Red Hat OpenShift

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 OKD 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.

The oc-mirror v1 plugin is now deprecated. The default mirroring method is changing to the oc-mirror v2 plugin in a future release, requiring you to specify the `--v1` flag to continue using the oc-mirror v1 plugin. Transition to the oc-mirror v2 plugin for continued support and improvements.

About the oc-mirror plugin

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

- Provides a centralized method to mirror OKD releases, Operators, helm charts, and other images.
- Maintains update paths for OKD and Operators.
- Uses a declarative image set configuration file to include only the OKD 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 OKD 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 OKD and Operators and performs dependency resolution as required.

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:

- Create an image set configuration file.
- 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.
- Configure your cluster to use the resources generated by the oc-mirror plugin.

- Repeat these steps to update your target mirror registry as necessary.

> 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.

oc-mirror plugin compatibility and support

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

> On `aarch64`, `ppc64le`, and `s390x` architectures the oc-mirror plugin is only supported for OKD versions 4.14 and later.

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

Additional resources

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

About the mirror registry

You can mirror the images that are required for OKD 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 OKD 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.

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 OKD clusters.

When you populate your mirror registry with OKD 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.

Red Hat does not test third party registries with OKD.

Additional resources

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

Prerequisites

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

  > 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 OKD 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 OKD in disconnected installations.

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.

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 Fedora version that you are using.

Procedure

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

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

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

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

> Do not rename the `oc-mirror` file.

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

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

Verification

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

```
$ oc mirror help
```

Additional resources

- Installing and using CLI plugins

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.

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

Prerequisites

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

Procedure

Complete the following steps on the installation host:

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

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

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

Example output

```
BGVtbYk3ZHAtqXs=
```

- Create a `.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.

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 OKD 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.

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

- 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`.

- 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-4.19
 (4)
      type: ocp
    graph: true
 (5)
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-
index:v4.19  (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 OKD 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 OKD 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.

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

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

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

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

- 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

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.

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.

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.

> 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

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

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.

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.

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.

> 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.

> 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

- Navigate to your output directory:

```
$ cd <path_to_output_directory>
```

- 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.

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

- Navigate into the `oc-mirror-workspace/` directory that was generated.
- Navigate into the results directory, for example, `results-1639608409/`.
- 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.

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

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

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

> OLM v1 uses the `ClusterCatalog` resource to retrieve information about the available cluster extensions in the mirror registry.
>
> The oc-mirror plugin v1 does not generate `ClusterCatalog` resources automatically; you must manually create them. For more information on creating and applying `ClusterCatalog` resources, see "Adding a catalog to a cluster" in "Extensions".

Prerequisites

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

Procedure

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

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

- 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/
```

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

- Verify that the `ImageContentSourcePolicy` resources were successfully installed by running the following command:

```
$ oc get imagecontentsourcepolicy
```

- Verify that the `CatalogSource` resources were successfully installed by running the following command:

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

Additional resources

- Adding a catalog to a cluster in "Extensions"

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.

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 OKD 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 OKD 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

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

- 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
```

- Navigate into the workspace directory that was generated:

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

- Review the `mapping.txt` file that was generated.

  This file contains a list of all images that would be mirrored.

- 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.

> 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.

Including local OCI Operator catalogs

While mirroring OKD 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.

> When mirroring local OCI catalogs, any OKD 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 OKD release to your mirror registry.

> If you used the Technology Preview OCI local catalogs feature for the oc-mirror plugin for OKD 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

- 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 OKD 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-4.19
 (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:v4.19           (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 OKD 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.

- 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

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 1. `ImageSetConfiguration` parameters

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

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

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

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

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

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

Parameter	Description	Values

`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`.
`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`.

Parameter	Description	Values
`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:v4.19`
		`    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:v4.19` .
`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:v4.19 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-v4.19`.

Parameter	Description	Values
`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`.
`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`

Parameter	Description	Values
`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-4.19`
`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-4.19`
`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: `4.19.1`
`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`.

Parameter	Description	Values
`mirror.platform.graph`	Indicates whether the OSUS graph is added to the image set and subsequently published to the mirror.	Boolean. The default value is `false` .
`storageConfig`	The back-end configuration of the image set.	Object
`storageConfig.local`	The local back-end configuration of the image set.	Object
`storageConfig.local.path`	The path of the directory to contain the image set metadata.	String. For example: `./path/to/dir/` .
`storageConfig.registry`	The registry back-end configuration of the image set.	Object
`storageConfig.registry.imageURL`	The back-end registry URI. Can optionally include a namespace reference in the URI.	String. For example: `quay.io/myuser/imageset:metadata` .
`storageConfig.registry.skipTLS`	Optionally skip TLS verification of the referenced back-end registry.	Boolean. The default value is `false` .

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.

Image set configuration examples

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

Use case: Including the shortest OKD update path

The following `ImageSetConfiguration` file uses a local storage backend and includes all OKD 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 OKD from a minimum to the latest version for multi-architecture releases

The following `ImageSetConfiguration` file uses a registry storage backend and includes all OKD 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 OKD 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

```yaml
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.

> 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:v4.19
      packages:
        - name: rhacs-operator
          channels:
          - name: stable
            minVersion: 4.0.1
```

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:v4.19
    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.

> 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:v4.19
    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:v4.19
      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:v4.19
    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-4.19
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19
 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
```

Use case: Including the upgrade path for EUS releases

The following `ImageSetConfiguration` file includes the `eus-<version>` channel, where the `maxVersion` value is at least two minor versions higher than the `minVersion` value.

For example, in this `ImageSetConfiguration` file, the `minVersion` is set to `4.12.28`, while the `maxVersion` for the `eus-4.14` channel is `4.14.16`.

Example `ImageSetConfiguration` file

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    graph: true # Required for the OSUS Operator
    architectures:
    - amd64
    channels:
    - name: stable-4.12
      minVersion: '4.12.28'
      maxVersion: '4.12.28'
      shortestPath: true
      type: ocp
    - name: eus-4.14
      minVersion: '4.12.28'
      maxVersion: '4.14.16'
      shortestPath: true
      type: ocp
```

Use case: Including the multi-arch OKD images and catalog for multicluster engine Operator

The following `ImageSetConfiguration` file includes multicluster engine for Kubernetes Operator and all OKD versions starting at a minimum version of `4.19.0` in the channel.

Example `ImageSetConfiguration` file

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL:
agent.agent.example.com:5000/openshift/release/metadata:latest/openshi
ft/release/metadata:latest
mirror:
  platform:
    architectures:
      - "multi"
    channels:
    - name: stable-4.19
      minVersion: 4.19.0
      maxVersion: 4.19.1
      type: ocp
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19
      packages:
        - name: multicluster-engine
```

Command reference for oc-mirror

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

Table 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.
`init`	Output an initial image set configuration template.
`list`	List available platform and Operator content and their version.

Subcommand	Description
`version`	Output the oc-mirror version.

Table 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`.

Flag	Description
`--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`).
`--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 recommended only when the image set was created with the `--ignore-history` flag.
`--skip-missing`	If an image is not found, skip it instead of reporting an error and aborting execution. Does not apply to custom images explicitly specified in the image set configuration.
`--skip-pruning`	Disable automatic pruning of images from the target mirror registry.
`--skip-verification`	Skip digest verification.
`--source-skip-tls`	Disable TLS validation for the source registry.
`--source-use-http`	Use plain HTTP for the source registry.
`-v`, `--verbose <int>`	Specify the number for the log level verbosity. Valid values are `0` - `9`. The default is `0`.

Additional resources

- About cluster updates in a disconnected environment