OpenShift Container Platform 4.19

断开连接的环境

在断开连接的环境中管理 OpenShift Container Platform 集群

Last Updated: 2025-12-11

OpenShift Container Platform 4.19 断开连接的环境

在断开连接的环境中管理 OpenShift Container Platform 集群

Legal Notice

Abstract

本文档提供有关在没有互联网访问的环境中安装、管理和配置 OpenShift Container Platform 集群的信息。

Table of Contents

第 1 章 关于断开连接的环境

断开连接的环境是一个无法完全访问互联网的环境。

OpenShift Container Platform 旨在执行取决于互联网连接的许多自动功能，如从 registry 检索发行镜像或检索集群的更新路径和建议。如果没有直接互联网连接，集群必须执行额外的设置和配置，才能在断开连接的环境中维护完整的功能。

1.1. 断开连接的环境术语表

虽然它在 OpenShift Container Platform 文档中使用，但 *断开连接的环境（disconnected environment）* 是一个广泛的术语，但可能会代表具有不同互联网连接级别的环境。对于一个特定级别的互联网连接，有时可能会使用其他术语来代表，这些环境可能需要额外的、特殊的配置。

下表描述了用于引用没有完全互联网连接的环境的不同术语：

表 1.1. 断开连接的环境术语

术语	描述
Air-gapped 网络	完全与外部网络隔离的环境或网络。 这种隔离取决于内部网络上的机器与外部网络的任何其他部分之间的物理分离或"空缺"。air-gapped 环境通常用于具有严格安全或监管要求的行业。
断开连接的环境	在一定级别上与外部网络隔离的环境或网络。 这种隔离可以通过在内部网络和外部网络上的计算机之间进行物理或逻辑分离来实现。无论与外部网络的隔离级别，在断开连接的环境中的集群都无法访问由红帽托管的公共服务，需要额外的设置来维护完整的集群功能。
受限网络	与外部网络有限连接的环境或网络。 物理网络和外部网络上的机器之间可能存在物理连接，但网络流量会被其他配置的限制，如防火墙和代理。

1.2. 使用断开连接的环境的首选方法

您可以有多个选项来管理断开连接的环境中的集群。例如，在 mirror 镜像时，可以选择使用 oc-mirror OpenShift CLI (**oc**) 插件，或使用 **ocadm** 命令。

但是，有些选项对断开连接的环境的用户提供了更简单、方便的用户体验，因此可以作为首选。

除非您的机构需要您选择另一个选项，否则使用以下方法 mirror 镜像、安装集群和更新集群：

- 使用 oc-mirror 插件 v2 镜像。
- 使用基于代理的安装程序安装集群。
- 使用本地 OpenShift Update Service 实例更新集群。

第 2 章 将连接的集群转换为断开连接的集群

在某些情况下，您可能需要将 OpenShift Container Platform 集群从连接的集群转换为断开连接的集群。

断开连接的集群（也称为受限集群）没有与互联网的活跃连接。因此，您必须镜像 registry 和安装介质的内容。您可以在可以访问互联网和您的封闭网络的主机上创建此镜像 registry，或者将镜像复制到可跨网络界限的设备中。

本主题描述了将现有连接的集群转换为断开连接的集群的一般过程。

2.1. 关于镜像 REGISTRY

您必须可以访问互联网来获取所需的容器镜像。使用一个替代的 registry 意味着，将 mirror registry 放在一个可访问您的网络以及互联网的 mirror 主机上。

您可以镜像 OpenShift Container Platform 安装所需的镜像，以及容器镜像 registry 的后续产品更新，如 Red Hat Quay、JFrog Artifactory、Sonatype Nexus Repository 或 Harbor。如果您无法访问大型容器 registry，可以使用 *mirror registry for Red Hat OpenShift*，它是包括在 OpenShift Container Platform 订阅中的一个小型容器 registry。

您可以使用支持 Docker v2-2 的任何容器 registry，如 Red Hat Quay，*mirror registry for Red Hat OpenShift*, Artifactory, Sonatype Nexus Repository, 或 Harbor。无论您所选 registry 是什么，都会将互联网上红帽托管站点的内容镜像到隔离的镜像 registry 相同。镜像内容后，您要将每个集群配置为从镜像 registry 中检索此内容。

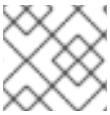
重要

OpenShift 镜像 registry 不能用作目标 registry，因为它不支持没有标签的推送，在镜像过程中需要这个推送。

如果选择的容器 registry 不是 *mirror registry for Red Hat OpenShift*，则需要集群中置备的每台机器都可以访问它。如果 registry 无法访问，安装、更新或常规操作（如工作负载重新定位）可能会失败。因此，您必须以高度可用的方式运行镜像 registry，镜像 registry 至少必须与 OpenShift Container Platform 集群的生产环境可用性相匹配。

使用 OpenShift Container Platform 镜像填充镜像 registry 时，可以遵循以下两种情况。如果您的主机可以同时访问互联网和您的镜像 registry，而不能访问您的集群节点，您可以直接从该机器中镜像该内容。这个过程被称为 *连接的镜像(mirror)*。如果没有这样的主机，则必须将该镜像文件镜像到文件系统中，然后将该主机或者可移动介质放入受限环境中。这个过程被称为 *断开连接的镜像*。

对于已镜像的 registry，若要查看拉取镜像的来源，您必须查看 **Trying 以访问** CRI-O 日志中的日志条目。查看镜像拉取源的其他方法（如在节点上使用 **crictl images** 命令）显示非镜像镜像名称，即使镜像是从镜像位置拉取的。



注意

红帽没有针对 OpenShift Container Platform 测试第三方 registry。

2.2. 先决条件

- 已安装 **oc** 客户端。
- 一个正在运行的集群。

- 安装的镜像 registry，这是支持托管 OpenShift Container Platform 集群的位置的 Docker v2-2 的容器镜像 registry，如以下 registry 之一：
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus 仓库
  - Harbor

  如果您有 Red Hat Quay 订阅，请参阅有关部署 Red Hat Quay 用于概念验证 的文档，或使用 Quay Operator。
- 必须将镜像存储库配置为共享镜像。例如，Red Hat Quay 仓库需要 机构 才能共享镜像。
- 访问互联网来获取所需的容器镜像。

2.3. 为镜像准备集群

在断开集群的连接前，您必须镜像(mirror)或复制(mirror)或复制(mirror)或复制镜像(mirror) registry。要镜像镜像，您必须通过以下方法准备集群：

- 将镜像 registry 证书添加到主机上的可信 CA 列表中。
- 创建包含您的镜像 pull secret 的 **.dockerconfigjson** 文件，该文件来自 **cloud.openshift.com** 令牌。

流程

1. 配置允许镜像镜像的凭证：
   a. 将镜像 registry 的 CA 证书（采用简单 PEM 或 DER 文件格式）添加到可信 CA 列表中。例如：

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

   其中, </path/to/cert.crt>
   指定本地文件系统中的证书路径。

   b. 更新 CA 信任。例如，在 Linux 中：

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

   c. 从全局 pull secret 中提取 **.dockerconfigjson** 文件：

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

   输出示例

   ```
   .dockerconfigjson
   ```

   d. 编辑 **.dockerconfigjson** 文件以添加您的镜像 registry 和身份验证凭证，并将它保存为新文件：

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

其中：

<local_registry>

指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。

auth

为您的镜像 registry 指定 base64 编码的用户名和密码。

<registry>:<port>/<namespace>

指定镜像 registry 详情。

<token>

为您的镜像 registry 指定 base64 编码的 **username:password**。

例如：

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

2.4. 对容器镜像进行镜像处理(MIRROR)

正确配置集群后，您可以将外部存储库中的镜像镜像到镜像存储库。

流程

1. 镜像 Operator Lifecycle Manager(OLM)镜像：

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

其中：

product-version

指定要与 OpenShift Container Platform 版本对应的标签，如 **4.8**。

mirror_registry

指定用于镜像 Operator 内容的目标 registry 和命名空间的完全限定域名(FQDN)，其中 **<namespace>** 是 registry 上的任何现有命名空间。

reg_creds

指定您修改的 **.dockerconfigjson** 文件的位置。

例如：

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

2. 为任何其他红帽提供的 Operator 镜像内容：

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

其中：

index_image

指定您要镜像的目录的索引镜像。

mirror_registry

指定要将 Operator 内容镜像到的目标 registry 和命名空间的 FQDN，其中 **<namespace>** 是 registry 上的任何现有命名空间。

reg_creds

可选：如果需要，指定 registry 凭证文件的位置。

例如：

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

3. 镜像 OpenShift Container Platform 镜像存储库：

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

其中：

product-version

指定与要安装的 OpenShift Container Platform 版本对应的标签，如 **4.8.15-x86_64**。

架构

为您的服务器指定构架类型，如 **x86_64**。

local_registry

指定镜像存储库的 registry 域名。

local_repository

指定要在 registry 中创建的存储库名称，如 **ocp4/openshift4**。

例如：

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

输出示例

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

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

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

4. 根据需要镜像任何其他 registry：

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

其他资源

- 如需有关镜像 Operator 目录的更多信息，请参阅 镜像 Operator 目录。
- 如需有关 **oc adm catalog mirror** 命令的更多信息，请参阅 OpenShift CLI 管理员命令参考。

2.5. 为镜像 REGISTRY 配置集群

在将镜像创建并镜像(mirror)到镜像 registry 后，您必须修改集群，以便 pod 可以从镜像 registry 中拉取镜像。

您必须：

- 将镜像 registry 凭证添加到全局 pull secret 中。
- 在集群中添加镜像 registry 服务器证书。
- 创建 **ImageContentSourcePolicy** 自定义资源(ICSP)，它将镜像 registry 与源 registry 关联。

1. 将镜像 registry 凭证添加到集群全局 pull-secret 中：

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

1 提供新 pull secret 文件的路径。

例如：

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

2. 将 CA 签名的镜像 registry 服务器证书添加到集群中的节点：

   a. 创建包含镜像 registry 的服务器证书的配置映射

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

   例如：

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

   b. 使用配置映射更新 **image.config.openshift.io/cluster** 自定义资源(CR)。OpenShift Container Platform 将对此 CR 的更改应用到集群中的所有节点：

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

   例如：

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

3. 创建一个 ICSP，将在线 registry 中的容器拉取请求重定向到镜像 registry：

   a. 创建 **ImageContentSourcePolicy** 自定义资源：

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

1. 指定镜像 registry 和存储库的名称。
2. 指定包含所镜像内容的在线 registry 和存储库。

b. 创建 ICSP 对象：

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

输出示例

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

OpenShift Container Platform 会将对此 CR 的更改应用到集群中的所有节点。

4. 验证已添加了镜像 registry 的凭证、CA 和 ICSP：

a. 登录到节点：

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

b. 将 **/host** 设置为 debug shell 中的根目录：

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

c. 检查凭证的 **config.json** 文件：

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

输出示例

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

1. 确保存在镜像 registry 和凭证。

d. 进入 **certs.d** 目录

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

e. 列出 **certs.d** 目录中的证书：

```
sh-4.4# ls
```

输出示例

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

1 确保镜像 registry 位于列表中。

f. 检查 ICSP 是否将镜像 registry 添加到 **registry.conf** 文件中：

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

输出示例

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

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

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

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

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

registry.mirror 参数表示在原始 registry 前搜索镜像 registry。

g. 退出节点。

```
sh-4.4# exit
```

2.6. 确保应用程序继续工作

在断开集群与网络的连接前，请确保集群按预期运行，并且所有应用程序都按预期工作。

流程

使用以下命令检查集群的状态：

- 确定您的 pod 正在运行：

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

输出示例

```
NAMESPACE                                          NAME                                                READY
STATUS    RESTARTS   AGE
kube-system                                        apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-0
1/1     Running    0          39m
kube-system                                        apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-1
1/1     Running    0          39m
kube-system                                        apiserver-watcher-ci-ln-47ltxtb-f76d1-mrffg-master-2
```

```
1/1     Running     0          39m
openshift-apiserver-operator                openshift-apiserver-operator-79c7c646fd-5rvr5
1/1     Running     3          45m
openshift-apiserver                         apiserver-b944c4645-q694g                   2/2
Running     0          29m
openshift-apiserver                         apiserver-b944c4645-shdxb                   2/2
Running     0          31m
openshift-apiserver                         apiserver-b944c4645-x7rf2                   2/2
Running     0          33m
 ...
```

- 确定您的节点处于 READY 状态：

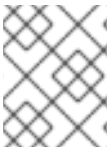
```
$ oc get nodes
```

输出示例

```
NAME                                STATUS   ROLES    AGE   VERSION
ci-ln-47ltxtb-f76d1-mrffg-master-0       Ready    master   42m   v1.32.3
ci-ln-47ltxtb-f76d1-mrffg-master-1       Ready    master   42m   v1.32.3
ci-ln-47ltxtb-f76d1-mrffg-master-2       Ready    master   42m   v1.32.3
ci-ln-47ltxtb-f76d1-mrffg-worker-a-gsxbz   Ready    worker   35m   v1.32.3
ci-ln-47ltxtb-f76d1-mrffg-worker-b-5qqdx   Ready    worker   35m   v1.32.3
ci-ln-47ltxtb-f76d1-mrffg-worker-c-rjkpq   Ready    worker   34m   v1.32.3
```

2.7. 断开集群与网络的连接

镜像所有所需的软件仓库并配置集群以作为断开连接的集群后，您可以断开集群与网络的连接。



注意

当集群丢失了互联网连接时，Insights Operator 会降级。您可以通过临时禁用 Insights Operator 来避免这个问题，直到您可以恢复它。

2.8. 恢复降级的 INSIGHTS OPERATOR

从网络断开集群的连接会导致集群丢失互联网连接。Insights Operator 会降级，因为它需要访问 Red Hat Insights。

本节描述了如何从降级的 Insights Operator 恢复。

流程

1. 编辑 **.dockerconfigjson** 文件以删除 **cloud.openshift.com** 条目，例如：

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

2. 保存该文件。

3. 使用编辑的 **.dockerconfigjson** 文件更新集群 secret：

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

4. 验证 Insights Operator 不再降级：

```
$ oc get co insights
```

输出示例

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

2.9. 恢复网络

如果要重新连接断开连接的集群并从在线 registry 中拉取镜像，请删除集群的 ImageContentSourcePolicy(ICSP)对象。如果没有 ICSP，对外部 registry 的拉取请求不再重定向到镜像 registry。

流程

1. 查看集群中的 ICSP 对象：

```
$ oc get imagecontentsourcepolicy
```

输出示例

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

2. 删除断开集群时创建的所有 ICSP 对象：

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

例如：

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

输出示例

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

3. 等待所有节点重启并返回到 READY 状态，并验证 **registry.conf** 文件是否指向原始 registry，而不是镜像 registry：

   a. 登录到节点：

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

   b. 将 **/host** 设置为 debug shell 中的根目录：

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

c. 检查 **registry.conf** 文件：

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

输出示例

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

1 您删除的、由 ICSP 创建的 **registry** 和 **registry.mirror** 条目已被删除。

第 3 章 关于断开连接的安装镜像

作为一个集群管理员，您可以使用镜像 registry 确保集群只使用满足机构对外部内容控制的容器镜像。

3.1. 用于在断开连接的环境中安装集群的镜像 REGISTRY

您必须将所需的容器镜像镜像(mirror)到断开连接的环境中，然后才能安装并置备集群。要 mirror 这些容器镜像，您必须有一个 mirror registry。考虑以下用于创建和使用 mirror registry 的选项：

- 如果您已有容器镜像 registry，如 Red Hat Quay，您可以使用它作为您的 mirror registry。如果您还没有 registry，您必须创建一个。
- 建立 registry 后，您需要镜像工具。要在断开连接的环境中将 OpenShift Container Platform 镜像存储库 mirror 到 mirror registry，您可以使用 oc-mirror OpenShift CLI (**oc**) 插件。oc-mirror 插件是一个单一工具，它会将所有所需的 OpenShift Container Platform 内容和其他镜像 mirror 到您的 mirror registry。oc-mirror 插件是镜像的首选方法。
- 另外，您可以使用 **oc adm** 命令 mirror OpenShift Container Platform 的发行版本和目录镜像。

第 4 章 使用 MIRROR REGISTRY FOR RED HAT OPENSHIFT 创建 MIRROR REGISTRY

mirror registry for Red Hat OpenShift 是一个小型灵活的容器 registry，作为目标，用于为断开连接的安装镜像(mirror)的 OpenShift Container Platform 所需的容器镜像。

如果您已有容器镜像 registry，如 Red Hat Quay，您可以跳过本节并直接 mirror OpenShift Container Platform 镜像仓库。

重要

mirror registry for Red Hat OpenShift 的目的并不是替代 Red Hat Quay 的生产环境部署。

4.1. 先决条件

- OpenShift Container Platform 订阅。
- 安装了 Podman 3.4.2 或更高版本以及 OpenSSL 的 Red Hat Enterprise Linux (RHEL) 8 和 9。
- Red Hat Quay 服务的完全限定域名，它必须通过 DNS 服务器解析。
- 目标主机上的基于密钥的 SSH 连接。为本地安装自动生成 SSH 密钥。对于远程主机，您必须生成自己的 SSH 密钥。
- 2 个或更多 vCPU。
- 8 GB RAM。
- OpenShift Container Platform 4.19 发行镜像大约需要 12 GB；OpenShift Container Platform 4.19 发行镜像和 OpenShift Container Platform 4.19 Red Hat Operator 镜像大约需要 358 GB。

重要

- 每个流推荐具有 1 TB 或更多空间。
- 这些要求基于本地测试结果，且只测试了发行镜像和 Operator 镜像。存储要求可能会因您的组织的需求而有所不同。例如，当镜像了多个 z-streams 时，则可能需要更多空间。您可以使用标准 Red Hat Quay 功能 或适当的 API 调用来 删除不必要的镜像并释放空间。

4.2. MIRROR REGISTRY FOR RED HAT OPENSHIFT 介绍

对于断开连接的 OpenShift Container Platform 部署，需要一个容器 registry 来安装集群。要在这样的集群中运行 production-grade registry 服务，您必须创建一个单独的 registry 部署来安装第一个集群。*mirror registry for Red Hat OpenShift* 可以解决这个问题，并包含在每个 OpenShift Container Platform 订阅中。它可用于从 OpenShift 控制台 **Downloads** 页面下载。

mirror registry for Red Hat OpenShift 允许用户使用 **mirror-registry** 命令行界面(CLI)工具安装一个较小的 Red Hat Quay 版本及其所需的组件。*mirror registry for Red Hat OpenShift* 会自动部署，带有预先配置的本地存储和本地数据库。它还包括自动生成的用户凭证和访问权限，其中只有一个输入集，且不需要额外配置选项。

mirror registry for Red Hat OpenShift 提供了一个预先确定的网络配置，并在成功时报告部署的组件凭证并访问 URL。另外还提供了一组有限的可选配置输入，如完全限定域名(FQDN)服务、超级用户名称和密

码，以及自定义 TLS 证书。这为用户提供了一个容器 registry，以便在受限网络环境中运行 OpenShift Container Platform 时，轻松创建所有 OpenShift Container Platform 发行版本内容的离线镜像。

如果在安装环境中已有另一个容器 registry，则使用 *mirror registry for Red Hat OpenShift* 是可选的。

4.2.1. Mirror registry for Red Hat OpenShift 限制

以下限制适用于 *mirror registry for Red Hat OpenShift* ：

- *mirror registry for Red Hat OpenShift* 并不是一个高度可用的 registry，且只支持本地文件系统存储。它并不适用于 OpenShift Container Platform 替换 Red Hat Quay 或内部镜像 registry。
- *mirror registry for Red Hat OpenShift* 的目的并不是替代 Red Hat Quay 的生产环境部署。
- *mirror registry for Red Hat OpenShift* 只支持托管安装断开连接的 OpenShift Container Platform 集群（如发行镜像或 Red Hat Operator 镜像）所需的镜像。它使用 Red Hat Enterprise Linux(RHEL)机器上的本地存储，而 RHEL 支持的存储也被 *mirror registry for Red Hat OpenShift* 支持。

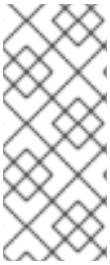
注意

因为 *mirror registry for Red Hat OpenShift* 使用本地存储，所以您应该了解 mirror 镜像时消耗的存储使用量，并使用 Red Hat Quay 的垃圾回收功能来缓解潜在的问题。有关此功能的更多信息，请参阅"Red Hat Quay 垃圾回收"。

- 对推送到 *mirror registry for Red Hat OpenShift* bootstrap 的红帽产品镜像的支持会被每个相应产品的有效订阅涵盖。进一步启用 bootstrap 体验的例外列表可在自助管理的 Red Hat OpenShift 大小和订阅指南中找到。
- 由客户创建的内容不应由 *mirror registry for Red Hat OpenShift* 托管。
- 不建议将 *mirror registry for Red Hat OpenShift* 与多个集群一起使用，因为多个集群可以在更新集群时造成单点故障。反之，使用 *mirror registry for Red Hat OpenShift* 安装一个集群，该集群可以托管一个生产环境级别的、高度可用的 registry，如 Red Hat Quay，可将 OpenShift Container Platform 内容提供给其他集群。

4.3. 使用 MIRROR REGISTRY FOR RED HAT OPENSHIFT 镜像一个本地主机

此流程解释了如何使用 **mirror-registry** 安装程序工具在本地主机上安装 *mirror registry for Red Hat OpenShift*。这样，用户可以创建在端口 443 上运行的本地主机 registry，以存储 OpenShift Container Platform 镜像的镜像。



注意

使用 **mirror-registry** CLI 工具安装 *mirror registry for Red Hat OpenShift* 会对机器进行几个更改。安装后，会创建一个 **$HOME/quay-install** 目录，其中包含安装文件、本地存储和配置捆绑包。如果部署目标是本地主机，则生成可信 SSH 密钥，并且设置主机计算机上的 systemd 文件，以确保容器运行时持久。另外，会创建一个名为 **init** 的初始用户，并自动生成的密码。所有访问凭证都会在安装例程的末尾打印。

流程

1. 从 OpenShift console **Downloads** 页下载最新版本的 *mirror registry for Red Hat OpenShift* 的 **mirror-registry.tar.gz** 软件包。

2. 使用 **mirror-registry** 工具，在本地主机上安装 *mirror registry for Red Hat OpenShift*。有关可用标志的完整列表，请参阅 "mirror registry for Red Hat OpenShift flags"。

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

3. 运行以下命令，使用安装期间生成的用户名和密码登录 registry：

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

1 您可以通过将您的系统配置为信任生成的 rootCA 证书来避免运行 **--tls-verify=false**。如需更多信息，请参阅"保护 Red Hat Quay"和"将系统配置为信任证书颁发机构"。

注意

您还可以在安装后通过 **https://<host.example.com>:8443** 访问 UI 登录。

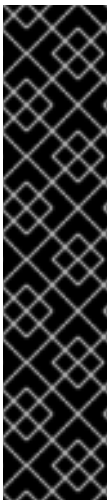
4. 您可以在登录后镜像 OpenShift Container Platform 镜像。根据您的需要，请参阅本文档的"镜像 OpenShift Container Platform 镜像存储库"或"镜像 Operator 目录"部分以用于此文档。

注意

如果因为存储层问题导致 *mirror registry for Red Hat OpenShift* 存储的镜像有问题，您可以在一个更稳定的存储上重新 mirror OpenShift Container Platform 镜像，或重新安装 mirror registry。

4.4. 从一个本地主机更新 MIRROR REGISTRY FOR RED HAT OPENSHIFT

此流程解释了如何使用 **upgrade** 命令从本地主机更新 *mirror registry for Red Hat OpenShift*。更新至最新版本可确保新的功能、错误修复和安全漏洞修复。



重要

当从版本 1 升级到版本 2 时，请注意以下限制：

- worker 数量被设置为 **1**，因为 SQLite 中不允许多个写入。
- 您不能使用 *mirror registry for Red Hat OpenShift* 用户接口 (UP)。
- 不要在升级过程中访问 **sqlite-storage** Podman 卷。
- 镜像 registry 会出现间歇性停机时间，因为它会在升级过程中重启。
- PostgreSQL 数据在 **/$HOME/quay-install/quay-postgres-backup/** 目录下备份，以进行恢复。

先决条件

- 您已在本地主机上安装了 *mirror registry for Red Hat OpenShift*。

流程

- 如果您要将 *mirror registry for Red Hat OpenShift* 从 1.3 升级到 2.y，且您的安装目录默认为 **/etc/quay-install**，您可以输入以下命令：

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

注意

- *mirror registry for Red Hat OpenShift* 将 Red Hat Quay 的 Podman 卷、Postgres 数据和 **/etc/quay-install** 数据迁移到新的 **$HOME/quay-install** 位置。这可让您在以后的升级过程中使用没有 **--quayRoot** 标记的 *mirror registry for Red Hat OpenShift*。
- 在使用 **./mirror-registry upgrade -v** 标记升级 *mirror registry for Red Hat OpenShift* 时需要包括在创建 mirror registry 时使用的相同的凭证。例如，如果使用 **--quayHostname <host_example_com>** 和 **--quayRoot <example_directory_name>** 安装了 *mirror registry for Red Hat OpenShift*，则必须包括该字符串来正确地升级 mirror registry。

- 如果您要将 *mirror registry for Red Hat OpenShift* 从 1.3 升级到 2.y，且您在 1.y 部署中使用自定义 quay 配置和存储目录，则必须传递-- **quayRoot** 和- **quayStorage** 标志。例如：

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

- 如果您要从 1.3 升级到 2.y *mirror registry for Red Hat OpenShift*，并希望指定自定义 SQLite 存储路径，您必须传递 **--sqliteStorage** 标志，例如：

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

验证

1. 运行以下命令，确保 *mirror registry for Red Hat OpenShift* 已更新：

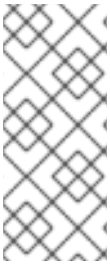
```
$ podman ps
```

输出示例

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

4.5. 使用 MIRROR REGISTRY FOR RED HAT OPENSHIFT 在远程主机上 MIRROR

此流程解释了如何使用 **mirror-registry** 工具在远程主机上安装 *mirror registry for Red Hat OpenShift*。这样，用户可以创建 registry 来保存 OpenShift Container Platform 镜像的镜像。



注意

使用 **mirror-registry** CLI 工具安装 *mirror registry for Red Hat OpenShift* 会对机器进行几个更改。安装后，会创建一个 **$HOME/quay-install** 目录，其中包含安装文件、本地存储和配置捆绑包。如果部署目标是本地主机，则生成可信 SSH 密钥，并且设置主机计算机上的 systemd 文件，以确保容器运行时持久。另外，会创建一个名为 **init** 的初始用户，并自动生成的密码。所有访问凭证都会在安装例程的末尾打印。

流程

1. 从 OpenShift console **Downloads** 页下载最新版本的 *mirror registry for Red Hat OpenShift* 的 **mirror-registry.tar.gz** 软件包。

2. 使用 **mirror-registry** 工具，在本地主机上安装 *mirror registry for Red Hat OpenShift*。有关可用标志的完整列表，请参阅 "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. 运行以下命令，使用安装期间生成的用户名和密码登录到 mirror registry：

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

1 您可以通过将您的系统配置为信任生成的 rootCA 证书来避免运行 **--tls-verify=false**。如需更多信息，请参阅"保护 Red Hat Quay"和"将系统配置为信任证书颁发机构"。

注意

您还可以在安装后通过 **https://<host.example.com>:8443** 访问 UI 登录。

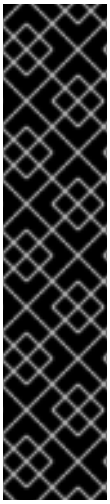
4. 您可以在登录后镜像 OpenShift Container Platform 镜像。根据您的需要，请参阅本文档的"镜像 OpenShift Container Platform 镜像存储库"或"镜像 Operator 目录"部分以用于此文档。

注意

如果因为存储层问题导致 *mirror registry for Red Hat OpenShift* 存储的镜像有问题，您可以在一个更稳定的存储上重新 mirror OpenShift Container Platform 镜像，或重新安装 mirror registry。

4.6. 从一个远程主机为 RED HAT OPENSHIFT 更新 MIRROR REGISTRY

此流程解释了如何使用 **upgrade** 命令从远程主机更新 *mirror registry for Red Hat OpenShift*。更新至最新版本可确保程序错误修复和安全漏洞。



重要

当从版本 1 升级到版本 2 时，请注意以下限制：

- worker 数量被设置为 **1**，因为 SQLite 中不允许多个写入。
- 您不能使用 *mirror registry for Red Hat OpenShift* 用户接口 (UP)。
- 不要在升级过程中访问 **sqlite-storage** Podman 卷。
- 镜像 registry 会出现间歇性停机时间，因为它会在升级过程中重启。
- PostgreSQL 数据在 **/$HOME/quay-install/quay-postgres-backup/** 目录下备份，以进行恢复。

先决条件

- 您已在远程主机上安装了 *mirror registry for Red Hat OpenShift*。

流程

- 要从远程主机升级 *mirror registry for Red Hat OpenShift*，请输入以下命令：

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

注意

在使用 **./mirror-registry upgrade -v** 标记升级 *mirror registry for Red Hat OpenShift* 时需要包括在创建 mirror registry 时使用的相同的凭证。例如，如果使用 **--quayHostname <host_example_com>** 和 **--quayRoot <example_directory_name>** 安装了 *mirror registry for Red Hat OpenShift*，则必须包括该字符串来正确地升级 mirror registry。

- 如果您要从 1.3 升级到 2.y *mirror registry for Red Hat OpenShift*，并希望指定自定义 SQLite 存储路径，您必须传递 **--sqliteStorage** 标志，例如：

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

验证

1. 运行以下命令，确保 *mirror registry for Red Hat OpenShift* 已更新：

```
$ podman ps
```

输出示例

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

4.7. 替换 MIRROR REGISTRY FOR RED HAT OPENSHIFT SSL/TLS 证书

在某些情况下，您可能想要为 *mirror registry for Red Hat OpenShift* 更新 SSL/TLS 证书。这在以下情况中很有用：

- 如果您需要替换当前的 *mirror registry for Red Hat OpenShift* 证书。
- 如果您使用与之前 *mirror registry for Red Hat OpenShift* 安装相同的证书。
- 如果您希望定期更新 *mirror registry for Red Hat OpenShift* 证书。

使用以下步骤替换 *mirror registry for Red Hat OpenShift* SSL/TLS 证书。

先决条件

- 您已从 OpenShift 控制台的 **Downloads** 页中下载并安装了 **./mirror-registry** 二进制文件。

流程

1. 输入以下命令安装 *mirror registry for Red Hat OpenShift*：

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

   这会将 *mirror registry for Red Hat OpenShift* 安装到 **$HOME/quay-install** 目录中。

2. 准备一个新的证书颁发机构(CA)捆绑包，并生成新的 **ssl.key** 和 **ssl.crt** 密钥文件。如需更多信息，请参阅为 Red Hat Quay 配置 SSL 和 TLS 。

3. 输入以下命令为 **/$HOME/quay-install** 分配一个环境变量，如 **QUAY** ：

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

4. 输入以下命令将新的 **ssl.crt** 文件复制到 **/$HOME/quay-install** 目录中：

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

5. 输入以下命令将新的 **ssl.key** 文件复制到 **/$HOME/quay-install** 目录中：

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

6. 输入以下命令重启 **quay-app** 应用程序 pod：

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

4.8. 卸载 MIRROR REGISTRY FOR RED HAT OPENSHIFT

使用以下步骤从本地主机卸载 *Red Hat OpenShift 的 mirror registry*。

先决条件

- 您已在本地主机上安装了 *mirror registry for Red Hat OpenShift*。

流程

- 运行以下命令，从本地主机中卸载 *mirror registry for Red Hat OpenShift* ：

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

注意

- 删除 *mirror registry for Red Hat OpenShift* 会在删除前提示用户。您可以使用 **--autoApprove** 来跳过此提示。
- 如果使用 **--quayRoot** 标志安装了 *mirror registry for Red Hat OpenShift*，则卸载时也需要使用 **--quayRoot** 标志。例如，如果您安装了带有 **--quayRoot example_directory_name** 的 *Red Hat OpenShift 的 mirror registry*，则必须包含该字符串才能正确卸载 mirror registry。

4.9. MIRROR REGISTRY FOR RED HAT OPENSHIFT 标记

以下标记可用于 *mirror registry for Red Hat OpenShift* ：

标记	描述
--autoApprove	禁用交互式提示的布尔值。如果设置为 **true**，则在卸载镜像 registry 时自动删除 **quayRoot** 目录。如果未指定，则默认为 **false**。
--initPassword	在 Quay 安装过程中创建的 init 用户的密码。必须至少包含八个字符，且不包含空格。
--initUser string	显示初始用户的用户名。若未指定，则默认为 **init**。
--no-color, **-c**	允许用户禁用颜色序列，在运行安装、卸载和升级命令时将其传播到 Ansible。
--quayHostname	客户端用来联系 registry 的镜像 registry 的完全限定域名。等同于 Quay **config.yaml** 中的 **SERVER_HOSTNAME**。必须可以被 DNS 解析。如果未指定，则默认为 **<targetHostname>:8443**。[1]
--quayStorage	保存 Quay 持久性存储数据的文件夹。默认为 **quay-storage** Podman 卷。卸载需要 root 权限。
--quayRoot, **-r**	保存容器镜像层和配置数据的目录，包括 **rootCA.key**、rootCA**.pem** 和 **rootCA.srl** 证书。如果未指定，则默认为 **$HOME/quay-install**。
--sqliteStorage	保存 SQLite 数据库数据的文件夹。如果没有指定，则默认为 **sqlite-storage** Podman 卷。卸载需要 root。
--ssh-key,**-k**	SSH 身份密钥的路径。如果未指定，则默认为 **~/.ssh/quay_installer**。

标记	描述
--sslCert	SSL/TLS 公钥/证书的路径。默认为 **{quayRoot}/quay-config**，并在未指定时自动生成。
--sslCheckSkip	跳过对 **config.yaml** 文件中的 **SERVER_HOSTNAME** 的检查证书主机名。[2]
--sslKey	用于 HTTPS 通信的 SSL/TLS 私钥路径。默认为 **{quayRoot}/quay-config**，并在未指定时自动生成。
--targetHostname, -H	要安装 Quay 的目标的主机名。默认为 **$HOST**，如本地主机（如果未指定）。
--targetUsername, -u	目标主机上的用户，将用于 SSH。默认为 **$USER**，例如，如果未指定，则默认为当前用户。
--verbose,-v	显示调试日志和 Ansible playbook 输出。
--version	显示 *mirror registry for Red Hat OpenShift*的版本。

1. 如果您的系统的公共 DNS 名称与本地主机名不同，则必须修改 **--quayHostname**。另外，**--quayHostname** 标志不支持使用 IP 地址的安装。需要使用主机名进行安装。
2. 当镜像 registry 在代理后面设置时，会使用 **--sslCheckSkip**，并且公开的主机名与内部 Quay 主机名不同。当用户不希望在安装过程中对提供的 Quay 主机名验证证书时，也可以使用它。

4.10. MIRROR REGISTRY FOR RED HAT OPENSHIFT 发行注记

mirror registry for Red Hat OpenShift 是一个小型灵活的容器 registry，作为目标，用于为断开连接的安装镜像(mirror)的 OpenShift Container Platform 所需的容器镜像。

本发行注记介绍了 OpenShift Container Platform 的 *mirror registry for Red Hat OpenShift* 。

4.10.1. Mirror registry for Red Hat OpenShift 2.0 发行注记

以下小节详细介绍了 mirror registry for Red Hat OpenShift 的每个 2.0 发行版本。

4.10.1.1. Mirror registry for Red Hat OpenShift 2.0.8

发布日期： 2025 年 10 月 16 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.12 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.2. Mirror registry for Red Hat OpenShift 2.0.7

发布日期： 2025 年 7 月 14 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.10 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.3. Mirror registry for Red Hat OpenShift 2.0.6

发布日期：2025 年 4 月 28 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.8 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.4. Mirror registry for Red Hat OpenShift 2.0.5

发布日期：2025 年 1 月 13 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.5 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.5. Mirror registry for Red Hat OpenShift 2.0.4

发布日期： 2025 年 1 月 6 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.4 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.6. Mirror registry for Red Hat OpenShift 2.0.3

发布日期： 2024 年 11 月 25 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.3 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.7. Mirror registry for Red Hat OpenShift 2.0.2

发布日期：2024 年 10 月 31 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.2 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.8. Mirror registry for Red Hat OpenShift 2.0.1

发布日期：2024 年 9 月 26 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.1 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

4.10.1.9. Mirror registry for Red Hat OpenShift 2.0.0

发布日期： 2024 年 9 月 3 日

Mirror registry for Red Hat OpenShift 现在包括在 Red Hat Quay 3.12.0 中。

以下公告适用于 *mirror registry for Red Hat OpenShift* ：

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

以下新功能包括在 *mirror registry for Red Hat OpenShift* 2.0.0 中：

- 随着 *mirror registry for Red Hat OpenShift* 的发布，内部数据库已从 PostgreSQL 升级到 SQLite。因此，数据现在默认存储在 **sqlite-storage** Podman 卷中，整个 tarball 大小会减少 300 MB。
  新的安装默认使用 SQLite。在升级到 2.0 之前，请参阅"根据您的环境，"从本地主机更新 mirror registry for Red Hat OpenShift"或"从远程主机更新 mirror registry for Red Hat OpenShift"。
- 添加了新功能标志 **--sqliteStorage**。使用这个标志，您可以手动设置保存 SQLite 数据库数据的位置。
- *Mirror registry for Red Hat OpenShift* 现在可用于 IBM Power 和 IBM Z 架构（ **s390x** 和 **ppc64le**）。

4.10.2. Mirror registry for Red Hat OpenShift 1.3 发行注记

要查看 *mirror registry for Red Hat OpenShift* 1.3 发行注记，请参阅 Mirror registry for Red Hat OpenShift 1.3 发行注记。

4.10.3. Mirror registry for Red Hat OpenShift 1.2 发行注记

要查看 *mirror registry for Red Hat OpenShift* 1.2 发行注记，请参阅 Mirror registry for Red Hat OpenShift 1.2 发行注记。

4.10.4. Mirror registry for Red Hat OpenShift 1.1 发行注记

要查看 *mirror registry for Red Hat OpenShift* 1.1 发行注记，请参阅 Mirror registry for Red Hat OpenShift 1.1 发行注记。

4.11. MIRROR REGISTRY FOR RED HAT OPENSHIFT 故障排除

为了帮助对 *mirror registry for Red Hat OpenShift* 进行故障排除，您可以收集由 mirror registry 安装的 systemd 服务的日志。安装以下服务：

- quay-app.service

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

先决条件

- 您已安装了 *mirror registry for Red Hat OpenShift*。

流程

- 如果使用 root 权限安装 *mirror registry for Red Hat OpenShift*，您可以输入以下命令获取其 systemd 服务的状态信息：

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

- 如果作为标准用户安装 *mirror registry for Red Hat OpenShift*，您可以输入以下命令获取其 systemd 服务的状态信息：

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

4.12. 其他资源

- Red Hat Quay 垃圾回收
- 保护 Red Hat Quay
- 将系统配置为信任证书颁发机构
- 镜像 OpenShift Container Platform 镜像存储库
- 镜像用于断开连接的集群的 Operator 目录

第 5 章 使用 OC-MIRROR 插件 V2 为断开连接的安装 MIRROR 镜像

如果从私有 registry 中的镜像 OpenShift Container Platform 容器镜像安装集群，则在断开连接的环境中运行集群。每当集群运行时，此 registry 必须正在运行。

您可以使用 oc-mirror 插件 v2 在完全或部分断开连接的环境中将镜像 mirror 到mirror registry。要从官方红帽 registry 下载所需的镜像，您必须从具有互联网连接的系统运行 oc-mirror 插件 v2。

5.1. 关于 OC-MIRROR 插件 V2

oc-mirror OpenShift CLI (**oc**) 插件是一个单一工具，可将所有所需的 OpenShift Container Platform 内容和其他镜像 (mirror)镜像到您的镜像 registry。

要使用 oc-mirror 的新版本，请在 oc-mirror 插件 v2 命令行中添加 **--v2** 标志。

oc-mirror 插件 v2 具有以下功能：

- 提供镜像 OpenShift Container Platform 发行版本、Operator、helm chart 和其他镜像的集中方法。
- 验证镜像设置配置中指定的完整镜像集是否已镜像到已镜像的 registry，无论镜像之前是否被镜像 (mirror)。
- 使用缓存系统而不是元数据，这可以防止在过程中的一步中出现故障时启动镜像过程。
- 通过将新镜像合并到存档中来维护最小归档大小。
- 使用通过镜像日期选择的内容生成镜像存档。
- 可以生成 **ImageDigestMirrorSet** (IDMS)和 **ImageTagMirrorSet** (ITMS)资源，这些资源覆盖了使用 v1 的每个镜像集合的镜像集，而不是 **ImageContentSourcePolicy** (ICSP)资源，它包括了对每个镜像操作设置的增量更改。
- 不执行自动修剪。v2 现在使用 **Delete** 功能，这可让用户更好地控制对镜像进行删除。
- 引入了对 **registry.conf** 文件的支持。此更改有助于在使用相同的缓存时镜像到多个 enclaves。

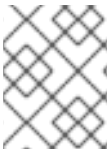
5.1.1. 高级别工作流

以下步骤概述了如何使用 oc-mirror 插件 v2 将镜像 mirror 到 mirror registry 的高级别工作流：

1. 创建镜像设置配置文件。
2. 使用以下工作流之一将镜像设置为目标镜像 registry：
   - 将镜像直接设置为目标镜像 registry (mirror 到 mirror)。
   - 将镜像设置为 disk (mirror to disk)，将 **tar** 文件传送到目标环境，然后将镜像设置为目标镜像 registry （disk to mirror)。
3. 配置集群以使用 oc-mirror 插件 v2 生成的资源。
4. 根据需要重复这些步骤以更新目标镜像 registry。

5.1.2. oc-mirror 插件 v2 兼容性和支持

OpenShift Container Platform 支持 oc-mirror 插件 v2。



注意

在 **aarch64**,**ppc64le**, 和 **s390x** 架构中，oc-mirror 插件 v2 只支持 OpenShift Container Platform 4.14 及更新的版本。

使用 oc-mirror 插件 v2 的最新可用版本，无论您需要镜像的 OpenShift Container Platform 版本是什么。

5.2. 先决条件

- 您必须在托管 OpenShift Container Platform 集群的位置（如 Red Hat Quay）中有一个支持 Docker V2-2 的容器镜像 registry。

注意

  - 如果使用 Red Hat Quay，请在 oc-mirror 插件中使用 3.6 或更高版本。请参阅在 OpenShift Container Platform 上部署 Red Hat Quay Operator (Red Hat Quay 文档)。如果您需要额外的帮助来选择并安装 registry，请联络您的销售代表或红帽支持。
  - 如果您没有容器镜像 registry 的现有解决方案，OpenShift Container Platform 订阅者会收到一个 mirror registry for Red Hat OpenShift。此镜像 registry 包含在您的订阅中，并充当小型容器 registry。您可以使用此 registry 为断开连接的安装镜像 OpenShift Container Platform 所需的容器镜像。

- 置备的集群中的每个机器都必须有权访问镜像 registry。如果 registry 无法访问，安装、更新或常规操作等任务（如工作负载重新定位）可能会失败。镜像 registry 必须以高可用性的方式运行，确保其可用性与 OpenShift Container Platform 集群的生产环境可用性一致。

5.3. 准备您的镜像主机

要将 oc-mirror 插件 v2 用于镜像镜像，您需要安装插件并使用容器镜像凭证创建文件，使您能够从红帽镜像到您的镜像。

5.3.1. 安装 oc-mirror OpenShift CLI 插件

安装 oc-mirror OpenShift CLI 插件以在断开连接的环境中管理镜像集。

先决条件

- 已安装 OpenShift CLI(**oc**)。如果您在完全断开连接的环境中镜像镜像集，请确保以下内容：
  - 您已在可访问互联网的主机上安装了 oc-mirror 插件。
  - 在断开连接的环境中的主机可以访问目标镜像 registry。
- 您已在使用 oc-mirror 的操作系统中，将 **umask** 参数设置为 **0022**。
- 您已为您要使用的 RHEL 版本安装了正确的二进制文件。

流程

1. 下载 oc-mirror CLI 插件：
   a. 进入到 Red Hat Hybrid Cloud Console 的 Downloads 页。
   b. 在 **OpenShift disconnected installation tools** 部分中，从下拉菜单中选择 **OpenShift Client (oc) mirror plugin** 的**OS type** 和 **Architecture type**。
   c. 点 **Download** 保存文件。
2. 运行以下命令来提取存档：

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

3. 如有必要，运行以下命令来将插件文件更新为可执行：

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

注意

不要重命名 **oc-mirror** 文件。

4. 运行以下命令，将文件放在 **PATH** 中（如 **/usr/local/bin** ）来安装 oc-mirror CLI 插件：

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

验证

- 运行以下命令验证 oc-mirror 插件 v2 是否已成功安装：

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

5.3.2. 配置允许对容器镜像进行镜像的凭证

创建容器镜像 registry 凭证文件，可让您将镜像从红帽 mirror 到您的镜像。在安装主机上完成以下步骤。

警告

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

先决条件

- 您已将镜像 registry 配置为在断开连接的环境中使用。
- 您在镜像 registry 中标识了镜像仓库的位置，以将容器镜像镜像(mirror)到这个位置。
- 您置备了一个镜像 registry 帐户，允许将镜像上传到该镜像仓库。

- 对镜像 registry 有写权限。

流程

1. 从 Red Hat OpenShift Cluster Manager 下载 **registry.redhat.io** pull secret。
2. 运行以下命令，以 JSON 格式生成 pull secret 副本：

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

指定到存储 pull secret 的目录的路径，以及您创建的 JSON 文件的名称。

pull secret 示例

```
{
  "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. 如果 **$XDG_RUNTIME_DIR/containers** 目录不存在，请输入以下命令来创建：

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

2. 将 pull secret 文件保存为 **$XDG_RUNTIME_DIR/containers/auth.json**。

3. 运行以下命令，为您的镜像 registry 生成 base64 编码的用户名和密码或令牌：

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

通过 **<user_name>** 和 **<password>** 指定 registry 的用户名和密码。

输出示例

```
BGVtbYk3ZHAtqXs=
```

4. 编辑 JSON 文件并添加描述 registry 的部分：

```
  "auths": {
```

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

- 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如：**registry.example.com** 或 **registry.example.com:8443**。
- 对于 **<credentials>** 值，请指定 mirror registry 的 base64 编码用户名和密码。

修改的 pull secret 示例

```
{
  "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. 将镜像集镜像(MIRROR)到镜像 REGISTRY

将镜像集镜像到镜像 registry 可确保所需的镜像在安全且受控的环境中可用，从而促进平稳部署、更新和维护任务。

5.4.1. 创建镜像设置配置

在使用 oc-mirror 插件 v2 mirror 镜像之前，必须先创建镜像设置配置文件。此镜像设置配置文件定义哪些 OpenShift Container Platform 发行版本、Operator 和其他要 mirror 的镜像，以及 oc-mirror 插件 v2 的其他配置设置。

先决条件

- 您已创建了容器镜像 registry 凭证文件。具体步骤，请参阅*配置允许镜像镜像的凭证*。

流程

- 创建 **ImageSetConfiguration** YAML 文件，并进行修改使其包含所需的镜像。

ImageSetConfiguration YAML 文件示例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v2alpha1
mirror:
  platform:
    channels:
    - name: stable-4.19 (1)
      minVersion: 4.19.2
      maxVersion: 4.19.2
    graph: true (2)
  operators:
    - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19 (3)
      packages: (4)
       - name: aws-load-balancer-operator
       - name: 3scale-operator
       - name: node-observability-operator
  additionalImages: (5)
   - name: registry.redhat.io/ubi8/ubi:latest
   - name:
registry.redhat.io/ubi9/ubi@sha256:20f695d2a91352d4eaa25107535126727b5945bff38ed36a
3e59590f495046f0
```

(1) 将频道设置为从中检索 OpenShift Container Platform 镜像。

(2) 添加 **graph: true** 以构建并推送 graph-data 镜像推送到镜像 registry。创建 OpenShift Update Service (OSUS) 需要 graph-data 镜像。**graph: true** 字段还会生成 **UpdateService** 自定义资源清单。**oc** 命令行界面 (CLI) 可以使用 **UpdateService** 自定义资源清单来创建 OSUS。如需更多信息，请参阅*关于 OpenShift Update Service*。

(3) 将 Operator 目录设置为从中检索 OpenShift Container Platform 镜像。

(4) 仅指定要包含在镜像集中的某些 Operator 软件包。删除此字段以检索目录中的所有软件包。

(5) 指定要在镜像集中包含的任何其他镜像。

注意

在 oc-mirror 插件 v2 中，您必须为 **additionalImages** 下列出的所有镜像使用显式 registry 主机名。否则，镜像会被镜像到不正确的目标路径。

其他资源

- 关于 OpenShift Update 服务

5.4.2. oc-mirror 工作流的比较

使用下表比较 mirror-to-disk (m2d), disk-to-mirror (d2m), 和 mirror-to-mirror (m2m) 支持的用例。

使用案例	Mirror To Disk (m2d)和 Disk To Mirror (d2m)	Mirror To Mirror (m2m)
目标 registry 存在于没有互联网访问且没有外部访问的环境中。	✓	
目标 registry 存在于没有互联网访问但可以从另一台机器访问的环境中。例如，目标 registry 位于堡垒主机中。		✓
您必须使用一个物理方法（如使用 USB 设备）将内容移到断开连接的环境中。	✓	
工作流将内容直接移到目标 registry 中，而无需生成中间 tar 文件。		✓
工作流使用一个内部缓存以在失败后恢复，但需要额外的磁盘空间。	✓	
工作流不使用缓存，在失败后需要重新开始整个过程，且不需要额外的磁盘空间。		✓

5.4.3. 在部分断开连接的环境中镜像设置的镜像

您可以在带有受限互联网访问的环境中使用 oc-mirror 插件 v2 将镜像集镜像到 registry。

先决条件

- 在运行 oc-mirror 插件 v2 的环境中，您可以访问互联网和镜像 registry。

流程

- 运行以下命令，将指定镜像设置配置中的镜像镜像到指定的 registry：

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

其中：

<image_set_configuration>
指定镜像设置配置文件的名称。

<file_path>
指定生成集群资源的目录。

<mirror_registry_url>
指定镜像存储并从中删除镜像 registry 的 URL 或地址。

验证

1. 进入 **<file_path>** 目录中生成的 **working-dir/cluster-resources** 目录。

2. 验证 **ImageDigestMirrorSet**、**ImageTagMirrorSet** 和 **CatalogSource** 资源是否存在 YAML 文件。

后续步骤

- 配置集群以使用 oc-mirror 插件 v2 生成的资源。

5.4.4. 镜像在完全断开连接的环境中设置的镜像

您可以在 OpenShift Container Platform 集群无法访问互联网的完全断开连接的环境中镜像镜像集。以下高级别工作流描述了镜像过程：

1. **Mirror to disk** ：将镜像集 mirror 到一个存档。
2. **Disk transfer** ：手动将存档传输到断开连接的镜像 registry 的网络。
3. **Disk to mirror**: 将存档中的镜像集 mirror 到目标断开连接的 registry。

5.4.4.1. 从镜像镜像到磁盘

您可以使用 oc-mirror 插件 v2 生成镜像集，并将内容保存到磁盘。之后，您可以将生成的镜像设置为断开连接的环境，并将其镜像到目标 registry。

oc-mirror 插件 v2 从镜像设置配置中指定的源中检索容器镜像，并将它们打包到本地目录中的 tar 存档中。

流程

- 运行以下命令，将指定镜像集配置中的镜像镜像到磁盘：

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

其中：

<image_set_configuration>
指定镜像设置配置文件的名称。

<file_path>
指定在其中生成包含镜像集的存档的目录。

验证

1. 进入生成的 **<file_path>** 目录。
2. 验证存档文件是否已生成。

后续步骤

- 从磁盘镜像到镜像

5.4.4.2. 从磁盘镜像到镜像

您可以使用 oc-mirror 插件 v2 将磁盘中的镜像集镜像到目标镜像 registry。

oc-mirror 插件 v2 从本地磁盘检索容器镜像并将其传送到指定的镜像 registry。

流程

1. 将包含 mirror 镜像集的磁盘传输到包含目标镜像 registry 的环境。
2. 运行以下命令，处理磁盘上的镜像集文件，并将内容镜像到目标镜像 registry：

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

其中：

<image_set_configuration>

指定镜像设置配置文件的名称。

<file_path>

指定包含存档的磁盘上的目录。此文件夹还包含您为集群生成的集群资源，如 ImageDigestMirrorSet (IDMS) 或 ImageTagMirrorSet (ITMS) 资源。

<mirror_registry_url>

指定用于存储镜像的 mirror registry 的 URL 或地址。

验证

1. 进入到 **working-dir** 中的 **cluster-resources** 目录，它在 **<file_path>** 目录中生成。
2. 验证 **ImageDigestMirrorSet**、**ImageTagMirrorSet** 和 **CatalogSource** 资源是否存在 YAML 文件。

后续步骤

- 配置集群以使用 oc-mirror 插件 v2 生成的资源。

5.5. 关于 OC-MIRROR 插件 V2 生成的自定义资源

oc-mirror 插件 v2 会自动生成以下自定义资源：

ImageDigestMirrorSet (IDMS)

在使用镜像摘要拉取规格时处理 registry 镜像规则。如果至少一个镜像集的镜像由摘要镜像，则会生成。

ImageTagMirrorSet (ITMS)

在使用镜像标签 pull 规格时处理 registry 镜像规则。如果至少通过标签对镜像集中的一个镜像进行镜像生成。

CatalogSource

检索有关镜像 registry 中可用 Operator 的信息。由 Operator Lifecycle Manager (OLM) Classic 使用。

ClusterCatalog

检索有关镜像 registry 中可用集群扩展（包括 Operator）的信息。由 OLM v1 使用。

UpdateService

为断开连接的环境提供更新图形数据。由 OpenShift Update Service 使用。

其他资源

- CatalogSource
- ImageDigestMirrorSet
- ImageTagMirrorSet
- 关于 OLM v1 中的目录

5.5.1. 修改由 oc-mirror 插件生成的资源的限制

当使用 oc-mirror 插件 v2 生成的资源来配置集群时，某些字段不能更改。修改这些字段可能会导致错误且不被支持。

下表列出了必须保持不变的资源及其字段：

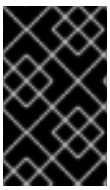
表 5.1. 不得修改 oc-mirror 生成的资源中的字段

资源	不能更改的字段
CatalogSource	**apiVersion**, **kind**, **spec.image**
ClusterCatalog	**apiVersion**, **kind**, **spec.source.image.ref**
ImageDigestMirrorSet	**apiVersion**, **kind**, **spec.imageDigestMirrors**
ImageTagMirrorSet	**apiVersion**, **kind**, **spec.imageTagMirrors**
签名 **ConfigMap**	**apiVersion**, **kind**, **metadata.namespace**, **binaryData**
UpdateService	**apiVersion**, **kind**, **spec.graphDataImage**, **spec.releases**

有关这些资源的更多信息，请参阅 **CatalogSource**、**ImageDigestMirrorSet** 和 **ImageTagMirrorSet** 的 OpenShift API 文档。

5.5.2. 配置集群以使用 oc-mirror 插件 v2 生成的资源

在将您的镜像集 mirror 到镜像 registry 后，您必须将生成的 **ImageDigestMirrorSet** (IDMS)、**ImageTagMirrorSet** (ITMS)、**CatalogSource** 和 **UpdateService** 资源应用到集群。



重要

在 oc-mirror 插件 v2 中，IDMS 和 ITMS 文件涵盖了整个镜像集，这与 oc-mirror 插件 v1 中的 **ImageContentSourcePolicy** (ICSP)文件不同。因此，即使您仅在增量镜像过程中添加新镜像，IDMS 和 ITMS 文件还包含集合的所有镜像。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

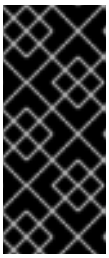
流程

1. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift CLI。
2. 运行以下命令，将结果目录中的 YAML 文件应用到集群：

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

3. 如果镜像(mirror)镜像，请运行以下命令将发行版本镜像签名应用到集群：

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



重要

如果您要镜像 Operator 而不是集群，请不要运行前面的命令。运行命令会导致错误，因为没有要应用的发行镜像签名。

另外，在相同的目录 **working-dir/cluster-resources/** 中提供了 YAML 文件。您可以使用 JSON 或 YAML 格式。

验证

1. 运行以下命令验证 **ImageDigestMirrorSet** 资源是否已成功安装：

```
$ oc get imagedigestmirrorset
```

要只查看 **oc-mirror** 创建的资源，请运行以下命令：

```
$ oc get imagedigestmirrorset -o jsonpath='{.items[?
(@.metadata.annotations.createdBy=="oc-mirror v2")].metadata.name}'
```

2. 运行以下命令验证 **ImageTagMirrorSet** 资源是否已成功安装：

```
$ oc get imagetagmirrorset
```

要只查看 **oc-mirror** 创建的资源，请运行以下命令：

```
$ oc get imagetagmirrorset -o jsonpath='{.items[?(@.metadata.annotations.createdBy=="oc-
mirror v2")].metadata.name}'
```

3. 运行以下命令验证 **CatalogSource** 资源是否已成功安装：

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

要只查看 **oc-mirror** 创建的资源，请运行以下命令：

```
$ oc get catalogsource -o jsonpath='{.items[?(@.metadata.annotations.createdBy=="oc-
mirror v2")].metadata.name}'
```

4. 运行以下命令验证 **ClusterCatalog** 资源是否已成功安装：

```
$ oc get clustercatalog
```

- 要只查看 **oc-mirror** 创建的资源，请运行以下命令：

```
$ oc get clustercatalog -o jsonpath='{.items[?(@.metadata.annotations.createdBy=="oc-mirror v2")].metadata.name}'
```

在将集群配置为使用 oc-mirror 插件 v2 生成的资源后，请参阅下一步以了解有关您可以使用 mirror 镜像执行的任务的信息。

其他资源

- OLM v1 中的断开连接的环境支持

5.6. 在断开连接的环境中删除镜像

如果您之前使用 oc-mirror 插件 v2 部署镜像，您可以删除这些镜像来释放镜像 registry 中的空间。oc-mirror 插件 v2 不会自动修剪没有包括在 **ImageSetConfiguration** 文件中的镜像。这可防止在对 **ImageSetConfig.yaml** 文件进行更改时意外删除必要的或部署的镜像。

您必须创建一个 **DeleteImageSetConfiguration** 文件，以指定要删除的镜像。

在以下示例中，**DeleteImageSetConfiguration** 文件会删除以下镜像：

- OpenShift Container Platform 4.13.3 的所有发行镜像。
- **aws-load-balancer-operator** v0.0.1 捆绑包及其所有相关镜像。
- **ubi** 和 **ubi-minimal** 的额外镜像，通过它们对应的摘要引用。

DeleteImageSetConfiguration 文件示例

```
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:bce7e9f69fb7d4533447232478fd825811c760288f87a35699f9c8f030f2c1a6
    - name: registry.redhat.io/ubi8/ubi-
minimal@sha256:8bedbe742f140108897fb3532068e8316900d9814f399d676ac78b46e740e34e
```

重要

- 考虑使用 mirror-to-disk 和 disk-to-mirror 工作流来减少删除问题。
- 在 oc-mirror 插件 v2 中，您必须为 **additionalImages** 下列出的所有镜像使用显式 registry 主机名。否则，镜像会被镜像到不正确的目标路径。

oc-mirror 插件 v2 只删除镜像的清单，这不会减少 registry 中占用的存储。

要从不必要的镜像（如带有删除的清单）中释放存储空间，您必须在容器 registry 上启用垃圾收集器。启用垃圾收集器后，registry 将删除不再引用任何清单的镜像 Blob，从而减少之前由已删除 Blob 占用的存储。启用垃圾收集器的过程因容器 registry 而异。

如需更多信息，请参阅"在分发 registry 中解决存储清理问题"。

重要

- 要在删除 Operator 镜像时跳过删除 Operator 目录镜像，您必须在 **DeleteImageSetConfiguration** 文件中列出 Operator 目录镜像下的特定 Operator。这样可确保只有指定的 Operator 被删除，而不是目录镜像。
  如果只指定 Operator 目录镜像，则该目录中的所有 Operator 以及目录镜像本身都将被删除。
- oc-mirror 插件 v2 不会自动删除 Operator 目录镜像，因为其他 Operator 可能仍然被部署并依赖于这些镜像。
  如果您不确定目录中的 Operator 没有保留在 registry 或集群中，您可以在 **DeleteImageSetConfiguration** 中明确将目录镜像添加到 **additionalImages** 中以移除它。
- 垃圾回收行为取决于 registry。有些 registry 不会自动删除已删除的镜像，需要系统管理员手动触发垃圾回收来释放空间。

其他资源

- 解决分发 registry 中的存储清理问题

5.6.1. 解决分发 registry 中的存储清理问题

分发 registry 中的一个已知问题会阻止垃圾收集器按预期释放存储。使用 Red Hat Quay 时不会出现这个问题。

流程

- 选择发行 registry 中已知问题的适当方法：
  - 要重启容器 registry，请运行以下命令：

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

  - 要在 registry 配置中禁用缓存，请执行以下步骤：
    i. 要禁用 **blobdescriptor** 缓存，请修改 **/etc/docker/registry/config.yml** 文件：

    ```
    version: 0.1
    log:
    ```

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

ii. 要应用这些更改，请运行以下命令重启容器 registry：

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

5.6.2. 从断开连接的环境中删除镜像

要使用 oc-mirror 插件 v2 从断开连接的环境中删除镜像，请按照以下步骤操作。

先决条件

- 您已在环境中启用了垃圾回收，以删除不再引用清单的镜像。

流程

1. 创建一个 **delete-image-set-config.yaml** 文件，并包含以下内容：

   DeleteImageSetConfiguration 文件

```
apiVersion: mirror.openshift.io/v2alpha1
kind: DeleteImageSetConfiguration
delete:
  platform:
    channels:
      - name: <channel_name> 1
        minVersion: <channel_min_version> 2
        maxVersion: <channel_max_version> 3
  operators:
    - catalog: <operator_catalog_name> 4
      packages:
      - name: <operator_name> 5
         minVersion: <operator_max_version> 6
         maxVersion: <operator_min_version> 7
  additionalImages:
    - name: <additional_images>
```

1 1 指定要删除的 OpenShift Container Platform 频道的名称，如 **stable-4.15**。

2 3 指定要在频道中删除的镜像版本范围，例如，**4.15.0** 为最小版本，**4.15.1** 为最大版本。要只删除一个版本的镜像，在 **minVersion** 和 **maxVersion** 字段中都使用要删除的版本号。

4 指定包含要删除的 Operator 的 Operator 目录镜像，如 **registry.redhat.io/redhat/redhat-operator-index:v4.14**。Operator 目录镜像不会被删除。对于集群中剩余的其他 Operator，可能需要在 registry 中存在。

5 指定要删除的特定 Operator，如 **aws-load-balancer-operator**。

6 7 指定为 Operator 删除的镜像版本范围，如最小版本为 **0.0.1**，最大版本为**0.0.2**。

2. 运行以下命令，创建一个 **delete-images.yaml** 文件：

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

其中：

<previously_mirrored_work_folder>

指定镜像以前被 mirror 到的或在 mirror 过程中存储镜像的目录。

<remote_registry>

指定将从中删除镜像的远程容器 registry 的 URL 或地址。

重要

在删除镜像时，指定正确的工作区目录。仅在从头开始镜像时修改或删除缓存目录，如设置一个新集群。如果对缓存目录的更改不正确，可能会破坏进一步的镜像操作。

3. 进入创建的 **<previously_mirrored_work_folder>/delete** 目录。

4. 验证 **delete-images.yaml** 文件是否已生成。

5. 手工确保文件中列出的每个镜像不再被集群需要，可以安全地从 registry 中删除。

6. 生成 **delete-images** YAML 文件后，运行以下命令来从远程 registry 中删除镜像：

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

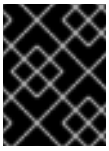
其中：

<previously_mirrored_work_folder>

指定镜像以前被 mirror 到的或在 mirror 过程中存储镜像的目录。

<remote_registry>

指定将从中删除镜像的远程容器 registry 的 URL 或地址。



重要

当使用 mirror-to-mirror 方法 mirror 镜像时，镜像不会在本地缓存，因此您无法从本地缓存中删除镜像。

5.7. 验证您选择的镜像以进行镜像

您可以使用 oc-mirror 插件 v2 执行不实际镜像任何镜像的测试运行(dry run)。这可让您查看要镜像的镜像列表。您也可以在早期使用空运行来捕获与镜像设置配置的任何错误。在镜像到磁盘工作流上运行空运行时，oc-mirror 插件 v2 会检查镜像集中的所有镜像是否在其缓存中可用。**missing.txt** 文件中列出了任何缺少的镜像。在镜像前执行空运行时，**missing.txt** 和 **mapping.txt** 文件都包含相同的镜像列表。

5.7.1. 为 oc-mirror 插件 v2 执行空运行

通过在不镜像的情况下执行空运行来验证您的镜像设置。这样可确保您的设置正确，并防止意外更改。

流程

- 要执行测试运行，请运行 **oc mirror** 命令，并在命令中使用 **--dry-run** 参数：

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

其中：

<image_set_config_yaml>
指定您创建的镜像设置配置文件。

<oc_mirror_workspace_path>
插入工作区路径的地址。

<mirror_registry_url>
插入要从中镜像或删除的远程容器 registry 的 URL 或地址。

输出示例

```
[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
```

验证

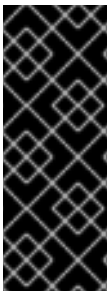
1. 进入生成的工作区目录：

```
$ cd <oc_mirror_workspace_path>
```

2. 检查生成的 **mapping.txt** 和 **missing.txt** 文件。这些文件包含将要镜像的所有镜像的列表。

5.7.2. oc-mirror 插件 v2 错误故障排除

oc-mirror 插件 v2 现在会在单独的文件中记录所有镜像同步错误，从而更轻松地跟踪和诊断失败。



重要

如果镜像（mirror）发行版本或发行组件镜像时出现错误，它们至关重要。这会立即停止镜像（mirror）过程。

镜像 (mirror) Operator、与 Operator 相关的镜像或其他镜像的错误不会停止镜像（mirror）过程。镜像过程会继续，oc-mirror 插件 v2 会在 **working-dir/logs** 目录下保存一个文件，描述哪些 Operator 无法镜像(mirror)。

当一个镜像无法被镜像（mirror）时，该镜像会作为一个或多个 Operator 捆绑包的一部分被镜像 (mirror)，oc-mirror 插件 v2 会通知用户 Operator 不完整，从而明确提供受错误影响的 Operator 捆绑包的信息。

流程

1. 检查与服务器相关的问题：

错误示例

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

   a. 在 oc-mirror 插件 v2 输出目录中，打开 **working-dir/logs** 文件夹中的 **mirroring_error_date_time.log** 文件。

   b. 查找表示服务器端问题的错误消息，如 **HTTP 500** 错误、令牌过期或超时。

   c. 如果问题仍然存在，请重试镜像过程或联系技术支持。

2. 检查对 Operator 进行镜像的过程不完整：

错误示例

```
error mirroring image docker://registry.redhat.io/3scale-amp2/zync-
rhel9@sha256:8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d
(Operator bundles: [3scale-operator.v0.11.12] - Operators: [3scale-operator]) error: initializing
source docker://localhost:55000/3scale-amp2/zync-
rhel9:8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d: reading
manifest 8bb6b31e108d67476cc62622f20ff8db34efae5d58014de9502336fcc479d86d in
localhost:55000/3scale-amp2/zync-rhel9: manifest unknown
error mirroring image docker://registry.redhat.io/3scale-amp2/3scale-rhel7-operator-
```

> metadata@sha256:de0a70d1263a6a596d28bf376158056631afd0b6159865008a7263a8e9bf0c7d error: skipping operator bundle docker://registry.redhat.io/3scale-amp2/3scale-rhel7-operator-metadata@sha256:de0a70d1263a6a596d28bf376158056631afd0b6159865008a7263a8e9bf0c7d because one of its related images failed to mirror
> error mirroring image docker://registry.redhat.io/3scale-amp2/system-rhel7@sha256:fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e (Operator bundles: [3scale-operator.v0.11.12] - Operators: [3scale-operator]) error: initializing source docker://localhost:55000/3scale-amp2/system-rhel7:fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e: reading manifest fe77272021867cc6b6d5d0c9bd06c99d4024ad53f1ab94ec0ab69d0fda74588e in localhost:55000/3scale-amp2/system-rhel7: manifest unknown

   a. 在控制台或日志文件中检查相关的警告，以了解哪些 Operator 不完整。
   如果一个 Operator 被标记为不完整，与该 Operator 相关的镜像可能会无法镜像 (mirror)。

   b. 手动镜像（mirror）缺少的镜像或重试镜像 (mirror) 的过程。

3. 检查与生成的集群资源相关的错误。即使无法镜像（mirror）某些镜像，oc-mirror v2 仍会为已成功镜像（mirror）的镜像生成集群资源，如 **IDMS.yaml** 和 **ITMS.yaml** 文件。

   a. 在输出目录中检查生成的文件。

   b. 如果特定镜像缺少了这些文件，请确保在镜像（mirror）过程中没有发生关键错误。

按照以下步骤，您可以更好地诊断问题并确保更顺畅地进行镜像。

5.8. ENCLAVE 支持的好处

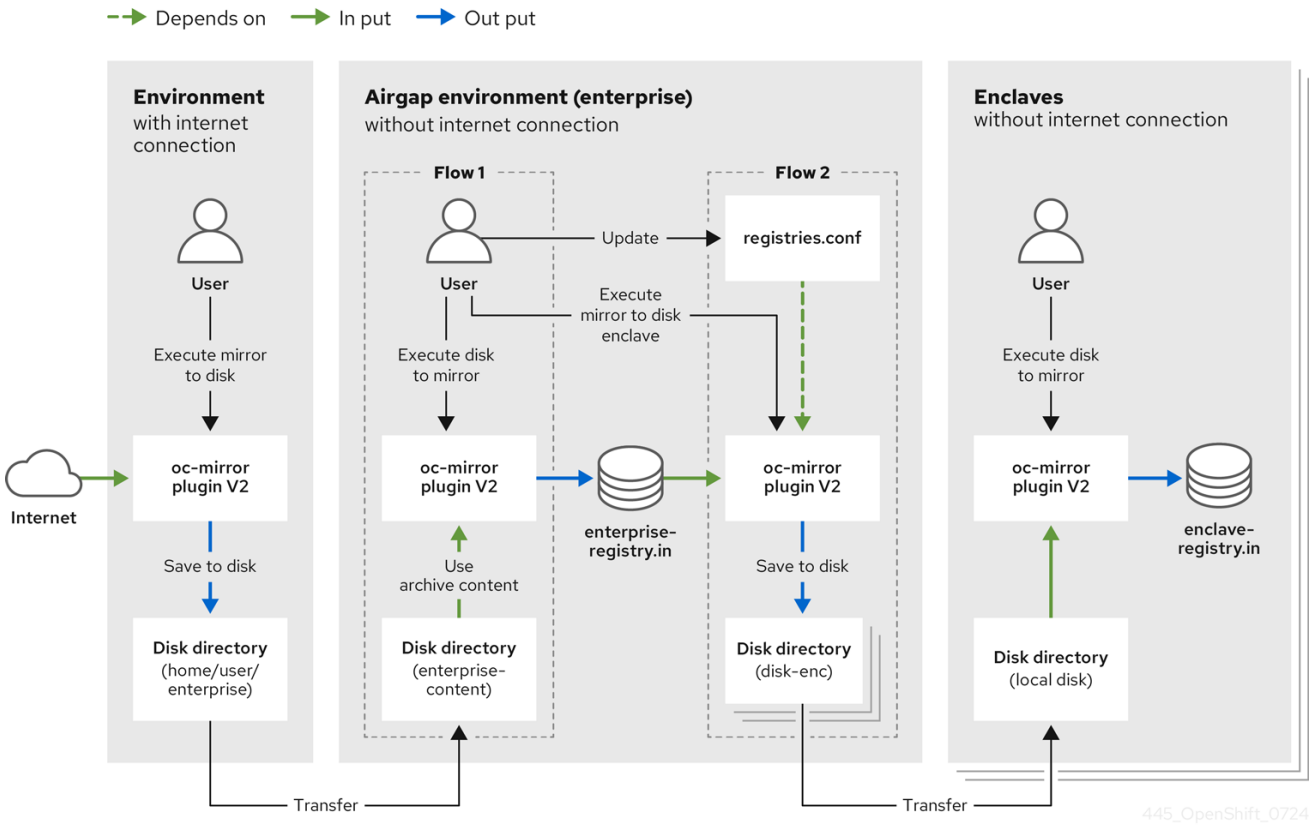
enclave 支持限制内部访问网络的特定部分。与一个 demilitarized zone (DMZ)网络不同，它允许通过防火墙界限和出站流量访问，enclaves 不跨防火墙界限。

新的 enclave 支持功能适用于在至少一个中间断开连接的网络中保护的多个 enclave 时需要镜像 (mirror)。

enclave 支持有以下优点：

- 您可以镜像多个 enclaves 的内容，并将其集中到一个内部 registry 中。因为有些客户希望在镜像的内容上运行安全检查，所以此设置他们可以一次性运行这些检查。然后，在镜像到下游 enclaves 之前，内容会被检查。
- 您可以直接从集中式内部 registry 中镜像内容，而无需为每个 enclave 重启镜像过程。
- 您可以最小化网络阶段的数据传输，以确保 Blob 或镜像仅从一个阶段传输到另一个阶段。

5.8.1. enclave 镜像工作流



上一个镜像概述了在不同环境中使用 oc-mirror 插件的流程，包括在有互联网连接和没有互联网连接的环境中进行。

带有互联网连接的环境 ：

1. 用户执行 oc-mirror 插件 v2，将在线 registry 的内容镜像到本地磁盘目录。
2. 镜像内容保存到磁盘中，用于在离线环境中使用。

断开连接的企业环境（没有互联网） ：

- 流 1：
  - 用户运行 oc-mirror 插件 v2，从磁盘目录中加载镜像的内容（来自在线的环境）加载到 **enterprise-registry.in** registry。
- 流 2：
  a. 更新 **registry.conf** 文件后，用户执行 oc-mirror 插件 v2 以将来自 **enterprise-registry.in** registry 的内容镜像到 enclave 环境中。
  b. 内容被保存到一个磁盘目录，用于转移到 enclave。

Enclave 环境 (没有互联网):

- 用户运行 oc-mirror 插件 v2，将目录从磁盘目录加载到 **enclave-registry.in** registry 中。

图显示了跨这些环境的数据流，侧重于使用 oc-mirror 来处理断开连接的环境和没有互联网连接的环境。

5.8.2. 镜像到 enclave

当镜像到 enclave 时，您必须首先将必要的镜像从一个或多个 enclaves 传送到企业中央 registry 中。

中央 registry 位于安全网络中，特别是断开连接的环境，且不会直接链接到公共互联网。但是，用户必须在一个可访问公共互联网的环境中执行 **oc mirror**。

流程

1. 在断开连接的环境中运行 oc-mirror 插件 v2 之前，请创建一个 **registry.conf** 文件。该文件的 TOML 格式如下所述：

注意

建议将文件存储在 **$HOME/.config/containers/registries.conf** 或 **/etc/containers/registries.conf** 下。

registry.conf 示例

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

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

2. 生成镜像存档。

   a. 要将所有 OpenShift Container Platform 内容收集到磁盘的一个归档中（**<file_path>/enterprise-content**），请运行以下命令：

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

isc.yaml 示例

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

生成存档后，它将传送到断开连接的环境中。传输机制不是 oc-mirror 插件 v2 的一部分。企业网络管理员需要负责制定传输策略。

在某些情况下，传输是手动完成的，例如在一个没有网络连接的环境中，将磁盘从一个物理机器中拔出，并插入到另外一个系统。在其他情况下，使用安全文件传输协议 (SFTP) 或其他协议。

3. 完成存档传输后，您可以再次执行 oc-mirror 插件 v2，以便将相关存档内容镜像到 registry (在示例中是 **entrerpise_registry.in**)，如下例所示：

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

其中：

- **--from** 指向包含存档的文件夹。它以 **file://** 开头。
- **docker://** 是镜像 (mirror) 的目的地，这是最后的参数。因为它是一个 docker registry。
- **-c** (**--config**)是一个强制参数。它可让 oc-mirror 插件 v2 最终仅将存档的子部分镜像到 registry。一个存档可能包含几个 OpenShift Container Platform 版本，但断开连接的环境或 enclave 可能只镜像几个。

4. 准备 **imageSetConfig** YAML 文件，该文件描述了要镜像到 enclave 的内容：

示例 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
```

您必须在可访问断开连接的 registry 的机器上运行 oc-mirror 插件 v2。在上例中，可以访问断开连接的环境 **enterprise-registry.in**。

5. 更新图形 URL
如果您使用 **graph:true**，oc-mirror 插件 v2 会尝试访问 **cincinnati** API 端点。由于此环境断开连接，因此请务必导出环境变量 **UPDATE_URL_OVERRIDE**，以引用 OpenShift Update Service (OSUS)的 URL：

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

有关在 OpenShift 集群上设置 OSUS 的更多信息，请参阅"使用 OpenShift Update Service 在断开连接的环境中更新集群"。

注意

当在 OpenShift Container Platform 延长更新支持 (EUS) 版本间进行更新时，还需要包括在当前版本和目标版本间的中间次版本的镜像。oc-mirror 插件 v2 可能并不总是自动检测此要求，因此请检查 Red Hat OpenShift Container Platform Update Graph 页以确认包括了所需的中间版本。

使用 Update Graph 页查找应用程序推荐的中间次版本，在使用 oc-mirror 插件 v2 时，在 **ImageSetConfiguration** 文件中包括这些版本。

6. 为 enclave 从企业 registry 中生成镜像存档。
   要为 **enclave1** 准备存档，用户使用针对那个 enclave 专用的 **imageSetConfiguration** 在企业断开连接的环境中执行 oc-mirror 插件 v2。这样可确保仅镜像 enclave 需要的镜像：

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

   此操作将所有 OpenShift Container Platform 内容收集到存档中，并在磁盘上生成存档。

7. 生成存档后，它将传送到 **enclave1** 网络。传输机制不是 oc-mirror 插件 v2 的责任。

8. 将内容镜像到 enclave registry
   完成存档传输后，用户可以再次执行 oc-mirror 插件 v2，以便将相关存档内容镜像到 registry。

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

   enclave1 中的 OpenShift Container Platform 集群的管理员现在可以安装或升级该集群。

5.9. OC-MIRROR 插件 V2 支持代理设置

oc-mirror 插件 v2 可以在代理配置的环境中运行。该插件可以使用系统代理设置来检索 OpenShift Container Platform、Operator 目录和 **additionalImages** registry 的镜像。

其他资源

- 使用 OpenShift Update Service 在断开连接的环境中更新集群
- 解决分发 registry 中的存储清理问题

5.10. 在 OC-MIRROR 插件 V2 中镜像和验证镜像签名

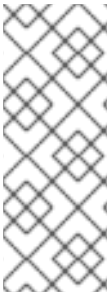
从 OpenShift Container Platform 4.19 开始，oc-mirror 插件 v2 支持镜像和验证容器镜像的基于 cosign 标签的签名。

5.10.1. 为 oc-mirror 插件 v2 启用签名镜像

默认情况下禁用签名镜像。您可以通过为 **oc mirror** 命令设置 **--remove-signatures=false** 标志来为所有镜像启用签名镜像。

启用后，oc-mirror 插件 v2 为以下镜像镜像基于 **Sigstore** 标签的签名：

- OpenShift Container Platform 发行镜像
- Operator 镜像
- 其他镜像
- Helm chart



注意

如果没有提供配置文件，oc-mirror 插件 v2 会在使用 **--remove-signatures=false** 标志时默认启用所有镜像的签名镜像。

要指定自定义配置目录，请使用 **--registries.d** 标志。

如需了解更多详细信息，请参阅 **containers-registries.d (5)** manual。

流程

1. 如果要为所有镜像启用签名镜像，请运行以下命令：

```
$ oc mirror --remove-signatures=false
```

2. 如果要为特定元素（如传输协议、registry、命名空间或镜像）启用或禁用签名镜像，请使用以下步骤：

   a. 在 **$HOME/.config/containers/registries.d/** 或 **/etc/containers/registries.d/** 目录中创建 YAML 文件。

   b. 指定 **use-sigstore-attachments** 参数，并在您要控制的特定元素下将其设置为 **true** 或 **false**，如下例所示：

   示例：禁用 quay.io registry 的签名镜像

```
# ...
docker:
  quay.io:
    use-sigstore-attachments: false
# ...
```

   示例：为所有 registry 启用签名镜像

```
# ...
default-docker:
  use-sigstore-attachments: true
# ...
```

5.10.2. 为 oc-mirror 插件 v2 启用签名验证

从 OpenShift Container Platform 4.19 开始，oc-mirror 插件 v2 支持签名验证，这默认禁用。启用后，插件会验证容器镜像是否与其签名匹配，确保它们没有被更改，并来自可信源。如果检测到签名不匹配，镜像工作流将失败。

流程

1. 如果要为所有镜像启用签名验证，请运行以下命令：

```
$ oc mirror --secure-policy=true
```

2. 如果要为特定元素（如传输协议、registry、命名空间或镜像）启用或禁用签名验证，请按照以下步骤操作：

a. 在 **$HOME/.config/containers/** 或 **/etc/containers/** 目录中创建 **policy.json** 文件。

注意

如果您的策略配置文件位于默认目录之外，您可以在 **oc mirror** 命令中使用 **--policy** 标志来指定其路径。

如需更多信息，请参阅 **containers-policy.json (5)**。

b. 使用适当的策略配置，为所需的范围（如 registry 或镜像）定义验证规则。您可以通过在每个元素中指定所需规则来设置验证要求。

示例：仅为特定镜像启用验证，并拒绝所有其他镜像

```
{
  "default": [{"type": "reject"}],
  "transports": {
    "docker": {
      "hostname:5000/myns/sigstore-signed-image": [
        {
          "type": "sigstoreSigned",
          "keyPath": "/path/to/sigstore-pubkey.pub",
          "signedIdentity": {"type": "matchRepository"}
        }
      ]
    }
  }
}
```

5.11. 过滤在 OPERATOR 目录中如何工作

oc-mirror 插件 v2 通过处理 **imageSetConfig** 中的信息来选择用于镜像的捆绑包列表。

当 oc-mirror 插件 v2 选择用于镜像的捆绑包时，它不会推断 Group Version Kind (GVK) 或捆绑包依赖项，从镜像集中省略它们。相反，它严格遵循用户指令。您必须明确指定任何所需的依赖软件包及其版本。

表 5.2. 使用下表查看在不同场景中包括哪些捆绑包版本

ImageSetConfig operator 过滤	预期的捆绑包版本

场景 1
 	
`mirror:`
`operators:`
`- catalog:`
`registry.redhat.io/redhat/redhat-operator-index:v4.10`
	对于目录中每个软件包，一个捆绑包，对应于该软件包的每个频道的头版本。

ImageSetConfig operator 过滤	预期的捆绑包版本
场景 2 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true	指定目录的所有频道的所有捆绑包。
场景 3 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 packages: - name: compliance-operator	一个捆绑包，对应于该软件包的每个频道的头（head）版本。
场景 4 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator	指定软件包的所有频道捆绑包。
场景 5 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator minVersion: 5.6.0	所有频道中的所有捆绑包（从 **minVersion** ）到该软件包的频道头。

ImageSetConfig operator 过滤	预期的捆绑包版本
场景 6 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator maxVersion: 6.0.0	所有频道中的所有捆绑包都低于该软件包的 **maxVersion**。
场景 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	所有频道中的所有捆绑包，在那个软件包的 **minVersion** 和 **maxVersion** 之间。频道头不包含，即使过滤中包含多个频道。
场景 8 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable	该软件包所选频道的头捆绑包。如果过滤的频道不是默认频道，则必须使用 **defaultChannel** 字段。

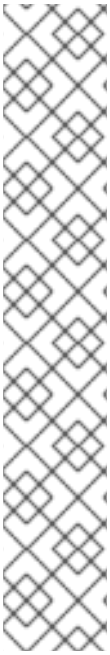
ImageSetConfig operator 过滤	预期的捆绑包版本
场景 9 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.10 full: true - packages: - name: elasticsearch-operator channels: - name: 'stable-v0'	指定的软件包和频道的所有捆绑包。如果过滤的频道不是默认频道，则应使用 **defaultChannel**。
场景 10 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable - name: stable-5.5	该软件包每个所选频道的头捆绑包。
场景 11 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0	在该软件包所选频道中，所有版本都以 **minVersion** 开头，直到频道头。如果过滤的频道不是默认频道，则必须使用 **defaultChannel** 字段。

ImageSetConfig operator 过滤	预期的捆绑包版本
场景 12 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable maxVersion: 6.0.0	在该软件包所选频道中，所有版本都最高为 **maxVersion**。频道头不包含，即使过滤中包含多个频道。如果此过滤导致带有多个头的频道，您可能会看到错误。如果过滤的频道不是默认频道，则必须使用 **defaultChannel** 字段。
场景 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	在该软件包的所选频道中，**minVersion** 和 **maxVersion** 之间的所有版本。频道头不包含，即使过滤中包含多个频道。如果此过滤导致带有多个头的频道，您可能会看到错误。如果过滤的频道不是默认频道，则必须使用 **defaultChannel** 字段。
场景 14 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	不要使用这个场景。通过频道过滤，以及不允许带有 **minVersion** 或 **maxVersion** 的软件包。

ImageSetConfig operator 过滤	预期的捆绑包版本
场景 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	请勿使用此场景。您不能使用 **full:true** 和 **minVersion** 或 **maxVersion** 进行过滤。
场景 16 mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.15 full: true packages: - name: compliance-operator channels - name: stable minVersion: 5.6.0 maxVersion: 6.0.0	请勿使用此场景。您不能使用 **full:true** 和 **minVersion** 或 **maxVersion** 进行过滤。

5.12. OC-MIRROR 插件 V2 的 IMAGESET 配置参数

oc-mirror 插件 v2 需要一个镜像设置配置文件，该文件定义哪些镜像要镜像(mirror)。下表列出了 **ImageSetConfiguration** 资源的可用参数。



注意

- 在为镜像选择捆绑包时，oc-mirror 插件 v2 不会自动检测 group/version/kind (GVK) 和捆绑包依赖项。您必须在 **ImageSetConfiguration** 文件中明确指定所需的 Operator、其频道和 Operator 版本。如需更多信息，请参阅 "opm CLI 参考"。
- 使用 **minVersion** 和 **maxVersion** 属性过滤特定 Operator 版本范围可能会导致多个频道头错误。错误信息将显示有**多个频道头**。这是因为在应用过滤器时，Operator 的更新图会被截断。
- OLM 要求每个 Operator 频道都包含组成一个更新图表的版本，它只有一个端点，即 Operator 的最新版本。在应用图形的过滤器范围时，可以进入两个或多个独立图形或具有多个端点的图形。
- 要避免这个错误，请不要过滤 Operator 的最新版本。如果您仍然遇到错误，具体取决于 Operator，则必须增加 **maxVersion** 属性，或者 **minVersion** 属性必须减少。因为每个 Operator 图都可以不同，所以您可能需要调整这些值，直到错误解决为止。

表 5.3. ImageSetConfiguration 参数

参数	描述	值
apiVersion	**ImageSetConfiguration** 内容的 API 版本。	字符串示例：**mirror.openshift.io/v2alpha1**
archiveSize	镜像集中的每个存档文件的最大大小（以 GiB 为单位）。	整数示例：**4**

参数	描述	值
kubeVirtContainer	当设置为 **true** 时，包含来自 HyperShift KubeVirt CoreOS 容器的镜像。	**ImageSetConfiguration** 文件的布尔值示例： apiVersion: mirror.openshift.io/v2alpha1 kind: ImageSetConfiguration mirror: platform: channels: - name: stable-4.16 minVersion: 4.16.0 maxVersion: 4.16.0 kubeVirtContainer: true
mirror	镜像集的配置。	对象
mirror.additionalImages	镜像集的额外镜像配置。	对象数组 Example: additionalImages: - name: registry.redhat.io/ubi8/ubi:latest
mirror.additionalImages.name	要 mirror 的镜像的标签或摘要。	字符串示例：**registry.redhat.io/ubi8/ubi:latest**
mirror.blockedImages	用于阻止镜像(tag)或摘要(SHA)的镜像列表。	字符串数组示例：**docker.io/library/alpine**

参数	描述	值
mirror.helm	镜像集的 helm 配置。oc-mirror 插件不支持使用手动修改的 **values.yaml** 文件的 helm chart。	对象
mirror.helm.local	要镜像的本地 helm chart。	对象数组。例如： local: - name: podinfo path: /test/podinfo-5.0.0.tar.gz
mirror.helm.local.name	要镜像的本地 helm chart 的名称。	字符串.例如：**podinfo**。
mirror.helm.local.path	到镜像的本地 helm chart 的路径。	字符串.例如：**/test/podinfo-5.0.0.tar.gz**。
mirror.helm.repositories	从其中镜像的的远程 helm 软件仓库。	对象数组。例如： repositories: - name: podinfo url: https://example.github.io/podinfo charts: - name: podinfo version: 5.0.0 imagePaths: - "{.spec.template.spec.custom[*].image}"
mirror.helm.repositories.name	从其中镜像(mirror)的 helm 存储库的名称。	字符串.例如：**podinfo**。
mirror.helm.repositories.url	从其中镜像(mirror)的 helm 存储库的 URL。	字符串.例如：**https://example.github.io/podinfo**。

参数	描述	值
mirror.helm.repositories.charts	要镜像的远程 helm chart。	对象数组。
mirror.helm.repositories.charts.name	要镜像的 helm chart 的名称。	字符串.例如：**podinfo**。
mirror.helm.repositories.charts.imagePaths	helm chart 中容器镜像的自定义路径。 +  **注意** **oc-mirror** 通过搜索已知的路径来检测并镜像 helm chart 中的容器镜像。您还可以使用此字段指定自定义路径。	字符串数组。例如：**"-{.spec.template.spec.custom[*].image}"**。
mirror.operators	镜像集的 Operator 配置。	对象数组 Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.19 packages: - name: elasticsearch-operator minVersion: '2.4.0'
mirror.operators.catalog	包括在镜像集中的 Operator 目录。	字符串示例：**registry.redhat.io/redhat/redhat-operator-index:v4.15**
mirror.operators.full	为 **true** 时，下载完整的目录、Operator 软件包或 Operator 频道。	布尔值，默认值为 **false**。

参数	描述	值
mirror.operators.packages	Operator 软件包配置。	对象数组 Example: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:4.19 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	镜像集中要包含的 Operator 软件包名称。	字符串 例：**elasticsearch-operator**
mirror.operators.packages.channels	Operator 软件包频道配置	对象
mirror.operators.packages.channels.name	Operator 频道名称（软件包中唯一）要包括在镜像集中。	字符串示例：**fast** 或 **stable-v4.15**
mirror.operators.packages.channels.maxVersion	Operator 镜像的最高版本，在其中存在所有频道。	字符串 例：**5.2.3-31**
mirror.operators.packages.channels.minVersion	Operator 的最低版本，用于镜像存在的所有频道	字符串 例：**5.2.3-31**
mirror.operators.packages.maxVersion	Operator 最高版本，可跨所有存在的频道进行镜像。	字符串 例：**5.2.3-31**
mirror.operators.packages.minVersion	Operator 的最低版本，用于镜像存在的所有频道。	字符串 例：**5.2.3-31**
mirror.operators.targetCatalog	将引用的目录镜像为的替代名称和可选命名空间层次结构	字符串示例：**my-namespace/my-operator-catalog**

参数	描述	值

mirror.operators.targetCatalogSourceTemplate	用于完成 oc-mirror 插件 v2 生成的 catalogSource 自定义资源的模板的路径。	字符串示例：**/tmp/catalog-source_template.yaml** 模板文件示例：
		apiVersion: operators.coreos.com/v1alpha1
		kind: CatalogSource
		metadata:
		  name: discarded
		  namespace: openshift-marketplace
		spec:
		  image: discarded
		  sourceType: grpc
		
		updateStrategy:
		
		registryPoll:
		  interval: 30m0s

mirror.operators.targetTag	附加到 **targetName** 或 **targetCatalog** 的替代标签。	字符串示例：**v1**
mirror.platform	镜像集的平台配置。	对象

参数	描述	值
mirror.platform.architectures	要镜像的平台发行版本有效负载的架构。	字符串数组示例： architectures: - amd64 - arm64 - multi - ppc64le - s390x 默认值为 **amd64**。值 **multi** 确保镜像支持所有可用架构，无需指定单个架构。
mirror.platform.channels	镜像集的平台频道配置。	对象数组示例： channels: - name: stable-4.12 - name: stable-4.19
mirror.platform.channels.full	为 **true** 时，将 **minVersion** 设置为频道中的第一个发行版本，将 **maxVersion** 设置为该频道的最后一个发行版本。	布尔值，默认为 **false**
mirror.platform.channels.name	版本频道的名称	字符串示例：**stable-4.15**
mirror.platform.channels.minVersion	要镜像引用的平台的最低版本。	字符串示例：**4.12.6**
mirror.platform.channels.maxVersion	要镜像引用的平台的最高版本。	字符串示例：**4.15.1**
mirror.platform.channels.shortestPath	切换最短的路径镜像或完整范围镜像。	布尔值，默认为 **false**
mirror.platform.channels.type	要镜像的平台的类型	字符串示例：**ocp** 或 **okd**。默认为 **ocp**。
mirror.platform.graph	指明是否将 OSUS 图表添加到镜像集中，然后发布到镜像。	布尔值，默认为 **false**

参数	描述	值
mirror.operators.packages.defaultChannel	在从过滤中排除默认频道时，必须定义。	对象数组。例如： mirror: operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19 packages: - name: rhods-operator defaultChannel: fast channels: - name: fast

5.12.1. DeleteImageSetConfiguration 参数

要将删除镜像与 oc-mirror 插件 v2 搭配使用，您必须使用 **DeleteImageSetConfiguration.yaml** 配置文件来定义要从镜像 registry 中删除哪些镜像。下表列出了 **DeleteImageSetConfiguration** 资源的可用参数。

表 5.4. DeleteImageSetConfiguration 参数

参数	描述	值
apiVersion	**DeleteImageSetConfiguration** 内容的 API 版本。	字符串示例：**mirror.openshift.io/v2alpha1**
delete	镜像集要删除的配置。	对象
delete.additionalImages	删除镜像集的额外镜像配置。	对象数组示例： additionalImages: - name: registry.redhat.io/ubi8/ubi:latest

参数	描述	值
delete.additionalImages.name	要删除的镜像的标签或摘要。	字符串 例：**registry.redhat.io/ubi8/ubi:latest**
delete.operators	删除镜像集的 Operator 配置。	对象数组示例： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:{product-version} packages: - name: elasticsearch-operator minVersion: '2.4.0'
delete.operators.catalog	要在 delete 镜像集中包含的 Operator 目录。	字符串示例：**registry.redhat.io/redhat/redhat-operator-index:v4.15**
delete.operators.full	如果为 true，则删除完整目录、Operator 软件包或 Operator 频道。	布尔值，默认为 **false**

参数	描述	值
delete.operators.packages	Operator 软件包配置	对象数组示例： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index: {product-version} packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
delete.operators.packages.name	要在 delete 镜像集中包含的 Operator 软件包名称。	字符串 例：**elasticsearch-operator**
delete.operators.packages.channels	Operator 软件包频道配置	对象
delete.operators.packages.channels.name	Operator 频道名称（在软件包中是唯一的）包括在 delete 镜像集中。	字符串示例： **fast** 或 **stable-v4.15**
delete.operators.packages.channels.maxVersion	在所选频道中删除的 Operator 的最高版本。	字符串 例：**5.2.3-31**
delete.operators.packages.channels.minVersion	在存在的选择中删除 Operator 的最低版本。	字符串 例：**5.2.3-31**
delete.operators.packages.maxVersion	在存在的所有频道中删除 Operator 的最高版本。	字符串 例：**5.2.3-31**
delete.operators.packages.minVersion	在存在的所有频道中删除 Operator 的最低版本。	字符串 例：**5.2.3-31**
delete.platform	镜像集的平台配置	对象

参数	描述	值
delete.platform.architectures	要删除的平台发行版本有效负载的架构。	字符串数组示例： architectures: - amd64 - arm64 - multi - ppc64le - s390x 默认值为 **amd64**
delete.platform.channels	镜像集的平台频道配置。	对象数组 Example: channels: - name: stable-4.12 - name: stable-4.19
delete.platform.channels.full	为 **true** 时，将 **minVersion** 设置为频道中的第一个发行版本，将 **maxVersion** 设置为该频道的最后一个发行版本。	布尔值，默认为 **false**
delete.platform.channels.name	版本频道的名称	字符串示例：**stable-4.15**
delete.platform.channels.minVersion	要删除的引用平台的最低版本。	字符串示例：**4.12.6**
delete.platform.channels.maxVersion	要删除的引用平台的最高版本。	字符串示例：**4.15.1**
delete.platform.channels.shortestPath	在删除最短路径并删除完整范围之间切换。	布尔值，默认为 **false**
delete.platform.channels.type	要删除的平台的类型	字符串示例：**ocp** 或 **okd**。默认为 **ocp**
delete.platform.graph	确定是否在镜像 registry 上也删除 OSUS 图形。	布尔值，默认为 **false**

其他资源

- opm CLI 参考

5.13. OC-MIRROR 插件 V2 的命令参考

下表描述了 **oc mirror** 子命令和 oc-mirror 插件 v2 的标志：

表 5.5. oc-mirror 插件 v2 的子命令和标志

子命令	描述
帮助	显示有关任何子命令的帮助
version	输出 oc-mirror 版本
delete	删除远程 registry 和本地缓存中的镜像。

表 5.6. oc mirror 标记

标记	描述
--authfile	显示身份验证文件的字符串路径。默认为 **${XDG_RUNTIME_DIR}/containers/auth.json**。
-c, **--config <string>**	指定镜像设置配置文件的路径。
--cache-dir <string>	使用此标志指定 oc-mirror 插件存储镜像 Blob 和清单在镜像操作过程中使用的持久缓存的目录。oc-mirror 插件使用 **disk-to-mirror** 和 **mirror-to-disk** 工作流中的缓存，但不会在 **mirror-to-mirror** 工作流中使用缓存。该插件使用缓存来执行增量镜像，并避免重新镜像未更改的镜像，这会节省时间并减少网络带宽使用量。默认缓存目录为 **$HOME**。如需更多信息，请参阅"关于 --cache-dir 和 --workspace 标记"。
--dest-tls-verify	在访问容器 registry 或守护进程时，需要 HTTPS 并验证证书。默认值为 **true**。
--dry-run	在不 mirror 镜像的情况下打印操作。
--from <string>	指定通过执行 oc-mirror 插件 v2 来加载目标 registry 生成的镜像设置存档的路径。
-h,**--help**	显示帮助
--image-timeout duration	mirror 镜像超时。默认值为 10m0s。有效的时间单位是 **ns**, **us** or **µs**, **ms**, **s**, **m**, 和 **h**。
--log-level <string>	显示字符串日志级别。支持的值包括 info、debug、trace、error。默认值为 **info**。
-p, **--port**	确定 oc-mirror 插件 v2 本地存储实例使用的 HTTP 端口。默认值为 **55000**。

标记	描述
--parallel-images <unit>	指定并行镜像的镜像数量。默认值为 **4**。
--parallel-layers <unit>	指定并行镜像的镜像层数量。默认值为 **5**。
--max-nested-paths <int>	指定限制嵌套路径的目标 registry 的最大嵌套路径数。默认值为 **0**。
--secure-policy	默认值为 **false**。如果您设置了非默认值，命令会启用签名验证，这是签名验证的安全策略。
--since	包含自指定日期以来的所有新内容（格式：**yyyy-mm-dd**）。如果没有提供，自以前的镜像(mirror)以来的新内容会被镜像。
--src-tls-verify	在访问容器 registry 或守护进程时，需要 HTTPS 并验证证书。
--strict-archive	默认值为 **false**。如果您设置了值，命令会生成比 **imageSetConfig** 自定义资源 (CR) 中设置的 **archiveSize** 的存档。如果正在归档的文件超过 **archiveSize** (GB)，则镜像已存在。
-v, --version	显示 oc-mirror 插件 v2 的版本。
--workspace	您可以使用 **--workspace** 标志指定一个目录，其中 oc-mirror 插件存储它在镜像操作过程中创建的工作文件，如 **ImageDigestMirrorSet** 和 **ImageTagMirrorSet** 清单。使用此目录将生成的配置应用到集群并重复镜像操作。如需更多信息，请参阅"关于 --cache-dir 和 --workspace 标记"。
--retry-delay duration	2 次重试之间延迟。默认值为 **1s**。
--retry-times <int>	要重试的次数。默认值为 **2**。
--rootless-storage-path <string>	覆盖默认容器无根存储路径（通常为 **etc/containers/storage.conf**）。
--remove-signatures	不从源镜像复制签名。
--registries.d	指定包含 registry 配置文件的目录。
--secure-policy=true	为所有镜像启用签名验证。
--policy	指定签名验证策略文件的路径。

5.13.1. 删除镜像的命令参考

下表描述了 **oc mirror** 子命令和用于删除镜像的标志：

表 5.7. 用于删除镜像的子命令和标志

子命令	描述
--authfile <string>	身份验证文件的路径。默认值为 **${XDG_RUNTIME_DIR}/containers/auth.json**。
--cache-dir <string>	使用此标志指定 oc-mirror 插件存储镜像 Blob 和清单在镜像操作过程中使用的持久缓存的目录。oc-mirror 插件使用 **disk-to-mirror** 和 **mirror-to-disk** 工作流中的缓存，但不会在 **mirror-to-mirror** 工作流中使用缓存。该插件使用缓存来执行增量镜像，并避免重新镜像未更改的镜像，这会节省时间并减少网络带宽使用量。默认缓存目录为 **$HOME**。如需更多信息，请参阅"关于 --cache-dir 和 --workspace 标记"。
-c <string>, --config <string>	删除 imageset 配置文件的路径。
--delete-id <string>	用于区分由删除功能创建的文件的版本。
--delete-v1-images	在迁移过程中（与 **--generate** 一起），以针对之前使用 oc-mirror 插件 v1 镜像进行镜像的目标。
--delete-yaml-file <string>	如果设置，使用生成的或更新的 yaml 文件来删除内容。
--dest-tls-verify	与容器 registry 或守护进程对话时，需要 HTTPS 并验证证书。默认值为 **true**。
--force-cache-delete	用于强制删除本地缓存清单和 blob。
--generate	用于为从本地缓存和远程 registry 中删除时使用的清单和 blob 列表生成 delete yaml。
-h,--help	显示帮助。
--log-level <string>	日志级别，是 **info**, **debug**, **trace**, 和 **error** 之一。默认值为 **info**。
--parallel-images <unit>	指明并行删除的镜像数量。默认值为 **4**。
--parallel-layers <unit>	指明并行镜像的镜像层数量。默认值为 **5**。
-p <unit>, --port <unit>	oc-mirror 的本地存储实例使用的 HTTP 端口。默认值为 **55000**。
--retry-delay	2 次重试之间的持续时间延迟。默认值为 **1s**。
--retry-times <int>	要重试的次数。默认值为 **2**。
--src-tls-verify	与容器 registry 或守护进程对话时，需要 HTTPS 并验证证书。默认值为 **true**。

子命令	描述
--workspace <string>	您可以使用 **--workspace** 标志指定一个目录，其中 oc-mirror 插件存储它在镜像操作过程中创建的工作文件，如 **ImageDigestMirrorSet** 和 **ImageTagMirrorSet** 清单。使用此目录将生成的配置应用到集群并重复镜像操作。如需更多信息，请参阅"关于 --cache-dir 和 --workspace 标记"。

5.13.2. 关于 --cache-dir 和 --workspace 标记

您可以使用 **--cache-dir** 标志指定一个目录，其中 oc-mirror 插件存储了镜像 Blob 和清单在镜像操作期间使用的持久缓存。

oc-mirror 插件使用 **mirror-to-disk** 和 **disk-to-mirror** 工作流中的缓存，但不会在 **mirror-to-mirror** 工作流中使用缓存。该插件使用缓存来执行增量镜像，并避免重新镜像未更改的镜像，这会节省时间并减少网络带宽使用量。

缓存目录仅包含最近一次成功的镜像操作的数据。如果您删除或损坏缓存目录，oc-mirror 插件会再次拉取镜像 Blob 和清单，这可以强制完全重新镜像并增加网络使用量。

您可以使用 **--workspace** 标志指定一个目录，其中 oc-mirror 插件存储它在镜像操作过程中创建的工作文件，如 **ImageDigestMirrorSet** 和 **ImageTagMirrorSet** 清单。您还可以使用工作区目录执行以下操作：

- 为发行版本和 Operator 镜像存储解压缩的元数据。
- 生成 tar 存档，以便在 **disk-to-mirror** 工作流中使用。
- 将生成的配置应用到集群。
- 重复或恢复以前的镜像操作。

如果您删除或修改 workspace 目录，则将来的镜像操作可能会失败，或者集群可能会使用不一致的镜像源。

警告

删除或修改缓存或工作区目录的内容可能会导致以下问题：

- 镜像操作失败或不完整。
- 增量镜像数据丢失。
- 完全重新镜像的要求以及增加的网络开销。

除非您完全了解相关的影响，否则不要修改、重新定位或删除这些目录。在成功的镜像操作后，您必须定期备份缓存目录。不需要备份工作区目录，因为每个镜像周期都会重新生成其内容。

请考虑以下最佳实践，以便可以更好地管理缓存和工作区目录：

- 使用持久性存储 ：在可靠和备份存储上放置缓存和工作区目录。
- 在成功操作后备份：常规备份缓存目录，特别是在完成镜像周期后。
- 恢复 ：如果出现数据丢失，请从备份中恢复缓存和工作区目录，以便在不执行完整重新镜像的情况下恢复镜像操作。
- 独立的环境：为不同的环境使用专用目录以防止冲突。

在运行 oc-mirror 命令时，使用以下示例指定 cache 和 workspace 目录：

```
$ oc mirror --config=imageset-config.yaml \
   file://local_mirror \
   --workspace /mnt/mirror-data/workspace \
   --cache-dir /mnt/mirror-data/cache
   --v2
```

镜像操作完成后，您的目录结构如下：

```
/mnt/mirror-data/
├──── cache/
│     ├──── manifests/
│     ├──── metadata.db
│     └──── previous-mirror-state.json
└──── workspace/
    ├──── imageset-config-state.yaml
    ├──── manifests/
    └──── icsp/
```

每次成功镜像操作后，您必须备份 **/mnt/mirror-data/cache** 目录。

其他资源

- 配置集群以使用 oc-mirror 生成的资源

5.14. 后续步骤

使用 oc-mirror 插件 v2 将镜像 mirror 到断开连接的环境中后，您可以执行以下操作：

- 在断开连接的环境中安装集群
- 在断开连接的环境中使用 Operator Lifecycle Manager。
- 使用 OpenShift Update Service 在断开连接的环境中更新集群

第 6 章 从 OC-MIRROR 插件 V1 迁移到 V2

oc-mirror v2 插件对镜像 mirror 工作流进行了重大更改。本指南为迁移提供了步骤说明，同时确保与 oc-mirror 插件 v2 兼容。

重要

您必须通过修改 API 版本并删除已弃用的字段来手动更新配置。如需更多信息，请参阅"从 oc-mirror 插件 v1 改为 v2"。

6.1. 从 OC-MIRROR 插件 V1 改为 V2

在从 oc-mirror 插件 v1 迁移到 v2 之前，请参阅 oc-mirror 插件 v1 和 v2 之间的以下区别：

- 显式版本选择：在使用 **oc-mirror** 时，用户必须明确指定 **--v2**。如果没有指定版本，则默认执行 v1。此行为预期会在以后的版本中改变，其中 **--v2** 是默认设置。
- 更新了命令：镜像工作流的命令已更改为与 oc-mirror 插件 v2 的新工作流保持一致。
  - 对于 mirror-to-disk，请运行以下命令：

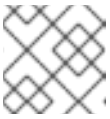
    ```
    $ oc-mirror --config isc.yaml file://<directory_name> --v2
    ```

  - 对于 disk-to-mirror，运行以下命令：

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

  - 对于 mirror-to-mirror，运行以下命令：

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

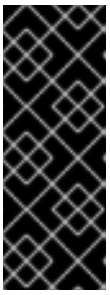
    注意

    mirror-to-mirror 操作现在需要 **--workspace**。

- API 版本更新： **ImageSetConfiguration** API 版本从 **v1alpha2** (v1)改为 **v2alpha1** (v2)。迁移前，您必须手动更新配置文件。
- 配置更改：
  - **storageConfig** 必须在 oc-mirror 插件 v2 中删除。
  - 现在，通过工作目录或本地缓存自动处理增量镜像。
- 在结果目录中更改：要应用到断开连接的集群的所有自定义资源都会在迁移后的 **<workspace_path>/working-dir/cluster-resources** 目录中生成。
  - oc-mirror 插件 v2 中的输出不会存储在与 oc-mirror 插件 v1 相同的位置。
  - 您必须检查工作目录中的 **cluster-resources** 文件夹以查找以下资源：

- **ImageDigestMirrorSet** (IDMS)
- **ImageTagMirrorSet** (ITMS)
- **CatalogSource**
- **ClusterCatalog**
- **UpdateService**

- 工作区和目录命名：遵循新的 oc-mirror v2 约定，以防止在不同版本间进行转换时潜在的数据不一致。
  - oc-mirror 插件 v1 **oc-mirror-workspace** 目录不再需要。
  - 为 oc-mirror 插件 v2 使用新目录以避免冲突。
- 将 **ImageContentSourcePolicy** (ICSP) 资源替换为 IDMS/ITMS ：



重要

删除所有 **ImageContentSourcePolicy** (ICSP)资源可能会删除与 oc-mirror 无关的配置。

为了避免意外删除，请在删除前识别 oc-mirror 生成的 ICSP 资源。如果您不确定，请与集群管理员进行检查。如需更多信息，请参阅"使用 oc-mirror 插件 v2 为断开连接的安装镜像镜像"。

  - 在 oc-mirror 插件 v2 中，ICSP 资源由 **ImageDigestMirrorSet** (IDMS)和 **ImageTagMirrorSet** (ITMS)资源替代。

6.2. 迁移到 OC-MIRROR 插件 V2

要从 oc-mirror 插件 v1 迁移到 v2，您必须手动更新 **ImageSetConfiguration** 文件，修改镜像命令并清理 v1 工件。按照以下步骤完成迁移。

流程

1. 修改 API 版本并删除 **ImageSetConfiguration** 中的已弃用字段。

带有 oc-mirror 插件 v1 配置的 ImageSetConfiguration 文件示例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
  platform:
    channels:
      - name: stable-4.17
    graph: true
  helm:
    repositories:
      - name: sbo
        url: https://redhat-developer.github.io/service-binding-operator-helm-chart/
  additionalImages:
```

```
    - name: registry.redhat.io/ubi8/ubi:latest
    - name: quay.io/openshifttest/hello-openshift@sha256:example_hash
  operators:
    - catalog: oci:///test/redhat-operator-index
      packages:
        - name: aws-load-balancer-operator
storageConfig:  # REMOVE this field in v2
  local:
    path: /var/lib/oc-mirror
```

带有 oc-mirror 插件 v2 配置的 ImageSetConfiguration 文件示例

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

注意

从 oc-mirror 插件 v1 迁移到 v2 时，您必须为 **additionalImages** 下列出的所有镜像使用显式 registry 主机名。否则，镜像会被镜像到不正确的目标路径。

2. 运行以下命令，检查工作目录中的 IDMS、ITMS、**CatalogSource** 和 **ClusterCatalog** 资源中的 **cluster-resources** 目录：

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

3. 迁移完成后，验证镜像的镜像和目录是否可用：

- 确保镜像过程中没有发生错误或警告。
- 确保没有生成错误文件(**working-dir/logs/mirroring_errors_YYYYMMdd_HHmmss.txt**)。

4. 使用以下命令验证镜像镜像和目录是否可用：

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

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

如需更多信息，请参阅"使用 oc-mirror 插件 v2 为断开连接的安装镜像镜像"。

5. 可选：使用 oc-mirror 插件 v1 删除镜像镜像：

   a. 使用 oc-mirror 插件 v1 镜像镜像。

   b. 将 **ImageSetConfiguration** 文件中的 API 版本从 **v1alpha2** (v1)更新至 **v2alpha1** (v2)，然后运行以下命令：

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

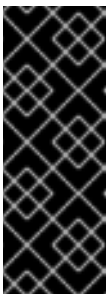
   

   注意

   storageConfig 不是 **ImageSetConfiguration** 和 **DeleteImageSetConfiguration** 文件中的有效字段。当升级到 oc-mirror 插件 v2 时删除此字段。

   c. 运行以下命令生成删除清单并删除 v1 镜像：

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

   重要

   oc-mirror 插件 v2 不会自动修剪目标 registry，这与 oc-mirror 插件 v1 不同。要清理不再需要的镜像，请使用 v2 中的删除功能及 **--delete-v1-images** 命令标志。

   删除所有使用 oc-mirror 插件 v1 的镜像后，您不再需要使用此标志。如果您需要删除使用 oc-mirror 插件 v2 镜像，请不要设置 **--delete-v1-images**。

   有关删除镜像的更多信息，请参阅"从断开连接的环境中删除镜像"。

   d. 运行以下命令，根据生成的清单删除镜像：

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

6.3. 其他资源

- 在部分断开连接的环境中镜像设置的镜像
- 镜像在完全断开连接的环境中设置的镜像
- 有关配置更改的详情，请参阅从 oc-mirror 插件 v1 改为 v2 。
- 有关删除镜像的更多信息，请参阅从断开连接的环境中删除镜像。

第 7 章 使用 OC-MIRROR 插件为断开连接的安装镜像镜像

可以在没有直接的互联网连接的受限网络中运行集群，方法是使用在一个私有 registry 中的 mirror OpenShift Container Platform 容器镜像安装集群。集群运行时必须始终运行此 registry。如需更多信息，请参阅先决条件部分。

您可以使用 oc-mirror OpenShift CLI (**oc**)插件在完全或部分断开连接的环境中将镜像镜像到镜像 registry。您必须从具有互联网连接的系统运行 oc-mirror，以便从官方红帽 registry 中下载所需的镜像。

重要

oc-mirror v1 插件已弃用。要防止在以后的版本中失败，请指定 **--v1** 标志来继续使用 v1 插件，或迁移到支持的 v2 插件，使用 **--v2** 标志。过渡到 oc-mirror v2 插件，以便继续获得支持和改进。

7.1. 关于 OC-MIRROR 插件

您可以使用 oc-mirror OpenShift CLI(**oc**)插件，使用单个工具将所有所需的 OpenShift Container Platform 内容和其他镜像(mirror)镜像到您的镜像 registry。它提供以下功能：

- 提供镜像 OpenShift Container Platform 发行版本、Operator、helm chart 和其他镜像的集中方法。
- 维护 OpenShift Container Platform 和 Operator 的更新路径。
- 使用声明的镜像设置配置文件来仅包含集群所需的 OpenShift Container Platform 发行版本、Operator 和镜像。
- 执行增量镜像，从而减少将来镜像集的大小。
- 从上一执行以来，从镜像集配置中排除的目标镜像 registry 中修剪镜像的镜像。
- （可选）为 OpenShift Update Service (OSUS) 使用生成支持工件。

使用 oc-mirror 插件时，您可以在镜像设置配置文件中指定要镜像的内容。在这个 YAML 文件中，您可以将配置微调为仅包含集群需要的 OpenShift Container Platform 发行版本和 Operator。这可减少您下载和传输所需的数据量。oc-mirror 插件也可以镜像任意 helm chart 和附加容器镜像，以帮助用户将其工作负载无缝同步到镜像 registry 中。

第一次运行 oc-mirror 插件时，它会使用所需内容填充您的镜像 registry，以执行断开连接的集群安装或更新。要让断开连接的集群继续接受更新，您必须更新镜像 registry。要更新您的镜像 registry，请使用与第一次运行相同的配置运行 oc-mirror 插件。oc-mirror 插件引用存储后端的元数据，并只下载上次运行该工具后所发布的元数据。这为 OpenShift Container Platform 和 Operator 提供了更新路径，并根据需要执行依赖项解析。

7.1.1. 高级别工作流

下列步骤概述了如何使用 oc-mirror 插件将镜像镜像到镜像 registry 的高级别工作流：

1. 创建镜像设置配置文件。
2. 使用以下方法之一将镜像设置为目标镜像 registry：
    - 将镜像直接设置为目标镜像 registry。

- 镜像集合镜像到磁盘，将镜像设置为目标环境，然后将镜像上传到目标镜像 registry。

3. 配置集群以使用 oc-mirror 插件生成的资源。
4. 根据需要重复这些步骤以更新目标镜像 registry。

重要

当使用 oc-mirror CLI 插件填充镜像 registry 时，必须使用 oc-mirror 插件对目标镜像 registry 进行进一步的更新。

7.2. OC-MIRROR 插件兼容性和支持

oc-mirror 插件支持为 OpenShift Container Platform 版本 4.12 及之后的版本的镜像 OpenShift Container Platform 有效负载镜像和 Operator 目录。

注意

在 **aarch64**, **ppc64le**, 和 **s390x** 架构中，oc-mirror 插件只支持 OpenShift Container Platform 版本 4.14 及更新的版本。

使用 oc-mirror 插件的最新版本，无论您需要镜像的 OpenShift Container Platform 版本是什么。

其他资源

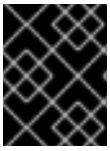
- 有关更新 oc-mirror 的详情，请参阅查看镜像拉取源。

7.3. 关于镜像 REGISTRY

您必须可以访问互联网来获取所需的容器镜像。使用一个替代的 registry 意味着，将 mirror registry 放在一个可访问您的网络以及互联网的 mirror 主机上。

您可以将 OpenShift Container Platform 安装和后续的产品更新到支持 Docker v2-2 （如 Red Hat Quay）的容器 mirror registry。如果您无法访问大型容器 registry，可以使用 *mirror registry for Red Hat OpenShift*，这是 OpenShift 中包含的小型容器 registry。

无论您所选 registry 是什么，都会将互联网上红帽托管站点的内容镜像到隔离的镜像 registry 相同。镜像内容后，您要将每个集群配置为从镜像 registry 中检索此内容。



重要

OpenShift 镜像 registry 不能用作目标 registry，因为它不支持没有标签的推送，在镜像过程中需要这个推送。

如果选择的容器 registry 不是 *mirror registry for Red Hat OpenShift*，则需要集群中置备的每台机器都可以访问它。如果 registry 无法访问，安装、更新或常规操作（如工作负载重新定位）可能会失败。因此，您必须以高度可用的方式运行镜像 registry，镜像 registry 至少必须与 OpenShift Container Platform 集群的生产环境可用性相匹配。

使用 OpenShift Container Platform 镜像填充镜像 registry 时，可以遵循以下两种情况。如果您的主机可以同时访问互联网和您的镜像 registry，而不能访问您的集群节点，您可以直接从该机器中镜像该内容。这个过程被称为 *连接的镜像(mirror)*。如果没有这样的主机，则必须将该镜像文件镜像到文件系统中，然后将该主机或者可移动介质放入受限环境中。这个过程被称为 *断开连接的镜像*。

对于已镜像的 registry，若要查看拉取镜像的来源，您必须查看 **Trying 以访问** CRI-O 日志中的日志条目。查看镜像拉取源的其他方法（如在节点上使用 **crictl images** 命令）显示非镜像镜像名称，即使镜像是从镜像位置拉取的。

注意

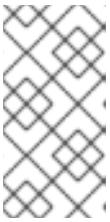
红帽没有针对 OpenShift Container Platform 测试第三方 registry。

其他资源

- 查看镜像拉取源。

7.4. 先决条件

- 您必须在托管 OpenShift Container Platform 集群的位置（如 Red Hat Quay）中有一个支持 Docker v2-2 的容器镜像 registry。



注意

如果使用 Red Hat Quay，则必须在 oc-mirror 插件中使用 3.6 或更高版本的版本。如果您有 Red Hat Quay 权利，请参阅有关部署 Red Hat Quay 以了解概念验证的文档，或使用 Red Hat Quay Operator。如果您需要额外的帮助来选择并安装 registry，请联络您的销售代表或红帽支持。

如果您还没有容器镜像 registry，OpenShift Container Platform 可以为订阅者提供一个 mirror registry for Red Hat OpenShift。*mirror registry for Red Hat OpenShift* 包含在您的订阅中，它是一个小型容器 registry，可用于在断开连接的安装中镜像 OpenShift Container Platform 所需的容器镜像。

7.5. 准备您的镜像主机

在使用 oc-mirror 插件镜像(mirror)前，您必须安装插件并创建容器镜像 registry 凭据文件，以允许从红帽镜像到您的镜像。

7.5.1. 安装 oc-mirror OpenShift CLI 插件

安装 oc-mirror OpenShift CLI 插件以在断开连接的环境中管理镜像集。

先决条件

- 已安装 OpenShift CLI(**oc**)。如果您在完全断开连接的环境中镜像镜像集，请确保以下内容：
  - 您已在可访问互联网的主机上安装了 oc-mirror 插件。
  - 在断开连接的环境中的主机可以访问目标镜像 registry。
- 您已在使用 oc-mirror 的操作系统中，将 **umask** 参数设置为 **0022**。
- 您已为您要使用的 RHEL 版本安装了正确的二进制文件。

流程

1. 下载 oc-mirror CLI 插件：

a. 进入到 Red Hat Hybrid Cloud Console 的 Downloads 页。

b. 在 **OpenShift disconnected installation tools** 部分中，从下拉菜单中选择 **OpenShift Client (oc) mirror plugin** 的**OS type** 和 **Architecture type**。

c. 点 **Download** 保存文件。

2. 运行以下命令来提取存档：

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

3. 如有必要，运行以下命令来将插件文件更新为可执行：

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

注意

不要重命名 **oc-mirror** 文件。

4. 运行以下命令，将文件放在 **PATH** 中（如 **/usr/local/bin** ）来安装 oc-mirror CLI 插件：

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

验证

- 运行以下命令验证 oc-mirror 插件 v1 是否已成功安装：

```
$ oc mirror help
```

其他资源

- 安装和使用 CLI 插件

7.5.2. 配置允许对容器镜像进行镜像的凭证

创建容器镜像 registry 凭证文件，可让您将镜像从红帽 mirror 到您的镜像。在安装主机上完成以下步骤。

警告

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

先决条件

- 您已将镜像 registry 配置为在断开连接的环境中使用。
- 您在镜像 registry 中标识了镜像仓库的位置，以将容器镜像镜像(mirror)到这个位置。

- 您置备了一个镜像 registry 帐户，允许将镜像上传到该镜像仓库。
- 对镜像 registry 有写权限。

流程

1. 从 Red Hat OpenShift Cluster Manager 下载 **registry.redhat.io** pull secret。
2. 运行以下命令，以 JSON 格式生成 pull secret 副本：

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

指定到存储 pull secret 的目录的路径，以及您创建的 JSON 文件的名称。

pull secret 示例

```
{
  "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. 将文件保存为 **~/.docker/config.json** 或 **$XDG_RUNTIME_DIR/containers/auth.json**：
   a. 如果 **.docker** 或 **$XDG_RUNTIME_DIR/containers** 目录不存在，请输入以下命令来创建：

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

   其中 **<directory_name>** 是 **~/.docker** 或 **$XDG_RUNTIME_DIR/containers**。

   b. 输入以下命令将 pull secret 复制到适当的目录中：

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

   <directory_name> 是 **~/.docker** 或 **$XDG_RUNTIME_DIR/containers**；**<auth_file>** 是 **config.json** 或 **auth.json**

1. 运行以下命令，为您的镜像 registry 生成 base64 编码的用户名和密码或令牌：

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

通过 **<user_name>** 和 **<password>** 指定 registry 的用户名和密码。

输出示例

```
BGVtbYk3ZHAtqXs=
```

2. 编辑 JSON 文件并添加描述 registry 的部分：

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

- 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如：**registry.example.com** 或 **registry.example.com:8443**。
- 对于 **<credentials>** 值，请指定 mirror registry 的 base64 编码用户名和密码。

修改的 pull secret 示例

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

7.6. 创建镜像设置配置

在使用 oc-mirror 插件镜像集之前，必须先创建镜像设置配置文件。此镜像设置配置文件定义哪些 OpenShift Container Platform 发行版本、Operator 和其他镜像要镜像，以及 oc-mirror 插件的其他配置设置。

您必须在镜像设置配置文件中指定存储后端。此存储后端可以是本地目录或支持 Docker v2-2 的 registry。oc-mirror 插件在创建镜像的过程中将元数据存储在这个存储后端中。

重要

不要删除或修改 oc-mirror 插件生成的元数据。每次针对同一镜像 registry 运行 oc-mirror 插件时，都必须使用相同的存储后端。

先决条件

- 您已创建了容器镜像 registry 凭证文件。具体步骤请参阅"配置允许镜像镜像的凭证"。

流程

1. 使用 **oc mirror init** 命令为镜像设置配置创建模板，并将其保存到名为 **imageset-config.yaml** 的文件中：

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

(1) 指定存储后端的位置，如 **example.com/mirror/oc-mirror-metadata**。

2. 编辑该文件并根据需要调整设置：

```
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) 添加 **archiveSize** 以设置镜像集合中的每个文件的最大大小（以 GiB 为单位）。

(2) 设置后端位置，以将镜像设置元数据保存到。此位置可以是 registry 或本地目录。必须指定 **storageConfig** 值。

(3) 设置存储后端的 registry URL。

4 将频道设置为从中检索 OpenShift Container Platform 镜像。

5 添加 **graph: true** 以构建并推送 graph-data 镜像推送到镜像 registry。创建 OpenShift Update Service (OSUS) 需要 graph-data 镜像。**graph: true** 字段还会生成 **UpdateService** 自定义资源清单。**oc** 命令行界面 (CLI) 可以使用 **UpdateService** 自定义资源清单来创建 OSUS。如需更多信息，请参阅*关于 OpenShift Update Service*。

6 将 Operator 目录设置为从中检索 OpenShift Container Platform 镜像。

7 仅指定要包含在镜像集中的某些 Operator 软件包。删除此字段以检索目录中的所有软件包。

8 仅指定要包含在镜像集中的 Operator 软件包的某些频道。即使您没有使用该频道中的捆绑包，还必须始终包含 Operator 软件包的默认频道。您可以运行以下命令来找到默认频道：**oc mirror list operators --catalog=<catalog_name> --package=<package_name>**。

9 指定要在镜像集中包含的任何其他镜像。

注意

graph: true 字段还会镜像 **ubi-micro** 镜像，以及其他镜像的镜像。

当升级 OpenShift Container Platform 延长更新支持 (EUS) 版本时，在当前和目标版本之间可能需要一个中间版本。例如，如果当前版本是 **4.14**，目标版本为 **4.16**，您可能需要在使用 oc-mirror 插件 v1 时在 **ImageSetConfiguration** 中包含版本，如 **4.15.8**。

oc-mirror 插件 v1 可能并不总是自动检测，因此请检查 Cincinnati 图形网页以确认任何所需的中间版本并手动添加到您的配置中。

有关各种镜像用例，请参阅"镜像设置配置参数"和"镜像设置配置示例"。

3. 保存更新的文件。
在镜像内容时，**oc mirror** 命令需要此镜像设置配置文件。

其他资源

- 镜像设置配置参数
- 镜像设置配置示例
- 在断开连接的环境中使用 OpenShift Update Service

7.7. 将镜像集镜像(MIRROR)到镜像 REGISTRY

您可以使用 oc-mirror CLI 插件在 部分断开连接的环境中或完全断开连接的环境中将镜像镜像到镜像 registry。

这些步骤假定您已设置了镜像 registry。

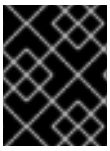
7.7.1. 在部分断开连接的环境中镜像设置的镜像

在部分断开连接的环境中，您可以直接镜像到目标镜像 registry 的镜像。

7.7.1.1. 镜像(mirror)到镜像(mirror)的镜像

您可以使用 oc-mirror 插件将镜像直接设置为在镜像设置过程中可访问的目标镜像 registry。

您必须在镜像设置配置文件中指定存储后端。这个存储后端可以是本地目录或 Docker v2 registry。oc-mirror 插件在创建镜像的过程中将元数据存储在这个存储后端中。



重要

不要删除或修改 oc-mirror 插件生成的元数据。每次针对同一镜像 registry 运行 oc-mirror 插件时，都必须使用相同的存储后端。

先决条件

- 您可以通过访问互联网来获取所需的容器镜像。
- 已安装 OpenShift CLI（**oc**）。
- 已安装 oc-mirror CLI 插件。
- 已创建镜像设置配置文件。

流程

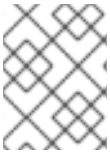
- 运行 **oc mirror** 命令将指定镜像集配置中的镜像镜像到指定的 registry:

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

1. 指定您创建的镜像设置配置文件。例如，**imageset-config.yaml**。
2. 指定要镜像设置文件的 registry。registry 必须以 **docker://** 开头。如果为镜像 registry 指定顶层命名空间，则必须在后续执行时使用此命名空间。

验证

1. 进入生成的 **oc-mirror-workspace/** 目录。
2. 导航到结果目录，例如，**results-1639608409/**。
3. 验证 **ImageContentSourcePolicy** 和 **CatalogSource** 资源是否存在 YAML 文件。



注意

ImageContentSourcePolicy YAML 文件的 **repositoryDigestMirrors** 部分在安装过程中用于 **install-config.yaml** 文件。

后续步骤

- 转换 **ImageContentSourcePolicy** YAML 内容，以用于手动配置 CRI-O。

- 如果需要，从 mirror 中将镜像同步到磁盘，以在断开连接的环境或离线环境中使用。
- 配置集群以使用 oc-mirror 生成的资源。

故障排除

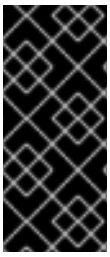
- 无法检索源镜像。

7.7.2. 镜像在完全断开连接的环境中设置的镜像

要镜像在完全断开连接的环境中设置的镜像，您必须首先将镜像集镜像到磁盘， 然后将磁盘上的镜像集文件镜像到一个镜像。

7.7.2.1. 从镜像镜像到磁盘

您可以使用 oc-mirror 插件生成镜像集，并将内容保存到磁盘。然后，生成的镜像集可以转移到断开连接的环境中，并镜像到目标 registry。



重要

根据镜像设置配置文件中指定的配置，使用 oc-mirror 的镜像可能会将几百 GB 数据下载到磁盘。

您填充镜像 registry 时初始镜像集下载通常是最大镜像。因为您只下载自上次运行命令以来更改的镜像，所以再次运行 oc-mirror 插件时，所生成的镜像集通常比较小。

您必须在镜像设置配置文件中指定存储后端。这个存储后端可以是本地目录或 docker v2 registry。oc-mirror 插件在创建镜像的过程中将元数据存储在这个存储后端中。

重要

不要删除或修改 oc-mirror 插件生成的元数据。每次针对同一镜像 registry 运行 oc-mirror 插件时，都必须使用相同的存储后端。

先决条件

- 您可以访问互联网来获取所需的容器镜像。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 oc-mirror CLI 插件。
- 您已创建了镜像设置配置文件。

流程

- 运行 **oc mirror** 命令将指定镜像集配置镜像到磁盘：

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

1 传递创建的镜像设置配置文件。此流程假设它名为 **imageset-config.yaml**。

2 指定要输出镜像集文件的目标目录。目标目录路径必须以 **file://** 开头。

验证

1. 进入您的输出目录：

```
$ cd <path_to_output_directory>
```

2. 验证是否创建了镜像设置 **.tar** 文件：

```
$ ls
```

输出示例

```
mirror_seq1_000000.tar
```

后续步骤

- 将镜像集 .tar 文件移动到断开连接的环境中。

故障排除

- 无法检索源镜像。

7.7.2.2. 从磁盘镜像到镜像

您可以使用 oc-mirror 插件将生成的镜像集的内容镜像到目标镜像 registry。

先决条件

- 您已在断开连接的环境中安装了 OpenShift CLI(**oc**)。
- 您已在断开连接的环境中安装了 oc-mirror CLI 插件。
- 已使用 **oc mirror** 命令生成镜像集文件。
- 您已将镜像集文件传送到断开连接的环境中。

流程

- 运行 **oc mirror** 命令，以处理磁盘上镜像集文件，并将内容镜像到目标镜像 registry：

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

1 传递镜像集 .tar 文件以进行镜像，在本例中名为 **mirror_seq1_000000.tar**。如果在镜像设置配置文件中指定了 **archiveSize** 值，则镜像集可能会划分为多个 .tar 文件。在这种情况下，您可以传递一个包含镜像设置 .tar 文件的目录。

2 指定要镜像设置文件的 registry。registry 必须以 **docker://** 开头。如果为镜像 registry 指定顶层命名空间，则必须在后续执行时使用此命名空间。

此命令使用镜像集更新镜像 registry，并生成 **ImageContentSourcePolicy** 和 **CatalogSource** 资源。

验证

1. 进入生成的 **oc-mirror-workspace/** 目录。
2. 导航到结果目录，例如，**results-1639608409/**。
3. 验证 **ImageContentSourcePolicy** 和 **CatalogSource** 资源是否存在 YAML 文件。

后续步骤

- 配置集群以使用 oc-mirror 生成的资源。

故障排除

- 无法检索源镜像。

7.8. 配置集群以使用 OC-MIRROR 生成的资源

将镜像设置为镜像 registry 后，您必须将生成的 **ImageContentSourcePolicy**、**CatalogSource** 和发行版本镜像签名资源应用到集群。

ImageContentSourcePolicy 资源将镜像 registry 与源 registry 关联，并将在线 registry 中的镜像拉取请求重定向到镜像 registry。Operator Lifecycle Manager (OLM) Classic 使用 **CatalogSource** 资源来检索有关镜像 registry 中可用 Operator 的信息。发行镜像签名用于验证镜像的发行镜像。

注意

OLM v1 使用 **ClusterCatalog** 资源来检索有关镜像 registry 中可用集群扩展的信息。

oc-mirror 插件 v1 不会自动生成 **ClusterCatalog** 资源；您必须手动创建它们。有关创建并应用 **ClusterCatalog** 资源的更多信息，请参阅"扩展"中的"将目录添加到集群"。

先决条件

- 您已将镜像设置为断开连接的环境中的 registry 镜像。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift CLI (**oc**)。
2. 运行以下命令，将结果目录中的 YAML 文件应用到集群：

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

3. 如果镜像(mirror)镜像，请运行以下命令将发行版本镜像签名应用到集群：

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

注意

如果要镜像 Operator 而不是集群，则不需要运行 **$ oc apply -f ./oc-mirror-workspace/results-1639608409/release-signatures/**。运行该命令将返回错误，因为没有要应用的发行版本镜像签名。

验证

1. 运行以下命令验证 **ImageContentSourcePolicy** 资源是否已成功安装：

```
$ oc get imagecontentsourcepolicy
```

2. 运行以下命令验证 **CatalogSource** 资源是否已成功安装：

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

其他资源

- 在"扩展"中将目录添加到集群中

7.9. 更新您的镜像 REGISTRY 内容

您可以通过更新镜像设置配置文件并将镜像集镜像到镜像 registry 来更新镜像 registry 内容。下次运行 oc-mirror 插件时，会生成一个镜像集，该镜像集仅包含之前执行以来的新和更新镜像。

在更新镜像 registry 时，您必须考虑以下注意事项：

- 如果镜像不再包含在生成和镜像的最新镜像集中，则会从目标镜像 registry 中修剪镜像。因此，请确保为以下关键组件相同的组合更新镜像，以便只创建并镜像不同的镜像集：
  - 镜像设置配置
  - 目标 registry
  - 存储配置
- 当要 mirror 或 mirror 到 mirror 工作流时，可以修剪镜像。
- 生成的镜像集必须按顺序推送到目标镜像 registry。您可以从生成的镜像设置归档文件的文件名中获取序列号。
- 不要删除或修改 oc-mirror 插件生成的元数据镜像。
- 如果您在初始镜像集创建过程中为镜像 registry 指定顶层命名空间，则每次针对同一镜像 registry 运行 oc-mirror 插件时都必须使用此命名空间。

有关更新镜像 registry 内容的工作流的更多信息，请参阅"高级别工作流"部分。

7.9.1. 镜像 registry 更新示例

本节论述了将镜像 registry 从磁盘更新到镜像的用例。

以前用于镜像的 ImageSetConfiguration 文件示例

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

7.9.1.1. 通过修剪现有镜像来镜像特定的 OpenShift Container Platform 版本

更新了 ImageSetConfiguration 文件

```
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) 使用 **stable-4.13** 修剪 **stable-4.12** 的所有镜像。

7.9.1.2. 通过修剪现有镜像升级到 Operator 的最新版本

更新了 ImageSetConfiguration 文件

```
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 使用相同的频道而没有指定版本会修剪现有镜像，并使用最新版本的镜像进行更新。

7.9.1.3. 通过修剪现有的 Operator 来镜像新 Operator

更新了 ImageSetConfiguration 文件

```
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 使用 **new_operator_name** 替换 **rhacs-operator** 修剪 Red Hat Advanced Cluster Security for Kubernetes Operator。

7.9.1.4. 修剪所有 OpenShift Container Platform 镜像

更新了 ImageSetConfiguration 文件

```
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:
```

-

其他资源

- 镜像设置配置示例
- 在部分断开连接的环境中镜像设置的镜像
- 镜像在完全断开连接的环境中设置的镜像
- 配置集群以使用 oc-mirror 生成的资源

7.10. 执行空运行

您可以使用 oc-mirror 来执行空运行，而无需实际镜像(mirror)。这可让您查看要镜像的镜像列表，以及从镜像 registry 修剪的所有镜像。使用空运行（dry run）还允许您在早期版本中捕获与镜像集配置相关的任何错误，或使用生成的镜像列表以及其他工具来执行镜像操作。

先决条件

- 您可以访问互联网来获取所需的容器镜像。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 oc-mirror CLI 插件。
- 您已创建了镜像设置配置文件。

流程

1. 使用 **--dry-run** 标志运行 **oc mirror** 命令来执行空运行：

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

1. 传递创建的镜像设置配置文件。此流程假设它名为 **imageset-config.yaml**。
2. 指定镜像 registry。在使用 **--dry-run** 标志时，不会镜像这个 registry。
3. 使用 **--dry-run** 标志来生成空运行工件，而不是实际的镜像设置文件。

输出示例

```
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. 进入生成的工作区目录：

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

3. 查看生成的 **mapping.txt** 文件。
此文件包含将要镜像的所有镜像的列表。

4. 查看生成的 **prune-plan.json** 文件。
此文件包含在发布镜像集时从镜像 registry 中修剪的所有镜像的列表。

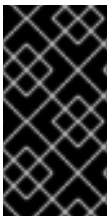
注意

只有在 oc-mirror 命令指向您的镜像 registry 且需要修剪的镜像时，才会生成 **prune-plan.json** 文件。

7.11. 包括本地 OCI OPERATOR 目录

虽然将 OpenShift Container Platform 发行版本、Operator 目录和其他额外的镜像从 registry mirror 到一个部分断开连接的集群中，但您还可以在一个本地磁盘中的基于文件的目录中包含 Operator 目录镜像。本地目录必须采用开放容器项目 (OCI) 格式。

本地目录及其内容会根据镜像设置配置文件中的过滤信息，mirror 到您的目标 mirror registry。

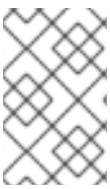


重要

在镜像本地 OCI 目录时，所有您要 mirror 的 OpenShift Container Platform 发行版本或其他需要和本地 OCI 格式目录一起 mirror 的镜像都必须从 registry 中拉取。

您无法在磁盘中一起 mirror OCI 目录和一个 oc-mirror 镜像集文件镜像。

使用 OCI 功能的一个用例是，您有一个 CI/CD 系统将 OCI 目录构建到磁盘上的位置，并需要将 OCI 目录以及 OpenShift Container Platform 发行版本一起 mirror 到您的 mirror 镜像 registry。



注意

如果您为 OpenShift Container Platform 4.12 的 oc-mirror 插件使用了预览预览的 OCI 本地目录功能，则无法再使用 oc-mirror 插件的 OCI 本地目录功能在本地复制目录，并将其转换为 OCI 格式作为 mirror 到一个完全断开连接的集群中的第一步。

先决条件

- 您可以访问互联网来获取所需的容器镜像。
- 已安装 OpenShift CLI(**oc**)。
- 已安装 oc-mirror CLI 插件。

流程

1. 创建镜像设置配置文件，并根据需要调整设置。
以下示例镜像设置配置在磁盘上 mirror 一个 OCI 目录，以及来自 **registry.redhat.io** 的 OpenShift Container Platform 发行版本和 UBI 镜像。

```
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. 设置后端位置，以将镜像设置元数据保存到。此位置可以是 registry 或本地目录。必须指定 **storageConfig** 值。
2. （可选）包括一个 OpenShift Container Platform 发行版本，以便从 **registry.redhat.io** mirror。
3. 指定磁盘上 OCI 目录位置的绝对路径。使用 OCI 功能时，路径必须以 **oci:**// 开头。
4. 另外，还可指定替代命名空间和名称来镜像目录。
5. （可选）指定要从 registry 中拉取的额外 Operator 目录。
6. （可选）指定要从 registry 中拉取的额外镜像。

注意

在 oc-mirror 插件 v2 中，您必须为 **additionalImages** 下列出的所有镜像使用显式 registry 主机名。否则，镜像会被镜像到不正确的目标路径。

2. 运行 **oc mirror** 命令将 OCI 目录 mirror 到目标 mirror registry：

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

1. 传递镜像设置配置文件。此流程假设它名为 **imageset-config.yaml**。

2 指定要将内容 mirror 到的 registry。registry 必须以 **docker://** 开头。如果为镜像 registry 指定顶层命名空间，则必须在后续执行时使用此命名空间。

另外，您可以指定其他标记来调整 OCI 功能的行为：

--oci-insecure-signature-policy

不要将签名推送到目标 mirror registry。

--oci-registries-config

指定 TOML 格式的 **registry.conf** 文件的路径。您可以使用它来从不同的 registry 中镜像，如用于测试的预生产位置，而无需更改镜像设置配置文件。这个标志只会影响本地 OCI 目录，而不会影响任何其他被镜像的内容。

registry.conf 文件示例

```
[[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
```

后续步骤

- 配置集群以使用 oc-mirror 生成的资源。

其他资源

- 配置集群以使用 oc-mirror 生成的资源

7.12. 镜像设置配置参数

oc-mirror 插件需要一个镜像设置配置文件，该文件定义哪些镜像要镜像(mirror)。下表列出了 **ImageSetConfiguration** 资源的可用参数。

表 7.1. ImageSetConfiguration 参数

参数	描述	值
apiVersion	**ImageSetConfiguration** 内容的 API 版本。	字符串.例如：**mirror.openshift.io/v1alpha2**。
archiveSize	镜像集中的每个存档文件的最大大小（以 GiB 为单位）。	整数.例如： **4**
mirror	镜像集的配置。	对象

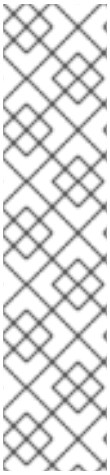
参数	描述	值
mirror.additionalImages	镜像集的额外镜像配置。	对象数组。例如： additionalImages: - name: registry.redhat.io/ubi8/ubi:latest
mirror.additionalImages.name	要 mirror 的镜像的标签或摘要。	字符串.例如：**registry.redhat.io/ubi8/ubi:latest**
mirror.blockedImages	阻止 mirror 的镜像的完整标签、摘要或模式。	字符串数组。例如：**docker.io/library/alpine**
mirror.helm	镜像集的 helm 配置。请注意，oc-mirror 插件只支持 helm chart，在呈现时不需要用户输入。	对象
mirror.helm.local	要镜像的本地 helm chart。	对象数组。例如： local: - name: podinfo path: /test/podinfo-5.0.0.tar.gz
mirror.helm.local.name	要镜像的本地 helm chart 的名称。	字符串.例如：**podinfo**。
mirror.helm.local.path	到镜像的本地 helm chart 的路径。	字符串.例如：**/test/podinfo-5.0.0.tar.gz**。

参数	描述	值
mirror.helm.repositories	从其中镜像的的远程 helm 软件仓库。	对象数组。例如： repositories: - name: podinfo url: https://example.github.io/podinfo charts: - name: podinfo version: 5.0.0
mirror.helm.repositories.name	从其中镜像(mirror)的 helm 存储库的名称。	字符串.例如：**podinfo**。
mirror.helm.repositories.url	从其中镜像(mirror)的 helm 存储库的 URL。	字符串.例如：**https://example.github.io/podinfo**。
mirror.helm.repositories.charts	要镜像的远程 helm chart。	对象数组。
mirror.helm.repositories.charts.name	要镜像的 helm chart 的名称。	字符串.例如：**podinfo**。
mirror.helm.repositories.charts.version	要镜像命名 helm chart 的版本。	字符串.例如：**5.0.0**。
mirror.operators	镜像集的 Operator 配置。	对象数组。例如： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19 packages: - name: elasticsearch-operator minVersion: '2.4.0'

参数	描述	值
mirror.operators.catalog	包括在镜像集中的 Operator 目录。	字符串.例如：**registry.redhat.io/redhat/redhat-operator-index:v4.19**。
mirror.operators.full	为 **true** 时，下载完整的目录、Operator 软件包或 Operator 频道。	布尔值.默认值为 **false**。
mirror.operators.packages	Operator 软件包配置。	对象数组。例如： operators: - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.19 packages: - name: elasticsearch-operator minVersion: '5.2.3-31'
mirror.operators.packages.name	镜像集中要包含的 Operator 软件包名称	字符串.例如：**elasticsearch-operator**。
mirror.operators.packages.channels	Operator 软件包频道配置。	对象
mirror.operators.packages.channels.name	Operator 频道名称（软件包中唯一）要包括在镜像集中。	字符串.例如：**fast** 或 **stable-v4.19**.
mirror.operators.packages.channels.maxVersion	Operator 镜像的最高版本，在其中存在所有频道。详情请查看以下备注。	字符串.例如：**5.2.3-31**
mirror.operators.packages.channels.minBundle	要包含的最小捆绑包的名称，以及频道头更新图中的所有捆绑包。仅在命名捆绑包没有语义版本元数据时设置此字段。	字符串.例如：**bundleName**
mirror.operators.packages.channels.minVersion	Operator 的最低版本，用于镜像存在的所有频道。详情请查看以下备注。	字符串.例如：**5.2.3-31**
mirror.operators.packages.maxVersion	Operator 最高版本，可跨所有存在的频道进行镜像。详情请查看以下备注。	字符串.例如：**5.2.3-31**。

参数	描述	值
mirror.operators.packages.minVersion	Operator 的最低版本，用于镜像存在的所有频道。详情请查看以下备注。	字符串.例如：**5.2.3-31**。
mirror.operators.skipDependencies	如果为 **true**，则不会包含捆绑包的依赖项。	布尔值.默认值为 **false**。
mirror.operators.targetCatalog	要镜像引用的目录的替代名称和可选命名空间层次结构。	字符串.例如：**my-namespace/my-operator-catalog**
mirror.operators.targetName	将引用的目录镜像为。 **targetName** 参数已弃用。改为使用 **targetCatalog** 参数。	字符串.例如： **my-operator-catalog**
mirror.operators.targetTag	附加到 **targetName** 或 **targetCatalog** 的替代标签。	字符串.例如： **v1**
mirror.platform	镜像集的平台配置。	对象
mirror.platform.architectures	要镜像的平台发行版本有效负载的架构。	字符串数组。例如： architectures: - amd64 - arm64 - multi - ppc64le - s390x 默认值为 **amd64**。值 **multi** 确保镜像支持所有可用架构，无需指定单个架构。
mirror.platform.channels	镜像集的平台频道配置。	对象数组。例如： channels: - name: stable-4.10 - name: stable-4.19
mirror.platform.channels.full	为 **true** 时，将 **minVersion** 设置为频道中的第一个发行版本，将 **maxVersion** 设置为该频道的最后一个发行版本。	布尔值.默认值为 **false**。

参数	描述	值
mirror.platform.channels.name	发行频道的名称。	字符串.例如：**stable-4.19**
mirror.platform.channels.minVersion	要镜像引用的平台的最低版本。	字符串.例如：**4.12.6**
mirror.platform.channels.maxVersion	要镜像引用的平台的最高版本。	字符串.例如：**4.19.1**
mirror.platform.channels.shortestPath	切换最短的路径镜像或完整范围镜像。	布尔值.默认值为 **false**。
mirror.platform.channels.type	要镜像的平台的类型。	字符串.例如： **ocp** 或 **okd**。默认为 **ocp**。
mirror.platform.graph	指明是否将 OSUS 图表添加到镜像集中，然后发布到镜像。	布尔值.默认值为 **false**。
storageConfig	镜像集的后端配置。	对象
storageConfig.local	镜像集的本地后端配置。	对象
storageConfig.local.path	包含镜像设置元数据的目录路径。	字符串.例如：**./path/to/dir/**。
storageConfig.registry	镜像集的 registry 后端配置。	对象
storageConfig.registry.imageURL	后端 registry URI。可以选择在 URI 中包含命名空间引用。	字符串.例如：**quay.io/myuser/imageset:metadata**。
storageConfig.registry.skipTLS	（可选）跳过引用的后端 registry 的 TLS 验证。	布尔值.默认值为 **false**。



注意

使用 **minVersion** 和 **maxVersion** 属性过滤特定 Operator 版本范围可能会导致多个频道头错误。错误信息将显示有**多个频道**头。这是因为在应用过滤器时，Operator 的更新图会被截断。

Operator Lifecycle Manager 要求每个 operator 频道都包含一个端点组成更新图表的版本，即 Operator 的最新版本。在应用图形的过滤器范围时，可以进入两个或多个独立图形或具有多个端点的图形。

要避免这个错误，请不要过滤 Operator 的最新版本。如果您仍然遇到错误，具体取决于 Operator，则必须增加 **maxVersion** 属性，或者 **minVersion** 属性必须减少。因为每个 Operator 图都可以不同，所以您可能需要调整这些值，直到错误解决为止。

7.13. 镜像设置配置示例

以下 **ImageSetConfiguration** 文件示例演示了各种镜像用例的配置。

7.13.1. 使用案例：包含最短的 OpenShift Container Platform 更新路径

以下 **ImageSetConfiguration** 文件使用本地存储后端，并包括所有 OpenShift Container Platform 版本，以及从最低 **4.11.37** 版本到最大 **4.12.15** 版本的更新路径。

ImageSetConfiguration 文件示例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  local:
    path: /home/user/metadata
mirror:
  platform:
    channels:
      - name: stable-4.12
        minVersion: 4.11.37
        maxVersion: 4.12.15
        shortestPath: true
```

7.13.2. 使用案例：包含对于多架构的版本的从最低到最新版本的所有 OpenShift Container Platform 版本

以下 **ImageSetConfiguration** 文件使用一个 registry 存储后端，并包括从最小 **4.13.4** 迁移到频道中最新版本的所有 OpenShift Container Platform 版本。对于每个使用此镜像集合配置的 oc-mirror，评估 **stable-4.13** 频道的最新发行版本，因此定期运行 oc-mirror 可确保您自动收到最新版本的 OpenShift Container Platform 镜像。

通过将 **platform.architectures** 的值设置为 **multi**，您可以确保支持多架构版本的镜像。

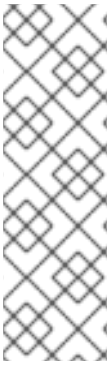
ImageSetConfiguration 文件示例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
```

```
  registry:
    imageURL: example.com/mirror/oc-mirror-metadata
    skipTLS: false
mirror:
  platform:
    architectures:
      - "multi"
    channels:
      - name: stable-4.13
        minVersion: 4.13.4
        maxVersion: 4.13.6
```

7.13.3. 使用案例：包含从最低到最新的 Operator 版本

以下 **ImageSetConfiguration** 文件使用本地存储后端，仅包含 **stable** 频道中从 4.0.1 及之后的版本开始的 Red Hat Advanced Cluster Security for Kubernetes Operator。



注意

当您指定了一个最小或最大版本范围时，可能不会接收该范围内的所有 Operator 版本。

默认情况下，oc-mirror 排除了 Operator Lifecycle Manager (OLM)规格中跳过或被较新的版本替换的任何版本。跳过的 Operator 版本可能会受到 CVE 或包含错误的影响。改为使用较新版本。有关跳过和替换版本的更多信息，请参阅使用 OLM 创建更新图表。

要接收指定范围内的所有 Operator 版本，您可以将 **mirror.operators.full** 字段设置为 **true**。

ImageSetConfiguration 文件示例

```
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
```

注意

要指定最大版本而不是最新的版本，请设置 **mirror.operators.packages.channels.maxVersion** 字段。

7.13.4. 使用案例：包含 Nutanix CSI Operator

以下 **ImageSetConfiguration** 文件使用本地存储后端，并包括 Nutanix CSI Operator、OpenShift Update Service (OSUS)图形镜像以及额外的 Red Hat Universal Base Image (UBI)。

ImageSetConfiguration 文件示例

```
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
```

7.13.5. 使用案例：包含默认 Operator 频道

以下 **ImageSetConfiguration** 文件包括 OpenShift Elasticsearch Operator 的 **stable-5.7** 和 **stable** 频道。即使只需要 **stable-5.7** 频道中的软件包，**stable** 频道也必须包含在 **ImageSetConfiguration** 文件中，因为它是 Operator 的默认频道。即使您没有使用该频道中的捆绑包，还必须始终包含 Operator 软件包的默认频道。

提示

您可以运行以下命令来找到默认频道：**oc mirror list operators --catalog=<catalog_name> --package=<package_name>**。

ImageSetConfiguration 文件示例

```
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
```

7.13.6. 使用案例：包含整个目录（所有版本）

以下 **ImageSetConfiguration** 文件将 **mirror.operators.full** 字段设置为 **true**，使其包含整个 Operator 目录的所有版本。

ImageSetConfiguration 文件示例

```
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
```

7.13.7. 使用案例：包含整个目录（仅限频道头）

以下 **ImageSetConfiguration** 文件包含整个 Operator 目录的频道头。

默认情况下，对于目录中的每个 Operator，oc-mirror 都包含来自默认频道的最新 Operator 版本（频道头）。如果要镜像所有 Operator 版本，而不仅仅是频道头，您必须将 **mirror.operators.full** 字段设置为 **true**。

本例还使用 **targetCatalog** 字段指定替代命名空间和名称来镜像目录。

ImageSetConfiguration 文件示例

```
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
```

7.13.8. 用例：包含任意镜像和 helm chart

以下 **ImageSetConfiguration** 文件使用 registry 存储后端，并包含 helm chart 和额外的 Red Hat Universal Base Image(UBI)。

ImageSetConfiguration 文件示例

```
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
```

7.13.9. 用例：包含 EUS 版本的升级路径

以下 **ImageSetConfiguration** 文件包含 **eus-<version>** 频道，其中 **maxVersion** 值至少要比 **minVersion** 的值高两个次版本。

例如，在这个 **ImageSetConfiguration** 文件中，**minVersion** 设置为 **4.12.28**，而 **eus-4.14** 频道的 **maxVersion** 为 **4.14.16**。

ImageSetConfiguration 文件示例

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

7.13.10. 用例：包含多架构 OpenShift Container Platform 镜像和用于多集群引擎 Operator 的目录

以下 **ImageSetConfiguration** 文件包含 Kubernetes Operator 的多集群引擎，以及从频道中的最低 **4.19.0** 版本开始的所有 OpenShift Container Platform 版本。

ImageSetConfiguration 文件示例

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL:
agent.agent.example.com:5000/openshift/release/metadata:latest/openshift/release/metadata:latest
mirror:
  platform:
    architectures:
      - "multi"
    channels:
    - name: stable-4.19
      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
```

7.14. OC-MIRROR 的命令参考

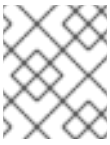
下表描述了 **oc mirror** 子命令和标志：

表 7.2. oc mirror 子命令

子命令	描述
completion	为指定的 shell 生成自动完成脚本。
describe	输出镜像集合的内容。
帮助	显示有关任何子命令的帮助。
init	输出初始镜像设置配置模板。
list	列出可用平台和 Operator 内容及其版本。
version	输出 oc-mirror 版本。

表 7.3. oc mirror 标记

标记	描述
-c, --config <string>	指定镜像设置配置文件的路径。
--continue-on-error	如果发生任何非镜像拉取相关的错误，请继续并尝试进行镜像(mirror)。

标记	描述
--dest-skip-tls	禁用目标 registry 的 TLS 验证。
--dest-use-http	使用 HTTP 用于目标 registry。
--dry-run	仅输出操作情况，不实际 mirror 镜像。生成 **mapping.txt** 和 **pruning-plan.json** 文件。
--from <string>	指定由执行 oc-mirror 生成的镜像设置归档的路径，以加载到目标 registry 中。
-h,--help	显示帮助。
--ignore-history	下载镜像和打包层时，忽略过去的镜像。禁用增量镜像，并可能会下载更多数据。
--manifests-only	为 **ImageContentSourcePolicy** 对象生成清单，将集群配置为使用镜像 registry，但不实际镜像任何镜像。要使用此标志，您必须使用 **--from** 标志传递镜像集存档。
--max-nested-paths <int>	指定限制嵌套路径的目标 registry 的最大嵌套路径数。默认值为 **0**。
--max-per-registry <int>	指定每个 registry 允许的并发请求数。默认值为 **6**。
--oci-insecure-signature-policy	在镜像本地 OCI 目录时不要推送签名（使用 **--include-local-oci-catalogs**）。
--oci-registries-config	提供 registry 配置文件，以指定在镜像本地 OCI 目录（使用 **--include-local-oci-catalogs**）时复制的替代 registry 位置。
--skip-cleanup	跳过删除工件目录。
--skip-image-pin	不要将镜像标签替换为 Operator 目录中的摘要。
--skip-metadata-check	发布镜像集时跳过元数据。  **注意** 只有在使用 **--ignore-history** 标志创建镜像集时才建议使用。
--skip-missing	如果没有找到镜像，则跳过它而不是报告错误并中止执行。不适用于在镜像设置配置中明确指定的自定义镜像。
--skip-pruning	从目标镜像 registry 禁用自动修剪镜像。

标记	描述
--skip-verification	跳过摘要验证。
--source-skip-tls	为源 registry 禁用 TLS 验证。
--source-use-http	将普通 HTTP 用于源 registry。
-v, --verbose <int>	指定日志级别详细程度的数量。有效值为 **0** - **9**。默认值为 **0**。

7.15. 其他资源

- 关于在断开连接的环境中的集群更新

第 8 章 使用 OC ADM 命令为断开连接的安装同步镜像

您可以确保集群只使用满足您机构对外部内容控制的容器镜像。在受限网络中置备的基础架构上安装集群前，您必须将所需的容器镜像镜像(mirror)到那个环境中。通过使用 **oc adm** 命令，您可以在 OpenShift 中镜像（mirror）发行版本和目录镜像。要镜像容器镜像，您必须有一个 registry 才能进行镜像(mirror)。

重要

您必须可以访问互联网来获取所需的容器镜像。在这一流程中，您要将镜像 registry 放在可访问您的网络以及互联网的镜像（mirror）主机上。如果您没有镜像主机的访问权限，请使用镜像 Operator 目录与断开连接的集群流程一起使用，将镜像复制到可跨网络界限的设备。

8.1. 先决条件

- 您必须在托管 OpenShift Container Platform 集群的位置（如以下 registry 之一）中有一个支持 Docker v2-2 的容器镜像 registry：
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus 仓库
  - Harbor

  如果您有 Red Hat Quay 权利，请参阅有关部署 Red Hat Quay 以了解概念验证的文档，或使用 Red Hat Quay Operator。如果您需要额外的帮助来选择并安装 registry，请联络您的销售代表或红帽支持。

- 如果您还没有容器镜像 registry，OpenShift Container Platform 可以为订阅者提供一个 mirror registry for Red Hat OpenShift。*mirror registry for Red Hat OpenShift* 包含在您的订阅中，它是一个小型容器 registry，可用于在断开连接的安装中镜像 OpenShift Container Platform 所需的容器镜像。

8.2. 关于镜像 REGISTRY

您必须可以访问互联网来获取所需的容器镜像。使用一个替代的 registry 意味着，将 mirror registry 放在一个可访问您的网络以及互联网的 mirror 主机上。

您可以镜像 OpenShift Container Platform 安装所需的镜像，以及容器镜像 registry 的后续产品更新，如 Red Hat Quay、JFrog Artifactory、Sonatype Nexus Repository 或 Harbor。如果您无法访问大型容器 registry，可以使用 *mirror registry for Red Hat OpenShift*，它是包括在 OpenShift Container Platform 订阅中的一个小型容器 registry。

您可以使用支持 Docker v2-2 的任何容器 registry，如 Red Hat Quay，*mirror registry for Red Hat OpenShift*, Artifactory, Sonatype Nexus Repository, 或 Harbor。无论您所选 registry 是什么，都会将互联网上红帽托管站点的内容镜像到隔离的镜像 registry 相同。镜像内容后，您要将每个集群配置为从镜像 registry 中检索此内容。

重要

OpenShift 镜像 registry 不能用作目标 registry，因为它不支持没有标签的推送，在镜像过程中需要这个推送。

如果选择的容器 registry 不是 *mirror registry for Red Hat OpenShift*，则需要集群中置备的每台机器都可以访问它。如果 registry 无法访问，安装、更新或常规操作（如工作负载重新定位）可能会失败。因此，您必须以高度可用的方式运行镜像 registry，镜像 registry 至少必须与 OpenShift Container Platform 集群的生产环境可用性相匹配。

使用 OpenShift Container Platform 镜像填充镜像 registry 时，可以遵循以下两种情况。如果您的主机可以同时访问互联网和您的镜像 registry，而不能访问您的集群节点，您可以直接从该机器中镜像该内容。这个过程被称为 *连接的镜像(mirror)*。如果没有这样的主机，则必须将该镜像文件镜像到文件系统中，然后将该主机或者可移动介质放入受限环境中。这个过程被称为 *断开连接的镜像*。

对于已镜像的 registry，若要查看拉取镜像的来源，您必须查看 **Trying 以访问** CRI-O 日志中的日志条目。查看镜像拉取源的其他方法（如在节点上使用 **crictl images** 命令）显示非镜像镜像名称，即使镜像是从镜像位置拉取的。

注意

红帽没有针对 OpenShift Container Platform 测试第三方 registry。

附加信息

有关查看 CRI-O 日志以查看镜像源的详情，请参阅查看镜像拉取源。

8.3. 准备您的镜像主机

执行镜像步骤前，必须准备主机以检索内容并将其推送到远程位置。

8.3.1. 安装 OpenShift CLI

您可以安装 OpenShift CLI(**oc**)来使用命令行界面与 OpenShift Container Platform 进行交互。您可以在 Linux、Windows 或 macOS 上安装 **oc**。

重要

如果安装了旧版本的 **oc**，则可能无法使用 OpenShift Container Platform 中的所有命令。

下载并安装新版本的 **oc**。

8.3.1.1. 在 Linux 上安装 OpenShift CLI

您可以按照以下流程在 Linux 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **Product Variant** 列表中选择构架。
3. 从 **Version** 列表中选择适当的版本。
4. 点 **OpenShift v4.19 Linux Clients** 条目旁的 **Download Now** 来保存文件。
5. 解包存档：

```
$ tar xvf <file>
```

6. 将 **oc** 二进制文件放到 **PATH 中的目录中**。
   要查看您的 **PATH**，请执行以下命令：

```
$ echo $PATH
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
$ oc <command>
```

8.3.1.2. 在 Windows 上安装 OpenShift CLI

您可以按照以下流程在 Windows 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **Version** 列表中选择适当的版本。
3. 点 **OpenShift v4.19 Windows Client** 条目旁的 **Download Now** 来保存文件。
4. 使用 ZIP 程序解压存档。
5. 将 **oc** 二进制文件移到 **PATH 中的目录中**。
   要查看您的 **PATH**，请打开命令提示并执行以下命令：

```
C:\> path
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

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

8.3.1.3. 在 macOS 上安装 OpenShift CLI

您可以按照以下流程在 macOS 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **版本** 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.19 macOS Clients** 条目旁的 **Download Now** 来保存文件。

注意

对于 macOS arm64，请选择 **OpenShift v4.19 macOS arm64 Client**条目。

4. 解包和解压存档。
5. 将 **oc** 二进制文件移到 PATH 的目录中。
   要查看您的 **PATH**，请打开终端并执行以下命令：

```
$ echo $PATH
```

验证

- 使用 **oc** 命令验证安装：

```
$ oc <command>
```

8.4. 配置允许对容器镜像进行镜像的凭证

创建容器镜像 registry 凭证文件，可让您将镜像从红帽 mirror 到您的镜像。在安装主机上完成以下步骤。

警告

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

先决条件

- 您已将镜像 registry 配置为在断开连接的环境中使用。
- 您在镜像 registry 中标识了镜像仓库的位置，以将容器镜像镜像(mirror)到这个位置。
- 您置备了一个镜像 registry 帐户，允许将镜像上传到该镜像仓库。
- 对镜像 registry 有写权限。

流程

1. 从 Red Hat OpenShift Cluster Manager 下载 **registry.redhat.io** pull secret。
2. 运行以下命令，以 JSON 格式生成 pull secret 副本：

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

   指定到存储 pull secret 的目录的路径，以及您创建的 JSON 文件的名称。

   pull secret 示例

```
{
  "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. 运行以下命令，为您的镜像 registry 生成 base64 编码的用户名和密码或令牌：

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

通过 **<user_name>** 和 **<password>** 指定 registry 的用户名和密码。

输出示例

```
BGVtbYk3ZHAtqXs=
```

4. 编辑 JSON 文件并添加描述 registry 的部分：

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

- 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如：**registry.example.com** 或 **registry.example.com:8443**。
- 对于 **<credentials>** 值，请指定 mirror registry 的 base64 编码用户名和密码。

修改的 pull secret 示例

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

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

8.5. 镜像 OPENSHIFT CONTAINER PLATFORM 镜像存储库

镜像要在集群安装或升级过程中使用的 OpenShift Container Platform 镜像仓库。在 mirror 主机上完成以下步骤。

先决条件

- 您的镜像主机可访问互联网。
- 您已将镜像 registry 配置为在受限网络中使用，并可访问您配置的证书和凭证。
- 您已从 Red Hat OpenShift Cluster Manager 下载了 pull secret，并已修改为包含镜像存储库的身份验证。
- 如果您使用自签名证书，已在证书中指定 Subject Alternative Name。

流程

1. 查看 OpenShift Container Platform 下载页，以确定您要安装的 OpenShift Container Platform 版本，并决定 Repository Tags 页中的相应标签（tag）。
2. 设置以下所需的环境变量：
   a. 导出发行版本信息：

   ```
   $ OCP_RELEASE=<release_version>
   ```

   对于 **<release_version>**，请指定与 OpenShift Container Platform 版本对应的标签，用于您的架构，如 **4.20.1**。

   b. 导出本地 registry 名称和主机端口：

   ```
   $ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
   ```

   对于 **<local_registry_host_name>**，请指定镜像存储库的 registry 域名；对于 **<local_registry_host_port>**，请指定用于提供内容的端口。

   c. 导出本地存储库名称：

   ```
   $ LOCAL_REPOSITORY='<local_repository_name>'
   ```

对于 **<local_repository_name>**，请指定要在 registry 中创建的仓库名称，如 **ocp4/openshift4**。

d. 导出要进行镜像的存储库名称：

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

对于生产环境版本，必须指定 **openshift-release-dev**。

e. 导出 registry pull secret 的路径：

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

对于 **<path_to_pull_secret>**，请指定您创建的镜像 registry 的 pull secret 的绝对路径和文件名。

f. 导出发行版本镜像：

```
$ RELEASE_NAME="ocp-release"
```

对于生产环境版本，您必须指定 **ocp-release**。

g. 为您的集群导出构架类型：

```
$ ARCHITECTURE=<cluster_architecture>
```

指定集群的构架，如 **x86_64**, **aarch64**, **s390x**, 获 **ppc64le**。

h. 导出托管镜像的目录的路径：

```
$ REMOVABLE_MEDIA_PATH=<path>
```

指定完整路径，包括开始的前斜杠(/)字符。

3. 将版本镜像(mirror)到镜像 registry：

- 如果您的镜像主机无法访问互联网，请执行以下操作：

  i. 将可移动介质连接到连接到互联网的系统。

  ii. 查看要镜像的镜像和配置清单：

```
$ 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. 记录上一命令输出中的 **imageContentSources** 部分。您的镜像信息与您的镜像存储库相对应，您必须在安装过程中将 **imageContentSources** 部分添加到 **install-config.yaml** 文件中。

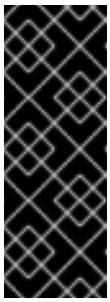
  iv. 将镜像镜像到可移动介质的目录中：

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

v. 将介质上传到受限网络环境中，并将镜像上传到本地容器 registry。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}
```

对于 **REMOVABLE_MEDIA_PATH** 变量，您必须使用与 mirror 镜像时指定的同一路径。



重要

运行 **oc image mirror** 可能会导致以下错误：**error: unable to retrieve source image**。当镜像索引包括对镜像 registry 中不再存在的镜像的引用时，会发生此错误。镜像索引可能会保留旧的引用，以便为运行这些镜像的用户在升级图表中显示新的升级路径。作为临时解决方案，您可以使用 **--skip-missing** 选项绕过错误并继续下载镜像索引。如需更多信息，请参阅 Service Mesh Operator 镜像失败。

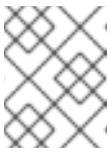
- 如果本地容器 registry 连接到镜像主机，请执行以下操作：

i. 使用以下命令直接将发行版镜像推送到本地 registry:

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

该命令将发行信息提取为摘要，其输出包括安装集群时所需的 **imageContentSources** 数据。

ii. 记录上一命令输出中的 **imageContentSources** 部分。您的镜像信息与您的镜像存储库相对应，您必须在安装过程中将 **imageContentSources** 部分添加到 **install-config.yaml** 文件中。



注意

镜像名称在镜像过程中被修补到 Quay.io， podman 镜像将在 bootstrap 虚拟机的 registry 中显示 Quay.io。

4. 要创建基于您镜像内容的安装程序，请提取内容并将其固定到发行版中：

- 如果您的镜像主机无法访问互联网，请运行以下命令：

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --icsp-file=<file> --
```

```
command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}" \
--insecure=true
```

可选：如果您不想为目标 registry 配置信任，请添加 **--insecure=true** 标志。

- 如果本地容器 registry 连接到镜像主机，请运行以下命令：

```
$ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}"
```

重要

要确保将正确的镜像用于您选择的 OpenShift Container Platform 版本，您必须从镜像内容中提取安装程序。

您必须在有活跃互联网连接的机器上执行这个步骤。

5. 对于使用安装程序置备的基础架构的集群，运行以下命令：

```
$ openshift-install
```

8.6. 在断开连接的环境中的 CLUSTER SAMPLES OPERATOR

在断开连接的环境中，在安装集群后执行额外的步骤来配置 Cluster Samples Operator。在准备过程中查阅以下信息：

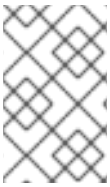
8.6.1. 协助镜像的 Cluster Samples Operator

在安装过程中，OpenShift Container Platform 在 **openshift-cluster-samples-operator** 命名空间中创建一个名为 **imagestreamtag-to-image** 的配置映射。

imagestreamtag-to-image 配置映射包含每个镜像流标签的条目（填充镜像）。

配置映射中 data 字段中每个条目的键格式为 **<image_stream_name>_<image_stream_tag_name>**。

在断开连接的 OpenShift Container Platform 安装过程中，Cluster Samples Operator 的状态被设置为 **Removed**。如果您将其改为 **Managed**，它会安装示例。



注意

在网络限制或断开连接的环境中使用示例可能需要通过网络访问服务。某些示例服务包括：Github、Maven Central、npm、RubyGems、PyPi 等。这可能需要执行额外的步骤，让 Cluster Samples Operator 对象能够访问它们所需的服务。

使用以下原则来决定需要 mirror 哪些镜像用于镜像流导入：

- 在 Cluster Samples Operator 被设置为 **Removed** 时，您可以创建镜像的 registry，或决定您要使用哪些现有镜像 registry。
- 使用新的配置映射作为指南来镜像您要镜像的 registry 的示例。

- 将没有镜像的任何镜像流添加到 Cluster Samples Operator 配置对象的 **skippedImagestreams** 列表中。
- 将 Cluster Samples Operator 配置对象的 **samplesRegistry** 设置为已镜像的 registry。
- 然后，将 Cluster Samples Operator 设置为 **Managed** 来安装您已镜像的镜像流。

8.7. 镜像用于断开连接的集群的 OPERATOR 目录

您可以使用 **oc adm catalog mirror** 命令将红帽提供的目录或自定义目录的 Operator 内容镜像到容器镜像 registry 中。目标 registry 必须支持 Docker v2-2。对于受限网络中的集群，此 registry 可以是集群有网络访问权限的 registry，如在受限网络集群安装过程中创建的镜像 registry。

重要

- OpenShift 镜像 registry 不能用作目标 registry，因为它不支持没有标签的推送，在镜像过程中需要这个推送。
- 运行 **oc adm catalog mirror** 可能会导致以下错误： **error: unable to retrieve source image**。当镜像索引包括对镜像 registry 中不再存在的镜像的引用时，会发生此错误。镜像索引可能会保留旧的引用，以便为运行这些镜像的用户在升级图表中显示新的升级路径。作为临时解决方案，您可以使用 **--skip-missing** 选项绕过错误并继续下载镜像索引。如需更多信息，请参阅 Service Mesh Operator 镜像失败。

oc adm catalog mirror 命令还会自动将在镜像过程中指定的索引镜像（无论是红帽提供的索引镜像还是您自己的自定义构建索引镜像）镜像到目标 registry。然后，您可以使用镜像的索引镜像创建一个目录源，允许 Operator Lifecycle Manager（OLM）将镜像目录加载到 OpenShift Container Platform 集群。

其他资源

- 在断开连接的环境中使用 Operator Lifecycle Manager。

8.7.1. 先决条件

与断开连接的集群一起使用的 Operator 目录具有以下先决条件：

- 没有网络访问限制的工作站
- **podman** 1.9.3 或更高版本。
- 如果要过滤或 *prune* 一个现存的目录，且仅选择性地镜像部分 Operator，请参阅以下部分：
  - 安装 opm CLI
  - 更新或过滤基于文件的目录镜像
- 如果要镜像红帽提供的目录，请在具有无网络访问限制的工作站中运行以下命令，以便与 **registry.redhat.io** 进行身份验证：

```
$ podman login registry.redhat.io
```

- 访问支持 Docker v22 的镜像 registry。

- 在镜像 registry 上，决定使用哪个存储库或命名空间来存储已镜像的 Operator 内容。例如，您可以创建一个 **olm-mirror** 存储库。
- 如果您的镜像 registry 无法访问互联网，请将可移动介质连接到您的没有网络访问限制的工作站。
- 如果您正在使用私有 registry，包括 **registry.redhat.io**，请将 **REG_CREDS** 环境变量设置为 registry 凭证的文件路径，以便在后续步骤中使用。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

8.7.2. 提取和镜像目录内容

oc adm catalog mirror 命令提取索引镜像的内容，以生成镜像所需的清单。命令的默认行为会生成清单，然后会自动将索引镜像以及索引镜像本身中的所有镜像内容镜像（mirror）到您的镜像 registry。

另外，如果您的镜像 registry 位于完全断开连接的主机上，或者断开连接的或 *airgapped* 主机上，您可以首先将内容镜像到可移动介质，将介质移到断开连接的环境中，然后将内容从介质镜像到 registry。

8.7.2.1. 将目录内容镜像到同一网络上的 registry

如果您的镜像 registry 与您的没有网络访问限制的工作站位于同一个网络中，请在您的工作站上执行以下操作：

流程

1. 如果您的镜像 registry 需要身份验证，请运行以下命令登录到 registry:

```
$ podman login <mirror_registry>
```

2. 运行以下命令，将内容提取并镜像到镜像 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. 指定您要镜像的目录的索引镜像。
2. 指定要将 Operator 内容镜像到的目标 registry 的完全限定域名(FQDN)。镜像 registry **<repository>** 可以是 registry 上的任何现有存储库或命名空间，如先决条件中所述，如 **olm-mirror**。如果在镜像过程中找到现有的存储库，存储库名称将添加到生成的镜像名称中。如果您不希望镜像名称包含存储库名称，请省略此行中的 **<repository>** 值，例如 **<mirror_registry>:<port>**。
3. 可选：如果需要，指定 registry 凭证文件的位置。**registry.redhat.io** 需要 **{REG_CREDS}**。
4. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。

5 可选：在有多个变体可用时，指定索引镜像的平台和架构。镜像被传递为 **'<platform>/<arch>[/<variant>]'**。这不适用于索引引用的镜像。有效值为 **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **linux/arm64**.

6 可选：只生成镜像所需的清单，而不实际将镜像内容镜像到 registry。这个选项对检查哪些将被镜像（mirror）非常有用，如果您只需要一小部分软件包，可以对映射列表进行修改。然后，您可以使用带有 **oc image mirror** 命令的 **mapping.txt** 文件来在以后的步骤中镜像修改的镜像列表。此标志用于从目录中对内容进行高级选择性镜像。

输出示例

```
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 命令生成的临时 **index.db** 数据库的目录。

2 记录生成的 manifests 目录名称。该目录在后续过程中被引用。

注意

Red Hat Quay 不支持嵌套仓库。因此，运行 **oc adm catalog mirror** 命令会失败，并显示 **401** 未授权错误。作为临时解决方案，您可以在运行 **oc adm catalog mirror** 命令时使用 **--max-components=2** 选项来禁用嵌套存储库的创建。有关此临时解决方案的更多信息，请参阅 Unauthorized error thrown while using catalog mirror command with Quay registry。

8.7.2.2. 将目录内容镜像到 airgapped registry

如果您的镜像 registry 位于完全断开连接的主机上，或 airgapped 主机上，请执行以下操作。

流程

1. 在您的工作站中运行以下命令，且没有网络访问权限将内容镜像到本地文件中：

```
$ 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 指定您要镜像的目录的索引镜像。

2 指定要镜像到当前目录中的本地文件的内容。

3 可选：如果需要，指定 registry 凭证文件的位置。

4 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。

5 可选：在有多个变体可用时，指定索引镜像的平台和架构。镜像被指定为 **'<platform>/<arch>[/<variant>]'**。这不适用于索引引用的镜像。有效值为 **linux/amd64**,

输出示例

```
...
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 记录生成的 manifests 目录名称。该目录在后续过程中被引用。

2 记录根据您提供的索引镜像扩展的 **file://** 路径。这个路径在后续步骤中被引用。

此命令会在当前目录中创建 **v2/** 目录中。

2. 将 **v2/** 目录复制到可移动介质。
3. 物理删除该介质并将其附加到断开连接的环境中可访问镜像 registry 的主机。
4. 如果您的镜像 registry 需要身份验证，请在断开连接的环境中的主机上运行以下命令以登录到 registry：

```
$ podman login <mirror_registry>
```

5. 从包含 **v2/** 目录的父目录运行以下命令，将镜像从本地文件上传到镜像 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
```

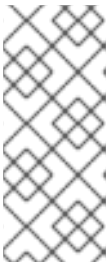
1 指定上一命令输出中的 **file://** 路径。

2 指定要将 Operator 内容镜像到的目标 registry 的完全限定域名(FQDN)。镜像 registry **<repository>** 可以是 registry 上的任何现有存储库或命名空间，如先决条件中所述，如 **olm-mirror**。如果在镜像过程中找到现有的存储库，存储库名称将添加到生成的镜像名称中。如果您不希望镜像名称包含存储库名称，请省略此行中的 **<repository>** 值，例如 **<mirror_registry>:<port>**。

3 可选：如果需要，指定 registry 凭证文件的位置。

4 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。

5 可选：在有多个变体可用时，指定索引镜像的平台和架构。镜像被指定为 **'<platform>/<arch>[/<variant>]'**。这不适用于索引引用的镜像。有效值为 **linux/amd64**, **linux/ppc64le**, **linux/s390x**, **linux/arm64**, 和 **.***



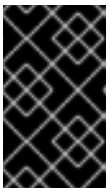
注意

Red Hat Quay 不支持嵌套仓库。因此，运行 **oc adm catalog mirror** 命令会失败，并显示 **401** 未授权错误。作为临时解决方案，您可以在运行 **oc adm catalog mirror** 命令时使用 **--max-components=2** 选项来禁用嵌套存储库的创建。有关此临时解决方案的更多信息，请参阅 Unauthorized error thrown while using catalog mirror command with Quay registry。

6. 再次运行 **oc adm catalog mirror** 命令。使用新镜像的索引镜像作为源，以及上一步中使用的同一镜像 registry 目标：

```
$ oc adm catalog mirror \
    <mirror_registry>:<port>/<index_image> \
    <mirror_registry>:<port>[/<repository>] \
    --manifests-only \ 1
    [-a ${REG_CREDS}] \
    [--insecure]
```

1 此步骤需要 **--manifests-only** 标志，以便该命令不会再次复制所有镜像的内容。



重要

这一步是必需的，因为上一步中生成的 **imageContentSourcePolicy.yaml** 文件中的镜像映射必须从本地路径更新为有效的镜像位置。如果不这样做，会在稍后的步骤中创建 **ImageContentSourcePolicy** 对象时会导致错误。

在镜像目录后，您可以继续执行集群的其余部分。在集群安装成功完成后，您必须指定此流程中的 manifests 目录来创建 **ImageContentSourcePolicy** 和 **CatalogSource** 对象。需要这些对象才能从 OperatorHub 安装 Operator。

8.7.3. 生成的清单

将 Operator 目录内容镜像到镜像 registry 后，会在当前目录中生成清单目录。

如果您将内容镜像到同一网络上的 registry，则目录名称采用以下模式：

```
manifests-<index_image_name>-<random_number>
```

如果您在上一节中将内容镜像到断开连接的主机上的 registry，则目录名称采用以下模式：

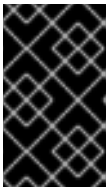
```
manifests-index/<repository>/<index_image_name>-<random_number>
```

注意

清单目录名称在后续过程中被引用。

manifests 目录包含以下文件，其中的一些文件可能需要进一步修改：

- **catalogSource.yaml** 文件是 **CatalogSource** 对象的基本定义，它预先填充索引镜像标签及其他相关元数据。此文件可原样使用，或进行相应修改来在集群中添加目录源。



重要

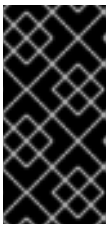
如果将内容镜像到本地文件，您必须修改 catalogSource **.yaml** 文件，从 **metadata.name** 字段中删除任何反斜杠(/)字符。否则，当您试图创建对象时，会失败并显示 "invalid resource name" 错误。

- 用来定义 **ImageContentSourcePolicy** 对象的 **imageContentSourcePolicy.yaml**，它可以将节点配置为在 Operator 清单中存储的镜像（image）引用和镜像 (mirror) 的 registry 间进行转换。

注意

如果您的集群使用 **ImageContentSourcePolicy** 对象来配置存储库镜像，则只能将全局 pull secret 用于镜像 registry。您不能在项目中添加 pull secret。

- **mapping.txt** 文件，在其中包含所有源镜像，并将它们映射到目标 registry。此文件与 **oc image mirror** 命令兼容，可用于进一步自定义镜像（mirror）配置。



重要

如果您在镜像过程中使用 **--manifests-only** 标志，并希望进一步调整要镜像的软件包子集，请参阅 OpenShift Container Platform 4.7 文档中的镜像软件包清单格式目录镜像流程中有关修改 **mapping.txt** 文件并使用 **oc image mirror** 命令的步骤。

8.7.4. 安装后的要求

在镜像目录后，您可以继续执行集群的其余部分。在集群安装成功完成后，您必须指定此流程中的 manifests 目录来创建 **ImageContentSourcePolicy** 和 **CatalogSource** 对象。这些对象需要填充和启用从 OperatorHub 安装 Operator。

其他资源

- 从镜像的 Operator 目录填充 OperatorHub
- 更新或过滤基于文件的目录镜像

8.8. 后续步骤

- 在您在受限网络中置备的基础架构上安装集群，如 VMware vSphere、裸机或 Amazon Web Services。

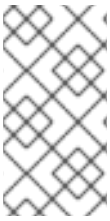
8.9. 其他资源

- 有关使用 must-gather 的更多信息，请参阅收集有关特定功能的数据。

第 9 章 在断开连接的环境中安装集群

您可以在断开连接的环境中安装 OpenShift Container Platform 集群，选择最适合您的要求的安装方法和基础架构。这包括在内部硬件或云托管服务（如 Amazon Web Services (AWS)）上安装 OpenShift Container Platform。

以下小节概述了在断开连接的环境中安装集群的所有支持方法。



注意

要了解使用特定方法安装集群的其他要求，请务必在文档的相应部分中查看其他内容。

例如，如果您计划使用安装程序置备的基础架构在 AWS 上安装集群，请参阅配置 AWS 帐户和准备在 AWS 上安装集群

9.1. 使用基于代理的安装程序安装集群

要了解更多有关使用基于代理的安装程序在断开连接的环境中安装集群的信息，请查看以下页面：

- 了解断开连接的安装镜像
- 使用基于代理的安装程序安装 OpenShift Container Platform 集群

9.2. 在 AMAZON WEB SERVICES 上安装集群

要在断开连接的环境中在 Amazon Web Services (AWS) 上安装集群，请参阅以下步骤：

- 安装程序置备的基础架构： 在受限网络中在 AWS 上安装集群
- 用户置备的基础架构：在带有用户置备的受限网络中的 AWS 上安装集群

9.3. 在 MICROSOFT AZURE 上安装集群

要在断开连接的环境中在 Microsoft Azure 上安装集群的更多信息，请参阅以下步骤：

- 安装程序置备的基础架构：在受限网络中在 Azure 上安装集群
- 用户置备的基础架构：在带有用户置备的受限网络中的 Azure 上安装集群

9.4. 在 GOOGLE CLOUD 上安装集群

要在断开连接的环境中在 Google Cloud 上安装集群，请参阅以下步骤：

- 安装程序置备的基础架构：在受限网络中在 Google Cloud 上安装集群
- 用户置备的基础架构：在带有用户置备的受限网络中的 Google Cloud 上安装集群

9.5. 在 IBM CLOUD 上安装集群

要在断开连接的环境中在 IBM Cloud® 上安装集群的更多信息，请参阅以下步骤：

- 在受限网络中的 IBM Cloud 上安装集群

9.6. 在 NUTANIX 上安装集群

要在断开连接的环境中了解更多有关在 Nutanix 上安装集群的信息，请参阅以下步骤：

- 在受限网络中的 Nutanix 上安装集群

9.7. 安装裸机集群

要在断开连接的环境中安装裸机集群的更多信息，请参阅以下步骤：

- 在受限网络中安装用户置备的裸机集群

9.8. 在 IBM Z (R) 或 IBM (R) LINUXONE 上安装集群

要在断开连接的环境中在 IBM Z® 或 IBM® LinuxONE 上安装集群，请参阅以下步骤：

- 在受限网络中的 IBM Z 和 IBM LinuxONE 中使用 z/VM 安装集群
- 在受限网络中的 IBM Z 和 IBM LinuxONE 上使用 RHEL KVM 安装集群
- 在受限网络中的 IBM Z 和 IBM LinuxONE 上的 LPAR 上安装集群

9.9. 在 IBM POWER 上安装集群

要在断开连接的环境中在 IBM Power 上安装集群的更多信息，请参阅以下步骤：

- 在受限网络中的 IBM Power 上安装集群

9.10. 在 OPENSTACK 上安装集群

要在断开连接的环境中在 Red Hat OpenStack Platform (RHOSP) 上安装集群的更多信息，请参阅以下步骤：

- 在受限网络中的 OpenStack 上安装集群

9.11. 在 VSPHERE 上安装集群

要在断开连接的环境中在 VMware vSphere 上安装集群的更多信息，请参阅以下步骤：

- 安装程序置备的基础架构：在受限网络中在 vSphere 上安装集群
- 用户置备的基础架构：在带有用户置备的受限网络中的 vSphere 上安装集群

第 10 章 在断开连接的环境中使用 OPERATOR LIFECYCLE MANAGER。

对于在断开连接的环境中的 OpenShift Container Platform 集群，Operator Lifecycle Manager (OLM) 默认无法访问托管在远程 registry 上的红帽提供的 OperatorHub 源，因为这些远程源需要足够的互联网连接。

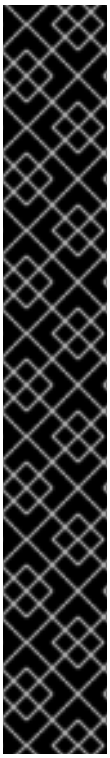
但是，作为集群管理员，如果您有一个有完全互联网访问的工作站，则仍可以让集群在断开连接的环境中使用 OLM。工作站需要完全访问互联网来拉取远程 OperatorHub 内容，用于准备远程源的本地镜像，并将内容推送到镜像 registry。

镜像 registry 可以位于堡垒主机上，它需要连接到您的工作站和断开连接的集群，或者一个完全断开连接的或 *airgapped* 主机，这需要可移动介质物理将镜像内容移到断开连接的环境中。

本指南描述了在断开连接的环境中启用 OLM 所需的流程：

- 为 OLM 禁用默认远程 OperatorHub 源。
- 使用有完全互联网访问的工作站来创建并推送 OperatorHub 内容的本地镜像到镜像 registry。
- 将 OLM 配置为从镜像 registry 上的本地源而不是默认的远程源安装和管理 Operator。

在断开连接的环境中启用 OLM 后，您可以继续使用不受限制的工作站在发布较新版本的 Operator 时保持本地 OperatorHub 源更新。



重要

虽然 OLM 可以从本地源管理 Operator，但给定 Operator 在断开连接的环境中成功运行仍取决于 Operator 本身满足以下条件：

- 在 **ClusterServiceVersion** (CSV) 对象的 **relatedImages**参数中列出所有相关的镜像，或 Operator 执行时可能需要的其他容器镜像。
- 通过摘要 (SHA) 而不是标签来引用所有指定的镜像。

您可以通过使用以下选择过滤，在 Red Hat Ecosystem Catalog 上搜索软件以获取支持以断开连接模式运行的红帽 Operator 列表：

类型	容器化应用程序
部署方法	Operator
基础架构特性	断开连接

其他资源

- 红帽提供的 Operator 目录

10.1. 先决条件

- 以具有 **cluster-admin** 权限的用户身份登录 OpenShift Container Platform 集群。

- 如果您在 IBM Z® 的断开连接的环境中使用 OLM，则必须至少为放置 registry 的目录分配 12 GB 的存储空间。

10.2. 禁用默认的 OPERATORHUB 目录源

在 OpenShift Container Platform 安装过程中，默认为 OperatorHub 配置由红帽和社区项目提供的源内容的 operator 目录。在受限网络环境中，必须以集群管理员身份禁用默认目录。然后，您可以将 OperatorHub 配置为使用本地目录源。

流程

- 通过在 **OperatorHub** 对象中添加 **disableAllDefaultSources: true 来** 禁用默认目录的源：

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

提示

或者，您可以使用 Web 控制台管理目录源。在 **Administration → Cluster Settings → Configuration → OperatorHub** 页面中，点 **Sources** 选项卡，您可以在其中创建、更新、删除、禁用和启用单独的源。

10.3. 对 OPERATOR 目录进行镜像(MIRROR)

有关与断开连接的集群一起使用的 Operator 目录的说明，请参阅镜像 Operator 目录以用于断开连接的集群。

重要

从 OpenShift Container Platform 4.11 开始，默认的红帽提供的 Operator 目录以基于文件的目录格式发布。通过以过时的 SQLite 数据库格式发布的 4.10，用于 OpenShift Container Platform 4.6 的默认红帽提供的 Operator 目录。

与 SQLite 数据库格式相关的 **opm** 子命令、标志和功能已被弃用，并将在以后的版本中删除。功能仍被支持，且必须用于使用已弃用的 SQLite 数据库格式的目录。

许多 **opm** 子命令和标志都用于 SQLite 数据库格式，如 **opm index prune**，它们无法使用基于文件的目录格式。有关使用基于文件的目录的更多信息，请参阅 Operator Framework 打包格式，管理自定义目录，以及使用 oc-mirror 插件 v2 为断开连接的安装 mirror 镜像 。

10.4. 在集群中添加目录源

将目录源添加到 OpenShift Container Platform 集群可为用户发现和安装 Operator。集群管理员可以创建一个 **CatalogSource** 对象来引用索引镜像。OperatorHub 使用目录源来填充用户界面。

提示

或者，您可以使用 Web 控制台管理目录源。在 **Administration → Cluster Settings → Configuration → OperatorHub** 页面中，点 **Sources** 选项卡，您可以在其中创建、更新、删除、禁用和启用单独的源。

先决条件

- 构建并推送索引镜像到 registry。

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 创建一个 **CatalogSource** 对象来引用索引镜像。如果使用 **oc adm catalog mirror** 命令将目录镜像到目标 registry，您可以使用 manifests 目录中生成的 **catalogSource.yaml** 文件作为起点。

   a. 根据您的规格修改以下内容，并将它保存为 **catalogSource.yaml** 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog (1)
  namespace: openshift-marketplace (2)
spec:
  sourceType: grpc
  grpcPodConfig:
    securityContextConfig: <security_mode> (3)
  image: <registry>/<namespace>/redhat-operator-index:v4.19 (4)
  displayName: My Operator Catalog
  publisher: <publisher_name> (5)
  updateStrategy:
    registryPoll: (6)
      interval: 30m
```

(1) 如果您在上传到 registry 前将内容镜像到本地文件，请从 **metadata.name** 字段中删除任何反斜杠(/)字符，以避免在创建对象时出现 "invalid resource name" 错误。

(2) 如果您希望目录源对所有命名空间中的用户全局可用，请指定 **openshift-marketplace** 命名空间。否则，您可以指定一个不同的命名空间来对目录进行作用域并只对该命名空间可用。

(3) 指定 **legacy** 或 **restricted** 的值。如果没有设置该字段，则默认值为 **legacy**。在以后的 OpenShift Container Platform 发行版本中，计划默认值为 **restricted**。

注意

如果您的目录无法使用 **restricted** 权限运行，建议您手动将此字段设置为 **legacy**。

(4) 指定索引镜像。如果您在镜像名称后指定了标签，如 **:v4.19**，则目录源 Pod 会使用镜像 pull 策略 **Always**，这意味着 pod 始终在启动容器前拉取镜像。如果您指定了摘要，如 **@sha256:<id>**，则镜像拉取策略为 **IfNotPresent**，这意味着仅在节点上不存在的镜像时才拉取镜像。

(5) 指定发布目录的名称或机构名称。

(6) 目录源可以自动检查新版本以保持最新。

   b. 使用该文件创建 **CatalogSource** 对象：

```
$ oc apply -f catalogSource.yaml
```

2. 确定成功创建以下资源。

   a. 检查 pod:

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

   输出示例

   ```
   NAME                                READY   STATUS    RESTARTS  AGE
   my-operator-catalog-6njx6           1/1     Running   0         28s
   marketplace-operator-d9f549946-96sgr  1/1     Running   0         26h
   ```

   b. 检查目录源：

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

   输出示例

   ```
   NAME              DISPLAY               TYPE PUBLISHER  AGE
   my-operator-catalog  My Operator Catalog  grpc          5s
   ```

   c. 检查软件包清单：

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

   输出示例

   ```
   NAME                  CATALOG               AGE
   jaeger-product            My Operator Catalog   93s
   ```

现在，您可以在 OpenShift Container Platform Web 控制台中通过 **OperatorHub** 安装 Operator。

其他资源

- 从私有 registry 访问 Operator 的镜像
- 自定义目录源的镜像模板
- 镜像拉取（pull）策略

10.5. 后续步骤

- 更新安装的 Operator

第 11 章 在断开连接的环境中更新集群

11.1. 关于在断开连接的环境中的集群更新

断开连接的环境是集群节点无法访问互联网或要在本地管理更新建议和发行镜像，用于策略或性能的目的。本节介绍镜像（mirror） OpenShift Container Platform 镜像、管理 OpenShift Update Service，并在断开连接的环境中执行集群更新。

11.1.1. 镜像 OpenShift Container Platform 镜像

要在断开连接的环境中更新集群，您的集群环境必须有权访问具有目标更新所需镜像和资源的镜像 registry。一个独立的容器镜像 registry 足以为断开连接的网络中的多个集群托管 mirror 的镜像。以下页提供了将镜像镜像到断开连接的集群中的存储库的说明：

- 镜像 OpenShift Container Platform 镜像

11.1.2. 在断开连接的环境中执行集群更新

您可以使用以下步骤之一更新断开连接的 OpenShift Container Platform 集群：

- 使用 OpenShift Update Service 在断开连接的环境中更新集群
- 在没有 OpenShift Update Service 的断开连接的环境中更新集群

11.1.3. 从集群中删除 OpenShift Update Service

您可以使用以下步骤从集群中卸载 OpenShift Update Service (OSUS) 的本地副本：

- 从集群中删除 OpenShift Update Service

11.2. 镜像 OPENSHIFT CONTAINER PLATFORM 镜像

您必须将容器镜像镜像到镜像 registry 中，然后才能在受限网络环境中更新集群。您还可以在连接的环境中使用此流程来确保集群只运行满足您机构对外部内容控制的批准的容器镜像。

注意

您的镜像 registry 必须始终在集群运行时都运行。

以下步骤概述了如何将镜像镜像到镜像 registry 的高级别工作流：

1. 在用于检索和推送发行镜像的所有设备上安装 OpenShift CLI (**oc**)。
2. 下载 registry pull secret，并将其添加到集群中。
3. 如果使用 oc-mirror OpenShift CLI (**oc**)插件 ：
   a. 在用于检索和推送发行镜像的所有设备中安装 oc-mirror 插件。
   b. 为插件创建镜像设置配置文件，以便在决定要镜像的发行镜像时使用。您可以稍后编辑此配置文件，以更改插件镜像的镜像。
   c. 将目标发行镜像直接镜像到镜像 registry 或可移动介质，然后镜像到镜像 registry。

d. 配置集群以使用 oc-mirror 插件生成的资源。

e. 根据需要重复这些步骤以更新您的镜像 registry。

4. 如果使用 **oc adm release mirror** 命令 ：

a. 设置环境变量，对应于您的环境以及您要镜像的发行镜像。

b. 将目标发行镜像直接镜像到镜像 registry 或可移动介质，然后镜像到镜像 registry。

c. 根据需要重复这些步骤以更新您的镜像 registry。

与使用 **oc adm release mirror** 命令相比，oc-mirror 插件具有以下优点：

- 它可以镜像容器镜像以外的内容。
- 在首次镜像镜像后，可以更轻松地更新 registry 中的镜像。
- oc-mirror 插件提供了一种自动从 Quay 镜像发行版本有效负载的方法，并为在断开连接的环境中运行的 OpenShift Update Service 构建最新的图形数据镜像。

11.2.1. 使用 oc-mirror 插件镜像资源

您可以使用 oc-mirror OpenShift CLI (**oc**)插件在完全或部分断开连接的环境中将镜像镜像到镜像 registry。您必须从具有互联网连接的系统运行 oc-mirror，以便从官方红帽 registry 中下载所需的镜像。

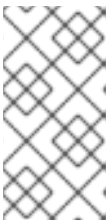
如需了解更多详细信息，请参阅使用 oc-mirror 插件 v2 为断开连接的安装镜像镜像 。

11.2.2. 使用 oc adm release mirror 命令 mirror 镜像

您可以使用 **oc adm release mirror** 命令将镜像镜像到您的镜像 registry。

11.2.2.1. 先决条件

- 您必须在托管 OpenShift Container Platform 集群的位置（如 Red Hat Quay）中有一个支持 Docker v2-2 的容器镜像 registry。



注意

如果使用 Red Hat Quay，则必须在 oc-mirror 插件中使用 3.6 或更高版本的版本。如果您有 Red Hat Quay 权利，请参阅有关部署 Red Hat Quay 以了解概念验证的文档，或使用 Quay Operator。如果您需要额外的帮助来选择并安装 registry，请联络您的销售代表或红帽支持。

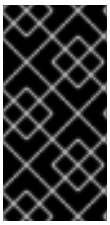
如果您没有容器镜像 registry 的现有解决方案，则 OpenShift Container Platform 订阅中包括了 mirror registry for Red Hat OpenShift 。*mirror registry for Red Hat OpenShift* 是一个小型容器 registry，可用于在断开连接的环境中 mirror OpenShift Container Platform 容器镜像。

11.2.2.2. 准备您的镜像主机

执行镜像步骤前，必须准备主机以检索内容并将其推送到远程位置。

11.2.2.2.1. 安装 OpenShift CLI

您可以安装 OpenShift CLI(**oc**)来使用命令行界面与 OpenShift Container Platform 进行交互。您可以在 Linux、Windows 或 macOS 上安装 **oc**。



重要

如果安装了旧版本的 **oc**，则可能无法使用 OpenShift Container Platform 中的所有命令。

下载并安装新版本的 **oc**。如果要在断开连接的环境中更新集群，请安装您要升级到的 **oc** 版本。

11.2.2.2.1.1. 在 Linux 上安装 OpenShift CLI

您可以按照以下流程在 Linux 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **Product Variant** 列表中选择构架。
3. 从 **Version** 列表中选择适当的版本。
4. 点 **OpenShift v4.19 Linux Clients** 条目旁的 **Download Now** 来保存文件。
5. 解包存档：

```
$ tar xvf <file>
```

6. 将 **oc** 二进制文件放到 **PATH** 中的目录中。
   要查看您的 **PATH**，请执行以下命令：

```
$ echo $PATH
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
$ oc <command>
```

11.2.2.2.1.2. 在 Windows 上安装 OpenShift CLI

您可以按照以下流程在 Windows 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **Version** 列表中选择适当的版本。
3. 点 **OpenShift v4.19 Windows Client** 条目旁的 **Download Now** 来保存文件。
4. 使用 ZIP 程序解压存档。

5. 将 **oc** 二进制文件移到 **PATH 中的目录中**。
   要查看您的 **PATH**，请打开命令提示并执行以下命令：

```
C:\> path
```

验证

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

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

11.2.2.2.1.3. 在 macOS 上安装 OpenShift CLI

您可以按照以下流程在 macOS 上安装 OpenShift CLI(**oc**)二进制文件。

流程

1. 进入到红帽客户门户网站上的 Download OpenShift Container Platform 页。
2. 从 **版本** 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.19 macOS Clients**条目旁的 **Download Now** 来保存文件。

注意

对于 macOS arm64，请选择 **OpenShift v4.19 macOS arm64 Client**条目。

4. 解包和解压存档。
5. 将 **oc** 二进制文件移到 PATH 的目录中。
   要查看您的 **PATH**，请打开终端并执行以下命令：

```
$ echo $PATH
```

验证

- 使用 **oc** 命令验证安装：

```
$ oc <command>
```

其他资源

- 安装和使用 CLI 插件

11.2.2.2.2. 配置允许对容器镜像进行镜像的凭证

创建容器镜像 registry 凭证文件，可让您将镜像从红帽 mirror 到您的镜像。在安装主机上完成以下步骤。

警告

安装集群时不要使用此镜像 registry 凭据文件作为 pull secret。如果在安装集群时提供此文件，集群中的所有机器都将具有镜像 registry 的写入权限。

先决条件

- 您已将镜像 registry 配置为在断开连接的环境中使用。
- 您在镜像 registry 中标识了镜像仓库的位置，以将容器镜像镜像(mirror)到这个位置。
- 您置备了一个镜像 registry 帐户，允许将镜像上传到该镜像仓库。
- 对镜像 registry 有写权限。

流程

1. 从 Red Hat OpenShift Cluster Manager 下载 **registry.redhat.io** pull secret。
2. 运行以下命令，以 JSON 格式生成 pull secret 副本：

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

指定到存储 pull secret 的目录的路径，以及您创建的 JSON 文件的名称。

pull secret 示例

```
{
  "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. 可选：如果使用 oc-mirror 插件，请将文件保存为 **~/.docker/config.json** 或 **$XDG_RUNTIME_DIR/containers/auth.json**。

- 如果 **.docker** 或 **$XDG_RUNTIME_DIR/containers** 目录不存在，请输入以下命令来创建：

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

其中 **<directory_name>** 是 **~/.docker** 或 **$XDG_RUNTIME_DIR/containers**。

- 输入以下命令将 pull secret 复制到适当的目录中：

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

其中 **<directory_name>** 是 **~/.docker** 或 **$XDG_RUNTIME_DIR/containers**，**<auth_file>** 是 **config.json** 或 **auth.json**。

3. 运行以下命令，为您的镜像 registry 生成 base64 编码的用户名和密码或令牌：

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

通过 **<user_name>** 和 **<password>** 指定 registry 的用户名和密码。

输出示例

```
BGVtbYk3ZHAtqXs=
```

4. 编辑 JSON 文件并添加描述 registry 的部分：

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

- 对于 **<mirror_registry>**，指定 registry 域名，以及您的镜像 registry 用来提供内容的可选端口。例如：**registry.example.com** 或 **registry.example.com:8443**。

- 对于 **<credentials>** 值，请指定 mirror registry 的 base64 编码用户名和密码。

修改的 pull secret 示例

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

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

11.2.2.3. 将镜像（images）mirror 到一个 mirror registry

重要

为了避免 OpenShift Update Service 应用程序过量内存用量，您必须将发行镜像镜像到单独的存储库，如以下步骤所述。

先决条件

- 您已将镜像 registry 配置为在受限网络中使用，并可访问您配置的证书和凭证。
- 您已从 Red Hat OpenShift Cluster Manager 下载了 pull secret，并已修改为包含镜像存储库的身份验证。
- 如果您使用自签名证书，已在证书中指定 Subject Alternative Name。

流程

1. 使用 Red Hat OpenShift Container Platform Upgrade Graph visualizer 和 update planner 计划从一个版本升级到另一个版本。OpenShift Upgrade Graph 提供频道图形，并演示了如何确认您的当前和预定集群版本之间有更新路径。
2. 设置所需的环境变量：
   a. 导出发行版本信息：

```
$ export OCP_RELEASE=<release_version>
```

   对于 **<release_version>**，请指定与升级到的 OpenShift Container Platform 版本对应的标签，如 **4.5.4**。

   b. 导出本地 registry 名称和主机端口：

```
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

   对于 **<local_registry_host_name>**，请指定镜像存储库的 registry 域名；对于 **<local_registry_host_port>**，请指定用于提供内容的端口。

   c. 导出本地存储库名称：

```
$ LOCAL_REPOSITORY='<local_repository_name>'
```

对于 **<local_repository_name>**，请指定要在 registry 中创建的仓库名称，如 **ocp4/openshift4**。

d. 如果使用 OpenShift Update Service，请导出一个额外的本地存储库名称，使其包含发行镜像：

```
$ LOCAL_RELEASE_IMAGES_REPOSITORY='<local_release_images_repository_name>'
```

对于 **<local_release_images_repository_name>**，请指定要在 registry 中创建的仓库名称，如 **ocp4/openshift4-release-images**。

e. 导出要进行镜像的存储库名称：

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

对于生产环境版本，必须指定 **openshift-release-dev**。

f. 导出 registry pull secret 的路径：

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

对于 **<path_to_pull_secret>**，请指定您创建的镜像 registry 的 pull secret 的绝对路径和文件名。

注意

如果您的集群使用 **ImageContentSourcePolicy** 对象来配置存储库镜像，则只能将全局 pull secret 用于镜像 registry。您不能在项目中添加 pull secret。

g. 导出发行版本镜像：

```
$ RELEASE_NAME="ocp-release"
```

对于生产环境版本，您必须指定 **ocp-release**。

h. 为您的集群导出构架类型：

```
$ ARCHITECTURE=<cluster_architecture> 1
```

1 指定集群的构架，如 **x86_64**, **aarch64**, **s390x**, 获 **ppc64le**。

i. 导出托管镜像的目录的路径：

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

1 指定完整路径，包括开始的前斜杠(/)字符。

3. 查看要镜像的镜像和配置清单：

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
```

```
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE}
--dry-run
```

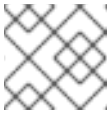
4. 将版本镜像镜像(mirror)到镜像 registry。

- 如果您的镜像主机无法访问互联网，请执行以下操作：

i. 将可移动介质连接到连接到互联网的系统。

ii. 将镜像和配置清单镜像到可移动介质的目录中：

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```



注意

此命令还会生成镜像发行镜像签名配置映射并将其保存到可移动介质中。

iii. 将介质上传到受限网络环境中，并将镜像上传到本地容器 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 对于 **REMOVABLE_MEDIA_PATH**，您必须使用与镜像镜像时指定的同一路径。

iv. 使用 **oc** 命令行界面(CLI)登录到您要升级的集群。

v. 将镜像发行镜像签名配置映射应用到连接的集群：

```
$ oc apply -f ${REMOVABLE_MEDIA_PATH}/mirror/config/<image_signature_file>
1
```

1 对于 **<image_signature_file>**，指定文件的路径和名称，例如 **signature-sha256-81154f5c03294534.yaml**。

vi. 如果使用 OpenShift Update Service，请将发行镜像镜像到单独的存储库：

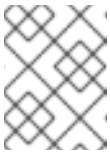
```
$ oc image mirror -a ${LOCAL_SECRET_JSON}
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_REL
EASE}-${ARCHITECTURE}
```

- 如果本地容器 registry 和集群连接到镜像主机，请执行以下操作：

i. 将发行镜像直接推送到本地 registry，并使用以下命令将配置映射应用到集群：

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --
```

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



注意

如果包含 **--apply-release-image-signature** 选项，不要为镜像签名验证创建配置映射。

ii. 如果使用 OpenShift Update Service，请将发行镜像镜像到单独的存储库：

```
$ oc image mirror -a ${LOCAL_SECRET_JSON}
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
${LOCAL_REGISTRY}/${LOCAL_RELEASE_IMAGES_REPOSITORY}:${OCP_REL
EASE}-${ARCHITECTURE}
```

11.3. 使用 OPENSHIFT UPDATE SERVICE 在断开连接的环境中更新集群

要获得与连接的集群类似的更新体验，您可以使用以下步骤在断开连接的环境中安装和配置 OpenShift Update Service（OSUS）。

以下步骤概述了如何使用 OSUS 在断开连接的环境中更新集群的高级工作流：

1. 配置对安全 registry 的访问。
2. 更新全局集群 pull secret 以访问您的镜像 registry。
3. 安装 OSUS Operator。
4. 为 OpenShift Update Service 创建图形数据容器镜像。
5. 安装 OSUS 应用程序，并将集群配置为使用您环境中的 OpenShift Update Service。
6. 如连接的集群一样，执行文档中的支持的更新步骤。

11.3.1. 在断开连接的环境中使用 OpenShift Update Service

OpenShift Update Service (OSUS) 为 OpenShift Container Platform 集群提供更新建议。红帽公开托管 OpenShift Update Service，连接的环境中的集群可以通过公共 API 连接到该服务，以检索更新建议。

但是，在断开连接的环境中的集群无法访问这些公共 API 来检索更新信息。要在断开连接的环境中提供类似的更新体验，您可以安装和配置 OpenShift Update Service，使其在断开连接的环境中可用。

单个 OSUS 实例能够为数千台集群提供建议。通过更改 replica 值，OSUS 可以水平扩展到 cater 到更多集群。因此，对于大多数断开连接的用例，一个 OSUS 实例就足够了。例如，红帽为整个连接的集群只运行一个 OSUS 实例。

如果要在不同环境中单独保留更新建议，您可以为每个环境运行一个 OSUS 实例。例如，如果您有单独的测试和暂存环境，您可能不希望在暂存环境中有一个集群来接收对版本 A 的更新建议（如果还没有在测试环境中测试该版本）。

以下小节介绍了如何安装一个 OSUS 实例，并将其配置为为集群提供更新建议。

其他资源

- 关于 OpenShift Update 服务
- 了解更新频道和发行版本

11.3.2. 先决条件

- 您必须安装了 **oc** 命令行界面（CLI）工具。
- 您需要在您的环境中置备一个容器镜像 registry，带有用于更新的容器镜像。请参阅 Mirroring OpenShift Container Platform images.

11.3.3. 为 OpenShift Update Service 配置对安全 registry 的访问

如果发行镜像包含在由自定义证书颁发机构签名的 HTTPS X.509 证书的 registry 中，请完成 为镜像 registry 访问配置额外信任存储 的步骤，以及对更新服务进行以下更改。

OpenShift Update Service Operator 需要 registry CA 证书中的配置映射键名称为 **updateservice-registry**。

更新服务的镜像 registry CA 配置映射示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  updateservice-registry: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 2
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1 OpenShift Update Service Operator 需要 registry CA 证书中的配置映射键名称 **updateservice-registry**。

2 如果 registry 带有端口，如 **registry-with-port.example.com:5000**，**:** 需要被 **..** 替换。

11.3.4. 更新全局集群 pull secret

您可以通过替换当前的 pull secret 或附加新的 pull secret 来更新集群的全局 pull secret。

当您需要一个单独的 registry 来存储镜像而不是使用安装过程中使用的 registry 时，请使用这个步骤。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 可选： 要将新的 pull secret 附加到现有 pull secret 中，请完成以下步骤：

   a. 输入以下命令下载 pull secret：

```
$ oc get secret/pull-secret -n openshift-config --template='{{index .data
".dockerconfigjson" | base64decode}}' > <pull_secret_location> 1
```

   1 **<pull_secret_location>** : 包含到 pull secret 文件的路径。

   b. 输入以下命令来添加新 pull secret：

```
$ oc registry login --registry="<registry>" \ 1
--auth-basic="<username>:<password>" \ 2
--to=<pull_secret_location> 3
```

   1 **<registry>** ：包含新的 registry。您可以在同一 registry 中包含多个软件仓库，例如 **--registry="<registry/my-namespace/my-repository>**。

   2 **<username>:<password>** : 包含新 registry 的凭证。

   3 **<pull_secret_location>** : 包含到 pull secret 文件的路径。

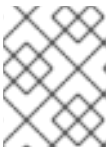
   您还可以对 pull secret 文件执行手动更新。

2. 输入以下命令为您的集群更新全局 pull secret：

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

   1 **<pull_secret_location>** : 包含到新 pull secret 文件的路径。

   该更新将推广至所有节点，可能需要一些时间，具体取决于集群大小。



注意

从 OpenShift Container Platform 4.7.4 开始，对全局 pull secret 的更改不再触发节点排空或重启。

11.3.5. 安装 OpenShift Update Service Operator

要安装 OpenShift Update Service，您必须首先使用 OpenShift Container Platform Web 控制台或 CLI 安装 OpenShift Update Service Operator。

注意

对于在断开连接的环境中安装的集群（也称为断开连接的集群），Operator Lifecycle Manager 默认无法访问托管在远程 registry 上的红帽提供的 OperatorHub 源，因为这些远程源需要有互联网连接。如需更多信息，请参阅在断开连接的环境中使用 Operator Lifecycle Manager。

11.3.5.1. 使用 Web 控制台安装 OpenShift Update Service Operator

您可以使用 Web 控制台安装 OpenShift Update Service Operator。

流程

1. 在 Web 控制台中，点 **Operators → OperatorHub**。

注意

在 **Filter by keyword...** 字段中输入 **Update Service**，以更快地查找 Operator。

2. 从可用的 Operator 列表中选择 **OpenShift Update Service**，然后点 **Install**。
   a. 选择一个 **更新频道**。
   b. 选择 **版本**。
   c. 在 **Installation Mode** 下选择 **A specific namespace on the cluster**。
   d. 为 **Installed Namespace** 选择一个命名空间，或接受推荐的命名空间 **openshift-update-service**。
   e. 选择一个 **更新批准**策略：
      - **Automatic** 策略允许 Operator Lifecycle Manager（OLM）在有新版本可用时自动更新 Operator。
      - **Manual** 策略要求集群管理员批准 Operator 更新。
   f. 点 **Install**。
3. 进入 **Operators → Installed Operators**，验证是否安装了 OpenShift Update Service Operator。
4. 确保 **OpenShift Update Service** 列在正确的命名空间中，**Status** 为 **Succeeded**。

11.3.5.2. 使用 CLI 安装 OpenShift Update Service Operator

您可以使用 OpenShift CLI（**oc**）安装 OpenShift Update Service Operator。

流程

1. 为 OpenShift Update Service Operator 创建命名空间：
   a. 为 OpenShift Update Service Operator 创建一个 **Namespace** 对象 YAML 文件，如 **update-service-namespace.yaml**：

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-update-service
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true" 1
```

1 将 **openshift.io/cluster-monitoring** 标签设置为在该命名空间中启用 Operator-recommended 集群监控。

b. 创建命名空间：

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

例如：

```
$ oc create -f update-service-namespace.yaml
```

2. 通过创建以下对象来安装 OpenShift Update Service Operator：

a. 创建一个 **OperatorGroup** 对象 YAML 文件，如 **update-service-operator-group.yaml**：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: update-service-operator-group
  namespace: openshift-update-service
spec:
  targetNamespaces:
  - openshift-update-service
```

b. 创建一个 **OperatorGroup** 对象：

```
$ oc -n openshift-update-service create -f <filename>.yaml
```

例如：

```
$ oc -n openshift-update-service create -f update-service-operator-group.yaml
```

c. 创建一个 **Subscription** 对象 YAML 文件，如 **update-service-subscription.yaml**：

订阅示例

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: update-service-subscription
  namespace: openshift-update-service
spec:
  channel: v1
  installPlanApproval: "Automatic"
  source: "redhat-operators" 1
  sourceNamespace: "openshift-marketplace"
  name: "cincinnati-operator"
```

1 指定提供 Operator 的目录源的名称。对于不使用自定义 Operator Lifecycle Manager（OLM）的集群，指定 **redhat-operators**。如果 OpenShift Container Platform 集群安装在断开连接的环境中，请指定配置 Operator Lifecycle Manager (OLM) 时创建的 **CatalogSource** 对象的名称。

d. 创建 **Subscription** 对象：

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

例如：

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

OpenShift Update Service Operator 被安装到 **openshift-update-service** 命名空间，并以 **openshift-update-service** 命名空间为目标。

3. 验证 Operator 安装：

```
$ oc -n openshift-update-service get clusterserviceversions
```

输出示例

```
NAME                         DISPLAY                VERSION   REPLACES   PHASE
update-service-operator.v4.6.0   OpenShift Update Service   4.6.0             Succeeded
...
```

如果列出了 OpenShift Update Service Operator，则会成功安装。版本号可能与所示不同。

其他资源

- 在命名空间中安装 Operator。

11.3.6. 创建 OpenShift Update Service 图形数据容器镜像

OpenShift Update Service 需要图形数据容器镜像，OpenShift Update Service 从中检索有关频道成员资格和阻止更新边缘的信息。图形数据通常直接从升级图形数据仓库中获取。在互联网连接不可用的环境中，从 init 容器加载此信息是使图形数据可供 OpenShift Update Service 使用的另一种方式。init 容器的角色是提供图形数据的本地副本，在 pod 初始化期间，init 容器会将数据复制到该服务可访问的卷中。

注意

oc-mirror OpenShift CLI (**oc**) 插件除镜像发行镜像外还会创建此图形数据容器镜像。如果您使用 oc-mirror 插件来镜像发行镜像，您可以跳过这个过程。

流程

1. 创建一个 Dockerfile，如 **./Dockerfile**，包含以下内容：

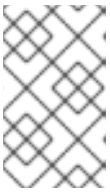
```
FROM registry.access.redhat.com/ubi9/ubi:latest

RUN curl -L -o cincinnati-graph-data.tar.gz
https://api.openshift.com/api/upgrades_info/graph-data

RUN mkdir -p /var/lib/cincinnati-graph-data && tar xvzf cincinnati-graph-data.tar.gz -C
/var/lib/cincinnati-graph-data/ --no-overwrite-dir --no-same-owner

CMD ["/bin/bash", "-c" ,"exec cp -rp /var/lib/cincinnati-graph-data/* /var/lib/cincinnati/graph-
data"]
```

2. 使用上一步中创建的 docker 文件来构建图形数据容器镜像，如 **registry.example.com/openshift/graph-data:latest**：

```
$ podman build -f ./Dockerfile -t registry.example.com/openshift/graph-data:latest
```

3. 将上一步中创建的 graph-data 容器镜像推送到 OpenShift Update Service 可以访问的存储库，如 **registry.example.com/openshift/graph-data:latest**：

```
$ podman push registry.example.com/openshift/graph-data:latest
```



注意

要将图形数据镜像推送到受限网络中的 registry，请将上一步中创建的 graph-data 容器镜像复制到可供 OpenShift Update Service 访问的存储库。运行 **oc image mirror --help** 查看可用选项。

11.3.7. 创建 OpenShift Update Service 应用程序

您可以使用 OpenShift Container Platform Web 控制台或 CLI 创建 OpenShift Update Service 应用程序。

11.3.7.1. 使用 Web 控制台创建 OpenShift Update Service 应用程序

您可以使用 OpenShift Container Platform Web 控制台使用 OpenShift Update Service Operator 创建 OpenShift Update Service 应用程序。

先决条件

- 已安装 OpenShift Update Service Operator。
- OpenShift Update Service graph-data 容器镜像已创建并推送到 OpenShift Update Service 访问的存储库。
- 已将当前发行版本和更新目标版本 mirror 到断开连接的环境中的 registry 中。

流程

1. 在 Web 控制台中，点 **Operators → Installed Operators**。
2. 从安装的 Operator 列表中选择 **OpenShift Update Service**。
3. 点 **Update Service** 选项卡。
4. 点 **Create UpdateService**。
5. 在 **Name** 字段中输入名称，如 **service**。
6. 在 **Graph Data Image** 字段中输入本地 pullspec，指向在"创建 OpenShift Update Service 图形数据容器镜像"中创建的图形数据容器镜像，如 **registry.example.com/openshift/graph-data:latest**。
7. 在 **Releases** 字段中，输入创建的 registry 和存储库，以在"镜像 OpenShift Container Platform 镜像存储库"中包括发行镜像，例如 **registry.example.com/ocp4/openshift4-release-images**。

8. 在 **Replicas** 字段中输入 **2**。
9. 单击 **Create** 以创建 OpenShift Update Service 应用。
10. 验证 OpenShift Update Service 应用程序：
    - 从 **Update Service** 选项卡中的 **UpdateServices** 列表中，点刚才创建的 Update Service 应用程序。
    - 单击 **Resources** 选项卡。
    - 验证每个应用资源的状态是否为 **Created**。

11.3.7.2. 使用 CLI 创建 OpenShift Update Service 应用程序

您可以使用 OpenShift CLI（**oc**）来创建 OpenShift Update Service 应用。

先决条件

- 已安装 OpenShift Update Service Operator。
- OpenShift Update Service graph-data 容器镜像已创建并推送到 OpenShift Update Service 访问的存储库。
- 已将当前发行版本和更新目标版本 mirror 到断开连接的环境中的 registry 中。

流程

1. 配置 OpenShift Update Service 目标命名空间，如 **openshift-update-service**：

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

   命名空间必须与 operator 组中的 **targetNamespaces** 值匹配。

2. 配置 OpenShift Update Service 应用程序的名称，如 **service**：

```
$ NAME=service
```

3. 按照"镜像 OpenShift Container Platform 镜像存储库"中配置，为发行镜像配置 registry 和存储库，如 **registry.example.com/ocp4/openshift4-release-images**：

```
$ RELEASE_IMAGES=registry.example.com/ocp4/openshift4-release-images
```

4. 将 graph-data 镜像的本地 pullspec 设置为在"创建 OpenShift Update Service 图形数据容器镜像"中创建的图形数据容器镜像，如 **registry.example.com/openshift/graph-data:latest**:

```
$ GRAPH_DATA_IMAGE=registry.example.com/openshift/graph-data:latest
```

5. 创建 OpenShift Update Service 应用程序对象：

```
$ oc -n "${NAMESPACE}" create -f - <<EOF
apiVersion: updateservice.operator.openshift.io/v1
kind: UpdateService
metadata:
```

```
  name: ${NAME}
spec:
  replicas: 2
  releases: ${RELEASE_IMAGES}
  graphDataImage: ${GRAPH_DATA_IMAGE}
EOF
```

6. 验证 OpenShift Update Service 应用程序：

   a. 使用以下命令获取策略引擎路由：

   ```
   $ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
   jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
   "${NAME}")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}"
   = http -o "${SCHEME}" = https; then break; fi; done
   ```

   您可能需要轮询，直到命令成功为止。

   b. 从策略引擎检索图形。确保为 **channel** 指定一个有效版本。例如，如果在 OpenShift Container Platform 4.19 中运行，请使用 **stable-4.19** ：

   ```
   $ while sleep 10; do HTTP_CODE="$(curl --header Accept:application/json --output
   /dev/stderr --write-out "%{http_code}" "${POLICY_ENGINE_GRAPH_URI}?
   channel=stable-4.6")"; if test "${HTTP_CODE}" -eq 200; then break; fi; echo
   "${HTTP_CODE}"; done
   ```

   这会轮询到图形请求成功为止，但生成的图形可能为空，具体取决于您已镜像的发行镜像。

注意

基于 RFC-1123 的策略引擎路由名称不能超过 63 个字符。如果您看到 **ReconcileCompleted** 状态为 **false**，原因为 **CreateRouteFailed** caused by **host must conform to DNS 1123 naming convention and must be no more than 63 characters**，请尝试使用较短的名称创建 Update Service。

11.3.8. 配置 Cluster Version Operator（CVO）

安装 OpenShift Update Service Operator 并创建 OpenShift Update Service 应用程序后，可以更新 Cluster Version Operator（CVO）从在您的环境中安装的 OpenShift Update Service 中拉取图形数据。

先决条件

- 已安装 OpenShift Update Service Operator。
- OpenShift Update Service graph-data 容器镜像已创建并推送到 OpenShift Update Service 访问的存储库。
- 已将当前发行版本和更新目标版本 mirror 到断开连接的环境中的 registry 中。
- OpenShift Update Service 应用已创建。

流程

1. 设置 OpenShift Update Service 目标命名空间，如 **openshift-update-service**：

   -

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

2. 设置 OpenShift Update Service 应用程序的名称，如 **service**：

```
$ NAME=service
```

3. 获取策略引擎路由：

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

4. 为拉取图形数据设置补丁：

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

5. 对 CVO 进行补丁，以在您的环境中使用 OpenShift Update Service：

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

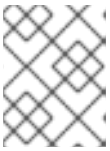
注意

请参阅配置集群范围代理将 CA 配置为信任更新服务器。

11.3.9. 后续步骤

在更新集群前，请确认满足以下条件：

- Cluster Version Operator (CVO) 被配置为使用安装的 OpenShift Update Service 应用程序。
- 新发行版本的发行镜像签名配置映射应用到集群。



注意

Cluster Version Operator (CVO) 使用发行版本镜像签名来确保发行镜像没有被修改，验证发行镜像是否与预期的结果匹配。

- 当前发行版本和更新目标发行镜像在断开连接的环境中的 registry 中被镜像到 registry。
- 最近的图形数据容器镜像已镜像到 registry。
- 安装了 OpenShift Update Service Operator 的最新版本。

注意

如果您尚未安装或更新 OpenShift Update Service Operator，则可能会有更新的版本可用。如需有关如何在断开连接的环境中更新 OLM 目录的更多信息，请参阅在断开连接的环境中使用 Operator Lifecycle Manager。

将集群配置为使用安装的 OpenShift Update Service 和本地镜像 registry 后，您可以使用以下任何更新方法：

- 使用 Web 控制台更新集群
- 使用 CLI 更新集群
- 仅执行 Control Plane 更新
- 执行 canary rollout 更新

11.4. 在没有 OPENSHIFT UPDATE SERVICE 的断开连接的环境中更新集群

使用以下步骤在断开连接的环境中更新集群，而无需访问 OpenShift Update Service。

11.4.1. 先决条件

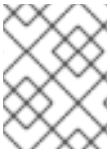
- 您必须安装了 **oc** 命令行界面（CLI）工具。
- 您必须使用容器镜像置备本地容器镜像 registry，如镜像 OpenShift Container Platform 镜像 中所述。
- 您必须可以使用具有 **admin** 权限的用户访问集群。请参阅使用 RBAC 定义和应用权限。
- 您需要具有最新的 etcd 备份，以防因为升级失败需要将集群恢复到以前的状态。
- 您已将之前通过 Operator Lifecycle Manager (OLM) 安装的所有 Operator 更新至与目标发行版本兼容的版本。更新 Operator 可确保当默认 OperatorHub 目录在集群升级过程中从当前次要版本切换到下一个次版本时，它们有有效的升级路径。如需了解如何检查兼容性以及更新已安装的 Operator 的更多信息，请参阅更新已安装的 Operator。
- 确保所有机器配置池 (MCP) 都正在运行且未暂停。在更新过程中跳过与暂停 MCP 关联的节点。如果要执行 canary rollout 更新策略，可以暂停 MCP。
- 如果您的集群使用手动维护的凭证，请更新新发行版本的云供应商资源。如需更多信息，包括如何确定这是集群的要求，请参阅准备使用手动维护的凭证更新集群。
- 如果您运行 Operator 或您已配置了 pod 中断预算，您可能会在升级过程中遇到中断。如果在 **PodDisruptionBudget** 中将 **minAvailable** 设置为 1，则节点会排空以应用可能会阻止驱除过程的待处理机器配置。如果重启了几个节点，则所有 pod 只能有一个节点上运行，**PodDisruptionBudget** 字段可能会阻止节点排空。

注意

如果您运行 Operator 或您已配置了 pod 中断预算，您可能会在升级过程中遇到中断。如果在 **PodDisruptionBudget** 中将 **minAvailable** 设置为 1，则节点会排空以应用可能会阻止驱除过程的待处理机器配置。如果重启了几个节点，则所有 pod 只能有一个节点上运行，**PodDisruptionBudget** 字段可能会阻止节点排空。

11.4.2. 暂停 MachineHealthCheck 资源

在更新过程中，集群中的节点可能会临时不可用。对于 worker 节点，**MachineHealthCheck** 资源可能会认为这样的节点不健康，并重新引导它们。为避免重新引导这样的节点，请在更新集群前暂停所有 **MachineHealthCheck** 资源。



注意

有些 **MachineHealthCheck** 资源可能不需要暂停。如果您的 **MachineHealthCheck** 资源依赖于不可恢复的条件，请暂停 MHC。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

1. 要列出您要暂停的所有可用 **MachineHealthCheck** 资源，请运行以下命令：

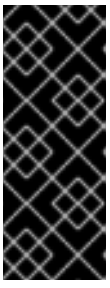
```
$ oc get machinehealthcheck -n openshift-machine-api
```

2. 要暂停机器健康检查，请将 **cluster.x-k8s.io/paused=""** 注解添加到 **MachineHealthCheck** 资源。运行以下命令:

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

注解的 **MachineHealthCheck** 资源类似以下 YAML 文件：

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



重要

更新集群后恢复机器健康检查。要恢复检查，请运行以下命令从 **MachineHealthCheck** 资源中删除暂停注解：

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

11.4.3. 检索发行镜像摘要

要使用 **oc adm upgrade** 命令和 **--to-image** 选项在断开连接的环境中更新集群，您必须引用与目标发行镜像对应的 sha256 摘要。

流程

1. 在连接到互联网的设备中运行以下命令：

```
$ oc adm release info -o 'jsonpath={.digest}{"\n"}' quay.io/openshift-release-dev/ocp-release:${OCP_RELEASE_VERSION}-${ARCHITECTURE}
```

对于 **{OCP_RELEASE_VERSION}**，请指定您要更新的 OpenShift Container Platform 版本，如 **4.10.16**。

对于 **{ARCHITECTURE}**，请指定集群的构架，如 **x86_64**, **aarch64**, **s390x**, 或 **ppc64le**。

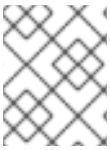
输出示例

```
sha256:a8bfba3b6dddd1a2fbbead7dac65fe4fb8335089e4e7cae327f3bad334add31d
```

2. 复制在更新集群时要使用的 sha256 摘要。

11.4.4. 更新断开连接的集群

将受限网络集群更新至您下载的发行镜像的 OpenShift Container Platform 版本。

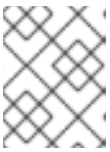


注意

如果您有一个本地 OpenShift Update Service，您可以使用连接的 Web 控制台或 CLI 指令来更新，而不是使用此流程。

先决条件

- 您已将新发行版本的镜像镜像（mirror）到 registry。
- 您已将发行镜像签名 ConfigMap 在新发行版本中应用到集群。



注意

发行镜像签名配置映射允许 Cluster Version Operator (CVO) 通过验证实际镜像签名是否与预期的签名匹配来确保发行镜像的完整性。

- 获取目标发行镜像的 sha256 摘要。
- 已安装 OpenShift CLI（**oc**）。
- 您暂停所有 **MachineHealthCheck** 资源。

流程

- 更新集群：

```
$ oc adm upgrade --allow-explicit-upgrade --to-image
<defined_registry>/<defined_repository>@<digest>
```

其中：

<defined_registry>

指定 mirror 到的镜像 registry 的名称。

<defined_repository>

指定要在镜像 registry 中使用的镜像存储库的名称。

<digest>

指定目标发行镜像的 sha256 摘要，例如 **sha256:81154f5c03294534e1eaf0319bef7a601134f891689ccede5d705ef659aa8c92**。

注意

- 请参阅"镜像 OpenShift Container Platform 镜像"以查看如何定义您的镜像 registry 和存储库名称。
- 如果您使用 **ImageContentSourcePolicy** 或 **ImageDigestMirrorSet**，您可以使用规范 registry 和存储库名称，而不是您定义的名称。规范 registry 名称为 **quay.io**，规范存储库名称为 **openshift-release-dev/ocp-release**。
- 您只能为具有 **ImageContentSourcePolicy**、**ImageDigestMirrorSet** 或 **ImageTagMirrorSet** 对象的集群配置全局 pull secret。您不能在项目中添加 pull secret。

其他资源

- 镜像 OpenShift Container Platform 镜像

11.4.5. 了解镜像 registry 仓库镜像

通过设置容器 registry 存储库镜像，您可以执行以下任务：

- 配置 OpenShift Container Platform 集群，以便重定向从源镜像 registry 上的存储库拉取（pull）镜像的请求，并通过已镜像 (mirror) 的镜像 registry 上的存储库来解决该请求。
- 为每个目标存储库识别多个已镜像 (mirror)的存储库，以确保如果一个镜像停止运作，仍可使用其他镜像。

OpenShift Container Platform 中的存储库镜像包括以下属性：

- 镜像拉取（pull）可应对 registry 停机的问题。
- 在断开连接的环境中的集群可以从关键位置（如 **quay.io**）拉取镜像，并让公司防火墙后面的 registry 提供请求的镜像。
- 发出镜像拉取（pull）请求时尝试特定 registry 顺序，通常最后才会尝试持久性 registry。
- 您所输入的镜像信息会添加到 OpenShift Container Platform 集群中每个节点上的 **/etc/containers/registries.conf** 文件中。

- 当节点从源存储库中请求镜像时，它会依次尝试每个已镜像的存储库，直到找到所请求的内容。如果所有镜像均失败，集群则会尝试源存储库。如果成功，则镜像拉取至节点中。

您可以使用以下方法设置存储库镜像：

- 在 OpenShift Container Platform 安装中：
  通过拉取（pull） OpenShift Container Platform 所需的容器镜像，然后将这些镜像放至公司防火墙后，即可将 OpenShift Container Platform 安装到受限网络中的数据中心。

- 安装 OpenShift Container Platform 后：
  如果您没有在 OpenShift Container Platform 安装过程中配置镜像，您可以在安装后使用以下自定义资源 (CR) 对象之一进行配置：

  - **ImageDigestMirrorSet** (IDMS).此对象允许您使用摘要规格从镜像 registry 中拉取镜像。IDMS CR 可让您设置回退策略，在镜像拉取失败时继续尝试从源 registry 中拉取。

  - **ImageTagMirrorSet** (ITMS)。此对象允许您使用镜像标签从已镜像的 registry 中拉取镜像。ITMS CR 可让您设置回退策略，在镜像拉取失败时继续尝试从源 registry 中拉取。

  - **ImageContentSourcePolicy** (ICSP)。此对象允许您使用摘要规格从镜像 registry 中拉取镜像。如果镜像无法正常工作，ICSP CR 始终回退到源 registry。

重要

使用 **ImageContentSourcePolicy** (ICSP)对象配置存储库镜像是一个已弃用的功能。弃用的功能仍然包含在 OpenShift Container Platform 中，并被支持。它将在以后的发行版本中删除，且不建议在新部署中使用。

如果您有用于创建 **ImageContentSourcePolicy** 对象的 YAML 文件，您可以使用 **oc adm migrate icsp** 命令将这些文件转换为 **ImageDigestMirrorSet** YAML 文件。如需更多信息，请参阅"协调 ImageContentSourcePolicy (ICSP) 文件以进行镜像 registry 存储库镜像"。

每个自定义资源对象都标识以下信息：

- 您希望镜像 (mirror) 的容器镜像存储库的源。

- 您要提供内容的每个镜像存储库的独立条目

请注意以下操作以及它们对节点排空行为的影响：

- 如果您创建一个 IDMS 或 ICSP CR 对象，MCO 不会排空或重启节点。

- 如果创建一个 ITMS CR 对象，MCO 会排空并重启节点。

- 如果您删除了 ITMS、IDMS 或 ICSP CR 对象，MCO 会排空并重启节点。

- 如果您修改 ITMS、IDMS 或 ICSP CR 对象，MCO 会排空并重启节点。

重要

- 当 MCO 检测到以下任何更改时，它会在不排空或重启节点的情况下应用更新：
  - 在机器配置的 **spec.config.passwd.users.sshAuthorizedKeys** 参数中更改 SSH 密钥。
  - 在 **openshift-config** 命名空间中更改全局 pull secret 或 pull secret。
  - Kubernetes API Server Operator 自动轮转 **/etc/kubernetes/kubelet-ca.crt** 证书颁发机构（CA）。
- 当 MCO 检测到对 **/etc/containers/registries.conf** 文件的更改时，如编辑 **ImageDigestMirrorSet**、**ImageTagMirrorSet** 或 **ImageContentSourcePolicy** 对象，它会排空对应的节点，应用更改并取消记录节点。对于以下更改，节点排空不会发生：
  - 增加了一个 registry，带有为每个镜像（mirror）设置了 **pull-from-mirror = "digest-only"** 参数。
  - 增加了一个镜像（mirror），带有在一个 registry 中设置的 **pull-from-mirror = "digest-only"** 参数。
  - 在 **unqualified-search-registries** 列表中添加项目。

对于新集群，您可以根据需要使用 IDMS、ITMS 和 ICSP CR 对象。但是，建议使用 IDMS 和 ITMS。

如果您升级了集群，则任何现有 ICSP 对象都会保持稳定，并且支持 IDMS 和 ICSP 对象。使用 ICSP 对象的工作负载可以按预期工作。但是，如果要利用 IDMS CR 中引入的回退策略，您可以使用 **oc adm migrate icsp** 命令将当前工作负载迁移到 IDMS 对象，如后面的**镜像 registry 存储库镜像**部分所示。迁移到 IDMS 对象不需要重启集群。

注意

如果您的集群使用 **ImageDigestMirrorSet**、**ImageTagMirrorSet** 或 **ImageContentSourcePolicy** 对象来配置存储库镜像，则只能使用镜像的 registry 的全局 pull secret。您不能在项目中添加 pull secret。

11.4.5.1. 配置镜像 registry 存储库镜像

您可以创建安装后镜像配置自定义资源 (CR)，将源镜像 registry 中的镜像拉取请求重定向到镜像 registry。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 通过以下方法配置已镜像的存储库：
   - 使用 Red Hat Quay 设置已镜像的存储库。您可以将镜像从一个存储库复制到另一个存储库，也可以使用 Red Hat Quay 随着时间的推移自动重复同步这些存储库。

- Red Hat Quay 存储库镜像

- 使用 **skopeo** 等工具手动将镜像从源存储库复制到已镜像的存储库。
例如，在 {op-system-base-full system} 上安装 skopeo RPM 软件包后，使用 **skopeo** 命令，如下例所示：

```
$ skopeo copy --all \
docker://registry.access.redhat.com/ubi9/ubi-minimal:latest@sha256:5cf... \
docker://example.io/example/ubi-minimal
```

在本例中，有一个名为 **example.io** 的容器镜像 registry，以及名为 **example** 的容器镜像 registry。您需要将 **ubi9/ubi-minimal** 镜像从 **registry.access.redhat.com** 复制到 **example.io**。创建已镜像的 registry 后，您可以将 OpenShift Container Platform 集群配置为将源存储库的请求重定向到已镜像的存储库。

2. 使用以下示例之一创建安装后镜像配置自定义资源(CR)：

- 根据需要，创建一个 **ImageDigestMirrorSet** 或 **ImageTagMirrorSet** CR，将源和镜像(mirror)替换为您自己的 registry、存储库对和镜像：

```
apiVersion: config.openshift.io/v1
kind: ImageDigestMirrorSet
metadata:
  name: ubi9repo
spec:
  imageDigestMirrors:
  - mirrors:
    - example.io/example/ubi-minimal
    - example.com/example2/ubi-minimal
    source: registry.access.redhat.com/ubi9/ubi-minimal
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.com/redhat
    source: registry.example.com/redhat
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.com
    source: registry.example.com
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net/image
    source: registry.example.com/example/myimage
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example
    mirrorSourcePolicy: AllowContactingSource
  - mirrors:
    - mirror.example.net/registry-example-com
    source: registry.example.com
    mirrorSourcePolicy: AllowContactingSource
```

- 创建 **ImageContentSourcePolicy** 自定义资源，将源和镜像替换为您自己的 registry、存储库对和镜像：

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

其中：

- mirror.registry.com:443/ocp/release

指定镜像 registry 和存储库的名称。

source: quay.io/openshift-release-dev/ocp-release

指定包含所镜像内容的在线 registry 和存储库。

3. 运行以下命令来创建新对象：

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

创建对象后，Machine Config Operator (MCO) 只会排空 **ImageTagMirrorSet** 对象的节点。MCO 不会排空 **ImageDigestMirrorSet** 和 **ImageContentSourcePolicy** 对象的节点。

4. 要检查是否应用了镜像的配置设置，请在其中一个节点上执行以下操作。

   a. 列出您的节点：

```
$ oc get node
```

输出示例

```
NAME                         STATUS                     ROLES    AGE  VERSION
ip-10-0-137-44.ec2.internal    Ready                        worker   7m   v1.32.3
ip-10-0-138-148.ec2.internal   Ready                         master   11m  v1.32.3
ip-10-0-139-122.ec2.internal   Ready                         master   11m  v1.32.3
ip-10-0-147-35.ec2.internal    Ready                        worker   7m   v1.32.3
ip-10-0-153-12.ec2.internal    Ready                        worker   7m   v1.32.3
ip-10-0-154-10.ec2.internal    Ready                        master   11m  v1.32.3
```

   b. 启动调试过程以访问节点：

```
$ oc debug node/ip-10-0-147-35.ec2.internal
```

输出示例

```
Starting pod/ip-10-0-147-35ec2internal-debug ...
To use host binaries, run `chroot /host`
```

c. 将您的根目录改为 **/host** ：

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

d. 检查 **/etc/containers/registries.conf** 文件，确保已完成更改：

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

以下输出代表了应用安装后镜像配置 CR 的 **registry.conf** 文件。

输出示例

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

[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi9/ubi-minimal"

  [[registry.mirror]]
    location = "example.io/example/ubi-minimal"
    pull-from-mirror = "digest-only"

  [[registry.mirror]]
    location = "example.com/example/ubi-minimal"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com"

  [[registry.mirror]]
    location = "mirror.example.net/registry-example-com"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/example"

  [[registry.mirror]]
    location = "mirror.example.net"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/example/myimage"

  [[registry.mirror]]
    location = "mirror.example.net/image"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com"
```

```
  [[registry.mirror]]
    location = "mirror.example.com"
    pull-from-mirror = "digest-only"

[[registry]]
  prefix = ""
  location = "registry.example.com/redhat"

  [[registry.mirror]]
    location = "mirror.example.com/redhat"
    pull-from-mirror = "digest-only"
[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi9/ubi-minimal"
  blocked = true

  [[registry.mirror]]
    location = "example.io/example/ubi-minimal-tag"
    pull-from-mirror = "tag-only"
```

其中：**[[registry]].location = "registry.access.redhat.com/ubi9/ubi-minimal"**:: 在一个 pull spec 中列出的存储库。**[[registry.mirror]].location = "example.io/example/ubi-minimal"**:: 表示该存储库的镜像。**[[registry.mirror]].pull-from-mirror = "digest-only"**:: 代表从 mirror 个拉取的镜像是一个摘要引用镜像。**[[registry]].blocked = true**:: 代表为这个仓库设置了 **NeverContactSource** 参数。**[[registry.mirror]].pull-from-mirror = "tag-only"**:: 代表从 mirror 个拉取的镜像是一个 tag 引用镜像。

e. 从源拉取镜像到节点，并检查是否通过 mirror 解析。

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi9/ubi-
minimal@sha256:5cf...
```

故障排除

如果存储库镜像流程未按规定工作，请使用以下有关存储库镜像如何工作的信息协助排查问题：

- 首个工作镜像用于提供拉取（pull）的镜像。
- 只有在无其他镜像工作时，才会使用主 registry。
- 从系统上下文，**Insecure** 标志用作回退。
- 最近更改了 **/etc/containers/registries.conf** 文件的格式。现在它是第 2 版，采用 TOML 格式。

11.4.5.2. 为镜像 registry 存储库镜像转换 ImageContentSourcePolicy (ICSP) 文件

使用 **ImageContentSourcePolicy** (ICSP)对象配置存储库镜像是一个已弃用的功能。

此功能仍然包含在 OpenShift Container Platform 中，并将继续被支持。但是，这个功能会在以后的发行版本中被删除，且不建议在新的部署中使用。

ICSP 对象被 **ImageDigestMirrorSet** 和 **ImageTagMirrorSet** 对象替代，以配置存储库镜像。如果您有用于创建 **ImageContentSourcePolicy** 对象的 YAML 文件，您可以使用 **oc adm migrate icsp** 命令将这些文件转换为 **ImageDigestMirrorSet** YAML 文件。命令将 API 更新至当前版本，将 **kind** 值更改为

ImageDigestMirrorSet，并将 **spec.repositoryDigestMirrors** 更改为 **spec.imageDigestMirrors**。文件的其余部分不会改变。

因为迁移不会更改 **registry.conf** 文件，所以集群不需要重启。

有关 **ImageDigestMirrorSet** 或 **ImageTagMirrorSet** 对象的更多信息，请参阅上一节中的"配置镜像 registry 存储库镜像"。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 确保集群中具有 **ImageContentSourcePolicy** 对象。

流程

1. 使用以下命令，将一个或多个 **ImageContentSourcePolicy** YAML 文件转换为 **ImageDigestMirrorSet** YAML 文件：

```
$ oc adm migrate icsp <file_name>.yaml <file_name>.yaml <file_name>.yaml --dest-dir
<path_to_the_directory>
```

其中：

<file_name>

指定源 **ImageContentSourcePolicy** YAML 的名称。您可以列出多个文件名。

--dest-dir

可选：指定输出 **ImageDigestMirrorSet** YAML 的目录。如果未设置，则会将该文件写入当前目录中。
例如，以下命令可将 **icsp.yaml** 和 **icsp-2.yaml** 文件转换，并将新的 YAML 文件保存到 **idms-files** 目录中。

```
$ oc adm migrate icsp icsp.yaml icsp-2.yaml --dest-dir idms-files
```

输出示例

```
wrote ImageDigestMirrorSet to idms-
files/imagedigestmirrorset_ubi8repo.5911620242173376087.yaml
wrote ImageDigestMirrorSet to idms-
files/imagedigestmirrorset_ubi9repo.6456931852378115011.yaml
```

2. 运行以下命令来创建 CR 对象：

```
$ oc create -f <path_to_the_directory>/<file-name>.yaml
```

其中：

<path_to_the_directory>

如果使用 **--dest-dir** 标志，请指定目录的路径。

<file_name>

指定 **ImageDigestMirrorSet** YAML 的名称。

3. 在推出 IDMS 对象后，删除 ICSP 对象。

11.4.6. 镜像镜像目录的范围，以减少集群节点重启的频率

您可以在存储库级别或更广泛的 registry 级别限定镜像目录。一个范围广泛的 **ImageContentSourcePolicy** 资源可减少节点在响应资源更改时需要重启的次数。

要强化 **ImageContentSourcePolicy** 资源中镜像目录的范围，请执行以下步骤。

先决条件

- 安装 OpenShift Container Platform CLI **oc**。
- 以具有 **cluster-admin** 特权的用户身份登录。
- 配置镜像镜像目录，以便在断开连接的集群中使用。

流程

1. 运行以下命令，为 **<local_registry>**、**<pull_spec>** 和 **<pull_secret_file>** 指定值：

```
$ oc adm catalog mirror <local_registry>/<pull_spec> <local_registry> -a <pull_secret_file> --icsp-scope=registry
```

其中：

<local_registry>
您为断开连接的集群配置的本地 registry，如 **local.registry:5000**。

<pull_spec>
是断开连接的 registry 中配置的 pull 规格，如 **redhat/redhat-operator-index:v4.19**

<pull_secret_file>
是 **.json** 文件格式的 **registry.redhat.io** pull secret。您可以从 Red Hat OpenShift Cluster Manager 下载 pull secret。

oc adm catalog mirror 命令创建 **/redhat-operator-index-manifests** 目录，并生成 **imageContentSourcePolicy.yaml**、**catalogSource.yaml** 和 **mapping.txt** 文件。

2. 将新的 **ImageContentSourcePolicy** 资源应用到集群：

```
$ oc apply -f imageContentSourcePolicy.yaml
```

验证

- 验证 **oc apply** 是否成功将更改应用到 **ImageContentSourcePolicy**:

```
$ oc get ImageContentSourcePolicy -o yaml
```

输出示例

```
apiVersion: v1
items:
- apiVersion: operator.openshift.io/v1alpha1
  kind: ImageContentSourcePolicy
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"operator.openshift.io/v1alpha1","kind":"ImageContentSourcePolicy","metadata":
{"annotations":{},"name":"redhat-operator-index"},"spec":{"repositoryDigestMirrors":
[{"mirrors":["local.registry:5000"],"source":"registry.redhat.io"}]}}
...
```

更新 **ImageContentSourcePolicy** 资源后，OpenShift Container Platform 会将新设置部署到每个节点，集群开始使用已镜像的存储库向源存储库发出请求。

11.4.7. 其他资源

- 在断开连接的环境中使用 Operator Lifecycle Manager。
- 机器配置概述

11.5. 从集群中删除 OPENSHIFT UPDATE SERVICE

要从集群中删除 OpenShift Update Service (OSUS) 的本地副本，您必须首先删除 OSUS 应用程序，然后卸载 OSUS Operator。

11.5.1. 删除 OpenShift Update Service 应用程序

您可以使用 OpenShift Container Platform Web 控制台或 CLI 删除 OpenShift Update Service 应用程序。

11.5.1.1. 使用 Web 控制台删除 OpenShift Update Service 应用程序

您可以使用 OpenShift Container Platform Web 控制台使用 OpenShift Update Service Operator 删除 OpenShift Update Service 应用程序。

先决条件

- 已安装 OpenShift Update Service Operator。

流程

1. 在 Web 控制台中，点 **Operators → Installed Operators**。
2. 从安装的 Operator 列表中选择 **OpenShift Update Service**。
3. 点 **Update Service** 选项卡。
4. 从安装的 OpenShift Update Service 应用列表中，选择要删除的应用，然后单击 **Delete UpdateService**。
5. 从 **Delete UpdateService?** 确认对话框中，单击 **Delete** 以确认删除。

11.5.1.2. 使用 CLI 删除 OpenShift Update Service 应用程序

您可以使用 OpenShift CLI（**oc**）删除 OpenShift Update Service 应用。

流程

1. 使用 OpenShift Update Service 应用程序在其中创建的命名空间获取 OpenShift Update Service 应用程序的名称，如 **openshift-update-service**：

```
$ oc get updateservice -n openshift-update-service
```

输出示例

```
NAME      AGE
service   6s
```

2. 使用上一步中的 **NAME** 值以及 OpenShift Update Service 应用程序创建命名空间删除 OpenShift Update Service 应用程序，如 **openshift-update-service**：

```
$ oc delete updateservice service -n openshift-update-service
```

输出示例

```
updateservice.updateservice.operator.openshift.io "service" deleted
```

11.5.2. 卸载 OpenShift Update Service Operator

您可以使用 OpenShift Container Platform Web 控制台或 CLI 卸载 OpenShift Update Service Operator。

11.5.2.1. 使用 Web 控制台卸载 OpenShift Update Service Operator

您可以使用 OpenShift Container Platform Web 控制台卸载 OpenShift Update Service Operator。

先决条件

- 所有 OpenShift Update Service 应用都已删除。

流程

1. 在 Web 控制台中，点 **Operators → Installed Operators**。
2. 从安装的 Operator 列表中选择 **OpenShift Update Service** 并点 **Uninstall Operator**。
3. 在 **Uninstall Operator?** 确认对话框中点 **Uninstall** 确认卸载。

11.5.2.2. 使用 CLI 卸载 OpenShift Update Service Operator

您可以使用 OpenShift CLI（**oc**）卸载 OpenShift Update Service Operator。

先决条件

- 所有 OpenShift Update Service 应用都已删除。

流程

1. 更改到包含 OpenShift Update Service Operator 的项目，如 **openshift-update-service**：

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

输出示例

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

2. 获取 OpenShift Update Service Operator operator 组的名称：

```
$ oc get operatorgroup
```

输出示例

```
NAME                              AGE
openshift-update-service-fprx2   4m41s
```

3. 删除 operator 组，如 **openshift-update-service-fprx2**：

```
$ oc delete operatorgroup openshift-update-service-fprx2
```

输出示例

```
operatorgroup.operators.coreos.com "openshift-update-service-fprx2" deleted
```

4. 获取 OpenShift Update Service Operator 订阅的名称：

```
$ oc get subscription
```

输出示例

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

5. 使用上一步中的 **Name** 值，在 **currentCSV** 字段中检查订阅的 OpenShift Update Service Operator 的当前版本：

```
$ oc get subscription update-service-operator -o yaml | grep " currentCSV"
```

输出示例

```
  currentCSV: update-service-operator.v0.0.1
```

6. 删除订阅，如 **update-service-operator**：

```
$ oc delete subscription update-service-operator
```

输出示例

```
subscription.operators.coreos.com "update-service-operator" deleted
```

7. 使用上一步中的 **currentCSV** 值删除 OpenShift Update Service Operator 的 CSV：

```
$ oc delete clusterserviceversion update-service-operator.v0.0.1
```

输出示例

```
clusterserviceversion.operators.coreos.com "update-service-operator.v0.0.1" deleted
```