OpenShift Container Platform 4.12

更新集群

更新 OpenShift Container Platform 集群

Last Updated: 2024-07-19

OpenShift Container Platform 4.12 更新集群

更新 OpenShift Container Platform 集群

法律通告

摘要

本文档提供了有关更新和升级 OpenShift Container Platform 集群的信息。更新集群的过程较简单，可以在不需要使集群离线的情况下进行。

目录

第 1 章 更新集群概述

您可以使用 Web 控制台或 OpenShift CLI (**oc**) 使用单一操作来更新 OpenShift Container Platform 4 集群。

1.1. 了解 OPENSHIFT CONTAINER PLATFORM 更新

关于 OpenShift Update Service ：对于可访问互联网的集群，红帽通过 OpenShift Container Platform 更新服务提供更新，它作为公共 API 后面的一个托管服务运行。

1.2. 了解更新频道和发行版本

更新频道和发行版本 ：使用更新频道，您可以选择更新策略。更新频道特定于 OpenShift Container Platform 的次要版本。更新频道只控制发行版本选择，不会影响您安装的集群版本。OpenShift Container Platform 的一个特定版本的 **openshift-install** 二进制文件总会安装该次版本。如需更多信息，请参阅以下：

- 升级版本路径
- fast 和 stable 频道的使用和策略
- 了解受限网络集群
- 在频道间切换
- 了解条件更新

1.3. 了解集群 OPERATOR 条件类型

集群 Operator 的状态包括它们的 condition 类型，它告知您 Operator 的健康状况的当前状态。以下定义涵盖了一些常见 ClusterOperator 条件类型的列表。省略了具有额外条件类型和特定 Operator 语言的 Operator。

Cluster Version Operator (CVO) 负责从集群 Operator 收集状态条件，以便集群管理员可以更好地了解 OpenShift Container Platform 集群的状态。

- available: 条件类型 **Available** 表示 Operator 功能且在集群中可用。如果状态是 **False**，则操作对象中的至少一个部分无法正常工作，并且条件要求管理员干预。
- progressing: 条件类型 **Progressing** 表示 Operator 正在主动推出新的代码、传播配置更改，或者从一个稳定状态移到另一个状态。
  当 Operator 协调之前已知状态时，Operator 不会报告条件类型 **Progressing** 为 **True**。如果观察到的集群状态已更改，且 Operator 会响应它，则状态将报告为 **True**，因为它从一个 steady 状态移到另一个状态。
- Degraded：条件类型 **Degraded** 表示 Operator 具有在一段时间内不匹配其所需状态的当前状态。周期可能会因组件而异，但 **Degraded** 状态代表 Operator 条件的持久性观察。因此，Operator 不会波动处于 Degraded 状态和没有处于 **Degraded** 状态。
  如果从一个状态转换到另一个状态的过渡在长时间内没有保留，则可能会有一个不同的条件类型来报告 **Degraded**。Operator 在正常更新过程中不会报告 **Degraded**。Operator 可能会报告 **Degraded**，以响应需要最终管理员干预的持久性基础架构失败。

注意

此条件类型仅表示可能需要调查和调整某项。只要 Operator 可用，**Degraded** 条件就不会造成用户工作负载失败或应用程序停机。

- Upgradeable: 条件类型 **Upgradeable** 表示 Operator 是否根据当前的集群状态安全更新。message 字段包含管理员对集群成功更新需要执行的操作的人类可读描述。当此条件为 **True**、**Unknown** 或缺失时，CVO 允许更新。
  当 **Upgradeable** 状态为 **False** 时，只有次版本更新会受到影响，CVO 会阻止集群执行受影响的更新，除非强制（强制）更新。

1.4. 了解集群版本状况类型

Cluster Version Operator (CVO) 监控集群 Operator 和其他组件，并负责收集集群版本及其 Operator 的状态。此状态包括条件类型，它告知您 OpenShift Container Platform 集群的健康状态和当前状态。

除了 **Available**,**Progressing**, 和 **Upgradeable** 外，还有影响集群版本和 Operator 的条件类型。

- Failing：集群版本状况类型 **Failing** 表示集群无法访问其所需状态，不健康，需要管理员干预。
- Invalid: 集群版本条件类型 **Invalid** 表示集群版本具有阻止服务器执行操作的错误。只要设置了此条件，CVO 仅协调当前状态。
- RetrievedUpdates ：集群版本条件类型 **RetrievedUpdates** 表示已从上游更新服务器检索可用更新。在检索前条件为 **Unknown**，如果更新最近失败或无法检索，则为 **False**；如果 **availableUpdates** 字段是最新且准确的，则为 **True**。
- ReleaseAccepted ：集群版本状况类型 **ReleaseAccepted**，并带有 **True** 状态表示请求的发行版本有效负载在镜像验证和预条件检查过程中没有失败。
- ImplicitlyEnabledCapabilities ：集群版本条件类型 **ImplicitlyEnabledCapabilities** 具有 **True** 状态，表示用户目前没有通过 **spec.capabilities** 请求的功能。如果任何相关资源之前由 CVO 管理，CVO 不支持禁用功能。

1.5. 准备执行 EUS 更新

准备执行 EUS 到 EUS 更新 ：由于 Kubernetes 的设计，次版本之间的所有 OpenShift Container Platform 升级都必须按顺序进行。您必须从 OpenShift Container Platform 4.10 更新至 4.11，然后再更新至 4.12。您无法直接从 OpenShift Container Platform 4.10 更新至 4.12。但是，如果您想要在两个延长更新支持(EUS)版本之间更新，您可以只发生一次重启非控制平面主机。如需更多信息，请参阅以下：

- 更新 EUS 到 EUS

1.6. 使用 WEB 控制台更新集群

使用 Web 控制台更新集群：您可以使用 Web 控制台更新 OpenShift Container Platform 集群。以下步骤在次版本中更新集群。您可以使用相同的说明在次版本之间更新集群。

- 执行 canary rollout 更新
- 暂停 MachineHealthCheck 资源
- 关于在单节点集群中更新 OpenShift Container Platform
- 使用Web控制台更新集群

- 使用 Web 控制台更改更新服务器

1.7. 使用 CLI 更新集群

使用 CLI 更新集群 ：您可以使用 OpenShift CLI (**oc**) 更新 OpenShift Container Platform 集群。以下步骤在次版本中更新集群。您可以使用相同的说明在次版本之间更新集群。

- 暂停 MachineHealthCheck 资源
- 关于在单节点集群中更新 OpenShift Container Platform
- 使用 CLI 更新集群
- 使用 CLI 更改更新服务器

1.8. 执行 CANARY ROLLOUT 更新

执行 Canary 推出部署更新 ： 通过控制对 worker 节点的更新，您可以确保关键任务应用程序在整个更新过程中仍然可用，即使更新过程会导致应用程序失败。根据您的机构需求，您可能需要更新一小部分 worker 节点，在一个时间段内评估集群和工作负载健康状况，然后更新剩余的节点。这称为 *Canary* 更新。或者，您可能还希望将通常需要主机重新引导的 worker 节点更新放入较小的定义的维护窗口（不可能一次使用大型维护窗口来更新整个集群）。您可以执行以下步骤：

- 创建机器配置池来执行 Canary rollout 更新
- 暂停机器配置池
- 执行集群更新
- 取消暂停机器配置池
- 将节点移动到原始机器配置池

1.9. 更新包含使用 RHEL 的计算（COMPUTE）系统的集群

更新包含 RHEL 计算机器的集群 ：如果集群包含 Red Hat Enterprise Linux (RHEL) 机器，则必须执行额外的步骤来更新这些机器。您可以执行以下步骤：

- 使用Web控制台更新集群
- 可选：添加 hook 以在RHEL系统上执行Ansible任务
- 更新集群中的RHEL compute 系统

1.10. 在断开连接的环境中更新集群

在断开连接的环境中的集群更新：如果镜像主机无法同时访问互联网和集群，您可以将镜像镜像到与环境断开连接的文件系统中。然后您可以使用那个主机或可移动介质来实现安装。如果本地容器 registry 和集群连接到镜像 registry 的主机，您可以直接将发行镜像推送到本地 registry。

- 准备您的镜像主机
- 配置允许对容器镜像进行镜像的凭证
- 镜像 OpenShift Container Platform 镜像存储库

- 更新断开连接的集群
- 配置镜像 registry 存储库镜像
- 镜像镜像目录的范围，以减少集群节点重启的频率
- 安装 OpenShift Update Service Operator
- 创建 OpenShift Update Service 应用程序
- 删除 OpenShift Update Service 应用程序
- 卸载 OpenShift Update Service Operator

1.11. 更新在 VSPHERE 上运行的节点上运行的硬件

更新 vSphere 上的硬件 ：您必须确保在 OpenShift Container Platform 支持的硬件版本中运行 vSphere 中运行的节点。目前，硬件版本 15 或更高版本都支持集群中的 vSphere 虚拟机。如需更多信息，请参阅以下：

- 更新 vSphere 上的虚拟硬件
- 在 vSphere 上调度虚拟硬件的更新

重要

OpenShift Container Platform 版本 4.12 需要 VMware 虚拟硬件版本 15 或更高版本。

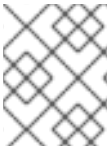
第 2 章 了解 OPENSHIFT 更新

2.1. OPENSHIFT 更新简介

在 OpenShift Container Platform 4 中，您可以使用 Web 控制台或 OpenShift CLI (**oc**) 使用单一操作来更新 OpenShift Container Platform 集群。平台管理员可以通过转至 web 控制台中的 **Administration → Cluster Settings** 或查看 **oc adm upgrade** 命令的输出来查看新的更新选项。

红帽托管了一个公共 OpenShift Update Service (OSUS)，它根据官方 registry 中的 OpenShift Container Platform 发行镜像提供更新可能性图。图包含任何公共 OCP 版本的更新信息。OpenShift Container Platform 集群默认配置为连接到 OSUS，OSUS 会使用已知更新目标的信息响应集群。

当集群管理员或自动更新控制器使用新版本编辑 Cluster Version Operator (CVO) 的自定义资源 (CR) 时，更新开始。要将集群与新指定版本协调，CVO 从镜像 registry 检索目标发行镜像，并开始将更改应用到集群。



注意

之前通过 Operator Lifecycle Manager (OLM) 安装的 Operator 会遵循不同的更新过程。如需更多信息，请参阅更新安装的 Operator。

目标发行镜像包含组成特定 OCP 版本的所有集群组件的清单文件。当将集群更新至新版本时，CVO 会在称为 Runlevels 的独立阶段应用清单。大多数（但不是全部清单）支持其中一个集群 Operator。当 CVO 将清单应用到集群 Operator 时，Operator 可能会执行更新任务将其与新的指定版本协调。

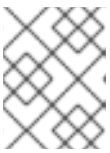
CVO 监控每个应用的资源的状态，以及所有集群 Operator 报告的状态。只有活跃 Runlevel 中的所有清单和集群 Operator 都达到稳定条件时，CVO 才会继续更新。在 CVO 通过此过程更新整个 control plane 后，Machine Config Operator (MCO) 会更新集群中每个节点的操作系统和配置。

2.1.1. 有关更新可用性的常见问题

OpenShift Container Platform 集群使用更新时，有几个因素会影响到 OpenShift Container Platform 集群。以下列表提供有关更新可用性的常见问题：

每个更新频道之间有什么区别？

- 一个新的发行版本最初添加到 **candidate** 频道中。
- 在成功测试后，**candidate** 频道的发行版本将提升到 **fast** 频道，则会发布勘误，并完全支持该发行版本。
- 延迟后，**fast** 频道中的一个发行版本最终会提升到 **stable** 频道。这个延迟代表了 **fast** 和 **stable** 频道之间的唯一区别。



注意

对于最新的 z-stream 版本，这个延迟通常是一周或两周。但是，初始更新到最新次版本的延迟可能需要更长的时间，通常为 45-90 天。

- 提升到 **stable** 频道的版本同时提升到 **eus** 频道。**eus** 频道的主要目的是，为了方便执行 EUS 到 EUS 更新的集群。

stable 频道安全或大于 fast 频道中的一个发行版本吗？

- 如果在 **fast** 频道中为发行版本发现了回归问题，它将被解析为与 **stable** 频道中发布的回归问题相同的扩展。
- **fast** 和 **stable** 频道中发行版本的唯一区别在于，一个发行版本仅会在出现在 **fast** 频道一段时间后才会出现在 **stable** 频道中，这样做可以有更长的时间来发行在更新中可能存在的风险。
- 在这个延迟后，**fast** 频道中可用的发行版本始终在 **stable** 频道中可用。

如果一个更新被支持但不推荐使用意味着什么？

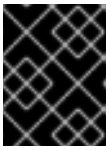
- 红帽会持续评估来自多个源的数据，以确定从一个版本更新到另一个版本是否会导致问题。如果确定了问题，用户可能不再建议更新路径。但是，即使不推荐更新路径，如果客户执行了更新，仍然被支持。
- 红帽不会阻止用户升级到特定版本。红帽可能会声明条件更新风险，这些风险可能不适用于特定集群。
    - 声明的风险提供有关受支持更新的更多上下文。集群管理员仍可接受该特定目标版本的风险和更新。虽然在条件风险上下文中不推荐使用这个更新。

如果特定版本的更新不再被推荐意味着什么？

- 如果因为回归的问题，红帽从任何支持的发行版本中删除更新建议，则会为更正回归的未来版本提供取代的更新建议。当缺陷被修正、测试并提升到您选择的频道时，可能会有延迟。

什么时候下一个 z-stream 版本会在 fast 和 stable 频道中出现？

- 虽然特定节奏可能会因多个因素而异，但对最新次版本的新 z-stream 版本通常会每周提供。随着时间推移，旧的次版本变得更稳定，可能需要更长的时间才提供新的 z-stream 版本。



重要

它们仅根据 z-stream 版本的相关数据进行估算。红帽保留根据需要更改发行频率的权利。任何数量的问题都可能导致此发行节奏出现异常和延迟。

- 发布 z-stream 版本后，它也会出现在该次版本的 **fast** 频道中。延迟后，z-stream 版本可能会出现在该次版本的 **stable** 频道中。

其他资源

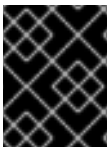
- 了解更新频道和发行版本

2.1.2. 关于 OpenShift Update 服务

OpenShift Update Service (OSUS) 为 OpenShift Container Platform 提供推荐的更新，包括 Red Hat Enterprise Linux CoreOS (RHCOS)。它提供了一个图表，其中包含组件 Operator 的*顶点（vertices）*和连接它们的 *边（edges）*。图中的边代表了您可以安全更新到的版本。顶点是更新的有效负载，用于指定受管集群组件的预期状态。

集群中的 Cluster Version Operator (CVO) 会检查 OpenShift Container Platform 更新服务，并根据当前组件版本和图中的信息决定有效的更新和更新路径。当您请求更新时，CVO 使用对应的发行镜像来更新集群。发行工件 (artifact) 作为容器镜像托管在 Quay 中。

为了让 OpenShift Update Service 仅提供兼容的更新，可以使用一个版本验证管道来驱动自动化过程。每个发行工件都会被验证是否与支持的云平台和系统架构以及其他组件包兼容。在管道确认有适用的版本后，OpenShift Update Service 会通知您它可用。



重要

OpenShift Update Service 显示当前集群的所有推荐更新。如果 OpenShift Update Service 不建议升级路径，这可能是因为更新或目标发行版本存在已知问题。

两个控制器在持续更新模式下运行。第一个控制器持续更新有效负载清单，将清单应用到集群，并输出 Operator 的受控推出的状态，以指示它们是否处于可用、升级或失败状态。第二个控制器轮询 OpenShift Update Service，以确定更新是否可用。

重要

仅支持更新到较新的版本。不支持将集群还原或回滚到以前的版本。如果您的更新失败，请联系红帽支持。

在更新过程中，Machine Config Operator(MCO)将新配置应用到集群机器。MCO 会处理由 **maxUnavailable** 字段指定的、协调机器配置池中的节点数量，并将它们标记为不可用。在默认情况下，这个值被设置为 **1**。MCO 根据 **topology.kubernetes.io/zone** 标签，按区字母更新受影响的节点。如果一个区域有多个节点，则首先更新最旧的节点。对于不使用区的节点，如裸机部署中的节点，节点会按使用的时间更新，首先更新最旧的节点。MCO 一次更新机器配置池中由 **maxUnavailable** 字段指定的节点数量。然后，MCO 会应用新配置并重启机器。

如果您将 Red Hat Enterprise Linux (RHEL) 机器用作 worker，MCO 不会在这些机器上更新 kubelet，因为您必须首先在这些机器上更新 OpenShift API。

当新版本规格应用到旧的 kubelet 时，RHEL 机器无法返回 **Ready** 状态。在机器可用前，您无法完成更新。但是，因为已设置了最大不可用节点数，所以可以在一定机器无法使用的情况下，确保正常的集群操作。

OpenShift Update Service 由 Operator 和一个或多个应用程序实例组成。

2.1.3. 常见术语

Control plane（控制平面）

control plane 由 control plane 机器组成，负责管理 OpenShift Container Platform 集群。control plane 机器管理计算机器（也被称为 worker）上的工作负载。

Cluster Version Operator

Cluster Version Operator (CVO)启动集群的更新过程。它根据当前的集群版本检查 OSUS，并检索包含可用或可能的更新路径的图形。

Machine Config Operator

Machine Config Operator (MCO)是一个集群级别的 Operator，用于管理操作系统和机器配置。通过 MCO，平台管理员可以配置和更新 worker 节点上的 systemd、CRI-O 和 Kubelet、内核、NetworkManager 和其他系统功能。

OpenShift 更新服务

OpenShift Update Service (OSUS)为 OpenShift Container Platform 提供无线更新，包括 Red Hat Enterprise Linux CoreOS(RHCOS)。它提供了一个图形或图表，其中包含组件 Operator 的顶点和连接它们的边。

Channels

Channels 声明了一个与 OpenShift Container Platform 次版本相关的更新策略。OSUS 使用这个配置的策略来推荐与该策略一致更新边缘。

推荐的更新边缘

推荐的更新边缘 是 OpenShift Container Platform 发行版本之间的建议更新。建议使用给定的更新，具体取决于集群配置的频道、当前版本、已知的错误和其他信息。OSUS 将建议的边缘与 CVO 通信，后者在每个集群中运行。

延长更新支持

所有 post-4.7 甚至编号的次版本都标记为 *延长更新支持* (EUS)版本。这些版本包括了在 EUS 版本之间更轻松地更新路径，可以简化 worker 节点的更新，并可以通过规划 EUS-to-EUS OpenShift Container Platform 版本的更新策略来减少重启 worker 节点的更新。
如需更多信息，请参阅 Red Hat OpenShift Extended Update Support(EUS)概述 。

其他资源

- 机器配置概述
- 在断开连接的环境中使用 OpenShift Update Service
- 更新频道

2.1.4. 其他资源

- 有关更新过程的每个主要方面的详情，请参考集群如何更新工作。

2.2. 集群更新如何工作

以下小节详细介绍了 OpenShift Container Platform (OCP) 更新过程的每个主要方面。有关更新如何工作的一般信息，请参阅 OpenShift 更新简介。

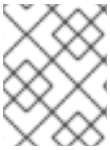
2.2.1. Cluster Version Operator

Cluster Version Operator (CVO) 是编配并协助 OpenShift Container Platform 更新过程的主要组件。在安装和标准集群操作过程中，CVO 会持续将受管集群 Operator 的清单与集群资源中的清单进行比较，并协调差异以确保这些资源的实际状态与所需状态匹配。

2.2.1.1. ClusterVersion 对象

Cluster Version Operator (CVO) 监控的资源之一是 **ClusterVersion** 资源。

管理员和 OpenShift 组件可以通过 **ClusterVersion** 对象与 CVO 通信或交互。所需的 CVO 状态通过 **ClusterVersion** 对象声明，当前 CVO 状态反映在对象的状态中。



注意

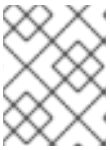
不要直接修改 **ClusterVersion** 对象。反之，使用 **oc** CLI 或 Web 控制台等接口来声明您的更新目标。

CVO 持续将集群与 **ClusterVersion** 资源的 **spec** 属性中声明的目标状态进行协调。当所需的发行版本与实际发行版本不同时，协调会更新集群。

更新可用性数据
ClusterVersion 资源还包含集群可用的更新信息。这包括可用更新，但不推荐因为应用到集群的已知风险而不推荐。这些更新称为条件更新。要了解 CVO 如何在 **ClusterVersion** 资源中维护此信息，请参阅"更新可用性评估"部分。

- 您可以使用以下命令检查所有可用更新：

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



注意

额外的 **--include-not-recommended** 参数包括可用的更新，但不推荐因为应用到集群的已知风险而不建议这样做。

输出示例

```
Cluster version is 4.10.22

Upstream is unset, so the cluster will use an appropriate default.
Channel: fast-4.11 (available channels: candidate-4.10, candidate-4.11, eus-4.10, fast-4.10, fast-4.11, stable-4.10)

Recommended updates:

  VERSION     IMAGE
  4.10.26     quay.io/openshift-release-dev/ocp-release@sha256:e1fa1f513068082d97d78be643c369398b0e6820afab708d26acda2262940954
  4.10.25     quay.io/openshift-release-dev/ocp-release@sha256:ed84fb3fbe026b3bbb4a2637ddd874452ac49c6ead1e15675f257e28664879cc
  4.10.24     quay.io/openshift-release-dev/ocp-release@sha256:aab51636460b5a9757b736a29bc92ada6e6e6282e46b06e6fd483063d590d62a
  4.10.23     quay.io/openshift-release-dev/ocp-release@sha256:e40e49d722cb36a95fa1c03002942b967ccbd7d68de10e003f0baa69abad457b

Supported but not recommended updates:

  Version: 4.11.0
  Image: quay.io/openshift-release-dev/ocp-release@sha256:300bce8246cf880e792e106607925de0a404484637627edf5f517375517d54a4
  Recommended: False
  Reason: RPMOSTreeTimeout
  Message: Nodes with substantial numbers of containers and CPU contention may not reconcile machine configuration https://bugzilla.redhat.com/show_bug.cgi?id=2111817#c22
```

oc adm upgrade 命令查询 **ClusterVersion** 资源以获取可用更新的信息，并以人类可读格式显示它。

- 直接检查 CVO 创建的底层可用性数据的一种方法是，使用以下命令查询 **ClusterVersion** 资源：

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

输出示例

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

- 类似的命令可用于检查条件更新：

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

输出示例

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

2.2.1.2. 更新可用性的评估

Cluster Version Operator (CVO) 会定期查询 OpenShift Update Service (OSUS)，以获取与更新可能相关的最新数据。这个数据基于集群的订阅频道。然后，CVO 将有关更新建议的信息保存到它的 **ClusterVersion** 资源的 **availableUpdates**或r **conditionalUpdates** 字段中。

CVO 定期检查条件更新以更新风险。这些风险通过 OSUS 提供的数据传递，其中包含可能会影响到该版本的集群更新的已知问题的每个版本的信息。大多数风险仅限于具有特定特征的集群，如在特定云平台中部署的特定大小或集群的集群。

CVO 根据每个条件更新的条件风险信息持续评估其集群特征。如果 CVO 发现集群与条件匹配，CVO 会将此信息存储在 **ClusterVersion** 资源的 **conditionalUpdates** 字段中。如果 CVO 发现集群与更新的风险不匹配，或者没有与更新相关的风险，它会将目标版本存储在 **ClusterVersion** 资源的 **availableUpdates** 字段中。

用户界面，Web 控制台或 OpenShift CLI (**oc**)，在管理员的部分标题中显示此信息。每个**支持但不推荐**的更新建议都包含有关风险的进一步资源的链接，以便管理员可以就更新做出明智的决定。

其他资源

- 更新建议删除和升级

2.2.2. 发行镜像

发行镜像是特定 OpenShift Container Platform (OCP) 版本的交付机制。它包含发行版本元数据、与发行版本匹配的 Cluster Version Operator (CVO) 二进制文件、部署单个 OpenShift 集群 Operator 所需的每个清单，以及对组成此 OpenShift 版本的所有容器镜像的 SHA 摘要版本引用列表。

您可以运行以下命令来检查特定发行镜像的内容：

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

输出示例

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

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

1. **ClusterResourceQuota** CRD 的清单，用于 Runlevel 03
2. **service-ca-operator** 的 **PrometheusRoleBinding** 资源清单，应用于 Runlevel 90

3 对所有所需镜像的 SHA 摘要版本引用列表

2.2.3. 更新过程工作流

以下步骤代表了 OpenShift Container Platform (OCP) 更新过程的详细工作流：

1. 目标版本存储在 **ClusterVersion** 资源的 **spec.desiredUpdate.version** 字段中，该字段可通过 Web 控制台或 CLI 进行管理。
2. Cluster Version Operator (CVO) 检测到 **ClusterVersion** 资源中的 **desiredUpdate** 与当前集群版本不同。使用 OpenShift Update Service 中的图形数据，CVO 将所需的集群版本解析为发行镜像的 pull spec。
3. CVO 验证发行镜像的完整性和真实性。红帽通过使用镜像 SHA 摘要作为唯一和不可变发行镜像标识符，发布有关在预定义位置发布的发行镜像的加密签名的声明。CVO 使用内置公钥列表来验证与检查的发行镜像匹配的声明是否存在和签名。
4. CVO 在 **openshift-cluster-version** 命名空间中创建一个名为 **version-$version-$hash** 的作业。此作业使用执行发行镜像的容器，因此集群通过容器运行时下载镜像。然后，作业会将清单和元数据从发行镜像提取到 CVO 访问的共享卷。
5. CVO 验证提取的清单和元数据。
6. CVO 检查一些 preconditions，以确保集群中没有检测到有问题的条件。某些条件可能会阻止更新进行。这些条件可以由 CVO 本身决定，或由单个集群 Operator 报告，用于检测 Operator 认为更新问题的一些集群详情。
7. CVO 以 **status.desired** 记录接受的发行版本，并创建一个有关新更新的 **status.history** 条目。
8. CVO 开始协调来自发行镜像的清单。集群 Operator 在称为 Runlevels 的独立阶段更新，CVO 确保 Runlevel 中的所有 Operator 在进入下一级别前完成更新。
9. CVO 本身的清单会在进程早期应用。应用 CVO 部署时，当前的 CVO pod 会停止，并使用新版本启动的 CVO pod。新的 CVO 继续协调剩余的清单。
10. 更新会进行，直到整个 control plane 更新至新版本。单个集群 Operator 可能会在集群域中执行更新任务，但它们通过 **Progressing=True** 条件报告其状态。
11. Machine Config Operator (MCO) 清单应用到进程末尾。然后，更新的 MCO 开始更新每个节点的系统配置和操作系统。每个节点可能会在开始重新接受工作负载前排空、更新和重启。

在 control plane 更新完成后，集群会报告（在更新所有节点之前）。更新后，CVO 维护所有集群资源以匹配发行镜像中交付的状态。

2.2.4. 了解更新期间如何应用清单

因为它们之间的依赖项，发行镜像中提供的一些清单必须按特定顺序应用。例如，必须在匹配的自定义资源前创建 **CustomResourceDefinition** 资源。另外，还需要更新单个集群 Operator 的逻辑顺序，以便最大程度降低集群中的中断。Cluster Version Operator (CVO) 通过运行级别 (runlevel) 的概念来实施这个逻辑顺序。

这些依赖项在发行镜像中清单的文件名中编码：

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

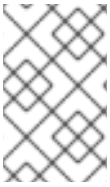
例如：

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

CVO 在内部为清单构建依赖项图，其中 CVO 遵循以下规则：

- 在更新过程中，在较高运行级别的清单之前会应用较低运行级别的清单。
- 在一个运行级别中，可以并行应用不同组件的清单。
- 在一个运行级别中，单个组件的清单以字典顺序应用。

然后，CVO 按照生成的依赖项图应用清单。



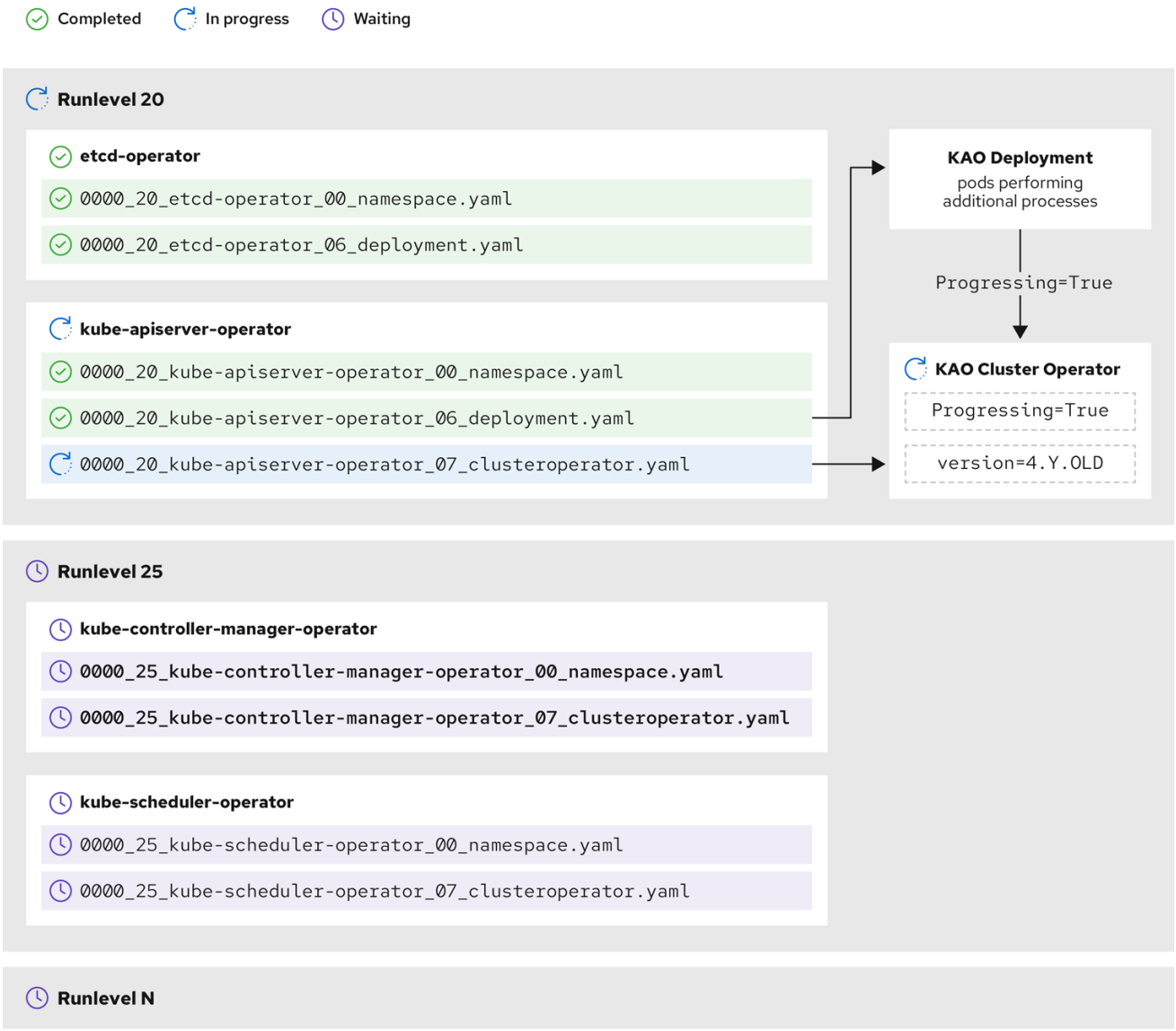
注意

对于某些资源类型，CVO 在应用清单后监控资源，并将其视为仅在资源达到稳定状态后成功更新。实现此状态可能需要一些时间。对于 **ClusterOperator** 资源，这尤其如此，而 CVO 会等待集群 Operator 更新其自身，然后更新其 **ClusterOperator** 状态。

CVO 在进入下一个运行级别前等待到 Runlevel 中的所有集群 Operator 满足以下条件：

- 集群 Operator 有一个 **Available=True** 条件。
- 集群 Operator 有一个 **Degraded=False** 条件。
- 集群 Operator 声明它们已在 ClusterOperator 资源中实现所需的版本。

有些操作可能需要大量时间来完成。CVO 等待操作完成，以确保后续运行级别可以安全继续。初始协调新发行版本的清单预计需要 60 到 120 分钟。请参阅**了解 OpenShift Container Platform 更新持续时间**以了解更多有关影响更新持续时间的信息。



在上例中，CVO 会等待所有工作在 Runlevel 20 中完成。CVO 将所有清单应用到 Runlevel 中的 Operator，但 **kube-apiserver-operator ClusterOperator** 在部署了新版本后执行一些操作。**kube-apiserver-operator ClusterOperator** 通过 **Progressing=True** 条件声明此进度，且没有在 **status.versions** 中声明新版本作为协调。CVO 等待 ClusterOperator 报告可接受的状态，然后在 Runlevel 25 开始协调清单。

其他资源

- 了解 OpenShift Container Platform 更新持续时间

2.2.5. 了解 Machine Config Operator 如何更新节点

Machine Config Operator (MCO) 将新机器配置应用到每个 control plane 节点和计算节点。在机器配置更新过程中，control plane 节点和计算节点被组织到自己的机器配置池中，其中机器池会并行更新。**.spec.maxUnavailable** 参数（默认值为 **1**）决定机器配置池中可以同时处理更新过程中的节点数。

当机器配置更新过程启动时，MCO 会检查池中当前不可用的节点数量。如果不可用的节点小于 **.spec.maxUnavailable** 的值，MCO 会在池中对可用节点启动以下操作序列：

1. cordon 和 drain 节点

注意

当节点被封锁时，工作负载无法调度到其中。

2. 更新节点的系统配置和操作系统 (OS)
3. 重新引导节点
4. 取消协调节点

一个节点无法处理这个过程，直到它被取消封锁，且工作负载可以再次调度到其中。MCO 开始更新节点，直到不可用节点的数量等于 **.spec.maxUnavailable** 的值。

当节点完成其更新并变为可用时，机器配置池中不可用的节点数量会重新小于 **.spec.maxUnavailable**。如果需要更新剩余的节点，MCO 会在节点上启动更新过程，直到再次达到 **.spec.maxUnavailable** 限制为止。此过程会重复，直到每个 control plane 节点和计算节点已更新。

以下示例工作流描述了这个过程在 5 个节点的机器配置池中如何发生，其中 **.spec.maxUnavailable** 为 3，所有节点最初可用：

1. MCO 会处理节点 1、2 和 3，并开始排空它们。
2. 节点 2 完成排空、重启并再次可用。MCO 对节点 4 进行 cordons，并开始排空节点。
3. 节点 1 完成排空、重新引导并再次可用。MCO 对节点 5 进行 cordons，并开始排空节点。
4. 节点 3 完成排空、重启并再次可用。
5. 节点 5 完成排空、重启并再次可用。
6. 节点 4 完成排空、重启并再次可用。

因为每个节点的更新过程独立于其他节点，所以上例中的一些节点会完全完成它们的更新顺序，由 MCO 封锁。

您可以运行以下命令来检查机器配置更新的状态：

```
$ oc get mcp
```

输出示例

```
NAME      CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master    rendered-master-acd1358917e9f98cbdb599aea622d78b    True      False      False      3
3              3                   0                   22h
worker    rendered-worker-1d871ac76e1951d32b2fe92369879826    False     True       False      2
1              1                   0                   22h
```

其他资源

- 机器配置概述

第 3 章 了解更新频道和发行版本

更新频道是一个升级机制，用户可以声明他们要更新集群的 OpenShift Container Platform 次版本。它们还允许用户选择更新更新的时间和级别，并通过 **fast**、**stable**、**candidate** 和 **eus** 频道选项。Cluster Version Operator 使用基于频道声明的更新图以及其他条件信息，以提供集群可用的推荐和条件更新列表。

升级频道与 OpenShift Container Platform 的次版本关联。频道中的版本号代表集群最终要升级到的目标次版本，即使它高于集群的当前次版本。

例如，OpenShift Container Platform 4.10 更新频道提供以下建议：

- 在 4.10 内更新。
- 4.9 中的更新。
- 从 4.9 升级到 4.10，允许所有 4.9 集群最终更新至 4.10，即使它们没有立即满足最小 z-stream 版本要求。
- 仅限 **eus-4.10**：在 4.8 中更新。
- 仅限 **eus-4.10**: 从 4.8 升级到 4.9 再到 4.10,，允许所有 4.8 集群最终更新至 4.10。

4.10 更新频道不推荐对 4.11 或更高版本的更新。这可确保管理员明确决定升级到下一个 OpenShift Container Platform 次版本。

更新频道只控制发行版本选择，不会影响您安装的集群版本。特定版本的 OpenShift Container Platform 的 **openshift-install** 二进制文件始终会安装该版本。

OpenShift Container Platform 4.12 提供了以下更新频道：

- **stable-4.12**
- **eus-4.y**（只提供 EUS 版本，旨在促进 EUS 版本之间的更新）
- **fast-4.12**
- **candidate-4.12**

如果您不希望 Cluster Version Operator 从升级建议服务获取可用的更新，您可以使用 OpenShift CLI 中的 **oc adm upgrade channel** 命令配置空频道。例如，当集群有受限网络访问且没有本地可访问的升级建议服务时，这个配置很有用。

警告

红帽建议升级到 OpenShift Update Service 建议的版本。对于次版本更新，版本必须是连续的。红帽没有测试在非连续地版本间的升级，无法保证与之前版本的兼容性。

3.1. 更新频道

3.1.1. fast-4.12 频道

当红帽声明版本成为正式发行 (GA) 版本时，**fast-4.12** 频道被更新为 OpenShift Container Platform 4.12 的新版本。因此，这些版本被完全支持，用于生产环境。

3.1.2. stable-4.12 频道

虽然当它们的勘误被发布后马上就会出现在 **fast-4.12** 频道中，但这些内容可能需要一段延迟时间会被添加到 **stable-4.12** 频道中。在这个延迟过程中，会从多个源收集数据并分析用于指示产品回归。收集大量数据点后，这些版本将添加到 stable 频道中。

注意

由于获得大量的数据点所需的时间因很多因素而异，因此在快速频道和稳定频道之间的延迟期间不会提供 Service LeveL Objective (SLO)。如需更多信息，请参阅"选择集群的正确频道"

新安装的集群默认为使用 stable 频道。

3.1.3. eus-4.y 频道

除了 stable 频道外，所有以数字相等的 OpenShift Container Platform 次版本都提供延长更新支持 (EUS)。提升到 stable 频道的版本也同时提升到 EUS 频道。EUS 频道的主要目的是，为了方便执行 EUS 到 EUS 更新的集群。

注意

标准和非 EUS 订阅者都可以访问所有 EUS 软件仓库和所需的 RPM(**rhel-*-eus-rpms**)，它们都能够支持关键目的，如调试和构建驱动程序。

3.1.4. candidate-4.12 频道

candidate-4.12 频道在构建后马上提供对这个版本的早期访问。只有候选频道中出现的版本可能不包含在 GA 之前删除最终 GA 版本或功能的完整功能集。另外，这些版本没有受到红帽质量保证的约束，可能不会为以后的 GA 版本提供更新路径。鉴于这些注意事项，候选通道仅适用于销毁和重新创建集群可接受的目的。

3.1.5. 更新频道中的建议

OpenShift Container Platform 维护一个更新建议服务，它知道已安装的 OpenShift Container Platform 版本以及频道中要获取的路径，以便您获得下一版本。更新路径还仅限于与当前所选频道及其提升特征相关的版本。

您可在频道中看到以下发行版本：

- 4.12.0
- 4.12.1
- 4.12.3
- 4.12.4

该服务只建议经过测试且没有已知的严重回归更新。例如，如果您的集群为 4.12.1，OpenShift Container Platform 推荐 4.12.4，建议从 4.12.1 升级到 4.12.4。

重要

您不需要一定在连续的补丁号间进行升级。在这个示例中，该频道并没有（且重来没有包括 4.12.2），因此不建议或不支持对 4.12.2 的更新。

3.1.6. 更新建议和升级

红帽会监控新发布的版本，以及把这些版本添加到支持的频道前后与那些版本关联的更新路径。

如果红帽从任何支持的发行版本中删除更新建议，则会为更正回归的未来版本提供取代的更新建议。但是，当缺陷被修正、测试并提升到您选择的频道时，可能会有一些延迟。

从 OpenShift Container Platform 4.10 开始，在确认更新风险时，它们会被声明为相关更新的条件更新风险。每个已知风险都可能适用于所有集群，或者只应用到与特定条件匹配的集群。有些示例包括将 **Platform** 设置为 **None**，或将 CNI 供应商设置为 **OpenShiftSDN**。Cluster Version Operator (CVO) 持续评估当前集群状态的已知风险。如果没有风险匹配，则建议更新。如果风险匹配，则支持这些更新但不推荐，并提供了一个参考链接。参考链接可帮助集群管理员决定是否接受风险和更新。

当红帽选择声明条件更新风险时，会在所有相关频道中同时采取该操作。Conditional Update risk 的声明可能会在更新被提升到支持的频道之前或之后发生。

3.1.7. 为集群选择正确的频道

选择适当的频道涉及两个决策。

首先，选择您要进行集群更新的次版本。选择与当前版本匹配的频道可确保您只应用 z-stream 更新，且不会接收功能更新。选择一个大于您当前版本的可用频道，以确保在一个或多个更新后，集群将更新至该版本。您的集群只提供与当前版本、下一个版本或下一个 EUS 版本匹配的频道。

注意

由于计划在很多次版本间更新的复杂性，频道可帮助计划在 EUS 到 EUS 更新之外进行更新。

其次，您应该选择您需要的 rollout 策略。当红帽声明了一个 GA 版本后，您可以选择从 fast 频道中选择更新，或者您可能要等待红帽将版本提升到 stable 频道。**fast-4.12** 和 **stable-4.12** 中提供的更新建议被完全支持，并从持续数据分析中同样获益。将发行版本提升到 stable 频道前的提升延迟会重新设置两个频道之间的唯一区别。对最新 z-streams 的更新通常会在一周或两个时间内提升到 stable 频道，但最初向最新次版本进行更新的时间更长时的延迟（通常为 45-90 天）。在选择所需频道时请考虑提升延迟，以等待到 stable 频道的提升可能会影响您的调度计划。

另外，有几个因素可能会导致机构永久或临时将集群移至 fast 频道，包括：

- 想要应用特定的修复，以便在不延迟的情况下影响您的环境。
- 在没有延迟的情况下修复 CVE 的应用程序。CVE 修复可能会引入回归问题，因此提升延迟仍然适用于带有 CVE 修复的 z-streams。
- 内部测试流程。如果您的组织需要数周时间来证明，则最好与我们提升流程同时测试，而不是等待。这也保证，红帽提供的任何遥测信号均是我们的推出中因素，因此可以更快地修复与您的问题相关的问题。

3.1.8. 受限网络集群

如果您自己为 OpenShift Container Platform 集群管理容器镜像，您必须考虑与产品关联的红帽勘误中的升级信息。在升级过程中，用户界面可能会提醒您在这些版本间进行切换，因此您必须在跳过这些警告前确定选择了正确的版本。

3.1.9. 在频道间切换

可以从 Web 控制台或通过 **adm upgrade channel** 命令来切换频道：

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

如果您切换到没有包括当前版本的频道，web 控制台将显示警报。在没有当前发行版本的频道中，web 控制台不推荐任何更新。但是，您可以在任何时候返回原始频道。

更改您的频道可能会影响集群的可支持性。可能适用以下条件：

- 如果您从 **stable-4.12** 频道改到 **fast-4.12** 频道，您的集群仍然被支持。
- 您可以随时切换到 **candidate-4.12** 频道，但此频道的一些发行版本可能不被支持。
- 如果您当前的发行本是正式发布版本，则可以从 **candidate-4.12** 频道切换到 **fast-4.12** 频道。
- 您始终可以从 **fast-4.12** 频道切换到 **stable-4.12** 频道。如果当前发行版本最近被提升，则可能会有最多一天的延迟该发行版本才会被提升到 **stable-4.12**。

其他资源

- 根据一个有条件的升级路径进行升级
- 为集群选择正确的频道

第 4 章 了解 OPENSHIFT CONTAINER PLATFORM 更新持续时间

OpenShift Container Platform 更新持续时间因部署拓扑而异。这个页可帮助您了解影响更新持续时间的因素，并估算集群更新在环境中所需的时间。

4.1. 先决条件

- 熟悉 OpenShift Container Platform 架构和 OpenShift Container Platform 更新。

4.2. 影响更新持续时间的因素

以下因素可能会影响您的集群更新持续时间：

- 通过 Machine Config Operator (MCO) 将计算节点重启到新机器配置
  - 机器配置池中的 **MaxUnavailable** 的值
  - pod 中断预算 (PDB) 中设定的最小副本数或百分比
- 集群中的节点数
- 集群节点的健康状况

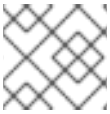
4.3. 集群更新阶段

在 OpenShift Container Platform 中，集群更新分为两个阶段：

- Cluster Version Operator (CVO) 目标更新有效负载部署
- Machine Config Operator (MCO) 节点更新

4.3.1. Cluster Version Operator 目标更新有效负载部署

Cluster Version Operator (CVO) 检索目标更新发行镜像并应用到集群。作为 pod 运行的所有组件都会在这个阶段更新，主机组件则由 Machine Config Operator (MCO) 更新。这个过程可能需要 60 到 120 分钟。



注意

更新的 CVO 阶段不会重启节点。

其他资源

- OpenShift 更新简介

4.3.2. Machine Config Operator 节点更新

Machine Config Operator (MCO) 将新机器配置应用到每个 control plane 和计算节点。在此过程中，MCO 在集群的每个节点中执行以下操作：

1. cordon 和 drain 所有节点
2. 更新操作系统 (OS)

3. 重新引导节点
4. 取消协调所有节点并在节点上调度工作负载

注意

当节点被封锁时，工作负载无法调度到其中。

完成此过程的时间取决于多个因素，包括节点和基础架构配置。此过程可能需要 5 分钟或更长时间来完成每个节点。

除了 MCO 外，您应该考虑以下参数的影响：

- control plane 节点更新持续时间是可预测的，通常比计算节点更短，因为 control plane 工作负载出于安全更新和快速排空进行了调优。
- 您可以通过将 **maxUnavailable** 字段设置为在 Machine Config Pool (MCP) 中大于 **1** 来并行更新计算节点。MCO 会处理 **maxUnavailable** 中指定的节点数量，并标记它们无法进行更新。
- 当您在 MCP 上增加 **maxUnavailable** 时，它可以帮助池更快地更新。但是，如果 **maxUnavailable** 太高，且同时处理几个节点，pod 中断预算 (PDB) 保护工作负载可能无法排空，因为无法找到调度的节点来运行副本。如果您为 MCP 增加 **maxUnavailable**，请确保仍然有足够的可调度节点来允许 PDB 保护的工作负载排空。
- 在开始更新前，您必须确保所有节点都可用。任何不可用的节点都可能会影响更新持续时间，因为节点不可用会影响 **maxUnavailable** 和 pod 中断预算。
  要从终端中检查节点状态，请运行以下命令：

  ```
  $ oc get node
  ```

  输出示例

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

  如果节点的状态为 **NotReady** 或 **SchedulingDisabled**，则该节点不可用，且这会影响更新持续时间。

  您可以通过展开 **Compute → Nodes** 从 web 控制台中的 **Administrator** 视角检查节点的状态。

其他资源

- 机器配置概述

- Pod 中断预算

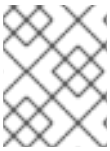
4.4. 估算集群更新时间

类似集群的历史更新持续时间为您提供了未来集群更新的最佳估算。但是，如果历史数据不可用，您可以使用以下约定来估算集群更新时间：

```
Cluster update time = CVO target update payload deployment time + (# node update iterations x MCO node update time)
```

节点更新迭代由并行更新的一个或多个节点组成。control plane 节点总会与计算节点并行更新。另外，根据 **maxUnavailable** 值可以并行更新一个或多个计算节点。

例如，要估算更新时间，请考虑一个具有三个 control plane 节点的 OpenShift Container Platform 集群，以及每个主机需要大约 5 分钟才能重启。



注意

重启特定节点所需的时间有很大不同。在云实例中，重新启动可能需要大约 1 到 2 分钟，而在物理主机中，重新引导可能需要超过 15 分钟。

场景 1

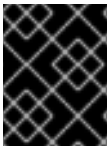
当您将 control plane 和计算节点机器配置池 (MCP) 的 **maxUnavailable** 设置为 **1** 时，所有 6 个计算节点会在每个迭代中逐一进行更新。

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

场景 2

当您为计算节点 MCP 将 **maxUnavailable** 设置为 **2** 时，两个计算节点会在每个迭代中并行更新。因此，它需要三个迭代来更新所有节点。

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



重要

对于 OpenShift Container Platform 中的所有 MCP，**maxUnavailable** 的默认设置为 **1**。建议您不要更改 control plane MCP 中的 **maxUnavailable**。

4.5. RED HAT ENTERPRISE LINUX (RHEL) 计算节点

Red Hat Enterprise Linux (RHEL) 计算节点需要额外使用 **openshift-ansible** 来更新节点二进制组件。更新 RHEL 计算节点的实际时间不应与 Red Hat Enterprise Linux CoreOS (RHCOS) 计算节点有很大不同。

其他资源

- 更新 RHEL 计算机器

第 5 章 准备升级到 OPENSHIFT CONTAINER PLATFORM 4.12

OpenShift Container Platform 4.12 使用 Kubernetes 1.25，它删除了几个已弃用的 API。

集群管理员必须在从 OpenShift Container Platform 4.11 升级到 4.12 前提供手动确认。这有助于防止升级到 OpenShift Container Platform 4.12 后出现问题，其中已删除的 API 仍在由运行或与集群交互的工作负载、工具或其他组件使用。管理员必须针对将要删除的任何 API 评估其集群，并迁移受影响的组件，以使用适当的新 API 版本。完成此评估和迁移后，管理员可以进行确认。

在将 OpenShift Container Platform 4.11 集群更新至 4.12 之前，您必须提供管理员确认。

5.1. 删除的 KUBERNETES API

OpenShift Container Platform 4.12 使用 Kubernetes 1.25，它删除了以下已弃用的 API。您必须迁移清单和 API 客户端以使用适当的 API 版本。有关迁移删除 API 的更多信息，请参阅 Kubernetes 文档。

表 5.1. 从 Kubernetes 1.25 中删除的 API

资源	删除的 API	迁移到	主要变化
CronJob	**batch/v1beta1**	**batch/v1**	否
EndpointSlice	**discovery.k8s.io/v1beta1**	**discovery.k8s.io/v1**	是
事件	**events.k8s.io/v1beta1**	**events.k8s.io/v1**	是
HorizontalPodAutoscaler	**autoscaling/v2beta1**	**autoscaling/v2**	否
PodDisruptionBudget	**policy/v1beta1**	**policy/v1**	是
PodSecurityPolicy	**policy/v1beta1**	Pod Security Admission [1]	是
RuntimeClass	**node.k8s.io/v1beta1**	**node.k8s.io/v1**	否

1. 如需有关 OpenShift Container Platform 中 pod 安全准入的更多信息，请参阅 *了解和管理 pod 安全准入*。

其他资源

- 了解并管理 pod 安全准入

5.2. 为删除的 API 评估集群

有多种方法可帮助管理员确定将使用移除的 API 的位置。但是，OpenShift Container Platform 无法识别所有实例，尤其是使用闲置或外部工具的工作负载。管理员负责针对已删除 API 实例正确评估所有工作负载和其他集成。

5.2.1. 查看警报以识别已删除 API 的使用

当使用 API 时会触发两个警报，该 API 将在以后的版本中删除：

- **APIRemovedInNextReleaseInUse** - 针对将在下一个 OpenShift Container Platform 发行版本中删除的 API。
- **APIRemovedInNextEUSReleaseInUse** - 针对将在下一个 OpenShift Container Platform 扩展更新支持（EUS）版本中删除的 API。

如果其中任何一个警报在集群中触发，请查看警报并采取措施通过迁移清单和 API 客户端以使用新的 API 版本来清除警报。

使用 **APIRequestCount** API 获取有关使用哪些 API 的更多信息，以及哪些工作负载正在使用删除的 API，因为警报不提供此信息。此外，一些 API 可能不会触发这些警报，但仍然被 **APIRequestCount** 捕获。将警报调优为不太敏感，以避免在生产环境中出现警报。

5.2.2. 使用 APIRequestCount 来识别已删除 API 的使用

您可以使用 **APIRequestCount API** 来跟踪 API 请求，并检查它们中的任何请求是否使用其中一个已删除的 API。

先决条件

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

流程

- 运行以下命令并检查输出的 **REMOVEDINRELEASE** 列以确定当前正在使用的 API:

```
$ oc get apirequestcounts
```

输出示例

```
NAME                                                                        REMOVEDINRELEASE
REQUESTSINCURRENTHOUR   REQUESTSINLAST24H
...
poddisruptionbudgets.v1.policy                                                                391                     8114
poddisruptionbudgets.v1beta1.policy                                           1.25          2                       23
podmonitors.v1.monitoring.coreos.com                                                          3                       70
podnetworkconnectivitychecks.v1alpha1.controlplane.operator.openshift.io                                      612
11748
pods.v1                                                                                    1531                    38634
podsecuritypolicies.v1beta1.policy                                            1.25          3                       39
podtemplates.v1                                                                            2                       79
preprovisioningimages.v1alpha1.metal3.io                                                                  2           39
priorityclasses.v1.scheduling.k8s.io                                                      12                      248
prioritylevelconfigurations.v1beta1.flowcontrol.apiserver.k8s.io              1.26          3
86
...
```

重要

您可以安全地忽略结果中出现的以下条目：

- **system:serviceaccount:kube-system:generic-garbage-collector** 和 **system:serviceaccount:kube-system:namespace-controller** 用户可能会出现在结果中，因为这些服务在搜索要删除的资源时调用所有注册的 API。
- **system:kube-controller-manager** 和 **system:cluster-policy-controller** 用户可能会出现在结果中，因为它们在强制执行各种策略时遍历所有资源。

您还可以使用 **-o jsonpath** 来过滤结果：

```
$ oc get apirequestcounts -o jsonpath='{range .items[?(@.status.removedInRelease!="")]}
{.status.removedInRelease}{"\t"}{.metadata.name}{"\n"}{end}'
```

输出示例

```
1.26 flowschemas.v1beta1.flowcontrol.apiserver.k8s.io
1.26 horizontalpodautoscalers.v2beta2.autoscaling
1.25 poddisruptionbudgets.v1beta1.policy
1.25 podsecuritypolicies.v1beta1.policy
1.26 prioritylevelconfigurations.v1beta1.flowcontrol.apiserver.k8s.io
```

5.2.3. 使用 APIRequestCount 来识别哪些工作负载正在使用删除的 API

您可以检查给定 API 版本的 **APIRequestCount** 资源，以帮助识别正在使用 API 的工作负载。

先决条件

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

流程

- 运行以下命令并检查 **username** 和 **userAgent** 字段，以帮助识别使用 API 的工作负载：

```
$ oc get apirequestcounts <resource>.<version>.<group> -o yaml
```

例如：

```
$ oc get apirequestcounts poddisruptionbudgets.v1beta1.policy -o yaml
```

您还可以使用 **-o jsonpath** 从 **APIRequestCount** 资源中提取 **username** 和 **userAgent** 值：

```
$ oc get apirequestcounts poddisruptionbudgets.v1beta1.policy \
  -o jsonpath='{range .status.currentHour..byUser[*]}{..byVerb[*].verb}{","}{.username}{","}
{.userAgent}{"\n"}{end}' \
  | sort -k 2 -t, -u | column -t -s, -NVERBS,USERNAME,USERAGENT
```

输出示例

```
VERBS  USERNAME                                                    USERAGENT
```

```
watch  system:serviceaccount:openshift-operators:3scale-operator
manager/v0.0.0
watch  system:serviceaccount:openshift-operators:datadog-operator-controller-manager
manager/v0.0.0
```

5.3. 迁移已删除 API 实例

有关如何迁移删除 Kubernetes API 的详情，请参阅 Kubernetes 文档中的已弃用 API 迁移指南。

5.4. 管理员确认

在为任何已删除的 API 评估集群并迁移了任何删除 API 后，您可以确认集群已准备好从 OpenShift Container Platform 4.11 升级到 4.12。

警告

请注意，管理员须负责确保已移除 API 的所有使用都已根据需要得到解决和迁移，然后再提供此管理员的确认。OpenShift Container Platform 可以协助评估，但无法识别所有可能移除的 API 的使用，特别是空闲的工作负载或外部工具。

先决条件

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

流程

- 运行以下命令，确认您已完成评估，集群已准备好在 OpenShift Container Platform 4.12 中删除 Kubernetes API：

```
$ oc -n openshift-config patch cm admin-acks --patch '{"data":{"ack-4.11-kube-1.25-api-
removals-in-4.12":"true"}}' --type=merge
```

第 6 章 准备执行 EUS 更新

由于 Kubernetes 的设计，次版本之间的所有 OpenShift Container Platform 升级都必须按顺序进行。您必须从 OpenShift Container Platform <4.y> 更新至 <4.y+1>，然后更新至 <4.y+2>。您无法直接从 OpenShift Container Platform <4.y> 更新至 <4.y+2>。但是，希望在两个延长更新支持 (EUS) 版本间更新的管理员可以这样做，而只需要重启一个非 control plane 主机。

重要

EUS 到 EUS 的更新只能在 **偶数的 OpenShift Container Platform 次版本之**间有效。

当尝试进行 EUS 到 EUS 更新时需要考虑很多注意事项。

- EUS 到 EUS 更新仅在所有涉及的版本在 **stable** 频道中提供。
- 如果您在升级到一个奇数次版本后但在升级到下一个偶数次版本前遇到了问题，则需要在继续进行前，把非控制平面主机完成升级到奇数次版本。
- 您可以通过更新 worker 或自定义池节点来对部分更新进行部分更新，以适应维护所需的时间。
- 您可以在多个维护窗口期间通过暂停中间步骤完成升级过程。但是，计划在 60 天内完成整个更新。确保完成正常的集群自动化过程非常重要，包括与证书轮转关联的操作。
- 在机器配置池被取消暂停且更新完成前，OpenShift Container Platform 的 <4.y+1> 和 <4.y+2> 中的一些功能和程序错误修复不可用。
- 所有集群可能会在不需要暂停池的情况下，对常规更新使用 EUS 频道进行更新，但只有具有非控制平面 **MachineConfigPools** 对象的集群可以进行 EUS 到 EUS 的更新，且需要暂停池。

6.1. EUS 到 EUS 更新

以下流程暂停所有非 master 机器配置池，并从 OpenShift Container Platform <4.y> 升级到 <4.y+1> 到 <4.y+2>，然后取消暂停之前暂停的机器配置池。以下过程减少了升级持续时间，以及重启 worker 节点的次数。

先决条件

- 查看 OpenShift Container Platform <4.y+1> 和 <4.y+2> 发行注记
- 查看任何层次产品和 Operator Lifecycle Manager (OLM) Operator 的发行注记和产品生命周期。有些操作可能需要在 EUS 到 EUS 更新之前或进行中进行。
- 确保您熟悉了特定于版本的前提条件，如删除已弃用的 API，在从 OpenShift Container Platform <4.y+1> 升级到 <4.y+2> 之前是必需的。

6.1.1. 使用 Web 控制台进行 EUS 到 EUS 更新

先决条件

- 验证机器配置池是否已取消暂停。
- 使用具有 **admin** 权限的用户登陆到 web 控制台。

流程

1. 使用 Web 控制台中的 Administrator 视角，将任何 Operator Lifecycle Manager (OLM) Operator 更新至与您的预期更新版本兼容的版本。您可以在"更新安装的 Operator"中找到有关如何执行此操作的更多信息；请参阅"添加资源"。

2. 验证所有机器配置池的状态是否为 **Up to date**，并且没有机器配置池显示 **UPDATING** 的状态。要查看所有机器配置池的状态，请点 **Compute → MachineConfigPools** 并查看 **Update status** 列的内容。

注意

如果您的机器配置池具有 **Updating** 状态，请等待此状态更改为 **Up to date**。这个过程可能需要几分钟时间。

3. 将频道设置为 **eus-<4.y+2>**。
要设置您的频道，请点 **Administration → Cluster Settings → Channel**。您可以点当前的超链接频道来编辑频道。

4. 暂停除 master 池外的所有 worker 机器池。您可以在 **Compute** 页面的 **MachineConfigPools** 标签页中执行此操作。选择您要暂停的机器配置池旁边的垂直图标，然后点 **暂停更新**。

5. 更新至 <4.y+1> 版本并完成到 **Save** 步骤。您可以在"使用 Web 控制台更新集群"中找到有关如何执行这些操作的更多信息；请参阅"添加资源"。

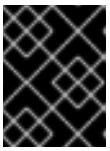
6. 通过查看集群的 **最新完成的版本**，确保 <4.y+1> 更新已完成。您可以在 **Details** 选项卡下的 **Cluster Settings** 页面中找到此信息。

7. 如有必要，使用 web 控制台中的 Administrator 视角更新 OLM Operator。您可以在"更新安装的 Operator"中找到有关如何执行此操作的更多信息；请参阅"添加资源"。

8. 更新至 <4.y+2> 版本并完成到 **Save** 步骤。您可以在"使用 Web 控制台更新集群"中找到有关如何执行这些操作的更多信息；请参阅"添加资源"。

9. 通过查看集群的 **最新完成的版本**，确保 <4.y+2> 更新已完成。您可以在 **Details** 选项卡下的 **Cluster Settings** 页面中找到此信息。

10. 取消暂停所有之前暂停的机器配置池。您可以在 **Compute** 页面的 **MachineConfigPools** 标签页中执行此操作。选择您要暂停的机器配置池旁边的垂直图标，然后点 **Unpause updates**。



重要

如果没有取消暂停池，集群就无法升级到将来的任何次要版本，而维护任务（如证书轮转）会被禁止。这会使集群面临未来降级的风险。

11. 验证之前暂停的池是否已更新，并且集群是否完成了更新到版本 <4.y+2>。
您可以通过 **Compute** 页中的 **MachineConfigPools** 标签页来验证您的池已更新。其 **Update status** 应用带有一个 **Up to date** 值。

    您可以通过查看集群的 **最新完成版本**来验证集群是否已完成更新。您可以在 **Details** 选项卡下的 **Cluster Settings** 页面中找到此信息。

其他资源

- 准备 Operator 更新
- 使用Web控制台更新集群

- 更新安装的 Operator

6.1.2. 使用 CLI 进行 EUS 到 EUS 更新

先决条件

- 验证机器配置池是否已取消暂停。
- 每次更新前，将 OpenShift CLI (**oc**) 更新至目标版本。

重要

强烈建议您跳过此先决条件。如果在更新前没有将 OpenShift CLI (**oc**) 更新至目标版本，则可能会出现意外的问题。

流程

1. 使用 Web 控制台中的 Administrator 视角，将任何 Operator Lifecycle Manager (OLM) Operator 更新至与您的预期更新版本兼容的版本。您可以在"更新安装的 Operator"中找到有关如何执行此操作的更多信息；请参阅"添加资源"。
2. 验证所有机器配置池的状态是否为 **UPDATED**，并且没有机器配置池显示 **UPDATING** 的状态。要查看所有机器配置池的状态，请运行以下命令：

```
$ oc get mcp
```

输出示例

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

3. 您当前版本为 <4.y>，您想要更新的预期版本为 <4.y+2>。运行以下命令，进入 **eus-<4.y+2>** 频道：

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

注意

如果您收到一条错误消息，表示 **eus-<4.y+2>** 不是可用频道之一，这表示红帽对 EUS 版本更新的推出仍在进行中。本推出过程通常在 GA 日期开始 45-90 天。

4. 运行以下命令，暂停除 master 池之外的所有 worker 机器池：

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

注意

您无法暂停 master 池。

5. 运行以下命令来更新到最新版本：

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

输出示例

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

6. 运行以下命令，查看集群版本以确保更新已完成：

```
$ oc adm upgrade
```

输出示例

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

7. 运行以下命令，更新至 <4.y+2> 版本：

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

8. 运行以下命令，检索集群版本以确保 <4.y+2> 更新已完成：

```
$ oc adm upgrade
```

输出示例

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

9. 要将 worker 节点更新至 <4.y+2>，请运行以下命令取消暂停所有之前暂停的机器配置池：

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

重要

如果没有取消暂停池，集群就无法升级到将来的任何次要版本，而维护任务（如证书轮转）会被禁止。这会使集群面临未来降级的风险。

10. 运行以下命令，验证之前暂停的池是否已更新，并升级到 <4.y+2> 版本：

```
$ oc get mcp
```

输出示例

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

其他资源

- 更新安装的 Operator

6.1.3. 通过 Operator Lifecycle Manager 安装的分层产品和 Operator 的 EUS 到 EUS 更新

除了 web 控制台和 CLI 提到的 EUS 到 EUS 更新步骤外，还需要考虑使用以下内容为集群执行 EUS 到 EUS 更新的步骤：

- 层次产品
- 通过 Operator Lifecycle Manager (OLM) 安装的 Operator

什么是层次产品？

层次产品指的是由多个底层产品组成的产品，它们旨在一起使用且不能分为单独的订阅。有关层次 OpenShift Container Platform 产品的示例，请参阅 OpenShift 的分层组件。

当您为分层产品的集群以及通过 OLM 安装的 Operator 执行 EUS 到 EUS 更新时，您必须完成以下操作：

1. 确保之前通过 OLM 安装的所有 Operator 均在其最新频道中更新至其最新版本。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，它们具有有效的升级路径。有关如何更新 Operator 的详情，请参考"添加资源"中的"准备 Operator 更新"。
2. 确认当前和预期的 Operator 版本之间的集群版本兼容性。您可以使用 Red Hat OpenShift Container Platform Operator Update Information Checker 验证 OLM Operator 兼容哪些版本。

例如，以下是为 OpenShift Data Foundation (ODF)执行从 <4.y> 到 <4.y+2> 的 EUS 到 EUS 更新的步骤。这可以通过 CLI 或 Web 控制台来完成。有关如何通过所需接口更新集群的详情，请参考 *使用 Web 控制台进行 EUS 到 EUS 的升级*和 "Additional 资源"中的"EUS 到 EUS 的更新"。

工作流示例

1. 暂停 worker 机器池。
2. 升级 OpenShift <4.y> 到 OpenShift <4.y+1>。
3. 升级 ODF <4.y> 到 ODF <4.y+1>。
4. 升级 OpenShift <4.y+1> 到 OpenShift <4.y+2>。
5. 升级到 ODF <4.y+2>。
6. 取消暂停 worker 机器池。

注意

升级到 ODF <4.y+2> 可以在 worker 机器池被取消暂停前或之后进行。

其他资源

- 准备 Operator 更新
- 使用 Web 控制台进行 EUS 到 EUS 更新
- 使用 CLI 进行 EUS 到 EUS 更新

第 7 章 准备使用手动维护的凭证更新集群

默认情况下，带有手动维护凭证的集群的 Cloud Credential Operator(CCO)U **gradable 状态为 False**。

- 对于次发行版本（例如从 4.12 升级到 4.13），这个状态会阻止升级，直到您解决了任何更新的权限并 **添加了 CloudCredential** 资源，以指示下一版本根据需要更新权限。此注解将 **Upgradable** 状态更改为 **True**。
- 对于 z-stream 版本（例如从 4.13.0 到 4.13.1），没有添加或更改任何权限，因此不会阻止升级。

在使用手动维护的凭证更新集群前，您必须在要升级到的 OpenShift Container Platform 版本的发行镜像中包含任何新的或更改的凭证。

7.1. 使用手动维护的凭证更新集群的要求

在通过 Cloud Credential Operator (CCO) 更新使用手动维护凭证的集群前，您必须为新版本更新云供应商资源。

如果使用 CCO 实用程序 (**ccoctl**) 配置集群的云凭证管理，请使用 **ccoctl** 实用程序更新资源。配置为在没有 **ccoctl** 工具的情况下使用手动模式的集群需要手动更新资源。

更新云供应商资源后，您必须更新集群的 **upgradeable-to** 注解，以指示它已准备好更新。

注意

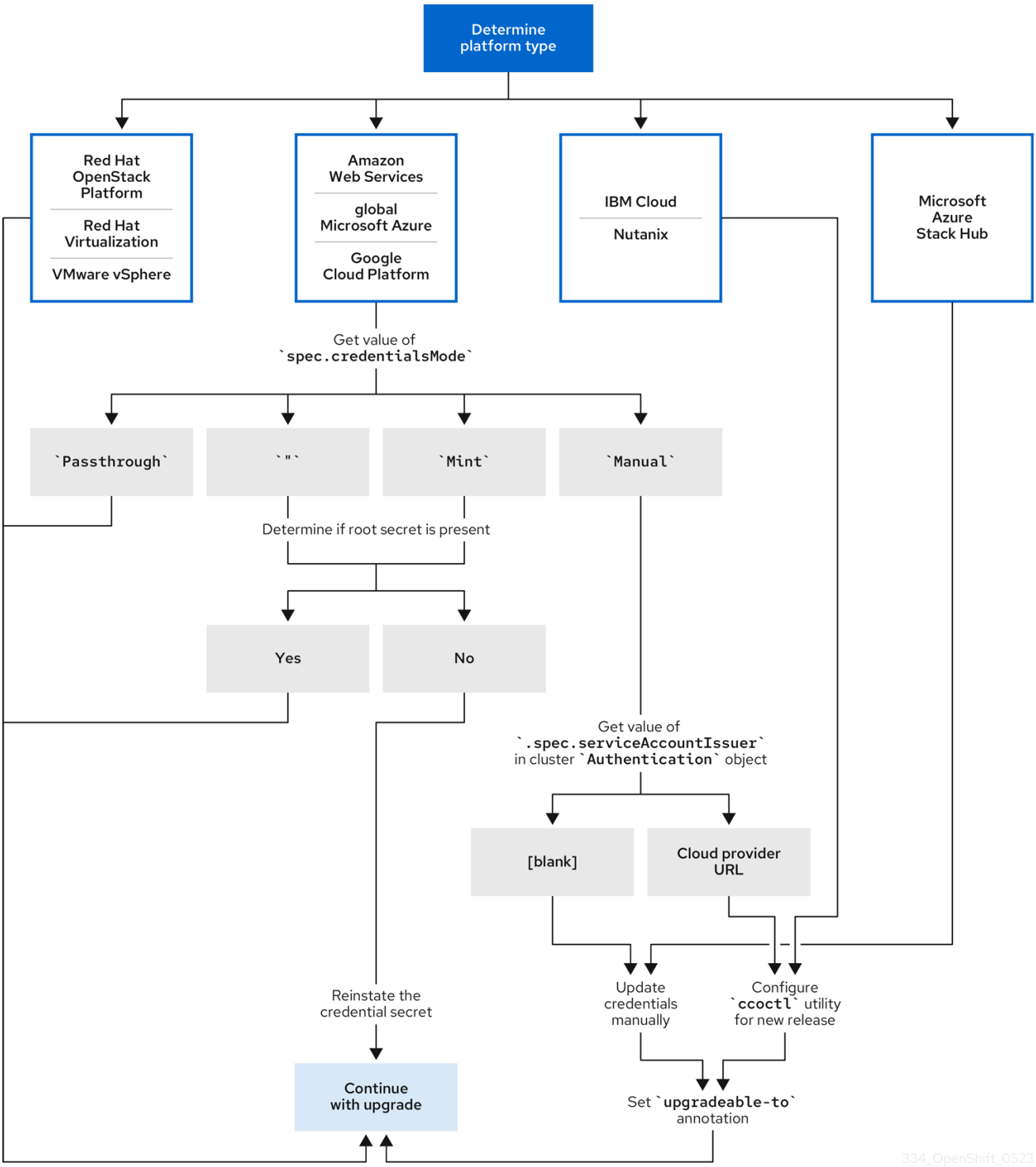
更新云供应商资源和 **upgradeable-to** 注解的过程只能通过命令行工具完成。

7.1.1. 云凭证配置选项并根据平台类型更新要求

有些平台只支持在一个模式中使用 CCO。对于在这些平台上安装的集群，平台类型决定了凭证更新要求。

对于支持多个模式中使用 CCO 的平台，您必须确定集群配置为使用哪种模式，并对该配置执行所需操作。

图 7.1. 按平台类型进行凭证更新要求



Red Hat OpenStack Platform (RHOSP)、Red Hat Virtualization (RHV)和 VMware vSphere

这些平台不支持在手动模式中使用 CCO。这些平台上的集群会自动处理云供应商资源的更改，不需要对 **upgradeable-to** 的注解进行更新。

这些平台上的集群管理员应该跳过更新过程的手动维护凭证部分。

{IBM-cloud-title} 和 Nutanix

在这些平台上安装的集群使用 **ccoctl** 工具进行配置。

这些平台上的集群管理员必须执行以下操作：

1. 为新版本配置 **ccoctl** 工具。

2. 使用 **ccoctl** 实用程序更新云供应商资源。
3. 指明集群可以使用 **upgradeable-to** 注解进行更新。

Microsoft Azure Stack Hub

这些集群使用带有长期凭证的手动模式，且不使用 **ccoctl** 工具。
这些平台上的集群管理员必须执行以下操作：

1. 为新版本手动更新云供应商资源。
2. 指明集群可以使用 **upgradeable-to** 注解进行更新。

Amazon Web Services (AWS)、全局 Microsoft Azure 和 Google Cloud Platform (GCP)

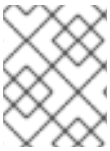
在这些平台上安装的集群支持多个 CCO 模式。
所需的更新过程取决于集群配置为使用的模式。如果您不确定 CCO 在集群中要使用的模式，您可以使用 Web 控制台或 CLI 来决定此信息。

其他资源

- 使用 Web 控制台确定 Cloud Credential Operator 模式
- 使用 CLI 确定 Cloud Credential Operator 模式
- 为集群更新配置 Cloud Credential Operator 工具
- 使用手动维护的凭证更新云供应商资源
- 关于 Cloud Credential Operator

7.1.2. 使用 Web 控制台确定 Cloud Credential Operator 模式

您可以使用 Web 控制台确定 Cloud Credential Operator (CCO) 配置为使用哪种模式。



注意

只有 Amazon Web Services (AWS)、全局 Microsoft Azure 和 Google Cloud Platform (GCP) 集群支持多个 CCO 模式。

先决条件

- 您可以使用集群管理员权限访问 OpenShift Container Platform 帐户。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
2. 导航至 **Administration → Cluster Settings**。
3. 在 **Cluster Settings** 页面中，选择 **Configuration** 选项卡。
4. 在 **Configuration resource** 下，选择 **CloudCredential**。
5. 在 **CloudCredential 详情**页中，选择 **YAML** 选项卡。

6. 在 YAML 块中，检查 **spec.credentialsMode** 的值。以下是可能的值，但它们可能并不会在所有平台上都被支持：

   - **''** ：CCO 在默认模式下运行。在这个配置中，CCO 以 mint 或 passthrough 模式运行，具体取决于安装期间提供的凭证。
   - **Mint** ：CCO 在 mint 模式中运行。
   - **Passthrough** ：CCO 在 passthrough 模式运行。
   - **Manual** ：CCO 以手动模式运行。

重要

要确定 AWS 或 GCP 集群的特定配置，其 **spec.credentialsMode** 为 **''**, **Mint**, 或 **Manual**，您必须进一步调查。

AWS 和 GCP 集群支持使用删除了 root secret 的 mint 模式。如果集群被特别配置为使用 mint 模式或默认使用 mint 模式，则必须在更新前确定集群中是否存在 root secret。

使用手动模式的 AWS 或 GCP 集群可能会被配置为使用 AWS 安全令牌服务 (STS) 或 GCP Workload Identity 从集群外部创建和管理云凭证。您可以通过检查集群 **Authentication** 对象来确定集群是否使用了此策略。

7. 只使用 mint 模式的 AWS 或 GCP 集群：要确定集群是否在没有 root secret 的情况下运行，请进入到 **Workloads → Secrets** 并为您的云供应商查找 root secret。

注意

确保将 **项目** 下拉菜单设置为 **All Projects**。

平台	Secret 名称
AWS	**aws-creds**
GCP	**gcp-credentials**

   - 如果您看到这些值之一，您的集群将使用 mint 或 passthrough 模式，并带有 root secret。
   - 如果没有看到这些值，您的集群将以 mint 模式使用 CCO，并删除 root secret。

8. 只使用手动模式的 AWS 或 GCP 集群 ：要确定集群是否被配置为从集群外创建和管理云凭证，您必须检查 cluster **Authentication** 对象 YAML 值。

   a. 导航至 **Administration → Cluster Settings**。

   b. 在 **Cluster Settings** 页面中，选择 **Configuration** 选项卡。

   c. 在 **Configuration resource** 下，选择 **Authentication**。

   d. 在 **Authentication details** 页面中，选择 **YAML** 选项卡。

e. 在 YAML 块中，检查 **.spec.serviceAccountIssuer** 参数的值。

- 包含与云供应商关联的 URL 的值表示 CCO 在 AWS STS 或 GCP Workload Identity 中使用手动模式从集群外部创建和管理云凭证。这些集群使用 **ccoctl** 工具进行配置。
- 空值 ('') 表示集群在手动模式中使用 CCO，但没有使用 **ccoctl** 工具进行配置。

后续步骤

- 如果您要更新以 mint 或 passthrough 模式运行的 CCO 的集群，且存在 root secret，则不需要更新任何云供应商资源，并可以继续进入更新过程的下一部分。
- 如果您的集群在 mint 模式中使用删除了 root secret 的 CCO，则必须重新使用管理员级别的凭证重新恢复凭证 secret，然后才能继续更新过程的下一部分。
- 如果您的集群使用 CCO 实用程序 (**ccoctl**) 配置，您必须执行以下操作：
  a. 为新版本配置 **ccoctl** 工具，并使用它来更新云供应商资源。
  b. 更新 **upgradeable-to** 注解，以指示集群已准备好更新。
- 如果您的集群以手动模式使用 CCO，但没有使用 **ccoctl** 工具配置，则必须执行以下操作：
  a. 为新版本手动更新云供应商资源。
  b. 更新 **upgradeable-to** 注解，以指示集群已准备好更新。

其他资源

- 为集群更新配置 Cloud Credential Operator 工具
- 使用手动维护的凭证更新云供应商资源

7.1.3. 使用 CLI 确定 Cloud Credential Operator 模式

您可以使用 CLI 确定 Cloud Credential Operator (CCO) 配置为使用哪种模式。

注意

只有 Amazon Web Services (AWS)、全局 Microsoft Azure 和 Google Cloud Platform (GCP) 集群支持多个 CCO 模式。

先决条件

- 您可以使用集群管理员权限访问 OpenShift Container Platform 帐户。
- 已安装 OpenShift CLI(**oc**)。

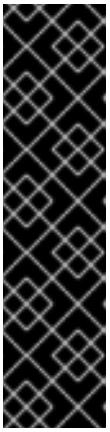
流程

1. 以具有 **cluster-admin** 角色的用户身份登录到集群中的 **oc**。
2. 要确定 CCO 被配置为使用的模式，请输入以下命令：

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

以下输出值可能，但并非所有平台上都不支持所有值：

- **''** ：CCO 在默认模式下运行。在这个配置中，CCO 以 mint 或 passthrough 模式运行，具体取决于安装期间提供的凭证。
- **Mint** ：CCO 在 mint 模式中运行。
- **Passthrough** ：CCO 在 passthrough 模式运行。
- **Manual** ：CCO 以手动模式运行。



重要

要确定 AWS 或 GCP 集群的特定配置，其 **spec.credentialsMode** 为 **''**, **Mint**, 或 **Manual**，您必须进一步调查。

AWS 和 GCP 集群支持使用删除了 root secret 的 mint 模式。如果集群被特别配置为使用 mint 模式或默认使用 mint 模式，则必须在更新前确定集群中是否存在 root secret。

使用手动模式的 AWS 或 GCP 集群可能会被配置为使用 AWS 安全令牌服务 (STS) 或 GCP Workload Identity 从集群外部创建和管理云凭证。您可以通过检查集群 **Authentication** 对象来确定集群是否使用了此策略。

3. 仅使用 mint 模式的 AWS 或 GCP 集群：要确定集群是否在没有 root secret 的情况下运行，请运行以下命令：

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

其中 **<secret_name>** 是 AWS 的 **aws-creds** 或 GCP 的 **gcp-credentials**。

如果存在 root secret，这个命令的输出会返回有关 secret 的信息。错误表示集群中不存在 root secret。

4. 仅使用手动模式的 AWS 或 GCP 集群 ：要确定集群是否被配置为从集群外部创建和管理云凭证，请运行以下命令：

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

此命令显示集群 **Authentication** 对象中 **.spec.serviceAccountIssuer** 参数的值。

- 包含与云供应商关联的 URL 的输出表示 CCO 在 AWS STS 或 GCP Workload Identity 中使用手动模式从集群外部创建和管理云凭证。这些集群使用 **ccoctl** 工具进行配置。
- 空输出表示集群以手动模式使用 CCO，但没有使用 **ccoctl** 工具进行配置。

后续步骤

- 如果您要更新以 mint 或 passthrough 模式运行的 CCO 的集群，且存在 root secret，则不需要更新任何云供应商资源，并可以继续进入更新过程的下一部分。

- 如果您的集群在 mint 模式中使用删除了 root secret 的 CCO，则必须重新使用管理员级别的凭证重新恢复凭证 secret，然后才能继续更新过程的下一部分。
- 如果您的集群使用 CCO 实用程序 (**ccoctl**) 配置，您必须执行以下操作：
  a. 为新版本配置 **ccoctl** 工具，并使用它来更新云供应商资源。
  b. 更新 **upgradeable-to** 注解，以指示集群已准备好更新。
- 如果您的集群以手动模式使用 CCO，但没有使用 **ccoctl** 工具配置，则必须执行以下操作：
  a. 为新版本手动更新云供应商资源。
  b. 更新 **upgradeable-to** 注解，以指示集群已准备好更新。

其他资源

- 为集群更新配置 Cloud Credential Operator 工具
- 使用手动维护的凭证更新云供应商资源

7.2. 为集群更新配置 CLOUD CREDENTIAL OPERATOR 工具

要升级以手动模式使用 Cloud Credential Operator (CCO) 从集群外部创建和管理云凭证集群，提取并准备 CCO 实用程序 (**ccoctl**) 二进制文件。

注意

ccoctl 工具是在 Linux 环境中运行的 Linux 二进制文件。

先决条件

- 您可以访问具有集群管理员权限的 OpenShift Container Platform 帐户。
- 已安装 OpenShift CLI(**oc**)。
- 集群被配置为使用 **ccoctl** 实用程序从集群外创建和管理云凭证。

流程

1. 运行以下命令来获取 OpenShift Container Platform 发行镜像：

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 运行以下命令，从 OpenShift Container Platform 发行镜像获取 CCO 容器镜像：

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

注意

确保 **$RELEASE_IMAGE** 的架构与将使用 **ccoctl** 工具的环境架构相匹配。

3. 运行以下命令，将 CCO 容器镜像中的 **ccoctl** 二进制文件提取到 OpenShift Container Platform 发行镜像中：

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

4. 运行以下命令更改权限以使 **ccoctl** 可执行：

```
$ chmod 775 ccoctl
```

验证

- 要验证 **ccoctl** 是否可使用，请运行以下命令来显示帮助文件：

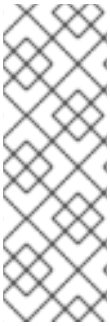
```
$ ccoctl --help
```

ccoctl --help 的输出

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  alibabacloud Manage credentials objects for alibaba cloud
  aws          Manage credentials objects for AWS cloud
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for IBM Cloud
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

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

7.3. 使用 CLOUD CREDENTIAL OPERATOR 工具更新云供应商资源

升级使用 CCO 实用程序 (**ccoctl**) 配置的 OpenShift Container Platform 集群的过程与在安装过程中创建云供应商资源类似。



注意

默认情况下，**ccoctl** 在运行命令的目录中创建对象。要在其他目录中创建对象，请使用 **--output-dir** 标志。此流程使用 **<path_to_ccoctl_output_dir>** 来引用这个目录。

在 AWS 集群中，一些 **ccoctl** 命令会发出 AWS API 调用来创建或修改 AWS 资源。您可以使用 **--dry-run** 标志来避免 API 调用。使用此标志可在本地文件系统中创建 JSON 文件。您可以使用 **--cli-input-json** 参数查看和修改 JSON 文件，然后使用 AWS CLI 工具应用它们。

先决条件

- 获取您要升级到的版本的 OpenShift Container Platform 发行镜像。

- 从发行镜像中提取并准备 **ccoctl** 二进制文件。

流程

1. 运行以下命令，从 OpenShift Container Platform 发行镜像中提取 **CredentialsRequest** 自定义资源 (CR) 列表：

```
$ oc adm release extract --credentials-requests \
  --cloud=<provider_type> \
  --to=<path_to_directory_with_list_of_credentials_requests>/credrequests \
  quay.io/<path_to>/ocp-release:<version>
```

   其中：

   - **<provider_type>** 是云供应商的值。有效值为 **alibabacloud**、**aws**、**gcp**、**ibmcloud** 和 **nutanix**。
   - **credrequests** 是存储 **CredentialsRequest** 对象列表的目录。如果该目录不存在，此命令就会创建该目录。

2. 对于发行镜像中的每个 **CredentialsRequest** CR，请确保集群中存在与 **spec.secretRef.namespace** 字段中的文本匹配的命名空间。此字段是保存凭证配置的生成的 secret 的位置。

   AWS CredentialsRequest 对象示例

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

   (1) 此字段指示需要存在的命名空间来存放生成的 secret。

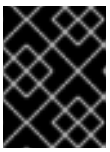
   其他平台的 **CredentialsRequest** CR 具有与不同平台值类似的格式。

3. 对于任何集群的 **CredentialsRequest** CR 还没有在 **spec.secretRef.namespace** 中指定的命名空间，运行以下命令创建命名空间：

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

4. 通过为您的云供应商运行命令，使用 **ccoctl** 工具处理 **credrequests** 目录中的所有 **CredentialsRequest** 对象。以下命令处理 **CredentialsRequest** 对象：

   - Alibaba Cloud: **ccoctl alibabacloud create-ram-users**
   - Amazon Web Services (AWS)： **ccoctl aws create-iam-roles**
   - Google Cloud Platform (GCP)： **ccoctl gcp create-all**
   - IBM Cloud: **ccoctl ibmcloud create-service-id**
   - Nutanix: **ccoctl nutanix create-shared-secrets**

   重要

   有关所需参数和特殊注意事项，请参阅云供应商的安装内容中的 **ccoctl** 工具说明。

   对于每个 **CredentialsRequest** 对象，**ccoctl** 会创建所需的供应商资源和 OpenShift Container Platform 发行镜像中的每个 **CredentialsRequest** 对象中定义的权限策略。

5. 运行以下命令，将 secret 应用到集群：

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

验证

您可以通过查询云供应商来验证是否已创建所需的供应商资源和权限策略。如需更多信息，请参阅有关列出角色或服务帐户的云供应商文档。

后续步骤

- 更新 **upgradeable-to** 注解，以指示集群已准备好升级。

其他资源

- 使用 **ccoctl** 工具为 OpenShift Container Platform 组件创建 Alibaba Cloud 凭证
- 使用 Cloud Credential Operator 实用程序创建 AWS 资源
- 使用 Cloud Credential Operator 实用程序创建 GCP 资源
- 为 IBM Cloud VPC 手动创建 IAM
- 为 Nutanix 配置 IAM
- 表示集群已准备好升级

7.4. 使用手动维护的凭证更新云供应商资源

在使用手动维护凭证升级集群前，您必须为要升级到的发行镜像创建新凭证。您还必须查看现有凭证所需的权限，并满足这些组件的新版本中任何新权限要求。

流程

1. 提取并检查 **新版本的 CredentialsRequest** 自定义资源。
   详情请参阅您的云供应商的“手动创建 IAM”部分来了解如何获取和使用您的云所需的凭证。

2. 更新集群中手动维护的凭证：

   - 为新发行镜像添加的任何 **CredentialsRequest** 自定义资源创建新 secret。
   - 如果存储在 secret 中的任何现有凭证的 **CredentialsRequest** 自定义资源更改了其权限要求，请根据需要更新权限。

3. 如果您的集群使用集群功能禁用一个或多个可选组件，请删除任何禁用组件的 **CredentialsRequest** 自定义资源。

AWS 上 OpenShift Container Platform 4.12 的 credrequests 目录的内容示例

```
0000_30_machine-api-operator_00_credentials-request.yaml 1
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml 2
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml 3
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 4
0000_50_cluster-network-operator_02-cncc-credentials.yaml 5
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml 6
```

1. Machine API Operator CR 是必需的。
2. 需要 Cloud Credential Operator CR。
3. Image Registry Operator CR 是必需的。
4. Ingress Operator CR 是必需的。
5. Network Operator CR 是必需的。
6. Storage Operator CR 是一个可选组件，可能会在集群中禁用。

GCP 上的 OpenShift Container Platform 4.12 的 credrequests 目录的内容示例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml 1
0000_30_machine-api-operator_00_credentials-request.yaml 2
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml 3
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml 4
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 5
0000_50_cluster-network-operator_02-cncc-credentials.yaml 6
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml 7
```

1. 需要 Cloud Controller Manager Operator CR。
2. Machine API Operator CR 是必需的。
3. 需要 Cloud Credential Operator CR。
4. Image Registry Operator CR 是必需的。
5. Ingress Operator CR 是必需的。

6 Network Operator CR 是必需的。

7 Storage Operator CR 是一个可选组件，可能会在集群中禁用。

后续步骤

- 更新 **upgradeable-to** 注解，以指示集群已准备好升级。

其他资源

- 为 AWS 手动创建 IAM
- 为 Azure 手动创建 IAM
- 为 Azure Stack Hub 手动创建 IAM
- 为 GCP 手动创建 IAM
- 表示集群已准备好升级

7.5. 表示集群已准备好升级

默认情况下，带有手动维护凭证的集群的 Cloud Credential Operator(CCO)U **gradable 状态为 False**。

先决条件

- 对于您要升级到的发行镜像，您可以手动处理任何新凭证，或使用 Cloud Credential Operator 实用程序 (**ccoctl**) 处理。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 以具有 **cluster-admin** 角色的用户身份登录到集群中的 **oc**。
2. 运行以下命令，编辑 **CloudCredential** 资源，以在 **metadata** 字段中添加 **upgradeable-to** 注解：

```
$ oc edit cloudcredential cluster
```

要添加的文本

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

其中 **<version_number>** 是您要升级到的版本，格式为 **x.y.z**。例如，OpenShift Container Platform 4.12.2 使用 **4.12.2**。

添加可升级状态进行更改的注解后，可能需要几分钟时间。

验证

验证

1. 在 Web 控制台的 **Administrator 视角中**，导航到 **Administration → Cluster Settings**。
2. 要查看 CCO 状态详情，请点击 **Cluster Operators** 列表中的 **cloud-credential**。
   - 如果 **Conditions** 部分中的 **Upgradeable** 状态为 **False**，请验证 **upgradeable-to** 注解没有拼写错误。
3. 当 **Conditions** 部分中的 **Upgradeable** 状态为 **True** 时，开始 OpenShift Container Platform 升级。

第 8 章 使用 WEB 控制台更新集群

您可以使用 Web 控制台在 OpenShift Container Platform 集群上执行次版本和补丁更新。

注意

使用 Web 控制台或 **oc adm upgrade channel *<channel>*** 更改更新频道。在改到一个 4.12 频道后，按使用 CLI 更新集群中的步骤完成更新。

8.1. 先决条件

- 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。
- 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。
- OpenShift Container Platform 4.12 中删除了对 RHEL7 worker 的支持。升级到 OpenShift Container Platform 4.12 之前，您必须将 RHEL7 worker 替换为 RHEL8 或 RHCOS worker。红帽不支持对 RHEL worker 的从 RHEL7 到 RHEL8 的原位升级；这些主机必须使用干净的操作系统安装替换。
- 确保之前通过 Operator Lifecycle Manager（OLM）安装的所有 Operator 均更新至其最新频道中的最新版本。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。如需更多信息，请参阅更新安装的 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。
- 要容纳更新所需的时间，您可以通过更新 worker 或自定义池节点来进行部分更新。您可以在每个池的进度栏中暂停并恢复。
- 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。
- 检查 Kubernetes 1.25 中删除的 API 列表，将任何受影响的组件迁移到使用新的 API 版本，并提供管理员确认。如需更多信息，请参阅准备升级到 OpenShift Container Platform 4.12 。
- 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。

重要

- 当更新无法完成时，Cluster Version Operator(CVO)会在尝试协调更新时报告任何阻塞组件的状态。当前还不支持将集群还原到以前的版本。如果您的更新无法完成，请联系红帽支持。
- 使用 **unsupportedConfigOverrides** 部分修改 Operator 配置不受支持，并可能会阻止集群更新。您必须在更新集群前删除此设置。

其他资源

- 非受管 Operator 的支持策略

8.2. 执行 CANARY ROLLOUT 更新

在某些情况下，您可能需要一个受控的更新过程，如您不希望特定节点与集群的其余部分同时更新。这些用例包括但不限于：

- 您有任务关键型应用程序，您希望在更新过程中仍然可以使用这些应用程序。在更新后，您可以慢慢地以小批的形式对应用进行测试。
- 您有一个短的维护窗口，在此期间不允许更新所有节点；或者您有多个维护窗口。

滚动更新过程**不是**典型的更新工作流。对于较大的集群，这可能是一个耗时的过程，需要您执行多个命令。这种复杂性可能会导致出现可能会影响整个集群的错误。建议您仔细考虑您的机构是否希望使用滚动更新，并在开始前仔细规划流程的实施。

本主题中描述的滚动更新过程涉及：

- 创建一个或多个自定义机器配置池 (MCP)。
- 标记您不想立即更新的每个节点，以将这些节点移至自定义 MCP。
- 暂停这些自定义 MCP，这会阻止对这些节点的更新。
- 执行集群更新。
- 取消暂停一个自定义 MCP，它会在这些节点上触发更新。
- 测试这些节点上的应用程序，以确保应用程序在这些新更新的节点上可以正常工作。
- （可选）从小批处理中的其余节点移除自定义标签，并在这些节点上测试应用。

注意

暂停 MCP 可防止 Machine Config Operator 在关联的节点上应用任何配置更改。暂停 MCP 还可以防止任何自动轮转的证书被推送到关联的节点，包括自动轮转 **kube-apiserver-to-kubelet-signer** CA 证书。

如果 **kube-apiserver-to-kubelet-signer** CA 证书过期且 MCO 尝试自动更新证书时，MCP 会暂停，但不会在相应机器配置池中的节点中应用新证书。这会导致多个 **oc** 命令失败，包括 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。如果在轮转证书时，如果 MCP 被暂停，则 OpenShift Container Platform Web 控制台的 Alerting UI 中收到警报。

在暂停 MCP 时应该非常小心，需要仔细考虑 **kube-apiserver-to-kubelet-signer** CA 证书过期的问题，且仅在短时间内暂停。

如果要使用 Canary rollout 更新过程，请参阅执行 Canary 推出部署更新。

8.3. 使用手动维护的凭证更新云供应商资源

在使用手动维护凭证升级集群前，您必须为要升级到的发行镜像创建新凭证。您还必须查看现有凭证所需的权限，并满足这些组件的新版本中任何新权限要求。

流程

1. 提取并检查 **新版本的 CredentialsRequest** 自定义资源。
   详情请参阅您的云供应商的“手动创建 IAM”部分来了解如何获取和使用您的云所需的凭证。

2. 更新集群中手动维护的凭证：

   - 为新发行镜像添加的任何 **CredentialsRequest** 自定义资源创建新 secret。
   - 如果存储在 secret 中的任何现有凭证的 **CredentialsRequest** 自定义资源更改了其权限要求，请根据需要更新权限。

3. 如果您的集群使用集群功能禁用一个或多个可选组件，请删除任何禁用组件的 **CredentialsRequest** 自定义资源。

AWS 上 OpenShift Container Platform 4.12 的 credrequests 目录的内容示例

```
0000_30_machine-api-operator_00_credentials-request.yaml 1
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml 2
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml 3
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 4
0000_50_cluster-network-operator_02-cncc-credentials.yaml 5
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml 6
```

1. Machine API Operator CR 是必需的。
2. 需要 Cloud Credential Operator CR。
3. Image Registry Operator CR 是必需的。
4. Ingress Operator CR 是必需的。
5. Network Operator CR 是必需的。
6. Storage Operator CR 是一个可选组件，可能会在集群中禁用。

GCP 上的 OpenShift Container Platform 4.12 的 credrequests 目录的内容示例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml 1
0000_30_machine-api-operator_00_credentials-request.yaml 2
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml 3
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml 4
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml 5
0000_50_cluster-network-operator_02-cncc-credentials.yaml 6
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml 7
```

1. 需要 Cloud Controller Manager Operator CR。
2. Machine API Operator CR 是必需的。
3. 需要 Cloud Credential Operator CR。
4. Image Registry Operator CR 是必需的。
5. Ingress Operator CR 是必需的。
6. Network Operator CR 是必需的。

7 Storage Operator CR 是一个可选组件，可能会在集群中禁用。

后续步骤

- 更新 **upgradeable-to** 注解，以指示集群已准备好升级。

其他资源

- 为 AWS 手动创建 IAM
- 为 Azure 手动创建 IAM
- 为 GCP 手动创建 IAM

8.4. 使用 WEB 控制台暂停 MACHINEHEALTHCHECK 资源

在升级过程中，集群中的节点可能会临时不可用。对于 worker 节点，机器健康检查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。

先决条件

- 您可以使用 **cluster-admin** 权限访问集群。
- 访问 OpenShift Container Platform web 控制台。

流程

1. 登陆到 OpenShift Container Platform Web 控制台。
2. 进入到 **Compute → MachineHealthChecks**。
3. 要暂停机器健康检查，请在每个 **MachineHealthCheck** 资源中添加 **cluster.x-k8s.io/paused=""** 注解。例如，要将注解添加到 **machine-api-termination-handler** 资源，请完成以下步骤：
   a. 点 **machine-api-termination-handler** 旁边的 Options 菜单 并点 **Edit annotations**。
   b. 在 **Edit annotations** 对话框中，点 **Add more**。
   c. 在 **Key** 和 **Value** 字段中，分别添加 **cluster.x-k8s.io/paused** 和 **""** 值，然后点 **Save**。

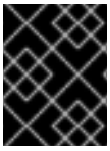
8.5. 关于更新单个节点 OPENSHIFT CONTAINER PLATFORM

您可以使用控制台或 CLI 更新或升级单节点 OpenShift Container Platform 集群。

但请注意以下限制：

- 不需要暂停 **MachineHealthCheck** 资源，因为没有其他节点可以执行健康检查。
- 不支持使用 etcd 备份来恢复单节点 OpenShift Container Platform 集群。但是，最好在升级失败时执行 etcd 备份。如果 control plane 健康，您可以使用备份将集群恢复到以前的状态。

- 更新单节点 OpenShift Container Platform 集群需要停机，并可包括自动重启。停机时间取决于更新有效负载，如下例所示：
  - 如果更新有效负载包含操作系统更新（需要重启），则停机时间会非常显著，并影响集群管理和用户工作负载。
  - 如果更新包含不需要重启的机器配置更改，则停机时间会减少，并且对集群管理和用户工作负载的影响会减少。在这种情况下，节点排空步骤会通过单节点 OpenShift Container Platform 跳过，因为集群中没有其他节点可以重新调度工作负载。
  - 如果更新有效负载不包含操作系统更新或机器配置更改，则会出现简短的 API 中断并快速解决。



重要

某些条件（如更新的软件包中的错误）可能会导致单个节点在重启后无法重启。在这种情况下，更新不会自动回滚。

其他资源

- 有关哪些机器配置更改需要重启的详情，请参考 了解 Machine Config Operator 中的备注。

8.6. 使用WEB控制台更新集群

如果有可用更新，您可以从Web控制台更新集群。

您可以在客户门户网站的勘误部分找到有关可用 OpenShift Container Platform 公告和更新的信息。

先决条件

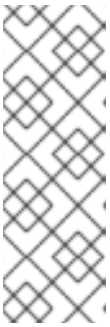
- 使用具有 **admin** 权限的用户登陆到 web 控制台。
- 暂停所有 **MachineHealthCheck** 资源。

流程

1. 在 web 控制台中点击 **Administration → Cluster Settings** 并查看 **Details** 选项卡中的内容。
2. 对于生产环境中的集群，请确保将 **Channel** 设置为您要升级到的版本的正确频道，如 **stable-4.12**。

重要

对于生产环境中的集群，您必须订阅一个 **stable-***, **eus-*** 或 **fast-*** 频道。



注意

当您准备好升级到下一个次版本时，请选择与该次版本对应的频道。声明更新频道后，集群可以更方便地为您的目标版本更新路径。集群可能需要一些时间来评估所有可用的更新，并提供最佳更新建议。更新建议可能会随时间变化，因为它们基于哪些更新选项。

如果您无法看到到目标次版本的更新路径，请保持将集群更新至当前版本的最新补丁版本，直到下一个次版本在路径中可用。

- 如果 **Update 状态** 不是 **Updates available**，则无法升级集群。
- **Select channel** 表示集群正在运行或正在更新的集群版本。

3. 选择要更新到的版本，然后单击 **Save**。
输入频道 **Update Status** 变为**Update to <product-version> in progress**，您可以通过监视 Operator 和节点的进度条来查看集群更新的进度。

注意

如果您要将集群升级到下一个次版本，如从 4.y 升级到 4.(y+1)，建议在部署依赖新功能的工作负载前确认您的节点已升级。任何尚未更新的 worker 节点池都会显示在 **Cluster Settings** 页面。

4. 更新完成后，Cluster Version Operator 会刷新可用更新，检查当前频道中是否有更多可用更新。
   - 如果有可用更新，请继续在当前频道中执行更新，直到您无法再更新为止。
   - 如果没有可用的更新，请将 **Channel** 改为下一个次版本的 **stable-***, **eus-*** 或 **fast-*** 频道，并更新至您在该频道中想要的版本。

   您可能需要执行一些过渡的更新，直到您到达您想要的版本。

8.7. 使用 WEB 控制台更改更新服务器

更改更新服务器是可选的。如果您在本地安装和配置了 OpenShift Update Service (OSUS)，您必须将服务器的 URL 设置为 **upstream**，以便在更新期间使用本地服务器。

流程

1. 导航到 **Administration → Cluster Settings**，点 **version**。
2. 点击 **YAML** 选项卡，然后编辑 **upstream** 参数值：

   输出示例

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

1 **<update-server-url>** 变量指定更新服务器的 URL。

   默认 **upstream** 是 **https://api.openshift.com/api/upgrades_info/v1/graph**。

3. 点击 **Save**。

其他资源

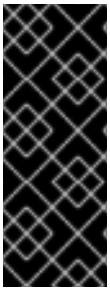
- 了解更新频道和发行版本

第 9 章 使用 CLI 更新集群

您可以使用 OpenShift CLI (**oc**)在 OpenShift Container Platform 集群上执行次要版本和补丁更新。

9.1. 先决条件

- 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。
- 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。
- OpenShift Container Platform 4.12 中删除了对 RHEL7 worker 的支持。升级到 OpenShift Container Platform 4.12 之前，您必须将 RHEL7 worker 替换为 RHEL8 或 RHCOS worker。红帽不支持对 RHEL worker 的 从 RHEL7 到 RHEL8 的原位升级 ; 这些主机必须使用干净的操作系统安装替换。
- 确保之前通过 Operator Lifecycle Manager（OLM）安装的所有 Operator 均更新至其最新频道中的最新版本。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。如需更多信息，请参阅更新安装的 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。
- 确保解决所有 **Upgradeable=False** 条件，以便集群可以更新到下一个次版本。当您有一个或多个无法升级的集群 Operator 时，**Cluster Settings** 页面的顶部会显示一个警报。您仍然可以更新到当前当前使用的次发行版本的下一个可用补丁更新。
- 检查 Kubernetes 1.25 中删除的 API 列表，将任何受影响的组件迁移到使用新的 API 版本，并提供管理员确认。如需更多信息，请参阅准备升级到 OpenShift Container Platform 4.12 。
- 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。



重要

- 当更新无法完成时，Cluster Version Operator(CVO)会在尝试协调更新时报告任何阻塞组件的状态。当前还不支持将集群还原到以前的版本。如果您的更新无法完成，请联系红帽支持。
- 使用 **unsupportedConfigOverrides** 部分修改 Operator 配置不受支持，并可能会阻止集群更新。您必须在更新集群前删除此设置。

其他资源

- 非受管 Operator 的支持策略

9.2. 暂停 MACHINEHEALTHCHECK 资源

在升级过程中，集群中的节点可能会临时不可用。对于 worker 节点，机器健康检查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。

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：

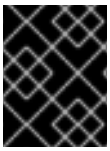
```
$ oc -n openshift-machine-api annotate mhc <mhc-name> cluster.x-k8s.io/paused-
```

9.3. 关于更新单个节点 OPENSHIFT CONTAINER PLATFORM

您可以使用控制台或 CLI 更新或升级单节点 OpenShift Container Platform 集群。

但请注意以下限制：

- 不需要暂停 **MachineHealthCheck** 资源，因为没有其他节点可以执行健康检查。
- 不支持使用 etcd 备份来恢复单节点 OpenShift Container Platform 集群。但是，最好在升级失败时执行 etcd 备份。如果 control plane 健康，您可以使用备份将集群恢复到以前的状态。
- 更新单节点 OpenShift Container Platform 集群需要停机，并可包括自动重启。停机时间取决于更新有效负载，如下例所示：
  - 如果更新有效负载包含操作系统更新（需要重启），则停机时间会非常显著，并影响集群管理和用户工作负载。
  - 如果更新包含不需要重启的机器配置更改，则停机时间会减少，并且对集群管理和用户工作负载的影响会减少。在这种情况下，节点排空步骤会通过单节点 OpenShift Container Platform 跳过，因为集群中没有其他节点可以重新调度工作负载。
  - 如果更新有效负载不包含操作系统更新或机器配置更改，则会出现简短的 API 中断并快速解决。



重要

某些条件（如更新的软件包中的错误）可能会导致单个节点在重启后无法重启。在这种情况下，更新不会自动回滚。

其他资源

- 有关哪些机器配置更改需要重启的详情，请参考 了解 Machine Config Operator 中的备注。

9.4. 使用 CLI 更新集群

您可以使用 OpenShift CLI (**oc**) 来检查和请求集群更新。

您可以在客户门户网站的勘误部分找到有关可用 OpenShift Container Platform 公告和更新的信息。

先决条件

- 安装与更新版本的版本匹配的 OpenShift CLI(**oc**)。
- 使用具有 **cluster-admin** 权限的用户登陆到集群。
- 暂停所有 **MachineHealthCheck** 资源。

流程

1. 查看可用更新，记录下要应用的更新的版本号：

```
$ oc adm upgrade
```

输出示例

```
Cluster version is 4.9.23
Upstream is unset, so the cluster will use an appropriate default.
```

```
Channel: stable-4.9 (available channels: candidate-4.10, candidate-4.9, fast-4.10, fast-4.9,
stable-4.10, stable-4.9, eus-4.10)
Recommended updates:
VERSION IMAGE
4.9.24  quay.io/openshift-release-dev/ocp-
release@sha256:6a899c54dda6b844bb12a247e324a0f6cde367e880b73ba110c056df6d018032
4.9.25  quay.io/openshift-release-dev/ocp-
release@sha256:2eafde815e543b92f70839972f585cc52aa7c37aa72d5f3c8bc886b0fd45707a

4.9.26  quay.io/openshift-release-dev/ocp-
release@sha256:3ccd09dd08c303f27a543351f787d09b83979cd31cf0b4c6ff56cd68814ef6c8

4.9.27  quay.io/openshift-release-dev/ocp-
release@sha256:1c7db78eec0cf05df2cead44f69c0e4b2c3234d5635c88a41e1b922c3bedae16

4.9.28  quay.io/openshift-release-dev/ocp-
release@sha256:4084d94969b186e20189649b5affba7da59f7d1943e4e5bc7ef78b981eafb7a8

4.9.29  quay.io/openshift-release-dev/ocp-
release@sha256:b04ca01d116f0134a102a57f86c67e5b1a3b5da1c4a580af91d521b8fa0aa6ec

4.9.31  quay.io/openshift-release-dev/ocp-
release@sha256:2a28b8ebb53d67dd80594421c39e36d9896b1e65cb54af81fbb86ea9ac3bf2d7
4.9.32  quay.io/openshift-release-dev/ocp-
release@sha256:ecdb6d0df547b857eaf0edb5574ddd64ca6d9aff1fa61fd1ac6fb641203bedfa
```

注意

- 如果没有可用的更新，则支持的更新可能仍可用。如需更多信息，请参阅*更新条件更新路径*。
- 有关如何执行 EUS 到 **EUS** 频道升级的详情，请参阅附加资源部分中列出的*准备执行 EUS 升级*页面。

2. 根据您的机构要求，设置适当的升级频道。例如，您可以将频道设置为 **stable-4.13** 或 **fast-4.13**。有关频道的更多信息，请参阅在额外资源项中的*了解更新频道和发行版本*。

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

例如，要将频道设置为 **stable-4.12** ：

```
$ oc adm upgrade channel stable-4.12
```

重要

对于生产环境中的集群，您必须订阅一个 **stable-***、**eus-*** 或 **fast-*** 频道。

注意

当您准备好升级到下一个次版本时，请选择与该次版本对应的频道。声明更新频道后，集群可以更方便地为您的目标版本更新路径。集群可能需要一些时间来评估所有可用的更新，并提供最佳更新建议。更新建议可能会随时间变化，因为它们基于哪些更新选项。

如果您无法看到到目标次版本的更新路径，请保持将集群更新至当前版本的最新补丁版本，直到下一个次版本在路径中可用。

3. 应用更新：

   - 要更新到最新版本：

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

   - 要更新到一个特定版本：

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

     1 1 **<version>** 是从 **oc adm upgrade** 命令的输出中获取的更新版本。

4. 查看 Cluster Version Operator 的状态：

   ```
   $ oc adm upgrade
   ```

5. 更新完成后，可以通过以下方法确认集群已更新为新版本：

   ```
   $ oc get clusterversion
   ```

   输出示例

   ```
   Cluster version is <version>

   Upstream is unset, so the cluster will use an appropriate default.
   Channel: stable-4.10 (available channels: candidate-4.10, candidate-4.11, eus-4.10, fast-4.10,
   fast-4.11, stable-4.10)

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

6. 如果您要将集群升级到下一个次版本，如 X.y 升级到 X. (y+1)，建议在部署依赖新功能的工作负载前确认您的节点已升级：

   ```
   $ oc get nodes
   ```

   输出示例

   ```
   NAME                         STATUS   ROLES    AGE   VERSION
   ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.25.0
   ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.25.0
   ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.25.0
   ```

```
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.25.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.25.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.25.0
```

其他资源

- 通过一个条件更新路径进行更新
- 准备执行 EUS 更新
- 了解更新频道和发行版本

9.5. 根据一个有条件的升级路径进行升级

您可以使用 Web 控制台或 OpenShift CLI(**oc**)根据一个有条件的升级路径进行升级。当一个有条件更新对于您的集群不推荐进行时，您可以使用 OpenShift CLI(**oc**)4.10 或更高版本来根据一个有条件的路径进行升级。

流程

1. 因为存在风险而不推荐的更新，您可以使用以下命令查看它的描述信息：

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

2. 如果集群管理员已评估了潜在的已知风险，并确定这些风险对于当前集群是可接受的，管理员可以执行以下命令来忽略这些安全保护并继续更新：

```
$ oc adm upgrade --allow-not-recommended --to <version> <.>
```

<.> **<version>**是从上一个命令输出中获取的被支持但不推荐的更新版本。

其他资源

- 了解更新频道和发行版本

9.6. 使用 CLI 更改更新服务器

更改更新服务器是可选的。如果您在本地安装和配置了 OpenShift Update Service (OSUS)，您必须将服务器的 URL 设置为 **upstream**，以便在更新期间使用本地服务器。**upstream** 的默认值是 **https://api.openshift.com/api/upgrades_info/v1/graph**。

流程

- 更改集群版本中的 **upstream** 参数值：

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

<update-server-url> 变量指定更新服务器的 URL。

输出示例

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

-

第 10 章 执行 CANARY ROLLOUT 更新

*金丝雀更新*是一种更新策略，其中 worker 节点更新以离散的、阶段阶段执行，而不是同时更新所有 worker 节点。这个策略在以下情况下很有用：

- 您需要一个受控的 worker 节点更新推出，以确保任务关键型应用程序在整个更新过程中仍然可用，即使更新过程会导致应用程序失败。
- 您需要更新一小部分 worker 节点，在一个时间段内评估集群和工作负载健康状况，然后更新剩余的节点。
- 您可能希望将通常需要主机重新引导的 worker 节点更新放入较小的定义的维护窗口（不可能一次使用大型维护窗口来更新整个集群）。

在这些情况下，您可以创建多个自定义机器配置池 (MCP)，以防止某些 worker 节点在更新集群时进行更新。在更新剩余的集群后，您可以在适当的时间批量更新这些 worker 节点。

10.1. 金丝雀更新策略示例

以下示例描述了一个金丝雀更新策略，其中有一个有 10% 的 100 个节点的集群，您有没有超过 4 小时的维护窗口，您知道排空并重启 worker 节点所需的时间不超过 8 分钟。

注意

以上值是仅限示例。排空节点所需的时间可能会因工作负载等因素而异。

定义自定义机器配置池

要将 worker 节点的更新组织到不同的独立阶段，您可以从定义以下 MCP 开始：

- **workerpool-canary**，有 10 个节点
- **workerpool-A**，有 30 个节点
- **workerpool-B**，有 30 个节点
- **workerpool-C**，有 30 个节点

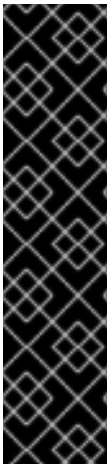
更新 Canary worker 池

在第一个维护窗口中，您将暂停 **workerpool-A**、**workerpool-B** 和 **workerpool-C** 的 MCP，然后启动集群更新。这将更新在 OpenShift Container Platform 上运行的组件，以及作为取消暂停的 **workerpool-canary** MCP 一部分的 10 个节点。其他三个 MCP 不会更新，因为它们已暂停。

确定是否继续剩余的 worker 池更新

如果出于某种原因，您确定集群或工作负载健康受到 **workerpool-canary** 更新的负面影响，那么在分析完问题前，您会在保持足够容量的同时，对那个池中的所有节点进行 cordon 和 drain 操作。当一切按预期工作时，您可以在决定取消暂停前评估集群和工作负载健康状况，从而更新 **workerpool-A**、**workerpool-B** 和 **workerpool-C** 在每个额外的维护窗口中连续工作。

使用自定义 MCP 管理 worker 节点更新提供了灵活性，但它可能是一个耗时的过程，需要您执行多个命令。这种复杂性可能会导致错误可能会影响整个集群。建议您仔细考虑您的组织需求，并在开始之前仔细规划流程的实施。

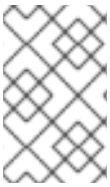


重要

暂停机器配置池可防止 Machine Config Operator 在关联的节点上应用任何配置更改。暂停 MCP 还可以防止任何自动轮转的证书被推送到关联的节点，包括自动轮转 **kube-apiserver-to-kubelet-signer** CA 证书。

当 **kube-apiserver-to-kubelet-signer** CA 证书过期且 MCO 尝试自动更新证书时，MCO 无法将新轮转的证书推送到这些节点。这会导致多个 **oc** 命令失败，包括 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。如果在轮转证书时，如果 MCP 被暂停，则 OpenShift Container Platform Web 控制台的 Alerting UI 中收到警报。

在暂停 MCP 时应该非常小心，需要仔细考虑 **kube-apiserver-to-kubelet-signer** CA 证书过期的问题，且仅在短时间内暂停。



注意

不建议将 MCP 更新至不同的 OpenShift Container Platform 版本。例如，请勿将一个 MCP 从 4.y.10 更新至 4.y.11，另一个更新为 4.y.12。这个场景还没有被测试，可能会导致未定义的集群状态。

10.2. 关于 CANARY ROLLOUT 更新过程和 MCP

在 OpenShift Container Platform 中，节点不会被单独考虑。相反，它们被分组到机器配置池中 (MCP)。默认情况下，OpenShift Container Platform 集群中的节点分组到两个 MCP 中：一个用于 control plane 节点，一个用于 worker 节点。OpenShift Container Platform 更新会同时影响所有 MCP。

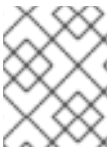
在更新过程中，如果指定了最大数字，Machine Config Operator (MCO) 会排空并封锁 MCP 中的最多为 **maxUnavailable** 中指定的数量的节点。默认情况下，**maxUnavailable** 设置为 **1**。排空节点取消调度节点上的所有 pod，并将该节点标记为不可调度。

节点排空后，Machine Config Daemon 应用一个新的机器配置，其中包括更新操作系统 (OS)。更新操作系统需要主机重新引导。

使用自定义机器配置池

要防止特定节点被更新，您可以创建自定义 MCP。因为 MCO 不会在暂停的 MCP 中更新节点，所以您可以在启动集群更新前暂停包含您不想更新的节点的 MCP。

使用一个或多个自定义 MCP 可以让您更好地控制您更新 worker 节点的顺序。例如，在更新第一个 MCP 中的节点后，您可以验证应用程序兼容性，然后逐步将其余节点更新至新版本。



注意

为确保 control plane 的稳定性，不支持从 control plane 节点创建自定义 MCP。Machine Config Operator (MCO) 会忽略为 control plane 节点创建的任何自定义 MCP。

使用自定义机器配置池时的注意事项

根据工作负载部署拓扑，请仔细考虑您创建的 MCP 数以及每个 MCP 中的节点数。例如，如果必须将更新适合特定的维护窗口，您必须了解在给定窗口中可以更新多少个节点 OpenShift Container Platform。这个数字取决于您具体的集群和工作负载特性。

您还必须考虑集群中有多少额外容量，以确定自定义 MCP 的数量以及每个 MCP 中的节点数。当应用程序无法在更新的节点上按预期工作时，您可以对池中那些节点进行 cordon 和 drain 操作，这会将应用 pod 移到其他节点上。但是，您必须确定剩余的 MCP 中的可用节点是否可以为您的应用程序提供足够的服务质量 (QoS)。

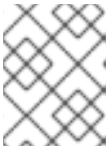
注意

您可以将这个更新过程与所有记录的 OpenShift Container Platform 更新过程一起使用。但是，该过程不适用于使用 Ansible playbook 进行更新的 Red Hat Enterprise Linux (RHEL) 机器。

10.3. 关于执行 CANARY ROLLOUT 更新

下列步骤概述了金丝雀 rollout 更新过程的高级工作流：

1. 根据 worker 池创建自定义机器配置池 (MCP)。



注意

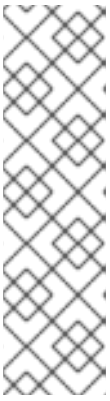
您可以更改 MCP 中的 **maxUnavailable** 设置，以指定在任意给定时间可以更新的机器的百分比或数量。默认值为 **1**。

2. 将节点选择器添加到自定义 MCP。对于您不想与剩余的集群同时更新的每个节点，请向节点添加匹配的标签。该标签将节点与 MCP 相关联。

重要

不要从节点中删除默认 worker 标签。节点必须具有 role 标签才能在集群中正常工作。

3. 在更新过程中暂停您不想更新的 MCP。



注意

暂停 MCP 也会暂停 **kube-apiserver-to-kubelet-signer** 自动 CA 证书轮转。在自安装日期起的 292 天生成新 CA 证书，旧证书将从安装日期的 365 天后删除。请参阅红帽 OpenShift 4 中的了解 CA 证书自动续订，以了解您在下一次自动 CA 证书轮转前的时间。

确保发生 CA 证书轮转时取消暂停池。如果 MCP 暂停，MCO 无法将新轮转的证书推送到这些节点。这会导致集群降级，并在多个 **oc** 命令中造成失败，包括 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。如果在轮转证书时，如果 MCP 被暂停，则 OpenShift Container Platform Web 控制台的 Alerting UI 中收到警报。

4. 执行集群更新。更新过程更新没有暂停的 MCP，包括 control plane 节点。

5. 在更新的节点上测试应用程序，以确保它们按预期工作。

6. 取消暂停剩余的 MCP，等待池中的节点完成更新，并在这些节点上测试应用程序。重复此过程，直到所有 worker 节点都已更新。

7. 可选：从更新的节点中删除自定义标签并删除自定义 MCP。

10.4. 创建机器配置池来执行 CANARY ROLLOUT 更新

要执行 Canary rollout 更新，您必须首先创建一个或多个自定义机器配置池 (MCP)。

流程

1. 运行以下命令列出集群中的 worker 节点：

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

输出示例

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

2. 对于您要延迟的每个节点，运行以下命令为节点添加自定义标签：

```
$ oc label node <node_name> node-role.kubernetes.io/<custom_label>=
```

例如：

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary=
```

输出示例

```
node/ci-ln-gtrwm8t-f76d1-spbl7-worker-a-xk76k labeled
```

3. 创建新的 MCP：

   a. 创建 MCP YAML 文件：

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

   (1) 为 MCP 指定名称。

   (2) 指定 **worker** 和自定义 MCP 名称。

   (3) 指定添加到此池中的自定义标签。

   b. 运行以下命令来创建 **MachineConfigPool** 对象：

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

输出示例

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

4. 运行以下命令，查看集群中的 MCP 列表及其当前状态：

```
$ oc get machineconfigpool
```

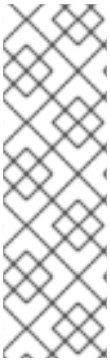
输出示例

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

创建新的机器配置池 **workerpool-canary**，机器计数中会显示您添加自定义标签的节点数量。worker MCP 机器数会减少相同的数字。更新机器数可能需要几分钟时间。在本例中，一个节点已从 **worker** MCP 移到 worker **pool-canary** MCP。

10.5. 暂停机器配置池

创建自定义机器配置池 (MCP) 后，您将暂停这些 MCP。暂停 MCP 可防止 Machine Config Operator (MCO) 更新与该 MCP 关联的节点。



注意

暂停 MCP 也会暂停 **kube-apiserver-to-kubelet-signer** 自动 CA 证书轮转。在自安装日期起的 292 天生成新 CA 证书，旧证书将从安装日期的 365 天后删除。请参阅红帽 OpenShift 4 中的了解 CA 证书自动续订，以了解您在下一次自动 CA 证书轮转前的时间。

确保发生 CA 证书轮转时取消暂停池。如果 MCP 暂停，MCO 无法将新轮转的证书推送到这些节点。这会导致集群降级，并在多个 **oc** 命令中造成失败，包括 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。如果在轮转证书时，如果 MCP 被暂停，则 OpenShift Container Platform Web 控制台的 Alerting UI 中收到警报。

流程

1. 运行以下命令修补您要暂停的 MCP：

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

例如：

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

输出示例

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

10.6. 执行集群更新

在机器配置池 (MCP) 进入就绪状态后，您可以执行集群更新。根据您的集群，请查看以下更新方法之一：

- 使用 Web 控制台更新集群
- 使用 CLI 更新集群

集群更新后，您可以一次开始取消暂停 MCP。

10.7. 取消暂停机器配置池

OpenShift Container Platform 更新完成后，一次取消暂停您的自定义机器配置池 (MCP)。取消暂停 MCP 允许 Machine Config Operator (MCO) 更新与该 MCP 关联的节点。

流程

1. 对您要取消暂停的 MCP 进行补丁：

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

例如：

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

输出示例

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

2. 可选：使用以下选项之一检查更新的进度：
   a. 点 **Administration → Cluster settings** 从 web 控制台检查进度。
   b. 运行以下命令检查进度：

```
$ oc get machineconfigpools
```

3. 在更新的节点上测试您的应用，以确保它们按预期工作。
4. 一次对任何其他暂停的 MCP 重复此过程。

注意

如果应用程序出现故障，如应用程序未在更新的节点上工作，您可以对池中的节点进行 cordon 和 drain 操作，这会将应用 pod 移到其他节点，以帮助维护应用程序的服务质量。第一个 MCP 不应大于过量容量。

10.8. 将节点移到原始机器配置池中

在自定义机器配置池(MCP)的节点上更新并验证应用程序后，删除添加到节点的自定义标签，将节点移回到其原始 MCP。

重要

节点必须具有角色才能在集群中正常工作。

流程

1. 对于自定义 MCP 中的每个节点，运行以下命令来从节点中删除自定义标签：

```
$ oc label node <node_name> node-role.kubernetes.io/<custom_label>-
```

例如：

```
$ oc label node ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz node-
role.kubernetes.io/workerpool-canary-
```

输出示例

```
node/ci-ln-0qv1yp2-f76d1-kl2tq-worker-a-j2ssz labeled
```

Machine Config Operator 将节点移回到原始 MCP，并将节点与 MCP 配置协调。

2. 要确保节点已从自定义 MCP 中删除，请运行以下命令来查看集群中的 MCP 列表及其当前状态：

```
$ oc get mcp
```

输出示例

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

当节点从自定义 MCP 中删除并移回原始 MCP 时，可能需要几分钟时间来更新机器计数。在这个示例中，将一个节点从删除的 **workerpool-canary MCP** 移到 **worker** MCP。

3. 可选：运行以下命令来删除自定义 MCP：

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

第 11 章 使用 BOOTUPD 更新 RHCOS 节点上的引导装载程序

要使用 **bootupd** 更新 RHCOS 节点上的引导装载程序，您必须在 RHCOS 机器上手动运行 **bootupctl update** 命令，或使用 **systemd** 单元提供机器配置。

与 **grubby** 或其他引导装载程序工具不同，boot **upd** 不管理内核空间配置，如传递内核参数。要配置内核参数，请参阅 向节点添加内核参数。

注意

您可以使用 **bootupd** 更新引导装载程序以防止 BootHole 漏洞。

11.1. 手动更新引导装载程序

您可以使用 **bootupctl** 命令行工具手动检查系统状态并更新引导装载程序。

1. 检查系统状态：

```
# bootupctl status
```

x86_64的输出示例

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

aarch64 的输出示例

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

2. OpenShift Container Platform 集群最初安装在版本 4.4 及更早的版本中，需要明确的采用阶段。
如果系统状态为 **Adoptable**，请执行采用：

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

输出示例

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

3. 如果有可用更新，请应用更新以便在下次重启时使更改生效：

```
# bootupctl update
```

输出示例

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

11.2. 通过机器配置自动更新引导装载程序

使用 **bootupd** 自动更新引导装载程序的另一种方法是创建一个 systemd 服务单元，该单元将在每次引导时更新引导装载程序。单元将在引导过程中运行 **bootupctl update** 命令，并通过机器配置在节点上安装。

注意

这个配置不会被默认启用，因为更新操作的意外中断可能会导致无法引导的节点。如果启用此配置，请确保在引导装载程序更新进行过程中避免中断节点。引导装载程序更新操作通常很快完成，因此风险较低。

1. 创建一个 Butane 配置文件 **99-worker-bootupctl-update.bu**，包括 **bootupctl-update.service** systemd 单元的内容。

注意

有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

输出示例

```
variant: openshift
version: 4.12.0
metadata:
  name: 99-worker-chrony (1)
  labels:
    machineconfiguration.openshift.io/role: worker (2)
systemd:
  units:
  - name: bootupctl-update.service
    enabled: true
    contents: |
      [Unit]
      Description=Bootupd automatic update

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

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

(1) (2) 在 control plane 节点上，在这两个位置中将 **master** 替换为 **worker**。

2. 使用 Butane 生成 **MachineConfig** 对象文件 **99-worker-bootupctl-update.yaml**，其中包含要传送到节点的配置：

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

3. 使用以下两种方式之一应用配置：

- 如果集群还没有运行，在生成清单文件后，将 **MachineConfig** 对象文件添加到 **<installation_directory>/openshift** 目录中，然后继续创建集群。

installation_directory/openshift 目录中，然后在该文件中

- 如果集群已在运行，请应用该文件：

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

第 12 章 更新包含使用 RHEL 的计算（COMPUTE）系统的集群

您可以在 OpenShift Container Platform 集群上执行次版本和补丁更新。如果您的集群包含 Red Hat Enterprise Linux (RHEL) 机器，则必须执行额外的步骤来更新这些机器。

12.1. 先决条件

- 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。
- 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。
- OpenShift Container Platform 4.12 中删除了对 RHEL7 worker 的支持。升级到 OpenShift Container Platform 4.12 之前，您必须将 RHEL7 worker 替换为 RHEL8 或 RHCOS worker。红帽不支持对 RHEL worker 的 从 RHEL7 到 RHEL8 的原位升级；这些主机必须使用干净的操作系统安装替换。
- 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。

其他资源

- 非受管 Operator 的支持策略

12.2. 使用WEB控制台更新集群

如果有可用更新，您可以从Web控制台更新集群。

您可以在客户门户网站的勘误部分找到有关可用 OpenShift Container Platform 公告和更新的信息。

先决条件

- 使用具有 **admin** 权限的用户登陆到 web 控制台。
- 暂停所有 **MachineHealthCheck** 资源。

流程

1. 在 web 控制台中点击 **Administration → Cluster Settings** 并查看 **Details** 选项卡中的内容。
2. 对于生产环境中的集群，请确保将 **Channel** 设置为您要升级到的版本的正确频道，如 **stable-4.12**。

重要

对于生产环境中的集群，您必须订阅一个 **stable-***, **eus-*** 或 **fast-*** 频道。

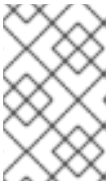
注意

当您准备好升级到下一个次版本时，请选择与该次版本对应的频道。声明更新频道后，集群可以更方便地为您的目标版本更新路径。集群可能需要一些时间来评估所有可用的更新，并提供最佳更新建议。更新建议可能会随时间变化，因为它们基于哪些更新选项。

如果您无法看到到目标次版本的更新路径，请保持将集群更新至当前版本的最新补丁版本，直到下一个次版本在路径中可用。

- 如果 **Update 状态** 不是 **Updates available**，则无法升级集群。
- **Select channel** 表示集群正在运行或正在更新的集群版本。

3. 选择要更新到的版本，然后单击 **Save**。
输入频道 **Update Status** 变为**Update to <product-version> in progress**，您可以通过监视 Operator 和节点的进度条来查看集群更新的进度。



注意

如果您要将集群升级到下一个次版本，如从 4.y 升级到 4.(y+1)，建议在部署依赖新功能的工作负载前确认您的节点已升级。任何尚未更新的 worker 节点池都会显示在 **Cluster Settings** 页面。

4. 更新完成后，Cluster Version Operator 会刷新可用更新，检查当前频道中是否有更多可用更新。
   - 如果有可用更新，请继续在当前频道中执行更新，直到您无法再更新为止。
   - 如果没有可用的更新，请将 **Channel** 改为下一个次版本的 **stable-***, **eus-*** 或 **fast-*** 频道，并更新至您在该频道中想要的版本。

   您可能需要执行一些过渡的更新，直到您到达您想要的版本。

注意

当您更新包含有 Red Hat Enterprise Linux (RHEL) worker 机器的集群时，这些 worker 会在更新过程中暂时不可用。当集群进入 **NotReady** 状态时，您需要针对每个 RHEL 机器运行升级 playbook 以完成更新。

12.3. 可选：添加 HOOK 以在RHEL系统上执行ANSIBLE任务

在OpenShift Container Platform更新期间，您可以使用*hook*在RHEL计算系统上运行Ansible任务。

12.3.1. 在升级过程中使用 Ansible hook

更新OpenShift Container Platform时，可以使用*hook*在执行特定操作时在Red Hat Enterprise Linux（RHEL）节点上运行自定义的任务。您可以使用 hook 提供定义了在执行特定任务之前或之后要运行的任务的文件。在OpenShift Container Platform集群中更新RHEL计算节点时，可以使用 hook 来验证或修改自定义的基础架构。

因为当 hook 失败时，这个操作将会失败，所以您必须把 hook 设计为可以多次运行，并且获得相同的结果。

hook 有以下限制： - hook 没有已定义或版本化的界面。它们可以使用内部的**openshift-ansible**变量，但这些变量可能会在将来的OpenShift Container Platform版本被修改或删除。 - hook 本身没有错误处理机制，因此 hook 中的错误会暂停更新过程。如果出现错误，则需要解决相关的问题，然后再次进行升级。

12.3.2. 配置Ansible inventory文件以使用 hook

您可以在 **hosts** inventory 文件的**all:vars** 部分中定义 Red Hat Enterprise Linux（RHEL）compute 机器（也称为 worker 机器）更新时使用的 hook 。

先决条件

- 您可以访问用于添加RHEL compute 系统集群的计算机。您必须有访问定义RHEL系统的**hosts** Ansible 清单文件的权限。

流程

1. 在设计了 hook 后，创建一个YAML文件，为其定义Ansible任务。此文件必须是一组任务，不能是一个 playbook，如以下示例所示：

```
---
# Trivial example forcing an operator to acknowledge the start of an upgrade
# file=/home/user/openshift-ansible/hooks/pre_compute.yml

- name: note the start of a compute machine update
  debug:
    msg: "Compute machine upgrade of {{ inventory_hostname }} is about to start"

- name: require the user agree to start an upgrade
  pause:
    prompt: "Press Enter to start the compute machine update"
```

2. 修改 **hosts** Ansible inventory 文件来指定 hook 文件。hook 文件作为参数值在 **[all:vars]** 部分指定。如下所示：

   清单文件中的 hook 定义示例

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

   为了避免歧义，请在其定义中使用 hook 文件的绝对路径而不要使用相对路径。

12.3.3. RHEL计算系统可用的 hook

在更新OpenShift Container Platform集群中的Red Hat Enterprise Linux（RHEL）compute 系统时，可以使用以下 hook。

Hook 名	描述

Hook 名	描述
openshift_node_pre_cordon_hook	• 在每个节点被封锁（cordon）**之前**运行。 • 此 hook 以串行方式针对**每个节点**运行。 • 如果某个任务必须针对其他主机运行，则该任务必须使用**delegate_to**或**local_action**。
openshift_node_pre_upgrade_hook	• 在每个节点被封锁**后**，且被更新**前**运行。 • 此 hook 以串行方式针对**每个节点**运行。 • 如果某个任务必须针对其他主机运行，则该任务必须使用**delegate_to**或**local_action**。
openshift_node_pre_uncordon_hook	• 在每个节点被更新**后**，且被取消封锁（uncordon）**前**运行。 • 此 hook 以串行方式针对**每个节点**运行。 • 如果某个任务必须针对其他主机运行，则该任务必须使用**delegate_to**或**local_action**。
openshift_node_post_upgrade_hook	• 每个节点未被取消封锁**后**运行。这是**最后一个**节点更新操作。 • 此 hook 以串行方式针对**每个节点**运行。 • 如果某个任务必须针对其他主机运行，则该任务必须使用**delegate_to**或**local_action**。

12.4. 更新集群中的RHEL COMPUTE 系统

在对集群进行更新后，必须更新集群中的Red Hat Enterprise Linux（RHEL）compute 系统。

重要

RHEL 计算机器支持 Red Hat Enterprise Linux (RHEL) 版本 8.6 及更新的版本。

如果您使用 RHEL 作为操作系统，您还可以将计算机器更新至 OpenShift Container Platform 的另一个次要版本。当执行次版本更新时，您不需要排除 RHEL 中的任何 RPM 软件包。

重要

您无法将 RHEL 7 计算机器升级到 RHEL 8。您必须部署新的 RHEL 8 主机，并且应该删除旧的 RHEL 7 主机。

先决条件

- 已更新了集群。

重要

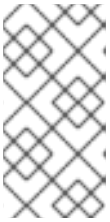
因为 RHEL 机器需要集群生成的资产才能完成更新过程，所以您必须在更新其中的 RHEL worker 机器前更新集群。

- 您可以访问用于将 RHEL 计算机器添加到集群的本地机器。您必须有权访问定义了 RHEL 系统及 **upgrade** playbook 的 **hosts** Ansible 清单文件。
- 对于次版本的更新，RPM 存储库使用的是集群上运行的相同版本的 OpenShift Container Platform。

流程

1. 停止并禁用主机上的防火墙：

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



注意

默认情况下，使用 "Minimal" 安装选项的基础操作系统 RHEL 启用 firewalld 保护。在主机上启用了 firewalld 服务会阻止您访问 worker 上的 OpenShift Container Platform 日志。如果您希望继续访问 worker 上的 OpenShift Container Platform 日志，以后不要启用 firewalld。

2. 启用 OpenShift Container Platform 4.12 所需的存储库：

   a. 在运行 Ansible playbook 的机器上，更新所需的存储库：

```
# subscription-manager repos --disable=rhocp-4.11-for-rhel-8-x86_64-rpms \
                             --disable=ansible-2.9-for-rhel-8-x86_64-rpms \
                             --enable=rhocp-4.12-for-rhel-8-x86_64-rpms
```

重要

自 OpenShift Container Platform 4.11 起，只有 RHEL 8 提供 Ansible playbook。如果 RHEL 7 系统用作 OpenShift Container Platform 4.10 Ansible playbook 的主机，您必须将 Ansible 主机升级到 RHEL 8，或者在 RHEL 8 系统中创建新的 Ansible 主机，并从旧的 Ansible 主机复制清单。

   b. 在运行 Ansible playbook 的机器上，更新 Ansible 软件包：

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

c. 在运行 Ansible playbook 的机器上，更新所需的软件包，包括 **openshift-ansible**：

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

d. 在每个 RHEL 计算节点上，更新所需的软件仓库：

```
# subscription-manager repos --disable=rhocp-4.11-for-rhel-8-x86_64-rpms \
                             --enable=rhocp-4.12-for-rhel-8-x86_64-rpms
```

3. 更新 RHEL worker 机器：

a. 检查 **/<path>/inventory/hosts** 中的 Ansible 清单文件并更新其内容，以便 RHEL 8 机器列在 **[workers]** 部分中，如下例所示：

```
[all:vars]
ansible_user=root
#ansible_become=True

openshift_kubeconfig_path="~/.kube/config"

[workers]
mycluster-rhel8-0.example.com
mycluster-rhel8-1.example.com
mycluster-rhel8-2.example.com
mycluster-rhel8-3.example.com
```

b. 进入 **openshift-ansible** 目录：

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

c. 运行 **upgrade** playbook:

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

1 对于**<path>**，指定您创建的Ansible库存文件的路径。

注意

upgrade playbook 仅升级 OpenShift Container Platform 软件包。它不会更新操作系统软件包。

4. 更新完所有 worker 后，确认所有集群节点已更新至新版本：

```
# oc get node
```

输出示例

```
NAME                    STATUS                     ROLES    AGE    VERSION
mycluster-control-plane-0   Ready                      master   145m   v1.25.0
mycluster-control-plane-1   Ready                      master   145m   v1.25.0
mycluster-control-plane-2   Ready                      master   145m   v1.25.0
mycluster-rhel8-0           Ready                      worker   98m    v1.25.0
```

```
mycluster-rhel8-1    Ready    worker   98m   v1.25.0
mycluster-rhel8-2    Ready    worker   98m   v1.25.0
mycluster-rhel8-3    Ready    worker   98m   v1.25.0
```

5. 可选：更新 **upgrade** playbook 没有更新的操作系统软件包。要更新不在 4.12 中的软件包，请使用以下命令：

```
# yum update
```

注意

如果您使用安装 4.12 时所用的相同 RPM 存储库，则不需要排除 RPM 软件包。

第 13 章 在断开连接的环境中更新集群

13.1. 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的环境。因此，您必须在 registry 中填充安装镜像。如果您的 registry 主机无法同时访问互联网和集群，您可以将镜像镜像到与这个环境断开连接的文件系统中，然后使用主机或可移动介质填补该空白。如果本地容器 registry 和集群连接到镜像 registry 的主机，您可以直接将发行镜像推送到本地 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。

13.1.1. 镜像 OpenShift Container Platform 镜像存储库

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。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 镜像存储库

13.1.2. 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

13.1.3. 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

13.2. 镜像 OPENSHIFT CONTAINER PLATFORM 镜像存储库

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。

13.2.1. 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 mirror registry for Red Hat OpenShift 会包括在 OpenShift Container Platform 订阅中。*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镜像 OpenShift Container Platform 容器镜像。

13.2.2. 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。

13.2.2.1. 通过下载二进制文件安装 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**，则无法使用 OpenShift Container Platform 4.12 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升级集群，请安装您要升级到的 **oc** 版本。

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. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 **产品变体** 下拉列表中选择架构。
3. 从 **版本** 下拉列表中选择适当的版本。
4. 点 **OpenShift v4.12 Linux Client** 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>
```

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. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 **版本** 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.12 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>
```

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. 导航到红帽客户门户网站上的 OpenShift Container Platform 下载页面。
2. 从 **版本** 下拉列表中选择适当的版本。
3. 点 **OpenShift v4.12 macOS Client** 条目旁的 **Download Now** 来保存文件。

注意

对于 macOS arm64，请选择 **OpenShift v4.12 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

- 安装 OpenShift CLI 后，可以使用 **oc** 命令：

```
$ oc <command>
```

其他资源

- 安装和使用 CLI 插件

13.2.2.2. 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，允许将红帽的镜像镜像到您的镜像环境中。

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。

警告

此过程需要您可以对镜像 registry 上的容器镜像 registry 进行写操作，并将凭证添加到 registry pull secret。

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。

流程

在安装主机上完成以下步骤：

1. 下载您的 **registry.redhat.io** pull secret（从Red Hat OpenShift Cluster Manager） 。
2. 以 JSON 格式创建您的 pull secret 副本：

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

1 指定到存储 pull secret 的文件夹的路径，以及您创建的 JSON 文件的名称。

该文件类似于以下示例：

```
{
  "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. 可选：如果使用 oc-mirror 插件，请将文件保存为 **~/.docker/config.json** 或 **$XDG_RUNTIME_DIR/containers/auth.json** ：
   a. 如果 **.docker** 或 **$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>
```

其中 **<directory_name>** 是 **~/.docker** 或 **$XDG_RUNTIME_DIR/containers**，& **lt;auth_file&** gt; 则是 **config.json** 或 **auth.json**。

4. 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 (1)
BGVtbYk3ZHAtqXs=
```

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

5. 编辑 JSON 文件并添加描述 registry 的部分：

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

(1) 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**

(2) 使用 **<credentials>** 为您的镜像 registry 指定 base64 编码的用户名和密码。

该文件类似于以下示例：

```
{
  "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"
    }
  }
}
```

13.2.3. 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。

13.2.3.1. 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。

重要

当使用 oc-mirror CLI 插件填充镜像 registry 时，必须使用 oc-mirror 工具对镜像 registry 进行进一步的更新。

13.2.3.2. oc-mirror 兼容性和支持

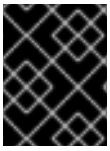
oc-mirror 插件支持为 OpenShift Container Platform 版本 4.9 及之后的版本的镜像 OpenShift Container Platform 有效负载镜像和 Operator 目录。

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。

13.2.3.3. 关于镜像 registry

您可以将 OpenShift Container Platform 安装和后续的产品更新镜像(mirror)到支持 Docker v2-2 （如 Red Hat Quay）的容器镜像（如 Red Hat Quay）的容器镜像。如果您无法访问大型容器 registry，可以使用 *Red Hat OpenShift 的镜像 registry*，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。



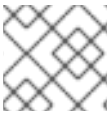
重要

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

- 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。

13.2.3.4. 安装 oc-mirror OpenShift CLI 插件

要使用 oc-mirror OpenShift CLI 插件来镜像 registry 镜像，您必须安装插件。如果您在一个完全断开连接的环境中镜像镜像集，请确保在具有互联网访问的主机上的 oc-mirror 插件以及可访问镜像 registry 的断开连接的环境中安装 oc-mirror 插件。

先决条件

- 已安装 OpenShift CLI（**oc**）。

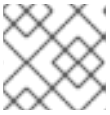
流程

1. 下载 oc-mirror CLI 插件。
   a. 进入到 OpenShift Cluster Manager Hybrid Cloud Console 的 Downloads 页面。
   b. 在 **OpenShift disconnected 安装工具**部分下，点 **Download** for **OpenShift Client(oc)mirror 插件** 并保存该文件。
2. 解压归档：

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

3. 如有必要，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. 通过将文件放在 **PATH** 中，例如 **/usr/local/bin**，安装 oc-mirror CLI 插件：

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

验证

- 运行 **oc mirror help** 来验证插件是否已成功安装：

```
$ oc mirror help
```

13.2.3.5. 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 example.com/mirror/oc-mirror-metadata > imageset-config.yaml (1)
```

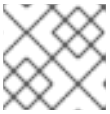
(1) 将 **example.com/mirror/oc-mirror-metadata** 替换为存储后端的 registry 的位置。

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-4.12                                    4
    type: ocp
  graph: true                                            5
operators:
- catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12   6
  packages:
  - name: serverless-operator                            7
    channels:
    - name: stable                                       8
additionalImages:
- name: registry.redhat.io/ubi8/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，除非您使用技术预览 OCI 功能。
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。

如需完整的参数列表，请参阅 *Image set configuration parameters*；对于不同的镜像用例，请参阅 *Image set configuration examples*。

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
- 关于 OpenShift Update 服务

13.2.3.6. 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。

以下流程假设您已设置了镜像 registry。

13.2.3.6.1. 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。

13.2.3.6.1.1. 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中的镜像镜像到指定的 registry:

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

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。

13.2.3.6.2. 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。

13.2.3.6.2.1. 从镜像镜像到磁盘

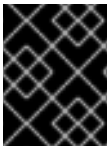
您可以使用 oc-mirror 插件生成镜像集，并将内容保存到磁盘。然后，生成的镜像集可以转移到断开连接的环境中，并镜像到目标 registry。

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。

13.2.3.6.2.2. 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。

13.2.3.7. 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)使用 **CatalogSource** 资源检索有关镜像 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。

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。
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
```

13.2.3.8. 保持镜像 registry 内容更新

使用初始镜像集填充目标镜像 registry 后，您必须定期更新它，使其具有最新的内容。如果可能，您可以设置 cron 任务以定期更新镜像 registry。

根据需要更新您的镜像设置配置，以添加或删除 OpenShift Container Platform 和 Operator 版本。从镜像 registry 中修剪移除的镜像。

13.2.3.8.1. 关于更新您的镜像 registry 内容

当您再次运行 oc-mirror 插件时，它会生成一个镜像集，该集合仅包含与之前执行后的全新和更新镜像。因为它只拉取自以前的镜像集的差异，所以所生成的镜像集通常比初始镜像集更小且更快。

重要

生成的镜像集是有序的，且必须推送到目标镜像 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。

添加新和更新的镜像

根据镜像设置配置中的设置，oc-mirror 的将来执行可能会镜像新的和更新镜像。查看镜像设置配置中的设置，以确保根据需要检索新版本。例如，如果要限制特定版本，可以将 Operator 的最小和最大版本设置为 mirror。另外，您可以将最小版本设置为镜像的起点，但保持对版本范围保持打开状态，以便在以后的 oc-mirror 上执行新的 Operator 版本。省略任何最小或最大版本会为您提供频道中的 Operator 的完整版本历史记录。省略明确指定的频道会为您提供指定 Operator 的所有频道的所有发行版本。省略任何命名 Operator 都会为您提供所有 Operator 的整个目录及其所有版本。

所有这些约束和条件均针对红帽每次调用 oc-mirror 时根据公开发布的内容进行评估。这样，它会自动获取新版本和全新的 Operator。约束只能通过列出所需的 Operator 集合来指定，它不会自动将其他新发布的 Operator 添加到镜像集中。您还可以指定特定的发行版本频道，将镜像限制为只为此频道，而不是任何已添加的新频道。这对于 Operator 产品（如 Red Hat Quay）来说，在其次发行版本中使用不同的发行频道。最后，您可以指定一个特定 Operator 的最大版本，这会导致工具只镜像指定的版本范围，这样您不会自动获得任何更新版本镜像 (mirror) 版本。在所有用例中，您必须更新镜像设置配置文件以扩大 Operator 镜像范围，以获取其他 Operator、新频道和较新版本的 Operator，以便在目标 registry 中提供。

建议将诸如频道规格或版本范围等约束与所选 Operator 的发行策略保持一致。例如，当 Operator 使用 **stable** 频道时，您应该将镜像限制到该频道，并有可能最小的版本来查找下载卷之间的正确平衡并定期获取稳定更新。如果 Operator 选择了发行版本频道方案，如 **stable-3.7**，您应该镜像该频道中的所有发行

版本。这可让您继续使用 Operator 的补丁版本，如 **3.7.1**。您还可以定期调整镜像设置配置，为新产品版本添加频道，如 **stable-3.8**。

修剪镜像
如果镜像不再包含在生成和镜像的最新镜像集中，则会自动从目标镜像 registry 中修剪镜像。这可让您轻松管理和清理不需要的内容并回收存储资源。

如果不再需要 OpenShift Container Platform 发行版本或 Operator 版本，您可以修改镜像设置配置以排除它们，并在镜像 (mirror) 后从镜像 registry 中修剪它们。这可以通过调整镜像设置配置文件中的每个 Operator 的最小或最大版本范围设置，或者从目录中镜像 (mirror) 的 Operator 中移除。您还可以从配置文件中删除整个 Operator 目录或整个 OpenShift Container Platform 版本。

重要

在以下情况下，镜像不会自动从目标镜像 registry 中修剪：

- 如果没有要进行镜像的新的镜像或已更新的镜像
- 如果您使用技术预览 OCI 功能

另外，如果 Operator publisher 从频道中删除 Operator 版本，则从目标镜像 registry 中删除了删除的版本。

要禁用从目标镜像 registry 中自动修剪镜像，请将 **--skip-pruning** 标志传递给 **oc mirror** 命令。

13.2.3.8.2. 更新您的镜像 registry 内容

将初始镜像设置为镜像 registry 后，您可以使用 oc-mirror 插件来保持断开连接的集群更新。

根据您的镜像设置配置，oc-mirror 会自动检测 OpenShift Container Platform 的较新版本，以及在完成 inital mirror 后发布的所选 Operator。建议您定期运行 oc-mirror，例如在每日 cron 作业中运行 oc-mirror，以及时接收产品和安全更新。

先决条件

- 您已使用 oc-mirror 插件将初始镜像设置为您的镜像 registry。
- 您可以访问用于进行 oc-mirror 插件的初始执行的存储后端。

注意

您必须使用与 oc-mirror 的初始执行相同镜像 registry 的存储后端。不要删除或修改 oc-mirror 插件生成的元数据镜像。

流程

1. 如有必要，更新您的镜像设置配置文件，以获取新的 OpenShift Container Platform 和 Operator 版本。对于示例镜像使用情况，请参阅 *Image set configuration examples*。
2. 按照您用来将初始镜像设置为镜像 registry 的步骤操作。具体步骤，请参阅*在部分断开连接的环境中镜像镜像集*，或*在完全断开连接的环境中镜像镜像集*。

重要

- 您必须提供相同的存储后端，以便只创建并镜像不同的镜像集。
- 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。

3. 配置集群以使用 oc-mirror 生成的资源。

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

13.2.3.9. 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。它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。

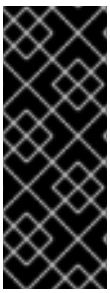
4. 查看生成的 **prune-plan.json** 文件。
此文件包含在发布镜像集时从镜像 registry 中修剪的所有镜像的列表。

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。

13.2.3.10. 以 OCI 格式镜像基于文件的目录 Operator 镜像

您可以使用 oc-mirror 插件以 Open Containerprojects (OCI)镜像格式而不是 Docker v2 格式镜像 Operator。您可以使用 OCI 格式将 Operator 镜像复制到磁盘上基于文件的目录。然后，您可以将本地 OCI 镜像复制到目标镜像 registry。



重要

使用 oc-mirror 插件以 OCI 格式镜像 Operator 镜像只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

使用 OCI 功能时，镜像不会自动从目标镜像 registry 中修剪。

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. 可选：检索您需要的目录和镜像，并将其保存到磁盘上。如果您已在磁盘上有 OCI 格式的目录镜像，您可以跳过这一步。

   a. 创建镜像设置配置文件：

   复制到磁盘的镜像设置配置文件示例

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
 operators:
 - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
   packages:
   - name: aws-load-balancer-operator
```

注意

使用 OCI 功能时，只有 **mirror.operators.catalog** 设置可供使用。

storageConfig 设置将被忽略，而是传递给 **oc mirror** 命令的位置。

   b. 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
  --use-oci-feature \ 2
  --oci-feature-action=copy \ 3
  oci://my-oci-catalog 4
```

1. 传递镜像设置配置文件。此流程假设它名为 **imageset-config.yaml**。
2. 使用 **--use-oci-feature** 标志来启用 OCI 功能。
3. 要将目录复制到磁盘，请将 **--oci-feature-action** 标志设置为 **copy**。
4. 在要输出目录的磁盘上指定一个目录。此流程假设它名为 **my-oci-catalog**。该路径必须以 **oci:**// 开头。如果指定的目录不是完整路径，则会在运行 **oc mirror** 命令的当前工作目录中创建该目录。

注意

您可以选择使用 **--oci-registries-config** 标志来指定到 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。

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
```

将 **registry.mirror** 部分中的 **location** 字段设置为您要从中拉取镜像的替代 registry 位置。**registry** 部分中的 **location** 字段必须与您在镜像设置配置文件中指定的 registry 位置相同。

c. 列出目录内容并验证是否已创建以下目录：

```
$ ls -l
```

输出示例

```
my-oci-catalog      1
oc-mirror-workspace 2
olm_artifacts       3
```

1. 包含 OCI 目录的目录。此流程假设它名为 **my-oci-catalog**。
2. 以原始格式在目录中包含每个镜像的目录。
3. 包含描述此目录引用的 Operator 捆绑包的文件的目录。

2. 更新镜像设置配置文件，以指定磁盘上要镜像到目标镜像 registry 的目录位置：

镜像设置配置文件示例，用于镜像 registry

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
mirror:
 operators:
 - catalog: oci:///home/user/oc-mirror/my-oci-catalog/redhat-operator-index 1
   packages:
   - name: aws-load-balancer-operator
```

1. 指定磁盘上 OCI 目录位置的绝对路径。此流程假设您使用 **my-oci-catalog** 作为目录并镜像 **redhat-operator-index** 目录。该路径必须以 **oci://** 开头。

3. 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 --config=./imageset-config.yaml \ 1
  --use-oci-feature \ 2
  --oci-feature-action=mirror \ 3
  docker://registry.example:5000 4
```

1. 传递更新的镜像设置配置文件。此流程假设它名为 **imageset-config.yaml**。
2. 使用 **--use-oci-feature** 标志来启用 OCI 功能。
3. 要将目录镜像到目标镜像 registry，请将 **--oci-feature-action** 标志设置为 **mirror**。
4. 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。

注意

您可以选择使用 **--oci-insecure-signature-policy** 标志将签名推送到目标镜像 registry。

后续步骤

- 配置集群以使用 oc-mirror 生成的资源。

其他资源

- 基于文件的目录

13.2.3.11. 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 13.1. 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:v4.12 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:v4.12**。
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:v4.12 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-v4.12**。
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	要包括的最小捆绑包的名称，以及将图中到频道头的所有捆绑包。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**。

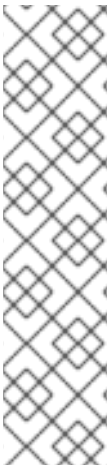
参数	描述	值
mirror.operators.packages.minVersion	Operator 的最低版本，用于镜像存在的所有频道。详情请查看以下备注。	字符串.例如：**5.2.3-31**。
mirror.operators.skipDependencies	如果为 **true**，则不会包含捆绑包的依赖项。	布尔值.默认值为 **false**。
mirror.operators.targetName	将引用的目录镜像为可选的替代名称。	字符串.例如： **my-operator-catalog**
mirror.operators.targetTag	附加到 **targetName** 的可选替代标签。	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`

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-4.12`

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**。
mirror.platform.channels.name	发行频道的名称。	字符串.例如：**stable-4.12**
mirror.platform.channels.minVersion	要镜像引用的平台的最低版本。	字符串.例如：**4.9.6**
mirror.platform.channels.maxVersion	要镜像引用的平台的最高版本。	字符串.例如：**4.12.1**
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**。

参数	描述	值
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，根据具体的 Operator，增加 **maxVersion** 属性，或者减少 **minVersion** 属性。因为每个 Operator 图都可以不同，所以您可能需要根据流程调整这些值，直到错误丢失为止。

13.2.3.12. 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。

使用案例：包含最短的 OpenShift Container Platform 升级路径
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** 版本的升级路径。

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
```

使用案例：包含从最小到最新的 OpenShift Container Platform 的所有版本
以下 **ImageSetConfiguration** 文件使用一个 registry 存储后端，并包括从最小 **4.10.10** 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-4.10** 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。

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:
    channels:
      - name: stable-4.10
        minVersion: 4.10.10
```

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。

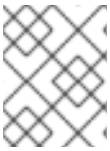
默认情况下，oc-mirror 排除了 Operator Lifecycle Manager (OLM)规格中跳过或被较新的版本替换的任何版本。跳过的 Operator 版本可能会受到 CVE 或包含错误的影响。改为使用较新版本。有关跳过和替换版本的更多信息，请参阅使用 OLM 创建更新图表。

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:v4.12
      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。

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-4.12
      type: ocp
    graph: true
  operators:
  - catalog: registry.redhat.io/redhat/certified-operator-index:v4.12
    packages:
    - name: nutanixcsioperator
      channels:
      - name: stable
  additionalImages:
  - name: registry.redhat.io/ubi9/ubi:latest
```

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:v4.12
    packages:
    - name: elasticsearch-operator
      channels:
      - name: stable-5.7
      - name: stable
```

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:v4.12
      full: true
```

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**。

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:v4.12
```

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-4.12
 operators:
   - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
 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
```

13.2.3.13. oc-mirror 的命令参考

下表描述了 **oc mirror** 子命令和标志：

表 13.2. oc mirror 子命令

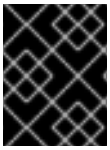
子命令	描述
completion	为指定的 shell 生成自动完成脚本。
describe	输出镜像集合的内容。
帮助	显示有关任何子命令的帮助。
init	输出初始镜像设置配置模板。
list	列出可用平台和 Operator 内容及其版本。
version	输出 oc-mirror 版本。

表 13.3. 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)。
--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 **2**。
--max-per-registry <int>	指定每个 registry 允许的并发请求数。默认值为 **6**。
--oci-feature-action	使用技术预览 OCI 功能时要执行的操作。选项为 **copy** 或 **mirror**。
--oci-insecure-signature-policy	在使用技术预览 OCI 功能时不要推送签名。
--oci-registries-config	提供 registry 配置文件，以指定在使用技术预览 OCI 功能时要复制的替代 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。只有在使用 **--ignore-history** 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。
--use-oci-feature	使用技术预览 OCI 功能复制 OCI 格式的镜像。
-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**。

13.2.4. 使用 oc adm release mirror 命令 mirror 镜像



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身份验证信息。
- 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. 为您的服务器导出构架类型，如 **x86_64**。

```
$ ARCHITECTURE=<server_architecture>
```

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
```

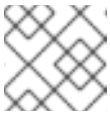
4. 将版本镜像镜像(mirror)到镜像 registry。

- 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}
```

13.3. 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本地 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。

13.3.1. 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，您可以在本地安装和配置 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本地 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

13.3.2. 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 镜像存储库 中所述。

13.3.3. 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，以及更新服务的以下更改。

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。

13.3.4. 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。

当用户使用单独的 registry 存储镜像而不使用安装过程中的 registry时，需要这个过程。

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 提供 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 提供新的 registry。您可以在同一个 registry 中包含多个软件仓库，例如：**--registry="<registry/my-namespace/my-repository>"**。

   2 提供新 registry 的凭据。

   3 提供 pull secret 文件的路径。

   另外，您可以对 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 提供新 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。

13.3.5. 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

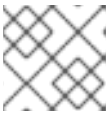
对于在断开连接的环境中安装的集群（也称为断开连接的集群），Operator Lifecycle Manager 默认无法访问托管在远程 registry 上的红帽提供的 OperatorHub 源，因为这些远程源需要有互联网连接。如需更多信息，请参阅在受限网络中使用 Operator Lifecycle Manager。

13.3.5.1. 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. 频道 **v1** 被选为 **Update Channel**，因为它是这个版本中唯一可用的频道。
   b. 在 **Installation Mode** 下选择 **A specific namespace on the cluster**。
   c. 为 **Installed Namespace** 选择一个命名空间，或接受推荐的命名空间 **openshift-update-service**。
   d. 选择一个 **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。
   e. 点 **Install**。
3. 通过切换到 **Operators → Installed Operators** 页来验证 OpenShift Update Service Operator 是否已安装。
4. 确保 **OpenShift Update Service** 列在所选命名空间中，**Status** 为 **Succeeded**。

13.3.5.2. 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
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
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
...
```

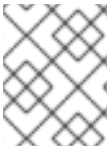
如果列出了 OpenShift Update Service Operator，则会成功安装。版本号可能与所示不同。

其他资源

- 在命名空间中安装 Operator。

13.3.6. 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：

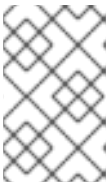
```
FROM registry.access.redhat.com/ubi8/ubi:8.1

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本地 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。

13.3.7. 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。

13.3.7.1. 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。
- 当前发行版本和更新目标版本已被 mirror 到本地可访问的 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，输入创建的本地 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**。

13.3.7.2. 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。
- 当前发行版本和更新目标版本已被 mirror 到本地可访问的 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，为发行镜像配置本地 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：

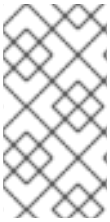
   ```
   $ while sleep 1; do POLICY_ENGINE_GRAPH_URI="$(oc -n "${NAMESPACE}" get -o
   jsonpath='{.status.policyEngineURI}/api/upgrades_info/v1/graph{"\n"}' updateservice
   "${NAME}")"; SCHEME="${POLICY_ENGINE_GRAPH_URI%%:*}"; if test "${SCHEME}"
   = http -o "${SCHEME}" = https; then break; fi; done
   ```

   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 4.12 中运行，请使用 **stable-4.12** ：

   ```
   $ 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。

13.3.7.2.1. 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）从本地安装的 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。
- 当前发行版本和更新目标版本已被 mirror 到本地可访问的 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 进行补丁以使用本地 OpenShift 更新服务：

```
$ oc patch clusterversion version -p $PATCH --type merge
```

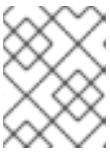
注意

请参阅 启用集群范围代理以将 CA 配置为信任更新服务器。

13.3.8. 后续步骤

在更新集群前，请确认满足以下条件：

- Cluster Version Operator (CVO) 被配置为使用本地安装的 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

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。

- 当前发行版本和更新目标发行镜像被镜像到本地可访问的 registry 中。
- 最近图数据容器镜像已镜像到本地 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，请参阅在受限网络中使用 Operator Lifecycle Manager。

将集群配置为使用本地安装的 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
- 准备执行 EUS 更新
- 执行 canary rollout 更新
- 更新包含使用 RHEL 的计算（compute）系统的集群

13.4. 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。

13.4.1. 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 镜像存储库 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。
- 如果更新失败，您必须有一个最新的 etcd 备份，您必须将 集群恢复到以前的状态。
- 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。

13.4.2. 暂停 MachineHealthCheck 资源

在升级过程中，集群中的节点可能会临时不可用。对于 worker 节点，机器健康检查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。

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-
```

13.4.3. 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。

13.4.4. 更新断开连接的集群

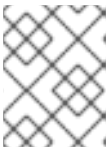
将受限网络集群更新至您下载的发行镜像的 OpenShift Container Platform 版本。

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** 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

13.4.5. 配置镜像 registry 存储库镜像

通过设置容器 registry 存储库镜像，您可以进行以下操作：

- 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 中存储库镜像的属性包括：

- 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。

可通过以下方式设置存储库镜像：

- 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：
  即使没有在 OpenShift Container Platform 安装期间配置镜像 (mirror)，之后您仍可使用 **ImageContentSourcePolicy** 对象进行配置。

以下流程提供安装后镜像配置，您可在此处创建 **ImageContentSourcePolicy** 对象来识别：

- 您希望镜像 (mirror) 的容器镜像存储库的源。
- 您希望为其提供从源存储库请求的内容的每个镜像存储库的单独条目。

注意

您只能为具有 **ImageContentSourcePolicy** 对象的集群配置全局 pull secret。您不能在项目中添加 pull secret。

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：
   - 按照 Red Hat Quay 存储库镜像 中所述，使用 Red Hat Quay 来设置已镜像的存储库。使用 Red Hat Quay 有助于您将镜像从一个存储库复制到另一存储库，并可随着时间的推移重复自动同步这些存储库。
   - 使用 **skopeo** 等工具手动将镜像从源目录复制到已镜像的存储库。
     例如：在 Red Hat Enterprise Linux（RHEL 7 或 RHEL 8）系统上安装 skopeo RPM 软件包后，使用 **skopeo** 命令，如下例所示：

     ```
     $ skopeo copy \
     docker://registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6 \
     docker://example.io/example/ubi-minimal
     ```

     在本例中，您有一个名为 **example.io** 的容器镜像 registry，其中包含一个名为 **example** 的镜像存储库，您希望将 **ubi8/ubi-minimal** 镜像从 **registry.access.redhat.com** 复制到此镜像存储库。创建该 registry 后，您可将 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. 登录您的 OpenShift Container Platform 集群。
3. 创建 **ImageContentSourcePolicy** 文件（如：**registryrepomirror.yaml**），将源和镜像 (mirror) 替换为您自己的 registry、存储库对和镜像中的源和镜像：

   ```
   apiVersion: operator.openshift.io/v1alpha1
   kind: ImageContentSourcePolicy
   metadata:
     name: ubi8repo
   spec:
     repositoryDigestMirrors:
     - mirrors:
       - example.io/example/ubi-minimal (1)
       - example.com/example/ubi-minimal (2)
       source: registry.access.redhat.com/ubi8/ubi-minimal (3)
     - mirrors:
       - mirror.example.com/redhat
       source: registry.redhat.io/openshift4 (4)
     - mirrors:
   ```

```
    - mirror.example.com
    source: registry.redhat.io (5)
  - mirrors:
    - mirror.example.net/image
    source: registry.example.com/example/myimage (6)
  - mirrors:
    - mirror.example.net
    source: registry.example.com/example (7)
  - mirrors:
    - mirror.example.net/registry-example-com
    source: registry.example.com (8)
```

1. 指明镜像 registry 和存储库的名称。
2. 表示每个目标仓库的多个镜像仓库。如果一个镜像停机，则目标仓库可以使用另一个镜像。
3. 指明包含所镜像内容的 registry 和存储库。
4. 您可以在 registry 中配置命名空间以使用该命名空间中的任何镜像。如果您使用 registry 域作为源，**ImageContentSourcePolicy** 资源将应用到 registry 中的所有存储库。
5. 如果配置 registry 名称，则 **ImageContentSourcePolicy** 资源将应用到源 registry 中的所有软件仓库。
6. 拉取镜像 **mirror.example.net/image@sha256:...**。
7. 从 mirror **mirror.example.net/myimage@sha256:...** 的源 registry 命名空间中拉取镜像 **myimage**。
8. 从 mirror registry **mirror.example.net/registry-example-com/example/myimage@sha256:...** 拉取镜像 **registry.example.com/example/myimage**。**ImageContentSourcePolicy** 资源会应用到源 registry 中的所有仓库到到 mirror registry **mirror.example.net/registry-example-com**。

4. 创建新的 **ImageContentSourcePolicy** 对象：

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

创建 **ImageContentSourcePolicy** 对象后，新的设置将部署到每个节点，集群开始使用已镜像的存储库来响应源存储库请求。

5. 要检查是否应用了已镜像的配置设置，在其中一个节点上执行以下内容。

   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   v1.25.0
ip-10-0-138-148.ec2.internal  Ready                      master   11m  v1.25.0
ip-10-0-139-122.ec2.internal  Ready                      master   11m  v1.25.0
```

```
ip-10-0-147-35.ec2.internal   Ready                worker   7m   v1.25.0
ip-10-0-153-12.ec2.internal   Ready                worker   7m   v1.25.0
ip-10-0-154-10.ec2.internal   Ready                master   11m  v1.25.0
```

Imagecontentsourcepolicy 资源不会重启节点。

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
```

输出示例

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

[[registry]]
  prefix = ""
  location = "registry.access.redhat.com/ubi8/ubi-minimal"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "example.io/example/ubi-minimal"

  [[registry.mirror]]
    location = "example.com/example/ubi-minimal"

[[registry]]
  prefix = ""
  location = "registry.example.com"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.example.net/registry-example-com"

[[registry]]
  prefix = ""
  location = "registry.example.com/example"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.example.net"
```

```
[[registry]]
  prefix = ""
  location = "registry.example.com/example/myimage"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.example.net/image"

[[registry]]
  prefix = ""
  location = "registry.redhat.io"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.example.com"

[[registry]]
  prefix = ""
  location = "registry.redhat.io/openshift4"
  mirror-by-digest-only = true

  [[registry.mirror]]
    location = "mirror.example.com/redhat"
```

e. 将镜像摘要从源拉取到节点，并检查是否通过镜像解析。**ImageContentSourcePolicy** 对象仅支持镜像摘要，不支持镜像标签。

```
sh-4.2# podman pull --log-level=debug registry.access.redhat.com/ubi8/ubi-minimal@sha256:5cfbaf45ca96806917830c183e9f37df2e913b187adb32e89fd83fa455ebaa6
```

存储库镜像故障排除

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。

13.4.6. 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:v4.12**

<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。

13.4.7. 其他资源

- 在受限网络中使用 Operator Lifecycle Manager
- 机器配置概述

13.5. 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。

13.5.1. 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。

13.5.1.1. 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。

13.5.1.2. 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
```

13.5.2. 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。

13.5.2.1. 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。

13.5.2.2. 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
```

第 14 章 更新在 VSPHERE 上运行的节点上运行的硬件

您必须确保您在 vSphere 中运行的节点在 OpenShift Container Platform 支持的硬件版本上运行。目前，硬件版本 15 或更高版本都支持集群中的 vSphere 虚拟机。

您可以立即更新虚拟硬件，或在 vCenter 中计划更新。

重要

OpenShift Container Platform 版本 4.12 需要 VMware 虚拟硬件版本 15 或更高版本。

14.1. 更新 VSPHERE 上的虚拟硬件

要在 VMware vSphere 上更新虚拟机 (VM) 的硬件，请单独更新您的虚拟机，以减少集群停机风险。

14.1.1. 为 vSphere 上的 control plane 节点更新虚拟硬件

要减少停机的风险，建议按顺序更新 control plane 节点。这样可确保 Kubernetes API 保持可用，etcd 保留仲裁。

先决条件

- 在托管 OpenShift Container Platform 集群的 vCenter 实例中具有执行所需权限的权限。
- 您的 vSphere ESXi 主机是 7.0U2 或更高版本。

流程

1. 列出集群中的 control plane 节点。

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

输出示例

```
NAME                 STATUS   ROLES    AGE   VERSION
control-plane-node-0   Ready    master   75m   v1.25.0
control-plane-node-1   Ready    master   75m   v1.25.0
control-plane-node-2   Ready    master   75m   v1.25.0
```

请注意 control plane 节点的名称。

2. 将 control plane 节点标记为不可调度。

```
$ oc adm cordon <control_plane_node>
```

3. 关闭与 control plane 节点关联的虚拟机 (VM)。在 vSphere 客户端中通过右键单击虚拟机并选择 **Power → Shut Down Guest OS**进行此操作。不要使用 **Power Off** 来关闭虚拟机，因为它可能无法安全地关闭。

4. 更新 vSphere 客户端中的虚拟机。如需更多信息，请参阅 VMware 文档中的手动升级虚拟机兼容性。

5. 打开与 control plane 节点关联的虚拟机。在 vSphere 客户端中通过右键单击虚拟机并选择 **Power On** 来进行此操作。

6. 等待节点报告为 **Ready** ：

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

7. 再次将 control plane 节点标记为可以调度：

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

8. 对集群中的每个 control plane 节点重复此步骤。

14.1.2. 更新 vSphere 上计算节点的虚拟硬件

要降低停机的风险，建议按顺序更新计算节点。

注意

可以在并行给定工作负载中更新多个计算节点，可以接受具有 **NotReady** 状态的多个节点。管理员负责确保所需的计算节点可用。

先决条件

- 在托管 OpenShift Container Platform 集群的 vCenter 实例中具有执行所需权限的权限。
- 您的 vSphere ESXi 主机是 7.0U2 或更高版本。

流程

1. 列出集群中的计算节点。

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

输出示例

```
NAME           STATUS   ROLES    AGE   VERSION
compute-node-0   Ready    worker   30m   v1.25.0
compute-node-1   Ready    worker   30m   v1.25.0
compute-node-2   Ready    worker   30m   v1.25.0
```

注意计算节点的名称。

2. 将计算节点标记为不可调度：

```
$ oc adm cordon <compute_node>
```

3. 从计算节点撤离容器集。执行此操作有多种方法。例如，您可以撤离节点上的所有或选定 pod：

```
$ oc adm drain <compute_node> [--pod-selector=<pod_selector>]
```

如需从节点上撤离 pod 的其他选项，请参阅"如何撤离节点上的 pod"部分。

4. 关闭与计算节点关联的虚拟机 (VM)。在 vSphere 客户端中通过右键单击虚拟机并选择 **Power → Shut Down Guest OS**进行此操作。不要使用 **Power Off** 来关闭虚拟机，因为它可能无法安全地关闭。

5. 更新 vSphere 客户端中的虚拟机。如需更多信息，请参阅 VMware 文档中的手动升级虚拟机兼容性。

6. 打开与计算节点关联的虚拟机。在 vSphere 客户端中通过右键单击虚拟机并选择 **Power On** 来进行此操作。

7. 等待节点报告为 **Ready** ：

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

8. 将计算节点再次标记为可调度：

```
$ oc adm uncordon <compute_node>
```

9. 对集群中的每个计算节点重复此步骤。

14.1.3. 为 vSphere 上的模板更新虚拟硬件

先决条件

- 在托管 OpenShift Container Platform 集群的 vCenter 实例中具有执行所需权限的权限。
- 您的 vSphere ESXi 主机是 7.0U2 或更高版本。

流程

1. 如果 RHCOS 模板被配置为 vSphere 模板，请在进行下一步前参阅将模板转换为一个虚拟机。

注意

从模板转换后，请不要打开虚拟机电源。

2. 更新 vSphere 客户端中的虚拟机。如需更多信息，请参阅 VMware 文档中的手动升级虚拟机兼容性。

3. 将 vSphere 客户端中的虚拟机从虚拟机转换为模板。如需更多信息，请参阅 VMware 文档中的将虚拟机转换为 vSphere 客户端中的模板。

其他资源

- 了解如何撤离节点上的 pod

14.2. 在 VSPHERE 上调度虚拟硬件的更新

虚拟机开启或重新启动时，可以计划进行虚拟硬件更新。您可以按照 VMware 文档中的虚拟机兼容性升级调度专门在 vCenter 中调度虚拟硬件更新。

在 OpenShift Container Platform 升级前调度升级时，在 OpenShift Container Platform 升级过程中重启节点时会进行虚拟硬件更新。

第 15 章 PREFLIGHT 验证内核模块管理 (KMM) 模块

在应用 KMM 模块的集群中执行升级前，管理员必须在集群升级和可能的内核升级后验证使用 KMM 安装的内核模块是否可以在节点上安装。preflight 会尝试并行验证集群中载入的每个**模块**。在启动一个**模块**的验证前，preflight 并不会等待一个**模块**的验证过程完成。

15.1. 启动验证

preflight 验证通过在集群中创建 **PreflightValidationOCP** 资源来触发。此 spec 包含两个字段：

```
type PreflightValidationOCPSpec struct {
 // releaseImage describes the OCP release image that all Modules need to be checked against.
 // +kubebuilder:validation:Required
 ReleaseImage string `json:"releaseImage"` (1)
 // Boolean flag that determines whether images build during preflight must also
 // be pushed to a defined repository
 // +optional
 PushBuiltImage bool `json:"pushBuiltImage"` (2)
}
```

1. **ReleaseImage** - 必需的字段，为集群升级到的 OpenShift Container Platform 版本提供发行镜像名称。
2. **PushBuiltImage** - 如果为 **true**，则构建和签名验证期间创建的镜像被推送到其存储库（默认为**false**）。

15.2. 验证生命周期

preflight 验证会尝试验证集群中载入的每个模块。preflight 会在验证成功后停止在 **模块** 资源上运行验证。如果模块验证失败，您可以更改模块定义，而 Preflight 将尝试在下一个循环中再次验证模块。

如果要为附加内核运行 Preflight 验证，则应该为该内核创建另一个 **PreflightValidationOCP** 资源。验证所有模块后，建议删除 **PreflightValidationOCP** 资源。

15.3. 验证状态

preflight 会报告它试图验证的集群中每个模块的状态和进度。

```
type CRStatus struct {
 // Status of Module CR verification: true (verified), false (verification failed),
 // error (error during verification process), unknown (verification has not started yet)
 // +required
 // +kubebuilder:validation:Required
 // +kubebuilder:validation:Enum=True;False
 VerificationStatus string `json:"verificationStatus"` (1)
 // StatusReason contains a string describing the status source.
 // +optional
 StatusReason string `json:"statusReason,omitempty"` (2)
 // Current stage of the verification process:
 // image (image existence verification), build(build process verification)
 // +required
 // +kubebuilder:validation:Required
```

```
// +kubebuilder:validation:Enum=Image;Build;Sign;Requeued;Done
VerificationStage string `json:"verificationStage"` (3)
// LastTransitionTime is the last time the CR status transitioned from one status to another.
// This should be when the underlying status changed.  If that is not known, then using the time when
the API field changed is acceptable.
// +required
// +kubebuilder:validation:Required
// +kubebuilder:validation:Type=string
// +kubebuilder:validation:Format=date-time
LastTransitionTime metav1.Time `json:"lastTransitionTime"
protobuf:"bytes,4,opt,name=lastTransitionTime"` (4)
}
```

以下字段适用于每个模块：

1. **VerificationStatus** - **true** 或 **false**，验证或未验证。
2. **StatusReason** - 有关状态的详细说明。
3. **VerificationStage** - 描述正在执行的验证阶段(Image、Build、Sign)。
4. **LastTransitionTime** - 最后一次更新到状态的时间。

15.4. 每个模块的 PREFLIGHT 验证阶段

preflight 在集群中的每个 KMM 模块上运行以下验证：

1. 镜像验证阶段
2. 构建验证阶段
3. 签名验证阶段

15.4.1. 镜像验证阶段

镜像验证始终是要执行的 preflight 验证的第一个阶段。如果镜像验证成功，则不会在该特定模块上运行其他验证。

镜像验证由两个阶段组成：

1. 镜像存在和可访问性。代码会尝试访问为模块中升级的内核定义的镜像，并获取其清单。
2. 验证在正确的路径中存在模块中定义的内核模块，以备将来 **modprobe** 执行。正确的路径为 **<dirname>/lib/modules/<upgraded_kernel>/**。

如果这个验证成功，这可能意味着内核模块是使用正确的 Linux 标头编译的。

15.4.2. 构建验证阶段

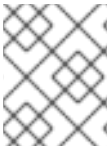
只有在镜像验证失败，且在与升级的内核相关的模块 中有一个 **build** 部分时，才会执行构建验证。构建验证尝试运行构建作业，并验证它是否已成功完成。

注意

在运行 **depmod** 时必须指定内核版本，如下所示：

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

如果在 **PreflightValidationOCP** 自定义资源(CR) 中定义 **PushBuiltImage** 标志，它将尝试将生成的镜像推送到其存储库中。生成的镜像名称取自 **Module** CR 的 **containerImage** 字段的定义。



注意

如果为升级的内核定义了 **sign** 部分，则生成的镜像不是 **Module** CR 的 **containerImage** 字段，而是临时镜像名称，因为生成的镜像应该是 Sign 流的产品。

15.4.3. 签名验证阶段

只有在镜像验证失败时，才会执行签名验证，**模块** 中有一个与升级内核相关的 **sign** 部分，并在与升级的内核相关的**模块**中存在 **build** 部分时成功完成构建部分。签名验证将尝试运行签名作业，并验证它是否已成功完成。

如果在 **PreflightValidationOCP** CR 中定义 **PushBuiltImage** 标志，则签名验证也将尝试将生成的镜像推送到其 registry。

生成的镜像始终是**模块**的 **containerImage** 字段中定义的镜像。输入镜像是 Build 阶段的输出，也可以是 **UnsignedImage** 字段中定义的镜像。

注意

如果存在 **build** 部分，则 **sign** 部分输入镜像是 **build** 部分的输出镜像。因此，为了使输入镜像可用于 **sign** 部分，必须在 **PreflightValidationOCP** CR 中定义 **PushBuiltImage** 标志。

15.5. PREFLIGHTVALIDATIONOCP 资源示例

本节演示了 YAML 格式的 **PreflightValidationOCP** 资源示例。

这个示例根据 OpenShift Container Platform 版本 4.11.18 中包含的即将推出的内核版本验证当前存在的模块，以下发行镜像指向：

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

由于 **.spec.pushBuiltImage** 设置为 **true**，KMM 会将生成的 Build/Sign 镜像推送到定义的存储库中。

```
apiVersion: kmm.sigs.x-k8s.io/v1beta1
kind: PreflightValidationOCP
metadata:
 name: preflight
spec:
 releaseImage: quay.io/openshift-release-dev/ocp-
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
 pushBuiltImage: true
```