a mirror registry account that allows images to be uploaded to that image repository.
- You have write access to the mirror registry.

Procedure

1. Download your **registry.redhat.io** pull secret from Red Hat OpenShift Cluster Manager .
2. Make a copy of your pull secret in JSON format by running the following command:

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

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

Example pull secret

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

```
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

3. Save the file as either ~/**.docker/config.json** or **$XDG_RUNTIME_DIR/containers/auth.json**:

   a. If the **.docker** or **$XDG_RUNTIME_DIR/containers** directories do not exist, create one by entering the following command:

   ```
   $ mkdir -p <directory_name>
   ```

   Where **<directory_name>** is either ~/**.docker** or **$XDG_RUNTIME_DIR/containers**.

   b. Copy the pull secret to the appropriate directory by entering the following command:

   ```
   $ cp <path>/<pull_secret_file_in_json> <directory_name>/<auth_file>
   ```

   The **<directory_name>** is either ~/**.docker** or **$XDG_RUNTIME_DIR/containers**, and **<auth_file>** is either **config.json** or **auth.json**

1. Generate the base64-encoded user name and password or token for your mirror registry by running the following command:

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

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

Example output

```
BGVtbYk3ZHAtqXs=
```

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

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

- For the **<mirror_registry>** value, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:8443**.

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

Example modified pull secret

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

7.6. CREATING THE IMAGE SET CONFIGURATION

Before you can use the oc-mirror plugin to mirror image sets, you must create an image set configuration file. This image set configuration file defines which OpenShift Container Platform releases, Operators, and other images to mirror, along with other configuration settings for the oc-mirror plugin.

You must specify a storage backend in the image set configuration file. This storage backend can be a local directory or a registry that supports Docker v2-2. The oc-mirror plugin stores metadata in this storage backend during image set creation.

IMPORTANT

Do not delete or modify the metadata that is generated by the oc-mirror plugin. You must use the same storage backend every time you run the oc-mirror plugin for the same mirror registry.

Prerequisites

- You have created a container image registry credentials file. For instructions, see "Configuring credentials that allow images to be mirrored".

Procedure

1. Use the **oc mirror init** command to create a template for the image set configuration and save it to a file called **imageset-config.yaml**:

```
$ oc mirror init --registry <storage_backend> > imageset-config.yaml (1)
```

(1) Specifies the location of your storage backend, such as **example.com/mirror/oc-mirror-metadata**.

2. Edit the file and adjust the settings as necessary:

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
archiveSize: 4 (1)
storageConfig: (2)
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata (3)
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.18 (4)
      type: ocp
    graph: true (5)
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18 (6)
    packages:
    - name: serverless-operator (7)
      channels:
      - name: stable (8)
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest (9)
  helm: {}
```

(1) Add **archiveSize** to set the maximum size, in GiB, of each file within the image set.

(2) Set the back-end location to save the image set metadata to. This location can be a registry or local directory. It is required to specify **storageConfig** values.

(3) Set the registry URL for the storage backend.

(4) Set the channel to retrieve the OpenShift Container Platform images from.

(5) Add **graph: true** to build and push the graph-data image to the mirror registry. The graph-data image is required to create OpenShift Update Service (OSUS). The **graph: true** field also generates the **UpdateService** custom resource manifest. The **oc** command-line interface (CLI) can use the **UpdateService** custom resource manifest to create OSUS. For more information, see *About the OpenShift Update Service* .

(6) Set the Operator catalog to retrieve the OpenShift Container Platform images from.

(7) Specify only certain Operator packages to include in the image set. Remove this field to retrieve all packages in the catalog.

(8) Specify only certain channels of the Operator packages to include in the image set. You must always include the default channel for the Operator package even if you do not use the bundles in that channel. You can find the default channel by running the following

command: **oc mirror list operators --catalog=<catalog_name> --package=<package_name>**.

9 Specify any additional images to include in image set.

NOTE

The **graph: true** field also mirrors the **ubi-micro** image along with other mirrored images.

When upgrading OpenShift Container Platform Extended Update Support (EUS) versions, an intermediate version might be required between the current and target versions. For example, if the current version is **4.14** and target version is **4.16**, you might need to include a version such as **4.15.8** in the **ImageSetConfiguration** when using the oc-mirror plugin v1.

The oc-mirror plugin v1 might not always detect this automatically, so check the Cincinnati graph web page to confirm any required intermediate versions and add them manually to your configuration.

See "Image set configuration parameters" for the full list of parameters and "Image set configuration examples" for various mirroring use cases.

3. Save the updated file.
This image set configuration file is required by the **oc mirror** command when mirroring content.

Additional resources

- Image set configuration parameters
- Image set configuration examples
- Using the OpenShift Update Service in a disconnected environment

7.7. MIRRORING AN IMAGE SET TO A MIRROR REGISTRY

You can use the oc-mirror CLI plugin to mirror images to a mirror registry in a partially disconnected environment or in a fully disconnected environment.

These procedures assume that you already have your mirror registry set up.

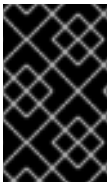
7.7.1. Mirroring an image set in a partially disconnected environment

In a partially disconnected environment, you can mirror an image set directly to the target mirror registry.

7.7.1.1. Mirroring from mirror to mirror

You can use the oc-mirror plugin to mirror an image set directly to a target mirror registry that is accessible during image set creation.

You are required to specify a storage backend in the image set configuration file. This storage backend can be a local directory or a Docker v2 registry. The oc-mirror plugin stores metadata in this storage backend during image set creation.



IMPORTANT

Do not delete or modify the metadata that is generated by the oc-mirror plugin. You must use the same storage backend every time you run the oc-mirror plugin for the same mirror registry.

Prerequisites

- You have access to the internet to get the required container images.
- You installed the OpenShift CLI (**oc**).
- You installed the oc-mirror CLI plugin.
- You created the image set configuration file.

Procedure

- Run the **oc mirror** command to mirror the images from the specified image set configuration to a specified registry:

```
$ oc mirror --config=./<imageset-config.yaml> \ 1
  docker://registry.example:5000                2
```

1. Specify the image set configuration file that you created. For example, **imageset-config.yaml**.
2. Specify the registry to mirror the image set file to. The registry must start with **docker://**. If you specify a top-level namespace for the mirror registry, you must also use this same namespace on subsequent executions.

Verification

1. Navigate into the **oc-mirror-workspace/** directory that was generated.
2. Navigate into the results directory, for example, **results-1639608409/**.
3. Verify that YAML files are present for the **ImageContentSourcePolicy** and **CatalogSource** resources.

NOTE

The **repositoryDigestMirrors** section of the **ImageContentSourcePolicy** YAML file is used for the **install-config.yaml** file during installation.

Next steps

- Convert the **ImageContentSourcePolicy** YAML content for use in manually configuring CRI-O.
- If required, mirror the images from mirror to disk for disconnected or offline use.
- Configure your cluster to use the resources generated by oc-mirror.

Troubleshooting

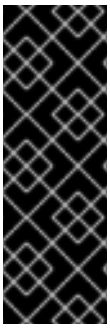
- Unable to retrieve source image .

7.7.2. Mirroring an image set in a fully disconnected environment

To mirror an image set in a fully disconnected environment, you must first mirror the image set to disk , then mirror the image set file on disk to a mirror .

7.7.2.1. Mirroring from mirror to disk

You can use the oc-mirror plugin to generate an image set and save the contents to disk. The generated image set can then be transferred to the disconnected environment and mirrored to the target registry.

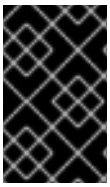


IMPORTANT

Depending on the configuration specified in the image set configuration file, using oc-mirror to mirror images might download several hundreds of gigabytes of data to disk.

The initial image set download when you populate the mirror registry is often the largest. Because you only download the images that changed since the last time you ran the command, when you run the oc-mirror plugin again, the generated image set is often smaller.

You are required to specify a storage backend in the image set configuration file. This storage backend can be a local directory or a docker v2 registry. The oc-mirror plugin stores metadata in this storage backend during image set creation.



IMPORTANT

Do not delete or modify the metadata that is generated by the oc-mirror plugin. You must use the same storage backend every time you run the oc-mirror plugin for the same mirror registry.

Prerequisites

- You have access to the internet to obtain the necessary container images.
- You have installed the OpenShift CLI (**oc**).
- You have installed the oc-mirror CLI plugin.
- You have created the image set configuration file.

Procedure

- Run the **oc mirror** command to mirror the images from the specified image set configuration to disk:

```
$ oc mirror --config=./imageset-config.yaml \ 1
  file://<path_to_output_directory>           2
```

1. Pass in the image set configuration file that was created. This procedure assumes that it is named **imageset-config.yaml**.
2. Specify the target directory where you want to output the image set file. The target directory path must start with **file://**.

Verification

1. Navigate to your output directory:

```
$ cd <path_to_output_directory>
```

2. Verify that an image set **.tar** file was created:

```
$ ls
```

Example output

```
mirror_seq1_000000.tar
```

Next steps

- Transfer the image set .tar file to the disconnected environment.

Troubleshooting

- Unable to retrieve source image .

7.7.2.2. Mirroring from disk to mirror

You can use the oc-mirror plugin to mirror the contents of a generated image set to the target mirror registry.

Prerequisites

- You have installed the OpenShift CLI (**oc**) in the disconnected environment.
- You have installed the oc-mirror CLI plugin in the disconnected environment.
- You have generated the image set file by using the **oc mirror** command.
- You have transferred the image set file to the disconnected environment.

Procedure

- Run the **oc mirror** command to process the image set file on disk and mirror the contents to a target mirror registry:

```
$ oc mirror --from=./mirror_seq1_000000.tar \ 1
  docker://registry.example:5000 2
```

1. Pass in the image set .tar file to mirror, named **mirror_seq1_000000.tar** in this example. If an **archiveSize** value was specified in the image set configuration file, the image set might

2. Specify the registry to mirror the image set file to. The registry must start with **docker://**. If you specify a top-level namespace for the mirror registry, you must also use this same namespace on subsequent executions.

This command updates the mirror registry with the image set and generates the **ImageContentSourcePolicy** and **CatalogSource** resources.

Verification

1. Navigate into the **oc-mirror-workspace/** directory that was generated.
2. Navigate into the results directory, for example, **results-1639608409/**.
3. Verify that YAML files are present for the **ImageContentSourcePolicy** and **CatalogSource** resources.

Next steps

- Configure your cluster to use the resources generated by oc-mirror.

Troubleshooting

- Unable to retrieve source image .

7.8. CONFIGURING YOUR CLUSTER TO USE THE RESOURCES GENERATED BY OC-MIRROR

After you have mirrored your image set to the mirror registry, you must apply the generated **ImageContentSourcePolicy**, **CatalogSource**, and release image signature resources into the cluster.

The **ImageContentSourcePolicy** resource associates the mirror registry with the source registry and redirects image pull requests from the online registries to the mirror registry. The **CatalogSource** resource is used by Operator Lifecycle Manager (OLM) Classic to retrieve information about the available Operators in the mirror registry. The release image signatures are used to verify the mirrored release images.

NOTE

OLM v1 uses the **ClusterCatalog** resource to retrieve information about the available cluster extensions in the mirror registry.

The oc-mirror plugin v1 does not generate **ClusterCatalog** resources automatically; you must manually create them. For more information on creating and applying **ClusterCatalog** resources, see "Adding a catalog to a cluster" in "Extensions".

Prerequisites

- You have mirrored the image set to the registry mirror in the disconnected environment.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Log in to OpenShift CLI (**oc**) as a user with the **cluster-admin** role.
2. Apply the YAML files from the results directory to the cluster by running the following command:

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/
```

3. If you mirrored release images, apply the release image signatures to the cluster by running the following command:

```
$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/
```

NOTE

If you are mirroring Operators instead of clusters, you do not need to run **$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/**. Running that command will return an error, as there are no release image signatures to apply.

Verification

1. Verify that the **ImageContentSourcePolicy** resources were successfully installed by running the following command:

```
$ oc get imagecontentsourcepolicy
```

2. Verify that the **CatalogSource** resources were successfully installed by running the following command:

```
$ oc get catalogsource -n openshift-marketplace
```

Additional resources

- Adding a catalog to a cluster in "Extensions"

7.9. UPDATING YOUR MIRROR REGISTRY CONTENT

You can update your mirror registry content by updating the image set configuration file and mirroring the image set to the mirror registry. The next time that you run the oc-mirror plugin, an image set is generated that only contains new and updated images since the previous execution.

While updating the mirror registry, you must take into account the following considerations:

- Images are pruned from the target mirror registry if they are no longer included in the latest image set that was generated and mirrored. Therefore, ensure that you are updating images for the same combination of the following key components so that only a differential image set is created and mirrored:
  - Image set configuration
  - Destination registry

    - Storage configuration

- The images can be pruned in case of disk to mirror or mirror to mirror workflow.
- The generated image sets must be pushed to the target mirror registry in sequence. You can derive the sequence number from the file name of the generated image set archive file.
- Do not delete or modify the metadata image that is generated by the oc-mirror plugin.
- If you specified a top-level namespace for the mirror registry during the initial image set creation, then you must use this same namespace every time you run the oc-mirror plugin for the same mirror registry.

For more information about the workflow to update the mirror registry content, see the "High level workflow" section.

7.9.1. Mirror registry update examples

This section covers the use cases for updating the mirror registry from disk to mirror.

Example ImageSetConfiguration file that was previously used for mirroring

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
```

7.9.1.1. Mirroring a specific OpenShift Container Platform version by pruning the existing images

Updated ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.13 (1)
```

```
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
```

1 Replacing by **stable-4.13** prunes all the images of **stable-4.12**.

7.9.1.2. Updating to the latest version of an Operator by pruning the existing images

Updated ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
        - name: rhacs-operator
          channels:
          - name: stable 1
```

1 Using the same channel without specifying a version prunes the existing images and updates with the latest version of images.

7.9.1.3. Mirroring a new Operator by pruning the existing Operator

Updated ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.12.1
        maxVersion: 4.12.1
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
```

```
    - name: <new_operator_name> (1)
      channels:
      - name: stable
```

(1) Replacing **rhacs-operator** with **new_operator_name** prunes the Red Hat Advanced Cluster Security for Kubernetes Operator.

7.9.1.4. Pruning all the OpenShift Container Platform images

Updated ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
      packages:
```

Additional resources

- Image set configuration examples
- Mirroring an image set in a partially disconnected environment
- Mirroring an image set in a fully disconnected environment
- Configuring your cluster to use the resources generated by oc-mirror

7.10. PERFORMING A DRY RUN

You can use oc-mirror to perform a dry run, without actually mirroring any images. This allows you to review the list of images that would be mirrored, as well as any images that would be pruned from the mirror registry. A dry run also allows you to catch any errors with your image set configuration early or use the generated list of images with other tools to carry out the mirroring operation.

Prerequisites

- You have access to the internet to obtain the necessary container images.
- You have installed the OpenShift CLI (**oc**).
- You have installed the oc-mirror CLI plugin.
- You have created the image set configuration file.

Procedure

1. Run the **oc mirror** command with the **--dry-run** flag to perform a dry run:

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000           \ 2
  --dry-run                                  3
```

1. Pass in the image set configuration file that was created. This procedure assumes that it is named **imageset-config.yaml**.
2. Specify the mirror registry. Nothing is mirrored to this registry as long as you use the **--dry-run** flag.
3. Use the **--dry-run** flag to generate the dry run artifacts and not an actual image set file.

Example output

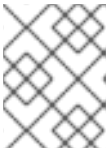
```
Checking push permissions for registry.example:5000
Creating directory: oc-mirror-workspace/src/publish
Creating directory: oc-mirror-workspace/src/v2
Creating directory: oc-mirror-workspace/src/charts
Creating directory: oc-mirror-workspace/src/release-signatures
No metadata detected, creating new workspace
wrote mirroring manifests to oc-mirror-workspace/operators.1658342351/manifests-redhat-operator-index

...

info: Planning completed in 31.48s
info: Dry run complete
Writing image mapping to oc-mirror-workspace/mapping.txt
```

2. Navigate into the workspace directory that was generated:

```
$ cd oc-mirror-workspace/
```

3. Review the **mapping.txt** file that was generated.
   This file contains a list of all images that would be mirrored.

4. Review the **pruning-plan.json** file that was generated.
   This file contains a list of all images that would be pruned from the mirror registry when the image set is published.



NOTE

The **pruning-plan.json** file is only generated if your oc-mirror command points to your mirror registry and there are images to be pruned.

7.11. INCLUDING LOCAL OCI OPERATOR CATALOGS

While mirroring OpenShift Container Platform releases, Operator catalogs, and additional images from a registry to a partially disconnected cluster, you can include Operator catalog images from a local file-based catalog on disk. The local catalog must be in the Open Container Initiative (OCI) format.

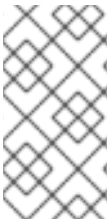
The local catalog and its contents are mirrored to your target mirror registry based on the filtering information in the image set configuration file.

IMPORTANT

When mirroring local OCI catalogs, any OpenShift Container Platform releases or additional images that you want to mirror along with the local OCI-formatted catalog must be pulled from a registry.

You cannot mirror OCI catalogs along with an oc-mirror image set file on disk.

One example use case for using the OCI feature is if you have a CI/CD system building an OCI catalog to a location on disk, and you want to mirror that OCI catalog along with an OpenShift Container Platform release to your mirror registry.



NOTE

If you used the Technology Preview OCI local catalogs feature for the oc-mirror plugin for OpenShift Container Platform 4.12, you can no longer use the OCI local catalogs feature of the oc-mirror plugin to copy a catalog locally and convert it to OCI format as a first step to mirroring to a fully disconnected cluster.

Prerequisites

- You have access to the internet to obtain the necessary container images.
- You have installed the OpenShift CLI (**oc**).
- You have installed the oc-mirror CLI plugin.

Procedure

1. Create the image set configuration file and adjust the settings as necessary.
   The following example image set configuration mirrors an OCI catalog on disk along with an OpenShift Container Platform release and a UBI image from **registry.redhat.io**.

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  local:
    path: /home/user/metadata                                    1
mirror:
  platform:
    channels:
    - name: stable-4.18                                          2
      type: ocp
    graph: false
  operators:
  - catalog: oci:///home/user/oc-mirror/my-oci-catalog           3
    targetCatalog: my-namespace/redhat-operator-index            4
    packages:
    - name: aws-load-balancer-operator
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18   5
    packages:
```

```
  - name: rhacs-operator
additionalImages:
- name: registry.redhat.io/ubi9/ubi:latest                    6
```

1. Set the back-end location to save the image set metadata to. This location can be a registry or local directory. It is required to specify **storageConfig** values.
2. Optionally, include an OpenShift Container Platform release to mirror from **registry.redhat.io**.
3. Specify the absolute path to the location of the OCI catalog on disk. The path must start with **oci:**// when using the OCI feature.
4. Optionally, specify an alternative namespace and name to mirror the catalog as.
5. Optionally, specify additional Operator catalogs to pull from a registry.
6. Optionally, specify additional images to pull from a registry.

NOTE

In oc-mirror plugin v2, you must use explicit registry hostnames for all images listed under **additionalImages**. Otherwise, images are mirrored to incorrect target paths.

2. Run the **oc mirror** command to mirror the OCI catalog to a target mirror registry:

```
$ oc mirror --config=./imageset-config.yaml \ 1
  docker://registry.example:5000                2
```

1. Pass in the image set configuration file. This procedure assumes that it is named **imageset-config.yaml**.
2. Specify the registry to mirror the content to. The registry must start with **docker:**//. If you specify a top-level namespace for the mirror registry, you must also use this same namespace on subsequent executions.

Optionally, you can specify other flags to adjust the behavior of the OCI feature:

--oci-insecure-signature-policy

Do not push signatures to the target mirror registry.

--oci-registries-config

Specify the path to a TOML-formatted **registries.conf** file. You can use this to mirror from a different registry, such as a pre-production location for testing, without having to change the image set configuration file. This flag only affects local OCI catalogs, not any other mirrored content.

Example registries.conf file

```
[[registry]]
 location = "registry.redhat.io:5000"
 insecure = false
```

```
blocked = false
mirror-by-digest-only = true
prefix = ""
[[registry.mirror]]
  location = "preprod-registry.example.com"
  insecure = false
```

Next steps

- Configure your cluster to use the resources generated by oc-mirror.

Additional resources

- Configuring your cluster to use the resources generated by oc-mirror

7.12. IMAGE SET CONFIGURATION PARAMETERS

The oc-mirror plugin requires an image set configuration file that defines what images to mirror. The following table lists the available parameters for the **ImageSetConfiguration** resource.

Table 7.1. ImageSetConfiguration parameters

Parameter	Description	Values
apiVersion	The API version for the **ImageSetConfiguration** content.	String. For example: **mirror.openshift.io/v1alpha2**.
archiveSize	The maximum size, in GiB, of each archive file within the image set.	Integer. For example: **4**
mirror	The configuration of the image set.	Object

mirror.additionalImages	The additional images configuration of the image set.	Array of objects. For example:
		`additionalImages:`
		`  - name: registry.redhat.io/ubi8/ubi:latest`

mirror.additionalImages.name	The tag or digest of the image to mirror.	String. For example: **registry.redhat.io/ubi8/ubi:latest**

Parameter	Description	Values
mirror.blockedImages	The full tag, digest, or pattern of images to block from mirroring.	Array of strings. For example: **docker.io/library/alpine**
mirror.helm	The helm configuration of the image set. Note that the oc-mirror plugin supports only helm charts that do not require user input when rendered.	Object

mirror.helm.local	The local helm charts to mirror.	Array of objects. For example:
		`local:`
		`- name: podinfo`
		`path: /test/podinfo-5.0.0.tar.gz`

mirror.helm.local.name	The name of the local helm chart to mirror.	String. For example: **podinfo**.
mirror.helm.local.path	The path of the local helm chart to mirror.	String. For example: **/test/podinfo-5.0.0.tar.gz**.

mirror.helm.repositories	The remote helm repositories to mirror from.	Array of objects. For example:
		`repositories:`
		`- name: podinfo`
		`url: https://example.github.io/podinfo`
		`charts:`
		`- name: podinfo`
		`version: 5.0.0`

mirror.helm.repositories.name	The name of the helm repository to mirror from.	String. For example: **podinfo**.

Parameter	Description	Values
mirror.helm.repositories.url	The URL of the helm repository to mirror from.	String. For example: **https://example.github.io/podinfo**.
mirror.helm.repositories.charts	The remote helm charts to mirror.	Array of objects.
mirror.helm.repositories.charts.name	The name of the helm chart to mirror.	String. For example: **podinfo**.
mirror.helm.repositories.charts.version	The version of the named helm chart to mirror.	String. For example: **5.0.0**.

mirror.operators	The Operators configuration of the image set.	Array of objects. For example:
		`operators:`
		`  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18`
		`    packages:`
		`    - name: elasticsearch-operator`
		`    minVersion: '2.4.0'`

mirror.operators.catalog	The Operator catalog to include in the image set.	String. For example: **registry.redhat.io/redhat/redhat-operator-index:v4.18**.
mirror.operators.full	When **true**, downloads the full catalog, Operator package, or Operator channel.	Boolean. The default value is **false**.

Parameter	Description	Values
mirror.operators.packages	The Operator packages configuration.	Array of objects. For example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	The Operator package name to include in the image set	String. For example: **elasticsearch-operator**.
mirror.operators.packages.channels	The Operator package channel configuration.	Object
mirror.operators.packages.channels.name	The Operator channel name, unique within a package, to include in the image set.	String. For example: **fast** or **stable-v4.18**.
mirror.operators.packages.channels.maxVersion	The highest version of the Operator mirror across all channels in which it exists. See the following note for further information.	String. For example: **5.2.3-31**
mirror.operators.packages.channels.minBundle	The name of the minimum bundle to include, plus all bundles in the update graph to the channel head. Set this field only if the named bundle has no semantic version metadata.	String. For example: **bundleName**
mirror.operators.packages.channels.minVersion	The lowest version of the Operator to mirror across all channels in which it exists. See the following note for further information.	String. For example: **5.2.3-31**

Parameter	Description	Values
mirror.operators.packages.maxVersion	The highest version of the Operator to mirror across all channels in which it exists. See the following note for further information.	String. For example: **5.2.3-31**.
mirror.operators.packages.minVersion	The lowest version of the Operator to mirror across all channels in which it exists. See the following note for further information.	String. For example: **5.2.3-31**.
mirror.operators.skipDependencies	If **true**, dependencies of bundles are not included.	Boolean. The default value is **false**.
mirror.operators.targetCatalog	An alternative name and optional namespace hierarchy to mirror the referenced catalog as.	String. For example: **my-namespace/my-operator-catalog**
mirror.operators.targetName	An alternative name to mirror the referenced catalog as. The **targetName** parameter is deprecated. Use the **targetCatalog** parameter instead.	String. For example: **my-operator-catalog**
mirror.operators.targetTag	An alternative tag to append to the **targetName** or **targetCatalog**.	String. For example: **v1**
mirror.platform	The platform configuration of the image set.	Object

Parameter	Description	Values

mirror.platform.architectures	The architecture of the platform release payload to mirror.	Array of strings. For example:
		`architectures:`
		`- amd64`
		`- arm64`
		`- multi`
		`- ppc64le`
		`- s390x`
		The default value is **amd64**. The value **multi** ensures that the mirroring is supported for all available architectures, eliminating the need to specify individual architectures.

mirror.platform.channels	The platform channel configuration of the image set.	Array of objects. For example:
		`channels:`
		`- name: stable-4.10`
		`- name: stable-4.18`

mirror.platform.channels.full	When **true**, sets the **minVersion** to the first release in the channel and the **maxVersion** to the last release in the channel.	Boolean. The default value is **false**.
mirror.platform.channels.name	The name of the release channel.	String. For example: **stable-4.18**
mirror.platform.channels.minVersion	The minimum version of the referenced platform to be mirrored.	String. For example: **4.12.6**
mirror.platform.channels.maxVersion	The highest version of the referenced platform to be mirrored.	String. For example: **4.18.1**

Parameter	Description	Values
mirror.platform.channels.shortestPath	Toggles shortest path mirroring or full range mirroring.	Boolean. The default value is **false**.
mirror.platform.channels.type	The type of the platform to be mirrored.	String. For example: **ocp** or **okd**. The default is **ocp**.
mirror.platform.graph	Indicates whether the OSUS graph is added to the image set and subsequently published to the mirror.	Boolean. The default value is **false**.
storageConfig	The back-end configuration of the image set.	Object
storageConfig.local	The local back-end configuration of the image set.	Object
storageConfig.local.path	The path of the directory to contain the image set metadata.	String. For example: **./path/to/dir/**.
storageConfig.registry	The registry back-end configuration of the image set.	Object
storageConfig.registry.imageURL	The back-end registry URI. Can optionally include a namespace reference in the URI.	String. For example: **quay.io/myuser/imageset:metadata**.
storageConfig.registry.skipTLS	Optionally skip TLS verification of the referenced back-end registry.	Boolean. The default value is **false**.

NOTE

Using the **minVersion** and **maxVersion** properties to filter for a specific Operator version range can result in a multiple channel heads error. The error message states that there are **multiple channel heads**. This is because when the filter is applied, the update graph of the Operator is truncated.

Operator Lifecycle Manager requires that every Operator channel contains versions that form an update graph with exactly one end point, that is, the latest version of the Operator. When the filter range is applied, that graph can turn into two or more separate graphs or a graph that has more than one end point.

To avoid this error, do not filter out the latest version of an Operator. If you still run into the error, depending on the Operator, either the **maxVersion** property must be increased or the **minVersion** property must be decreased. Because every Operator graph can be different, you might need to adjust these values until the error resolves.

7.13. IMAGE SET CONFIGURATION EXAMPLES

The following **ImageSetConfiguration** file examples show the configuration for various mirroring use cases.

7.13.1. Use case: Including the shortest OpenShift Container Platform update path

The following **ImageSetConfiguration** file uses a local storage backend and includes all OpenShift Container Platform versions along the shortest update path from the minimum version of **4.11.37** to the maximum version of **4.12.15**.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.11.37
        maxVersion: 4.12.15
        shortestPath: true
```

7.13.2. Use case: Including all versions of OpenShift Container Platform from a minimum to the latest version for multi-architecture releases

The following **ImageSetConfiguration** file uses a registry storage backend and includes all OpenShift Container Platform versions starting at a minimum version of **4.13.4** to the latest version in the channel. On every invocation of oc-mirror with this image set configuration, the latest release of the **stable-4.13** channel is evaluated, so running oc-mirror at regular intervals ensures that you automatically receive the latest releases of OpenShift Container Platform images.

By setting the value of **platform.architectures** to **multi**, you can ensure that the mirroring is supported for multi-architecture releases.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  platform:
    architectures:
      - "multi"
    channels:
      - name: stable-4.13
        minVersion: 4.13.4
        maxVersion: 4.13.6
```

7.13.3. Use case: Including Operator versions from a minimum to the latest

The following **ImageSetConfiguration** file uses a local storage backend and includes only the Red Hat Advanced Cluster Security for Kubernetes Operator, versions starting at 4.0.1 and later in the **stable** channel.

NOTE

When you specify a minimum or maximum version range, you might not receive all Operator versions in that range.

By default, oc-mirror excludes any versions that are skipped or replaced by a newer version in the Operator Lifecycle Manager (OLM) specification. Operator versions that are skipped might be affected by a CVE or contain bugs. Use a newer version instead. For more information on skipped and replaced versions, see Creating an update graph with OLM.

To receive all Operator versions in a specified range, you can set the **mirror.operators.full** field to **true**.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
      packages:
        - name: rhacs-operator
```

```
    channels:
    - name: stable
      minVersion: 4.0.1
```

NOTE

To specify a maximum version instead of the latest, set the **mirror.operators.packages.channels.maxVersion** field.

7.13.4. Use case: Including the Nutanix CSI Operator

The following **ImageSetConfiguration** file uses a local storage backend and includes the Nutanix CSI Operator, the OpenShift Update Service (OSUS) graph image, and an additional Red Hat Universal Base Image (UBI).

Example ImageSetConfiguration file

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  registry:
    imageURL: mylocalregistry/ocp-mirror/openshift4
    skipTLS: false
mirror:
  platform:
    channels:
    - name: stable-4.11
      type: ocp
    graph: true
  operators:
  - catalog: registry.redhat.io/redhat/certified-operator-index:v4.18
    packages:
    - name: nutanixcsioperator
      channels:
      - name: stable
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest
```

7.13.5. Use case: Including the default Operator channel

The following **ImageSetConfiguration** file includes the **stable-5.7** and **stable** channels for the OpenShift Elasticsearch Operator. Even if only the packages from the **stable-5.7** channel are needed, the **stable** channel must also be included in the **ImageSetConfiguration** file, because it is the default channel for the Operator. You must always include the default channel for the Operator package even if you do not use the bundles in that channel.

TIP

You can find the default channel by running the following command: **oc mirror list operators --catalog=<catalog_name> --package=<package_name>**.

Example ImageSetConfiguration file

```
-
```

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
    packages:
    - name: elasticsearch-operator
      channels:
      - name: stable-5.7
      - name: stable
```

7.13.6. Use case: Including an entire catalog (all versions)

The following **ImageSetConfiguration** file sets the **mirror.operators.full** field to **true** to include all versions for an entire Operator catalog.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
      full: true
```

7.13.7. Use case: Including an entire catalog (channel heads only)

The following **ImageSetConfiguration** file includes the channel heads for an entire Operator catalog.

By default, for each Operator in the catalog, oc-mirror includes the latest Operator version (channel head) from the default channel. If you want to mirror all Operator versions, and not just the channel heads, you must set the **mirror.operators.full** field to **true**.

This example also uses the **targetCatalog** field to specify an alternative namespace and name to mirror the catalog as.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
```

```
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
    targetCatalog: my-namespace/my-operator-catalog
```

7.13.8. Use case: Including arbitrary images and helm charts

The following **ImageSetConfiguration** file uses a registry storage backend and includes helm charts and an additional Red Hat Universal Base Image (UBI).

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
archiveSize: 4
storageConfig:
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
 platform:
   architectures:
     - "s390x"
   channels:
     - name: stable-4.18
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
 helm:
   repositories:
     - name: redhat-helm-charts
       url: https://raw.githubusercontent.com/redhat-developer/redhat-helm-charts/master
       charts:
         - name: ibm-mongodb-enterprise-helm
           version: 0.2.0
 additionalImages:
   - name: registry.redhat.io/ubi9/ubi:latest
```

7.13.9. Use case: Including the upgrade path for EUS releases

The following **ImageSetConfiguration** file includes the **eus-<version>** channel, where the **maxVersion** value is at least two minor versions higher than the **minVersion** value.

For example, in this **ImageSetConfiguration** file, the **minVersion** is set to **4.12.28**, while the **maxVersion** for the **eus-4.14** channel is **4.14.16**.

Example ImageSetConfiguration file

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    graph: true # Required for the OSUS Operator
    architectures:
    - amd64
    channels:
```

```
    - name: stable-4.12
      minVersion: '4.12.28'
      maxVersion: '4.12.28'
      shortestPath: true
      type: ocp
    - name: eus-4.14
      minVersion: '4.12.28'
      maxVersion: '4.14.16'
      shortestPath: true
      type: ocp
```

7.13.10. Use case: Including the multi-arch OpenShift Container Platform images and catalog for multicluster engine Operator

The following **ImageSetConfiguration** file includes multicluster engine for Kubernetes Operator and all OpenShift Container Platform versions starting at a minimum version of **4.18.0** in the channel.

Example ImageSetConfiguration file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL:
agent.agent.example.com:5000/openshift/release/metadata:latest/openshift/release/metadata:latest
mirror:
  platform:
    architectures:
      - "multi"
    channels:
    - name: stable-4.18
      minVersion: 4.18.0
      maxVersion: 4.18.1
      type: ocp
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.18
      packages:
        - name: multicluster-engine
```

7.14. COMMAND REFERENCE FOR OC-MIRROR

The following tables describe the **oc mirror** subcommands and flags:

Table 7.2. oc mirror subcommands

Subcommand	Description
completion	Generate the autocompletion script for the specified shell.
describe	Output the contents of an image set.
help	Show help about any subcommand.

Subcommand	Description
init	Output an initial image set configuration template.
list	List available platform and Operator content and their version.
version	Output the oc-mirror version.

Table 7.3. oc mirror flags

Flag	Description
-c, **--config <string>**	Specify the path to an image set configuration file.
--continue-on-error	If any non image-pull related error occurs, continue and attempt to mirror as much as possible.
--dest-skip-tls	Disable TLS validation for the target registry.
--dest-use-http	Use plain HTTP for the target registry.
--dry-run	Print actions without mirroring images. Generates **mapping.txt** and **pruning-plan.json** files.
--from <string>	Specify the path to an image set archive that was generated by an execution of oc-mirror to load into a target registry.
-h, **--help**	Show the help.
--ignore-history	Ignore past mirrors when downloading images and packing layers. Disables incremental mirroring and might download more data.
--manifests-only	Generate manifests for **ImageContentSourcePolicy** objects to configure a cluster to use the mirror registry, but do not actually mirror any images. To use this flag, you must pass in an image set archive with the **--from** flag.
--max-nested-paths <int>	Specify the maximum number of nested paths for destination registries that limit nested paths. The default is **0**.
--max-per-registry <int>	Specify the number of concurrent requests allowed per registry. The default is **6**.
--oci-insecure-signature-policy	Do not push signatures when mirroring local OCI catalogs (with **--include-local-oci-catalogs**).

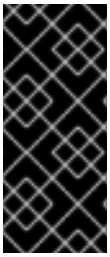
Flag	Description
--oci-registries-config	Provide a registries configuration file to specify an alternative registry location to copy from when mirroring local OCI catalogs (with **--include-local-oci-catalogs**).
--skip-cleanup	Skip removal of artifact directories.
--skip-image-pin	Do not replace image tags with digest pins in Operator catalogs.
--skip-metadata-check	Skip metadata when publishing an image set. NOTE This is recommended only when the image set was created with the **--ignore-history** flag.
--skip-missing	If an image is not found, skip it instead of reporting an error and aborting execution. Does not apply to custom images explicitly specified in the image set configuration.
--skip-pruning	Disable automatic pruning of images from the target mirror registry.
--skip-verification	Skip digest verification.
--source-skip-tls	Disable TLS validation for the source registry.
--source-use-http	Use plain HTTP for the source registry.
-v, **--verbose <int>**	Specify the number for the log level verbosity. Valid values are **0** - **9**. The default is **0**.

7.15. ADDITIONAL RESOURCES

- About cluster updates in a disconnected environment

CHAPTER 8. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION BY USING THE OC ADM COMMAND

You can ensure your clusters only use container images that satisfy your organizational controls on external content. Before you install a cluster on infrastructure that you provision in a restricted network, you must mirror the required container images into that environment. By using the **oc adm** command, you can mirror release and catalog images in OpenShift. To mirror container images, you must have a registry for mirroring.



IMPORTANT

You must have access to the internet to obtain the necessary container images. In this procedure, you place your mirror registry on a mirror host that has access to both your network and the internet. If you do not have access to a mirror host, use the Mirroring Operator catalogs for use with disconnected clusters procedure to copy images to a device you can move across network boundaries with.

8.1. PREREQUISITES

- You must have a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as one of the following registries:
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus Repository
  - Harbor

  If you have an entitlement to Red Hat Quay, see the documentation on deploying Red Hat Quay for proof-of-concept purposes or by using the Red Hat Quay Operator. If you need additional assistance selecting and installing a registry, contact your sales representative or Red Hat Support.
- If you do not already have an existing solution for a container image registry, subscribers of OpenShift Container Platform are provided a mirror registry for Red Hat OpenShift . The *mirror registry for Red Hat OpenShift* is included with your subscription and is a small-scale container registry that can be used to mirror the required container images of OpenShift Container Platform in disconnected installations.

8.2. ABOUT THE MIRROR REGISTRY

You must have access to the internet to obtain the necessary container images. Using an alternative registry means that you place the mirror registry on a mirror host that has access to both your network and the internet.

You can mirror the images that are required for OpenShift Container Platform installation and subsequent product updates to a container mirror registry such as Red Hat Quay, JFrog Artifactory, Sonatype Nexus Repository, or Harbor. If you do not have access to a large-scale container registry, you can use the *mirror registry for Red Hat OpenShift*, a small-scale container registry included with OpenShift Container Platform subscriptions.

You can use any container registry that supports Docker v2-2, such as Red Hat Quay, the *mirror registry*

for Red Hat OpenShift, Artifactory, Sonatype Nexus Repository, or Harbor. Regardless of your chosen registry, the procedure to mirror content from Red Hat hosted sites on the internet to an isolated image registry is the same. After you mirror the content, you configure each cluster to retrieve this content from your mirror registry.

IMPORTANT

The OpenShift image registry cannot be used as the target registry because it does not support pushing without a tag, which is required during the mirroring process.

If choosing a container registry that is not the *mirror registry for Red Hat OpenShift*, it must be reachable by every machine in the clusters that you provision. If the registry is unreachable, installation, updating, or normal operations such as workload relocation might fail. For that reason, you must run mirror registries in a highly available way, and the mirror registries must at least match the production availability of your OpenShift Container Platform clusters.

When you populate your mirror registry with OpenShift Container Platform images, you can follow two scenarios. If you have a host that can access both the internet and your mirror registry, but not your cluster nodes, you can directly mirror the content from that machine. This process is referred to as *connected mirroring*. If you have no such host, you must mirror the images to a file system and then bring that host or removable media into your restricted environment. This process is referred to as *disconnected mirroring*.

For mirrored registries, to view the source of pulled images, you must review the **Trying to access** log entry in the CRI-O logs. Other methods to view the image pull source, such as using the **crictl images** command on a node, show the non-mirrored image name, even though the image is pulled from the mirrored location.

NOTE

Red Hat does not test third party registries with OpenShift Container Platform.

Additional information

For information about viewing the CRI-O logs to view the image source, see Viewing the image pull source.

8.3. PREPARING YOUR MIRROR HOST

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

8.3.1. Installing the OpenShift CLI on Linux

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

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

8.3.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

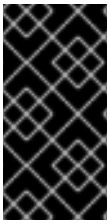
Verification

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

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

8.3.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS.



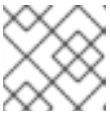
IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.



NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

5. Unpack and unzip the archive.
6. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

8.4. CONFIGURING CREDENTIALS THAT ALLOW IMAGES TO BE MIRRORED

Create a container image registry credentials file so that you can mirror images from Red Hat to your mirror. Complete the following steps on the installation host.

WARNING

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

Prerequisites

- You configured a mirror registry to use in your disconnected environment.
- You identified an image repository location on your mirror registry to mirror images into.
- You provisioned a mirror registry account that allows images to be uploaded to that image repository.
- You have write access to the mirror registry.

Procedure

1. Download your **registry.redhat.io** pull secret from Red Hat OpenShift Cluster Manager .
2. Make a copy of your pull secret in JSON format by running the following command:

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

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

Example pull secret

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

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

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

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

Example output

```
BGVtbYk3ZHAtqXs=
```

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

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

- For the **<mirror_registry>** value, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:8443**.
- For the **<credentials>** value, specify the base64-encoded user name and password for the mirror registry.

Example modified pull secret

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

8.5. MIRRORING THE OPENSHIFT CONTAINER PLATFORM IMAGE REPOSITORY

Mirror the OpenShift Container Platform image repository to your registry to use during cluster installation or upgrade. Complete the following steps on the mirror host.

Prerequisites

- Your mirror host has access to the internet.
- You configured a mirror registry to use in your restricted network and can access the certificate and credentials that you configured.
- You downloaded the pull secret from Red Hat OpenShift Cluster Manager and modified it to include authentication to your mirror repository.
- If you use self-signed certificates, you have specified a Subject Alternative Name in the certificates.

Procedure

1. Review the OpenShift Container Platform downloads page to determine the version of OpenShift Container Platform that you want to install and determine the corresponding tag on the Repository Tags page.
2. Set the following required environment variables:
   a. Export the release version:

      ```
      $ OCP_RELEASE=<release_version>
      ```

      For **<release_version>**, specify the tag that corresponds to the version of OpenShift Container Platform to install, such as **4.20.1**.

   b. Export the local registry name and host port:

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

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

   c. Export the local repository name:

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

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

   d. Export the name of the repository to mirror:

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

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

e. Export the path to your registry pull secret:

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

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

f. Export the release mirror:

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

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

g. Export the type of architecture for your cluster:

```
$ ARCHITECTURE=<cluster_architecture>
```

Specify the architecture of the cluster, such as **x86_64**, **aarch64**, **s390x**, or **ppc64le**.

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

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

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

3. Mirror the version images to the mirror registry:

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

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

ii. Review the images and configuration manifests to mirror:

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

iii. Record the entire **imageContentSources** section from the output of the previous command. The information about your mirrors is unique to your mirrored repository, and you must add the **imageContentSources** section to the **install-config.yaml** file during installation.

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

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

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

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

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

IMPORTANT

Running the **oc image mirror** command might result in the following error: **error: unable to retrieve source image**. This error occurs when image indexes include references to images that no longer exist on the image registry. Image indexes might retain older references to allow users running those images an upgrade path to newer points on the upgrade graph. As a temporary workaround, you can use the **--skip-missing** option to bypass the error and continue downloading the image index. For more information, see Service Mesh Operator mirroring failed.

- If the local container registry is connected to the mirror host, take the following actions:

  i. Directly push the release images to the local registry by using following command:

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

  This command pulls the release information as a digest, and its output includes the **imageContentSources** data that you require when you install your cluster.

  ii. Record the entire **imageContentSources** section from the output of the previous command. The information about your mirrors is unique to your mirrored repository, and you must add the **imageContentSources** section to the **install-config.yaml** file during installation.

NOTE

The image name gets patched to Quay.io during the mirroring process, and the Podman images will show Quay.io in the registry on the bootstrap virtual machine.

4. To create the installation program that is based on the content that you mirrored, extract it and pin it to the release:

   - If your mirror host does not have internet access, run the following command:

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --icsp-file=<file> --
command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
```

```
${ARCHITECTURE}" \
--insecure=true
```

Optional: If you do not want to configure trust for the target registry, add the **--insecure=true** flag.

- If the local container registry is connected to the mirror host, run the following command:

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}"
```

IMPORTANT

To ensure that you use the correct images for the version of OpenShift Container Platform that you selected, you must extract the installation program from the mirrored content.

You must perform this step on a machine with an active internet connection.

5. For clusters using installer-provisioned infrastructure, run the following command:

```
$ openshift-install
```

8.6. THE CLUSTER SAMPLES OPERATOR IN A DISCONNECTED ENVIRONMENT

In a disconnected environment, you must take additional steps after you install a cluster to configure the Cluster Samples Operator. Review the following information in preparation.

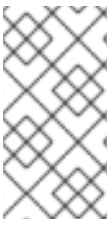
8.6.1. Cluster Samples Operator assistance for mirroring

During installation, OpenShift Container Platform creates a config map named **imagestreamtag-to-image** in the **openshift-cluster-samples-operator** namespace.

The **imagestreamtag-to-image** config map contains an entry, the populating image, for each image stream tag.

The format of the key for each entry in the data field in the config map is **<image_stream_name>_<image_stream_tag_name>**.

During a disconnected installation of OpenShift Container Platform, the status of the Cluster Samples Operator is set to **Removed**. If you choose to change it to **Managed**, it installs samples.



NOTE

The use of samples in a network-restricted or discontinued environment might require access to services external to your network. Some example services include: Github, Maven Central, npm, RubyGems, PyPi and others. There might be additional steps to take that allow the Cluster Samples Operators objects to reach the services they require.

Use the following principles to determine which images you need to mirror for your image streams to import:

- While the Cluster Samples Operator is set to **Removed**, you can create your mirrored registry, or determine which existing mirrored registry you want to use.
- Mirror the samples you want to the mirrored registry using the new config map as your guide.
- Add any of the image streams you did not mirror to the **skippedImagestreams** list of the Cluster Samples Operator configuration object.
- Set **samplesRegistry** of the Cluster Samples Operator configuration object to the mirrored registry.
- Then set the Cluster Samples Operator to **Managed** to install the image streams you have mirrored.

8.7. MIRRORING OPERATOR CATALOGS FOR USE WITH DISCONNECTED CLUSTERS

You can mirror the Operator contents of a Red Hat-provided catalog, or a custom catalog, into a container image registry using the **oc adm catalog mirror** command. The target registry must support Docker v2-2. For a cluster on a restricted network, this registry can be one that the cluster has network access to, such as a mirror registry created during a restricted network cluster installation.

IMPORTANT

- The OpenShift image registry cannot be used as the target registry because it does not support pushing without a tag, which is required during the mirroring process.
- Running **oc adm catalog mirror** might result in the following error: **error: unable to retrieve source image**. This error occurs when image indexes include references to images that no longer exist on the image registry. Image indexes might retain older references to allow users running those images an upgrade path to newer points on the upgrade graph. As a temporary workaround, you can use the **--skip-missing** option to bypass the error and continue downloading the image index. For more information, see Service Mesh Operator mirroring failed.

The **oc adm catalog mirror** command also automatically mirrors the index image that is specified during the mirroring process, whether it be a Red Hat-provided index image or your own custom-built index image, to the target registry. You can then use the mirrored index image to create a catalog source that allows Operator Lifecycle Manager (OLM) to load the mirrored catalog onto your OpenShift Container Platform cluster.

Additional resources

- Using Operator Lifecycle Manager in disconnected environments

8.7.1. Prerequisites

Mirroring Operator catalogs for use with disconnected clusters has the following prerequisites:

- Workstation with unrestricted network access.

- **podman** version 1.9.3 or later.
- If you want to filter, or *prune*, an existing catalog and selectively mirror only a subset of Operators, see the following sections:
  - Installing the opm CLI
  - Updating or filtering a file-based catalog image
- If you want to mirror a Red Hat-provided catalog, run the following command on your workstation with unrestricted network access to authenticate with **registry.redhat.io**:

```
$ podman login registry.redhat.io
```

- Access to a mirror registry that supports Docker v2-2.
- On your mirror registry, decide which repository, or namespace, to use for storing mirrored Operator content. For example, you might create an **olm-mirror** repository.
- If your mirror registry does not have internet access, connect removable media to your workstation with unrestricted network access.
- If you are working with private registries, including **registry.redhat.io**, set the **REG_CREDS** environment variable to the file path of your registry credentials for use in later steps. For example, for the **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

8.7.2. Extracting and mirroring catalog contents

The **oc adm catalog mirror** command extracts the contents of an index image to generate the manifests required for mirroring. The default behavior of the command generates manifests, then automatically mirrors all of the image content from the index image, as well as the index image itself, to your mirror registry.

Alternatively, if your mirror registry is on a completely disconnected, or *airgapped*, host, you can first mirror the content to removable media, move the media to the disconnected environment, then mirror the content from the media to the registry.

8.7.2.1. Mirroring catalog contents to registries on the same network

If your mirror registry is co-located on the same network as your workstation with unrestricted network access, take the following actions on your workstation.

Procedure

1. If your mirror registry requires authentication, run the following command to log in to the registry:

```
$ podman login <mirror_registry>
```

2. Run the following command to extract and mirror the content to the mirror registry:

```
$ oc adm catalog mirror \
```

```
<index_image> \ 1
<mirror_registry>:<port>[/<repository>] \ 2
[-a ${REG_CREDS}] \ 3
[--insecure] \ 4
[--index-filter-by-os='<platform>/<arch>'] \ 5
[--manifests-only] 6
```

1. Specify the index image for the catalog that you want to mirror.
2. Specify the fully qualified domain name (FQDN) for the target registry to mirror the Operator contents to. The mirror registry **<repository>** can be any existing repository, or namespace, on the registry, for example **olm-mirror** as outlined in the prerequisites. If there is an existing repository found during mirroring, the repository name is added to the resulting image name. If you do not want the image name to include the repository name, omit the **<repository>** value from this line, for example **<mirror_registry>:<port>**.
3. Optional: If required, specify the location of your registry credentials file. **{REG_CREDS}** is required for **registry.redhat.io**.
4. Optional: If you do not want to configure trust for the target registry, add the **--insecure** flag.
5. Optional: Specify which platform and architecture of the index image to select when multiple variants are available. Images are passed as **'<platform>/<arch>[/<variant>]'**. This does not apply to images referenced by the index. Valid values are **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **linux/arm64**.
6. Optional: Generate only the manifests required for mirroring without actually mirroring the image content to a registry. This option can be useful for reviewing what will be mirrored, and lets you make any changes to the mapping list, if you require only a subset of packages. You can then use the **mapping.txt** file with the **oc image mirror** command to mirror the modified list of images in a later step. This flag is intended for only advanced selective mirroring of content from the catalog.

Example output

```
src image has index label for database path: /database/index.db
using database path mapping: /database/index.db:/tmp/153048078
wrote database to /tmp/153048078 1
...
wrote mirroring manifests to manifests-redhat-operator-index-1614211642 2
```

1. Directory for the temporary **index.db** database generated by the command.
2. Record the manifests directory name that is generated. This directory is referenced in subsequent procedures.

NOTE

Red Hat Quay does not support nested repositories. As a result, running the **oc adm catalog mirror** command will fail with a **401** unauthorized error. As a workaround, you can use the **--max-components=2** option when running the **oc adm catalog mirror** command to disable the creation of nested repositories. For more information on this workaround, see the Unauthorized error thrown while using catalog mirror command with Quay registry Knowledgebase Solution.

Additional resources

- Architecture and operating system support for Operators

8.7.2.2. Mirroring catalog contents to airgapped registries

If your mirror registry is on a completely disconnected, or airgapped, host, take the following actions.

Procedure

1. Run the following command on your workstation with unrestricted network access to mirror the content to local files:

```
$ oc adm catalog mirror \
    <index_image> \ 1
    file:///local/index \ 2
    -a ${REG_CREDS} \ 3
    --insecure \ 4
    --index-filter-by-os='<platform>/<arch>' 5
```

1. Specify the index image for the catalog that you want to mirror.
2. Specify the content to mirror to local files in your current directory.
3. Optional: If required, specify the location of your registry credentials file.
4. Optional: If you do not want to configure trust for the target registry, add the **--insecure** flag.
5. Optional: Specify which platform and architecture of the index image to select when multiple variants are available. Images are specified as **'<platform>/<arch>[/<variant>]'**. This does not apply to images referenced by the index. Valid values are **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **linux/arm64**, and **.***

Example output

```
...
info: Mirroring completed in 5.93s (5.915MB/s)
wrote mirroring manifests to manifests-my-index-1614985528 1
```

```
To upload local images to a registry, run:

oc adm catalog mirror file://local/index/myrepo/my-index:v1 REGISTRY/REPOSITORY (2)
```

(1) Record the manifests directory name that is generated. This directory is referenced in subsequent procedures.

(2) Record the expanded **file://** path that is based on your provided index image. This path is referenced in a subsequent step.

This command creates a **v2/** directory in your current directory.

2. Copy the **v2/** directory to removable media.
3. Physically remove the media and attach it to a host in the disconnected environment that has access to the mirror registry.
4. If your mirror registry requires authentication, run the following command on your host in the disconnected environment to log in to the registry:

```
$ podman login <mirror_registry>
```

5. Run the following command from the parent directory containing the **v2/** directory to upload the images from local files to the mirror registry:

```
$ oc adm catalog mirror \
  file://local/index/<repository>/<index_image>:<tag> \ (1)
  <mirror_registry>:<port>[/<repository>] \ (2)
  -a ${REG_CREDS} \ (3)
  --insecure \ (4)
  --index-filter-by-os='<platform>/<arch>' (5)
```

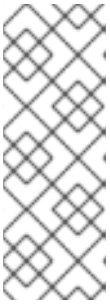
(1) Specify the **file://** path from the previous command output.

(2) Specify the fully qualified domain name (FQDN) for the target registry to mirror the Operator contents to. The mirror registry **<repository>** can be any existing repository, or namespace, on the registry, for example **olm-mirror** as outlined in the prerequisites. If there is an existing repository found during mirroring, the repository name is added to the resulting image name. If you do not want the image name to include the repository name, omit the **<repository>** value from this line, for example **<mirror_registry>:<port>**.

(3) Optional: If required, specify the location of your registry credentials file.

(4) Optional: If you do not want to configure trust for the target registry, add the **--insecure** flag.

(5) Optional: Specify which platform and architecture of the index image to select when multiple variants are available. Images are specified as **'<platform>/<arch>[/<variant>]'**. This does not apply to images referenced by the index. Valid values are **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **linux/arm64**, and **.***



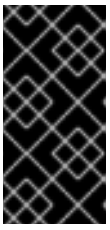
NOTE

Red Hat Quay does not support nested repositories. As a result, running the **oc adm catalog mirror** command will fail with a **401** unauthorized error. As a workaround, you can use the **--max-components=2** option when running the **oc adm catalog mirror** command to disable the creation of nested repositories. For more information on this workaround, see the Unauthorized error thrown while using catalog mirror command with Quay registry Knowledgebase Solution.

6. Run the **oc adm catalog mirror** command again. Use the newly mirrored index image as the source and the same mirror registry target used in the previous step:

```
$ oc adm catalog mirror \
    <mirror_registry>:<port>/<index_image> \
    <mirror_registry>:<port>[/<repository>] \
    --manifests-only \ 1
    [-a ${REG_CREDS}] \
    [--insecure]
```

1 The **--manifests-only** flag is required for this step so that the command does not copy all of the mirrored content again.



IMPORTANT

This step is required because the image mappings in the **imageContentSourcePolicy.yaml** file generated during the previous step must be updated from local paths to valid mirror locations. Failure to do so will cause errors when you create the **ImageContentSourcePolicy** object in a later step.

After you mirror the catalog, you can continue with the remainder of your cluster installation. After your cluster installation has finished successfully, you must specify the manifests directory from this procedure to create the **ImageContentSourcePolicy** and **CatalogSource** objects. These objects are required to enable installation of Operators from OperatorHub.

Additional resources

- Architecture and operating system support for Operators

8.7.3. Generated manifests

After mirroring Operator catalog content to your mirror registry, a manifests directory is generated in your current directory.

If you mirrored content to a registry on the same network, the directory name takes the following pattern:

```
manifests-<index_image_name>-<random_number>
```

If you mirrored content to a registry on a disconnected host in the previous section, the directory name takes the following pattern:

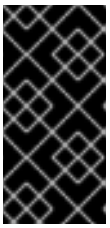
```
manifests-index/<repository>/<index_image_name>-<random_number>
```

NOTE

The manifests directory name is referenced in subsequent procedures.

The manifests directory contains the following files, some of which might require further modification:

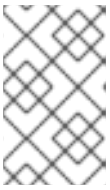
- The **catalogSource.yaml** file is a basic definition for a **CatalogSource** object that is pre-populated with your index image tag and other relevant metadata. This file can be used as is or modified to add the catalog source to your cluster.



IMPORTANT

If you mirrored the content to local files, you must modify your **catalogSource.yaml** file to remove any backslash (/) characters from the **metadata.name** field. Otherwise, when you attempt to create the object, it fails with an "invalid resource name" error.

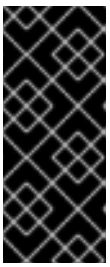
- The **imageContentSourcePolicy.yaml** file defines an **ImageContentSourcePolicy** object that can configure nodes to translate between the image references stored in Operator manifests and the mirrored registry.



NOTE

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

- The **mapping.txt** file contains all of the source images and where to map them in the target registry. This file is compatible with the **oc image mirror** command and can be used to further customize the mirroring configuration.



IMPORTANT

If you used the **--manifests-only** flag during the mirroring process and want to further trim the subset of packages to mirror, see the steps in the Mirroring a package manifest format catalog image procedure of the OpenShift Container Platform 4.7 documentation about modifying your **mapping.txt** file and using the file with the **oc image mirror** command.

8.7.4. Postinstallation requirements

After you mirror the catalog, you can continue with the remainder of your cluster installation. After your cluster installation has finished successfully, you must specify the manifests directory from this procedure to create the **ImageContentSourcePolicy** and **CatalogSource** objects. These objects are required to populate and enable installation of Operators from OperatorHub.

Additional resources

- Populating OperatorHub from mirrored Operator catalogs
- Updating or filtering a file-based catalog image

8.8. NEXT STEPS

- Install a cluster on infrastructure that you provision in your restricted network, such as on VMware vSphere, bare metal, or Amazon Web Services.

8.9. ADDITIONAL RESOURCES

- See Gathering data about specific features for more information about using must-gather.

CHAPTER 9. INSTALLING A CLUSTER IN A DISCONNECTED ENVIRONMENT

You can install an OpenShift Container Platform cluster in a disconnected environment, choosing the installation method and infrastructure that best suits your requirements. This includes installing OpenShift Container Platform on either on-premise hardware or on a cloud hosting service such as Amazon Web Services (AWS).

The following sections outline all of the supported methods for installing a cluster in a disconnected environment.

NOTE

In order to learn about other requirements for installing a cluster using a particular method, be sure to review other content in the procedure's respective section of the documentation.

For example, if you plan to install a cluster on AWS with installer-provisioned infrastructure, see Configuring an AWS account and Preparing to install a cluster on AWS

9.1. INSTALLING A CLUSTER WITH THE AGENT-BASED INSTALLER

To learn more about installing a cluster in a disconnected environment with the Agent-based installer, see the following pages:

- Understanding disconnected installation mirroring
- Installing an OpenShift Container Platform cluster with the Agent-based Installer

9.2. INSTALLING A CLUSTER ON AMAZON WEB SERVICES

To learn more about installing a cluster on Amazon Web Services (AWS) in a disconnected environment, see the following procedures:

- Installer-provisioned infrastructure: Installing a cluster on AWS in a restricted network
- User-provisioned infrastructure: Installing a cluster on AWS in a restricted network with user-provisioned infrastructure

9.3. INSTALLING A CLUSTER ON MICROSOFT AZURE

To learn more about installing a cluster on Microsoft Azure in a disconnected environment, see the following procedures:

- Installer-provisioned infrastructure: Installing a cluster on Azure in a restricted network
- User-provisioned infrastructure: Installing a cluster on Azure in a restricted network with user-provisioned infrastructure

9.4. INSTALLING A CLUSTER ON GOOGLE CLOUD

To learn more about installing a cluster on Google Cloud in a disconnected environment, see the following procedures:

- Installer-provisioned infrastructure: Installing a cluster on Google Cloud in a restricted network
- User-provisioned infrastructure: Installing a cluster on Google Cloud in a restricted network with user-provisioned infrastructure

9.5. INSTALLING A CLUSTER ON IBM CLOUD

To learn more about installing a cluster on IBM Cloud® in a disconnected environment, see the following procedure:

- Installing a cluster on IBM Cloud in a restricted network

9.6. INSTALLING A CLUSTER ON NUTANIX

To learn more about installing a cluster on Nutanix in a disconnected environment, see the following procedure:

- Installing a cluster on Nutanix in a restricted network

9.7. INSTALLING A BARE-METAL CLUSTER

To learn more about installing a bare-metal cluster in a disconnected environment, see the following procedure:

- Installing a user-provisioned bare metal cluster on a restricted network

9.8. INSTALLING A CLUSTER ON IBM Z(R) OR IBM(R) LINUXONE

To learn more about installing a cluster on IBM Z® or IBM® LinuxONE in a disconnected environment, see the following procedures:

- Installing a cluster with z/VM on IBM Z and IBM LinuxONE in a restricted network
- Installing a cluster with RHEL KVM on IBM Z and IBM LinuxONE in a restricted network
- Installing a cluster in an LPAR on IBM Z and IBM LinuxONE in a restricted network

9.9. INSTALLING A CLUSTER ON IBM POWER

To learn more about installing a cluster on IBM Power in a disconnected environment, see the following procedure:

- Installing a cluster on IBM Power in a restricted network

9.10. INSTALLING A CLUSTER ON OPENSTACK

To learn more about installing a cluster on Red Hat OpenStack Platform (RHOSP) in a disconnected environment, see the following procedure:

- Installing a cluster on OpenStack in a restricted network

9.11. INSTALLING A CLUSTER ON VSPHERE

To learn more about installing a cluster on VMware vSphere in a disconnected environment, see the following procedures:

- Installer-provisioned infrastructure: Installing a cluster on vSphere in a restricted network
- User-provisioned infrastructure: Installing a cluster on vSphere in a restricted network with user-provisioned infrastructure

CHAPTER 10. USING OPERATOR LIFECYCLE MANAGER IN DISCONNECTED ENVIRONMENTS

For OpenShift Container Platform clusters in disconnected environments, Operator Lifecycle Manager (OLM) by default cannot access the Red Hat-provided OperatorHub sources hosted on remote registries because those remote sources require full internet connectivity.

However, as a cluster administrator you can still enable your cluster to use OLM in a disconnected environment if you have a workstation that has full internet access. The workstation, which requires full internet access to pull the remote OperatorHub content, is used to prepare local mirrors of the remote sources, and push the content to a mirror registry.

The mirror registry can be located on a bastion host, which requires connectivity to both your workstation and the disconnected cluster, or a completely disconnected, or *airgapped*, host, which requires removable media to physically move the mirrored content to the disconnected environment.

This guide describes the following process that is required to enable OLM in disconnected environments:

- Disable the default remote OperatorHub sources for OLM.
- Use a workstation with full internet access to create and push local mirrors of the OperatorHub content to a mirror registry.
- Configure OLM to install and manage Operators from local sources on the mirror registry instead of the default remote sources.

After enabling OLM in a disconnected environment, you can continue to use your unrestricted workstation to keep your local OperatorHub sources updated as newer versions of Operators are released.

IMPORTANT

While OLM can manage Operators from local sources, the ability for a given Operator to run successfully in a disconnected environment still depends on the Operator itself meeting the following criteria:

- List any related images, or other container images that the Operator might require to perform their functions, in the **relatedImages** parameter of its **ClusterServiceVersion** (CSV) object.
- Reference all specified images by a digest (SHA) and not by a tag.

You can search software on the Red Hat Ecosystem Catalog for a list of Red Hat Operators that support running in disconnected mode by filtering with the following selections:

Type	Containerized application
Deployment method	Operator
Infrastructure features	Disconnected

Additional resources

- Red Hat-provided Operator catalogs
- Enabling your Operator for restricted network environments

10.1. PREREQUISITES

- You are logged in to your OpenShift Container Platform cluster as a user with **cluster-admin** privileges.
- If you are using OLM in a disconnected environment on IBM Z®, you must have at least 12 GB allocated to the directory where you place your registry.

10.2. DISABLING THE DEFAULT OPERATORHUB CATALOG SOURCES

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator. You can then configure OperatorHub to use local catalog sources.

Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

-

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

10.3. MIRRORING AN OPERATOR CATALOG

For instructions about mirroring Operator catalogs for use with disconnected clusters, see Mirroring Operator catalogs for use with disconnected clusters.

IMPORTANT

As of OpenShift Container Platform 4.11, the default Red Hat-provided Operator catalog releases in the file-based catalog format. The default Red Hat-provided Operator catalogs for OpenShift Container Platform 4.6 through 4.10 released in the deprecated SQLite database format.

The **opm** subcommands, flags, and functionality related to the SQLite database format are also deprecated and will be removed in a future release. The features are still supported and must be used for catalogs that use the deprecated SQLite database format.

Many of the **opm** subcommands and flags for working with the SQLite database format, such as **opm index prune**, do not work with the file-based catalog format. For more information about working with file-based catalogs, see Operator Framework packaging format, Managing custom catalogs, and Mirroring images for a disconnected installation by using the oc-mirror plugin v2.

10.4. ADDING A CATALOG SOURCE TO A CLUSTER

Adding a catalog source to an OpenShift Container Platform cluster enables the discovery and installation of Operators for users. Cluster administrators can create a **CatalogSource** object that references an index image. OperatorHub uses catalog sources to populate the user interface.

TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

Prerequisites

- You built and pushed an index image to a registry.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Create a **CatalogSource** object that references your index image. If you used the **oc adm catalog mirror** command to mirror your catalog to a target registry, you can use the generated **catalogSource.yaml** file in your manifests directory as a starting point.

a. Modify the following to your specifications and save it as a **catalogSource.yaml** file:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog (1)
  namespace: openshift-marketplace (2)
spec:
  sourceType: grpc
  grpcPodConfig:
    securityContextConfig: <security_mode> (3)
  image: <registry>/<namespace>/redhat-operator-index:v4.18 (4)
  displayName: My Operator Catalog
  publisher: <publisher_name> (5)
  updateStrategy:
    registryPoll: (6)
      interval: 30m
```

(1) If you mirrored content to local files before uploading to a registry, remove any backslash (/) characters from the **metadata.name** field to avoid an "invalid resource name" error when you create the object.

(2) If you want the catalog source to be available globally to users in all namespaces, specify the **openshift-marketplace** namespace. Otherwise, you can specify a different namespace for the catalog to be scoped and available only for that namespace.

(3) Specify the value of **legacy** or **restricted**. If the field is not set, the default value is **legacy**. In a future OpenShift Container Platform release, it is planned that the default value will be **restricted**.

NOTE

If your catalog cannot run with **restricted** permissions, it is recommended that you manually set this field to **legacy**.

(4) Specify your index image. If you specify a tag after the image name, for example **:v4.18**, the catalog source pod uses an image pull policy of **Always**, meaning the pod always pulls the image prior to starting the container. If you specify a digest, for example **@sha256:<id>**, the image pull policy is **IfNotPresent**, meaning the pod pulls the image only if it does not already exist on the node.

(5) Specify your name or an organization name publishing the catalog.

(6) Catalog sources can automatically check for new versions to keep up to date.

b. Use the file to create the **CatalogSource** object:

```
$ oc apply -f catalogSource.yaml
```

2. Verify the following resources are created successfully.

a. Check the pods:

```
$ oc get pods -n openshift-marketplace
```

Example output

```
NAME                                READY   STATUS    RESTARTS  AGE
my-operator-catalog-6njx6           1/1     Running   0         28s
marketplace-operator-d9f549946-96sgr  1/1     Running   0         26h
```

b. Check the catalog source:

```
$ oc get catalogsource -n openshift-marketplace
```

Example output

```
NAME                DISPLAY             TYPE PUBLISHER  AGE
my-operator-catalog  My Operator Catalog  grpc          5s
```

c. Check the package manifest:

```
$ oc get packagemanifest -n openshift-marketplace
```

Example output

```
NAME                    CATALOG             AGE
jaeger-product          My Operator Catalog  93s
```

You can now install the Operators from the **OperatorHub** page on your OpenShift Container Platform web console.

Additional resources

- Accessing images for Operators from private registries
- Image template for custom catalog sources
- Image pull policy

10.5. NEXT STEPS

- Updating installed Operators

CHAPTER 11. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT

11.1. ABOUT CLUSTER UPDATES IN A DISCONNECTED ENVIRONMENT

A disconnected environment is one in which your cluster nodes cannot access the internet or where you want to manage update recommendations and release images locally for policy or performance purposes. This section covers mirroring OpenShift Container Platform images, managing an OpenShift Update Service, and performing cluster updates in a disconnected environment.

11.1.1. Mirroring OpenShift Container Platform images

To update your cluster in a disconnected environment, your cluster environment must have access to a mirror registry that has the necessary images and resources for your targeted update. A single container image registry is sufficient to host mirrored images for several clusters in the disconnected network. The following page has instructions for mirroring images onto a repository in your disconnected cluster:

- Mirroring OpenShift Container Platform images

11.1.2. Performing a cluster update in a disconnected environment

You can use one of the following procedures to update a disconnected OpenShift Container Platform cluster:

- Updating a cluster in a disconnected environment using the OpenShift Update Service
- Updating a cluster in a disconnected environment without the OpenShift Update Service

11.1.3. Uninstalling the OpenShift Update Service from a cluster

You can use the following procedure to uninstall a local copy of the OpenShift Update Service (OSUS) from your cluster:

- Uninstalling the OpenShift Update Service from a cluster

11.2. MIRRORING OPENSHIFT CONTAINER PLATFORM IMAGES

You must mirror container images onto a mirror registry before you can update a cluster in a disconnected environment. You can also use this procedure in connected environments to ensure your clusters run only approved container images that have satisfied your organizational controls for external content.

NOTE

Your mirror registry must be running at all times while the cluster is running.

The following steps outline the high-level workflow on how to mirror images to a mirror registry:

1. Install the OpenShift CLI (**oc**) on all devices being used to retrieve and push release images.
2. Download the registry pull secret and add it to your cluster.

3. If you use the oc-mirror OpenShift CLI (**oc**) plugin:
    a. Install the oc-mirror plugin on all devices being used to retrieve and push release images.
    b. Create an image set configuration file for the plugin to use when determining which release images to mirror. You can edit this configuration file later to change which release images that the plugin mirrors.
    c. Mirror your targeted release images directly to a mirror registry, or to removable media and then to a mirror registry.
    d. Configure your cluster to use the resources generated by the oc-mirror plugin.
    e. Repeat these steps as needed to update your mirror registry.
4. If you use the **oc adm release mirror** command:
    a. Set environment variables that correspond to your environment and the release images you want to mirror.
    b. Mirror your targeted release images directly to a mirror registry, or to removable media and then to a mirror registry.
    c. Repeat these steps as needed to update your mirror registry.

Compared to using the **oc adm release mirror** command, the oc-mirror plugin has the following advantages:

- It can mirror content other than container images.
- After mirroring images for the first time, it is easier to update images in the registry.
- The oc-mirror plugin provides an automated way to mirror the release payload from Quay, and also builds the latest graph data image for the OpenShift Update Service running in the disconnected environment.

11.2.1. Mirroring resources using the oc-mirror plugin

You can use the oc-mirror OpenShift CLI (**oc**) plugin to mirror images to a mirror registry in your fully or partially disconnected environments. You must run oc-mirror from a system with internet connectivity to download the required images from the official Red Hat registries.

See Mirroring images for a disconnected installation by using the oc-mirror plugin v2 for additional details.

11.2.2. Mirroring images using the oc adm release mirror command

You can use the **oc adm release mirror** command to mirror images to your mirror registry.

11.2.2.1. Prerequisites

- You must have a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as Red Hat Quay.

NOTE

If you use Red Hat Quay, you must use version 3.6 or later with the oc-mirror plugin. If you have an entitlement to Red Hat Quay, see the documentation on deploying Red Hat Quay for proof-of-concept purposes or by using the Quay Operator. If you need additional assistance selecting and installing a registry, contact your sales representative or Red Hat Support.

If you do not have an existing solution for a container image registry, the mirror registry for Red Hat OpenShift is included in OpenShift Container Platform subscriptions. The *mirror registry for Red Hat OpenShift* is a small-scale container registry that you can use to mirror OpenShift Container Platform container images in disconnected installations and updates.

11.2.2.2. Preparing your mirror host

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

11.2.2.2.1. Installing the OpenShift CLI on Linux

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

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform 4.18. Download and install the new version of **oc**. If you are updating a cluster in a disconnected environment, install the **oc** version that you plan to update to.

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** drop-down list.
3. Select the appropriate version from the **Version** drop-down list.
4. Click **Download Now** next to the **OpenShift v4.18 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

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

```
$ oc <command>
```

11.2.2.2.2. Installing the OpenShift CLI on Windows

You can install the OpenShift CLI (**oc**) binary on Windows.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**. If you are updating a cluster in a disconnected environment, install the **oc** version that you plan to update to.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4.18 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

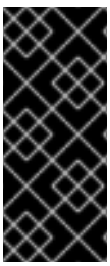
Verification

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

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

11.2.2.2.3. Installing the OpenShift CLI on macOS

You can install the OpenShift CLI (**oc**) binary on macOS.



IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**. If you are updating a cluster in a disconnected environment, install the **oc** version that you plan to update to.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.

3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.18 macOS Clients** entry and save the file.

NOTE

For macOS arm64, choose the **OpenShift v4.18 macOS arm64 Client** entry.

5. Unpack and unzip the archive.
6. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

Additional resources

- Installing and using CLI plugins

11.2.2.2.4. Configuring credentials that allow images to be mirrored

Create a container image registry credentials file so that you can mirror images from Red Hat to your mirror. Complete the following steps on the installation host.

WARNING

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

Prerequisites

- You configured a mirror registry to use in your disconnected environment.
- You identified an image repository location on your mirror registry to mirror images into.
- You provisioned a mirror registry account that allows images to be uploaded to that image repository.
- You have write access to the mirror registry.

Procedure

1. Download your **registry.redhat.io** pull secret from Red Hat OpenShift Cluster Manager .

2. Make a copy of your pull secret in JSON format by running the following command:

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

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

Example pull secret

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

1. Optional: If using the oc-mirror plugin, save the file as either **~/.docker/config.json** or **$XDG_RUNTIME_DIR/containers/auth.json**:

   - If the **.docker** or **$XDG_RUNTIME_DIR/containers** directories do not exist, create one by entering the following command:

     ```
     $ mkdir -p <directory_name>
     ```

     Where **<directory_name>** is either **~/.docker** or **$XDG_RUNTIME_DIR/containers**.

   - Copy the pull secret to the appropriate directory by entering the following command:

     ```
     $ cp <path>/<pull_secret_file_in_json> <directory_name>/<auth_file>
     ```

     Where **<directory_name>** is either **~/.docker** or **$XDG_RUNTIME_DIR/containers**, and **<auth_file>** is either **config.json** or **auth.json**.

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

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

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

Example output

```
BGVtbYk3ZHAtqXs=
```

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

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

- For the **<mirror_registry>** value, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example, **registry.example.com** or **registry.example.com:8443**.
- For the **<credentials>** value, specify the base64-encoded user name and password for the mirror registry.

Example modified pull secret

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

11.2.2.3. Mirroring images to a mirror registry

IMPORTANT

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

Prerequisites

- You configured a mirror registry to use in your disconnected environment and can access the certificate and credentials that you configured.
- You downloaded the pull secret from Red Hat OpenShift Cluster Manager and modified it to include authentication to your mirror repository.
- If you use self-signed certificates, you have specified a Subject Alternative Name in the certificates.

Procedure

1. Use the Red Hat OpenShift Container Platform Update Graph visualizer and update planner to plan an update from one version to another. The OpenShift Update Graph provides channel graphs and a way to confirm that there is an update path between your current and intended cluster versions.
2. Set the required environment variables:
   a. Export the release version:

   ```
   $ export OCP_RELEASE=<release_version>
   ```

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

   b. Export the local registry name and host port:

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

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

   c. Export the local repository name:

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

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

   d. If you are using the OpenShift Update Service, export an additional local repository name to contain the release images:

   ```
   $ LOCAL_RELEASE_IMAGES_REPOSITORY='<local_release_images_repository_name>'
   ```

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

e. Export the name of the repository to mirror:

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

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

f. Export the path to your registry pull secret:

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

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

NOTE

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

g. Export the release mirror:

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

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

h. Export the type of architecture for your cluster:

```
$ ARCHITECTURE=<cluster_architecture> 1
```

1 Specify the architecture of the cluster, such as **x86_64**, **aarch64**, **s390x**, or **ppc64le**.

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

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

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

3. Review the images and configuration manifests to mirror:

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

4. Mirror the version images to the mirror registry.

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

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

  ii. Mirror the images and configuration manifests to a directory on the removable media:

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

NOTE

This command also generates and saves the mirrored release image signature config map onto the removable media.

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

```
$ oc image mirror  -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} 1
```

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

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

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

```
$ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>
1
```

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

vi. If you are using the OpenShift Update Service, mirror the release image to a separate repository:

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

- If the local container registry and the cluster are connected to the mirror host, take the following actions:

i. Directly push the release images to the local registry and apply the config map to the cluster by using following command:

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

NOTE

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

ii. If you are using the OpenShift Update Service, mirror the release image to a separate repository:

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

11.3. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT USING THE OPENSHIFT UPDATE SERVICE

To get an update experience similar to connected clusters, you can use the following procedures to install and configure the OpenShift Update Service (OSUS) in a disconnected environment.

The following steps outline the high-level workflow on how to update a cluster in a disconnected environment using OSUS:

1. Configure access to a secured registry.
2. Update the global cluster pull secret to access your mirror registry.
3. Install the OSUS Operator.
4. Create a graph data container image for the OpenShift Update Service.
5. Install the OSUS application and configure your clusters to use the OpenShift Update Service in your environment.
6. Perform a supported update procedure from the documentation as you would with a connected cluster.

11.3.1. Using the OpenShift Update Service in a disconnected environment

The OpenShift Update Service (OSUS) provides update recommendations to OpenShift Container Platform clusters. Red Hat publicly hosts the OpenShift Update Service, and clusters in a connected environment can connect to the service through public APIs to retrieve update recommendations.

However, clusters in a disconnected environment cannot access these public APIs to retrieve update information. To have a similar update experience in a disconnected environment, you can install and configure the OpenShift Update Service so that it is available within the disconnected environment.

A single OSUS instance is capable of serving recommendations to thousands of clusters. OSUS can be scaled horizontally to cater to more clusters by changing the replica value. So for most disconnected use cases, one OSUS instance is enough. For example, Red Hat hosts just one OSUS instance for the entire fleet of connected clusters.

If you want to keep update recommendations separate in different environments, you can run one OSUS instance for each environment. For example, in a case where you have separate test and stage environments, you might not want a cluster in a stage environment to receive update recommendations

to version A if that version has not been tested in the test environment yet.

The following sections describe how to install an OSUS instance and configure it to provide update recommendations to a cluster.

Additional resources

- About the OpenShift Update Service
- Understanding update channels and releases

11.3.2. Prerequisites

- You must have the **oc** command-line interface (CLI) tool installed.
- You must provision a container image registry in your environment with the container images for your update, as described in Mirroring OpenShift Container Platform images .

11.3.3. Configuring access to a secured registry for the OpenShift Update Service

If the release images are contained in a registry whose HTTPS X.509 certificate is signed by a custom certificate authority, complete the steps in Configuring additional trust stores for image registry access along with following changes for the update service.

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

Image registry CA config map example for the update service

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  updateservice-registry: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 2
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1. The OpenShift Update Service Operator requires the config map key name **updateservice-registry** in the registry CA cert.
2. If the registry has the port, such as **registry-with-port.example.com:5000**, **:** should be replaced with **..**.

11.3.4. Updating the global cluster pull secret

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

Use the procedure when you need a separate registry to store images than the registry used during installation.

Prerequisites

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

Procedure

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

   a. Enter the following command to download the pull secret:

   ```
   $ oc get secret/pull-secret -n openshift-config --template='{{index .data
   ".dockerconfigjson" | base64decode}}' > <pull_secret_location> 1
   ```

   1 **<pull_secret_location>**: Include the path to the pull secret file.

   b. Enter the following command to add the new pull secret:

   ```
   $ oc registry login --registry="<registry>" \ 1
   --auth-basic="<username>:<password>" \ 2
   --to=<pull_secret_location> 3
   ```

   1 **<registry>**: Include the new registry. You can include many repositories within the same registry, for example: **--registry="<registry/my-namespace/my-repository>**.

   2 **<username>:<password>**: Include the credentials of the new registry.

   3 **<pull_secret_location>**: Include the path to the pull secret file.

   You can also perform a manual update to the pull secret file.

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

   ```
   $ oc set data secret/pull-secret -n openshift-config \
     --from-file=.dockerconfigjson=<pull_secret_location> 1
   ```

   1 **<pull_secret_location>**: Include the path to the new pull secret file.

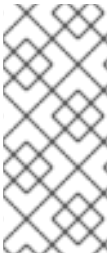
   This update rolls out to all nodes, which can take some time depending on the size of your cluster.

NOTE

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

11.3.5. Installing the OpenShift Update Service Operator

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



NOTE

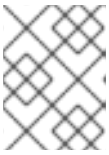
For clusters that are installed in disconnected environments, also known as disconnected clusters, Operator Lifecycle Manager by default cannot access the Red Hat-provided OperatorHub sources hosted on remote registries because those remote sources require full internet connectivity. For more information,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

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

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

Procedure

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



NOTE

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

2. Choose **OpenShift Update Service** from the list of available Operators, and click **Install**.
   a. Select an **Update channel**.
   b. Select a **Version**.
   c. Select **A specific namespace on the cluster** under **Installation Mode**.
   d. Select a namespace for **Installed Namespace** or accept the recommended namespace **openshift-update-service**.
   e. Select an **Update approval** strategy:
      - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
      - The **Manual** strategy requires a cluster administrator to approve the Operator update.
   f. Click **Install**.
3. Go to **Operators → Installed Operators** and verify that the OpenShift Update Service Operator is installed.
4. Ensure that **OpenShift Update Service** is listed in the correct namespace with a **Status** of **Succeeded**.

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

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

Procedure

1. Create a namespace for the OpenShift Update Service Operator:

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

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

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

   b. Create the namespace:

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

   For example:

   ```
   $ oc create -f update-service-namespace.yaml
   ```

2. Install the OpenShift Update Service Operator by creating the following objects:

   a. Create an **OperatorGroup** object YAML file, for example, **update-service-operator-group.yaml**:

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

   b. Create an **OperatorGroup** object:

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

   For example:

   ```
   $ oc -n openshift-update-service create -f update-service-operator-group.yaml
   ```

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

   Example Subscription

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   ```

```
metadata:
  name: update-service-subscription
  namespace: openshift-update-service
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators" (1)
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"
```

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

d. Create the **Subscription** object:

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

For example:

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

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

3. Verify the Operator installation:

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

Example output

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

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

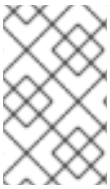
Additional resources

- Installing Operators in your namespace .

11.3.6. Creating the OpenShift Update Service graph data container image

The OpenShift Update Service requires a graph data container image, from which the OpenShift Update Service retrieves information about channel membership and blocked update edges. Graph data is typically fetched directly from the update graph data repository. In environments where an internet connection is unavailable, loading this information from an init container is another way to make the

graph data available to the OpenShift Update Service. The role of the init container is to provide a local copy of the graph data, and during pod initialization, the init container copies the data to a volume that is accessible by the service.



NOTE

The oc-mirror OpenShift CLI (**oc**) plugin creates this graph data container image in addition to mirroring release images. If you used the oc-mirror plugin to mirror your release images, you can skip this procedure.

Procedure

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

```
FROM registry.access.redhat.com/ubi9/ubi:latest

RUN curl -L -o cincinnati-graph-data.tar.gz
https://api.openshift.com/api/upgrades_info/graph-data

RUN mkdir -p /var/lib/cincinnati-graph-data && tar xvzf cincinnati-graph-data.tar.gz -C
/var/lib/cincinnati-graph-data/ --no-overwrite-dir --no-same-owner

CMD ["/bin/bash", "-c" ,"exec cp -rp /var/lib/cincinnati-graph-data/* /var/lib/cincinnati/graph-
data"]
```

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

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

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

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

NOTE

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

11.3.7. Creating an OpenShift Update Service application

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

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

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

Prerequisites

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

Procedure

1. In the web console, click **Operators → Installed Operators**.
2. Choose **OpenShift Update Service** from the list of installed Operators.
3. Click the **Update Service** tab.
4. Click **Create UpdateService**.
5. Enter a name in the **Name** field, for example, **service**.
6. Enter the local pullspec in the **Graph Data Image** field to the graph data container image created in "Creating the OpenShift Update Service graph data container image", for example, **registry.example.com/openshift/graph-data:latest**.
7. In the **Releases** field, enter the registry and repository created to contain the release images in "Mirroring the OpenShift Container Platform image repository", for example, **registry.example.com/ocp4/openshift4-release-images**.
8. Enter **2** in the **Replicas** field.
9. Click **Create** to create the OpenShift Update Service application.
10. Verify the OpenShift Update Service application:
    - From the **UpdateServices** list in the **Update Service** tab, click the Update Service application just created.
    - Click the **Resources** tab.
    - Verify each application resource has a status of **Created**.

11.3.7.2. Creating an OpenShift Update Service application by using the CLI

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

Prerequisites

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

Procedure

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

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

   The namespace must match the **targetNamespaces** value from the operator group.

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

```
$ NAME=service
```

3. Configure the registry and repository for the release images as configured in "Mirroring the OpenShift Container Platform image repository", for example, **registry.example.com/ocp4/openshift4-release-images**:

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

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

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

5. Create an OpenShift Update Service application object:

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

6. Verify the OpenShift Update Service application:

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

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

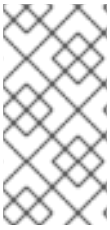
   You might need to poll until the command succeeds.

   b. Retrieve a graph from the policy engine. Be sure to specify a valid version for **channel**. For example, if running in OpenShift Container Platform 4.18, use **stable-4.18**:

```
$ while sleep 10; do HTTP_CODE="$(curl --header Accept:application/json --output
/dev/stderr --write-out "%{http_code}" "${POLICY_ENGINE_GRAPH_URI}?
```

```
channel=stable-4.6")"; if test "${HTTP_CODE}" -eq 200; then break; fi; echo
"${HTTP_CODE}"; done
```

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



NOTE

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

11.3.8. Configuring the Cluster Version Operator (CVO)

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

Prerequisites

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

Procedure

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

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

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

```
$ NAME=service
```

3. Obtain the policy engine route:

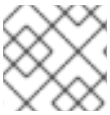
```
$ POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
"${NAME}")"
```

4. Set the patch for the pull graph data:

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

5. Patch the CVO to use the OpenShift Update Service in your environment:

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



NOTE

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

11.3.9. Next steps

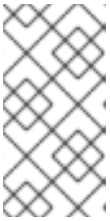
Before updating your cluster, confirm that the following conditions are met:

- The Cluster Version Operator (CVO) is configured to use your installed OpenShift Update Service application.
- The release image signature config map for the new release is applied to your cluster.

NOTE

The Cluster Version Operator (CVO) uses release image signatures to ensure that release images have not been modified, by verifying that the release image signatures match the expected result.

- The current release and update target release images are mirrored to a registry in the disconnected environment.
- A recent graph data container image has been mirrored to your registry.
- A recent version of the OpenShift Update Service Operator is installed.



NOTE

If you have not recently installed or updated the OpenShift Update Service Operator, there might be a more recent version available. See Using Operator Lifecycle Manager in disconnected environments for more information about how to update your OLM catalog in a disconnected environment.

After you configure your cluster to use the installed OpenShift Update Service and local mirror registry, you can use any of the following update methods:

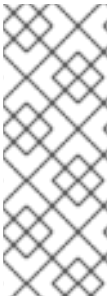
- Updating a cluster using the web console
- Updating a cluster using the CLI
- Performing a Control Plane Only update
- Performing a canary rollout update
- Updating a cluster that includes RHEL compute machines

11.4. UPDATING A CLUSTER IN A DISCONNECTED ENVIRONMENT WITHOUT THE OPENSHIFT UPDATE SERVICE

Use the following procedures to update a cluster in a disconnected environment without access to the OpenShift Update Service.

11.4.1. Prerequisites

- You must have the **oc** command-line interface (CLI) tool installed.
- You must provision a local container image registry with the container images for your update, as described in Mirroring OpenShift Container Platform images .
- You must have access to the cluster as a user with **admin** privileges. See Using RBAC to define and apply permissions.
- You must have a recent etcd backup in case your update fails and you must restore your cluster to a previous state.
- 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.
- You must 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 .
- 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.



NOTE

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.

11.4.2. 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-
```

11.4.3. Retrieving a release image digest

In order to update a cluster in a disconnected environment using the **oc adm upgrade** command with the **--to-image** option, you must reference the sha256 digest that corresponds to your targeted release image.

Procedure

1. Run the following command on a device that is connected to the internet:

```
$ oc adm release info -o 'jsonpath={.digest}{"\n"}' quay.io/openshift-release-dev/ocp-release:${OCP_RELEASE_VERSION}-${ARCHITECTURE}
```

   For **{OCP_RELEASE_VERSION}**, specify the version of OpenShift Container Platform to which you want to update, such as **4.10.16**.

   For **{ARCHITECTURE}**, specify the architecture of the cluster, such as **x86_64**, **aarch64**, **s390x**, or **ppc64le**.

   Example output

```
sha256:a8bfba3b6dddd1a2fbbead7dac65fe4fb8335089e4e7cae327f3bad334add31d
```

2. Copy the sha256 digest for use when updating your cluster.

11.4.4. Updating the disconnected cluster

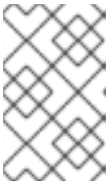
Update the disconnected cluster to the OpenShift Container Platform version that you downloaded the release images for.

NOTE

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

Prerequisites

- You mirrored the images for the new release to your registry.
- You applied the release image signature ConfigMap for the new release to your cluster.



NOTE

The release image signature config map allows the Cluster Version Operator (CVO) to ensure the integrity of release images by verifying that the actual image signatures match the expected signatures.

- You obtained the sha256 digest for your targeted release image.
- You installed the OpenShift CLI (**oc**).
- You paused all **MachineHealthCheck** resources.

Procedure

- Update the cluster:

```
$ oc adm upgrade --allow-explicit-upgrade --to-image
<defined_registry>/<defined_repository>@<digest>
```

Where:

<defined_registry>

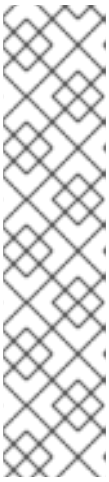
Specifies the name of the mirror registry you mirrored your images to.

<defined_repository>

Specifies the name of the image repository you want to use on the mirror registry.

<digest>

Specifies the sha256 digest for the targeted release image, for example, **sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92**.



NOTE

- See "Mirroring OpenShift Container Platform images" to review how your mirror registry and repository names are defined.
- If you used an **ImageContentSourcePolicy** or **ImageDigestMirrorSet**, you can use the canonical registry and repository names instead of the names you defined. The canonical registry name is **quay.io** and the canonical repository name is **openshift-release-dev/ocp-release**.
- You can only configure global pull secrets for clusters that have an **ImageContentSourcePolicy**, **ImageDigestMirrorSet**, or **ImageTagMirrorSet** object. You cannot add a pull secret to a project.

Additional resources

- Mirroring OpenShift Container Platform images

11.4.5. Understanding image registry repository mirroring

By setting up container registry repository mirroring, you can perform the following tasks:

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

Repository mirroring in OpenShift Container Platform includes the following attributes:

- Image pulls are resilient to registry downtimes.
- Clusters in disconnected environments can pull images from critical locations, such as **quay.io**, and have registries behind a company firewall provide the requested images.
- A particular order of registries is tried when an image pull request is made, with the permanent registry typically being the last one tried.

- The mirror information you enter is added to the **/etc/containers/registries.conf** file on every node in the OpenShift Container Platform cluster.
- When a node makes a request for an image from the source repository, it tries each mirrored repository in turn until it finds the requested content. If all mirrors fail, the cluster tries the source repository. If successful, the image is pulled to the node.

You can set up repository mirroring in the following ways:

- At OpenShift Container Platform installation:
  By pulling container images needed by OpenShift Container Platform and then bringing those images behind your company's firewall, you can install OpenShift Container Platform into a data center that is in a disconnected environment.
- After OpenShift Container Platform installation:
  If you did not configure mirroring during OpenShift Container Platform installation, you can do so postinstallation by using any of the following custom resource (CR) objects:
  - **ImageDigestMirrorSet** (IDMS). This object allows you to pull images from a mirrored registry by using digest specifications. The IDMS CR enables you to set a fall back policy that allows or stops continued attempts to pull from the source registry if the image pull fails.
  - **ImageTagMirrorSet** (ITMS). This object allows you to pull images from a mirrored registry by using image tags. The ITMS CR enables you to set a fall back policy that allows or stops continued attempts to pull from the source registry if the image pull fails.
  - **ImageContentSourcePolicy** (ICSP). This object allows you to pull images from a mirrored registry by using digest specifications. The ICSP CR always falls back to the source registry if the mirrors do not work.

IMPORTANT

Using an **ImageContentSourcePolicy** (ICSP) object to configure repository mirroring is a deprecated feature. Deprecated functionality is still included in OpenShift Container Platform and continues to be supported. It will be removed in a future release and is not recommended for new deployments.

If you have existing YAML files that you used to create **ImageContentSourcePolicy** objects, you can use the **oc adm migrate icsp** command to convert those files to a **ImageDigestMirrorSet** YAML files. For more information, see "Converting ImageContentSourcePolicy (ICSP) files for image registry repository mirroring".

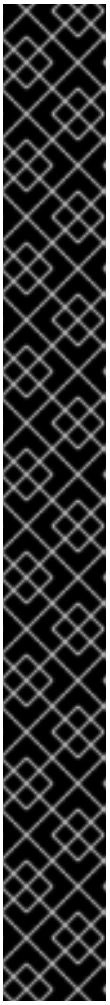
Each of these custom resource objects identify the following information:

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

Note the following actions and how they affect node drain behavior:

- If you create an IDMS or ICSP CR object, the MCO does not drain or reboot the node.
- If you create an ITMS CR object, the MCO drains and reboots the node.

- If you delete an ITMS, IDMS, or ICSP CR object, the MCO drains and reboots the node.
- If you modify an ITMS, IDMS, or ICSP CR object, the MCO drains and reboots the node.



IMPORTANT

- When the MCO detects any of the following changes, it applies the update without draining or rebooting the node:
  - Changes to the SSH key in the **spec.config.passwd.users.sshAuthorizedKeys** parameter of a machine config.
  - Changes to the global pull secret or pull secret in the **openshift-config** namespace.
  - Automatic rotation of the **/etc/kubernetes/kubelet-ca.crt** certificate authority (CA) by the Kubernetes API Server Operator.
- When the MCO detects changes to the **/etc/containers/registries.conf** file, such as editing an **ImageDigestMirrorSet**, **ImageTagMirrorSet**, or **ImageContentSourcePolicy** object, it drains the corresponding nodes, applies the changes, and uncordons the nodes. The node drain does not happen for the following changes:
  - The addition of a registry with the **pull-from-mirror = "digest-only"** parameter set for each mirror.
  - The addition of a mirror with the **pull-from-mirror = "digest-only"** parameter set in a registry.
  - The addition of items to the **unqualified-search-registries** list.

For new clusters, you can use IDMS, ITMS, and ICSP CRs objects as needed. However, using IDMS and ITMS is recommended.

If you upgraded a cluster, any existing ICSP objects remain stable, and both IDMS and ICSP objects are supported. Workloads that use ICSP objects continue to function as expected. However, if you want to take advantage of the fallback policies introduced in the IDMS CRs, you can migrate current workloads to IDMS objects by using the **oc adm migrate icsp** command as shown in the **Converting ImageContentSourcePolicy (ICSP) files for image registry repository mirroring** section that follows. Migrating to IDMS objects does not require a cluster reboot.

NOTE

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

11.4.5.1. Configuring image registry repository mirroring

You can create postinstallation mirror configuration custom resources (CR) to redirect image pull requests from a source image registry to a mirrored image registry.

Prerequisites

.

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

Procedure

1. Configure mirrored repositories, by either:

   - Setting up a mirrored repository with Red Hat Quay. You can copy images from one repository to another and also automatically sync those repositories repeatedly over time by using Red Hat Quay.

     - Red Hat Quay Repository Mirroring

   - Using a tool such as **skopeo** to copy images manually from the source repository to the mirrored repository.
     For example, after installing the skopeo RPM package on a {op-system-base-full system}, use the **skopeo** command as shown in the following example:

     ```
     $ skopeo copy --all \
     docker://registry.access.redhat.com/ubi9/ubi-minimal:latest@sha256:5cf... \
     docker://example.io/example/ubi-minimal
     ```

     In this example, you have a container image registry named **example.io** and image repository named **example**. You want to copy the **ubi9/ubi-minimal** image from **registry.access.redhat.com** to **example.io**. After you create the mirrored registry, you can configure your OpenShift Container Platform cluster to redirect requests made to the source repository to the mirrored repository.

2. Create a postinstallation mirror configuration custom resource (CR), by using one of the following examples:

   - Create an **ImageDigestMirrorSet** or **ImageTagMirrorSet** CR, as needed, replacing the source and mirrors with your own registry and repository pairs and images:

     ```
     apiVersion: config.openshift.io/v1
     kind: ImageDigestMirrorSet
     metadata:
       name: ubi9repo
     spec:
       imageDigestMirrors:
       - mirrors:
         - example.io/example/ubi-minimal
         - example.com/example2/ubi-minimal
         source: registry.access.redhat.com/ubi9/ubi-minimal
         mirrorSourcePolicy: AllowContactingSource
       - mirrors:
         - mirror.example.com/redhat
         source: registry.example.com/redhat
         mirrorSourcePolicy: AllowContactingSource
       - mirrors:
         - mirror.example.com
         source: registry.example.com
         mirrorSourcePolicy: AllowContactingSource
       - mirrors:
         - mirror.example.net/image
         source: registry.example.com/example/myimage
     ```

```
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net/registry-example-com
    source: registry.example.com
    mirrorSourcePolicy: AllowContactingSource
```

- Create an **ImageContentSourcePolicy** custom resource, replacing the source and mirrors with your own registry and repository pairs and images:

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mirror-ocp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:443/ocp/release
    source: quay.io/openshift-release-dev/ocp-release
  - mirrors:
    - mirror.registry.com:443/ocp/release
    source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

where:

- mirror.registry.com:443/ocp/release

Specifies the name of the mirror image registry and repository.

source: quay.io/openshift-release-dev/ocp-release

Specifies the online registry and repository containing the content that is mirrored.

3. Create the new object by running the following command:

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

After the object is created, the Machine Config Operator (MCO) drains the nodes for **ImageTagMirrorSet** objects only. The MCO does not drain the nodes for **ImageDigestMirrorSet** and **ImageContentSourcePolicy** objects.

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

   a. List your nodes:

```
$ oc get node
```

Example output

```
NAME                          STATUS                     ROLES    AGE  VERSION
ip-10-0-137-44.ec2.internal   Ready                      worker   7m   v1.31.3
ip-10-0-138-148.ec2.internal  Ready                      master   11m  v1.31.3
ip-10-0-139-122.ec2.internal  Ready                      master   11m  v1.31.3
```

```
ip-10-0-147-35.ec2.internal   Ready    worker   7m   v1.31.3
ip-10-0-153-12.ec2.internal   Ready    worker   7m   v1.31.3
ip-10-0-154-10.ec2.internal   Ready    master   11m  v1.31.3
```

b. Start the debugging process to access the node:

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

Example output

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

c. Change your root directory to **/host**:

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

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

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

The following output represents a **registries.conf** file where postinstallation mirror configuration CRs are applied.

Example output

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

[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi9/ubi-minimal"

  [[registry.mirror]]
    location = "example.io/example/ubi-minimal"
    pull-from-mirror = "digest-only"

  [[registry.mirror]]
    location = "example.com/example/ubi-minimal"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com"

  [[registry.mirror]]
    location = "mirror.example.net/registry-example-com"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/example"

  [[registry.mirror]]
```

```
    location = "mirror.example.net"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/example/myimage"

  [[registry.mirror]]
    location = "mirror.example.net/image"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com"

  [[registry.mirror]]
    location = "mirror.example.com"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/redhat"

  [[registry.mirror]]
    location = "mirror.example.com/redhat"
    pull-from-mirror = "digest-only"
[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi9/ubi-minimal"
  blocked = true

  [[registry.mirror]]
    location = "example.io/example/ubi-minimal-tag"
    pull-from-mirror = "tag-only"
```

where: **[[registry]].location = "registry.access.redhat.com/ubi9/ubi-minimal"**:: The repository listed in a pull spec. **[[registry.mirror]].location = "example.io/example/ubi-minimal"**:: Indicates the mirror for that repository. **[[registry.mirror]].pull-from-mirror = "digest-only"**:: Means that the image pull from the mirror is a digest reference image. **[[registry]].blocked = true**:: Indicates that the **NeverContactSource** parameter is set for this repository. **[[registry.mirror]].pull-from-mirror = "tag-only"**:: Indicates that the image pull from the mirror is a tag reference image.

e. Pull an image to the node from the source and check if it is resolved by the mirror.

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi9/ubi-minimal@sha256:5cf...
```

Troubleshooting

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

- The first working mirror is used to supply the pulled image.
- The main registry is only used if no other mirror works.

- From the system context, the **Insecure** flags are used as fallback.
- The format of the **/etc/containers/registries.conf** file has changed recently. It is now version 2 and in TOML format.

11.4.5.2. Converting ImageContentSourcePolicy (ICSP) files for image registry repository mirroring

Using an **ImageContentSourcePolicy** (ICSP) object to configure repository mirroring is a deprecated feature.

This functionality is still included in OpenShift Container Platform and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments.

ICSP objects are being replaced by **ImageDigestMirrorSet** and **ImageTagMirrorSet** objects to configure repository mirroring. If you have existing YAML files that you used to create **ImageContentSourcePolicy** objects, you can use the **oc adm migrate icsp** command to convert those files to an **ImageDigestMirrorSet** YAML file. The command updates the API to the current version, changes the **kind** value to **ImageDigestMirrorSet**, and changes **spec.repositoryDigestMirrors** to **spec.imageDigestMirrors**. The rest of the file is not changed.

Because the migration does not change the **registries.conf** file, the cluster does not need to reboot.

For more information about **ImageDigestMirrorSet** or **ImageTagMirrorSet** objects, see "Configuring image registry repository mirroring" in the previous section.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- Ensure that you have **ImageContentSourcePolicy** objects on your cluster.

Procedure

1. Use the following command to convert one or more **ImageContentSourcePolicy** YAML files to an **ImageDigestMirrorSet** YAML file:

```
$ oc adm migrate icsp <file_name>.yaml <file_name>.yaml <file_name>.yaml --dest-dir
<path_to_the_directory>
```

   where:

   <file_name>

   Specifies the name of the source **ImageContentSourcePolicy** YAML. You can list multiple file names.

   --dest-dir

   Optional: Specifies a directory for the output **ImageDigestMirrorSet** YAML. If unset, the file is written to the current directory.
   For example, the following command converts the **icsp.yaml** and **icsp-2.yaml** file and saves the new YAML files to the **idms-files** directory.

```
$ oc adm migrate icsp icsp.yaml icsp-2.yaml --dest-dir idms-files
```

-

Example output

```
wrote ImageDigestMirrorSet to idms-files/imagedigestmirrorset_ubi8repo.5911620242173376087.yaml
wrote ImageDigestMirrorSet to idms-files/imagedigestmirrorset_ubi9repo.6456931852378115011.yaml
```

2. Create the CR object by running the following command:

```
$ oc create -f <path_to_the_directory>/<file-name>.yaml
```

where:

<path_to_the_directory>

Specifies the path to the directory, if you used the **--dest-dir** flag.

<file_name>

Specifies the name of the **ImageDigestMirrorSet** YAML.

3. Remove the ICSP objects after the IDMS objects are rolled out.

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

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

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

Prerequisites

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

Procedure

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

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

where:

<local_registry>

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

<pull_spec>

is the pull specification as configured in your disconnected registry, for example, **redhat/redhat-operator-index:v4.18**

<pull_secret_file>

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

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

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

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

Verification

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

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

Example output

```
apiVersion: v1
items:
- apiVersion: operator.openshift.io/v1alpha1
  kind: ImageContentSourcePolicy
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"operator.openshift.io/v1alpha1","kind":"ImageContentSourcePolicy","metadata":
{"annotations":{},"name":"redhat-operator-index"},"spec":{"repositoryDigestMirrors":
[{"mirrors":["local.registry:5000"],"source":"registry.redhat.io"}]}}
...
```

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

11.4.7. Additional resources

- Using Operator Lifecycle Manager in disconnected environments
- Machine Config Overview

11.5. UNINSTALLING THE OPENSHIFT UPDATE SERVICE FROM A CLUSTER

To remove a local copy of the OpenShift Update Service (OSUS) from your cluster, you must first delete the OSUS application and then uninstall the OSUS Operator.

11.5.1. Deleting an OpenShift Update Service application

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

11.5.1.1. Deleting an OpenShift Update Service application by using the web console

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

Prerequisites

- The OpenShift Update Service Operator has been installed.

Procedure

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

11.5.1.2. Deleting an OpenShift Update Service application by using the CLI

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

Procedure

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

```
$ oc get updateservice -n openshift-update-service
```

Example output

```
NAME      AGE
service   6s
```

2. Delete the OpenShift Update Service application using the **NAME** value from the previous step and the namespace the OpenShift Update Service application was created in, for example, **openshift-update-service**:

```
$ oc delete updateservice service -n openshift-update-service
```

Example output

```
updateservice.updateservice.operator.openshift.io "service" deleted
```

11.5.2. Uninstalling the OpenShift Update Service Operator

You can uninstall the OpenShift Update Service Operator by using the OpenShift Container Platform web console or CLI.

11.5.2.1. Uninstalling the OpenShift Update Service Operator by using the web console

You can use the OpenShift Container Platform web console to uninstall the OpenShift Update Service Operator.

Prerequisites

- All OpenShift Update Service applications have been deleted.

Procedure

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

11.5.2.2. Uninstalling the OpenShift Update Service Operator by using the CLI

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

Prerequisites

- All OpenShift Update Service applications have been deleted.

Procedure

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

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

Example output

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

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

```
$ oc get operatorgroup
```

Example output

```
NAME                              AGE
openshift-update-service-fprx2    4m41s
```

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

```
$ oc delete operatorgroup openshift-update-service-fprx2
```

Example output

```
operatorgroup.operators.coreos.com "openshift-update-service-fprx2" deleted
```

4. Get the name of the OpenShift Update Service Operator subscription:

```
$ oc get subscription
```

Example output

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

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

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

Example output

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

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

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

Example output

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

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

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

Example output

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