OpenShift Container Platform 4.17

Disconnected installation mirroring

Mirroring the installation container images

Last Updated: 2024-09-30

OpenShift Container Platform 4.17 Disconnected installation mirroring

Mirroring the installation container images

Legal Notice

Abstract

This document describes how to mirror the installation container images for a disconnected OpenShift Container Platform installation.

Table of Contents

CHAPTER 1. ABOUT DISCONNECTED INSTALLATION MIRRORING

You can use a mirror registry to ensure that 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. To mirror container images, you must have a registry for mirroring.

1.1. CREATING 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 can create a mirror registry using the *mirror registry for Red Hat OpenShift*.

1.2. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION

You can use one of the following procedures to mirror your OpenShift Container Platform image repository to your mirror registry:

- Mirroring images for a disconnected installation
- Mirroring images for a disconnected installation using the oc-mirror plugin

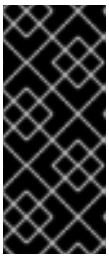
CHAPTER 2. CREATING A MIRROR REGISTRY WITH MIRROR REGISTRY FOR RED HAT OPENSHIFT

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 .

2.1. 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 4.17 release images, or about 358 GB for OpenShift Container Platform 4.17 release images and OpenShift Container Platform 4.17 Red Hat Operator images. Up to 1 TB per stream or more is suggested.



IMPORTANT

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.

2.2. 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 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 using the **mirror-registry** command line interface (CLI) tool. The *mirror registry for Red Hat OpenShift* is deployed automatically with preconfigured 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

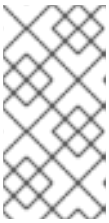
inputs like fully qualified domain name (FQDN) services, superuser name and password, and custom TLS certificates are also provided. This provides users with a container registry so that they can easily create an offline mirror of all OpenShift Container Platform release content when running OpenShift 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.

2.2.1. 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. It is advised to leverage 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.

2.3. 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 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.

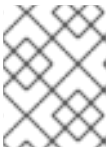
2. Install the *mirror registry for Red Hat OpenShift* on your local host with your current user account by using the **mirror-registry** tool. For a full list of available flags, see "mirror registry for Red Hat OpenShift flags".

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

3. Use the user name and password generated during installation to log into 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 "Using SSL to protect connections to 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 remirror the OpenShift Container Platform images, or reinstall mirror registry on more stable storage.

2.4. 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 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-instal/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 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:

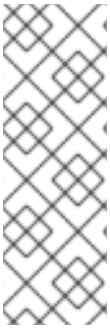
```
$ sudo ./mirror-registry upgrade --quayHostname <host_example_com> --quayRoot
<example_directory_name>  --quayStorage <example_directory_name>/quay-storage -v
```

- 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
```

2.5. 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 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 user name and password generated during installation to log into the mirror registry by running the following command:

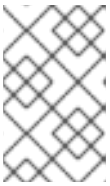
```
$ podman login -u init \
  -p <password> \
  <host_example_com>:8443> \
  --tls-verify=false 1
```

1 You can avoid running **--tls-verify=false** by configuring your system to trust the generated rootCA certificates. See "Using SSL to protect connections to 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 remirror the OpenShift Container Platform images, or reinstall mirror registry on more stable storage.

2.6. 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 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-instal/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
```

2.7. 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 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 Using SSL/TLS to protect connections to 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 restart quay-app
```

2.8. UNINSTALLING THE MIRROR REGISTRY FOR RED HAT OPENSHIFT

- You can 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* will prompt 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.

2.9. 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.
--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]

Flags	Description
--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.

2.10. 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.

2.10.1. 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

2.10.1.1. 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-2023:5277 - mirror registry for Red Hat OpenShift 2.0.0

2.10.1.1.1. New features

- 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.

2.10.2. 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.

2.10.3. 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.

2.10.4. 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.

2.11. 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-postgres.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:

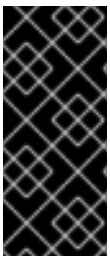
```
$ systemctl --user status <service>
```

2.12. ADDITIONAL RESOURCES

- Red Hat Quay garbage collection
- Using SSL to protect connections to 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 3. MIRRORING IMAGES FOR A DISCONNECTED INSTALLATION

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. 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.

3.1. PREREQUISITES

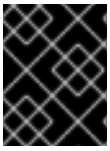
- You must have a container image registry that supports Docker v2-2 in the location that will host the OpenShift Container Platform cluster, such as one of the following registries:
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus Repository
  - Harbor

  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.

3.2. ABOUT THE MIRROR REGISTRY

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

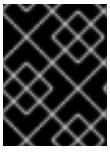
For information about viewing the CRI-O logs to view the image source, see Viewing the image pull source.

3.3. 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.

3.3.1. Installing the OpenShift CLI

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 4.17. Download and install the new version of **oc**.

Installing the OpenShift CLI on Linux

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

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 v4.17 Linux Client** 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>
```

Installing the OpenShift CLI on Windows

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

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.17 Windows Client** entry and save the file.
4. Unzip the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

Verification

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

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

Installing the OpenShift CLI on macOS

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

Procedure

1. Navigate to the OpenShift Container Platform downloads page on the Red Hat Customer Portal.

2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4.17 macOS Client** entry and save the file.

NOTE

For macOS arm64, choose the **OpenShift v4.17 macOS arm64 Client** entry.

4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

3.4. CONFIGURING CREDENTIALS THAT ALLOW IMAGES TO BE MIRRORED

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

WARNING

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

WARNING

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

Prerequisites

- You configured a mirror registry to use in your 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.

Procedure

Complete the following steps on the installation host:

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:

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

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

The contents of the file resemble the following example:

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

```
$ echo -n '<user_name>:<password>' | base64 -w0 1
BGVtbYk3ZHAtqXs=
```

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

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

```
  "auths": {
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
```

```
      "email": "you@example.com"
    }
  },
```

1. 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**
2. Specify the base64-encoded user name and password for the mirror registry.

The file resembles the following example:

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

3.5. MIRRORING THE OPENSHIFT CONTAINER PLATFORM IMAGE REPOSITORY

Mirror the OpenShift Container Platform image repository to your registry to use during cluster installation or upgrade.

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

Complete the following steps on the mirror host:

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 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 **4.5.4**.

   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. 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> 1
   ```

   1 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> (1)
```

(1) 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} (1)
```

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

IMPORTANT

Running **oc image 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.

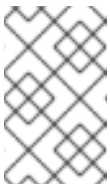
- If the local container registry is connected to the mirror host, take the following actions:

  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:

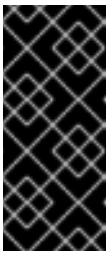
- If your mirror host does not have internet access, run the following command:

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --icsp-file=<file> --
command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}" \
--insecure=true (1)
```

(1) 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
```

3.6. THE CLUSTER SAMPLES OPERATOR IN A DISCONNECTED ENVIRONMENT

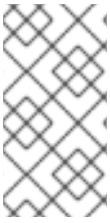
In a disconnected environment, you must take additional steps after you install a cluster to configure the Cluster Samples Operator. Review the following information in preparation.

3.6.1. 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 may 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's objects to reach the services they require.

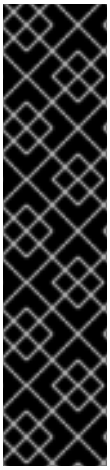
You can use this config map as a reference for which images need to be mirrored for your image streams to import.

- 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.

3.7. 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 on restricted networks

3.7.1. 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
```

3.7.2. 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.

3.7.2.1. 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

3.7.2.2. 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 **.***

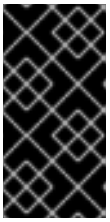
NOTE

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

3.7.3. 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:

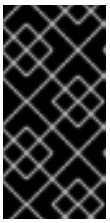
```
manifests-index/<repository>/<index_image_name>-<random_number>
```

NOTE

The manifests directory name is referenced in subsequent procedures.

The manifests directory contains the following files, some of which might require further modification:

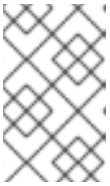
- The **catalogSource.yaml** file is a basic definition for a **CatalogSource** object that is pre-populated with your index image tag and other relevant metadata. This file can be used as is or modified to add the catalog source to your cluster.



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.

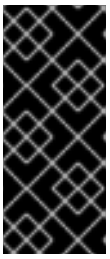
- The **imageContentSourcePolicy.yaml** file defines an **ImageContentSourcePolicy** object that can configure nodes to translate between the image references stored in Operator manifests and the mirrored registry.



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.

3.7.4. 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

3.8. 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.

3.9. ADDITIONAL RESOURCES

- See Gathering data about specific features for more information about using must-gather.

CHAPTER 4. 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.

4.1. 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.

4.1.1. 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.

4.2. 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.

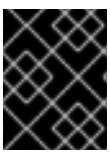
Additional resources

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

4.3. ABOUT THE MIRROR REGISTRY

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

- For information about viewing the CRI-O logs to view the image source, see Viewing the image pull source.

4.4. 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.

4.5. 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.

4.5.1. 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.

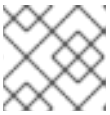
Procedure

1. Download the oc-mirror CLI plugin.
   a. Navigate to the Downloads page of the OpenShift Cluster Manager.
   b. Under the **OpenShift disconnected installation tools** section, click **Download** for **OpenShift Client (oc) mirror plugin** and save the file.
2. Extract the archive:

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

3. If necessary, update the plugin file to be executable:

```
$ 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**:

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

Verification

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

```
$ oc mirror help
```

Additional resources

- Installing and using CLI plugins

4.5.2. Configuring credentials that allow images to be mirrored

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

WARNING

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

WARNING

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

Prerequisites

- You configured a mirror registry to use in your 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.

Procedure

Complete the following steps on the installation host:

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:

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

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

The contents of the file resemble the following example:

```
{
  "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 either as ~/**.docker/config.json** or **$XDG_RUNTIME_DIR/containers/auth.json**.

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

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

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

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

```
"auths": {
  "<mirror_registry>": { (1)
    "auth": "<credentials>", (2)
    "email": "you@example.com"
  }
},
```

(1) 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**

(2) Specify the base64-encoded user name and password for the mirror registry.

The file resembles the following example:

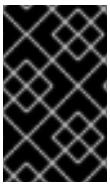
```
{
 "auths": {
  "registry.example.com": {
    "auth": "BGVtbYk3ZHAtqXs=",
    "email": "you@example.com"
  },
  "cloud.openshift.com": {
    "auth": "b3BlbnNo...",
    "email": "you@example.com"
  },
  "quay.io": {
    "auth": "b3BlbnNo...",
    "email": "you@example.com"
  },
  "registry.connect.redhat.com":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

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

4.6. 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)
```

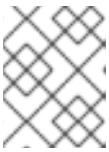
(1) Specifies the location of your storage backend, such as **example.com/mirror/oc-mirror-metadata**.

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-4.17                                 (4)
```

```
    type: ocp
  graph: true                                    5
operators:
- catalog: registry.redhat.io/redhat/redhat-operator-index:v4.17   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.

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

4.7. 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.

4.7.1. 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.

4.7.1.1. 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 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

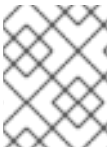
- Run the **oc mirror** command to mirror the images from the specified image set configuration to a specified registry:

```
$ 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

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

Troubleshooting

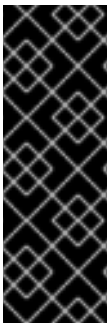
- Unable to retrieve source image .

4.7.2. 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 .

4.7.2.1. 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 .

4.7.2.2. 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 be broken up into multiple .tar files. In this situation, you can pass in a directory that contains the image set .tar files.
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 .

4.8. 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) to retrieve information about the available

Operators in the mirror registry. The release image signatures are used to verify the mirrored release images.

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 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 ./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
```

4.9. 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.

4.9.1. 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
```

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**.

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.

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.

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

4.10. 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.

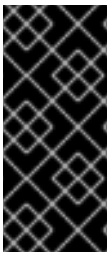
NOTE

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.

4.11. 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-4.17                                        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:v4.17          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.

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

4.12. 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 4.1. 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**
mirror.blockedImages	The full tag, digest, or pattern of images to block from mirroring.	Array of strings. For example: **docker.io/library/alpine**

Parameter	Description	Values
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**.
mirror.helm.repositories.url	The URL of the helm repository to mirror from.	String. For example: **https://example.github.io/podinfo**.

Parameter	Description	Values
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:v4.17`
		`    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:v4.16**.
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:v4.17 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-v4.16**.
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**
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**.

Parameter	Description	Values
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

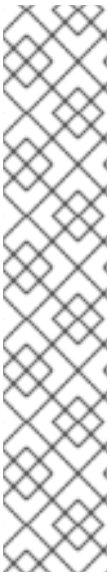
mirror.platform.architectures	The architecture of the platform release payload to mirror.	Array of strings. For example:
		
		``
		architectures:
		  - amd64
		  - arm64
		  - multi
		  - ppc64le
		  - s390x
		``
		
		The default value is **amd64**. The value **multi** ensures that the mirroring is supported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

Parameter	Description	Values

mirror.platform.channels	The platform channel configuration of the image set.	Array of objects. For example:
		`channels:`
		`- name: stable-4.10`
		`- name: stable-4.17`

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-4.16**
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: **4.16.1**
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/**.

Parameter	Description	Values
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.

4.13. IMAGE SET CONFIGURATION EXAMPLES

The following **ImageSetConfiguration** file examples show the configuration for various mirroring use cases.

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
```

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.

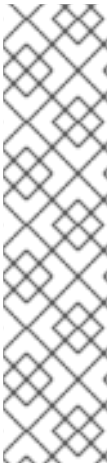
By setting the value of **platform.architectures** to **multi**, you can ensure that the mirroring is supported for multi-architecture releases.

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
```

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:v4.17
      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.

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:v4.17
    packages:
    - name: nutanixcsioperator
      channels:
      - name: stable
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest
```

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:v4.17
    packages:
    - name: elasticsearch-operator
      channels:
      - name: stable-5.7
      - name: stable
```

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:v4.17
      full: true
```

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:v4.17
    targetCatalog: my-namespace/my-operator-catalog
```

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-4.17
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.17
 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
```

4.14. COMMAND REFERENCE FOR OC-MIRROR

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

Table 4.2. 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 4.3. 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. This is only recommended when the image set was created with **--ignore-history**.
--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**.

4.15. ADDITIONAL RESOURCES

- About cluster updates in a disconnected environment

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

You can run your cluster in a restricted network without direct internet connectivity 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.

Just as you can use the **oc-mirror** OpenShift CLI (**oc**) plugin, you can also 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.

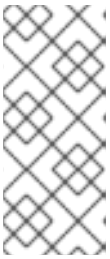
IMPORTANT

oc-mirror plugin v2 is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

5.1. 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 the documentation on 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.

High level workflow

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

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.

5.2. 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 Technology Preview version of oc-mirror, add the **--v2** flag to the oc-mirror plugin v2 command line.

oc-mirror plugin v2 has the following features:

- Verifies that the complete image set specified in the image set config is mirrored to the mirrored registry, regardless of whether the images were previously mirrored or not.
- Uses a cache system instead of metadata.
- 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), **ImageTagMirrorSet** (ITMS), instead of **ImageContentSourcePolicy** (ICSP) for the full image set, rather than just for the incremental changes.
- Saves filter Operator versions by bundle name.
- Does not perform automatic pruning. V2 now has a **Delete** feature, which grants users more control over deleting images.
- Introduces support for **registries.conf**. This change facilitates mirroring to multiple enclaves while using the same cache.

5.2.1. 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 versions 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.

5.3. PREPARING YOUR MIRROR HOSTS

To use the oc-mirror plugin v2 for image mirroring, you need to 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.

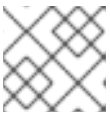
Procedure

1. Download the oc-mirror CLI plugin.
   a. Navigate to the Downloads page of the OpenShift Cluster Manager.
   b. Under the **OpenShift disconnected installation tools** section, click **Download** for **OpenShift Client (oc) mirror plugin** and save the file.
2. Extract the archive:

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

3. If necessary, update the plugin file to be executable:

```
$ 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**:

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

Verification

- Verify that the plugin for oc-mirror 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 that enables you to mirror images from Red Hat to your mirror.

WARNING

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

WARNING

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

Prerequisites

- You configured a mirror registry to use in your 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.

Procedure

Complete the following steps on the installation host:

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:

```
$ cat ./pull-secret | jq . > <path>/<pull_secret_file_in_json> 1
```

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

The contents of the file resemble the following example:

```
{
  "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 **$XDG_RUNTIME_DIR/containers/auth.json**.

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

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

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

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

```
"auths": {
  "<mirror_registry>": { (1)
    "auth": "<credentials>", (2)
    "email": "you@example.com"
  }
},
```

(1) 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**

(2) Specify the base64-encoded user name and password for the mirror registry.

The file resembles the following example:

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

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

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. Building the image set configuration

The oc-mirror plugin v2 uses the image set configuration as the input file to determine the required images for mirroring.

Example for the ImageSetConfiguration input file

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    channels:
    - name: stable-4.13
      minVersion: 4.13.10
      maxVersion: 4.13.10
    graph: true
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15
      packages:
       - name: aws-load-balancer-operator
       - name: 3scale-operator
       - name: node-observability-operator
  additionalImages:
   - name: registry.redhat.io/ubi8/ubi:latest
   - name:
registry.redhat.io/ubi9/ubi@sha256:20f695d2a91352d4eaa25107535126727b5945bff38ed36a3e59590f495046f0
```

5.4.2. 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 isc.yaml --workspace file://<file_path> docker://<mirror_registry_url> --v2 1
```

1 Specify 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 **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.

5.4.3. 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.

1. **Mirror to disk**: Prepare an archive containing the image set for mirroring. Internet access is required.
2. **Manual step**: Transfer the archive to the network of the disconnected mirror registry.
3. **Disk to mirror**: To mirror the image set from the archive to the target disconnected registry, run oc-mirror plugin v2 from the environment that has access to the mirror registry.

5.4.3.1. Mirroring from mirror to disk

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

oc-mirror plugin v2 retrieves the container images from the source specified in the image set configuration 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 isc.yaml file://<file_path> --v2 1
```

1 Add the required file path.

Verification

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

Next steps

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

5.4.3.2. 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

- 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 isc.yaml --from file://<file_path> docker://<mirror_registry_url> --v2 1
```

1 Specify 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 **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. ADDITIONAL RESOURCES

- Updating a cluster in a disconnected environment using the OpenShift Update Service .

5.6. ABOUT CUSTOM RESOURCES GENERATED BY V2

With oc-mirror plugin v2, **ImageDigestMirrorSet** (IDMS) and **ImageTagMirrorSet** (ITMS) are generated by default if at least one image is found to which a tag refers. These sets contain mirrors for images referenced by digest or tag in releases, Operator catalogs and additional images.

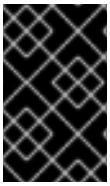
The **ImageDigestMirrorSet** (IDMS) links the mirror registry to the source registry and forwards image pull requests using digest specifications. The **ImagetagMirrorSet** (ITMS) resource, however, redirects image pull requests by using image tags.

Operator Lifecycle Manager (OLM) uses the **CatalogSource** resource to retrieve information about the available Operators in the mirror registry.

The OSUS service uses the **UpdateService** resource to provide Cincinnati graph to the disconnected environment.

5.6.1. 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** to the cluster.



IMPORTANT

In oc-mirror plugin v2, the IDMS and ITMS files cover the entire image set, unlike the 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

- 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
```

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
```

5.7. DELETION OF IMAGES FROM YOUR DISCONNECTED ENVIRONMENT

Before you can use oc-mirror plugin v2, you must delete previously deployed images. oc-mirror plugin v2 no longer performs automatic pruning.

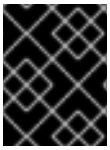
You must create the **DeleteImageSetConfiguration** file to delete image configuration when using oc-mirror plugin v2. This prevents accidentally deleting necessary or deployed images when making changes with **ImageSetConfig.yaml**.

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

- All images of OpenShift Container Platform release 4.13.3.
- The catalog image **redhat-operator-index v4.12**.
- The **aws-load-balancer-operator** v0.0.1 bundle and all its related images.
- The additional images **ubi** and **ubi-minimal** referenced by their corresponding digests.

Example: DeleteImageSetConfig

```
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 mirroring issues.

In the image delete workflow, 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, thereby reducing the storage previously occupied by the deleted blobs. Enabling the garbage collector differs depending on your container registry.

5.7.1. Deleting the images from disconnected environment

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

Procedure

1. Create a YAML file that deletes previous images:

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

   Where:

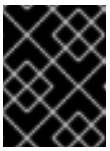
   - **<previously_mirrored_work_folder>**: Use the directory where images were previously mirrored or stored during the mirroring process.
   - **<remote_registry>**: Insert the URL or address of the remote container registry from which images will be deleted.

2. Go to the **<previously_mirrored_work_folder>/delete directory** that was created.
3. Verify that the **delete-images.yaml** file has been generated.
4. Manually ensure that each image listed in the file is no longer needed by the cluster and can be safely removed from the registry.
5. After you generate the **delete** YAML file, delete the images from the remote registry:

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

   Where:

   - **<previously_mirrored_work_folder>**: Specify your previously mirrored work folder.



IMPORTANT

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

5.8. 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.

5.8.1. Performing 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> --from file://<oc_mirror_workspace_path>
docker://<mirror_registry_url> --dry-run --v2
```

Where:

- **<image_set_config_yaml>**: Use the image set configuration file that you just 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 deleted.

Example output

```
$ oc mirror --config /tmp/isc_dryrun.yaml file://<oc_mirror_workspace_path> --dry-run --
v2

[INFO]   : :warning:  --v2 flag identified, flow redirected to the oc-mirror v2 version. This is
Tech Preview, it is still under development and it is not production ready.
[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

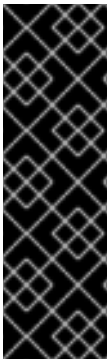
1. Navigate to the workspace directory that was generated:

```
$ 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.

5.9. 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.



IMPORTANT

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

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

The 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.

5.9.1. 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".

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.

5.10. 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.

Bundle versions typically use semantic versioning standards (SemVer), and you can sort bundles within a channel by version. You can select buncles that fall within a specific range in the **ImageSetConfig**.

This selection algorithm ensures consistent outcomes compared to oc-mirror plugin v1. However, it does not include upgrade graph details, such as **replaces**, **skip**, and **skipRange**. This approach differs from the OLM algorithm. It might mirror more bundles than necessary for upgrading a cluster because of potentially shorter upgrade paths between the **minVersion** and **maxVersion**.

Table 5.1. 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, 1 bundle, corresponding to the head version of the default channel for 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

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 of the default channel for that package

ImageSetConfig operator filtering	Expected bundle versions
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 the default channel, from the **minVersion**, up to the channel head for that package that do not rely on the shortest path from upgrade the graph.
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 the default channel that are lower than the **maxVersion** for that package.
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 the default channel, 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.

ImageSetConfig operator filtering	Expected bundle versions
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.
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 specified packages and channels.
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.

ImageSetConfig operator filtering	Expected bundle versions
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. This scenario does not relyon the shortest path from the upgrade graph.
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 the **maxVersion** (not relying on the shortest path from the upgrade graph). The head of the channel is not included, even if multiple channels are included in the filtering.
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**, not relying on the shortest path from the upgrade graph. The head of the channel is not included, even if multiple channels are included in the filtering.

ImageSetConfig operator filtering	Expected bundle versions
Scenario 14 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14 packages: - name: aws-load-balancer-operator bundles: - name: aws-load-balancer-operator.v1.1.0 - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas	Only the bundles specified for each package are included in the filtering.
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. filtering by channel and by package with a **minVersion** or **maxVersion** is not allowed.
Scenario 16 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 17 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

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 5.2. **ImageSetConfiguration** parameters

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

Parameter	Description	Values
archiveSize	The maximum size, in GiB, of each archive file within the image set.	Integer Example: **4**
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**
mirror.blockedImages	The full tag, digest, or pattern of images to block from mirroring.	Array of strings Example: **docker.io/library/alpine**
mirror.operators	The Operators configuration of the image set.	Array of objects Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.17 packages: - name: elasticsearch-operator minVersion: '2.4.0'
mirror.operators.catalog	The Operator catalog to include in the image set.	String Example: **registry.redhat.io/redhat/redhat-operator-index:v4.15**

Parameter	Description	Values
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:4.17 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**
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**

Parameter	Description	Values
mirror.operators.packages.bundles	Selected bundles configuration	Array of objects Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.17 packages: - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas
mirror.operators.packages.bundles.name	Name of the bundle selected for mirror (as it appears in the catalog).	String Example : **3scale-operator.v0.10.0-mas**
mirror.operators.targetCatalog	An alternative name and optional namespace hierarchy to mirror the referenced catalog as	String Example: **my-namespace/my-operator-catalog**

Parameter	Description	Values

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/v2alpha1`
		`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-4.17`

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**
mirror.platform.channels.shortestPath	Toggles shortest path mirroring or full range mirroring.	Boolean The default value is **false**

Parameter	Description	Values
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**

5.11.1. Delete ImageSet Configuration parameters

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

Table 5.3. **DeleteImageSetConfiguration** parameters

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

Parameter	Description	Values
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**
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'

Parameter	Description	Values
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**

Parameter	Description	Values
delete.operators.packages.bundles	The selected bundles configuration	Array of objects You cannot choose both channels and bundles for the same operator. Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:{product-version} packages: - name: 3scale-operator bundles: - name: 3scale-operator.v0.10.0-mas
delete.operators.packages.bundles.name	Name of the bundle selected to delete (as it is displayed in the catalog)	String Example : **3scale-operator.v0.10.0-mas**
delete.platform	The platform configuration of the image set	Object
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**

Parameter	Description	Values

delete.platform.channels	The platform channel configuration of the image set.	Array of objects
		
		Example:
		
		`channels:`
		`- name: stable-4.12`
		`- name: stable-4.17`

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

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 5.4. Subcommands and flags for the oc-mirror plugin v2

Subcommand	Description
help	Show help about any subcommand
version	Output the oc-mirror version

Subcommand	Description
delete	Deletes images in remote registry and local cache.

Table 5.5. 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.
--dest-tls-verify	Requires HTTPS and verifies certificates when accessing the container registry or daemon.
--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
--loglevel	Displays string log levels. Supported values include info, debug, trace, error. The default is **info**.
-p, **--port**	Determines the HTTP port used by oc-mirror plugin v2 local storage instance. The default is **55000**.
--max-nested-paths <int>	Specifies the maximum number of nested paths for destination registries that limit nested paths. The default is **0**.
--secure-policy	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	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.

Flag	Description
--workspace	Determines string oc-mirror plugin v2 workspace where resources and internal artifacts are generated.

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