Assisted Installer for OpenShift Container Platform 2024

使用辅助安装程序安装 OpenShift Container Platform

用户指南

Last Updated: 2024-07-26

Assisted Installer for OpenShift Container Platform 2024 使用辅助安装程序安装 OpenShift Container Platform

用户指南

法律通告

摘要

有关辅助安装程序的信息及其用法

目录

前言

使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。由于这项工作的艰巨性，这些变化正在尽可能地逐步更新。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

您可以通过在 Jira 中提交 **Create Issue** 表单来提供反馈或报告错误。Jira 问题将在 Red Hat Hybrid Cloud Infrastructure Jira 项目中创建，您可以在其中跟踪您反馈的进度。

1. 确保您已登录到 Jira。如果您没有 Jira 帐户，请创建一个帐户来提交反馈。
2. 点 **Create Issue**
   a. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节号以及问题的详细描述。不要修改表单中的任何其他字段。
3. 点 **Create**。

我们感谢您对我们文档的反馈。

第 1 章 关于辅助安装程序

Red Hat OpenShift Container Platform 的 Assisted Installer 是 Red Hat Hybrid Cloud Console 上提供的用户友好的安装解决方案。辅助安装程序支持各种专注于裸机、Nanix、vSphere 和 Oracle Cloud Infrastructure 的部署平台。

您可以在具有可选 HTTP/S 代理的连接的环境中为以下平台安装 OpenShift Container Platform：

- 高可用性 OpenShift Container Platform 或单节点 OpenShift 集群
- 在带有完整平台基础的裸机或 vSphere 上，或没有集成的其他虚拟平台上的 OpenShift Container Platform
- 可选，OpenShift Virtualization 和 Red Hat OpenShift Data Foundation

1.1. 功能

辅助安装程序提供安装功能作为服务。此软件即服务(SaaS)方法有以下特性：

Web 界面

- 您可以使用 Hybrid Cloud Console 安装集群，而不是手动创建安装配置文件。

没有 bootstrap 节点

- 您不需要 bootstrap 节点，因为 bootstrap 进程运行在集群的一个节点上。

简化的安装工作流

- 您不需要深入了解 OpenShift Container Platform 来部署集群。辅助安装程序提供合理的默认配置。
- 您不需要在本地运行 OpenShift Container Platform 安装程序。
- 您可以访问最新测试的 z-stream 版本的最新辅助安装程序。

高级网络选项

- 辅助安装程序支持具有 SDN 和 OVN 的 IPv4 网络，只具有 OVN 的 IPv6 和双栈网络、基于 NMState 的静态 IP 寻址和 HTTP/S 代理。
- OVN 是 OpenShift Container Platform 4.12 及更新版本的默认 Container Network Interface (CNI)。
- SDN 最高支持 OpenShift Container Platform 4.14，并在 OpenShift Container Platform 4.15 中已弃用。

预安装验证

- 在安装前，辅助安装程序检查以下配置：
  - 网络连接
  - 网络带宽
  - 连接到注册中心

- 域名的上游 DNS 解析
- 集群节点之间的时间同步
- 集群节点硬件
- 安装配置参数

REST API

- 您可以使用辅助安装程序 REST API 自动化安装过程。

1.2. 自定义安装

您可以通过选择一个或多个选项来自定义安装。

这些选项作为 Operator 安装，用于打包、部署和管理 control plane 上的服务和应用程序。

如果需要一些高级配置选项，您可以在安装后部署这些 Operator。

OpenShift Virtualization

您可以部署 OpenShift Virtualization 以执行以下任务：

- 创建和管理 Linux 和 Windows 虚拟机 (VM)。
- 在集群中运行 pod 和虚拟机工作负载。
- 通过各种控制台和 CLI 工具连接到虚拟机。
- 导入和克隆现有虚拟机。
- 管理附加到虚拟机中的网络接口控制器和存储磁盘。
- 在节点间实时迁移虚拟机。

Kubernetes 的多集群引擎

您可以部署 Kubernetes 的多集群引擎，以便在大型的多集群环境中执行以下任务：

- 从您的初始集群中置备和管理额外的 Kubernetes 集群。
- 通过分离 control plane 和 data plane，可以使用托管的 control plane 来降低管理成本并优化集群部署。
- 使用 GitOps Zero Touch Provisioning 大规模管理远程边缘站点。
  您可以在所有 OpenShift Container Platform 集群中使用 Red Hat OpenShift Data Foundation 部署多集群引擎。

重要

在*没有* OpenShift Data Foundation 的情况下部署多集群引擎会导致以下情况：

- 多节点集群 ：没有配置存储。您需要在安装后配置存储。
- 单节点 OpenShift：安装了 LVM 存储。

您必须查看先决条件，以确保您的环境有足够的额外资源用于多集群引擎。

逻辑卷管理器存储

您可以使用逻辑卷管理器存储(LVM Storage)在有限的资源集群中动态置备块存储。

Red Hat OpenShift Data Foundation

您可以使用 Red Hat OpenShift Data Foundation 文件、块和对象存储。所有 OpenShift Container Platform 集群都建议使用此存储选项。OpenShift Data Foundation 需要单独的订阅。

其他资源

- *Operators*。
- OpenShift Virtualization 产品概述。
- OpenShift Virtualization 文档.
- "在 *架构*"中关于 Kubernetes Operator 的多集群引擎 "。
- 架构中的"介绍托管 control *plane* "。
- *边缘计算*.
- *在存储* 中使用逻辑卷管理器" 持久性存储"。
- OpenShift Data Foundation 数据表.
- OpenShift Data Foundation 文档。

1.3. API 支持政策

在宣布弃用后，辅助安装程序 API 最少支持三个月。

第 2 章 先决条件

Assisted Installer 验证以下先决条件以确保成功安装。

如果使用防火墙，需要进行相关的配置，以便 Assisted Installer 可以访问它所需要的资源。

2.1. 支持的 CPU 架构

在以下 CPU 构架上支持辅助安装程序：

- x86_64
- arm64
- ppc64le
- s390x

2.2. 资源要求

本节论述了不同集群和安装选项的资源要求。

Kubernetes 的多集群引擎需要额外的资源。

如果使用存储部署多集群引擎，如 OpenShift Data Foundation 或 LVM Storage，还必须为每个节点分配其他资源。

2.2.1. 多节点集群资源要求

多节点集群的资源要求取决于具体的安装。

多节点集群基本安装

- control plane 节点：
  - 4 个 CPU 内核
  - 16 GB RAM
  - 100 GB 存储

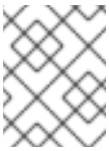
注意

具有快速的磁盘，etcd **wal_fsync_duration_seconds** p99 持续时间小于 10 ms。如需更多信息，请参阅红帽知识库解决方案 如何在 OCP 中使用"fio"检查 Etcd 磁盘性能。

- 计算节点：
  - 2 个 CPU 内核
  - 8 GB RAM
  - 100 GB 存储

多节点集群 + 多集群引擎

- 额外 4 个 CPU 内核
- 额外 16 GB RAM



注意

如果您在没有 OpenShift Data Foundation 的情况下部署多集群引擎，则不会配置存储。您可在安装后配置存储。

多节点集群 + 多集群引擎 + OpenShift Data Foundation 或 LVM Storage

- 额外 75 GB 存储

2.2.2. 单节点 OpenShift 资源要求

单节点 OpenShift 的资源要求取决于具体的安装。

单节点 OpenShift 基本安装

- 8 个 CPU 内核
- 16 GB RAM
- 100 GB 存储

单节点 OpenShift + 多集群引擎

- 额外 8 个 CPU 内核
- 额外 32 GB RAM

注意

如果您在没有 OpenShift Data Foundation 的情况下部署多集群引擎，则会启用 LVM Storage。

单节点 OpenShift + 多集群引擎 + OpenShift Data Foundation 或 LVM Storage

- 额外的 95 GB 存储

2.3. 网络要求

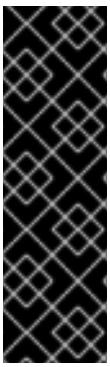
对于 **VMware** 类型的主机，将 **clusterSet disk.enableUUID** 设置为 **true**，即使平台不是 vSphere。

2.4. 网络

网络必须满足以下要求：

- DHCP 服务器，除非使用静态 IP 地址。

- 基本域名。您必须确保满足以下要求：
  - 没有通配符，如 ***.<cluster_name>.<base_domain>**，否则安装将无法进行。
  - **api.<cluster_name>.<base_domain>** 的 DNS A/AAAA 记录。
  - ***.apps.<cluster_name>.<base_domain>** 的带有 DNS A/AAAA 记录。
- 为 API URL 打开端口 **6443**，以允许防火墙外的用户使用 **oc** CLI 工具访问集群。
- 控制台打开端口 443，以允许防火墙外的用户访问控制台。
- **在使用 User Managed Networking 时，集群中每个节点都需要一个 DNS A/AAAA 记录，否则安装将无法进行。安装完成后，集群中的每个节点都需要 DNS A/AAAA 记录来连接到集群，但在使用 Cluster Managed Networking 时，安装可以在没有 A/AAAA 记录的情况下继续。**
- **在使用静态 IP 地址时，如需要使用预设置的主机名进行引导，则集群中每个节点都需要一个 DNS PTR 记录。否则，当使用静态 IP 地址时，辅助安装程序提供的自动重命名功能会将把节点重命名为其网络接口 MAC 地址。**



重要

- 顶级域注册器中的 **DNS A/AAAA** 记录设置可能需要大量时间更新。在安装前确保 **A/AAAA** 记录 **DNS** 设置可以正常工作，以防止安装延迟。
- **有关 DNS 记录示例，请参阅 *DNS 配置示例*。**

OpenShift Container Platform 集群的网络还必须满足以下要求：

- **所有集群节点之间的连接**
- **每个节点到互联网的连接**
- 访问 **NTP 服务器，以便在集群节点之间进行时间同步**

2.5. DNS 配置示例

本节提供了 A 和 PTR 记录配置示例，它们满足使用辅助安装程序部署 OpenShift Container Platform 的 DNS 要求。示例不是为选择一个 DNS 解决方案而提供建议。

在这个示例中，集群名称为 ocp4，基域是 example.com。

2.5.1. DNS A 记录配置示例

以下示例是一个 BIND 区域文件，其显示了使用辅助安装程序安装的集群中名称解析的 A 记录示例。

DNS 区域数据库示例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1.example.com.  IN A 192.168.1.1
smtp.example.com.  IN A 192.168.1.5
;
helper.example.com.  IN A 192.168.1.5
;
api.ocp4.example.com.  IN A 192.168.1.5 1
api-int.ocp4.example.com. IN A 192.168.1.5 2
;
*.apps.ocp4.example.com. IN A 192.168.1.5 3
;
control-plane0.ocp4.example.com. IN A 192.168.1.97 4
control-plane1.ocp4.example.com. IN A 192.168.1.98
control-plane2.ocp4.example.com. IN A 192.168.1.99
;
worker0.ocp4.example.com. IN A 192.168.1.11 5
worker1.ocp4.example.com. IN A 192.168.1.7
;
;EOF
```

1

为 Kubernetes API 提供名称解析。记录引用 API 负载均衡器的 IP 地址。

2

为 Kubernetes API 提供名称解析。记录引用 API 负载均衡器的 IP 地址，用于内部集群通信。

3

为通配符路由提供名称解析。记录引用应用程序入口负载均衡器的 IP 地址。应用程序入口负载均衡器以运行 Ingress Controller Pod 的机器为目标。默认情况下，入口控制器 pod 运行在 worker 机器上。

注意

在这个示例中，将相同的负载均衡器用于 Kubernetes API 和应用入口流量。在生产环境中，您可以单独部署 API 和应用程序入口负载均衡器，以便可以隔离扩展每个负载均衡器基础架构。

4

为控制平面机器提供名称解析。

5

为 worker 机器提供名称解析。

2.5.2. DNS PTR 记录配置示例

以下示例是一个 BIND 区域文件，其显示了使用辅助安装程序安装的集群中反向名称解析的 PTR 记录示例。

反向记录的 DNS 区域数据库示例

```
$$TTL 1W
@ IN SOA ns1.example.com. root (
  2019070700 ; serial
  3H  ; refresh (3 hours)
  30M  ; retry (30 minutes)
  2W  ; expiry (2 weeks)
  1W ) ; minimum (1 week)
 IN NS ns1.example.com.
```

```
;
5.1.168.192.in-addr.arpa. IN PTR api.ocp4.example.com. 1
5.1.168.192.in-addr.arpa. IN PTR api-int.ocp4.example.com. 2
;
97.1.168.192.in-addr.arpa. IN PTR control-plane0.ocp4.example.com. 3
98.1.168.192.in-addr.arpa. IN PTR control-plane1.ocp4.example.com.
99.1.168.192.in-addr.arpa. IN PTR control-plane2.ocp4.example.com.
;
11.1.168.192.in-addr.arpa. IN PTR worker0.ocp4.example.com. 4
7.1.168.192.in-addr.arpa. IN PTR worker1.ocp4.example.com.
;
;EOF
```

1

为 Kubernetes API 提供反向 DNS 解析。PTR 记录引用 API 负载均衡器的记录名称。

2

为 Kubernetes API 提供反向 DNS 解析。PTR 记录引用 API 负载均衡器的记录名称，用于内部集群通信。

3

为 control plane 机器提供反向 DNS 解析。

4

为 worker 机器提供反向 DNS 解析。

注意

OpenShift Container Platform 应用程序通配符不需要 PTR 记录。

2.6. PREFLIGHT 验证

Assisted Installer 确保集群在安装前满足先决条件，因为它消除了复杂的安装后故障排除，从而节省大量时间和工作时间。在节点上安装软件前，辅助安装程序执行以下验证：

- 确保网络连接
- 确保有足够的网络带宽
- 确保连接到 registry
- 确保任何上游 DNS 都可以解析所需的域名
- 确保集群节点之间的时间同步
- 验证集群节点是否满足最低硬件要求
- 验证安装配置参数

如果 Assisted Installer 没有成功验证忘记要求，安装将无法进行。

第 3 章 使用 ASSISTED INSTALLER WEB 控制台安装

确保满足集群节点和网络要求后，您可以开始安装集群。

3.1. 预安装注意事项

在使用 Assisted Installer 安装 OpenShift Container Platform 前，您必须考虑以下配置选择：

- 要使用的基本域
- 要安装的 OpenShift Container Platform 产品版本
- 是否要安装完整集群还是单节点 OpenShift
- 是否使用 DHCP 服务器或静态网络配置
- 是否使用 IPv4 还是双栈网络
- 是否要安装 OpenShift Virtualization
- 是否要安装 Red Hat OpenShift Data Foundation
- 是否为 Kubernetes 安装多集群引擎
- 在 vSphere 或 Nutanix 上安装时是否与平台集成
- 是否要安装混合集群架构

3.2. 设置集群详情

要使用 Assisted Installer Web 用户界面创建集群，请使用以下步骤。

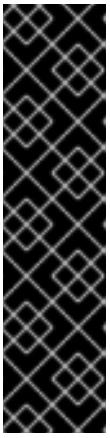
流程

1. 登录到 Red Hat Hybrid Cloud 控制台。
2. 在 Red Hat OpenShift 标题中，点 Scale your application。
3. 在菜单中，单击 Clusters。
4. 点 Create cluster。
5. 点 Datacenter 选项卡。
6. 在 Assisted Installer 下，点 Create cluster。
7. 在 Cluster name 字段中输入集群名称。
8. 在 Base domain 字段中输入集群的基域。集群的所有子域将使用此基域。

注意

基域必须是有效的 DNS 名称。不得为基域设置通配符域。

9. 选择要安装的 OpenShift Container Platform 版本。



重要

- 对于 **IBM Power 和 IBM zSystems 平台，只支持 OpenShift Container Platform 4.13 及更新版本。**
- 对于混合架构集群安装，请选择 **OpenShift Container Platform 4.12 或更高版本，并使用 -multi 选项。有关安装混合架构集群的说明，请参阅 *其它资源*。**

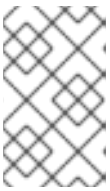
10. 可选： **如果您要在单一节点上安装 OpenShift Container Platform，请选择 Install single node Openshift (SNO)。**

注意

目前，IBM zSystems 和 IBM Power 平台还不支持 SNO。

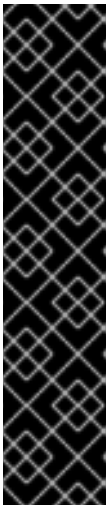
11. 可选： **Assisted Installer 已关联有与您的帐户关联的 pull secret。如果要使用不同的 pull secret，请选择 Edit pull secret。**

12. 可选：**如果您要在第三方平台上安装 OpenShift Container Platform，请从 Integrate with external parter platforms 列表中选择平台。有效值为 Nutanix、vSphere 或 Oracle Cloud Infrastructure。辅助安装程序默认没有平台集成。**



注意

有关每个外部合作伙伴集成的详情，请参考 *其它资源*。

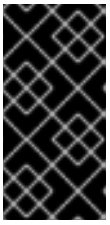


重要

辅助安装程序支持 OpenShift Container Platform 4.14 及之后版本中的 Oracle Cloud Infrastructure (OCI)集成。对于 OpenShift Container Platform 4.14，OCI 集成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。

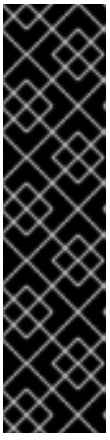
13. 可选：辅助安装程序默认使用 x86_64 CPU 架构。如果要在不同的架构上安装 OpenShift Container Platform，请选择要使用的架构。有效值为 arm64, ppc64le, 和 s390x。请记住，一些功能不适用于 arm64, ppc64le, 和 s390x CPU 架构。



重要

对于混合架构集群安装，请使用默认的 x86_64 架构。有关安装混合架构集群的说明，请参阅 *其它资源*。

14. 可选：如果您至少有一个自定义清单要包括在安装中，请选择 Include custom manifests 。自定义清单包含目前在辅助安装程序种不支持的其它配置。选中复选框来向向导中添加 Custom manifests 页面，您可以在其中上传清单。



重要

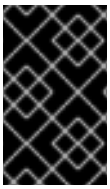
- 如果要在 Oracle Cloud Infrastructure (OCI)第三方平台上安装 OpenShift Container Platform，则需要强制添加 Oracle 提供的自定义清单。
- 如果您已经添加了自定义清单，取消选中 Include custom manifests 框会自动将它们全部删除。将要求您确认删除。

15. 可选：辅助安装程序默认为 DHCP 网络。如果集群节点需要使用静态 IP 配置、网桥或绑定，而不是使用 DHCP，请选择 Static IP、bridge 和 bond。

注意

Oracle Cloud 基础设施上的 OpenShift Container Platform 安装不支持静态 IP 配置。

16. 可选：如果要启用安装磁盘的加密，对单节点 OpenShift，在Enable encryption of installation disks 中选择 Control plane node, worker。对于多节点集群，您可以选择 Control plane 节点 来加密 control plane 节点安装磁盘，选择 Workers 来加密 worker 节点安装磁盘。



重要

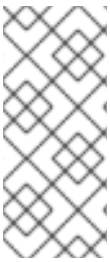
您不能在安装后更改基域、SNO 复选框、CPU 架构、主机的网络配置或磁盘加密。

其他资源

- 可选：在 Nutanix 上安装
- 可选：在 vSphere 上安装
- 可选：在 Oracle Cloud Infrastructure (OCI)上安装

3.3. 可选：配置静态网络

辅助安装程序支持最高到 OpenShift Container Platform 4.14 和 OVN 的带有 SDN 的 IPv4 网络，且只支持具有 OVN 的 IPv6 和双栈网络。Assisted Installer 支持使用 IP 地址/MAC 地址映射使用静态网络接口配置网络。Assisted Installer 还支持使用 NMState 库配置主机网络接口，即主机的声明网络管理器 API。您可以使用 NMState 部署带有静态 IP 寻址、绑定、VLAN 和其他高级网络功能的主机。首先，您必须设置网络范围内的配置。然后，您必须为每个主机创建特定于主机的配置。



注意

对于 IBM Z 上使用 z/VM 的安装，请确保 z/VM 节点和 vSwitch 为静态网络和 NMState 进行了正确配置。另外，z/VM 节点必须分配有固定的 MAC 地址,因为池 MAC 地址可能会导致 NMState 有问题。

流程

1. 选择互联网协议版本。有效选项为 IPv4 和 Dual stack。

2. 如果集群主机位于共享 VLAN 上，请输入 VLAN ID。

3. 输入网络范围内的 IP 地址。如果选择了 Dual stack 网络，则必须同时输入 IPv4 和 IPv6 地址。

   a. 以 CIDR 标记形式输入集群网络的 IP 地址范围。

   b. 输入默认网关 IP 地址。

   c. 输入 DNS 服务器 IP 地址。

4. 输入特定于主机的配置。

   a. 如果您只设置使用单一网络接口的静态 IP 地址，请使用表单视图输入主机的 IP 地址和 MAC 地址。

   b. 如果您使用多个接口、绑定或其他高级网络功能，请使用 YAML 视图，并为使用 NMState 语法的每个主机输入所需的网络状态。然后，为在网络配置中使用的每个主机接口添加 MAC 地址和接口名称。

其他资源

- NMState 版本 2.1.4

3.4. 可选：安装 OPERATOR

此步骤是可选的。

有关先决条件和配置选项，请查看产品文档：

- OpenShift Virtualization
- Kubernetes 的多集群引擎
- Red Hat OpenShift Data Foundation
- 逻辑卷管理器存储

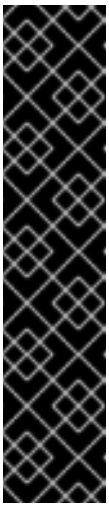
如果需要高级选项，请在安装集群后安装 Operator。

流程

1. **从以下选项中选择一个或多个：**

   - **安装 OpenShift Virtualization**
   - **安装多集群引擎**

     您可以在所有 OpenShift Container Platform 集群中使用 OpenShift Data Foundation 部署多集群引擎。

     重要

     在*没有* OpenShift Data Foundation 的情况下部署多集群引擎会导致以下存储配置：

     - **多节点集群 ：没有配置存储。您必须在安装后配置存储。**
     - **单节点 OpenShift：安装了 LVM 存储。**
   - **安装逻辑卷管理器存储**

- 安装 OpenShift Data Foundation

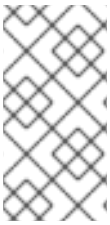
2. 点击 **Next**。

3.5. 在集群中添加主机

您必须将一个或多个主机添加到集群中。将主机添加到集群涉及生成发现 ISO。发现 ISO 使用代理运行 Red Hat Enterprise Linux CoreOS (RHCOS)内存中。

使用下表识别 IBM Z® 架构的镜像文件类型：

架构	引导方法	镜像类型
逻辑分区-Classic	iPXE	完整镜像文件：下载自包含的 ISO 镜像
逻辑分区保护管理器	ISO 或 iPXE	最小镜像文件：下载引导时获取内容的 ISO 镜像



注意

在使用 z/VM 或逻辑分区(LPAR)节点的 IBM Z (s390x)上安装 ISO 镜像不支持；使用带有 iPXE"引导主机"过程。RHEL KVM 上安装支持 ISO 镜像和 iPXE。

对集群中的每个主机执行以下步骤。

流程

1. 单击 **Add hosts** 按钮，并选择调配类型。

   a. 选择 **Minimal image file: Provision with virtual media** 以下载一个较小的镜像，该镜像会获取引导所需的数据。节点必须具有虚拟介质功能。这是 x86_64 和 arm64 架构的推荐方法。

   b. 选择 **Full image file: Provision with physical media** 以下载更大的完整镜像。这是对 ppc64le 架构和使用 RHEL KVM 安装时对 s390x 架构的推荐方法。

2. 选择 **iPXE: Provision from your network server**，以使用 **iPXE** 引导主机。对带有 **z/VM** 或 **LPAR** 节点的 **IBM Z** 使用此方法。

注意

- 如果您在 **RHEL KVM** 上安装，在某些情况下，**KVM** 主机上的虚拟机没有在第一次引导时重启，需要手动重启。
- 如果在 **Oracle Cloud Infrastructure** 上安装 **OpenShift Container Platform**，请只选择 **Minimal image file: Provision with virtual media**。

3. 可选：激活 **Run workloads on control plane nodes** 选项，以便可以在默认的 **worker** 节点上调度工作负载外，还可以在 **control plane** 节点上运行工作负载。

注意

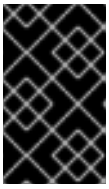
这个选项可用于带有五个或更多节点的集群。对于少于五个节点的集群，系统默认仅在 **control plane** 节点上运行工作负载。如需了解更多详细信息，请参阅*附加资源*中的*配置可调度的 control plane 节点*。

4. 可选： 如果集群主机位于需要使用代理的防火墙后面，请选择 **Configure cluster-wide proxy settings**。输入代理服务器的 **HTTP** 和 **HTTPS URL** 的用户名、密码、**IP** 地址和端口。

注意

代理用户名和密码必须采用 **URL** 编码。

5. 可选：添加一个 **SSH** 公钥，以便可以以 **core** 用户身份连接到集群节点。通过登录到集群节点，您可以在安装过程中为您提供调试信息。



重要

不要在生产环境中跳过这个过程，在生产环境中需要灾难恢复和调试。

a. 如果本地计算机上没有现有 SSH 密钥对，请按照 为集群节点 SSH 访问生成密钥对 的步骤进行操作。

b. 在 SSH public key 字段中，单击 Browse 来上传包含 SSH 公钥的 id_rsa.pub 文件。或者，将文件拖放到文件管理器的字段中。要查看文件管理器中的文件，请在菜单中选择 Show hidden files。

6. 可选： 如果集群主机位于带有重新加密 man-in-the-middle (MITM)代理的网络中，或者集群需要信任证书用于容器镜像 registry，请选择 Configure cluster-wide trusted certificate。以 X.509 格式添加额外的证书。

7. 如果需要，配置发现镜像。

8. 可选：如果您要在一个平台上安装并希望与该平台集成，请选择 Integrate with your virtualization platform。您必须引导所有主机，并确保它们出现在主机清单中。所有主机必须位于同一平台上。

9. 点 Generate Discovery ISO 或 Generate Script File。

10. 下载发现 ISO 或 iPXE 脚本。

11. 使用发现镜像或 iPXE 脚本引导主机。

其他资源

- 配置发现镜像了解更多详细信息。
- 使用发现镜像引导主机获取更多详细信息。
- Red Hat Enterprise Linux 9 - 配置和管理虚拟化了解更多详细信息。
- 如何配置 VIOS Media Repository/Virtual Media Library了解更多详情。

- 使用 Web 控制台在 Nutanix 中添加主机
- 在 vSphere 中添加主机
- 配置可调度的 control plane 节点

3.6. 配置主机

使用发现 ISO 引导主机后，主机将显示在页面底部的表中。您可以选择为每个主机配置主机名和角色。如果需要，您还可以删除主机。

流程

1. 在主机的 Options (⋮) 菜单中，选择 Change hostname。如有必要，为主机输入一个新名称，然后点更改。您必须确保每个主机具有有效且唯一的主机名。

   或者，从 Actions 列表中选择 Change hostname 以重命名多个所选主机。在 Change Hostname 对话框中，输入新名称并包含 {{n}}，使每个主机名都是唯一的。然后点 更改。

   注意

   在键入时，您可以看到在 Preview 窗格中显示的新名称。所有选定的主机的名称都几乎相同，唯一的不同是每个主机都带有一个一位的递增数字。

2. 在 Options (⋮) 菜单中，您可以选择 Delete host 来删除一个主机。点 Delete 以确认删除。

   或者，从 Actions 列表中选择 Delete 同时删除多个所选主机。然后点 Delete hosts。

   注意

   在常规部署中，集群可以有三个或更多主机，其中三个主机必须是 control plane 主机。如果您删除一个也是 control plane 的主机，或者您只保留两个主机，则会出现一条信息表示系统未就绪。要恢复主机，您需要使用发现 ISO 重启它。

3. 在主机的 Options (⋮) 菜单中，可以选择 View host events（可选）。列表中的事件按时间顺序显示。

4. 对于多主机集群，在主机名旁边的 Role 列中，您可以点菜单来更改主机的角色。

   如果您没有选择角色，则 Assisted Installer 会自动分配角色。control plane 节点的最低硬件要求超过 worker 节点。如果为主机分配角色，请确保为满足最低硬件要求的主机分配 control plane 角色。

5. 点 Status 链接，以查看主机的硬件、网络和操作器验证。

6. 点主机名左侧的箭头，以展开主机详细信息。

当所有集群主机都显示为 Ready 状态后，继续下一步。

3.7. 配置存储磁盘

主机发现期间检索到的每个主机可能有多个存储磁盘。在辅助安装程序向导的 Storage 页面中列出了主机的存储磁盘。

您可以选择修改每个磁盘的默认配置。

更改安装磁盘

辅助安装程序默认随机分配一个安装磁盘。如果主机有多个存储磁盘，您可以选择一个不同的磁盘作为安装磁盘。这会自动取消之前磁盘的分配。

流程

1. 导航到向导的 Storage 页面。

2. 扩展主机以显示关联的存储磁盘。

3. 从 Role 列表中选择 Installation disk。

4. 当所有存储磁盘都返回到 Ready 状态时，继续下一步。

禁用磁盘格式化

辅助安装程序在安装过程中默认将所有可引导磁盘标记为进行格式化，无论它们是否已被定义为安装磁盘。格式化会导致数据丢失。

您可以选择禁用特定磁盘的格式化。这应该谨慎执行，因为可引导磁盘可能会干扰安装过程，主要在引导顺序方面。

您无法禁用对安装磁盘的格式化。

流程

1. 导航到向导的 Storage 页面。
2. 扩展主机以显示关联的存储磁盘。
3. 取消磁盘的 Format。
4. 当所有存储磁盘都返回到 Ready 状态时，继续下一步。

其他资源

- 配置主机

3.8. 配置网络

在安装 OpenShift Container Platform 之前，您必须配置集群网络。

流程

1. 在 Networking 页面中，如果还没有为您选择，请选择以下之一：

- **Cluster-Managed Networking：选择集群管理的联网意味着 Assisted Installer 将配置标准网络拓扑，包括 keepalived 和 Virtual Router Redundancy Protocol (VRRP) 用于管理 API 和 Ingress VIP 地址。**

注意

  - **目前，OpenShift Container Platform 版本 4.13 中的 IBM zSystems 和 IBM Power 不支持集群管理的网络。**
  - **Oracle Cloud Infrastructure (OCI)仅适用于带有用户管理的网络配置的 OpenShift Container Platform 4.14。**

- **用户管理的联网 ：选择用户管理的网络允许您使用非标准网络拓扑部署 OpenShift Container Platform。例如，如果要使用外部负载均衡器而不是 keepalived 和 VRRP 部署，或者您想要在多个不同 L2 网络片段中部署集群节点。**

2. **对于集群管理的联网，请配置以下设置：**

   a. **定义 Machine 网络。您可以使用默认网络或选择子网。**

   b. **定义一个 API 虚拟 IP。API 虚拟 IP 为所有用户提供与平台交互的端点。**

   c. **定义一个 Ingress 虚拟 IP。Ingress 虚拟 IP 为来自集群外部的应用程序流量提供端点。**

3. **对于用户管理的网络，请配置以下设置：**

   a. **选择您的网络堆栈类型 ：**

      - **IPv4 ：当您的主机只使用 IPv4 时，请选择此类型。**
      - **Dual-stack：当主机将 IPv4 与 IPv6 一起使用时，您可以选择双栈。**

b. 定义 Machine 网络。您可以使用默认网络或选择子网。

c. 定义一个 API 虚拟 IP。API 虚拟 IP 为所有用户提供与平台交互的端点。

d. 定义一个 Ingress 虚拟 IP。Ingress 虚拟 IP 为来自集群外部的应用程序流量提供端点。

e. 可选：您可以选择 Allocate IPs via DHCP server 来自动分配 API IP 和 Ingress IP 使用 DHCP 服务器。

4. 可选： 选择 Use advanced networking 来配置以下高级网络属性：

- Cluster network CIDR ：定义从中分配 Pod IP 地址的 IP 地址块。
- Cluster network host prefix ：定义分配给每个节点的子网前缀长度。
- 服务网络 CIDR ：定义用于服务 IP 地址的 IP 地址。
- Network type: 选择 Software-Defined Networking (SDN) 用于标准网络，或 Open Virtual Networking (OVN) 用于 IPv6, 双堆栈和电信功能。在 OpenShift Container Platform 4.12 及更新的版本中，OVN 是默认的 Container Network Interface (CNI)。在 OpenShift Container Platform 4.15 及更新版本中，不支持 软件定义的网络(SDN)。

其他资源

- 网络配置

3.9. 添加自定义清单

自定义清单是一个 JSON 或 YAML 文件，其包含辅助安装程序用户界面目前不支持的高级配置。您可以创建一个自定义清单，或使用第三方提供的清单。

您可以将自定义清单从文件系统上传到 openshift 文件夹或 manifests 文件夹。对允许的自定义清单文件的数量没有限制。

一次只能上传一个文件。但是，每个上传的 YAML 文件可以包含多个自定义清单。上传多文档 YAML 清单比单独添加 YAML 文件要快。

对于包含一个自定义清单的文件，可接受的文件扩展名包括 .yaml、.yml 或 .json。

一个自定义清单示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
```

对于包含多个自定义清单的文件，可接受的文件类型包括 .yaml 或 .yml。

多个自定义清单示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
---
apiVersion: machineconfiguration.openshift.io/v2
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-openshift-machineconfig-worker-kargs
```

```
spec:
  kernelArguments:
    - loglevel=5
```

注意

- 当在 Oracle Cloud Infrastructure (OCI)外部平台上安装 OpenShift Container Platform 时，您必须添加 Oracle 提供的自定义清单。对于其它外部合作伙伴集成，如 vSphere 或 Nutanix，此步骤是可选的。
- 有关自定义清单的更多信息，请参阅 *其它资源*。

在辅助安装程序用户界面中上传自定义清单

在上传自定义清单时，输入清单文件名，并选择目标文件夹。

先决条件

- 您至少有一个保存在文件系统中的自定义清单文件。

流程

1. 在向导的 Cluster details 页面中，选择 Include custom manifests 复选框。
2. 在 Custom manifest 页面中，在 folder 字段中选择您要保存自定义清单文件的辅助安装程序文件夹。选项包括 openshift 或 manifest。
3. 在 Filename 字段中，输入清单文件的名称，包括扩展名。例如，manifest1.json 或 multiple1.yaml。
4. 在 Content 下，点 Upload 图标或 Browse 按钮来上传文件。或者，将文件拖到您文件系统的 Content 字段中。
5. 要上传另一个清单，请单击 Add another manifest，并重复此过程。这将保存之前上传的清单。

6. 点 Next 保存所有清单，然后进入到 Review and create 页面。上传的自定义清单列在 Custom manifests 下。

在辅助安装程序用户界面中修改自定义清单

您可以更改上传的自定义清单的文件夹和文件名。您还可以复制现有清单的内容，或将其下载到 Chrome 下载设置中定义的文件夹中。

无法修改上传的清单的内容。但是，您可以覆盖该文件。

先决条件

- 您至少上传了一个自定义清单文件。

流程

1. 要更改文件夹，请从 Folder 列表中为清单选择一个其他的文件夹。
2. 要修改文件名，请在 File name 字段中输入清单的新名称。
3. 要覆盖清单，请在同一文件夹中使用相同的文件名保存新清单。
4. 要在文件系统中将清单保存为一个文件，请点 Download 图标。
5. 要复制清单，请点击 Copy to clipboard 图标。
6. 要应用更改，请单击 Add another manifest 或 Next。

在辅助安装程序用户界面中删除自定义清单

在安装前，您可以通过以下两种方式之一删除上传的自定义清单：

- 单独删除一个或多个清单。
- 一次删除所有清单。

删除清单后，您无法撤销操作。临时解决办法是再次上传清单。

删除单个清单

您可以一次删除一个清单。此选项不允许您删除最后剩余的清单。

先决条件

- 您已至少上传了两个自定义清单文件。

流程

1. 进入到 Custom manifests 页面。
2. 将鼠标悬停在清单名称上，以显示 Delete (-)图标。
3. 单击该图标，然后在对话框中单击 Delete。

删除所有清单

您可以一次删除所有自定义清单。这也会隐藏 Custom manifest 页面。

先决条件

- 您至少上传了一个自定义清单文件。

流程

1. 进入到向导的 Cluster details 页面。

2. 清除 Include custom manifests 复选框。

3. 在 Remove custom manifests 对话框中，点 Remove。

其他资源

- 清单配置文件
- 多文档 YAML 文件

3.10. 预安装验证

Assisted Installer 确保集群在安装前满足先决条件，因为它消除了复杂的安装后故障排除，从而节省大量时间和工作时间。在安装集群前，请确保集群和每个主机都通过了预安装验证。

其他资源

- 预安装验证

3.11. 安装集群

完成配置并且所有节点都为 Ready 后，您可以开始安装。安装过程需要花费相当长的时间，您可以从 Assisted Installer Web 控制台监控安装。节点将在安装过程中重新引导，它们将在安装后进行初始化。

流程

1. 按 Begin Installation。

2. 单击 Host Inventory 列表中 Status 列中的链接，以查看特定主机的安装状态。

3.12. 完成安装

安装和初始化集群后，Assisted Installer 表示安装已完成。Assisted Installer 提供控制台 URL、kubeadmin 用户名和密码以及 kubeconfig 文件。另外，辅助安装程序提供集群详情，包括

OpenShift Container Platform 版本、基础域、CPU 架构、API 和 Ingress IP 地址，以及集群和服务网络 IP 地址。

先决条件

- **已安装 oc CLI 工具。**

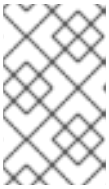
流程

1. **复制 kubeadmin 用户名和密码。**

2. **下载 kubeconfig 文件，并将其复制到工作目录中的 auth 目录中：**

```
$ mkdir -p <working_directory>/auth
```

```
$ cp kubeconfig <working_directory>/auth
```



注意

kubeconfig 文件可在完成安装后的 20 天下载。

3. **在您的环境中添加 kubeconfig 文件：**

```
$ export KUBECONFIG=<your working directory>/auth/kubeconfig
```

4. **使用 oc CLI 工具登录：**

```
$ oc login -u kubeadmin -p <password>
```

将 <password> 替换为 kubeadmin 用户的密码。

5. **点 Web 控制台 URL 或点 Launch OpenShift Console 来打开控制台。**

6. **输入 kubeadmin 用户名和密码。按照 OpenShift Container Platform 控制台中的说明来配置身份提供程序并配置警报接收器。**

7. **添加 OpenShift Container Platform 控制台的书签。**

8. **完成所有安装后平台集成步骤。**

其他资源

- Nutanix 安装后配置
- vSphere 安装后配置

第 4 章 使用 ASSISTED INSTALLER API 安装

确保满足集群节点和网络要求后，您可以使用 Assisted Installer API 开始安装集群。要使用 API，您必须执行以下步骤：

- 设置 API 身份验证。
- 配置 pull secret。
- 注册新的集群定义。
- 为集群创建基础架构环境。

执行这些步骤后，您可以修改集群定义，创建发现 ISO，在集群中添加主机，并安装集群。本文档并没有包括 Assisted Installer API 的每个端点，但您可以查看 API viewer 或 swagger.yaml 文件中的详细内容。

4.1. 生成离线令牌

从 Assisted Installer Web 控制台下载离线令牌。您将使用离线令牌来设置 API 令牌。

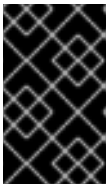
先决条件

- 安装 jq。
- 以具有集群创建权限的用户身份登录 OpenShift Cluster Manager。

流程

1. 在菜单中，单击 Downloads。
2. 在 OpenShift Cluster Manager API Token 下的 Tokens 部分中，点 View API Token。

3. 点 Load Token。



重要

禁用弹出窗口阻塞。

4. 在 API 令牌 部分中，复制离线令牌。

5. 在终端中，将离线令牌设置为 OFFLINE_TOKEN 变量：

```
$ export OFFLINE_TOKEN=<copied_token>
```

提示

要使离线令牌永久生效，请将其添加到您的配置集中。

6. （可选）确认 OFFLINE_TOKEN 变量定义。

```
$ echo ${OFFLINE_TOKEN}
```

4.2. 使用 REST API 进行身份验证

API 调用需要通过 API 令牌进行身份验证。假设您使用 API_TOKEN 作为变量名称，请将 -H "Authorization: Bearer ${API_TOKEN}" 添加到 API 调用中以与 REST API 进行身份验证。

注意

API 令牌在 15 分钟后过期。

先决条件

- 您已生成了 OFFLINE_TOKEN 变量。

流程

1. 在命令行终端上，使用 OFFLINE_TOKEN 设置 API_TOKEN 变量来验证用户。

```
$ export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
)
```

2. 确认 API_TOKEN 变量定义：

```
$ echo ${API_TOKEN}
```

3. 在您的路径中为令牌生成方法创建一个脚本。例如：

```
$ vim ~/.local/bin/refresh-token
```

```
export API_TOKEN=$( \
  curl \
  --silent \
  --header "Accept: application/json" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=refresh_token" \
  --data-urlencode "client_id=cloud-services" \
  --data-urlencode "refresh_token=${OFFLINE_TOKEN}" \
  "https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token" \
  | jq --raw-output ".access_token" \
)
```

然后保存文件。

4. 更改文件模式使其可执行：

```
$ chmod +x ~/.local/bin/refresh-token
```

5. 刷新 API 令牌：

```
$ source refresh-token
```

6. 运行以下命令验证您可以访问 API：

```
$ curl -s https://api.openshift.com/api/assisted-install/v2/component-versions -H
"Authorization: Bearer ${API_TOKEN}" | jq
```

输出示例

```
{
  "release_tag": "v2.11.3",
  "versions": {
    "assisted-installer": "registry.redhat.io/rhai-tech-preview/assisted-installer-
rhel8:v1.0.0-211",
    "assisted-installer-controller": "registry.redhat.io/rhai-tech-preview/assisted-
installer-reporter-rhel8:v1.0.0-266",
    "assisted-installer-service": "quay.io/app-sre/assisted-service:78d113a",
    "discovery-agent": "registry.redhat.io/rhai-tech-preview/assisted-installer-agent-
rhel8:v1.0.0-195"
  }
}
```

4.3. 配置 PULL SECRET

许多 Assisted Installer API 调用都需要 pull secret。将 pull secret 下载到文件中，以便您可以在 API 调用中引用它。pull secret 是一个 JSON 对象，它将作为请求的 JSON 对象中的值包含在其中。必须格式化 pull secret JSON 来转义引号。例如：

之前

```
{"auths":{"cloud.openshift.com": ...
```

After

```
{\"auths\":{\"cloud.openshift.com\": ...
```

流程

1. 在菜单中，点 OpenShift。
2. 在子菜单中，点 Downloads。
3. 在 Pull secret 下的 Tokens 部分中，点 Download。
4. 要使用 shell 变量中的 pull secret，请执行以下命令：

```
$ export PULL_SECRET=$(cat ~/Downloads/pull-secret.txt | jq -R .)
```

5. 要使用 jq 分片 pull secret 文件，请在 pull_secret 变量中引用它，将值传送到 tojson，以确保它被正确格式化为转义的 JSON。例如：

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
      --slurpfile pull_secret ~/Downloads/pull-secret.txt ' (1)
  {
    "name": "testcluster",
    "high_availability_mode": "None",
    "openshift_version": "4.11",
    "pull_secret": $pull_secret[0] | tojson, (2)
    "base_dns_domain": "example.com"
  }
')"
```

(1) 对 pull secret 文件进行 Slurp。

(2) 格式化 pull secret 以转义 JSON 格式。

6. 确认 PULL_SECRET 变量定义：

```
$ echo ${PULL_SECRET}
```

4.4. 可选：生成 SSH 公钥

在 OpenShift Container Platform 安装过程中，您可以选择向安装程序提供 SSH 公钥。当排除安装错误时，这对于启动到远程节点的 SSH 连接很有用。

如果您在本地机器上没有一个用于身份验证的现有 SSH 密钥，请现在创建一个。

先决条件

- 生成 OFFLINE_TOKEN 和 API_TOKEN 变量。

流程

1. 从终端中的 root 用户获取 SSH 公钥：

```
$ cat /root/.ssh/id_rsa.pub
```

2. 将 SSH 公钥设置为 CLUSTER_SSHKEY 变量：

```
$ CLUSTER_SSHKEY=<downloaded_ssh_key>
```

3. 确认 CLUSTER_SSHKEY 变量定义：

```
$ echo ${CLUSTER_SSHKEY}
```

4.5. 注册新集群

要使用 API 注册新集群定义，请使用 /v2/clusters 端点。

以下参数是必需的：

- name
- openshift-version
- pull_secret
- cpu_architecture:x86_64,arm64,ppc64le,s390x, 或 multi，用于多架构集群。

有关注册新集群时可设置的字段的详情，请参阅 API viewer 中的 cluster-create-params 模型。设置 olm_operators 字段时，请参阅 *其它资源*，以了解安装 Operator 的详情。

先决条件

- 您已生成了一个有效的 API_TOKEN。令牌每 15 分钟过期一次。
- 您已下载了 pull secret。
- 可选： 您已将 pull secret 分配给 $PULL_SECRET 变量。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 使用以下方法之一注册新集群：

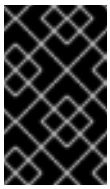
   - 通过在请求中引用 pull secret 文件来注册集群：

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
  -H "Authorization: Bearer ${API_TOKEN}" \
  -H "Content-Type: application/json" \
  -d "$(jq --null-input \
      --slurpfile pull_secret ~/Downloads/pull-secret.txt ' \
```

```
{ \
  "name": "testcluster", \
  "openshift_version": "4.16", \
  "cpu_architecture" : "<architecture_name>", \ 1
  "high_availability_mode": "<mode>", \ 2
  "base_dns_domain": "example.com", \
  "pull_secret": $pull_secret[0] | tojson \
} \
')" | jq '.id'
```

1

有效值为 x86_64,arm64,ppc64le,s390x,multi。



重要

为混合架构集群指定 multi。

2

对于一个高可用性的多节点集群，将值设为 Full，对于单节点 OpenShift 集群，将值设为 None。

- 通过将配置写入 JSON 文件来注册集群：

```
$ cat << EOF > cluster.json
{
 "name": "testcluster",
 "openshift_version": "4.16",
 "high_availability_mode": "<mode>", 1
 "base_dns_domain": "example.com",
 "network_type": "examplenetwork",
 "cluster_network_cidr":"11.111.1.0/14"
 "cluster_network_host_prefix": 11,
 "service_network_cidr": "111.11.1.0/16",
 "api_vips":[{"ip": ""}],
 "ingress_vips": [{"ip": ""}],
 "vip_dhcp_allocation": false,
 "additional_ntp_source": "clock.redhat.com,clock2.redhat.com",
 "ssh_public_key": "$CLUSTER_SSHKEY",
 "pull_secret": $PULL_SECRET
}
EOF
```

1

对于一个高可用性的多节点集群，将值设为 Full，对于单节点 OpenShift 集群，将值设为 None。

- 应用配置：

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/clusters" \
  -d @./cluster.json \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq '.id'
```

3. 将返回的 cluster_id 分配给 CLUSTER_ID 变量，并导出它：

```
$ export CLUSTER_ID=<cluster_id>
```

注意

如果关闭终端会话，则需要在新的终端会话中再次导出 CLUSTER_ID 变量。

4. 检查新集群的状态：

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
  | jq
```

注册新的集群定义后，为集群创建基础架构环境。

注意

在创建基础架构环境前，您无法在 Assisted Installer 用户界面中看到集群配置设置。

其他资源

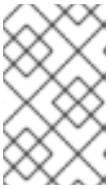
- 修改集群

- 安装一个混合架构集群
- 可选：在 Nutanix 上安装
- 可选：在 vSphere 上安装
- 可选：在 Oracle Cloud Infrastructure 上安装

4.5.1. 可选：安装 Operator

您可以在注册新集群时安装以下 Operator：

- OpenShift Virtualization Operator

  注意

  目前，IBM zSystems 和 IBM Power 不支持 OpenShift Virtualization。

- 多集群引擎 Operator
- OpenShift Data Foundation Operator
- LVM 存储 Operator

如果需要高级选项，请在安装集群后安装 Operator。

流程

- 运行以下命令：

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
```

```
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "4.15",
  "cpu_architecture" : "x86_64",
  "base_dns_domain": "example.com",
 "olm_operators": [
 { "name": "mce" } 1
 ,
 { "name": "odf" } 2
  ]
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

1

为 OpenShift Virtualization 指定 cnv，为多集群引擎指定 mce，为 OpenShift Data Foundation 指定 odf，或为 LVM Storage 指定 lvm。

2

本例在多节点集群上安装多集群引擎和 OpenShift Data Foundation。为单节点 OpenShift 集群指定 mce 和 lvm。

其他资源

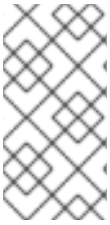
- OpenShift Virtualization 文档
- Red Hat OpenShift Cluster Manager 文档
- Red Hat OpenShift Data Foundation 文档
- Logical Volume Manager Storage 文档

4.6. 修改集群

要使用 API 修改集群定义，请使用 /v2/clusters/{cluster_id} 端点。修改集群资源是添加设置（如更改网络类型或启用用户管理的网络）的常见操作。如需了解在修改集群定义时您可以设置的字段的详情，请

参阅 API viewer 中的 v2-cluster-update-params 模型。

您可以从已注册的集群资源中添加或删除 Operator。



注意

要在节点上创建分区，请参阅 OpenShift Container Platform 文档中的 在节点上配置存储。

先决条件

- 您已创建了新的集群资源。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 修改集群。例如，更改 SSH 密钥：

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "ssh_public_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDZrD4LMkAEeoU2vShhF8VM+cCZtVRgB7t
qtsMxms2q3TOJZAgfuqReKYWm+OLOZTD+DO3Hn1pah/mU3u7uJfTUg4wEX0Le8zBu
9xJVym0BVmSFkzHflJVTn6SfZ81NqcalisGWkpmkKXVCdnVAX6RsbHfpGKk9YPQarm
RCn5KzkelJK4hrSWpBPjdzkFXalpf64JBZtew9XVYA3QeXklcFuq7NBuUH9BonroPEmlX
NOa41PUP1IWq3mERNgzHZiuU8Ks/pFuU5HCMvv4qbTOlhiig7vidlmHPpqYT/TCkuVi5w
0ZZgkkBeLnxWxH0ldrfzgFBYAxnpTU8lh/4VhG538lx1hxPaM6cXds2ic71mBbtbSrk+zjtN
PaeYk1O7UpcCw4jjHspU/rVV/DY51D5gSiiuaFPBMucnYPgUxy4FMBFfGrmGLlzTKiLzcz
0DiSz1jBeTQOX++1nz+KDLBD8CPdi5k4dq7lLkapRk85qdEvgaG5RlHMSPSS3wDrQ51f
D8= user@hostname"
}
' | jq
```

4.6.1. 修改 Operator

您可以从之前作为安装的一部分的已注册的集群资源中添加或删除 Operator。这只有在启动 OpenShift Container Platform 安装前才有可能。

您可以为 /v2/clusters/{cluster_id} 端点使用 PATCH 方法来设置所需的 Operator 定义。

先决条件

- 您已刷新了 API 令牌。
- 您已将 CLUSTER_ID 导出为一个环境变量。

流程

- 运行以下命令来修改 Operator：

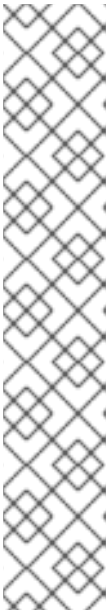
```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "olm_operators": [{"name": "mce"}, {"name": "cnv"}], 1
}
' | jq '.id'
```

1 为 OpenShift Virtualization 指定 cnv、为多集群引擎指定 mce、为 Red Hat OpenShift Data Foundation 指定 odf 或为 Logical Volume Manager Storage 指定 lvm。要删除之前安装的 Operator，请从值列表中排除它。要删除所有之前安装的 Operator，请指定一个空数组 ："olm_operators": []。

输出示例

```
{
 <various cluster properties>,
 "monitored_operators": [
  {
   "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
   "name": "console",
   "operator_type": "builtin",
   "status_updated_at": "0001-01-01T00:00:00.000Z",
```

```
    "timeout_seconds": 3600
   },
   {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "cvo",
    "operator_type": "builtin",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "timeout_seconds": 3600
   },
   {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "mce",
    "namespace": "multicluster-engine",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "multicluster-engine",
    "timeout_seconds": 3600
   },
   {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "cnv",
    "namespace": "openshift-cnv",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "hco-operatorhub",
    "timeout_seconds": 3600
   },
   {
    "cluster_id": "b5259f97-be09-430e-b5eb-d78420ee509a",
    "name": "lvm",
    "namespace": "openshift-local-storage",
    "operator_type": "olm",
    "status_updated_at": "0001-01-01T00:00:00.000Z",
    "subscription_name": "local-storage-operator",
    "timeout_seconds": 4200
   }
  ],
  <more cluster properties>
```



注意

输出是新集群状态的描述。输出中的 monitored_operators 属性包含两种类型的 Operator：

- "operator_type": "builtin" ：此类型的 Operator 是 OpenShift Container Platform 不可分割的一部分。
- "operator_type": "olm" ：此类型的 Operator 被用户手动或自动添加为一个依赖项。在本例中，LVM Storage Operator 被自动添加为 OpenShift Virtualization 的一个依赖项。

4.7. 注册新的基础架构环境

使用 Assisted Installer API 注册新集群定义后，使用 v2/infra-envs 端点创建一个基础架构环境。注册新的基础架构环境需要以下设置：

- name
- pull_secret
- cpu_architecture

有关注册新基础架构环境时可以设置的字段的详情，请参阅 API viewer 中的 infra-env-create-params 模型。您可以在创建基础架构环境后修改基础架构环境。作为最佳实践，请考虑在创建新基础架构环境时包括 cluster_id。cluster_id 将基础架构环境与集群定义相关联。在创建新基础架构环境时，辅助安装程序也会生成发现 ISO。

先决条件

- 您已生成了一个有效的 API_TOKEN。令牌每 15 分钟过期一次。
- 您已下载了 pull secret。

- 可选： 注册一个新的集群定义并导出 cluster_id。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 注册新的基础架构环境。提供名称，最好包含集群名称。本例提供集群 ID，用于将基础架构环境与集群资源关联。以下示例指定了 image_type。您可以指定 full-iso 或 minimal-iso。默认值为 minimal-iso。

   a. 可选： 您可以通过在请求中分片 pull secret 文件来注册新的基础架构环境：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt \
  --arg cluster_id ${CLUSTER_ID} '
    {
      "name": "testcluster-infra-env",
      "image_type":"full-iso",
      "cluster_id": $cluster_id,
      "cpu_architecture" : "<architecture_name>", (1)
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq '.id'
```

注意

(1) 表示有效的值。它们包括：x86_64,arm64,ppc64le,s390x,multi

   b. 可选： 您可以通过将配置写入 JSON 文件并在请求中引用它来注册新的基础架构环境：

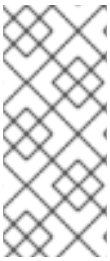
```
$ cat << EOF > infra-envs.json
{
 "name": "testcluster",
 "pull_secret": $PULL_SECRET,
 "proxy": {
   "http_proxy": "",
```

```
    "https_proxy": "",
    "no_proxy": ""
  },
  "ssh_authorized_key": "$CLUSTER_SSHKEY",
  "image_type": "full-iso",
  "cluster_id": "${CLUSTER_ID}",
  "openshift_version": "4.11"
}
EOF
```

```
$ curl -s -X POST "https://api.openshift.com/api/assisted-install/v2/infra-envs"
 -d @./infra-envs.json
 -H "Content-Type: application/json"
 -H "Authorization: Bearer $API_TOKEN"
 | jq '.id'
```

3. 为 INFRA_ENV_ID 变量分配返回的 id 并导出它：

```
$ export INFRA_ENV_ID=<id>
```



注意

创建基础架构环境并通过 cluster_id 将其与集群定义相关联后，您可以在 Assisted Installer Web 用户界面中看到集群设置。如果您关闭终端会话，则需要在新终端会话中重新导出 id。

4.8. 修改基础架构环境

您可以使用 /v2/infra-envs/{infra_env_id} 端点修改基础架构环境。修改基础架构环境是添加设置（如网络、SSH 密钥或 ignition 配置覆盖）的常见操作。

如需了解在修改基础架构环境时您可以设置的字段的详情，请参阅 API viewer 中的 infra-env-update-params 模型。修改新的基础架构环境时，辅助安装程序也会重新生成发现 ISO。

先决条件

- 您已创建了新的基础架构环境。

流程

1. 刷新 API 令牌：

-

```
$ source refresh-token
```

2. 修改基础架构环境：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
    {
      "image_type":"minimal-iso",
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq
```

4.8.1. 可选：添加内核参数

通过 Assisted Installer 为 Red Hat Enterprise Linux CoreOS (RHCOS)内核提供内核参数意味着在引导时将特定的参数或选项传递给内核，特别是您无法自定义发现 ISO 的内核参数。内核参数可控制内核行为的各个方面以及操作系统配置，影响硬件交互、系统性能和功能。内核参数用于自定义或告知节点的 RHCOS 内核有关硬件配置、调试首选项、系统服务以及其他低级设置。

RHCOS 安装程序 kargs modify 命令支持 append、delete 和 replace 选项。

您可以使用 /v2/infra-envs/{infra_env_id} 端点修改基础架构环境。修改新的基础架构环境时，辅助安装程序也会重新生成发现 ISO。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 修改内核参数：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
```

```
    {
      "kernel_arguments": [{ "operation": "append", "value": "<karg>=<value>" }], (1)
      "image_type":"minimal-iso",
      "pull_secret": $pull_secret[0] | tojson
    }
')" | jq
```

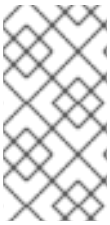
(1) 将 <karg> 替换为内核参数，将 <value> 替换为内核参数值。例如：rd.net.timeout.carrier=60。您可以通过为每个内核参数添加 JSON 对象来指定多个内核参数。

4.9. 添加主机

配置集群资源和基础架构环境后，下载发现 ISO 镜像。您可以从两个镜像中选择：

- 完整 ISO 镜像 ：在启动时使用完整的 ISO 镜像必须自我包含。该镜像包括引导和启动辅助安装程序代理所需的所有内容。ISO 镜像的大小大约为 1GB。这是使用 RHEL KVM 安装时对 s390x 架构的推荐方法。

- 最小 ISO 镜像：当虚拟介质连接的带宽有限时，使用最小 ISO 镜像。这是默认设置。镜像仅包含使用联网引导主机所需的内容。在引导时会下载大多数内容。ISO 镜像大小为 100MB。



注意

目前，对于具有 z/VM 的 IBM Z (s390x)上的安装不支持 ISO 镜像。详情请参阅 使用 iPXE 引导主机。

您可以使用三种方法，使用发现镜像引导主机。详情请参阅 使用发现镜像引导主机。

先决条件

- 您已创建了集群。

- 您已创建了基础架构环境。

- 您已完成配置。
- 如果集群主机位于需要使用代理的防火墙后面，您已为代理服务器的 HTTP 和 HTTPS URL 配置了用户名、密码、IP 地址和端口。

注意

代理用户名和密码必须采用 URL 编码。

- 您已选择了镜像类型，或者使用默认的 minimal-iso。

流程

1. 如果需要，配置发现镜像。详情请参阅 配置发现镜像。
2. 刷新 API 令牌：

```
$ source refresh-token
```

3. 获取下载 URL：

```
$ curl -H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID}/downloads/image-url
```

输出示例

```
{
  "expires_at": "2024-02-07T20:20:23.000Z",
  "url": "https://api.openshift.com/api/assisted-images/bytoken/<TOKEN>/<OCP_VERSION>/<CPU_ARCHITECTURE>/<FULL_OR_MINIMAL_IMAGE>.iso"
}
```

4. 下载发现镜像：

```
$ wget -O discovery.iso <url>
```

将 <url> 替换为上一步中的下载 URL。

5. 使用发现镜像引导主机。

6. 为主机分配角色。

其他资源

- 配置发现镜像
- 使用发现镜像引导主机
- 使用 API 在 Nutanix 中添加主机
- 在 vSphere 中添加主机
- 为主机分配角色
- 使用 iPXE 引导主机

4.10. 修改主机

添加主机后，根据需要修改主机。最常见的修改是 host_name 和 host_role 参数。

您可以使用 /v2/infra-envs/{infra_env_id}/hosts/{host_id} 端点修改主机。如需了解在修改主机时您可以设置的字段的详情，请参阅 API viewer 中的 host-update-params 模型。

主机可以是两个角色中的一个：

- master: 带有 master 角色的一个主机，作为控制平面（control plane）主机。
- worker: 带有 worker 角色的主机，作为 worker 主机。

默认情况下，辅助安装程序将主机设置为 auto-assign，这意味着安装程序会自动确定主机是 master 角色还是 worker 角色。使用以下流程设置主机的角色：

先决条件

- 您已将主机添加到集群中。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 获取主机 ID：

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

3. 修改主机：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
   {
     "host_role":"worker"
```

```
    "host_name" : "worker-1"
  }
' | jq
```

1 将 <host_id> 替换为主机的 ID。

4.10.1. 修改存储磁盘配置

主机发现期间检索到的每个主机都可能有多个存储磁盘。您可以选择修改每个磁盘的默认配置。

先决条件

- 配置集群并发现主机。详情请查看 *其它资源*。

查看存储磁盘

您可以查看集群中的主机，以及每个主机上的磁盘。这可让您对特定磁盘执行操作。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 获取集群的主机 ID：

```
$ curl -s "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

输出示例

```
$ "1022623e-7689-8b2d-7fbd-e6f4d5bb28e5"
```

注意

这是单个主机的 ID。多个主机 ID 用逗号分开。

3. 获取特定主机的磁盘：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-H "Authorization: Bearer ${API_TOKEN}" \
| jq '.inventory | fromjson | .disks'
```

1 将 <host_id> 替换为相关主机的 ID。

输出示例

```
$ [
  {
    "by_id": "/dev/disk/by-id/wwn-0x6c81f660f98afb002d3adc1a1460a506",
    "by_path": "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:0:0",
    "drive_type": "HDD",
    "has_uuid": true,
    "hctl": "1:2:0:0",
    "id": "/dev/disk/by-id/wwn-0x6c81f660f98afb002d3adc1a1460a506",
    "installation_eligibility": {
      "eligible": true,
      "not_eligible_reasons": null
    },
    "model": "PERC_H710P",
    "name": "sda",
    "path": "/dev/sda",
    "serial": "0006a560141adc3a2d00fb8af960f681",
    "size_bytes": 6595056500736,
    "vendor": "DELL",
    "wwn": "0x6c81f660f98afb002d3adc1a1460a506"
  }
]
```

注意

这是一个磁盘的输出。它包含磁盘的 disk_id 和 installation_eligibility 属性。

更改安装磁盘

辅助安装程序默认随机分配一个安装磁盘。如果主机有多个存储磁盘，您可以选择一个不同的磁盘作为安装磁盘。这会自动取消之前磁盘的分配。

您可以选择任何其 installation_eligibility 属性为 eligible: true 的磁盘为安装磁盘。

流程

1. 获取主机和存储磁盘 ID。详情请参阅 *查看存储磁盘*。

2. 可选：识别当前安装磁盘：

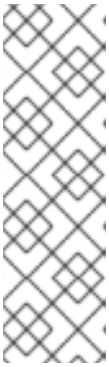
```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-H "Authorization: Bearer ${API_TOKEN}" \
| jq '.installation_disk_id'
```

1 将 <host_id> 替换为相关主机的 ID。

3. 分配一个新的安装磁盘：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Content-Type: application/json" \
-H "Authorization: Bearer ${API_TOKEN}" \

{
  "disks_selected_config": [
    {
      "id": "<disk_id>", 2
      "role": "install"
```

```
    }
  ]
}
```



注意

1 将 <host_id> 替换为主机的 ID。

2 将 <disk_id> 替换为新安装磁盘的 ID。

禁用磁盘格式化

辅助安装程序在安装过程中默认将所有可引导磁盘标记为进行格式化，无论它们是否已被定义为安装磁盘。格式化会导致数据丢失。

您可以选择禁用特定磁盘的格式化。这应该谨慎执行，因为可引导磁盘可能会干扰安装过程，主要在引导顺序方面。

您无法禁用对安装磁盘的格式化。

流程

1. 获取主机和存储磁盘 ID。详情请参阅 *查看存储磁盘*。

2. 运行以下命令：

```
$  curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \ 1
-X PATCH \
-H "Content-Type: application/json" \
-H "Authorization: Bearer ${API_TOKEN}" \

{
 "disks_skip_formatting": [
   {
     "disk_id": "<disk_id>", 2
     "skip_formatting": true 3
   }
 ]
}
```

注意

1 将 <host_id> 替换为主机的 ID。

2 将 <disk_id> 替换为磁盘的 ID。如果有多个磁盘，使用逗号将 ID 分开。

3 要重新启用格式化，请将值改为 false。

4.11. 添加自定义清单

自定义清单是一个 JSON 或 YAML 文件，其包含辅助安装程序用户界面目前不支持的高级配置。您可以创建一个自定义清单，或使用第三方提供的清单。要使用 API 创建一个自定义清单，请使用 /v2/clusters/$CLUSTER_ID/manifests 端点。

您可以使用辅助安装程序 API 将 base64 编码的自定义清单上传到 openshift 文件夹或 manifests 文件夹。对允许的自定义清单数量没有限制。

一次只能上传一个 base64 编码的 JSON 清单。但是，每个上传的 base64 编码的 YAML 文件可以包含多个自定义清单。上传多文档 YAML 清单比单独添加 YAML 文件要快。

对于包含一个自定义清单的文件，可接受的文件扩展名包括 .yaml、.yml 或 .json。

一个自定义清单示例

```
{
  "apiVersion": "machineconfiguration.openshift.io/v1",
  "kind": "MachineConfig",
  "metadata": {
    "labels": {
      "machineconfiguration.openshift.io/role": "primary"
    },
    "name": "10_primary_storage_config"
  },
  "spec": {
    "config": {
      "ignition": {
        "version": "3.2.0"
```

```
        },
        "storage": {
          "disks": [
            {
              "device": "</dev/xxyN>",
              "partitions": [
                {
                  "label": "recovery",
                  "startMiB": 32768,
                  "sizeMiB": 16384
                }
              ]
            }
          ],
          "filesystems": [
            {
              "device": "/dev/disk/by-partlabel/recovery",
              "label": "recovery",
              "format": "xfs"
            }
          ]
        }
      }
    }
}
```

对于包含多个自定义清单的文件，可接受的文件类型包括 .yaml 或 .yml。

多个自定义清单示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - loglevel=7
---
apiVersion: machineconfiguration.openshift.io/v2
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-openshift-machineconfig-worker-kargs
```

```
spec:
  kernelArguments:
    - loglevel=5
```

注意

- 当在 Oracle Cloud Infrastructure (OCI)外部平台上安装 OpenShift Container Platform 时，您必须添加 Oracle 提供的自定义清单。对于其它外部合作伙伴集成，如 vSphere 或 Nutanix，此步骤是可选的。
- 有关自定义清单的更多信息，请参阅 *其它资源*。

先决条件

- 您已生成了一个有效的 API_TOKEN。令牌每 15 分钟过期一次。
- 您已注册了一个新的集群定义，并将 cluster_id 导出到 $CLUSTER_ID BASH 变量。

流程

1. 创建一个自定义清单文件。
2. 使用合适的文件格式的扩展名保存自定义清单文件。
3. 刷新 API 令牌：

```
$ source refresh-token
```

4. 运行以下命令，将自定义清单添加到集群中：

```
$ curl -X POST "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID/manifests" \
    -H "Authorization: Bearer $API_TOKEN" \
    -H "Content-Type: application/json" \
    -d '{
```

```
        "file_name":"manifest.json",
        "folder":"manifests",
        "content":"'"$(base64 -w 0 ~/manifest.json)"'"
    }' | jq
```

将 manifest.json 替换为清单文件的名称。manifest.json 的第二个实例是文件的路径。确保路径正确。

输出示例

```
{
  "file_name": "manifest.json",
  "folder": "manifests"
}
```

注意

base64 -w 0 命令使用 base64 将清单编码为一个字符串，并省略回车。带回车的的编码将产生一个异常。

5. 验证辅助安装程序是否添加了清单：

```
curl -X GET "https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/manifests/files?folder=manifests&file_name=manifest.json" -H "Authorization: Bearer $API_TOKEN"
```

将 manifest.json 替换为清单文件的名称。

其他资源

- 清单配置文件
- 多文档 YAML 文件

4.12. 预安装验证

Assisted Installer 确保集群在安装前满足先决条件，因为它消除了复杂的安装后故障排除，从而节省大量时间和工作时间。在安装集群前，请确保集群和每个主机都通过了预安装验证。

其他资源

- 预安装验证

4.13. 安装集群

一旦集群主机经过验证，您就可以安装集群。

先决条件

- 您已创建了集群和基础架构环境。
- 您已将主机添加到基础架构环境中。
- 主机已通过验证。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 安装集群：

```
$ curl -H "Authorization: Bearer $API_TOKEN" \
-X POST \
https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/actions/install |
jq
```

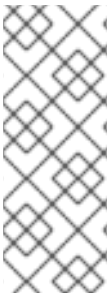
3. 完成所有安装后平台集成步骤。

其他资源

- Nutanix 安装后配置
- vSphere 安装后配置

第 5 章 可选：启用磁盘加密

您可以使用 TPM v2 或 Tang 加密模式启用安装磁盘加密。



注意

在某些情况下，当您在固件中为裸机主机启用 TPM 磁盘加密时，然后从您使用辅助安装程序生成的 ISO 引导它，集群部署可能会卡住。如果以前的安装在主机中保留了 TPM 加密密钥，则可能会出现这种情况。如需更多信息，请参阅 BZ#2011634。如果您遇到这个问题，请联系红帽支持。

5.1. 启用 TPM V2 加密

先决条件

- 检查每个主机上的 BIOS 是否启用了 TPM v2 加密。大多数 Dell 系统都需要这个设置。检查您的计算机手册。Assisted Installer 还验证固件中是否启用了 TPM。详情请查看 Assisted Installer API 中的 disk-encruption 模式。

重要

验证每个节点上是否安装了 TPM v2 加密芯片，并在固件中启用。

流程

1. 可选： 使用 web 控制台，在用户界面向导的 集群详情 步骤中，选择在 control plane 节点、worker 或这两个节点上启用 TPM v2 加密。

2. 可选：使用 API，请遵循"修改主机"过程。将 disk_encryption.enable_on 设置设置为 all, masters, 或 workers。将 disk_encryption.mode 设置设置为 tpmv2。

   a. 刷新 API 令牌：

   ```
   $ source refresh-token
   ```

   b. 启用 TPM v2 加密：

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "disk_encryption": {
    "enable_on": "none",
    "mode": "tpmv2"
  }
}
' | jq
```

enable_on 的有效设置是 all, master, worker, 或 none。

5.2. 启用 TANG 加密

先决条件

- 您可以使用 Red Hat Enterprise Linux(RHEL)8 机器来生成 Tang Exchange 密钥的指纹。

流程

1. 设置 Tang 服务器或访问现有服务器。具体步骤请查看 网络绑定磁盘加密。您可以设置多个 Tang 服务器，但 Assisted Installer 在安装过程中需要可以连接到所有 Tang 服务器。

2. 在 Tang 服务器上，使用 tang-show-keys 检索 Tang 服务器的 thumbprint：

```
$ tang-show-keys <port>
```

可选：将 <port> 替换为端口号。默认端口号为 80。

thumbprint 示例

```
1gYTN_LpU9ZMB35yn5IbADY5OQ0
```

3. 可选：使用 jose 获取 Tang 服务器的 thumbprint。

   a. 确保在 Tang 服务器中安装了 jose ：

   ```
   $ sudo dnf install jose
   ```

   b. 在 Tang 服务器上，使用 jose 检索 thumbprint:

   ```
   $ sudo jose jwk thp -i /var/db/tang/<public_key>.jwk
   ```

   将 <public_key> 替换为 Tang 服务器的公共交换密钥。

   thumbprint 示例

   ```
   1gYTN_LpU9ZMB35yn5lbADY5OQ0
   ```

4. 可选：在用户界面向导的 Cluster details 步骤中，选择在 control plane 节点、worker 或这两个节点上启用 Tang 加密。您需要为 Tang 服务器输入 URL 和 thumbprints。

5. 可选：使用 API，请遵循"修改主机"过程。

   a. 刷新 API 令牌：

   ```
   $ source refresh-token
   ```

   b. 将 disk_encryption.enable_on 设置设置为 all, masters, 或 workers。将 disk_encryption.mode 设置设置为 tang。设置 disk_encyrption.tang_servers，以提供一个或多个 Tang 服务器的 URL 和 thumbprint 详情：

   ```
   $ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
   -X PATCH \
   -H "Authorization: Bearer ${API_TOKEN}" \
   ```

```
-H "Content-Type: application/json" \
-d '
{
  "disk_encryption": {
    "enable_on": "all",
    "mode": "tang",
    "tang_servers": "
[{\"url\":\"http://tang.example.com:7500\",\"thumbprint\":\"PLjNyRdGw03zlRoGjQY
MahSZGu9\"},
{\"url\":\"http://tang2.example.com:7500\",\"thumbprint\":\"XYjNyRdGw03zlRoGjQY
MahSZGu3\"}]"
  }
}
' | jq
```

enable_on 的有效设置是 all, master, worker, 或 none。在 tang_servers 值中，注释掉对象中的引号。

5.3. 其他资源

- 修改主机

第 6 章 可选：配置可调度的 CONTROL PLANE 节点

在高可用性部署中，三个或更多节点组成 control plane。control plane 节点用于管理 OpenShift Container Platform 并运行 OpenShift 容器。剩余的节点是 worker，用于运行客户容器和工作负载。可以是一个到数千个 worker 节点。

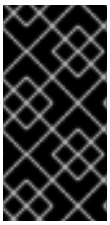
对于单节点 OpenShift 集群或包含四个节点的集群，系统会自动调度工作负载在 control plane 节点上运行。

对于有 5 到 10 个节点的集群，除了在 worker 节点上运行外，您可以选择将工作负载调度到 control plane 节点上运行。建议使用这个选项来提高效率并防止出现资源使用率不足的情况。您可以在安装设置过程中选择这个选项，也可以在安装后进行选择。

对于有多于十个节点的大型集群，不建议使用这个选项。

本节解释了如何使用 Assisted Installer Web 控制台和 API 来调度工作负载在 control plane 节点上运行，作为安装设置的一部分。

有关如何在安装后配置可调度 control plane 节点的说明，请参阅 OpenShift Container Platform 文档中的 将 control plane 节点配置为可以调度。



重要

当您将 control plane 节点从默认的不可调度配置为可以调度时，需要额外的订阅。这是因为 control plane 节点随后变为 worker 节点。

6.1. 使用 WEB 控制台配置可调度的 CONTROL PLANE

先决条件

- 您已设置了集群详情。
- 您要安装 OpenShift Container Platform 4.14 或更高版本。

流程

1. 登录到 Red Hat Hybrid Cloud 控制台，并按照使用 Assisted Installer Web 控制台安装 OpenShift Container Platform 的说明进行操作。详情请参阅 *附加资源中的使用 Assisted Installer Web 控制台安装*。

2. 当您进入 Host discovery 页面时，点 Add hosts。

3. （可选）根据需要更改 Provisioning 类型和其他设置。所有选项都与可调度的 control plane 兼容。

4. 点 Generate Discovery ISO 下载 ISO。

5. 将 control plane 节点上的运行工作负载 设置为 on。

注意

对于有四个或更少节点的集群，这个选项会被自动激活且不能改变。

6. 点击 Next。

6.2. 使用 API 配置可调度的 CONTROL PLANE

使用 schedulable_masters 属性启用工作负载在 control plane 节点上运行。

先决条件

- 您已生成了一个有效的 API_TOKEN。令牌每 15 分钟过期一次。
- 您已创建了 $PULL_SECRET 变量。
- 您要安装 OpenShift Container Platform 4.14 或更高版本。

流程

1. 按照使用 Assisted Installer API 安装辅助安装程序的说明进行操作。详情请参阅 *其它资源* 部分中的 *使用辅助安装程序 API 安装*。

2. 当您进入注册新集群这一步时，设置 schedulable_masters 属性，如下所示：

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
  "schedulable_masters": true (1)
}
' | jq
```

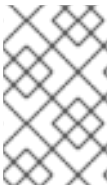
(1) 启用在 control plane 节点上调度工作负载。

6.3. 其他资源

- 使用 Assisted Installer Web 控制台安装
- 使用辅助安装程序 API 安装

第 7 章 配置发现镜像

Assisted Installer 使用初始镜像来运行在尝试安装 OpenShift Container Platform 前执行硬件和网络验证的代理。您可以使用 Ignition 自定义发现镜像。



注意

对发现镜像的修改不会在系统中保留。

7.1. 创建 IGNITION 配置文件

Ignition 是一个低级系统配置实用程序，它是 *initramfs* 临时初始根文件系统的一部分。当 Ignition 在第一次引导时运行时，它会在 Ignition 配置文件中找到配置数据，并在调用 switch_root 之前将其应用到主机，以 pivot 到主机的根文件系统。

Ignition 使用 JSON 配置规格文件来代表第一次引导时发生的一组更改。

重要

不支持比 3.2 更新的 Ignition 版本，并引发一个错误。

流程

1. 创建 Ignition 文件并指定配置规格版本：

```
$ vim ~/ignition.conf
```

```
{
  "ignition": { "version": "3.1.0" }
}
```

2. 将配置数据添加到 Ignition 文件。例如，为 core 用户添加密码。

   a. 生成密码哈希：

```
$ openssl passwd -6
```

b. 在 core 用户中添加生成的密码哈希：

```
{
  "ignition": { "version": "3.1.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "passwordHash":
"$6$spam$M5LGSMGyVD.9XOboxcwrsnwNdF4irpJdAWy.1Ry55syyUiUsslzIAHaOr
UHr2zg6ruD8YNBPW9kW0H8EnKXyc1"
      }
    ]
  }
}
```

3. 保存 Ignition 文件并将其导出到 IGNITION_FILE 变量：

```
$ export IGNITION_FILE=~/ignition.conf
```

7.2. 使用 IGNITION 修改发现镜像

创建 Ignition 配置文件后，您可以使用 Assisted Installer API 修补基础架构环境来修改发现镜像。

先决条件

- 如果使用 Web 控制台创建集群，则已设置了 API 身份验证。
- 您有一个基础架构环境，并将基础架构环境 id 导出至 INFRA_ENV_ID 变量。
- 您有一个有效的 Ignition 文件，并将文件名导出为 $IGNITION_FILE。

流程

1. 创建 ignition_config_override JSON 对象并将其重定向到文件中：

```
$ jq -n \
  --arg IGNITION "$(jq -c . $IGNITION_FILE)" \
  '{ignition_config_override: $IGNITION}' \
  > discovery_ignition.json
```

2. 刷新 API 令牌：

```
$ source refresh-token
```

3. 对基础架构环境进行补丁：

```
$ curl \
  --header "Authorization: Bearer $API_TOKEN" \
  --header "Content-Type: application/json" \
  -XPATCH \
  -d @discovery_ignition.json \
  https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID | jq
```

ignition_config_override 对象引用 Ignition 文件。

4. 下载更新的发现镜像。

第 8 章 使用发现镜像引导主机

Assisted Installer 使用初始镜像来运行在尝试安装 OpenShift Container Platform 前执行硬件和网络验证的代理。您可以使用三种方法使用发现镜像引导主机：

- USB 驱动器
- RedFish 虚拟介质
- iPXE

8.1. 在 USB 驱动器中创建 ISO 镜像

您可以使用包含发现 ISO 镜像的 USB 驱动器安装 Assisted Installer 代理。使用 USB 驱动器启动主机为软件安装准备主机。

流程

1. 在管理主机上，在 USB 端口中插入 USB 驱动器。
2. 将 ISO 镜像复制到 USB 驱动器中，例如：

```
# dd if=<path_to_iso> of=<path_to_usb> status=progress
```

其中：

<path_to_iso>

是下载的发现 ISO 文件的相对路径，如 discovery.iso。

<path_to_usb>

是连接的 USB 驱动器的位置，例如 /dev/sdb。

将 ISO 复制到 USB 驱动器后，您可以使用 USB 驱动器在集群主机上安装辅助安装程序代理。

8.2. 使用 USB 驱动器引导

要使用可引导 USB 驱动器将节点注册到 Assisted Installer，请使用以下步骤。

流程

1. 将 RHCOS 发现 ISO USB 驱动器插入到目标主机中。
2. 在服务器固件设置中配置启动驱动器顺序，以便从附加的发现 ISO 启动，然后重启服务器。
3. 等待主机启动。

   a. 对于 Web 控制台安装，在管理主机上返回浏览器。等待主机出现在已发现的主机列表中。

   b. 对于 API 安装，刷新令牌，检查启用的主机计数，并收集主机 ID：

```
$ source refresh-token
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.enabled_host_count'
```

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

输出示例

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

8.3. 使用 REDFISH API 从 HTTP 托管 ISO 镜像引导

您可以使用 Redfish Baseboard Management Controller (BMC) API 安装的 ISO 来置备网络中的主机。

先决条件

- 下载安装 Red Hat Enterprise Linux CoreOS (RHCOS) ISO。

流程

1. 将 ISO 文件复制到网络中的 HTTP 服务器。
2. 从托管 ISO 文件引导主机，例如：
   a. 运行以下命令调用 redfish API，将托管的 ISO 设置为 VirtualMedia 引导介质：

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"Image":"<hosted_iso_file>", "Inserted": true}' \
-H "Content-Type: application/json" \
-X POST
<host_bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/VirtualMedia.InsertMedia
```

其中：

<bmc_username>:<bmc_password>

是目标主机 BMC 的用户名和密码。

<hosted_iso_file>

是托管安装 ISO 的 URL，例如：https://example.com/rhcos-live-minimal.iso。ISO 必须从目标主机机器中访问。

<host_bmc_address>

是目标主机计算机的 BMC IP 地址。

b. 运行以下命令，将主机设置为从 VirtualMedia 设备引导：

```
$ curl -k -u <bmc_username>:<bmc_password> \
-X PATCH -H 'Content-Type: application/json' \
-d '{"Boot": {"BootSourceOverrideTarget": "Cd", "BootSourceOverrideMode":
"UEFI", "BootSourceOverrideEnabled": "Once"}}' \
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1
```

c. 重启主机：

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "ForceRestart"}' \
-H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerS
ystem.Reset
```

d. 可选：如果主机已关闭，您可以使用 {"ResetType": "On"} 开关引导它。运行以下命令：

```
$ curl -k -u <bmc_username>:<bmc_password> \
-d '{"ResetType": "On"}' -H 'Content-type: application/json' \
-X POST
<host_bmc_address>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerS
ystem.Reset
```

8.4. 使用 IPXE 引导主机

Assisted Installer 提供了一个 iPXE 脚本，包括为基础架构环境引导发现镜像所需的所有工件。由于 iPXE 的当前 HTTPS 实施的限制，建议是下载并公开 HTTP 服务器中所需的工件。目前，即使 iPXE 支持 HTTPS 协议，支持的算法也比较旧且不推荐。

支持的密码的完整列表位于 https://ipxe.org/crypto 中。

先决条件

- 已使用 API 创建基础架构环境，或者已使用 Web 控制台创建集群。

- 在 shell 中将您的基础架构环境 ID 导出为 $INFRA_ENV_ID。

- 您在访问 API 时具有凭证，并在 shell 中将令牌导出为 $API_TOKEN。

注意

如果使用 Web 控制台配置 iPXE，则 $INFRA_ENV_ID 和 $API_TOKEN 变量被预先设置。

- 您有一个 HTTP 服务器来托管镜像。

注意

IBM Power 只支持 PXE，这还需要： 您已在 /var/lib/tftpboot安装了 grub2。已安装了 DHCP 和 TFTP for PXE

流程

1. 直接从 Web 控制台下载 iPXE 脚本，或者从 Assisted Installer 获取 iPXE 脚本：

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/infra-
envs/$INFRA_ENV_ID/downloads/files?file_name=ipxe-script > ipxe-script
```

Example

```
#!ipxe
initrd --name initrd http://api.openshift.com/api/assisted-
images/images/<infra_env_id>/pxe-initrd?arch=x86_64&image_token=
<token_string>&version=4.10
kernel http://api.openshift.com/api/assisted-images/boot-artifacts/kernel?
arch=x86_64&version=4.10 initrd=initrd
coreos.live.rootfs_url=http://api.openshift.com/api/assisted-images/boot-
artifacts/rootfs?arch=x86_64&version=4.10 random.trust_cpu=on
rd.luks.options=discard ignition.firstboot ignition.platform.id=metal console=tty1
```

```
console=ttyS1,115200n8 coreos.inst.persistent-kargs="console=tty1
console=ttyS1,115200n8"
boot
```

2. 通过从 ipxe-script 中提取 URL 下载所需的工件。

   a. 下载初始 RAM 磁盘：

   ```
   $ awk '/^initrd /{print $NF}' ipxe-script | curl -o initrd.img
   ```

   b. 下载 linux 内核：

   ```
   $ awk '/^kernel /{print $2}' ipxe-script | curl -o kernel
   ```

   c. 下载根文件系统：

   ```
   $ grep ^kernel ipxe-script | xargs -n1| grep ^coreos.live.rootfs_url | cut -d = -f 2- |
   curl -o rootfs.img
   ```

3. 将 URL 更改为 ipxe-script' 中不同的工件，以匹配本地 HTTP 服务器。例如：

   ```
   #!ipxe
   set webserver http://192.168.0.1
   initrd --name initrd $webserver/initrd.img
   kernel $webserver/kernel initrd=initrd coreos.live.rootfs_url=$webserver/rootfs.img
   random.trust_cpu=on rd.luks.options=discard ignition.firstboot
   ignition.platform.id=metal console=tty1 console=ttyS1,115200n8
   coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8"
   boot
   ```

4. 可选：当在 IBM zSystems 上使用 RHEL KVM 安装时，您必须通过指定附加内核参数来引导主机

   ```
   random.trust_cpu=on rd.luks.options=discard ignition.firstboot
   ignition.platform.id=metal console=tty1 console=ttyS1,115200n8
   coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8
   ```

注意

如果您在 RHEL KVM 上使用 iPXE 安装，在一些情况下，虚拟机主机上的虚拟机不会在第一次引导时重启，需要手动启动。

5. 可选：在 IBM Power 上安装时，您必须下载 intramfs、kernel 和 root，如下所示：

   a. 将 initrd.img 和 kernel.img 复制到 PXE 目录 '/var/lib/tftpboot/rhcos '

   b. 将 rootfs.img 复制到 HTTPD 目录 '/var/www/html/install '

   c. 在 '/var/lib/tftpboot/boot/grub2/grub.cfg ' 中添加以下条目：

```
if [ ${net_default_mac} == fa:1d:67:35:13:20 ]; then
default=0
fallback=1
timeout=1
menuentry "CoreOS (BIOS)" {
echo "Loading kernel"
linux "/rhcos/kernel.img" ip=dhcp rd.neednet=1 ignition.platform.id=metal
ignition.firstboot coreos.live.rootfs_url=http://9.114.98.8:8000/install/rootfs.img
echo "Loading initrd"
initrd "/rhcos/initrd.img"
}
fi
```

第 9 章 为主机分配角色

您可以为发现的主机分配角色。这些角色定义集群中主机的功能。角色可以是标准 Kubernetes 类型之一： control plane (master) 或 worker。

主机必须满足您选择的角色的最低要求。您可以通过引用本文档的先决条件部分或使用 preflight 要求 API 来查找硬件要求。

如果您没有选择角色，系统会为您选择一个。您可以在安装启动前随时更改角色。

9.1. 使用 WEB 控制台选择角色

您可以在主机完成其发现后选择角色。

流程

1. 进入 Host Discovery 选项卡，向下滚动到 Host Inventory 表。
2. 选择所需主机的 Auto-assign 下拉菜单。
3. 选择 Control plane 节点 来为这个主机分配一个 control plane 角色。
4. 选择 Worker 为这个主机分配一个 worker 角色。
5. 检查验证状态。

9.2. 使用 API 选择角色

您可以使用 /v2/infra-envs/{infra_env_id}/hosts/{host_id} 端点为主机选择一个角色。主机可以是两个角色之一：

- master: 带有 master 角色的一个主机，作为控制平面（control plane）主机。

- **worker: 带有 worker 角色的主机，作为 worker 主机。**

默认情况下，辅助安装程序将主机设置为 auto-assign，这意味着安装程序将确定主机是否是 master 角色还是 worker 角色。使用这个流程设置主机的角色。

先决条件

- **您已将主机添加到集群中。**

流程

1. **刷新 API 令牌：**

```
$ source refresh-token
```

2. **获取主机 ID：**

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID" \
--header "Content-Type: application/json" \
  -H "Authorization: Bearer $API_TOKEN" \
| jq '.host_networks[].host_ids'
```

输出示例

```
[
  "1062663e-7989-8b2d-7fbb-e6f4d5bb28e5"
]
```

3. **修改 host_role 设置：**

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
```

```
-d '
  {
    "host_role":"worker"
  }
' | jq
```

将 <host_id> 替换为主机的 ID。

9.3. 自动分配角色

如果您没有自行分配角色，安装程序会自动为主机选择一个角色。角色选择机制决定主机的内存、CPU 和磁盘空间。它的目的是，为可以满足 control plane 节点的最低要求的 3 个弱主机分配 control plane 角色。所有其他主机默认为 worker 节点。目标是提供足够的资源来运行 control plane，并保留具有更多资源的主机用于运行实际工作负载。

您可以在安装前随时覆盖 auto-assign 决定。

这个验证可以确保，自动选择是一个有效的值。

9.4. 其他资源

先决条件

第 10 章 预安装验证

10.1. 预安装验证的定义

Assisted Installer 旨在使集群安装尽可能简单、高效且无错误。Assisted Installer 在开始安装前对配置和收集的遥测执行验证检查。

Assisted Installer 将使用安装前提供的信息，如 control plane 拓扑、网络配置和主机名。它还将使用您要安装的主机的实时遥测。

当主机引导发现 ISO 时，代理将在主机上启动。代理会将主机状态的信息发送到 Assisted Installer。

辅助安装程序使用所有这些信息来计算实时安装验证。所有验证都可以是，阻止安装，或不阻塞安装。

10.2. 阻塞和非阻塞验证

阻塞验证将阻止安装的进度，这意味着您需要解决问题，并在继续操作前通过阻塞验证。

非阻塞验证是一个警告，并将告诉您可能引起问题的事情。

10.3. 验证类型

Assisted Installer 执行两种类型的验证：

主机

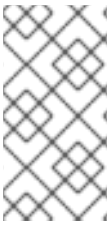
主机验证可确保给定主机的配置对安装有效。

Cluster

集群验证可确保整个集群的配置对安装有效。

10.4. 主机验证

10.4.1. 使用 REST API 获取主机验证



注意

如果使用 Web 控制台，这些验证都会按名称显示。要获取与标签一致的验证列表，请使用以下步骤。

先决条件

- 已安装 jq 工具。
- 已使用 API 创建 Infrastructure 环境，或使用 Web 控制台创建集群。
- 您的主机使用发现 ISO 引导
- 在 shell 中以 CLUSTER_ID 的形式导出集群 ID。
- 您在访问 API 时具有凭证，并在 shell 中导出令牌作为 API_TOKEN。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 获取所有主机的所有验证：

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
  | jq -r .[].validations_info \
  | jq 'map(.[])'
```

3. 获取所有主机的没有通过的验证：

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
```

```
https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID/hosts \
| jq -r .[].validations_info \
| jq 'map(.[]) | map(select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.4.2. 详细的主机验证

参数	验证类型	描述
connected	非阻塞	检查主机最近与 Assisted Installer 进行通信。
has-inventory	非阻塞	检查 Assisted Installer 是否从主机收到清单。
has-min-cpu-cores	非阻塞	检查 CPU 内核数是否满足最低要求。
has-min-memory	非阻塞	检查内存量是否满足最低要求。
has-min-valid-disks	非阻塞	检查至少一个可用磁盘是否满足资格标准。
has-cpu-cores-for-role	阻塞	检查内核数是否满足主机角色的最低要求。
has-memory-for-role	阻塞	检查内存量是否满足主机角色的最低要求。
ignition-downloadable	阻塞	对于第 2 天，检查主机是否可以从第 1 天集群中下载 ignition 配置。
belongs-to-majority-group	阻塞	大多数组是集群中最大的 full-mesh 连接组，所有成员都可以与所有其他成员通信。此验证检查多节点中的第 1 天集群在大多数组中。
valid-platform-network-settings	阻塞	检查平台是否对网络设置有效。
ntp-synced	非阻塞	检查 NTP 服务器是否已成功用于同步主机上的时间。
container-images-available	非阻塞	检查容器镜像是否已成功从镜像 registry 中拉取。
sufficient-installation-disk-speed	阻塞	检查磁盘是否加快了之前安装的指标满足要求（如果存在）。
sufficient-network-latency-requirement-for-role	阻塞	检查集群中主机之间的平均网络延迟是否满足要求。
sufficient-packet-loss-requirement-for-role	阻塞	检查集群中主机之间的网络数据包丢失是否满足要求。
has-default-route	阻塞	检查主机是否配置了默认路由。

参数	验证类型	描述
api-domain-name-resolved-correctly	阻塞	对于带有用户管理网络的多节点集群。检查主机是否可以解析集群的 API 域名。
api-int-domain-name-resolved-correctly	阻塞	对于带有用户管理网络的多节点集群。检查主机是否可以解析集群的内部 API 域名。
apps-domain-name-resolved-correctly	阻塞	对于带有用户管理网络的多节点集群。检查主机是否可以解析集群的内部 apps 域名。
compatible-with-cluster-platform	非阻塞	检查主机是否与集群平台兼容
dns-wildcard-not-configured	阻塞	检查通配符 DNS *.<cluster_name>.<base_domain> 是否已配置，因为这会导致 OpenShift 的已知问题
disk-encryption-requirements-satisfied	非阻塞	检查配置的主机和磁盘加密的类型是否满足要求。
non-overlapping-subnets	阻塞	检查此主机没有重叠的子网。
hostname-unique	阻塞	检查主机名是否在集群中是唯一的。
hostname-valid	阻塞	检查主机名的有效性，这意味着它与常规主机名格式匹配且没有被禁止。
belongs-to-machine-cidr	阻塞	检查主机 IP 是否在机器 CIDR 的地址范围内。
lso-requirements-satisfied	阻塞	验证集群是否满足 Local Storage Operator 的要求。
odf-requirements-satisfied	阻塞	验证集群是否满足 Openshift Data Foundation Operator 的要求。 ● 集群至少有 3 个主机。 ● 集群只有 3 个 master 或至少 3 个 worker。 ● 集群有 3 个有资格的磁盘，每个主机都必须有合格的磁盘。 ● 对于有超过三个主机的集群，主机角色不能为 "Auto Assign"。

参数	验证类型	描述
cnv-requirements-satisfied	阻塞	验证集群是否满足容器原生虚拟化的要求。 ● 主机的 BIOS 必须启用 CPU 虚拟化。 ● 主机必须具有足够的 CPU 内核和 RAM 用于容器原生虚拟化。 ● 如果需要，将验证 Host Path Provisioner。
lvm-requirements-satisfied	阻塞	验证集群是否满足逻辑卷管理器 Operator 的要求。 ● 主机至少有一个额外的空磁盘，没有分区且未被格式化。
vsphere-disk-uuid-enabled	非阻塞	验证每个有效磁盘是否将 *disk.EnableUUID* 设置为 *true*。在 VSphere 中，这将导致每个磁盘都有一个 UUID。
compatible-agent	阻塞	检查发现代理版本是否与代理 docker 镜像版本兼容。
no-skip-installation-disk	阻塞	检查安装磁盘是否没有跳过磁盘格式。
no-skip-missing-disk	阻塞	检查标记为跳过格式的所有磁盘是否在清单中。磁盘 ID 可能会在重启后改变，此验证可防止出现此问题的问题。
media-connected	阻塞	检查安装介质到主机的连接。
machine-cidr-defined	非阻塞	检查集群是否存在机器网络定义。
id-platform-network-settings	阻塞	检查平台是否与网络设置兼容。只有安装单节点 Openshift 或使用用户管理时，某些平台才被允许。

10.5. 集群验证

10.5.1. 使用 REST API 获取集群验证

如果使用 Web 控制台，这些验证都会按名称显示。要获取与标签一致的验证列表，请按照以下流程操作。

先决条件

- **已安装 jq 工具。**

- 已使用 API 创建 Infrastructure 环境，或使用 Web 控制台创建集群。
- 在 shell 中以 CLUSTER_ID 的形式导出集群 ID。
- 您在访问 API 时具有凭证，并在 shell 中导出令牌作为 API_TOKEN。

流程

1. 刷新 API 令牌：

```
$ source refresh-token
```

2. 获取所有集群验证：

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
  | jq -r .validations_info \
  | jq 'map(.[])'
```

3. 获取没有通过的集群验证：

```
$ curl \
  --silent \
  --header "Authorization: Bearer $API_TOKEN" \
  https://api.openshift.com/api/assisted-install/v2/clusters/$CLUSTER_ID \
  | jq -r .validations_info \
  | jq '. | map(.[] | select(.status=="failure" or .status=="pending")) | select(length>0)'
```

10.5.2. 详细的集群验证

参数	验证类型	描述
machine-cidr-defined	非阻塞	检查集群是否存在机器网络定义。
cluster-cidr-defined	非阻塞	检查集群是否存在集群网络定义。
service-cidr-defined	非阻塞	检查集群是否存在服务网络定义。

参数	验证类型	描述
no-cidrs-overlapping	阻塞	检查定义的网络没有重叠。
networks-same-address-families	阻塞	检查定义的网络是否共享相同的地址系列 (有效地址系列为 IPv4、IPv6)
network-prefix-valid	阻塞	检查集群网络前缀，以确保其有效，并为所有主机有足够的地址空间。
machine-cidr-equals-to-calculated-cidr	阻塞	对于非用户管理的网络集群。检查 **apiVIPs** 或 **ingressVIPs** 是否是机器 CIDR 的成员（如果存在）。
api-vips-defined	非阻塞	对于非用户管理的网络集群。检查 **apiVIPs** 是否存在。
api-vips-valid	阻塞	对于非用户管理的网络集群。检查 **apiVIPs** 是否属于机器 CIDR，且没有使用。
ingress-vips-defined	阻塞	对于非用户管理的网络集群。检查 **ingressVIPs** 是否存在。
ingress-vips-valid	非阻塞	对于非用户管理的网络集群。检查 **ingressVIPs** 是否属于机器 CIDR，且没有使用。
all-hosts-are-ready-to-install	阻塞	检查集群中的所有主机是否处于 "ready to install" 状态。
sufficient-masters-count	阻塞	此验证只适用于多节点集群。 ● 集群必须有三个 master。 ● 如果集群有 worker 节点，则至少必须存在 2 个 worker 节点。
dns-domain-defined	非阻塞	检查集群是否存在基本 DNS 域。
pull-secret-set	非阻塞	检查 pull secret 存在。不会检查 pull secret 是否有效或被授权。
ntp-server-configured	阻塞	检查每个主机的时钟与其他主机的时钟没有进行同步的时间不超过 4 分钟。
lso-requirements-satisfied	阻塞	验证集群是否满足 Local Storage Operator 的要求。

参数	验证类型	描述
odf-requirements-satisfied	阻塞	验证集群是否满足 Openshift Data Foundation Operator 的要求。 ● 集群至少有 3 个主机。 ● 集群只有 3 个 master 或至少 3 个 worker。 ● 集群有 3 个有资格的磁盘，每个主机都必须有合格的磁盘。
cnv-requirements-satisfied	阻塞	验证集群是否满足容器原生虚拟化的要求。 ● 集群的 CPU 架构是 x86
lvm-requirements-satisfied	阻塞	验证集群是否满足逻辑卷管理器 Operator 的要求。 ● 集群必须是单一节点。 ● 集群必须运行 Openshift >= 4.11.0。
network-type-valid	阻塞	检查网络类型的有效性（如果存在）。 ● 网络类型必须是 OpenshiftSDN 或 OVNKubernetes。 ● OpenShiftSDN 不支持 IPv6 或单节点 Openshift。OpenShift Container Platform 4.15 及更新版本不支持 OpenShiftSDN。 ● OVNKubernetes 不支持 VIP DHCP 分配。

第 11 章 网络配置

这部分论述了使用 Assisted Installer 进行网络配置的基础知识。

11.1. 集群网络

OpenShift 使用各种网络类型和地址，并在下表中列出。

类型	DNS	描述
clusterNetwork		从中分配 Pod IP 地址的 IP 地址池。
serviceNetwork		服务的 IP 地址池。
machineNetwork		组成集群的机器的 IP 地址块。
apiVIP	**api.<clustername.clusterdomain>**	用于 API 通信的 VIP。此设置必须在 DNS 中提供或预先配置，以便默认名称可以正确解析。如果要使用双栈网络部署，则必须是 IPv4 地址。
apiVIPs	**api.<clustername.clusterdomain>**	用于 API 通信的 VIP。此设置必须在 DNS 中提供或预先配置，以便默认名称可以正确解析。如果使用双栈网络，则第一个地址必须是 IPv4 地址，第二个地址必须是 IPv6 地址。您还必须设置 **apiVIP** 设置。
ingressVIP	***.apps.<clustername.clusterdomain>**	用于入口流量的 VIP。如果要使用双栈网络部署，则必须是 IPv4 地址。
ingressVIPs	***.apps.<clustername.clusterdomain>**	用于入口流量的 VIP。如果您使用双栈网络部署，则第一个地址必须是 IPv4 地址，第二个地址必须是 IPv6 地址。您还必须设置 **ingressVIP** 设置。

注意

OpenShift Container Platform 4.12 引入了新的 apiVIPs 和 ingressVIPs 设置，用于接受双栈网络的多个 IP 地址。使用双栈网络时，第一个 IP 地址必须是 IPv4 地址，第二个 IP 地址必须是 IPv6 地址。新设置将替换 apiVIP 和 IngressVIP，但在使用 API 修改配置时，您必须同时设置新的和旧设置。

根据所需的网络堆栈，您可以选择不同的网络接口控制器。目前，辅助服务可以使用以下配置之一部署 OpenShift Container Platform 集群：

- IPv4
- 双栈 (IPv4 + IPv6)

支持的网络接口控制器取决于所选的堆栈，并在下表中总结。如需详细的 Container Network Interface (CNI)网络供应商功能比较，请参阅 OCP 网络文档。

堆栈	SDN	OVN
IPv4	是	是
dual-stack	否	是

注意

OVN 是 OpenShift Container Platform 4.12 及更新的版本中的默认 Container Network Interface (CNI)。SDN 最多支持 OpenShift Container Platform 4.14，但不支持 OpenShift Container Platform 4.15 及更新版本。

11.1.1. 限制

11.1.1.1. SDN

- 单节点 OpenShift 不支持 SDN 控制器。
- SDN 控制器不支持 IPv6。
- OpenShift Container Platform 4.15 及更新版本不支持 SDN 控制器。如需更多信息，请参阅 OpenShift Container Platform 发行注记中的 OpenShift SDN 网络插件。

11.1.1.2. OVN-Kubernetes

请参阅 OCP 文档中的 OVN-Kubernetes 限制部分。

11.1.2. 集群网络

集群网络是一个网络，集群中部署的每个 Pod 都从中获取其 IP 地址。如果工作负载可以在组成集群的多个节点间存在，因此网络供应商可以根据 Pod 的 IP 地址轻松查找单个节点。为此，clusterNetwork.cidr 被进一步分成 clusterNetwork.hostPrefix 中定义的大小的子网。

主机前缀指定分配给集群中每个节点的子网的长度。集群如何为多节点集群分配地址的示例：

```
---
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
---
```

使用上述代码片段创建 3 节点集群可以创建以下网络拓扑：

- 在节点 #1 中调度的 Pod 从 10.128.0.0/23 获取 IP
- 在节点 #2 中调度的 Pod 从 10.128.2.0/23 获取 IP
- 在节点 #3 中调度的 Pod 从 10.128.4.0/23 获取 IP

解释 OVN-K8s 超出了本文档的范围，但上述的模式提供了在不同节点之间路由 Pod 到Pod 流量的方法，而无需保留 Pod 和对应节点之间的大量映射列表。

11.1.3. 机器网络

机器网络是由组成集群用来相互通信的所有主机使用的网络。这也是必须包括 API 和 Ingress VIP 的子网。

11.1.4. 与多节点集群相比的 SNO

取决于您要部署单节点 OpenShift 还是多节点集群，需要使用不同的值。下表更详细地说明了这一点。

参数	SNO	使用 DHCP 模式的多节点集群	没有 DHCP 模式的多节点集群
clusterNetwork	必填	必填	必填
serviceNetwork	必填	必填	必填
machineNetwork	可以自动分配 (*)	可以自动分配 (*)	可以自动分配 (*)
apiVIP	禁止	禁止	必填
apiVIPs	禁止	禁止	4.12 及更新的版本需要
ingressVIP	禁止	禁止	必填
ingressVIPs	禁止	禁止	4.12 及更新的版本需要

如果只有一个主机网络，则机器网络 CIDR 的自动分配。否则，您需要明确指定它。

11.1.5. air-gapped 环境

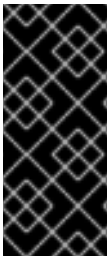
在没有互联网访问的情况下部署集群的工作流有一些超出本文档范围的先决条件。您可以参阅 Zero Touch Provisioning the hard way Git repository 以了解更多信息。

11.2. VIP DHCP 分配

VIP DHCP 分配是允许用户跳过为 API 和 Ingress 手动提供虚拟 IP 和 Ingress 的要求，利用服务从 DHCP 服务器自动分配这些 IP 地址。

如果您启用这个功能，而不是使用来自集群配置的 api_vips 和 ingress_vips，服务会发送租期分配请求，并根据回复相应地使用 VIP。该服务将从 Machine Network 分配 IP 地址。

请注意，这不是 OpenShift Container Platform 功能，它已在辅助服务中实施，以简化配置。



重要

VIP DHCP 分配目前仅限于 OpenShift Container Platform SDN 网络类型。OpenShift Container Platform 版本 4.15 及更新版本不支持 SDN。因此，对 VIP DHCP 分配的支持也会从 OpenShift Container Platform 4.15 及之后版本结束。

11.2.1. 启用自动分配的有效负载示例

```
---
{
  "vip_dhcp_allocation": true,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    }
  ],
  "service_networks": [
    {
      "cidr": "172.30.0.0/16"
    }
  ],
  "machine_networks": [
    {
      "cidr": "192.168.127.0/24"
    }
  ]
}
---
```

11.2.2. 禁用自动分配的有效负载示例

```
---
{
  "api_vips": [
    {
       "ip": "192.168.127.100"
    }
  ],
  "ingress_vips": [
    {
       "ip": "192.168.127.101"
    }
  ],
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
```

```
    "host_prefix": 23
   }
  ],
  "service_networks": [
   {
    "cidr": "172.30.0.0/16"
   }
  ]
}
---
```

11.3. 其他资源

- 裸机 IPI 文档 提供了 VIP 地址语法的额外说明。

11.4. 了解用户管理和集群管理的网络之间的区别

用户管理的网络是 Assisted Installer 中的一项功能，它允许具有非标准网络拓扑的客户部署 OpenShift Container Platform 集群。示例包括：

- 具有不需要使用 keepalived 和 VRRP 处理 VIP 地址的外部负载均衡器的客户。
- 使用在许多不同 L2 网络片段间分布的集群节点部署。

11.4.1. 验证

在允许安装启动前，辅助安装程序中会发生各种网络验证。当您启用用户管理的网络时，以下验证更改：

- L3 连接检查 (ICMP) 被执行，而不是 L2 检查 (ARP)

11.5. 静态网络配置

您可以在生成或更新发现 ISO 时使用静态网络配置。

11.5.1. 先决条件

- 熟悉 NMState。

11.5.2. NMState 配置

YAML 格式的 NMState 文件指定主机所需的网络配置。它具有在发现时将替换为接口的实际名称的接口的逻辑名称。

11.5.2.1. NMState 配置示例

```
---
dns-resolver:
  config:
    server:
    - 192.168.126.1
interfaces:
- ipv4:
    address:
    - ip: 192.168.126.30
      prefix-length: 24
    dhcp: false
    enabled: true
  name: eth0
  state: up
  type: ethernet
- ipv4:
    address:
    - ip: 192.168.141.30
      prefix-length: 24
    dhcp: false
    enabled: true
  name: eth1
  state: up
  type: ethernet
routes:
  config:
  - destination: 0.0.0.0/0
    next-hop-address: 192.168.126.1
    next-hop-interface: eth0
    table-id: 254
---
```

11.5.3. MAC 接口映射

MAC 接口映射是一个属性，它使用主机上的实际接口映射 NMState 配置中定义的逻辑接口。

映射应始终使用主机上的物理接口。例如，当 NMState 配置定义了绑定或 VLAN 时，映射应该只包含父接口的条目。

11.5.3.1. MAC 接口映射示例

```
---
mac_interface_map: [
  {
    mac_address: 02:00:00:2c:23:a5,
    logical_nic_name: eth0
  },
  {
    mac_address: 02:00:00:68:73:dc,
    logical_nic_name: eth1
  }
]
---
```

11.5.4. 额外的 NMState 配置示例

以下示例仅用于显示部分配置。您不应该原样使用它们，而是根据您的具体环境对它们进行相应的调整。如果使用错误，可能会导致您的机器没有网络连接。

11.5.4.1. 标记的 VLAN

```
---
  interfaces:
  - ipv4:
      address:
      - ip: 192.168.143.15
        prefix-length: 24
      dhcp: false
      enabled: true
    ipv6:
      enabled: false
    name: eth0.404
    state: up
    type: vlan
    vlan:
      base-iface: eth0
      id: 404
      reorder-headers: true
---
```

11.5.4.2. 网络绑定

```
---
  interfaces:
  - ipv4:
      address:
      - ip: 192.168.138.15
        prefix-length: 24
```

```
      dhcp: false
      enabled: true
    ipv6:
      enabled: false
    link-aggregation:
      mode: active-backup
      options:
        all_slaves_active: delivered
        miimon: "140"
      slaves:
      - eth0
      - eth1
    name: bond0
    state: up
    type: bond
---
```

11.6. 使用 API 应用静态网络配置

您可以使用 Assisted Installer API 应用静态网络配置。

先决条件

1. 您已使用 API 创建基础架构环境，或者已使用 Web 控制台创建集群。
2. 在 shell 中将您的基础架构环境 ID 导出为 $INFRA_ENV_ID。
3. 您在访问 API 时具有凭证，并在 shell 中将令牌导出为 $API_TOKEN。
4. 您有带有静态网络配置的 YAML 文件，作为 server-a.yaml 和 server-b.yaml。

流程

1. 使用 API 请求创建一个临时文件 /tmp/request-body.txt ：

```
---
jq -n --arg NMSTATE_YAML1 "$(cat server-a.yaml)" --arg NMSTATE_YAML2 "$(cat
server-b.yaml)" \
'{
  "static_network_config": [
    {
      "network_yaml": $NMSTATE_YAML1,
      "mac_interface_map": [{"mac_address": "02:00:00:2c:23:a5", "logical_nic_name":
```

```
"eth0"}, {"mac_address": "02:00:00:68:73:dc", "logical_nic_name": "eth1"}]
    },
    {
     "network_yaml": $NMSTATE_YAML2,
     "mac_interface_map": [{"mac_address": "02:00:00:9f:85:eb", "logical_nic_name":
"eth1"}, {"mac_address": "02:00:00:c8:be:9b", "logical_nic_name": "eth0"}]
    }
 ]
}' >> /tmp/request-body.txt
---
```

2. 刷新 API 令牌：

```
$ source refresh-token
```

3. 将请求发送到 Assisted Service API 端点：

```
---
$ curl -H "Content-Type: application/json" \
-X PATCH -d @/tmp/request-body.txt \
-H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID
---
```

11.7. 其他资源

- 使用 Web 控制台应用静态网络配置

11.8. 转换为双栈网络

双栈 IPv4/IPv6 配置允许部署驻留在 IPv4 和 IPv6 子网中的 pod 的集群。

11.8.1. 先决条件

- 熟悉 OVN-K8s 文档

11.8.2. 单一节点 OpenShift 的有效负载示例

```
---
{
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
```

```
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
  ],
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.3. 由多个节点组成的 OpenShift Container Platform 集群的有效负载示例

```
---
{
  "vip_dhcp_allocation": false,
  "network_type": "OVNKubernetes",
  "user_managed_networking": false,
  "api_vips": [
    {
      "ip": "192.168.127.100"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
    }
  ],
  "ingress_vips": [
    {
      "ip": "192.168.127.101"
    },
    {
      "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7335"
    }
  ],
  "cluster_networks": [
    {
      "cidr": "10.128.0.0/14",
      "host_prefix": 23
    },
    {
      "cidr": "fd01::/48",
      "host_prefix": 64
    }
  ],
  "service_networks": [
    {"cidr": "172.30.0.0/16"}, {"cidr": "fd02::/112"}
  ],
```

```
  "machine_networks": [
    {"cidr": "192.168.127.0/24"},{"cidr": "1001:db8::/120"}
  ]
}
---
```

11.8.4. 限制

在使用双栈网络时，api_vips IP 地址和 ingress_vips IP 地址设置必须是主 IP 地址系列的设置，其必须是 IPv4 地址。目前，红帽不支持将 IPv6 作为主要 IP 地址系列的双栈 VIP 或双栈网络。红帽支持双栈网络，并将 IPv4 作为主要 IP 地址系列，以及 IPv6 作为辅助 IP 地址系列。因此，在输入 IP 地址值时，您必须在 IPv6 条目前面放置 IPv4 条目。

11.9. 其他资源

- 了解 OpenShift 网络
- OpenShift SDN - CNI 网络供应商
- OVN-Kubernetes - CNI 网络供应商
- 双栈服务配置场景
- 在裸机 OCP 上安装。
- Cluster Network Operator 配置。

第 12 章 扩展集群

您可以使用用户界面或 API 添加主机来扩展使用 Assisted Installer 安装的集群。

其他资源

- 将节点添加到集群时的 API 连接失败
- 在 OpenShift 集群中配置多架构计算机器

12.1. 检查多架构支持

在添加具有不同架构的节点前，您必须检查集群是否可以支持多个架构。

流程

1. 使用 CLI 登录集群。
2. 运行以下命令，检查集群是否使用构架有效负载：

```
$ oc adm release info -o json | jq .metadata.metadata
```

验证

- 如果您看到以下输出，代表您的集群支持多种架构：

```
{
  "release.openshift.io/architecture": "multi"
}
```

12.2. 安装多架构集群

具有 x86_64 control plane 的集群可以支持有两个不同 CPU 架构的 worker 节点。混合架构集群结合了每个架构的优势，并支持各种工作负载。

例如，您可以在带有 x86_64 节点的现有的 OpenShift Container Platform 集群中添加 arm64, IBM

Power, 或 IBM zSystems worker 节点。

安装的主要步骤如下：

1. 创建并注册多架构集群。
2. 创建 x86_64 基础架构环境，下载 x86_64 的 ISO 发现镜像，并添加 control plane。控制平面必须具有 x86_64 架构。
3. 创建 arm64, IBM Power, 或 IBM zSystems 基础架构环境，下载 arm64, IBM Power 或 IBM zSystems 的 ISO 发现镜像，并添加 worker 节点。

支持的平台

下表列出了支持每个 OpenShift Container Platform 版本的混合架构集群的平台。对您安装的版本使用合适的平台。

OpenShift Container Platform 版本	支持的平台	第 1 天控制平面架构	第 2 天节点架构
4.12.0	• Microsoft Azure (TP)	• **x86_64**	• **arm64**
4.13.0	• Microsoft Azure • Amazon Web Services • 裸机(TP)	• **x86_64** • **x86_64** • **x86_64**	• **arm64** • **arm64** • **arm64**

OpenShift Container Platform 版本	支持的平台	第 1 天控制平面架构	第 2 天节点架构
4.14.0	• Microsoft Azure • Amazon Web Services • 裸机 • Google Cloud Platform • IBM® Power® • IBM Z®	• **x86_64** • **x86_64** • **x86_64** • **x86_64** • **x86_64** • **x86_64**	• **arm64** • **arm64** • **arm64** • **arm64** • **ppc64le** • **s390x**

重要

红帽产品服务等级协议(SLA)不支持技术预览(TP)功能，可能无法正常工作。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。

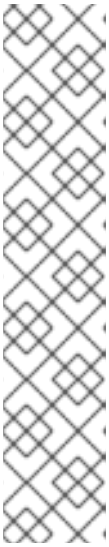
主要步骤

1. 启动使用 API 安装 OpenShift Container Platform 的流程。详情请参阅 *其它资源* 部分中的 *使用辅助安装程序 API 安装*。

2. 当您到达安装的 "Registering a new cluster" 步骤时，将集群注册为 multi-architecture 集群：

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "<version-number>-multi", (1)
  "cpu_architecture" : "multi" (2)
  "high_availability_mode": "full" (3)
  "base_dns_domain": "example.com",
```

```
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```



注意

1. 对 OpenShift Container Platform 版本号使用 multi- 选项，例如，"4.12-multi" 。
2. 将 'CPU architecture'设置为 "multi "。
3. 使用 full 值来指示多节点 OpenShift Container Platform。

3. 当您到达安装的"Registering a new infrastructure environment"步骤时，将 cpu_architecture 设置为 x86_64 ：

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-envs \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
 --slurpfile pull_secret ~/Downloads/pull-secret.txt \
 --arg cluster_id ${CLUSTER_ID} '
  {
    "name": "testcluster-infra-env",
    "image_type":"full-iso",
    "cluster_id": $cluster_id,
    "cpu_architecture" : "x86_64"
    "pull_secret": $pull_secret[0] | tojson
  }
')" | jq '.id'
```

4. 当您到达安装的"Adding hosts"步骤时，将 host_role 设置为 master ：

注意

如需更多信息，请参阅 *其它资源* 中的 *将角色分配给主机*。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
```

```
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
  {
    "host_role":"master"
  }
' | jq
```

5. 下载 x86_64 架构的发现镜像。

6. 使用生成的发现镜像引导 x86_64 架构主机。

7. 开始安装，并等待集群完全安装好。

8. 重复安装的"Registering a new infrastructure environment"步骤。这次，将 cpu_architecture 设置为以下之一：ppc64le（用于 IBM Power）、s390x（用于 IBM Z）或 arm64。例如：

```
$ curl -s -X POST https://api.openshift.com/api/assisted-install/v2/clusters \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d "$(jq --null-input \
  --slurpfile pull_secret ~/Downloads/pull-secret.txt '
{
  "name": "testcluster",
  "openshift_version": "4.12",
  "cpu_architecture" : "arm64"
  "high_availability_mode": "full"
  "base_dns_domain": "example.com",
  "pull_secret": $pull_secret[0] | tojson
}
')" | jq '.id'
```

9. 重复安装的"Adding hosts"步骤。这次，将 host_role 设置为 worker ：

注意

如需了解更多详细信息，请参阅 *其它资源* 中的 *将角色分配给主机*。

```
$ curl https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/hosts/<host_id> \
-X PATCH \
```

```
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
  {
    "host_role":"worker"
  }
' | jq
```

10. 下载 arm64 、 ppc64 或 s390x 架构的发现镜像。

11. 使用生成的发现镜像引导架构主机。

12. 开始安装，并等待集群完全安装好。

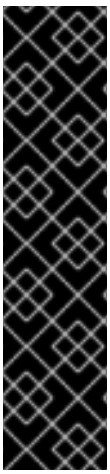
验证

- 运行以下命令，查看集群中的 arm64、ppc64le 或 s390x worker 节点：

```
$ oc get nodes -o wide
```

12.3. 使用 WEB 控制台添加主机

您可以将主机添加到使用 Assisted Installer 创建的集群。



重要

- 只有运行 OpenShift Container Platform 版本 4.11 及之后的版本的集群才支持将主机添加到 Assisted Installer 集群。

- 在第 2 天操作中添加 control plane 节点时，请确保新节点共享与第 1 天网络相同的子网。子网在 install-config.yaml 文件的 machineNetwork 字段中指定。这个要求适用于集群管理的网络，如裸机或 vSphere，不适用于用户管理的网络。

流程

1. 登录 OpenShift Cluster Manager，再点您要扩展的集群。

2. 点 Add hosts 并下载新主机的发现 ISO，添加 SSH 公钥并根据需要配置集群范围的代理设置。

3. 可选：根据需要修改 ignition 文件。

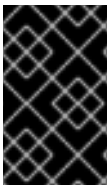
4. 使用发现 ISO 引导目标主机，并等待在控制台中发现主机。

5. 选择主机角色。它可以是 worker 或 control plane 主机。

6. 开始安装。

7. 当安装继续进行时，安装会为主机生成待处理的证书签名请求 (CSR)。出现提示时，批准待处理的 CSR 以完成安装。

   主机成功安装后，它在集群 Web 控制台中列为主机。



重要

新的主机将使用与原始集群相同的方法进行加密。

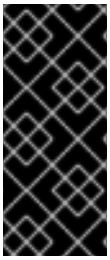
12.4. 使用 API 添加主机

您可以使用 Assisted Installer REST API 将主机添加到集群。

先决条件

- 安装 Red Hat OpenShift Cluster Manager CLI (ocm)。
- 以具有集群创建权限的用户身份登录 OpenShift Cluster Manager。
- 安装 jq。

- 确保您要扩展的集群存在所有必需的 DNS 记录。



重要

在第 2 天操作中添加 control plane 节点时，请确保新节点共享与第 1 天网络相同的子网。子网在 install-config.yaml 文件的 machineNetwork 字段中指定。这个要求适用于集群管理的网络，如裸机或 vSphere，不适用于用户管理的网络。

流程

1. 针对 Assisted Installer REST API 进行身份验证，并为会话生成 API 令牌。生成的令牌有效期仅为 15 分钟。

2. 运行以下命令设置 $API_URL 变量：

```
$ export API_URL=<api_url> 1
```

1 将 <api_url> 替换为 Assisted Installer API URL，例如 https://api.openshift.com

3. 运行以下命令导入集群：

   a. 设置 $CLUSTER_ID 变量。登录到集群并运行以下命令：

```
$ export CLUSTER_ID=$(oc get clusterversion -o
jsonpath='{.items[].spec.clusterID}')
```

   b. 设置用于导入集群的 $CLUSTER_REQUEST 变量：

```
$ export CLUSTER_REQUEST=$(jq --null-input --arg openshift_cluster_id
"$CLUSTER_ID" '{
  "api_vip_dnsname": "<api_vip>", 1
  "openshift_cluster_id": $CLUSTER_ID,
  "name": "<openshift_cluster_name>" 2
}')
```

1

将 <api_vip> 替换为集群 API 服务器的主机名。这可以是 API 服务器的 DNS 域，也可以是主机可访问的单一节点的 IP 地址。例如： api.compute-1.example.com。

2

将 <openshift_cluster_name> 替换为集群的纯文本名称。集群名称应与在第 1 天集群安装过程中设置的集群名称匹配。

c. 导入集群并设置 $CLUSTER_ID 变量。运行以下命令：

```
$ CLUSTER_ID=$(curl "$API_URL/api/assisted-install/v2/clusters/import" -H
"Authorization: Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-
Type: application/json' \
  -d "$CLUSTER_REQUEST" | tee /dev/stderr | jq -r '.id')
```

4. 运行以下命令，为集群生成 InfraEnv 资源并设置 $INFRA_ENV_ID 变量：

a. 从位于 console.redhat.com 的 Red Hat OpenShift Cluster Manager 下载 pull secret 文件。

b. 设置 $INFRA_ENV_REQUEST 变量：

```
export INFRA_ENV_REQUEST=$(jq --null-input \
    --slurpfile pull_secret <path_to_pull_secret_file> \ 1
    --arg ssh_pub_key "$(cat <path_to_ssh_pub_key>)" \ 2
    --arg cluster_id "$CLUSTER_ID" '{
  "name": "<infraenv_name>", 3
  "pull_secret": $pull_secret[0] | tojson,
  "cluster_id": $cluster_id,
  "ssh_authorized_key": $ssh_pub_key,
  "image_type": "<iso_image_type>" 4
}')
```

1

将 <path_to_pull_secret_file > 替换为在 console.redhat.com 上从 Red Hat OpenShift Cluster Manager 下载的 pull secret 的本地文件的路径。

2

将 <path_to_ssh_pub_key> 替换为访问主机所需的公共 SSH 密钥的路径。如果没有设置这个值，则无法在发现模式下访问主机。

3

4

将 <iso_image_type> 替换为 ISO 镜像类型，可以是 full-iso 或 minimal-iso。

c. 将 $INFRA_ENV_REQUEST 发布到 /v2/infra-envs API，并设置 $INFRA_ENV_ID 变量：

```
$ INFRA_ENV_ID=$(curl "$API_URL/api/assisted-install/v2/infra-envs" -H
"Authorization: Bearer ${API_TOKEN}" -H 'accept: application/json' -H 'Content-
Type: application/json' -d "$INFRA_ENV_REQUEST" | tee /dev/stderr | jq -r '.id')
```

5. 运行以下命令，获取集群主机的发现 ISO 的 URL：

```
$ curl -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID" -H
"Authorization: Bearer ${API_TOKEN}" | jq -r '.download_url'
```

输出示例

```
https://api.openshift.com/api/assisted-images/images/41b91e72-c33e-42ee-b80f-
b5c5bbf6431a?
arch=x86_64&image_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2NTY
wMjYzNzEsInN1YiI6IjQxYjkxZTcyLWMzM2UtNDJlZS1iODBmLWI1YzViYmY2NDMxYSJ9.
1EX_VGaMNejMhrAvVRBS7PDPIQtbOOc8LtG8OukE1a4&type=minimal-
iso&version=4.12
```

6. 下载 ISO：

```
$ curl -L -s '<iso_url>' --output rhcos-live-minimal.iso 1
```

1

将 <iso_url> 替换为上一步中的 ISO URL。

7. 从下载的 rhcos-live-minimal.iso 中引导新的 worker 主机。

8. 获取*没有*安装的集群中的主机列表。继续运行以下命令，直到新主机显示：

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization:
Bearer ${API_TOKEN}" | jq -r '.hosts[] | select(.status != "installed").id'
```

输出示例

```
2294ba03-c264-4f11-ac08-2f1bb2f8c296
```

9. 为新主机设置 $HOST_ID 变量，例如：

```
$ HOST_ID=<host_id> 1
```

1 将 <host_id> 替换为上一步中的主机 ID。

10. 运行以下命令检查主机是否已就绪：

注意

确保复制整个命令，包括完整的 jq 表达式。

```
$ curl -s $API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID -H "Authorization:
Bearer ${API_TOKEN}" | jq '
def host_name($host):
  if (.suggested_hostname // "") == "" then
    if (.inventory // "") == "" then
      "Unknown hostname, please wait"
    else
      .inventory | fromjson | .hostname
    end
  else
```

```
        .suggested_hostname
    end;

def is_notable($validation):
    ["failure", "pending", "error"] | any(. == $validation.status);

def notable_validations($validations_info):
    [
        $validations_info // "{}"
        | fromjson
        | to_entries[].value[]
        | select(is_notable(.))
    ];

{
    "Hosts validations": {
        "Hosts": [
            .hosts[]
            | select(.status != "installed")
            | {
                "id": .id,
                "name": host_name(.),
                "status": .status,
                "notable_validations": notable_validations(.validations_info)
            }
        ]
    },
    "Cluster validations info": {
        "notable_validations": notable_validations(.validations_info)
    }
}
' -r
```

输出示例

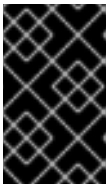
```
{
  "Hosts validations": {
    "Hosts": [
      {
        "id": "97ec378c-3568-460c-bc22-df54534ff08f",
        "name": "localhost.localdomain",
        "status": "insufficient",
        "notable_validations": [
          {
            "id": "ntp-synced",
            "status": "failure",
            "message": "Host couldn't synchronize with any NTP server"
          },
          {
            "id": "api-domain-name-resolved-correctly",
            "status": "error",
            "message": "Parse error for domain name resolutions result"
          },
```

```
        {
          "id": "api-int-domain-name-resolved-correctly",
          "status": "error",
          "message": "Parse error for domain name resolutions result"
        },
        {
          "id": "apps-domain-name-resolved-correctly",
          "status": "error",
          "message": "Parse error for domain name resolutions result"
        }
      ]
    }
  ]
 },
 "Cluster validations info": {
  "notable_validations": []
 }
}
```

11. 当上一个命令显示主机就绪时，通过运行以下命令来使用 /v2/infra-envs/{infra_env_id}/hosts/{host_id}/actions/install API 开始安装：

```
$ curl -X POST -s "$API_URL/api/assisted-install/v2/infra-envs/$INFRA_ENV_ID/hosts/$HOST_ID/actions/install"  -H "Authorization: Bearer ${API_TOKEN}"
```

12. 当安装继续进行时，安装会为主机生成待处理的证书签名请求 (CSR)。



重要

您必须批准 CSR 才能完成安装。

运行以下 API 调用以监控集群安装：

```
$ curl -s "$API_URL/api/assisted-install/v2/clusters/$CLUSTER_ID" -H "Authorization: Bearer ${API_TOKEN}" | jq '{
  "Cluster day-2 hosts":
    [
      .hosts[]
      | select(.status != "installed")
      | {id, requested_hostname, status, status_info, progress, status_updated_at,
```

```
updated_at, infra_env_id, cluster_id, created_at}
      ]
}'
```

输出示例

```
{
  "Cluster day-2 hosts": [
    {
      "id": "a1c52dde-3432-4f59-b2ae-0a530c851480",
      "requested_hostname": "control-plane-1",
      "status": "added-to-existing-cluster",
      "status_info": "Host has rebooted and no further updates will be posted. Please
check console for progress and to possibly approve pending CSRs",
      "progress": {
        "current_stage": "Done",
        "installation_percentage": 100,
        "stage_started_at": "2022-07-08T10:56:20.476Z",
        "stage_updated_at": "2022-07-08T10:56:20.476Z"
      },
      "status_updated_at": "2022-07-08T10:56:20.476Z",
      "updated_at": "2022-07-08T10:57:15.306369Z",
      "infra_env_id": "b74ec0c3-d5b5-4717-a866-5b6854791bd3",
      "cluster_id": "8f721322-419d-4eed-aa5b-61b50ea586ae",
      "created_at": "2022-07-06T22:54:57.161614Z"
    }
  ]
}
```

13. 可选：运行以下命令以查看集群的所有事件：

```
$ curl -s "$API_URL/api/assisted-install/v2/events?cluster_id=$CLUSTER_ID" -H
"Authorization: Bearer ${API_TOKEN}" | jq -c '.[] | {severity, message, event_time,
host_id}'
```

输出示例

```
{"severity":"info","message":"Host compute-0: updated status from insufficient to
known (Host is ready to be installed)","event_time":"2022-07-
08T11:21:46.346Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from known to
installing (Installation is in progress)","event_time":"2022-07-
08T11:28:28.647Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
```

```
{"severity":"info","message":"Host compute-0: updated status from installing to
installing-in-progress (Starting installation)","event_time":"2022-07-
08T11:28:52.068Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Uploaded logs for host compute-0 cluster 8f721322-
419d-4eed-aa5b-61b50ea586ae","event_time":"2022-07-
08T11:29:47.802Z","host_id":"9d7b3b44-1125-4ad0-9b14-76550087b445"}
{"severity":"info","message":"Host compute-0: updated status from installing-in-
progress to added-to-existing-cluster (Host has rebooted and no further updates will
be posted. Please check console for progress and to possibly approve pending
CSRs)","event_time":"2022-07-08T11:29:48.259Z","host_id":"9d7b3b44-1125-4ad0-
9b14-76550087b445"}
{"severity":"info","message":"Host: compute-0, reached installation stage
Rebooting","event_time":"2022-07-08T11:29:48.261Z","host_id":"9d7b3b44-1125-4ad0-
9b14-76550087b445"}
```

14. 登录到集群并批准待处理的 CSR 以完成安装。

验证

- 检查新主机是否已成功添加到集群中，状态为 Ready ：

```
$ oc get nodes
```

输出示例

```
NAME                        STATUS   ROLES          AGE   VERSION
control-plane-1.example.com   Ready    master,worker   56m   v1.25.0
compute-1.example.com         Ready    worker         11m   v1.25.0
```

12.5. 在一个健康的集群中安装主 CONTROL PLANE 节点

此流程描述了如何在健康的 OpenShift Container Platform 集群上安装主 control plane 节点。

如果集群不健康，则在管理前需要额外的操作。如需更多信息，请参阅 *其它资源*。

先决条件

- 您使用带有正确的 etcd-operator 版本的 OpenShift Container Platform 4.11 或更高版本。
- 您已安装了一个具有至少三个节点的健康集群。
- 您已创建了单个 control plane 节点。

流程

1. 检索待处理的 CertificateSigningRequests (CSR)：

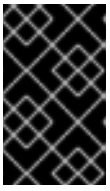
```
$ oc get csr | grep Pending
```

输出示例

```
csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
<none>              Pending
csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
<none>              Pending
```

2. 批准待处理的 CSR：

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```



重要

您必须批准 CSR 才能完成安装。

3. 确认主节点处于 Ready 状态：

```
$ oc get nodes
```

输出示例

```
NAME      STATUS   ROLES    AGE    VERSION
master-0  Ready    master   4h42m  v1.24.0+3882f8f
worker-1  Ready    worker   4h29m  v1.24.0+3882f8f
master-2  Ready    master   4h43m  v1.24.0+3882f8f
master-3  Ready    master   4h27m  v1.24.0+3882f8f
worker-4  Ready    worker   4h30m  v1.24.0+3882f8f
master-5  Ready    master   105s   v1.24.0+3882f8f
```

注意

当集群运行功能 Machine API 时，etcd-operator 需要引用新节点的 Machine Custom Resource (CR)。

4. 将 Machine CR 与 BareMetalHost 和 Node 链接：

   a. 使用具有唯一 .metadata.name 值的 BareMetalHost CR：

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: custom-master3
  namespace: openshift-machine-api
  annotations:
spec:
  automatedCleaningMode: metadata
  bootMACAddress: 00:00:00:00:00:02
  bootMode: UEFI
  customDeploy:
    method: install_coreos
  externallyProvisioned: true
  online: true
  userData:
    name: master-user-data-managed
    namespace: openshift-machine-api
```

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

b.

应用 BareMetalHost CR：

```
$ oc apply -f <filename>
```

c.

使用唯一的 .machine.name 值创建 Machine CR：

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  annotations:
    machine.openshift.io/instance-state: externally provisioned
    metal3.io/BareMetalHost: openshift-machine-api/custom-master3
  finalizers:
  - machine.machine.openshift.io
  generation: 3
  labels:
    machine.openshift.io/cluster-api-cluster: test-day2-1-6qv96
    machine.openshift.io/cluster-api-machine-role: master
    machine.openshift.io/cluster-api-machine-type: master
  name: custom-master3
  namespace: openshift-machine-api
spec:
  metadata: {}
  providerSpec:
    value:
      apiVersion: baremetal.cluster.k8s.io/v1alpha1
      customDeploy:
        method: install_coreos
      hostSelector: {}
      image:
        checksum: ""
        url: ""
      kind: BareMetalMachineProviderSpec
      metadata:
        creationTimestamp: null
      userData:
        name: master-user-data-managed
```

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

d.

应用 Machine CR：

```
$ oc apply -f <filename>
```

e.

使用 link-machine-and-node.sh 脚本链接 BareMetalHost, Machine, 和 Node：

```
#!/bin/bash

# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

# set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" ] || [ -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

# uid=$(echo "${node}" | cut -f1 -d':')
node_name=$(echo "${node}" | cut -f2 -d':')

oc proxy &
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespa
ces/openshift-machine-api/baremetalhosts"

function print_nics() {
    local ips
    local eob
    declare -a ips

    readarray -t ips < <(echo "${1}" \
                        | jq '.[] | select(. | .type == "InternalIP") | .address' \
                        | sed 's/"//g')

    eob=','
    for (( i=0; i<${#ips[@]}; i++ )); do
        if [ $((i+1)) -eq ${#ips[@]} ]; then
            eob=""
        fi
        cat <<- EOF
          {
            "ip": "${ips[$i]}",
            "mac": "00:00:00:00:00:00",
            "model": "unknown",
            "speedGbps": 10,
            "vlanId": 0,
            "pxe": true,
            "name": "eth1"
          }${eob}
EOF
    done
```

```
}

function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff

    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null >
/dev/null; do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$((curr_time - start_time))
        if [[ $time_diff -gt $timeout ]]; then
            printf '\nTimed out waiting for %s' "${name}"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept:
application/json" -H "Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api "${node_name}" -o json | jq -c
'.status.addresses')

machine_data=$(oc get machines.machine.openshift.io -n openshift-machine-api -
o json "${machine}")
host=$(echo "$machine_data" | jq
'.metadata.annotations["metal3.io/BareMetalHost"]' | cut -f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") |
.address' | sed 's/"//g')

set +e
```

```
read -r -d '' host_patch << EOF
{
  "status": {
    "hardware": {
      "hostname": "${hostname}",
      "nics": [
$(print_nics "${addresses}")
      ],
      "systemVendor": {
        "manufacturer": "Red Hat",
        "productName": "product name",
        "serialNumber": ""
      },
      "firmware": {
        "bios": {
          "date": "04/01/2014",
          "vendor": "SeaBIOS",
          "version": "1.11.0-2.el7"
        }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
        "arch": "x86_64",
        "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
        "clockMegahertz": 2199.998,
        "count": 4,
        "flags": []
      }
    }
  }
}
EOF
set -e

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    "${HOST_PROXY_API_PATH}/${host}/status" \
    -H "Content-type: application/merge-patch+json" \
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-5
```

5. 确认 etcd 成员：

```
$ oc rsh -n openshift-etcd etcd-worker-2
etcdctl member list -w table
```

输出示例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

6. 确认 etcd-operator 配置适用于所有节点：

```
$ oc get clusteroperator etcd
```

输出示例

```
NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE MESSAGE
etcd 4.11.5 True False False 5h54m
```

7. 确认 etcd-operator 健康状况：

```
$ oc rsh -n openshift-etcd etcd-worker-0
etcdctl endpoint health
```

输出示例

```
192.168.111.26 is healthy: committed proposal: took = 11.297561ms
192.168.111.25 is healthy: committed proposal: took = 13.892416ms
192.168.111.28 is healthy: committed proposal: took = 11.870755ms
```

8. **确认节点健康状况：**

```
$ oc get nodes
```

输出示例

```
NAME      STATUS   ROLES    AGE     VERSION
master-0  Ready    master   6h20m   v1.24.0+3882f8f
worker-1  Ready    worker   6h7m    v1.24.0+3882f8f
master-2  Ready    master   6h20m   v1.24.0+3882f8f
master-3  Ready    master   6h4m    v1.24.0+3882f8f
worker-4  Ready    worker   6h7m    v1.24.0+3882f8f
master-5  Ready    master   99m     v1.24.0+3882f8f
```

9. **确认 ClusterOperators 健康状况：**

```
$ oc get ClusterOperators
```

输出示例

```
NAME                                       VERSION AVAILABLE PROGRESSING DEGRADED SINCE
MSG
authentication                             4.11.5  True      False       False    5h57m
baremetal                                  4.11.5  True      False       False    6h19m
cloud-controller-manager                   4.11.5  True      False       False    6h20m
cloud-credential                           4.11.5  True      False       False    6h23m
cluster-autoscaler                         4.11.5  True      False       False    6h18m
config-operator                            4.11.5  True      False       False    6h19m
console                                    4.11.5  True      False       False    6h4m
csi-snapshot-controller                    4.11.5  True      False       False    6h19m
dns                                        4.11.5  True      False       False    6h18m
etcd                                       4.11.5  True      False       False    6h17m
image-registry                             4.11.5  True      False       False    6h7m
ingress                                    4.11.5  True      False       False    6h6m
insights                                   4.11.5  True      False       False    6h12m
kube-apiserver                             4.11.5  True      False       False    6h16m
kube-controller-manager                    4.11.5  True      False       False    6h16m
kube-scheduler                             4.11.5  True      False       False    6h16m
kube-storage-version-migrator              4.11.5  True      False       False    6h19m
machine-api                                4.11.5  True      False       False    6h15m
machine-approver                           4.11.5  True      False       False    6h19m
```

```
machine-config                      4.11.5  True      False      False    6h18m
marketplace                         4.11.5  True      False      False    6h18m
monitoring                          4.11.5  True      False      False    6h4m
network                             4.11.5  True      False      False    6h20m
node-tuning                         4.11.5  True      False      False    6h18m
openshift-apiserver                 4.11.5  True      False      False    6h8m
openshift-controller-manager        4.11.5  True      False      False    6h7m
openshift-samples                   4.11.5  True      False      False    6h12m
operator-lifecycle-manager          4.11.5  True      False      False    6h18m
operator-lifecycle-manager-catalog  4.11.5  True      False      False    6h19m
operator-lifecycle-manager-pkgsvr   4.11.5  True      False      False    6h12m
service-ca                          4.11.5  True      False      False    6h19m
storage                             4.11.5  True      False      False    6h19m
```

10. 确认 ClusterVersion ：

```
$ oc get ClusterVersion
```

输出示例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         5h57m   Cluster version is 4.11.5
```

11. 删除旧的 control plane 节点：

    a. 删除 BareMetalHost CR：

    ```
    $ oc delete bmh -n openshift-machine-api custom-master3
    ```

    b. 确认 Machine 不健康：

    ```
    $ oc get machine -A
    ```

    输出示例

```
NAMESPACE               NAME                         PHASE    AGE
openshift-machine-api  custom-master3                Running  14h
openshift-machine-api  test-day2-1-6qv96-master-0     Failed   20h
openshift-machine-api  test-day2-1-6qv96-master-1     Running  20h
openshift-machine-api  test-day2-1-6qv96-master-2     Running  20h
openshift-machine-api  test-day2-1-6qv96-worker-0-8w7vr  Running  19h
openshift-machine-api  test-day2-1-6qv96-worker-0-rxddj  Running  19h
```

c. 删除 Machine CR：

```
$ oc delete machine -n openshift-machine-api   test-day2-1-6qv96-master-0
machine.machine.openshift.io "test-day2-1-6qv96-master-0" deleted
```

d. 确认删除 Node CR：

```
$ oc get nodes
```

输出示例

```
NAME      STATUS   ROLES    AGE   VERSION
worker-1   Ready    worker   19h   v1.24.0+3882f8f
master-2   Ready    master   20h   v1.24.0+3882f8f
master-3   Ready    master   19h   v1.24.0+3882f8f
worker-4   Ready    worker   19h   v1.24.0+3882f8f
master-5   Ready    master   15h   v1.24.0+3882f8f
```

12. 检查 etcd-operator 日志以确认 etcd 集群的状态：

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
```

输出示例

■

```
E0927 07:53:10.597523       1 base_controller.go:272]
ClusterMemberRemovalController reconciliation failed: cannot remove member:
192.168.111.23 because it is reported as healthy but it doesn't have a machine nor a
node resource
```

13. 删除物理机器，以允许 etcd-operator 协调集群成员：

```
$ oc rsh -n openshift-etcd etcd-worker-2
etcdctl member list -w table; etcdctl endpoint health
```

输出示例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
192.168.111.26 is healthy: committed proposal: took = 10.458132ms
192.168.111.25 is healthy: committed proposal: took = 11.047349ms
192.168.111.28 is healthy: committed proposal: took = 11.414402ms
```

其他资源

- 在不健康集群中安装主 control plane 节点

12.6. 在不健康集群中安装主 CONTROL PLANE 节点

此流程描述了如何在不健康的 OpenShift Container Platform 集群上安装主 control plane 节点。

先决条件

- 您已安装了一个具有至少三个节点的健康集群。

- 您已创建了第 2 天 控制平面。
- 您已创建了单个 control plane 节点。

流程

1. 确认集群的初始状态：

```
$ oc get nodes
```

输出示例

```
NAME      STATUS     ROLES    AGE   VERSION
worker-1  Ready      worker   20h   v1.24.0+3882f8f
master-2  NotReady   master   20h   v1.24.0+3882f8f
master-3  Ready      master   20h   v1.24.0+3882f8f
worker-4  Ready      worker   20h   v1.24.0+3882f8f
master-5  Ready      master   15h   v1.24.0+3882f8f
```

2. 确认 etcd-operator 检测到集群不健康：

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf
```

输出示例

```
E0927 08:24:23.983733       1 base_controller.go:272] DefragController reconciliation
failed: cluster is unhealthy: 2 of 3 members are available, worker-2 is unhealthy
```

3. 确认 etcdctl 成员：

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

输出示例

```
+--------+---------+--------+-------------+-------------+---------+
|   ID   | STATUS  |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+-------------+-------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+-------------+-------------+---------+
```

4. 确认 etcdctl 报告集群的不健康成员：

```
$ etcdctl endpoint health
```

输出示例

```
{"level":"warn","ts":"2022-09-
27T08:25:35.953Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retryi
ng of unary invoker failed","target":"etcd-
endpoints://0xc000680380/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection
error: desc = \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route
to host\""}
192.168.111.28 is healthy: committed proposal: took = 12.465641ms
192.168.111.26 is healthy: committed proposal: took = 12.297059ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

5. 通过删除 Machine 自定义资源来删除不健康的 control plane：

```
$ oc delete machine -n openshift-machine-api test-day2-1-6qv96-master-2
```

注意

如果不健康的集群无法成功运行，则不会删除 Machine 和 Node 自定义资源 (CR)。

6. 确认 etcd-operator 没有删除不健康的机器：

```
$ oc logs -n openshift-etcd-operator etcd-operator-8668df65d-lvpjf -f
```

输出示例

```
I0927 08:58:41.249222       1 machinedeletionhooks.go:135] skip removing the deletion hook from machine test-day2-1-6qv96-master-2 since its member is still present with any of: [{InternalIP } {InternalIP 192.168.111.26}]
```

7. 手动删除不健康的 etcdctl 成员：

```
$ oc rsh -n openshift-etcd etcd-worker-3\
etcdctl member list -w table
```

输出示例

```
+--------+---------+--------+--------------+--------------+---------+
|   ID   | STATUS |  NAME  |  PEER ADDRS  | CLIENT ADDRS | LEARNER |
+--------+---------+--------+--------------+--------------+---------+
|2c18942f| started |worker-3|192.168.111.26|192.168.111.26|  false  |
|61e2a860| started |worker-2|192.168.111.25|192.168.111.25|  false  |
|ead4f280| started |worker-5|192.168.111.28|192.168.111.28|  false  |
+--------+---------+--------+--------------+--------------+---------+
```

8. 确认 etcdctl 报告集群的不健康成员：

```
$ etcdctl endpoint health
```

-

输出示例

```
{"level":"warn","ts":"2022-09-
27T10:31:07.227Z","logger":"client","caller":"v3/retry_interceptor.go:62","msg":"retryi
ng of unary invoker failed","target":"etcd-
endpoints://0xc0000d6e00/192.168.111.25","attempt":0,"error":"rpc error: code =
DeadlineExceeded desc = latest balancer error: last connection error: connection
error: desc = \"transport: Error while dialing dial tcp 192.168.111.25: connect: no route
to host\""}
192.168.111.28 is healthy: committed proposal: took = 13.038278ms
192.168.111.26 is healthy: committed proposal: took = 12.950355ms
192.168.111.25 is unhealthy: failed to commit proposal: context deadline exceeded
Error: unhealthy cluster
```

9. 通过删除 etcdctl 成员自定义资源来删除不健康的集群：

```
$ etcdctl member remove 61e2a86084aafa62
```

输出示例

```
Member 61e2a86084aafa62 removed from cluster 6881c977b97990d7
```

10. 运行以下命令确认 etcdctl 的成员：

```
$ etcdctl member list -w table
```

输出示例

```
+----------+---------+--------+-------------+-------------+-------+
|    ID    | STATUS  |  NAME  | PEER ADDRS  | CLIENT ADDRS |LEARNER|
+----------+---------+--------+-------------+-------------+-------+
```

```
| 2c18942f | started |worker-3|192.168.111.26|192.168.111.26| false |
| ead4f280 | started |worker-5|192.168.111.28|192.168.111.28| false |
+----------+---------+--------+--------------+--------------+-------+
```

11. **检查并批准证书签名请求**

    a. 查看证书签名请求 (CSR)：

    ```
    $ oc get csr | grep Pending
    ```

    输出示例

    ```
    csr-5sd59   8m19s   kubernetes.io/kube-apiserver-client-kubelet
    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper
    <none>              Pending
    csr-xzqts   10s     kubernetes.io/kubelet-serving                 system:node:worker-6
    <none>              Pending
    ```

    b. 批准所有待处理的 CSR：

    ```
    $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
    {{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
    ```

    注意

    您必须批准 CSR 才能完成安装。

12. **确认 control plane 节点就绪状态：**

    ```
    $ oc get nodes
    ```

    输出示例

```
NAME      STATUS   ROLES    AGE     VERSION
worker-1  Ready    worker   22h     v1.24.0+3882f8f
master-3  Ready    master   22h     v1.24.0+3882f8f
worker-4  Ready    worker   22h     v1.24.0+3882f8f
master-5  Ready    master   17h     v1.24.0+3882f8f
master-6  Ready    master   2m52s   v1.24.0+3882f8f
```

13. 验证 Machine, Node 和 BareMetalHost 自定义资源。

如果集群使用功能 Machine API 运行，etcd-operator 需要 Machine CR。存在时，Machine CR 会在 Running 阶段显示。

14. 创建与 BareMetalHost 和 Node 链接的 Machine 自定义资源。

确保有 Machine CR 引用新添加的节点。

重要

boot-it-yourself 将不会创建 BareMetalHost 和 Machine CR，因此您必须创建它们。如果无法创建 BareMetalHost 和 Machine CR，在运行 etcd-operator 时会生成错误。

15. 添加 BareMetalHost 自定义资源：

```
$ oc create bmh -n openshift-machine-api custom-master3
```

16. 添加 Machine 自定义资源：

```
$ oc create machine -n openshift-machine-api custom-master3
```

17. 运行 link-machine-and-node.sh 脚本链接 BareMetalHost, Machine, 和 Node：

```
#!/bin/bash
```

```
# Credit goes to https://bugzilla.redhat.com/show_bug.cgi?id=1801238.
# This script will link Machine object and Node object. This is needed
# in order to have IP address of the Node present in the status of the Machine.

# set -x
set -e

machine="$1"
node="$2"

if [ -z "$machine" ] || [ -z "$node" ]; then
    echo "Usage: $0 MACHINE NODE"
    exit 1
fi

# uid=$(echo "${node}" | cut -f1 -d':')
node_name=$(echo "${node}" | cut -f2 -d':')

oc proxy &
proxy_pid=$!
function kill_proxy {
    kill $proxy_pid
}
trap kill_proxy EXIT SIGINT

HOST_PROXY_API_PATH="http://localhost:8001/apis/metal3.io/v1alpha1/namespaces/
openshift-machine-api/baremetalhosts"

function print_nics() {
    local ips
    local eob
    declare -a ips

    readarray -t ips < <(echo "${1}" \
                        | jq '.[] | select(. | .type == "InternalIP") | .address' \
                        | sed 's/"//g')

    eob=','
    for (( i=0; i<${#ips[@]}; i++ )); do
        if [ $((i+1)) -eq ${#ips[@]} ]; then
            eob=""
        fi
        cat <<- EOF
          {
            "ip": "${ips[$i]}",
            "mac": "00:00:00:00:00:00",
            "model": "unknown",
            "speedGbps": 10,
            "vlanId": 0,
            "pxe": true,
            "name": "eth1"
          }${eob}
EOF
    done
}
```

```
function wait_for_json() {
    local name
    local url
    local curl_opts
    local timeout

    local start_time
    local curr_time
    local time_diff

    name="$1"
    url="$2"
    timeout="$3"
    shift 3
    curl_opts="$@"
    echo -n "Waiting for $name to respond"
    start_time=$(date +%s)
    until curl -g -X GET "$url" "${curl_opts[@]}" 2> /dev/null | jq '.' 2> /dev/null >
/dev/null; do
        echo -n "."
        curr_time=$(date +%s)
        time_diff=$((curr_time - start_time))
        if [[ $time_diff -gt $timeout ]]; then
            printf '\nTimed out waiting for %s' "${name}"
            return 1
        fi
        sleep 5
    done
    echo " Success!"
    return 0
}
wait_for_json oc_proxy "${HOST_PROXY_API_PATH}" 10 -H "Accept:
application/json" -H "Content-Type: application/json"

addresses=$(oc get node -n openshift-machine-api "${node_name}" -o json | jq -c
'.status.addresses')

machine_data=$(oc get machines.machine.openshift.io -n openshift-machine-api -o
json "${machine}")
host=$(echo "$machine_data" | jq '.metadata.annotations["metal3.io/BareMetalHost"]' |
cut -f2 -d/ | sed 's/"//g')

if [ -z "$host" ]; then
    echo "Machine $machine is not linked to a host yet." 1>&2
    exit 1
fi

# The address structure on the host doesn't match the node, so extract
# the values we want into separate variables so we can build the patch
# we need.
hostname=$(echo "${addresses}" | jq '.[] | select(. | .type == "Hostname") | .address' |
sed 's/"//g')

set +e
read -r -d '' host_patch << EOF
```

```
{
  "status": {
    "hardware": {
      "hostname": "${hostname}",
      "nics": [
$(print_nics "${addresses}")
      ],
      "systemVendor": {
        "manufacturer": "Red Hat",
        "productName": "product name",
        "serialNumber": ""
      },
      "firmware": {
        "bios": {
          "date": "04/01/2014",
          "vendor": "SeaBIOS",
          "version": "1.11.0-2.el7"
        }
      },
      "ramMebibytes": 0,
      "storage": [],
      "cpu": {
        "arch": "x86_64",
        "model": "Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz",
        "clockMegahertz": 2199.998,
        "count": 4,
        "flags": []
      }
    }
  }
}
EOF
set -e

echo "PATCHING HOST"
echo "${host_patch}" | jq .

curl -s \
    -X PATCH \
    "${HOST_PROXY_API_PATH}/${host}/status" \
    -H "Content-type: application/merge-patch+json" \
    -d "${host_patch}"

oc get baremetalhost -n openshift-machine-api -o yaml "${host}"
```

```
$ bash link-machine-and-node.sh custom-master3 worker-3
```

18. 运行以下命令确认 etcdctl 的成员：

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl member list -w table
```

输出示例

```
+---------+-------+--------+-------------+-------------+-------+
|   ID    | STATUS|  NAME  |   PEER ADDRS | CLIENT ADDRS |LEARNER|
+---------+-------+--------+-------------+-------------+-------+
| 2c18942f|started|worker-3|192.168.111.26|192.168.111.26| false |
| ead4f280|started|worker-5|192.168.111.28|192.168.111.28| false |
| 79153c5a|started|worker-6|192.168.111.29|192.168.111.29| false |
+---------+-------+--------+-------------+-------------+-------+
```

19. 确认 etcd Operator 已配置了所有节点：

```
$ oc get clusteroperator etcd
```

输出示例

```
NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE
etcd 4.11.5 True False False 22h
```

20. 确认 etcdctl 的健康状况：

```
$ oc rsh -n openshift-etcd etcd-worker-3
etcdctl endpoint health
```

输出示例

```
192.168.111.26 is healthy: committed proposal: took = 9.105375ms
192.168.111.28 is healthy: committed proposal: took = 9.15205ms
192.168.111.29 is healthy: committed proposal: took = 10.277577ms
```

21. **确认节点的健康状况：**

```
$ oc get Nodes
```

输出示例

```
NAME      STATUS   ROLES    AGE   VERSION
worker-1   Ready    worker   22h   v1.24.0+3882f8f
master-3   Ready    master   22h   v1.24.0+3882f8f
worker-4   Ready    worker   22h   v1.24.0+3882f8f
master-5   Ready    master   18h   v1.24.0+3882f8f
master-6   Ready    master   40m   v1.24.0+3882f8f
```

22. **确认 ClusterOperators 的健康状况：**

```
$ oc get ClusterOperators
```

输出示例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
authentication                             4.11.5    True        False         False      150m
baremetal                                  4.11.5    True        False         False      22h
cloud-controller-manager                   4.11.5    True        False         False      22h
cloud-credential                           4.11.5    True        False         False      22h
cluster-autoscaler                         4.11.5    True        False         False      22h
config-operator                            4.11.5    True        False         False      22h
console                                    4.11.5    True        False         False      145m
csi-snapshot-controller                    4.11.5    True        False         False      22h
dns                                        4.11.5    True        False         False      22h
etcd                                       4.11.5    True        False         False      22h
image-registry                             4.11.5    True        False         False      22h
ingress                                    4.11.5    True        False         False      22h
insights                                   4.11.5    True        False         False      22h
kube-apiserver                             4.11.5    True        False         False      22h
kube-controller-manager                    4.11.5    True        False         False      22h
kube-scheduler                             4.11.5    True        False         False      22h
kube-storage-version-migrator              4.11.5    True        False         False      148m
machine-api                                4.11.5    True        False         False      22h
machine-approver                           4.11.5    True        False         False      22h
machine-config                             4.11.5    True        False         False      110m
marketplace                                4.11.5    True        False         False      22h
```

```
monitoring                 4.11.5    True        False        False      22h
network                    4.11.5    True        False        False      22h
node-tuning                4.11.5    True        False        False      22h
openshift-apiserver        4.11.5    True        False        False      163m
openshift-controller-manager       4.11.5    True        False        False      22h
openshift-samples          4.11.5    True        False        False      22h
operator-lifecycle-manager         4.11.5    True        False        False      22h
operator-lifecycle-manager-catalog 4.11.5    True        False        False      22h
operator-lifecycle-manager-pkgsvr  4.11.5    True        False        False      22h
service-ca                 4.11.5    True        False        False      22h
storage                    4.11.5    True        False        False      22h
```

23. 确认 ClusterVersion ：

```
$ oc get ClusterVersion
```

输出示例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.11.5    True        False         22h     Cluster version is 4.11.5
```

12.7. 其他资源

- 在一个健康的集群中安装主 control plane 节点
- 使用 REST API 进行身份验证

第 13 章 可选：在 NUTANIX 上安装

如果在 Nutanix 上安装 OpenShift Container Platform，辅助安装程序可将 OpenShift Container Platform 集群与 Nutanix 平台集成，这会将 Machine API 公开给 Nutanix，并使用 Nutanix Container Storage Interface (CSI) 动态置备存储容器。

重要

要部署 OpenShift Container Platform 集群并维护其日常操作，您需要访问具有所需环境要求的 Nutanix 帐户。详情请参阅 环境要求。

13.1. 使用 UI 在 NUTANIX 中添加主机

要使用用户界面 (UI) 在 Nutanix 中添加主机，请从 Assisted Installer 生成发现镜像 ISO。使用最小的发现镜像 ISO。这是默认设置。镜像仅包含使用联网引导主机所需的内容。在引导时会下载大多数内容。ISO 镜像大小为 100MB。

完成后，您必须为 Nutanix 平台创建一个镜像，并创建 Nutanix 虚拟机。

先决条件

- 您已在 Assisted Installer UI 中创建了集群配置集。
- 您已设置了 Nutanix 集群环境，并记录集群名称和子网名称。

流程

1. 在 Cluster details 中，从 Integrate with external partner platforms 下拉列表中选择 Nutanix。Include custom manifest 复选框是可选的。
2. 在 Host discovery 中，单击 Add hosts 按钮。
3. 可选：添加一个 SSH 公钥，以便可以以 core 用户身份连接到 Nutanix 虚拟机。通过登录到集群主机，您可以在安装过程中为您提供调试信息。

    a.

如果本地计算机上没有现有 SSH 密钥对，请按照 为集群节点 SSH 访问生成密钥对 的步骤进行操作。

b. 在 SSH public key 字段中，单击 Browse 来上传包含 SSH 公钥的 id_rsa.pub 文件。或者，将文件拖放到文件管理器的字段中。要查看文件管理器中的文件，请在菜单中选择 Show hidden files。

4. 选择所需的置备类型。

注意

Minimal image file: Provision with virtual media 下载一个将获取引导所需数据的较小的镜像。

5. 在 Networking 中，选择 Cluster-managed networking。Nutanix 不支持 User-managed networking。

a. 可选： 如果集群主机位于需要使用代理的防火墙后面，请选择 Configure cluster-wide proxy settings。输入代理服务器的 HTTP 和 HTTPS URL 的用户名、密码、IP 地址和端口。

注意

代理用户名和密码必须采用 URL 编码。

b. 可选：如果要使用 ignition 文件引导它，请配置发现镜像。如需了解更多详细信息，请参阅配置发现镜像。

6. 点 Generate Discovery ISO。

7. 复制 发现 ISO URL。

8. 在 Nutanix Prism UI 中，按照指示 从 Assisted Installer 中上传发现镜像。

9. 在 Nutanix Prism UI 中，通过 Prism Central 创建 control plane (master) 虚拟机。

   a. 输入 Name。例如，control-plane 或 master。

   b. 输入虚拟机数量。对于 control plane，这应该是 3。

   c. 确保剩余的设置满足 control plane 主机的最低要求。

10. 在 Nutanix Prism UI 中，通过 Prism Central 创建 worker 虚拟机。

    a. 输入 Name。例如，worker。

    b. 输入虚拟机数量。您应该至少创建 2 个 worker 节点。

    c. 确保剩余的设置满足 worker 主机的最低要求。

11. 返回到 Assisted Installer 用户界面，并等待 Assisted Installer 发现主机，每个都处于 Ready 状态。

12. 继续安装过程。

13.2. 使用 API 在 NUTANIX 中添加主机

要使用 API 在 Nutanix 中添加主机，请从 Assisted Installer 生成发现镜像 ISO。使用最小的发现镜像 ISO。这是默认设置。镜像仅包含使用联网引导主机所需的内容。在引导时会下载大多数内容。ISO 镜像大小为 100MB。

完成后，您必须为 Nutanix 平台创建一个镜像，并创建 Nutanix 虚拟机。

先决条件

- 您已设置 Assisted Installer API 身份验证。
- 您已创建了 Assisted Installer 集群配置集。
- 您已创建了辅助安装程序基础架构环境。
- 在 shell 中将您的基础架构环境 ID 导出为 $INFRA_ENV_ID。
- 您已完成 Assisted Installer 集群配置。
- 您已设置了 Nutanix 集群环境，并记录集群名称和子网名称。

流程

1. 如果要使用 ignition 文件引导，请配置发现镜像。
2. 创建 Nutanix 集群配置文件来保存环境变量：

```
$ touch ~/nutanix-cluster-env.sh
```

```
$ chmod +x ~/nutanix-cluster-env.sh
```

如果需要启动新的终端会话，您可以轻松重新加载环境变量。例如：

```
$ source ~/nutanix-cluster-env.sh
```

3. 将 Nutanix 集群的名称分配给配置文件中的 NTX_CLUSTER_NAME 环境变量：

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_NAME=<cluster_name>
EOF
```

将 <cluster_name> 替换为 Nutanix 集群的名称。

4. 将 Nutanix 集群的子网名称分配给配置文件中的 NTX_SUBNET_NAME 环境变量：

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_NAME=<subnet_name>
EOF
```

将 <subnet_name> 替换为 Nutanix 集群子网的名称。

5. 刷新 API 令牌：

```
$ source refresh-token
```

6. 获取下载 URL：

```
$ curl -H "Authorization: Bearer ${API_TOKEN}" \
https://api.openshift.com/api/assisted-install/v2/infra-
envs/${INFRA_ENV_ID}/downloads/image-url
```

7. 创建 Nutanix 镜像配置文件：

```
$ cat << EOF > create-image.json
{
 "spec": {
  "name": "ocp_ai_discovery_image.iso",
  "description": "ocp_ai_discovery_image.iso",
  "resources": {
    "architecture": "X86_64",
    "image_type": "ISO_IMAGE",
    "source_uri": "<image_url>",
    "source_options": {
      "allow_insecure_connection": true
    }
  }
 },
 "metadata": {
  "spec_version": 3,
  "kind": "image"
 }
}
EOF
```

将 <image_url> 替换为从上一步中下载的镜像 URL。

8. **创建 Nutanix 镜像：**

```
$ curl  -k -u <user>:'<password>' -X 'POST' \
'https://<domain-or-ip>:<port>/api/nutanix/v3/images \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./create-image.json | jq '.metadata.uuid'
```

将 <user> 替换为 Nutanix 用户名。将 '<password>' 替换为 Nutanix 密码。将 <domain-or-ip> 替换为 Nutanix plaform 的域名或 IP 地址。将 <port> 替换为 Nutanix 服务器的端口。端口默认为 9440。

9. **将返回的 UUID 分配给配置文件中的 NTX_IMAGE_UUID 环境变量：**

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_IMAGE_UUID=<uuid>
EOF
```

10. **获取 Nutanix 集群 UUID：**

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/clusters/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "cluster"
}'  | jq '.entities[] | select(.spec.name=="<nutanix_cluster_name>") | .metadata.uuid'
```

将 <user> 替换为 Nutanix 用户名。将 '<password>' 替换为 Nutanix 密码。将 <domain-or-ip> 替换为 Nutanix plaform 的域名或 IP 地址。将 <port> 替换为 Nutanix 服务器的端口。端口默认为 9440。将 <nutanix_cluster_name> 替换为 Nutanix 集群的名称。

11. **将返回的 Nutanix 集群 UUID 分配给配置文件中的 NTX_CLUSTER_UUID 环境变量：**

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_CLUSTER_UUID=<uuid>
EOF
```

将 <uuid> 替换为 Nutanix 集群的返回 UUID。

12. **获取 Nutanix 集群的子网 UUID：**

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/subnets/list' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "kind": "subnet",
  "filter": "name==<subnet_name>"
}' | jq '.entities[].metadata.uuid'
```

将 <user> 替换为 Nutanix 用户名。将 '<password>' 替换为 Nutanix 密码。将 <domain-or-ip> 替换为 Nutanix plaform 的域名或 IP 地址。将 <port> 替换为 Nutanix 服务器的端口。端口默认为 9440。将 <subnet_name> 替换为集群子网的名称。

13. **将返回的 Nutanix 子网 UUID 分配给配置文件中的 NTX_CLUSTER_UUID 环境变量：**

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_SUBNET_UUID=<uuid>
EOF
```

将 <uuid> 替换为集群子网的返回 UUID。

14. **确保设置了 Nutanix 环境变量：**

```
$ source ~/nutanix-cluster-env.sh
```

15. **为每个 Nutanix 主机创建一个虚拟机配置文件。创建三个 control plane (master) 虚拟机和至少两个 worker 虚拟机。例如：**

```
$ touch create-master-0.json
```

```
$ cat << EOF > create-master-0.json
{
  "spec": {
    "name": "<host_name>",
    "resources": {
      "power_state": "ON",
      "num_vcpus_per_socket": 1,
      "num_sockets": 16,
      "memory_size_mib": 32768,
      "disk_list": [
        {
```

```
            "disk_size_mib": 122880,
            "device_properties": {
              "device_type": "DISK"
            }
          },
          {
            "device_properties": {
              "device_type": "CDROM"
            },
            "data_source_reference": {
              "kind": "image",
              "uuid": "$NTX_IMAGE_UUID"
            }
          }
        ],
        "nic_list": [
          {
            "nic_type": "NORMAL_NIC",
            "is_connected": true,
            "ip_endpoint_list": [
              {
                "ip_type": "DHCP"
              }
            ],
            "subnet_reference": {
              "kind": "subnet",
              "name": "$NTX_SUBNET_NAME",
              "uuid": "$NTX_SUBNET_UUID"
            }
          }
        ],
        "guest_tools": {
          "nutanix_guest_tools": {
            "state": "ENABLED",
            "iso_mount_state": "MOUNTED"
          }
        }
      },
      "cluster_reference": {
        "kind": "cluster",
        "name": "$NTX_CLUSTER_NAME",
        "uuid": "$NTX_CLUSTER_UUID"
      }
    },
    "api_version": "3.1.0",
    "metadata": {
      "kind": "vm"
    }
}
EOF
```

将 <host_name> 替换为主机的名称。

16. 引导每个 Nutanix 虚拟机：

```
$ curl -k -u <user>:'<password>' -X 'POST' \
  'https://<domain-or-ip>:<port>/api/nutanix/v3/vms' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @./<vm_config_file_name> | jq '.metadata.uuid'
```

将 <user> 替换为 Nutanix 用户名。将 '<password>' 替换为 Nutanix 密码。将 <domain-or-ip> 替换为 Nutanix plaform 的域名或 IP 地址。将 <port> 替换为 Nutanix 服务器的端口。端口默认为 9440。将 <vm_config_file_name> 替换为虚拟机配置文件的名称。

17. 将返回的虚拟机 UUID 分配给配置文件中的唯一环境变量：

```
$ cat << EOF >> ~/nutanix-cluster-env.sh
export NTX_MASTER_0_UUID=<uuid>
EOF
```

将 <uuid> 替换为虚拟机返回的 UUID。

注意

环境变量必须具有每个虚拟机的唯一名称。

18. 等待 Assisted Installer 发现每个虚拟机，并已通过验证。

```
$ curl -s -X GET "https://api.openshift.com/api/assisted-
install/v2/clusters/$CLUSTER_ID"
--header "Content-Type: application/json"
-H "Authorization: Bearer $API_TOKEN"
| jq '.enabled_host_count'
```

19. 修改集群定义以启用与 Nutanix 集成：

```
$ curl https://api.openshift.com/api/assisted-install/v2/clusters/${CLUSTER_ID} \
-X PATCH \
-H "Authorization: Bearer ${API_TOKEN}" \
-H "Content-Type: application/json" \
-d '
{
```

```
  "platform_type":"nutanix"
}
' | jq
```

20. 继续安装过程。

13.3. NUTANIX 安装后配置

按照以下步骤完成并验证 OpenShift Container Platform 与 Nutanix 云提供商的集成。

先决条件

- Assisted Installer 成功完成安装集群。
- 集群连接到 console.redhat.com。
- 您可以访问 Red Hat OpenShift Container Platform 命令行界面。

13.3.1. 更新 Nutanix 配置设置

使用辅助安装程序在 Nutanix 平台上安装 OpenShift Container Platform 后，您必须手动更新以下 Nutanix 配置设置：

- <prismcentral_username> : Nutanix Prism Central 用户名。
- <prismcentral_password> : Nutanix Prism Central 密码。
- <prismcentral_address> : Nutanix Prism Central 地址。
- <prismcentral_port>: Nutanix Prism Central 端口。
- <prismelement_username> : Nutanix Prism Element 用户名。

- <prismelement_password> : Nutanix Prism Element 密码。
- <prismelement_address> : Nutanix Prism Element 地址。
- <prismelement_port>: Nutanix Prism Element 端口。
- <prismelement_clustername>: Nutanix Prism Element 集群名称。
- <nutanix_storage_container> : Nutanix Prism storage 容器。

流程

1. 在 OpenShift Container Platform 命令行界面中，更新 Nutanix 集群配置设置：

```
$ oc patch infrastructure/cluster --type=merge --patch-file=/dev/stdin <<-EOF
{
  "spec": {
    "platformSpec": {
      "nutanix": {
        "prismCentral": {
          "address": "<prismcentral_address>",
          "port": <prismcentral_port>
        },
        "prismElements": [
          {
            "endpoint": {
              "address": "<prismelement_address>",
              "port": <prismelement_port>
            },
            "name": "<prismelement_clustername>"
          }
        ]
      },
      "type": "Nutanix"
    }
  }
}
EOF
```

输出示例

```
infrastructure.config.openshift.io/cluster patched
```

如需了解更多详细信息，请参阅在 Nutanix 上创建机器集。

2. 创建 Nutanix secret：

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: nutanix-credentials
  namespace: openshift-machine-api
type: Opaque
stringData:
 credentials: |
[{"type":"basic_auth","data":{"prismCentral":
{"username":"${<prismcentral_username>}","password":"${<prismcentral_password>
}"},"prismElements":null}}]
EOF
```

输出示例

```
secret/nutanix-credentials created
```

3. 安装 OpenShift Container Platform 版本 4.13 或更高版本时，更新 Nutanix 云提供商配置：

   a. 获取 Nutanix 云提供商配置 YAML 文件：

```
$ oc get cm cloud-provider-config -o yaml -n openshift-config > cloud-provider-
config-backup.yaml
```

   b. 创建一个配置文件的备份：

```
$ cp cloud-provider-config_backup.yaml cloud-provider-config.yaml
```

c. 打开配置 YAML 文件：

```
$ vi cloud-provider-config.yaml
```

d. 编辑配置 YAML 文件，如下所示：

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: cloud-provider-config
  namespace: openshift-config
data:
  config: |
    {
     "prismCentral": {
      "address": "<prismcentral_address>",
      "port":<prismcentral_port>,
      "credentialRef": {
        "kind": "Secret",
        "name": "nutanix-credentials",
        "namespace": "openshift-cloud-controller-manager"
      }
     },
     "topologyDiscovery": {
      "type": "Prism",
      "topologyCategories": null
     },
     "enableCustomLabeling": true
    }
```

e. 应用配置更新：

```
$ oc apply -f cloud-provider-config.yaml
```

输出示例

```
Warning: resource configmaps/cloud-provider-config is missing the
kubectl.kubernetes.io/last-applied-configuration annotation which is required by
oc apply. oc apply should only be used on resources created declaratively by
either oc create --save-config or oc apply. The missing annotation will be patched
automatically.

configmap/cloud-provider-config configured
```

13.3.2. 创建 Nutanix CSI Operator 组

为 Nutanix CSI Operator 创建一个 Operator 组。

注意

有关 operator 组的描述和相关的概念，请参阅 *其它资源* 中的 *常用 Operator 框架术语*。

流程

1. 打开 Nutanix CSI Operator Group YAML 文件：

```
$ vi openshift-cluster-csi-drivers-operator-group.yaml
```

2. 编辑 YAML 文件，如下所示：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  generateName: openshift-cluster-csi-drivers
  namespace: openshift-cluster-csi-drivers
spec:
  targetNamespaces:
  - openshift-cluster-csi-drivers
  upgradeStrategy: Default
```

3. 创建 Operator 组：

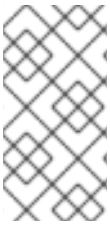
```
$ oc create -f openshift-cluster-csi-drivers-operator-group.yaml
```

输出示例

```
operatorgroup.operators.coreos.com/openshift-cluster-csi-driversjw9cd created
```

13.3.3. 安装 Nutanix CSI Operator

Kubernetes 的 Nutanix Container Storage Interface (CSI) Operator 部署和管理 Nutanix CSI 驱动程序。



注意

有关通过 OpenShift Container Platform Web 控制台执行此步骤的说明，请参阅 *其它资源* 中的 *Nutanix CSI Operator* 文档中的 *安装 Operator* 部分。

流程

1. 获取 Nutanix CSI Operator YAML 文件的参数值：

   a. 检查 Nutanix CSI Operator 是否存在：

   ```
   $ oc get packagemanifests | grep nutanix
   ```

   输出示例

   ```
   nutanixcsioperator   Certified Operators   129m
   ```

   b. 将 Operator 的默认渠道分配给一个 BASH 变量：

   ```
   $ DEFAULT_CHANNEL=$(oc get packagemanifests nutanixcsioperator -o jsonpath=
   {.status.defaultChannel})
   ```

   c. 将 Operator 的起始集群服务版本(CSV)分配给一个 BASH 变量：

   ```
   $ STARTING_CSV=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
   {.status.channels[*].currentCSV\})
   ```

d. **将订阅的目录源分配给一个 BASH 变量：**

```
$ CATALOG_SOURCE=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
{.status.catalogSource\})
```

e. **将 Nutanix CSI Operator 源命名空间分配给一个 BASH 变量：**

```
$ SOURCE_NAMESPACE=$(oc get packagemanifests nutanixcsioperator -o jsonpath=\
{.status.catalogSourceNamespace\})
```

2. **使用 BASH 变量创建 Nutanix CSI Operator YAML 文件：**

```
$ cat << EOF > nutanixcsioperator.yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: nutanixcsioperator
  namespace: openshift-cluster-csi-drivers
spec:
  channel: $DEFAULT_CHANNEL
  installPlanApproval: Automatic
  name: nutanixcsioperator
  source: $CATALOG_SOURCE
  sourceNamespace: $SOURCE_NAMESPACE
  startingCSV: $STARTING_CSV
EOF
```

3. **创建 CSI Nutanix Operator：**

```
$ oc apply -f nutanixcsioperator.yaml
```

输出示例

```
subscription.operators.coreos.com/nutanixcsioperator created
```

4. **运行以下命令，直到 Operator 订阅状态变为 AtLatestKnown。这表明 Operator 订阅已创建，并可能需要一些时间。**

-

```
$ oc get subscription nutanixcsioperator -n openshift-cluster-csi-drivers -o 'jsonpath=
{..status.state}'
```

13.3.4. 部署 Nutanix CSI 存储驱动程序

Kubernetes 的 Nutanix Container Storage Interface (CSI)驱动程序为有状态的应用程序提供可扩展的和持久的存储。

注意

有关通过 OpenShift Container Platform Web 控制台执行此步骤的说明，请参阅 *其它资源* 中 *Nutanix CSI Operator* 文档中的 *使用 Operator 安装 CSI 驱动程序* 部分。

流程

1. **创建一个 NutanixCsiStorage 资源来部署驱动程序：**

```
$ cat <<EOF | oc create -f -
apiVersion: crd.nutanix.com/v1alpha1
kind: NutanixCsiStorage
metadata:
  name: nutanixcsistorage
  namespace: openshift-cluster-csi-drivers
spec: {}
EOF
```

输出示例

```
snutanixcsistorage.crd.nutanix.com/nutanixcsistorage created
```

2. **为 CSI 存储驱动程序创建一个 Nutanix secret YAML 文件：**

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: ntnx-secret
  namespace: openshift-cluster-csi-drivers
stringData:
```

```
  # prism-element-ip:prism-port:admin:password
  key:
<prismelement_address:prismelement_port:prismcentral_username:prismcentral_password>
(1)
EOF
```

注意

(1) **使用实际值替换这些参数，并保持相同的格式。**

输出示例

```
secret/nutanix-secret created
```

13.3.5. 验证安装后配置

运行以下命令来验证配置。

流程

1. 验证您是否可以创建一个存储类：

```
$ cat <<EOF | oc create -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: nutanix-volume
  annotations:
    storageclass.kubernetes.io/is-default-class: 'true'
provisioner: csi.nutanix.com
parameters:
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-cluster-csi-drivers
  csi.storage.k8s.io/provisioner-secret-name: ntnx-secret
  storageContainer: <nutanix_storage_container> (1)
  csi.storage.k8s.io/controller-expand-secret-name: ntnx-secret
  csi.storage.k8s.io/node-publish-secret-namespace: openshift-cluster-csi-drivers
  storageType: NutanixVolumes
  csi.storage.k8s.io/node-publish-secret-name: ntnx-secret
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-cluster-csi-drivers
```

```
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
EOF
```

注意

1

从 Nutanix 配置中获取 <nutanix_storage_container>，例如 SelfServiceContainer。

输出示例

```
storageclass.storage.k8s.io/nutanix-volume created
```

2. 验证您是否可以创建 Nutanix 持久性卷声明(PVC)：

   a. 创建持久性卷声明(PVC)：

```
$ cat <<EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nutanix-volume-pvc
  namespace: openshift-cluster-csi-drivers
  annotations:
    volume.beta.kubernetes.io/storage-provisioner: csi.nutanix.com
    volume.kubernetes.io/storage-provisioner: csi.nutanix.com
  finalizers:
    - kubernetes.io/pvc-protection
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: nutanix-volume
  volumeMode: Filesystem
EOF
```

   输出示例

```
persistentvolumeclaim/nutanix-volume-pvc created
```

b. 验证持久性卷声明(PVC)状态是否为 Bound：

```
$ oc get pvc -n openshift-cluster-csi-drivers
```

输出示例

```
NAME                  STATUS   VOLUME   CAPACITY   ACCESS MODES
STORAGECLASS     AGE
nutanix-volume-pvc   Bound                                      nutanix-volume   52s
```

其他资源

- 在 Nutanix 上创建机器集。
- Nutanix CSI Operator
- 存储管理
- 常见 Operator 框架术语

第 14 章 可选：在 VSPHERE 上安装

辅助安装程序将 OpenShift Container Platform 集群与 vSphere 平台集成，其将 Machine API 公开给 vSphere ，并启用自动扩展。

14.1. 在 VSPHERE 中添加主机

您可以使用在线 vSphere 客户端或 govc vSphere CLI 工具将主机添加到 Assisted Installer 集群。以下流程演示了使用 govc CLI 工具添加主机。要使用在线 vSphere 客户端，请参阅 vSphere 的文档。

要使用 vSphere govc CLI 在 vSphere 中添加主机，请从 Assisted Installer 生成发现镜像 ISO。最小发现镜像 ISO 是默认设置。此镜像仅包含使用网络引导主机所需的内容。在引导时会下载大多数内容。ISO 镜像大小为 100MB。

完成后，您必须为 vSphere 平台创建一个镜像，并创建 vSphere 虚拟机。

先决条件

- 您使用 vSphere 7.0.2 或更高版本。
- 已安装并配置了 vSphere govc CLI 工具。
- 在 vSphere 中，已将 clusterSet disk.enableUUID 设置为 true。
- 您已在 Assisted Installer web 控制台中创建集群，或者
- 您已使用 API 创建了一个辅助安装程序集群配置文件和基础架构环境。
- 您已在 shell 中将您的基础架构环境 ID 导出为 $INFRA_ENV_ID。

流程

1. 如果要使用 ignition 文件引导，请配置发现镜像。

2. 在 Cluster details 中，从 Integrate with external partner platforms 下拉列表中选择 vSphere。Include custom manifest 复选框是可选的。

3. 在 Host discovery 中，单击 Add hosts 按钮，并选择调配类型。

4. 添加 SSH 公钥，以便您可以以 core 用户身份连接到 vSphere 虚拟机。通过登录到集群主机，您可以在安装过程中为您提供调试信息。

   a. 如果本地计算机上没有现有 SSH 密钥对，请按照 为集群节点 SSH 访问生成密钥对 的步骤进行操作。

   b. 在 SSH public key 字段中，单击 Browse 来上传包含 SSH 公钥的 id_rsa.pub 文件。或者，将文件拖放到文件管理器的字段中。要查看文件管理器中的文件，请在菜单中选择 Show hidden files。

5. 选择所需的发现镜像 ISO。

注意

Minimal image file: Provision with virtual media 下载一个将获取引导所需数据的较小的镜像。

6. 在 Networking 中，选择 Cluster-managed networking 或 User-managed networking:

   a. 可选： 如果集群主机位于需要使用代理的防火墙后面，请选择 Configure cluster-wide proxy settings。输入代理服务器的 HTTP 和 HTTPS URL 的用户名、密码、IP 地址和端口。

注意

代理用户名和密码必须采用 URL 编码。

   b. 可选： 如果集群主机位于带有重新加密 man-in-the-middle (MITM) 代理的网络中，或者集群需要信任证书用于容器镜像 registry，请选择 Configure cluster-wide trusted

certificate 并添加额外的证书。

c. 可选：如果要使用 ignition 文件引导它，请配置发现镜像。如需更多信息，*请参阅附加资源*。

7. 点 Generate Discovery ISO。

8. 复制 发现 ISO URL。

9. 下载发现 ISO：

```
$ wget - O vsphere-discovery-image.iso <discovery_url>
```

将 <discovery_url> 替换为上一步中的 Discovery ISO URL。

10. 在命令行中，关闭并删除任何已存在的虚拟机：

```
$ for VM in $(/usr/local/bin/govc ls /<datacenter>/vm/<folder_name>)
do
  /usr/local/bin/govc vm.power -off $VM
  /usr/local/bin/govc vm.destroy $VM
done
```

将 <datacenter> 替换为数据中心的名称。将 <folder_name> 替换为虚拟机清单文件夹的名称。

11. 如果存在，从数据存储中删除预先存在的 ISO 镜像：

```
$ govc datastore.rm -ds <iso_datastore> <image>
```

将 <iso_datastore> 替换为数据存储的名称。使用 ISO 镜像的名称替换 image。

12. 上传辅助安装程序发现 ISO：

```
$ govc datastore.upload -ds <iso_datastore>  vsphere-discovery-image.iso
```

- 将 <iso_datastore> 替换为数据存储的名称。

注意

集群中的所有节点都必须从发现镜像引导。

13. 引导三个 control plane 节点：

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=16 \
        -m=32768 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

详情请参阅 vm.create。

注意

示例中演示了 control plane 节点所需的最小所需资源。

14. 至少引导两个 worker 节点：

```
$ govc vm.create -net.adapter <network_adapter_type> \
        -disk.controller <disk_controller_type> \
        -pool=<resource_pool> \
        -c=4 \
        -m=8192 \
        -disk=120GB \
        -disk-datastore=<datastore_file> \
        -net.address="<nic_mac_address>" \
        -iso-datastore=<iso_datastore> \
        -iso="vsphere-discovery-image.iso" \
        -folder="<inventory_folder>" \
        <hostname>.<cluster_name>.example.com
```

详情请参阅 vm.create。

注意

示例中演示了 worker 节点所需的最小所需资源。

15. 确保虚拟机正在运行：

```
$  govc ls /<datacenter>/vm/<folder_name>
```

将 <datacenter> 替换为数据中心的名称。将 <folder_name> 替换为虚拟机清单文件夹的名称。

16. 2 分钟后，关闭虚拟机：

```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
   govc vm.power -s=true $VM
done
```

将 <datacenter> 替换为数据中心的名称。将 <folder_name> 替换为虚拟机清单文件夹的名称。

17. 将 disk.enableUUID 设置为 TRUE ：

```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
   govc vm.change -vm $VM -e disk.enableUUID=TRUE
done
```

将 <datacenter> 替换为数据中心的名称。将 <folder_name> 替换为虚拟机清单文件夹的名称。

注意

您必须在所有节点上将 disk.enableUUID 设置为 TRUE，才能使用 vSphere 启用自动扩展。

18. 重启虚拟机：

```
$ for VM in $(govc ls /<datacenter>/vm/<folder_name>)
do
   govc vm.power -on=true $VM
done
```

将 <datacenter> 替换为数据中心的名称。将 <folder_name> 替换为虚拟机清单文件夹的名称。

19. 返回到 Assisted Installer 用户界面，并等待 Assisted Installer 发现主机，每个都处于 Ready 状态。

20. 如果需要，选择角色。

21. 在网络 中，清除 通过 DHCP 服务器分配 IP 复选框。

22. 设置 API VIP 地址。

23. 设置 Ingress VIP 地址。

24. 继续安装过程。

其他资源

- 配置发现镜像

14.2. 使用 CLI 的 VSPHERE 安装后配置

在启用了平台集成功能的 vSphere 上使用 Assisted Installer 安装 OpenShift Container Platform 集群后，您必须手动更新以下 vSphere 配置设置：

- vCenter 用户名
- vCenter 密码
- vCenter 地址
- vCenter 集群
- datacenter
- datastore
- folder

先决条件

- Assisted Installer 成功完成安装集群。
- 集群连接到 console.redhat.com。

流程

1. 为 vCenter 生成 base64 编码的用户名和密码：

```
$ echo -n "<vcenter_username>" | base64 -w0
```

将 <vcenter_username> 替换为您的 vCenter 用户名。

```
$ echo -n "<vcenter_password>" | base64 -w0
```

将 <vcenter_password> 替换为您的 vCenter 密码。

2. 备份 vSphere 凭证：

```
$ oc get secret vsphere-creds -o yaml -n kube-system > creds_backup.yaml
```

3. 编辑 vSphere 凭证：

```
$ cp creds_backup.yaml vsphere-creds.yaml
```

```
$ vi vsphere-creds.yaml
```

```
apiVersion: v1
data:
  <vcenter_address>.username: <vcenter_username_encoded>
  <vcenter_address>.password: <vcenter_password_encoded>
kind: Secret
metadata:
  annotations:
    cloudcredential.openshift.io/mode: passthrough
  creationTimestamp: "2022-01-25T17:39:50Z"
  name: vsphere-creds
  namespace: kube-system
  resourceVersion: "2437"
  uid: 06971978-e3a5-4741-87f9-2ca3602f2658
type: Opaque
```

将 <vcenter_address> 替换为 vCenter 地址。将 <vcenter_username_encoded> 替换为 vSphere 用户名的 base64 编码版本。将 <vcenter_password_encoded> 替换为 vSphere 密码的 base64 编码版本。

4. 替换 vSphere 凭证：

```
$ oc replace -f vsphere-creds.yaml
```

5. 重新部署 kube-controller-manager pod：

```
$ oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-"$( date --rfc-3339=ns )""}}' --type=merge
```

6. 备份 vSphere 云供应商配置：

```
$ oc get cm cloud-provider-config -o yaml -n openshift-config > cloud-provider-config_backup.yaml
```

7. 编辑云供应商配置：

```
$ cloud-provider-config_backup.yaml cloud-provider-config.yaml
```

```
$ vi cloud-provider-config.yaml
```

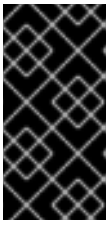
```
apiVersion: v1
data:
  config: |
    [Global]
    secret-name = "vsphere-creds"
    secret-namespace = "kube-system"
    insecure-flag = "1"

    [Workspace]
    server = "<vcenter_address>"
    datacenter = "<datacenter>"
    default-datastore = "<datastore>"
    folder = "/<datacenter>/vm/<folder>"

    [VirtualCenter "<vcenter_address>"]
    datacenters = "<datacenter>"
kind: ConfigMap
metadata:
  creationTimestamp: "2022-01-25T17:40:49Z"
  name: cloud-provider-config
  namespace: openshift-config
  resourceVersion: "2070"
  uid: 80bb8618-bf25-442b-b023-b31311918507
```

将 <vcenter_address> 替换为 vCenter 地址。将 <datacenter> 替换为数据中心的名称。将 <datastore> 替换为数据存储的名称。将 <folder> 替换为包含集群虚拟机的文件夹。

8. 应用云供应商配置：

```
$ oc apply -f cloud-provider-config.yaml
```

9. 使用 uninitialized 污点为集群加污点：



重要

如果要安装 OpenShift Container Platform 4.13 或更高版本，请执行以下的第 9 步到第 12 步。

a. 识别要加污点的节点：

```
$ oc get nodes
```

b. 对每个节点运行以下命令：

```
$ oc adm taint node <node_name>
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

将 <node_name> 替换为节点的名称。

Example

```
$ oc get nodes
NAME           STATUS   ROLES              AGE   VERSION
master-0   Ready    control-plane,master   45h   v1.26.3+379cd9f
master-1   Ready    control-plane,master   45h   v1.26.3+379cd9f
worker-0   Ready    worker             45h   v1.26.3+379cd9f
worker-1   Ready    worker             45h   v1.26.3+379cd9f
master-2   Ready    control-plane,master   45h   v1.26.3+379cd9f

$ oc adm taint node master-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node master-2
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-0
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
$ oc adm taint node worker-1
node.cloudprovider.kubernetes.io/uninitialized=true:NoSchedule
```

10. 备份基础架构配置：

```
$ oc get infrastructures.config.openshift.io -o yaml >
infrastructures.config.openshift.io.yaml.backup
```

11. 编辑基础架构配置：

```
$ cp infrastructures.config.openshift.io.yaml.backup
infrastructures.config.openshift.io.yaml
```

```
$ vi infrastructures.config.openshift.io.yaml
```

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: Infrastructure
  metadata:
    creationTimestamp: "2022-05-07T10:19:55Z"
    generation: 1
    name: cluster
    resourceVersion: "536"
    uid: e8a5742c-6d15-44e6-8a9e-064b26ab347d
  spec:
    cloudConfig:
      key: config
      name: cloud-provider-config
    platformSpec:
      type: VSphere
      vsphere:
        failureDomains:
        - name: assisted-generated-failure-domain
          region: assisted-generated-region
          server: <vcenter_address>
          topology:
            computeCluster: /<data_center>/host/<vcenter_cluster>
            datacenter: <data_center>
            datastore: /<data_center>/datastore/<datastore>
            folder: "/<data_center>/path/to/folder"
            networks:
            - "VM Network"
            resourcePool: /<data_center>/host/<vcenter_cluster>/Resources
          zone: assisted-generated-zone
        nodeNetworking:
          external: {}
          internal: {}
        vcenters:
        - datacenters:
          - <data_center>
          server: <vcenter_address>

kind: List
metadata:
  resourceVersion: ""
```

将 <vcenter_address> 替换为您的 vCenter 地址。将 <datacenter> 替换为 vCenter 数据中心的名称。将 <datastore> 替换为 vCenter 数据存储的名称。将 <folder> 替换为包含集群虚拟机的文件夹。将 <vcenter_cluster> 替换为安装 OpenShift Container Platform 的 vSphere vCenter 集群。

12. 应用基础架构配置：

```
$ oc apply -f infrastructures.config.openshift.io.yaml --overwrite=true
```

14.3. 使用 WEB 控制台进行 VSPHERE 安装后配置

在启用了平台集成功能的 vSphere 上使用 Assisted Installer 安装 OpenShift Container Platform 集群后，您必须手动更新以下 vSphere 配置设置：

- vCenter 地址
- vCenter 集群
- vCenter 用户名
- vCenter 密码
- 数据中心
- 默认数据存储
- 虚拟机文件夹

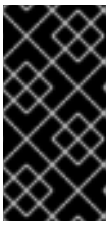
先决条件

- Assisted Installer 成功完成安装集群。

- 集群连接到 console.redhat.com。

流程

1. 在 Administrator 视角中，进入到 Home → Overview。
2. 在 Status 下，点 vSphere connection 打开 vSphere 连接配置向导。
3. 在 vCenter 字段中，输入 vSphere vCenter 服务器的网络地址。这可以是域名，也可以是 IP 地址。它会出现在 vSphere Web 客户端 URL 中，例如 https://[your_vCenter_address]/ui。
4. 在 vCenter cluster 字段中，输入安装 OpenShift Container Platform 的 vSphere vCenter 集群名称。



重要

如果安装了 OpenShift Container Platform 4.13 或更高版本，则此步骤是必需的。

5. 在 Username 字段中，输入 vSphere vCenter 用户名。
6. 在 Password 字段中输入您的 vSphere vCenter 密码。

警告

系统将用户名和密码存储在集群的 kube-system 命名空间中的 vsphere-creds secret 中。不正确的 vCenter 用户名或密码使集群节点不可调度。

7. 在 Datacenter 字段中，输入 vSphere 数据中心的名称，其中包含用于托管集群的虚拟机；例如，SDDC-Datacenter。

8. 在 Default data store 字段中，输入存储持久数据卷的 vSphere 数据存储；例如 /SDDC-Datacenter/datastore/datastorename。

警告

在保存配置后，更新 vSphere 数据中心或默认数据存储会分离任何活跃的 vSphere PersistentVolume。

9. 在 Virtual Machine Folder 字段中，输入包含集群虚拟机的数据中心文件夹；例如，/SDDC-Datacenter/vm/ci-ln-hjg4vg2-c61657-t2gzr。要使 OpenShift Container Platform 安装成功，组成集群的所有虚拟机都必须位于单个数据中心文件夹中。

10. 点 Save Configuration。这会更新 openshift-config 命名空间中的 cloud-provider-config 文件，并启动配置过程。

11. 重新打开 vSphere 连接配置，再展开 Monitored operators 面板。检查 Operator 的状态是否为 Progressing 或 Healthy。

验证

连接配置过程更新 Operator 状态和 control plane 节点。完成大约需要一小时才能完成。在配置过程中，节点将重新引导。以前绑定的 PersistentVolumeClaims 对象可能会断开连接。

按照以下步骤监控配置过程。

1. 检查配置过程是否已成功完成：

   a. 在 OpenShift Container Platform Administrator 视角中，进入到 Home → Overview。

   b. 在 Status 下点 Operators。等待所有操作器状态从 Progressing 变为 All

succeeded。Failed 状态表示配置失败。

c. 在 Status 下，点 Control Plane。等待所有 Control Pane 组件的响应率返回到 100%。失败的 control plane 组件表示配置失败。

失败表示至少一个连接设置不正确。更改 vSphere 连接配置向导中的设置，然后再次保存配置。

2. 通过执行以下步骤来检查您是否可以绑定 PersistentVolumeClaims 对象：

a. 使用以下 YAML 创建 StorageClass 对象：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: vsphere-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
 datastore: YOURVCENTERDATASTORE
 diskformat: thin
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

b. 使用以下 YAML 创建 PersistentVolumeClaims 对象：

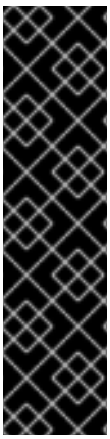
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: test-pvc
 namespace: openshift-config
 annotations:
   volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/vsphere-volume
 finalizers:
   - kubernetes.io/pvc-protection
spec:
 accessModes:
   - ReadWriteOnce
 resources:
   requests:
    storage: 10Gi
 storageClassName: vsphere-sc
 volumeMode: Filesystem
```

具体步骤，请参阅 OpenShift Container Platform 文档中的动态置备。要对

PersistentVolumeClaims 对象进行故障排除，进入到 OpenShift Container Platform Web 控制台的 Administrator 视角中的 Storage → PersistentVolumeClaims。

第 15 章 可选：在 ORACLE CLOUD INFRASTRUCTURE (OCI)上安装

从 OpenShift Container Platform 4.14 及更新版本开始，您可以使用辅助安装程序，使用您提供的基础架构在 Oracle Cloud Infrastructure 上安装集群。Oracle Cloud Infrastructure 提供可满足您对法规遵从性、性能和成本效益的需要的服务。您可以访问 OCI 资源管理器配置来置备和配置 OCI 资源。



重要

对于 OpenShift Container Platform 4.14 和 4.15，OCI 集成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。

本节概述了辅助安装程序 Web 控制台中支持与 Oracle Cloud Infrastructure 集成所需的步骤。它没有记录要在 Oracle 云基础架构中执行的步骤，也没有涵盖这两个平台之间的集成。有关完整和全面的流程，请参阅 使用辅助安装程序在 OCI 上安装集群。

15.1. 生成一个 OCI 兼容的发现 ISO 镜像

通过完成所需的步骤，在辅助安装程序中生成发现 ISO 镜像。在 Oracle Cloud Infrastructure 上安装 OpenShift Container Platform 前，您必须将镜像上传到 Oracle Cloud Infrastructure。

先决条件

- 您在 Oracle Cloud Infrastructure 上创建了一个子隔层和对象存储桶。请参阅 OpenShift Container Platform 文档中的 创建 OCI 资源和服务。
- 您满足安装集群所需的要求。详情请查看先决条件。

流程

1. 登录到 Red Hat Hybrid Cloud 控制台。
2. 点 Create cluster。

3. 在 Cluster type 页面上，点 Datacenter 选项卡。

4. 在 Assisted Installer 部分中，点 Create cluster。

5. 在 Cluster Details 页面上，完成以下字段：

   a. 在 Cluster name 字段中，指定集群名称，如 ocidemo。

   b. 在 Base domain 字段中，指定集群的基域，如 splat-oci.devcluster.openshift.com。在创建隔层和区域后，从 OCI 中获取基域。

   c. 在 OpenShift version 字段中，指定 OpenShift 4.15 或更高版本。

   d. 在 CPU architecture 字段中，指定 x86_64 或 Arm64。

   e. 从 Integrate with external partner platforms 列表中，选择 Oracle Cloud Infrastructure。Include custom manifests 复选框被自动选择。

6. 点击 Next。

7. 在 Operators 页面上，单击 Next。

8. 在 Host Discovery 页面上，执行以下操作：

   a. 单击 Add host 以显示对话框。

   b. 对于 SSH public key 字段，请从本地系统上传一个公共 SSH 密钥。您可以使用 ssh-keygen 生成一个 SSH 密钥对。

c. 点 Generate Discovery ISO 生成发现镜像 ISO 文件。

d. 将文件下载到您的本地系统。然后，您将文件上传到 Oracle Cloud Infrastructure 中的存储桶，来作为一个对象。

15.2. 分配节点角色和自定义清单

置备 Oracle Cloud Infrastructure (OCI)资源并将 OpenShift Container Platform 自定义清单文件上传到 OCI 后，您必须在创建一个实例 OCI 前，完成辅助安装程序上剩余的集群安装步骤。

先决条件

- 您在 OCI 上创建资源堆栈，堆栈包括自定义清单文件和 OCI 资源管理器配置资源。详情请参阅 OpenShift Container Platform 文档中的 下载清单文件和部署资源。

流程

1. 从 Red Hat Hybrid Cloud Console 进到 Host discovery 页面。
2. 在 Role 列下，为每个目标主机名分配一个 Control plane node 或 Worker 节点角色。点击 Next。
3. 接受 Storage 和 Networking 页面的默认设置。
4. 点 Next 进到 Custom manifests 页面。
5. 在 Folder 字段中，选择 manifests。
6. 在 File name 字段中，输入一个值，如 oci-ccm.yml。
7. 在 Content 部分中，单击 Browse。选择 custom_ manifest/manifests/oci-ccm.yml 中的 CCM 清单。

8. 单击 **Add another inventory**。对 **Oracle** 提供的以下清单重复同样的步骤：

   - **CSI** 驱动程序清单：**custom_ manifest/manifests/oci-csi.yml**。
   - **CCM** 机器配置：**custom_ manifest/openshift/machineconfig-ccm.yml.**
   - **CSI** 驱动程序机器配置：**custom_ manifest/openshift/machineconfig-csi.yml**。

9. 完成 **Review and create** 步骤，在 **OCI** 上创建 **OpenShift Container Platform** 集群。

10. 点 **Install cluster** 来完成集群安装。

第 16 章 故障排除

有些情况下，辅助安装程序无法开始安装，或者集群无法正确安装。在这些事件中，了解可能的故障模式以及如何对故障进行故障排除非常有用。

16.1. 发现 ISO 问题故障排除

Assisted Installer 使用 ISO 镜像来运行将主机注册到集群的代理，并在尝试安装 OpenShift 前执行硬件和网络验证。您可以按照以下步骤排除与主机发现相关的问题。

使用发现 ISO 镜像启动主机后，辅助安装程序发现主机并在 Assisted Service web 控制台中显示它。如需了解更多详细信息，请参阅配置发现镜像。

16.1.1. 验证发现代理是否正在运行

先决条件

- 已使用 API 创建基础架构环境，或使用 Web 控制台创建集群。
- 已使用基础架构环境发现 ISO 引导主机，主机无法注册。
- 您有到主机的 SSH 访问权限。
- 在生成发现 ISO 前，在"添加主机"对话框中提供了 SSH 公钥，以便您可以在无需密码的情况下通过 SSH 连接到机器。

流程

1. 验证您的主机已开机。
2. 如果您选择了 DHCP 网络，检查 DHCP 服务器是否已启用。
3. 如果您选择了 静态 IP、网桥和绑定网络，请检查您的配置是否正确。

4. 验证您可以使用 SSH、一个控制台（如 BMC）或虚拟机控制台来访问主机机器：

```
$ ssh core@<host_ip_address>
```

如果没有存储在默认目录中，您可以使用 -i 参数指定私钥文件。

```
$ ssh -i <ssh_private_key_file> core@<host_ip_address>
```

如果您没有 ssh 到主机，则主机在启动过程中会失败，或者无法配置网络。

登录后，您应该看到这个信息：

登录示例

```
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
This is a host being installed by the OpenShift Assisted Installer.
It will be installed from scratch during the installation.
The primary service is agent.service. To watch its status, run:
sudo journalctl -u agent.service
To view the agent log, run:
sudo journalctl TAG=agent
** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** **
```

如果没有看到这条消息，这意味着主机没有使用 assisted-installer ISO 引导。确保正确配置了引导顺序（主机应该从 live-ISO 启动一次）。

5. 检查代理服务日志：

```
$ sudo journalctl -u agent.service
```

在以下示例中，错误表示存在网络问题：

代理服务日志的代理服务日志截图示例

```
Oct 15 11:26:35 localhost systemd[1]: agent. service: Service RestartSec=3s expired, scheduling restart.
Oct 15 11:26:35 localhost systemd[1]: agent. service: Scheduled restart job, restart counter is at 9.
Oct 15 11:26:35 localhost systemd[1]: Stopped agent. service.
Oct 15 11:26:35 localhost systemd[1]: Starting agent. service...
Oct 15 11:26:35 localhost podman[1834]: Trying to pull quay.io/ocpmetal/assisted-installer-agent: lates
Oct 15 11:26:35 localhost podman[1834]:
Get "https://quay.io/v2/": dial top: lookup quay.io on [::1]:53: read udp [::1]:582
Oct 15 11:26:35 localhost podman[1834]: Error: unable to pull quay. io/ocpmetal/assisted-installer-agent:latest: unable to pull
Oct 15 11:26:35 localhost systemd[1]: agent. service: Control process exited, code=exited status=125
Oct 15 11:26:35 localhost systemd[1]: agent. service: Failed with result 'exit-code'.
Oct 15 11:26:35 localhost systemd[1]: Failed to start agent. service.
```

如果拉取代理镜像出现错误，请检查代理设置。验证主机是否已连接到网络。您可以使用 nmcli 来获取有关网络配置的额外信息。

16.1.2. 验证代理可以访问 assisted-service

先决条件

- 已使用 API 创建 Infrastructure 环境，或使用 Web 控制台创建集群。
- 已使用基础架构环境发现 ISO 引导主机，主机无法注册。
- 您验证了发现代理正在运行。

流程

- 检查代理日志以验证代理可以访问 Assisted Service：

```
$ sudo journalctl TAG=agent
```

以下示例中的错误表示代理无法访问 Assisted Service。

代理日志示例

```
Jul 21 19:39:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:39:44" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432367\": dial tcp: lookup api.stage.openshift.com on 192.168.131.1:53: read udp 192.168.131.11:58
016->192.168.131.1:53: i/o timeout" file="step_processor.go:238" request_id=00a041ba-0314-4d00-83f1-486b36b
d02bb
Jul 21 19:40:44 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:44" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=6cb39274-7f5b-4099-9894-912702c77
b09
Jul 21 19:40:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:40:54" leve
l=warning msg="Could not query next steps: Get \"https://api.stage.openshift.com/api/assisted-install/v2/in
fra-envs/ba747803-f85d-40f4-8af4-01d7f0d8914f/hosts/8daa0b33-d10a-46aa-ab59-ea9be2e0c4d9/instructions?times
tamp=1658432444\": net/http: TLS handshake timeout" file="step_processor.go:238" request_id=6cb39274-7f5b-4
099-9894-912702c77b09
Jul 21 19:41:54 test-infra-cluster-7c35a054-master-1 next_step_runne[1909]: time="21-07-2022 19:41:54" leve
l=info msg="Query for next steps" file="step_processor.go:223" request_id=8ca23a1c-4d5c-494b-8d59-9846fcdff
b9e
```

检查您为集群配置的代理设置。如果配置，代理必须允许访问 Assisted Service URL。

16.2. 最小发现 ISO 问题故障排除

当虚拟介质连接的带宽有限时，应使用最小 ISO 镜像。它只包括使用联网引导主机所需的内容。在引导时会下载大多数内容。与 1GB 相比，生成的 ISO 镜像的大小大约为 100MB，用于完整的 ISO 镜像。

16.2.1. 通过中断引导过程对最小 ISO 引导失败的故障排除

如果您的环境需要静态网络配置来访问 Assisted Installer 服务，则该配置中的任何问题可能会阻止最小 ISO 正确引导。如果引导屏幕显示主机无法下载根文件系统镜像，则可能无法正确配置网络。

您可以在 bootstrap 过程早期中断内核引导，然后再下载根文件系统镜像。这可让您访问根控制台并查看网络配置。

rootfs 下载失败示例

```
[**    ] A start job is running for Acquire ▪rootfs image (1min 41s / no limit)[
  104.578592] coreos-livepxe-rootfs[922]: curl: (6) Could not resolve host: api.
openshift.com
[  104.579201] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  104.579600] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  104.580107] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▪rootfs image (1min 46s / no limit)[
  109.619825] coreos-livepxe-rootfs[925]: curl: (6) Could not resolve host: api.
openshift.com
[  109.620608] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  109.621053] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  109.621564] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▪rootfs image (1min 51s / no limit)[
  114.647843] coreos-livepxe-rootfs[928]: curl: (6) Could not resolve host: api.
openshift.com
[  114.648464] coreos-livepxe-rootfs[849]: Couldn't establish connectivity with
the server specified by:
[  114.648821] coreos-livepxe-rootfs[849]: coreos.live.rootfs_url=https://api.op
enshift.com/api/assisted-images/boot-artifacts/rootfs?arch=x86_64&version=4.11
[  114.649323] coreos-livepxe-rootfs[849]: Retrying in 5s...
[   ***] A start job is running for Acquire ▪rootfs image (1min 53s / no limit)
```

流程

1. 将 .spec.kernelArguments 小节添加到您要部署的集群的 infraEnv 对象中：

注意

有关修改基础架构环境的详情，请参考*附加资源*。

```
# ...
spec:
  clusterRef:
    name: sno1
    namespace: sno1
  cpuArchitecture: x86_64
  ipxeScriptType: DiscoveryImageAlways
  kernelArguments:
  - operation: append
    value: rd.break=initqueue (1)
  nmStateConfigLabelSelector:
    matchLabels:
      nmstate-label: sno1
  pullSecretRef:
    name: assisted-deployment-pull-secret
```

(1) rd.break=initqueue 中断在 dracut 主循环上的引导。详情请参阅 对内核引导进行故障排除时可以使用的 rd.break 选项。

2. 等待相关的节点自动重新引导，并使引导在 iniqueue 阶段中止，然后再下载 rootfs。您将被重定向到 root 控制台。

3. 找出并更改不正确的网络配置。以下是一些有用的诊断命令：

   a. 使用 journalctl 查看系统日志，例如：

   ```
   # journalctl -p err //Sorts logs by errors
   # journalctl -p crit //Sorts logs by critical errors
   # journalctl -p warning //Sorts logs by warnings
   ```

   b. 使用 nmcli 查看网络连接信息，如下所示：

   ```
   # nmcli conn show
   ```

   c. 检查配置文件是否有不正确的网络连接，例如：

```
# cat /etc/assisted/network/host0/eno3.nmconnection
```

4. 按 control+d 恢复 bootstrap 过程。服务器下载 rootfs 并完成该过程。

5. 重新打开 infraEnv 对象并删除 .spec.kernelArguments 小节。

其他资源

- 修改基础架构环境

16.3. 更正主机的引导顺序

当作为发现镜像一部分的安装完成后，辅助安装程序将重启主机。 主机必须从其安装磁盘引导，才能继续组成集群。 如果您没有正确配置主机的引导顺序，它将从另一个磁盘启动，这会中断安装。

如果主机再次引导发现镜像，辅助安装程序将立即检测到此事件，并将主机的状态设置为 Installing Pending User Action。 或者，如果 Assisted Installer 没有检测到主机已在分配的时间内引导了正确的磁盘，它也将设置此主机状态。

流程

- 重启主机并设置其引导顺序以从安装磁盘引导。如果您没有选择安装磁盘，辅助安装程序为您选择一个。要查看所选安装磁盘，请点击以展开主机清单中的主机信息，并检查哪个磁盘具有"安装磁盘"角色。

16.4. 修复部分成功安装

有些情况下，辅助安装程序声明安装成功，即使它遇到错误：

- 如果您请求安装 OLM operator 且一个或多个无法安装，请登录到集群的控制台来修复失败。

- 如果您请求安装多个 worker 节点，且至少有一个无法安装，但至少两个成功，请将失败的 worker 添加到安装的集群。

16.5. 将节点添加到集群时的 API 连接失败

当您将节点添加到现有集群来作为第 2 天操作的一部分时，节点会从第 1 天集群下载 ignition 配置文件。如果下载失败，且节点无法连接到集群，则 Host discovery 步骤中的主机的状态将变为 Insufficient。点击此状态会显示以下错误消息：

> The host failed to download the ignition file from <URL>. You must ensure the host can reach the URL. Check your DNS and network configuration or update the IP address or domain used to reach the cluster.
>
> error: ignition file download failed.... no route to host

连接失败有很多可能的原因。以下是一些推荐的操作。

流程

1. 检查集群的 IP 地址和域名：
   a. 点 set the IP or domain used to reach the cluster 超链接。
   b. 在 Update cluster hostname 窗口中，输入正确的集群的 IP 地址或域名。
2. 检查您的 DNS 设置，以确保 DNS 可以解析您提供的域。
3. 确保端口 22624 在所有防火墙中已打开。
4. 检查主机的代理日志，以验证代理是否可以通过 SSH 访问辅助服务：

```
$ sudo journalctl TAG=agent
```

注意

如需了解更多详细信息，请参阅 验证代理是否可以访问辅助服务。