OpenShift Container Platform 4.6

在裸机上部署安装程序置备的集群

安装 IPI OpenShift Container Platform 裸机集群

Last Updated: 2023-02-27

OpenShift Container Platform 4.6 在裸机上部署安装程序置备的集群

安装 IPI OpenShift Container Platform 裸机集群

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本文档提供在使用安装程序置备的集群的裸机基础架构上安装和卸载 OpenShift Container Platform 集群的说明。

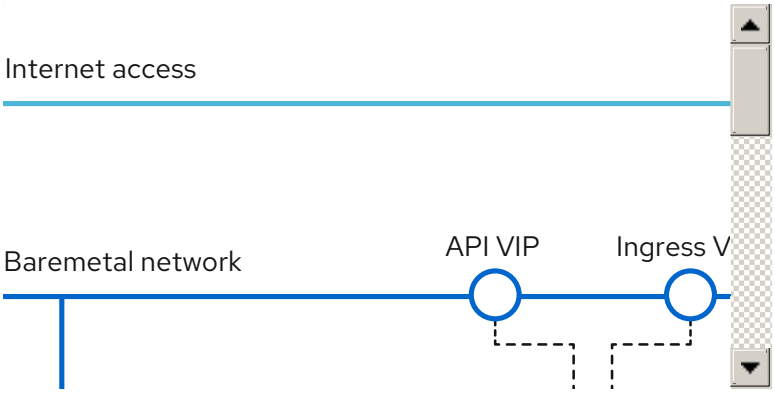
目录

第 1 章 在裸机上部署安装程序置备的集群

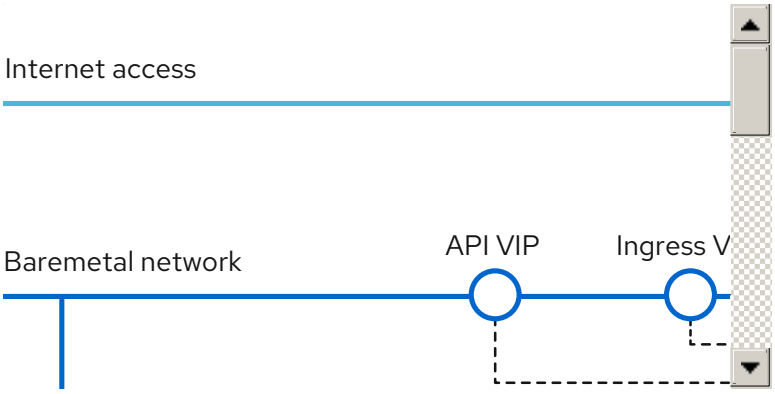
1.1. 概述

安装程序置备的安装支持在裸机节点上安装 OpenShift Container Platform。本指南提供了一种成功安装的方法。

在裸机上的安装程序置备安装过程中，裸机节点上标记为 **provisioner** 的安装程序会创建一个 bootstrap 虚拟机。bootstrap 虚拟机的角色是协助部署 OpenShift Container Platform 集群。bootstrap 虚拟机通过网桥接连接到 **baremetal** 网络和 **provisioning** 网络（如果存在）。



当完成 OpenShift Container Platform control plane 节点安装并可以正常工作时，安装程序会自动销毁 bootstrap 虚拟机，并相应地将虚拟 IP 地址（VIP）移到适当的节点。API VIP 移至 control plane 节点，Ingress VIP 移至 worker 节点。



1.2. 先决条件

OpenShift Container Platform 安装程序置备的安装需要：

1. 安装了 Red Hat Enterprise Linux（RHEL）8.x 的一个置备程序节点。
2. 三个 control plane 节点。
3. 对每个节点的 Baseboard Management Controller（BMC）访问。
4. 至少一个网络：
    a. 一个**必需的**可路由网络
    b. 一个用于置备节点的**可选**网络 ; 和

c. 一个**可选的**管理网络。

在开始 OpenShift Container Platform 安装程序置备的安装前，请确保硬件环境满足以下要求。

1.2.1. 节点要求

安装程序置备的安装有多个硬件节点要求：

- **CPU 架构：** 所有节点都必须使用 **x86_64** CPU 架构。
- **类似的节点：** 红帽建议每个角色都有相同的配置。也就是说，红帽推荐节点具有相同的品牌和型号，它们具有相同的 CPU、内存和存储配置。
- **Baseboard Management Controller: provisioner** 节点必须能够访问每个 OpenShift Container Platform 集群节点的基板管理控制器（BMC）。您可以使用 IPMI、RedFish 或一个私有协议。
- **最新一代：** 节点必须是最新一代。因为安装程序置备的安装依赖于 BMC 协议，所以硬件必须支持 IPMI 密码套件 17。另外，RHEL 8 附带了 RAID 控制器的最新驱动程序。确保节点足以支持 **provisioner** 节点运行 RHEL 8，支持 control plane 和 worker 节点运行 RHCOS 8。
- **registry 节点：**（可选）如果设置了一个断开连接的 mirrored registry，建议 registry 驻留在自己的节点上。
- **provisioner 节点：** 安装程序置备的安装需要一个 **provisioner** 节点。
- **Control plane:** 安装程序置备的安装需要三个 control plane 节点才能进行高可用性。
- **Worker 节点：** 虽然不需要，但典型的生产环境集群具有一个或多个 worker 节点。较小的集群在开发和测试过程中为管理员和开发人员提供更多资源。
- **网络接口：** 每个节点必须至少有一个网络接口用于可路由的 **baremetal** 网络。在使用 **provisioning** 网络进行部署时，每个节点都必须有一个网络接口用于 **provisioning** 网络。使用 **provisioning** 网络是默认配置。对于 provisioning 网络，在 control plane 节点之间必须一致网络接口命名。例如，如果 control plane 节点为 provisioning 网络使用 **eth0** NIC，其他 control plane 节点也必须使用它。
- **统一的可扩展固件接口（UEFI）**：在 **provisioning** 网络中使用 IPv6 时，,安装程序置备的安装需要在所有 OpenShift Container Platform 节点上进行 UEFI 引导。另外，UEFI Device PXE Settings 必需被设置为在 **provisioning** 网络 NIC 中使用 IPv6 协议，但 **忽略 provisioning 网络会删除这个要求。**

1.2.2. 网络要求

OpenShift Container Platform 安装程序置备的安装默认涉及多个网络要求。首先，安装程序置备的安装涉及一个不可路由的 **provisioning** 网络，用于在每个裸机节点上置备操作系统，以及一个可路由的 **baremetal** 网络。因为安装程序置备的安装会部署 **ironic-dnsmasq**，所以网络不能有在同一广播域中运行的其他 DHCP 服务器。网络管理员必须为 OpenShift Container Platform 集群中的每个节点保留 IP 地址。

网络时间协议（NTP）

建议集群中的每个 OpenShift Container Platform 节点都可以访问可使用 DHCP 发现的网络时间协议（NTP）服务器。虽然在没有 NTP 服务器的情况下安装是可能的，异步服务器时钟可能会导致错误。使用 NTP 服务器可以防止这个问题。

配置 NIC

OpenShift Container Platform 使用两个网络部署：

- **provisioning**：**provisioning** 网络是一个 **可选的**、不可路由的网络，用于在作为 OpenShift Container Platform 集群一部分的每个节点上置备底层操作系统。当使用 **provisioning** 网络进行部署时，每个节点上的第一个 NIC（如 **eth0** 或 **eno1** ） **必须**是与 **provisioning** 网络的接口。
- **baremetal**：**baremetal** 网络是一个可路由的网络。当使用 **provisioning** 网络部署时，每个节点的第二个 NIC（如 **eth1** 或 **eno2**）**必须**是与 **baremetal** 网络的接口。当在没有 **provisioning** 网络的情况下部署时，您可以使用每个节点上的任意 NIC 作为与 **provisioning** 的接口。

重要

每个 NIC 应该位于与适当网络对应的独立 VLAN 中。

配置 DNS 服务器

客户端通过 **baremetal** 网络访问 OpenShift Container Platform 集群节点。网络管理员必须配置子域或子区，其中 CN 扩展是集群名称。

```
<cluster-name>.<domain-name>
```

例如：

```
test-cluster.example.com
```

使用 DHCP 服务器为节点保留 IP 地址

对于 **baremetal** 网络，网络管理员必须保留一组 IP 地址，其中包括：

1. 两个虚拟 IP 地址。
   - API 端点的一个 IP 地址
   - 一个用于通配符入口端点的 IP 地址
2. 一个用于 provisioner 节点的 IP 地址。
3. 每个 control plane（master）节点有一个 IP 地址。
4. 每个 worker 节点一个 IP 地址（如果适用）。

下表提供了一个完全限定域名的示范性实施API 和 Nameserver 地址以规范名称（canonical name）扩展开头。control plane 和 worker 节点的主机名只是示例，您可以使用您喜欢的任何主机命名规则。

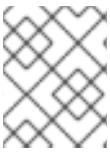
使用	主机名	IP
API	*api.<cluster-name>.<domain>*	*<ip>*
Ingress LB (apps)	**.apps.<cluster-name>.<domain>*	*<ip>*
Provisioner node	*provisioner.<cluster-name>.<domain>*	*<ip>*
Master-0	*openshift-master-0.<cluster-name>.<domain>*	*<ip>*

使用	主机名	IP
Master-1	*openshift-master-1.<cluster-name>-.<domain>*	*<ip>*
Master-2	*openshift-master-2.<cluster-name>.<domain>*	*<ip>*
Worker-0	*openshift-worker-0.<cluster-name>.<domain>*	*<ip>*
Worker-1	*openshift-worker-1.<cluster-name>.<domain>*	*<ip>*
Worker-n	*openshift-worker-n.<cluster-name>.<domain>*	*<ip>*

无 provisioning 网络的额外要求

所有安装程序置备的安装都需要一个 **baremetal** 网络。**baremetal** 网络是一个可路由的网络，用于外部网络访问外部世界。除了向 OpenShift Container Platform 集群节点提供的 IP 地址外，没有 **provisioning** 网络的安装还需要以下：

- 在 **install-config.yaml** 配置文件中将来自 **baremetal** 网络中一个可用 IP 地址设置为 **bootstrapProvisioningIP** 配置设置。
- 在 **install-config.yaml** 配置文件中将来自 **baremetal** 网络中的一个可用 IP 地址设置为 **provisioningHostIP** 配置设置。
- 使用 RedFish Virtual Media/iDRAC Virtual Media 部署 OpenShift Container Platform 集群。



注意

当使用 **provisioning** 网络时，不需要为 **bootstrapProvisioningIP** 和 **provisioningHostIP** 配置额外的 IP 地址。

带外管理 IP 地址的端口访问

带外管理 IP 地址位于独立于节点的网络中。为确保带外管理在安装过程中可以与裸机节点通信，必须将带外管理 IP 地址授予访问权限到 TCP 6180 端口。

1.2.3. 配置节点

在使用 provisioning 网络时配置节点

集群中的每个节点都需要以下配置才能正确安装。

警告

如果在节点间不匹配则会导致安装失败。

虽然集群节点可以包含多于 2 个 NIC，但安装过程只关注于前两个 NIC:

NIC	网络	VLAN
NIC1	**provisioning**	<provisioning-vlan>
NIC2	**baremetal**	<baremetal-vlan>

NIC1 是一个不可路由的网络（provisioning），仅用于安装 OpenShift Container Platform 集群。

置备程序节点上的 Red Hat Enterprise Linux（RHEL）8.x 安装过程可能会有所不同。要使用本地 Satellite 服务器或者 PXE 服务器安装 Red Hat Enterprise Linux（RHEL）8.x，PXE 启用 NIC2。

PXE	引导顺序
NIC1 PXE-enabled **provisioning** 网络	1
NIC2 **baremetal** 网络。启用 PXE 是可选的。	2

注意

确保在所有其他 NIC 中禁用 PXE。

配置 control plane 和 worker 节点，如下所示：

PXE	引导顺序
NIC1 PXE-enabled（provisioning 网络）	1

在没有 provisioning 网络的情况下配置节点

安装过程需要一个 NIC：

NIC	网络	VLAN
NICx	**baremetal**	<baremetal-vlan>

NICx 是一个可路由的网络（baremetal），它用于安装 OpenShift Container Platform 集群，并可路由到互联网。

1.2.4. 带外管理

节点通常还会有一个额外的、由 Baseboard Management Controller（BMC）使用的 NIC。这些 BMC 必须可以通过 provisioner 节点访问。

每个节点需要可以通过带外管理进行访问。在使用带外管理网络时，provisioner 节点需要访问带外管理网络才能成功安装 OpenShift Container Platform 4。

带外管理设置已超出本文档的范围。我们建议单独设置一个带外管理网络。但是，使用 provisioning 网络或 baremetal 网络是有效的选项。

1.2.5. 安装所需的数据

在安装 OpenShift Container Platform 集群前，从所有集群节点收集以下信息：

- 带外管理 IP
  - 示例
    - Dell (iDRAC) IP
    - HP (iLO) IP

使用 provisioning 网络时

- NIC1 (provisioning) MAC 地址
- NIC2 (baremetal) MAC 地址

在省略 provisioning 网络时

- NICx (baremetal) MAC 地址

1.2.6. 节点验证清单

使用 provisioning 网络时

❑ 为 provisioning 网络配置了 NIC1 VLAN。

❑ 为 baremetal 网络配置了 NIC2 VLAN。

❑ 在 provisioner、control plane (master) 和 worker 节点上，NIC1 启用了 PXE。

❑ PXE 在所有其他 NIC 上都被禁用。

❑ 配置了 control plane 和 worker 节点。

❑ 所有节点都可以通过带外管理访问。

❑ 已创建一个单独的管理网络（可选）

❑ 安装所需的数据。

在忽略 provisioning 网络时

- ❑ 为 baremetal 网络配置 NICx VLAN。
- ❑ 配置了 control plane 和 worker 节点。
- ❑ 所有节点都可以通过带外管理访问。
- ❑ 已创建一个单独的管理网络（可选）
- ❑ 安装所需的数据。

1.3. 为 OPENSHIFT 安装设置环境

1.3.1. 在置备程序节点上安装 RHEL

在网络配置完成后，下一步是在置备程序节点上安装 RHEL 8.x。在安装 OpenShift Container Platform 集群时，安装程序使用 provisioner 节点作为编配器。在本文档中，在置备程序节点上安装 RHEL 超出了范围。但是，选项包括但不局限于使用 RHEL Satellite 服务器、PXE 或安装介质。

1.3.2. 为 OpenShift Container Platform 安装准备 provisioner 序节点

执行以下步骤准备环境。

流程

1. 通过 ssh 登录到 provisioner 节点。
2. 创建一个非 root 用户（kni），并为该用户提供 sudo 权限。

```
# useradd kni
# passwd kni
# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
# chmod 0440 /etc/sudoers.d/kni
```

3. 为新用户生成 ssh 密钥。

```
# su - kni -c "ssh-keygen -t ed25519 -f /home/kni/.ssh/id_rsa -N ''"
```

4. 使用新创建的用户登陆到 provisioner 节点。

```
# su - kni
$
```

5. 使用 Red Hat Subscription Manager 注册 provisioner 节点：

```
$ sudo subscription-manager register --username=<user> --password=<pass> --auto-attach
$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-appstream-rpms --enable=rhel-8-for-x86_64-baseos-rpms
```

注意

有关 Red Hat Subscription Manager 的详情，请查看使用和配置 Red Hat Subscription Manager。

6. 安装以下软件包。

```
$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. 修改用户以便为新创建的用户中添加 libvirt 组。

```
$ sudo usermod --append --groups libvirt <user>
```

8. 重启 firewalld 并启用 http 服务。

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --zone=public --add-service=http --permanent
$ sudo firewall-cmd --reload
```

9. 启动并启用 libvirtd 服务。

```
$ sudo systemctl enable libvirtd --now
```

10.

创建 default 存储池并启动它。

```
$ sudo virsh pool-define-as --name default --type dir --target /var/lib/libvirt/images
$ sudo virsh pool-start default
$ sudo virsh pool-autostart default
```

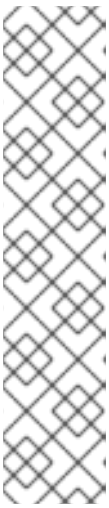
11.

配置网络。

注意

此步骤也可以从 web 控制台运行。

```
$ export PUB_CONN=<baremetal_nic_name>
$ export PROV_CONN=<prov_nic_name>
$ sudo nohup bash -c "
    nmcli con down \"$PROV_CONN\"
    nmcli con down \"$PUB_CONN\"
    nmcli con delete \"$PROV_CONN\"
    nmcli con delete \"$PUB_CONN\"
    # RHEL 8.1 appends the word \"System\" in front of the connection, delete in case it
exists
    nmcli con down \"System $PUB_CONN\"
    nmcli con delete \"System $PUB_CONN\"
    nmcli connection add ifname provisioning type bridge con-name provisioning
    nmcli con add type bridge-slave ifname \"$PROV_CONN\" master provisioning
    nmcli connection add ifname baremetal type bridge con-name baremetal
    nmcli con add type bridge-slave ifname \"$PUB_CONN\" master baremetal
    pkill dhclient;dhclient baremetal
    nmcli connection modify provisioning ipv6.addresses fd00:1101::1/64 ipv6.method
manual
    nmcli con down provisioning
    nmcli con up provisioning
"
```



注意

执行此步骤后 ssh 连接可能会断开。

只要无法通过 baremetal 网络路由，IPv6 地址可以是任何地址。

在使用 IPv6 地址时，请确保启用了 UEFI，并且将 UEFI PXE 设置设为 IPv6 协议。

12. 在 provisioning 网络连接上配置 IPv4 地址。

```
$ nmcli connection modify provisioning ipv4.addresses 172.22.0.254/24 ipv4.method
manual
```

13. 重新 ssh 到 provisioner 节点（如果需要）。

```
# ssh kni@provisioner.<cluster-name>.<domain>
```

14. 验证连接桥接是否已正确创建。

```
$ sudo nmcli con show
```

```
NAME               UUID                                  TYPE      DEVICE
baremetal          4d5133a5-8351-4bb9-bfd4-3af264801530  bridge    baremetal
provisioning       43942805-017f-4d7d-a2c2-7cb3324482ed  bridge    provisioning
virbr0             d9bca40f-eee1-410b-8879-a2d4bb0465e7  bridge    virbr0
bridge-slave-eno1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eno1
bridge-slave-eno2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eno2
```

15. 创建一个 pull-secret.txt 文件。

```
$ vim pull-secret.txt
```

在 Web 浏览器中，进入 Install on Bare Metal with user-provisioned infrastructure，再向下滚动到 Downloads 部分。点 Copy pull secret。将内容粘贴到 pull-secret.txt 文件中，并将内容保存到 kni 用户的主目录中。

1.3.3. 检索 OpenShift Container Platform 安装程序

使用安装程序的 latest-4.x 版本来部署最新的 OpenShift Container Platform 发行版本：

```
$ export VERSION=latest-4.6
export RELEASE_IMAGE=$(curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/release.txt | grep 'Pull From: quay.io' | awk -F ' ' '{print $3}')
```

其他资源

- 如需了解不同发行 频道 的说明，请参阅 OpenShift Container Platform 升级频道和发行版本。

1.3.4. 提取 OpenShift Container Platform 安装程序

在获取安装程序后，下一步就是展开它。

流程

1. 设置环境变量：

```
$ export cmd=openshift-baremetal-install
$ export pullsecret_file=~/pull-secret.txt
$ export extract_dir=$(pwd)
```

2. 获取 oc 二进制文件：

```
$ curl -s https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/$VERSION/openshift-client-linux.tar.gz | tar zxvf - oc
```

3. 展开安装程序：

```
$ sudo cp oc /usr/local/bin
$ oc adm release extract --registry-config "${pullsecret_file}" --command=$cmd --to
"${extract_dir}" ${RELEASE_IMAGE}
$ sudo cp openshift-baremetal-install /usr/local/bin
```

1.3.5. 创建 RHCOS 镜像缓存（可选）

要使用镜像缓存，您必须下载两个镜像：Bootstrap 虚拟机使用的 Red Hat Enterprise Linux CoreOS（RHCOS）镜像，以及安装程序用来置备不同节点的 RHCOS 镜像。镜像缓存是可选的，但在有限带宽的网络中运行安装程序时特别有用。

如果您在带有有限带宽的网络上运行安装程序，而 RHCOS 镜像下载时间超过 15 到 20 分钟，则安装程序会超时。在这样的情况下，可以将镜像缓存到网页服务器上。

使用以下步骤安装包含镜像的容器。

1. 安装 podman。

```
$ sudo dnf install -y podman
```

2. 打开防火墙端口 8080 以用于 RHCOS 镜像缓存。

```
$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

```
$ sudo firewall-cmd --reload
```

3. 创建用于存储 bootstraposimage 和 clusterosimage 的目录。

```
$ mkdir /home/kni/rhcos_image_cache
```

4. 为新创建的目录设置正确的 SELinux 上下文。

```
$ sudo semanage fcontext -a -t httpd_sys_content_t
"/home/kni/rhcos_image_cache(/.*)?"
$ sudo restorecon -Rv rhcos_image_cache/
```

5. 从安装程序获取提交 ID。这个 ID 决定了安装程序需要下载的镜像。

```
$ export COMMIT_ID=$(/usr/local/bin/openshift-baremetal-install version | grep '^built
from commit' | awk '{print $4}')
```

6. 获取安装程序将部署到节点上的 RHCOS 镜像的 URI。

```
$ export RHCOS_OPENSTACK_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.jso
n  | jq .images.openstack.path | sed 's/"//g')
```

7. 获取安装程序要在 bootstrap 虚拟机上部署的 RHCOS 镜像的 URI。

```
$ export RHCOS_QEMU_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.jso
n  | jq .images.qemu.path | sed 's/"//g')
```

8. 获取发布镜像的路径。

```
$ export RHCOS_PATH=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.jso
n | jq .baseURI | sed 's/"//g')
```

9. 获取要在 bootstrap 虚拟机上部署的 RHCOS 镜像的 SHA 哈希。

```
$ export RHCOS_QEMU_SHA_UNCOMPRESSED=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.jso
n  | jq -r '.images.qemu["uncompressed-sha256"]')
```

10. 获取要在节点上部署的 RHCOS 镜像的 SHA 哈希。

```
$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.jso
n  | jq -r '.images.openstack.sha256')
```

11. 下载这些镜像并将其放在 /home/kni/rhcos_image_cache 目录中。

```
$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_QEMU_URI}
$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o
/home/kni/rhcos_image_cache/${RHCOS_OPENSTACK_URI}
```

12. 对于新创建的文件，确认 SELinux 类型为 httpd_sys_content_t。

```
$ ls -Z /home/kni/rhcos_image_cache
```

13. 创建 pod。

```
$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
quay.io/centos7/httpd-24-centos7:latest
```

1.3.6. 配置文件

1.3.6.1. 配置 install-config.yaml 文件

install-config.yaml 文件需要一些额外的信息。大多数信息用于指导安装程序，以便安装的集群有可用硬件的足够信息以可以完全管理它们。

1. 配置 install-config.yaml。更改适当的变量以匹配环境，包括 pullSecret 和 sshKey。

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster-name>
networking:
  machineCIDR: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2 (1)
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api-ip>
    ingressVIP: <wildcard-ip>
    provisioningNetworkInterface: <NIC1>
    provisioningNetworkCIDR: <CIDR>
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (2)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
      - name: <openshift-master-1>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (3)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
      - name: <openshift-master-2>
        role: master
        bmc:
          address: ipmi://<out-of-band-ip> (4)
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
        hardwareProfile: default
      - name: <openshift-worker-0>
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip> (5)
          username: <user>
```

```
      password: <password>
    bootMACAddress: <NIC1-mac-address>
    hardwareProfile: unknown
  - name: <openshift-worker-1>
    role: worker
    bmc:
      address: ipmi://<out-of-band-ip>
      username: <user>
      password: <password>
    bootMACAddress: <NIC1-mac-address>
    hardwareProfile: unknown
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

1 根据作为 OpenShift Container Platform 集群一部分的 worker 节点数量来缩放 worker 机器。

2 3 4 5 如需了解更多选项，请参阅 BMC 寻址部分。

2. 创建用于存储集群配置的目录。

```
$ mkdir ~/clusterconfigs
$ cp install-config.yaml ~/clusterconfigs
```

3. 在安装 OpenShift Container Platform 集群前，请确保关闭所有裸机节点。

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

4. 如果以前的部署尝试中保留任何旧的 bootstrap 资源，则删除旧的 bootstrap 资源。

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

1.3.6.2. 在 install-config.yaml 文件中设置代理设置（可选）

要使用代理部署 OpenShift Container Platform 集群，请对 install-config.yaml 文件进行以下更改。

```
apiVersion: v1
baseDomain: <domain>
proxy:
  httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
  httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
  noProxy: <WILDCARD_OF_DOMAIN>,<PROVISIONING_NETWORK/CIDR>,
<BMC_ADDRESS_RANGE/CIDR>
```

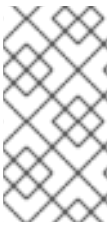
以下是带有值的 noProxy 示例。

```
noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

启用代理后，请在对应的键/值对中设置代理的适当值。

主要考虑：

- 如果代理没有 HTTPS 代理，将 httpsProxy 的值从 https:// 改为 http://。
- 如果使用 provisioning 网络，将其包含在 noProxy 设置中，否则安装程序将失败。
- 将所有代理设置设置为 provisioner 节点中的环境变量。例如：HTTP_PROXY、HTTPS_PROXY 和 NO_PROXY。



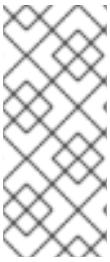
注意

使用 IPv6 置备时，您无法在 noProxy 设置中定义 CIDR 地址块。您必须单独定义每个地址。

1.3.6.3. 针对没有 provisioning 的网络修改 install-config.yaml 文件（可选）

要在没有 provisioning 网络的情况下部署 OpenShift Container Platform 集群，需要对 install-config.yaml 文件进行以下更改。

```
platform:
  baremetal:
    apiVIP: <apiVIP>
    ingressVIP: <ingress/wildcard VIP>
    provisioningNetwork: "Disabled"
    provisioningHostIP: <baremetal_network_IP1>
    bootstrapProvisioningIP: <baremetal_network_IP2>
```



注意

对于 **provisioningHostIP** 和 **bootstrapProvisioningIP** 配置设置，需要提供两个来自 **baremetal** 网络的 IP 地址，并删除 **provisioningBridge** 和 **provisioningNetworkCIDR** 配置设置。

1.3.6.4. 额外的 install-config 参数

以下表格中包括了 **install-config.yaml** 文件所需的参数、**hosts** 参数和 **bmc** 参数。

表 1.1. 所需的参数

参数	默认	Description
baseDomain		集群的域名。例如：**example.com**。
sshKey		**sshKey** 配置设置包含 **~/.ssh/id_rsa.pub** 文件中访问 control plane 节点和 worker 节点所需的密钥。通常，这个密钥来自 **provisioner** 节点。
pullSecret		**pullSecret** 配置设置包含准备置备程序节点时从 Install OpenShift on Bare Metal 页中下载的 pull secret 的副本。
metadata: name:		给 OpenShift Container Platform 集群的名称。例如, **openshift**。
networking: machineCIDR:		外部网络的公共 CIDR（Classless Inter-Domain Routing）。例如：**10.0.0.0/24** .

参数	默认	Description
compute: - name: worker		OpenShift Container Platform 集群需要为 worker（或 compute）节点提供名称，即使没有节点也是如此。
compute: replicas: 2		Replicas 设置 OpenShift Container Platform 集群中的 worker（或 compute）节点的数量。
controlPlane: name: master		OpenShift Container Platform 集群需要一个 control plane（master）节点的名称。
controlPlane: replicas: 3		replicas 设置作为 OpenShift Container Platform 集群一部分的 control plane（master）节点的数量。
provisioningNetworkInterface		连接到 provisioning 网络的 control plane 节点上的网络接口名称。
defaultMachinePlatform		用于没有平台配置的机器池的默认配置。
apiVIP	**api.<clustername.clusterdomain>**	用于内部 API 通信的 VIP。 这个设置必须提供，或者在 DNS 中预先配置，以便正确解析默认名称。
disableCertificateVerification	**False**	**redfish** 和 **redfish-virtualmedia** 需要这个参数来管理 BMC 地址。当 BMC 地址使用自签名证书时，这个值应该是 **True**。
ingressVIP	**test.apps.<clustername.clusterdomain>**	用于入口流量的 VIP。

表 1.2. 可选参数

参数	默认	描述
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	定义 **provisioning** 网络上节点的 IP 范围。

参数	默认	描述
provisioningNetworkCIDR	**172.22.0.0/24**	用于置备的网络的 CIDR。在 **provisioning** 网络中不使用默认地址范围时需要这个选项。
clusterProvisioningIP	**provisioningNetworkCIDR** 的第三个 IP 地址。	运行置备服务的集群中的 IP 地址。默认为 **provisioning** 子网的第三个 IP 地址。例如： **172.22.0.3**。
bootstrapProvisioningIP	**provisioningNetworkCIDR** 的第二个 IP 地址。	在安装程序部署 control plane（master）节点时运行置备服务的 bootstrap 虚拟机上的 IP。默认为 **provisioning** 子网的第二个 IP。例如, **172.22.0.2**。 在没有 **provisioning** 网络时，将此值设置为 **baremetal** 网络中的一个可用的 IP 地址。
externalBridge	**baremetal**	附加到 **baremetal** 网络的 hypervisor**baremetal** 网桥的名称。
provisioningBridge	**provisioning**	附加到 **provisioning** 网络的 **provisioner** 主机上的 **provisioning** 网桥的名称。
defaultMachinePlatform		用于没有平台配置的机器池的默认配置。
bootstrapOSImage		用于覆盖 bootstrap 节点的默认操作系统镜像的 URL。URL 必须包含镜像的 SHA-256 哈希。例如：**https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>**;。
clusterOSImage		用于覆盖集群节点默认操作系统的 URL。URL 必须包含镜像的 SHA-256 哈希。例如，**https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**;。
provisioningNetwork		将这个参数设置为 **Disabled** 以禁用 **provisioning** 网络的要求。用户只能执行基于虚拟介质的置备，或使用协助的安装使集群启用。如果使用电源管理，则需要从机器网络访问 BMC。用户必须在外部网络上提供两个用于置备服务的 IP 地址。将此参数设置为 **managed**（默认参数）来完全管理 provisioning 网络，包括 DHCP、TFTP 等等。 将此参数设置为 **非受管状态**，仍然可启用置备网络，但需要手动配置 DHCP。建议使用虚拟介质置备，但在需要时仍可使用 PXE。
provisioningHostingIp		当 **provisioningNetwork** 配置设置为 **Disabled** 时，把这个参数设为**baremetal** 网络中的一个可用的 IP 地址。
httpProxy		将此参数设置为环境中使用的适当 HTTP 代理。

参数	默认	描述
httpsProxy		将此参数设置为环境中使用的适当 HTTPS 代理。
noProxy		将这个参数设置为适合环境中代理使用的排除项。

Hosts

hosts 参数是用于构建集群的独立裸机资产列表。

名称	默认	Description
name		与详情关联的 **BareMetalHost** 资源的名称。例如, **openshift-master-0**。
role		裸机节点的角色。**master** 或 **worker**。
bmc		基板管理控制器的连接详情。如需了解更多详细信息，请参阅 BMC 寻址部分。
bootMACAddress		主机用来在 **provisioning** 网络中引导的 NIC 的 MAC 地址。

1.3.6.5. BMC 地址

每个 bmc 条目的 address 字段都是连接到 OpenShift Container Platform 集群节点的 URL，包括 URL 方案中的控制器类型以及在网络中的位置。

IPMI

IPMI 主机使用 ipmi://<out-of-band-ip>:<port>，如果没有指定，默认使用端口 623。以下示例演示了 install-config.yaml 文件中的 IPMI 配置。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
```

RedFish for HPE

要启用 RedFish，使用 redfish:// 或 redfish+http:// 禁用 TLS。安装程序需要主机名或者 IP 地址以及到系统 ID 的路径。以下示例演示了 install-config.yaml 文件中的 RedFish 配置。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

虽然建议为带外管理地址提供颁发机构证书，但在使用自签名证书时，您必须在 bmc 配置中包括 disableCertificateVerification: True。以下示例演示了在 install-config.yaml 文件中使用 disableCertificateVerification: True 配置参数的 RedFish 配置。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```

RedFish for Dell

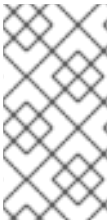
要启用 RedFish，使用 redfish:// 或 redfish+http:// 禁用 TLS。安装程序需要主机名或者 IP 地址以及到系统 ID 的路径。以下示例演示了 install-config.yaml 文件中的 RedFish 配置。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```

虽然建议为带外管理地址提供颁发机构证书，但在使用自签名证书时，您必须在 bmc 配置中包括 disableCertificateVerification: True。以下示例演示了在 install-config.yaml 文件中使用

disableCertificateVerification: True 配置参数的 RedFish 配置。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
          disableCertificateVerification: True
```



注意

目前，只有带有 iDRAC 固件版本 4.20.20.20 或更高版本的 Dell 支持 RedFish，用于裸机部署上的安装程序置备的 OpenShift Container Platform 安装。

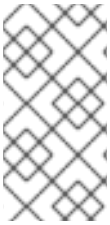
RedFish Virtual Media for HPE

要为 HPE 服务器启用 RedFish Virtual Media，在 address 设置中使用 redfish-virtualmedia://。以下示例演示了在 install-config.yaml 文件中使用 RedFish Virtual Media。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: redfish-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/1
          username: <user>
          password: <password>
```

RedFish Virtual Media for Dell

对于 Dell 服务器中的 RedFish Virtual Media，在 address 设置中使用 idrac-virtualmedia://。

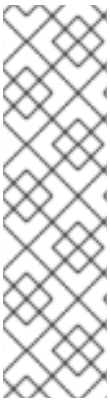


注意

Dell 服务器中的 redfish Virtual Media 在 OpenShift Container Platform 4.6 中有一个已知的问题。4.6.1 点发行版本会解决这个问题。

以下示例演示了在 install-config.yaml 文件中使用 iDRAC Virtual Media。

```
platform:
  baremetal:
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: idrac-virtualmedia://<out-of-band-ip>/redfish/v1/Systems/System.Embedded.1
          username: <user>
          password: <password>
```



注意

idrac-virtualmedia 需要 iDRAC 固件版本 4.20.20.20 或更高版本。

通过 iDRAC 控制台，确保 OpenShift Container Platform 集群节点带有 AutoAttach Enabled。菜单路径是：Configuration→Virtual Media→Attach Mode→AutoAttach.

1.3.6.6. Root device hints

rootDeviceHints 参数使安装程序能够将 Red Hat Enterprise Linux CoreOS（RHCOS）镜像置备到特定设备。安装程序会按照发现设备的顺序检查设备，并将发现的值与 hint 值进行比较。安装程序会使用第一个与 hint 值匹配的发现设备。该配置可包括多个 hint，但只有与所有 hint 都匹配的设备才会被安装程序选择。

表 1.3. 子字段

子字段	描述
deviceName	包含类似 **/dev/vda** 的 Linux 设备名称的字符串。hint 必须与实际值完全匹配。
hctl	包含类似 **0:0:0:0** 的 SCSI 总线地址的字符串。hint 必须与实际值完全匹配。
model	包含特定厂商的设备标识符的字符串。hint 可以是实际值的子字符串。
vendor	包含该设备厂商或制造商名称的字符串。hint 可以是实际值的子字符串。
serialNumber	包含设备序列号的字符串。hint 必须与实际值完全匹配。
minSizeGigabytes	代表设备最小值的一个整数，以 GB 为单位。

子字段	描述
wwn	包含唯一存储标识符的字符串。hint 必须与实际值完全匹配。
wwnWithExtension	包含唯一存储标识符且附加厂商扩展的字符串。hint 必须与实际值完全匹配。
wwnVendorExtension	包含唯一厂商存储标识符的字符串。hint 必须与实际值完全匹配。
rotational	指明该设备为旋转磁盘（true）还是非旋转磁盘（false）的布尔值。

示例用法

```
- name: master-0
  role: master
  bmc:
    address: ipmi://10.10.0.3:6203
    username: admin
    password: redhat
  bootMACAddress: de:ad:be:ef:00:40
  rootDeviceHints:
    deviceName: "/dev/sda"
```

1.3.6.7. 创建 OpenShift Container Platform 清单

1. 创建 OpenShift Container Platform 清单。

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs create manifests
```

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings
WARNING Discarding the OpenShift Manifest that was provided in the target directory because its dependencies are dirty and it needs to be regenerated
```

1.3.7. 创建断开连接的 registry（可选）

在某些情况下,您可能想要使用安装 registry 的本地副本安装 OpenShift KNI 集群。这可能会提高网络效率，因为集群节点位于无法访问互联网的网络中。

一个本地镜像的 registry 副本需要以下内容：

- registry 节点的证书。这可以是自签名证书。
- 一个 webserver - 这由系统中的容器提供。
- 一个更新的 pull secret，其中包含证书和本地存储库信息。

注意

在 registry 节点上创建断开连接的 registry 是可选的。在后续的小节中可以看到它们是可选的，只有在 registry 节点上创建断开连接的 registry 时才需要执行相关的步骤。在 registry 节点上创建断开连接的 registry 时，您应该执行标记为"（可选）" 的子章节。

1.3.7.1. 准备 registry 节点以托管已镜像的 registry（可选）

对 registry 节点进行以下更改。

流程

1. 打开 registry 节点上的防火墙端口。

```
$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
$ sudo firewall-cmd --reload
```

2. 为 registry 节点安装所需的软件包。

```
$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. 创建保存存储库信息的目录结构。

```
$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

1.3.7.2. 生成自签名证书（可选）

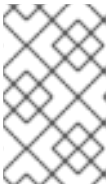
为 registry 节点生成自签名证书，并将其放在 /opt/registry/certs 目录中。

流程

1. 根据情况调整证书信息。

```
$ host_fqdn=$( hostname --long )
$ cert_c="<Country Name>"   # Country Name (C, 2 letter code)
$ cert_s="<State>"          # Certificate State (S)
$ cert_l="<Locality>"       # Certificate Locality (L)
$ cert_o="<Organization>"   # Certificate Organization (O)
$ cert_ou="<Org Unit>"      # Certificate Organizational Unit (OU)
$ cert_cn="${host_fqdn}"    # Certificate Common Name (CN)

$ openssl req \
    -newkey rsa:4096 \
    -nodes \
    -sha256 \
    -keyout /opt/registry/certs/domain.key \
    -x509 \
    -days 365 \
    -out /opt/registry/certs/domain.crt \
    -addext "subjectAltName = DNS:${host_fqdn}" \
    -subj
"/C=${cert_c}/ST=${cert_s}/L=${cert_l}/O=${cert_o}/OU=${cert_ou}/CN=${cert_cn}"
```



注意

当替换 <Country Name> 时，请确保它只包含两个字母。例如，US。

2. 使用新证书更新 registry 节点的 ca-trust。

```
$ sudo cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

1.3.7.3. 创建 registry podman 容器（可选）

registry 容器使用 /opt/registry 目录来保存证书、认证文件以及存储其数据文件。

registry 容器使用 httpd，并需要 htpasswd 文件进行身份验证。

流程

1. 在 /opt/registry/auth 中创建一个 htpasswd 文件供容器使用。

```
$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

将 <user> 替换为用户名，将 <passwd> 替换为密码。

2. 创建并启动 registry 容器。

```
$ podman create \
  --name ocpdiscon-registry \
  -p 5000:5000 \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" \
  -e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
  -e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
  -e "REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true" \
  -v /opt/registry/data:/var/lib/registry:z \
  -v /opt/registry/auth:/auth:z \
  -v /opt/registry/certs:/certs:z \
  docker.io/library/registry:2
```

```
$ podman start ocpdiscon-registry
```

1.3.7.4. 复制和更新 pull-secret（可选）

将 pull secret 文件从置备程序节点复制到 registry 节点，并修改该文件，使其包含新 registry 节点的身份验证信息。

流程

1. 复制 pull-secret.txt 文件。

```
$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. 使用 registry 节点的完全限定域名更新 host_fqdn 环境变量。

```
$ host_fqdn=$( hostname --long )
```

3. 使用用于创建 htpasswd 文件的 http 凭证的 base64 编码来更新 b64auth 环境变量。

```
$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

将 <username> 替换为用户名，将 <passwd> 替换为密码。

4. 设置 AUTHSTRING 环境变量使用 base64 授权字符串。$USER 变量是包含当前用户名称的环境变量。

```
$ AUTHSTRING="{\"$host_fqdn:5000\": {\"auth\": \"$b64auth\",\"email\":
\"$USER@redhat.com\"}}"
```

5. 更新 pull-secret.txt 文件。

```
$ jq ".auths += $AUTHSTRING" < pull-secret.txt > pull-secret-update.txt
```

1.3.7.5. 对存储库进行镜像（可选）

流程

1. 将 provisioner 节点中的 oc 二进制文件复制到 registry 节点。

```
$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. 设置所需的环境变量。

   a. 设置发行版本：

```
$ VERSION=<release_version>
```

对于 <release_version>，请指定与 OpenShift Container Platform 版本对应的标签，用于安装，如 4.6。

b. 设置本地 registry 名称和主机端口：

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

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

c. 设置本地存储库名称：

```
$ LOCAL_REPO='<local_repository_name>'
```

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

3. 将远程安装镜像镜像(mirror)到本地存储库。

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.txt \
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

1.3.7.6. 修改 install-config.yaml 文件，使用断开连接的 registry（可选）

在 provisioner 节点上，install-config.yaml 文件应该使用从 pull-secret-update.txt 文件中新创建的 pull-secret。install-config.yaml 文件还必须包含断开连接的 registry 节点的证书和 registry 信息。

流程

1. 将断开连接的 registry 节点的证书添加到 install-config.yaml 文件中。证书应该跟着 "additionalTrustBundle: |" 行，并带有正确的缩进（通常是两个空格）。

```
$ echo "additionalTrustBundle: |" >> install-config.yaml
$ sed -e 's/^/  /' /opt/registry/certs/domain.crt >> install-config.yaml
```

2. 将 registry 的镜像信息添加到 install-config.yaml 文件中。

```
$ echo "imageContentSources:" >> install-config.yaml
```

```
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-config.yaml
$ echo "- mirrors:" >> install-config.yaml
$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-config.yaml
$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >> install-
config.yaml
```

注意

将 registry.example.com 替换为 registry 的完全限定域名。

1.3.8. 在 worker 节点上部署路由器

在安装过程中，安装程序会在 worker 节点上部署路由器 Pod。默认情况下，安装程序会安装两个路由器 Pod。如果初始集群只有一个 worker 节点，或者如果部署的集群需要额外的路由器来处理用于 OpenShift Container Platform 集群中服务的外部流量负载，您可以创建一个 yaml 文件来设置适当数量的路由器副本。

注意

默认情况下，安装程序会部署两个路由器。如果集群至少有两个 worker 节点，您可以跳过此部分。如需有关 Ingress Operator 的更多信息，请参阅 OpenShift Container Platform 中的 Ingress Operator。

注意

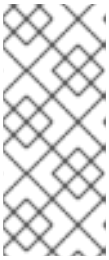
如果集群没有 worker 节点，安装程序默认会在 control plane 节点上部署两个路由器。如果集群没有 worker 节点，可以跳过本节。

流程

1. 创建一个 router-replicas.yaml 文件。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: <num-of-router-pods>
  endpointPublishingStrategy:
    type: HostNetwork
  nodePlacement:
```

```
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/worker: ""
```



注意

将 <num-of-router-pods> 替换为适当的值。如果只使用一个 worker 节点，将 replicas: 设置为 1。如果使用 3 个以上 worker 节点，您可以根据情况增加 replicas: 的默认值（2）。

2. 将 router-replicas.yaml 文件保存并复制到 clusterconfigs/openshift 目录中。

```
cp ~/router-replicas.yaml clusterconfigs/openshift/99_router-replicas.yaml
```

1.3.9. 安装的验证清单

❑ 已检索到 OpenShift Container Platform 安装程序。

❑ 已展开 OpenShift Container Platform 安装程序。

❑ 已配置了 install-config.yaml 所需的参数。

❑ 已为 install-config.yaml 配置了 hosts 参数。

❑ 已配置 install-config.yaml 的 bmc 参数。

❑ 在 bmc 的 address 字段中配置的值已被应用。

❑ 创建断开连接的 registry（可选）。

❑ （可选）如果使用，断开连接的 registry 设置。

- ❑ （可选）在 worker 节点上部署了路由。

1.3.10. 通过 OpenShift Container Platform 安装程序部署集群

运行 OpenShift Container Platform 安装程序：

```
$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create cluster
```

1.3.11. 安装后

在部署过程中，您可以通过对安装目录文件夹中的 .openshift_install.log 日志文件发出 tail 命令来检查安装的整体状态。

```
$ tail -f /path/to/install-dir/.openshift_install.log
```

1.3.12. 准备在裸机上重新安装集群

在在裸机上重新安装集群前，您必须执行清理操作。

流程

1. 删除或重新格式化 bootstrap、control plane（也称为 master）节点和 worker 节点的磁盘。如果您在虚拟机监控程序环境中工作，您必须添加您要删除的任何磁盘。

2. 删除之前生成的工件：

```
$ cd ; /bin/rm -rf auth/ bootstrap.ign master.ign worker.ign metadata.json \
.openshift_install.log .openshift_install_state.json
```

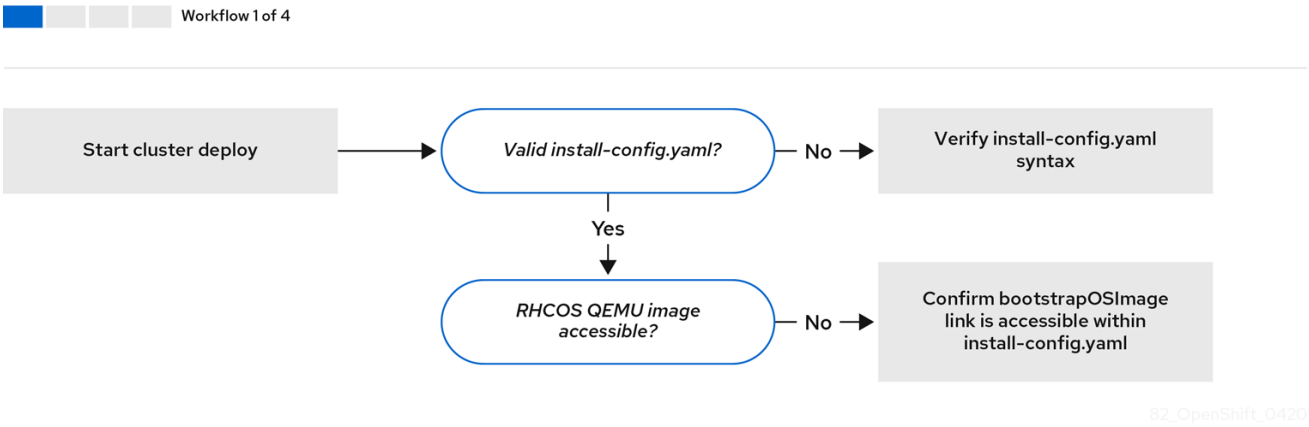
3. 生成新清单和 Ignition 配置文件。如需更多信息，请参阅"创建 Kubernetes 清单和 Ignition 配置文件"。

4. 将安装程序创建的新 bootstrap、control plane 和计算节点 Ignition 配置文件上传到 HTTP 服务器。这将覆盖以前的 Ignition 文件。

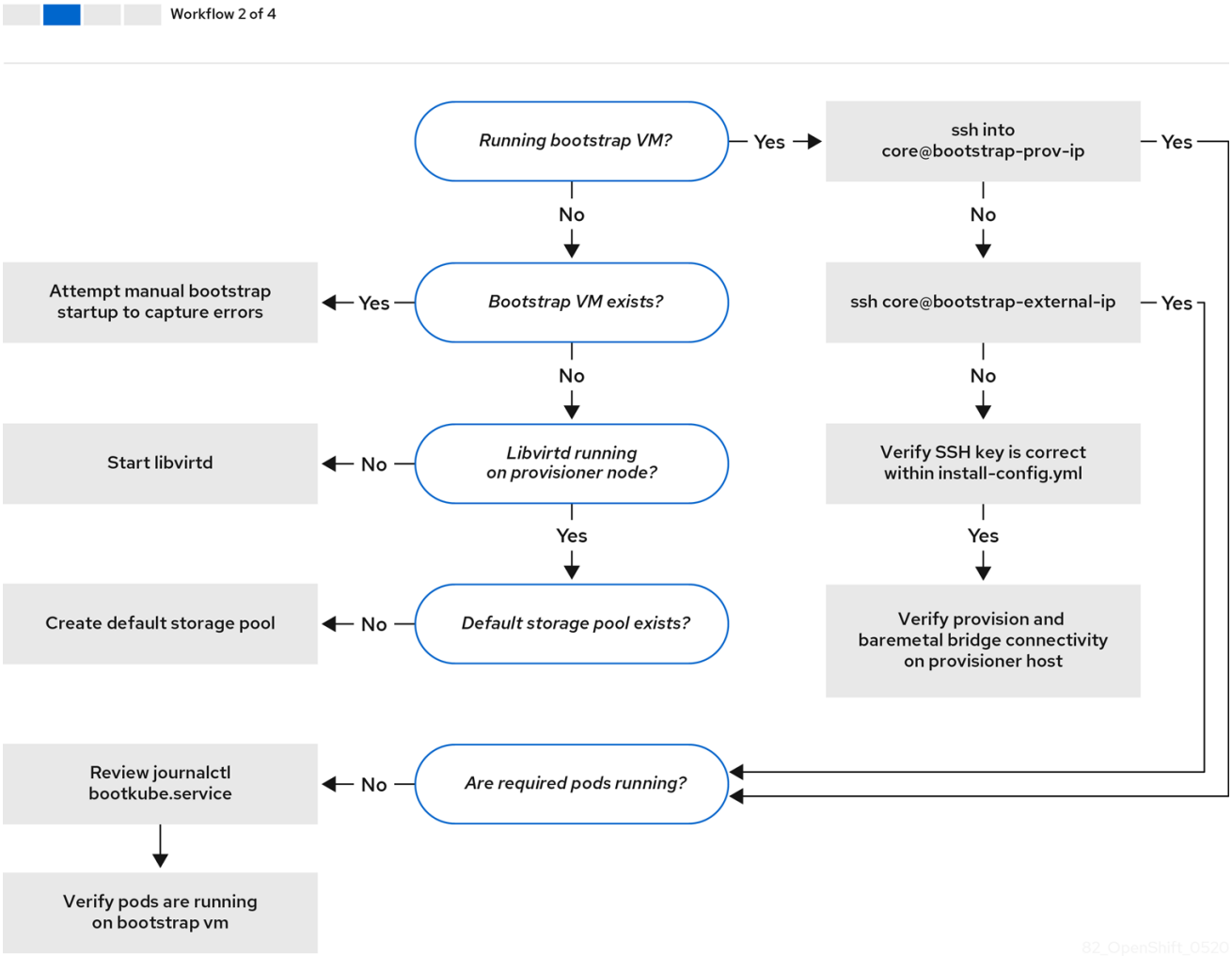
1.4. 故障排除

1.4.1. 安装程序工作流故障排除

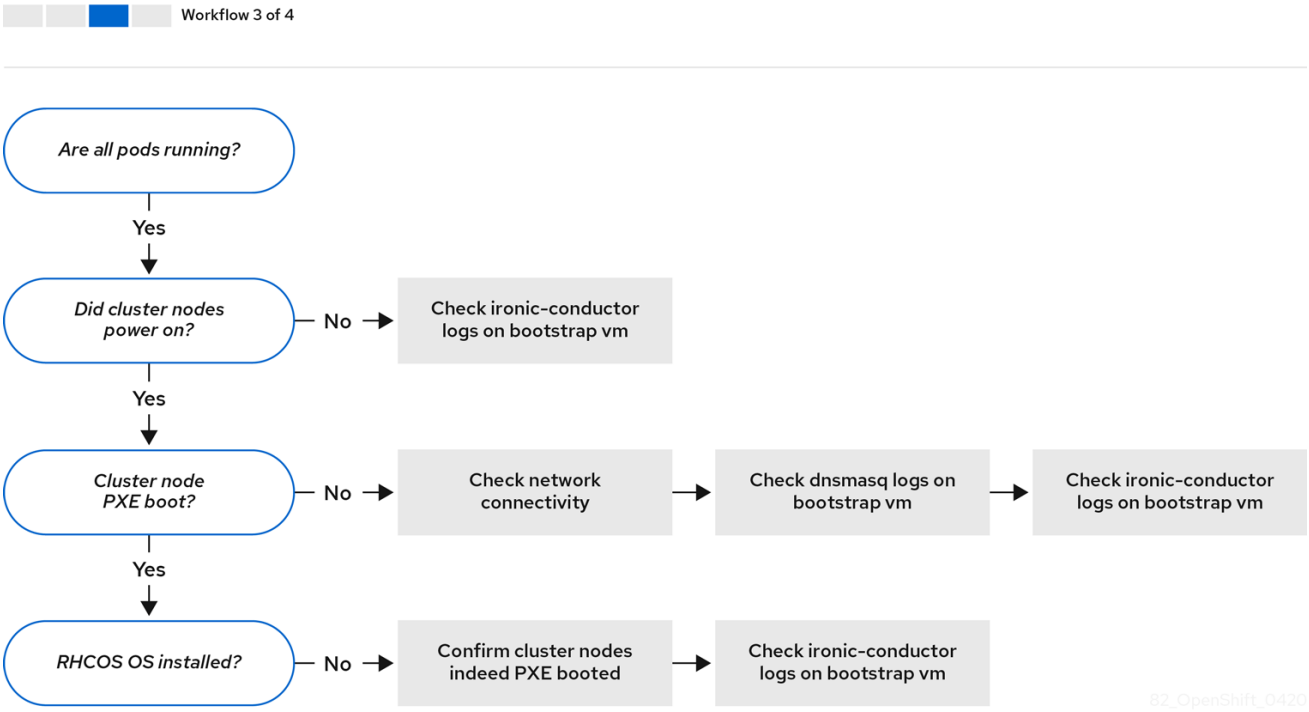
在对安装环境进行故障排除之前，了解裸机上安装程序置备安装的整体流至关重要。下面的图表提供了故障排除流程，并按部就班地划分环境。



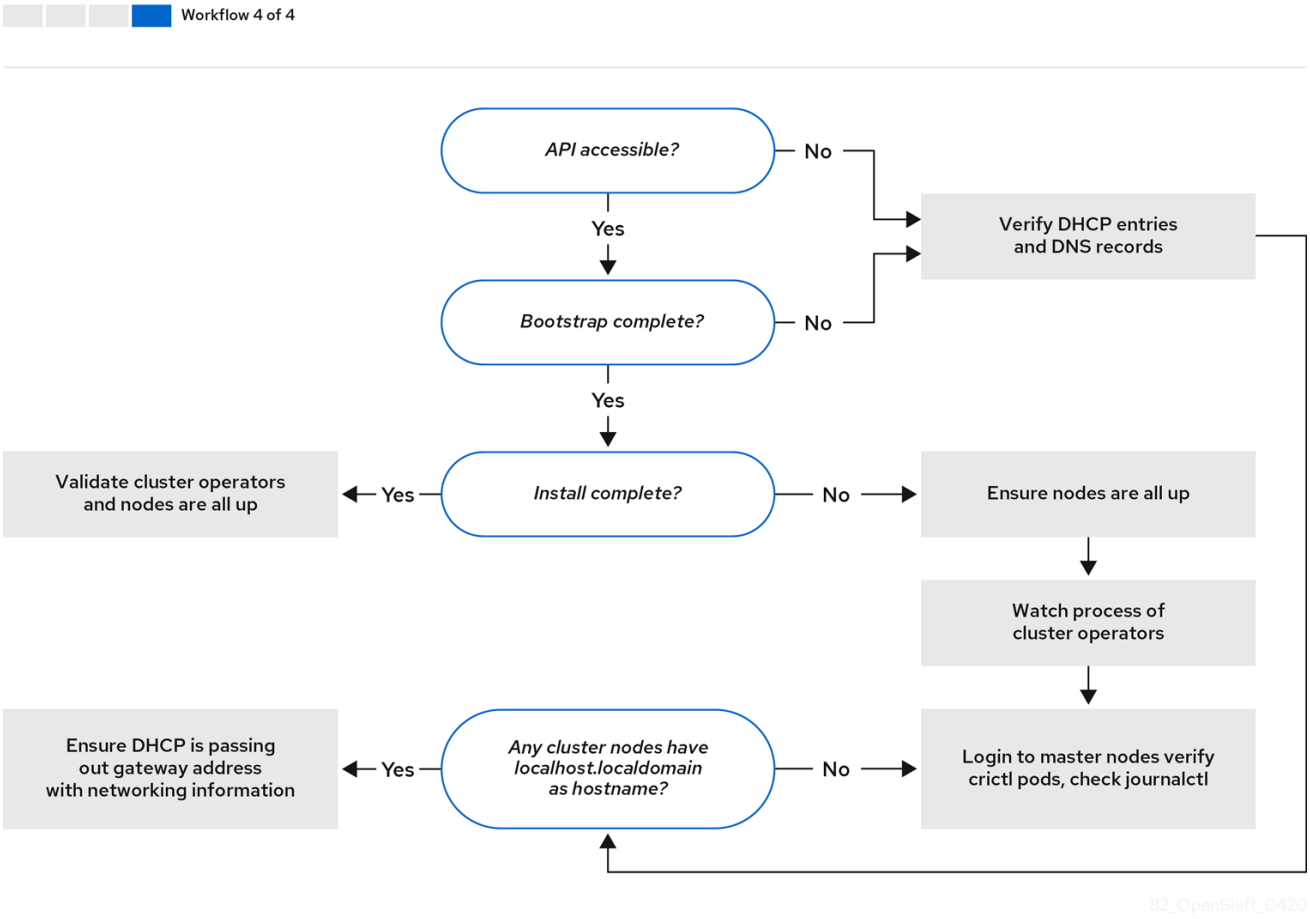
当 install-config.yaml 文件出错或者无法访问 Red Hat Enterprise Linux CoreOS（RHCOS）镜像时，*工作流 1（共 4 步）* 的工作流演示了故障排除工作流。故障排除建议可在 故障排除 install-config.yaml 中找到。



工作流 2（共 4 步） **描述了对 bootstrap 虚拟机问题、 无法引导集群节点的 bootstrap 虚拟机 以及检查日志 的故障排除工作流。当在没有 provisioning 网络的情况下安装 OpenShift Container Platform 集群时，这个工作流不适用。**



*工作流 3（共 4 步）*描述了 没有 PXE 引导的集群节点的故障排除工作流。



工作流 4（共 4 步） 演示了 无法访问的 API 的故障、验证的安装 的故障排除工作流。

1.4.2. install-config.yaml 故障排除

install-config.yaml 配置文件代表作为 OpenShift Container Platform 集群一部分的所有节点。该文件包含由 apiVersion、baseDomain、imageContentSources 和虚拟 IP 地址组成的必要选项。如果在 OpenShift Container Platform 集群部署早期发生错误，则 install-config.yaml 配置文件中可能会出现错误。

流程

1. 使用 YAML-tips 中的指南。
2. 使用 syntax-check 来验证 YAML 语法是否正确。
3. 验证 Red Hat Enterprise Linux CoreOS（RHCOS）QEMU 镜像是否已正确定义，并可以通过 install-config.yaml 提供的 URL 访问。例如：

```
$ curl -s -o /dev/null -I -w "%{http_code}\n" http://webserver.example.com:8080/rhcos-44.81.202004250133-0-qemu.x86_64.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636fbd0f1ce7
```

如果输出为 200，则代表从存储 bootstrap 虚拟机镜像的 webserver 返回了有效的响应。

1.4.3. Bootstrap 虚拟机问题

OpenShift Container Platform 安装程序生成 bootstrap 节点虚拟机，该虚拟机处理置备 OpenShift Container Platform 集群节点。

流程

1. 触发安装程序后约 10 到 15 分钟，使用 virsh 命令检查 bootstrap 虚拟机是否可正常工作：

```
$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```

注意

bootstrap 虚拟机的名称始终是集群名称再加上一组随机字符，并以"bootstrap"结尾。

如果 bootstrap 虚拟机在 10 到 15 分钟后还没有运行，请检查其没有运行的原因。可能的问题包括：

2. 确定在该系统中运行了 libvirtd：

```
$ systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
     Docs: man:libvirtd(8)
           https://libvirt.org
```

```
Main PID: 9850 (libvirtd)
  Tasks: 20 (limit: 32768)
 Memory: 74.8M
 CGroup: /system.slice/libvirtd.service
         ├─ 9850 /usr/sbin/libvirtd
```

如果 bootstrap 虚拟机可以正常工作，登录到它。

3. 使用 virsh console 命令查找 bootstrap 虚拟机的 IP 地址：

```
$ sudo virsh console example.com
```

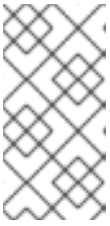
```
Connected to domain example.com
Escape character is ^]

Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg
(ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnlio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```

重要

当在没有 provisioning 网络的情况下部署 OpenShift Container Platform 集群时，您必须使用公共 IP 地址，而不是像 172.22.0.2 这样的私有 IP 地址。

4. 获取 IP 地址后，使用 ssh 命令登录到 bootstrap 虚拟机：



注意

在上一步的控制台输出中，您可以使用 ens3 提供的 IPv6 IP 地址或 ens4 提供的 IPv4 IP。

```
$ ssh core@172.22.0.2
```

如果您无法成功登录到 bootstrap 虚拟机，您可能会遇到以下情况之一：

-

无法访问 172.22.0.0/24 网络。验证 provisioner 主机上的网络连接，特别是 provisioner 网桥上的连接。如果您不使用 provisioning 网络，则不会有这个问题。

- 您无法通过公共网络访问 bootstrap 虚拟机。当尝试通过 baremetal 网络进行 SSH 时，验证 provisioner 主机的连接，特别是 baremetal 网桥的连接。

- 存在 Permission denied (publickey,password,keyboard-interactive) 问题。当尝试访问 bootstrap 虚拟机时，可能会出现 Permission denied 错误。验证试图登录到虚拟机的用户的 SSH 密钥是否在 install-config.yaml 文件中设置。

1.4.3.1. Bootstrap 虚拟机无法引导集群节点

在部署期间，bootstrap 虚拟机可能无法引导集群节点，这会阻止虚拟机使用 RHCOS 镜像置备节点。这可能是因为以下原因：

- install-config.yaml 文件有问题。

- 通过裸机网络进行带外网络访问的问题。

要验证这个问题，有三个与 ironic 相关的容器：

- ironic-api

- ironic-conductor

- ironic-inspector

流程

1. 登录到 bootstrap 虚拟机：

```
$ ssh core@172.22.0.2
```

2.

2. 要检查容器日志，请执行以下操作：

```
[core@localhost ~]$ sudo podman logs -f <container-name>
```

将 <container-name> 替换为 ironic-api、ironic-conductor 或 ironic-inspector 之一。如果您遇到 control plane 节点没有通过 PXE 引导的问题，请检查 ironic-conductor pod。ironic-conductor pod 包含了有关尝试引导集群节点的最详细信息，因为它尝试通过 IPMI 登录该节点。

潜在原因

集群节点在部署启动时可能处于 ON 状态。

解决方案

在通过 IPMI 开始安装前关闭 OpenShift Container Platform 集群节点：

```
$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip> power off
```

1.4.3.2. 检查日志

在下载或访问 RHCOS 镜像时，首先请验证 install-config.yaml 配置文件中的 URL 是否正确。

内部 webserver 托管 RHCOS 镜像的示例

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-qemu.x86_64.qcow2.gz?
sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-openstack.x86_64.qcow2.gz?
sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e811abb78fe3b0
```

ipa-downloader 和 coreos-downloader 容器从 Webserver 或外部 quay.io registry 下载资源（由 install-config.yaml 配置文件中指定的为准）。验证以下两个容器是否正在运行，并根据需要检查其日志：

- ipa-downloader

- coreos-downloader

流程

1. 登录到 bootstrap 虚拟机：

```
$ ssh core@172.22.0.2
```

2. 检查 bootstrap 虚拟机中的 ipa-downloader 和 coreos-downloader 容器的状态：

```
[core@localhost ~]$ sudo podman logs -f ipa-downloader
```

```
[core@localhost ~]$ sudo podman logs -f coreos-downloader
```

如果 bootstrap 虚拟机无法访问镜像的 URL，使用 curl 命令验证虚拟机能否访问镜像。

3. 要检查 bootkube 日志以了解在部署阶段是否启动了所有容器，请执行以下操作：

```
[core@localhost ~]$ journalctl -xe
```

```
[core@localhost ~]$ journalctl -b -f -u bootkube.service
```

4. 验证所有 pod 都在运行，包括 dnsmasq、mariadb、httpd 和 ironic：

```
[core@localhost ~]$ sudo podman ps
```

5. 如果 pod 存在问题，请检查相关的容器日志。要检查 ironic-api 的日志，请执行以下操作：

```
[core@localhost ~]$ sudo podman logs <ironic-api>
```

1.4.4. 集群节点不能 PXE 引导

当 OpenShift Container Platform 集群节点无法 PXE 引导时，在不能 PXE 引导的集群节点上执行以下检查。如果没有 provisioning 网络，则在安装 OpenShift Container Platform 集群时不适用此步骤。

流程

1. 检查到 **provisioning** 网络的网络连接。

2. 确保 **provisioning** 网络的 **NIC** 上启用了 **PXE**，并且其他所有 **NIC** 都禁用了 **PXE**。

3. 验证 **install-config.yaml** 配置文件是否有正确的硬件配置集，以及连接到 **provisioning** 网络的 **NIC** 的引导 **MAC** 地址。例如：

control plane 节点设置

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: default          #control plane node settings
```

Worker 节点设置

```
bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
hardwareProfile: unknown          #worker node settings
```

1.4.5. API 无法访问

当集群正在运行且客户端无法访问 **API** 时，可能是因为域名解析的问题影响对 **API** 的访问。

流程

1. 主机名解析：检查集群节点带有完全限定域名，而不只是 **localhost.localdomain**。例如：

```
$ hostname
```

如果没有设定主机名，请设置正确的主机名。例如：

```
$ hostnamectl set-hostname <hostname>
```

2. 错误的名称解析：使用 dig 和 nslookup 检查每个节点在 DNS 服务器中是否可以被正确解析。例如：

```
$ dig api.<cluster-name>.example.com
```

```
; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster-name>.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
;; QUESTION SECTION:
;api.<cluster-name>.example.com. IN A

;; ANSWER SECTION:
api.<cluster-name>.example.com. 10800 IN A 10.19.13.86

;; AUTHORITY SECTION:
<cluster-name>.example.com. 10800 IN NS <cluster-name>.example.com.

;; ADDITIONAL SECTION:
<cluster-name>.example.com. 10800 IN A 10.19.14.247

;; Query time: 0 msec
;; SERVER: 10.19.14.247#53(10.19.14.247)
;; WHEN: Tue May 19 20:30:59 UTC 2020
;; MSG SIZE  rcvd: 140
```

示例中的输出显示 api.<cluster-name>.example.com VIP 的适当 IP 地址为 10.19.13.86。这个 IP 地址应该位于 baremetal 网络中。

1.4.6. 清理以前的安装

如果上一个部署失败，在尝试再次部署 OpenShift Container Platform 前从失败的尝试中删除工件。

流程

1. 在安装 OpenShift Container Platform 集群前关闭所有裸机节点：

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

2. 删除以前部署中保留的旧的 bootstrap 资源：

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

3. 从 clusterconfigs 目录中删除以下内容以防止 Terraform 失败：

```
$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform* ~/clusterconfigs/tls
~/clusterconfigs/metadata.json
```

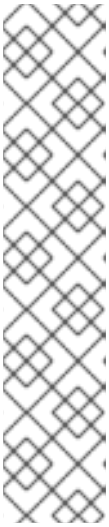
1.4.7. 创建 registry 的问题

在创建断开连接的 registry 时，在尝试对 registry 进行镜像时，可能会遇到 "User Not Authorized" 错误。如果您没有在现有的 pull-secret.txt 文件中添加新身份验证，则可能会出现这个错误。

流程

1. 检查以确保身份验证成功：

```
$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```



注意

用于镜像安装镜像的变量输出示例：

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

在为 OpenShift 安装设置环境章节中的获取 OpenShift 安装程序步骤中，设置了 RELEASE_IMAGE 和 VERSION 值。

2. 镜像 registry 后，确认您可以在断开连接的环境中访问它：

```
$ curl -k -u <user>:<password> https://registry.example.com:<registry-port>/v2/_catalog
{"repositories":["<Repo-Name>"]}
```

1.4.8. 其它问题

1.4.8.1. 解决 runtime network not ready 错误

部署集群后您可能会收到以下错误：

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady message:Network plugin returns error: Missing CNI default network`
```

Cluster Network Operator 负责部署网络组件以响应安装程序创建的特殊对象。它会在安装过程的早期阶段运行（在 Control Plane（master）节点启动后，bootstrap control plane 被停止前运行）。它可能会显示更细微的安装程序问题，比如启动 Control Plane（master）节点时延迟时间过长，或者 apiserver 通讯的问题。

流程

1. 检查 openshift-network-operator 命名空间中的 pod：

```
$ oc get all -n openshift-network-operator
```

```
NAME                                    READY STATUS              RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating   0          149m
```

2. 在 provisioner 节点上，确定存在网络配置：

```
$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
```

如果不存在，代表安装程序没有创建它。要确定安装程序没有创建它的原因，请执行以下操作：

```
$ openshift-install create manifests
```

3. 检查 network-operator 是否正在运行：

```
$ kubectl -n openshift-network-operator get pods
```

4. 检索日志：

```
$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

在具有三个或更多 Control Plane(master)节点的高可用性集群上，Operator 将执行领导选举机制，所有其他 Operator 会休眠。如需了解更多详细信息，请参阅 故障排除。

1.4.8.2. 集群节点没有通过 DHCP 获得正确的 IPv6 地址

如果集群节点没有通过 DHCP 获得正确的 IPv6 地址，请检查以下内容：

1. 确定保留的 IPv6 地址不在 DHCP 范围之外。

2. 在 DHCP 服务器的 IP 地址保留中，确保保留指定了正确的 DHCP 唯一识别符（DUID）。

例如：

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and
'18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. 确保路由声明（Route Announcement）正在正常工作。

4. 确定 DHCP 服务器正在侦听提供 IP 地址范围所需的接口。

1.4.8.3. 集群节点没有通过 DHCP 获得正确的主机名

在 IPv6 部署过程中，集群节点必须通过 DHCP 获得其主机名。有时 NetworkManager 不会立即分配主机名。Control Plane（master）节点可能会报告错误，例如：

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

这个错误表示集群节点可能在没有从 DHCP 服务器收到主机名的情况下引导，这会导致 kubelet 使用 localhost.localdomain 主机名引导。要解决这个问题，强制节点更新主机名。

流程

1. 检索 主机名：

```
[core@master-X ~]$ hostname
```

如果主机名是 localhost，请执行以下步骤。

注意

其中 X 是 control plane 节点（也称为 master 节点）号。

2. 强制集群节点续订 DHCP 租期：

```
[core@master-X ~]$ sudo nmcli con up "<bare-metal-nic>"
```

将 <bare-metal-nic> 替换为与 baremetal 网络对应的有线连接。

3. 再次检查 主机名：

```
[core@master-X ~]$ hostname
```

4. 如果主机名仍然是 localhost.localdomain，重启 NetworkManager：

```
[core@master-X ~]$ sudo systemctl restart NetworkManager
```

5. 如果主机名仍然是 localhost.localdomain，请等待几分钟并再次检查。如果主机名还是 localhost.localdomain，重复前面的步骤。

6. 重启 nodeip-configuration 服务：

```
[core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
```

此服务将使用正确的主机名引用来重新配置 kubelet 服务。

7. 因为 kubelet 在上一步中有所改变，所以重新加载单元文件定义：

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. 重启 kubelet 服务：

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. 确保 kubelet 使用正确的主机名引导：

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

如果集群节点在启动并运行集群后没有通过 DHCP 获得正确的主机名（例如在重启过程中），集群将

会有一个待处理的 csr。不要批准 csr，否则可能会出现其他问题。

处理 csr

1. 在集群上获取 CSR:

```
$ oc get csr
```

2. 验证待处理的 csr 是否包含 Subject Name: localhost.localdomain:

```
$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 --decode | openssl req -noout -text
```

3. 删除包含 Subject Name: localhost.localdomain 的任何 csr:

```
$ oc delete csr <wrong_csr>
```

1.4.8.4. 路由无法访问端点

在安装过程中，可能会遇到虚拟路由器冗余协议（VRRP）冲突。如果以前使用一个特定集群名称部署的集群中的个 OpenShift Container Platform 节点仍在运行，且这个节点不是使用相同集群名称部署的当前 OpenShift Container Platform 集群的一部分，则可能会出现冲突。例如，一个集群使用集群名称 openshift 部署，它部署了三个 Control Plane（master）节点和三个 worker 节点。之后，一个单独的安装会使用相同的集群名称 openshift，但这个重新部署只安装了三个 control plane(master)节点，以前部署的三个 worker 节点处于 ON 状态。这可能导致 Virtual Router Identifier（VRID）冲突和 VRRP 冲突。

1. 获取路由：

```
$ oc get route oauth-openshift
```

2. 检查服务端点：

```
$ oc get svc oauth-openshift
```

```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. 尝试从 Control Plane（master）节点访问该服务：

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. 找到 provisioner 节点的 authentication-operator 错误：

```
$ oc logs deployment/authentication-operator -n openshift-authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-5fd3c0459d98", APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason: 'OperatorStatusChanged' Status for clusteroperator/authentication changed: Degraded message changed from "IngressStateEndpointsDegraded: All 2 endpoints for oauth-server are reporting"
```

解决方案

1. 确保每个部署的集群名称都是唯一的，确保没有冲突。

2. 关闭所有不是使用相同集群名称的集群部署的一部分的节点。否则，OpenShift Container Platform 集群的身份验证 pod 可能无法成功启动。

1.4.8.5. 在 Firstboot 过程中 Ignition 失败

在 Firstboot 过程中，Ignition 配置可能会失败。

流程

1. 连接到 Ignition 配置失败的节点：

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. 重启 machine-config-daemon-firstboot 服务：

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

1.4.8.6. NTP 没有同步

OpenShift Container Platform 集群的部署需要集群节点间的 NTP 时钟已同步。如果没有同步时钟，当时间差大于 2 秒时，部署可能会因为时钟偏移而失败。

流程

1. 检查集群节点的 AGE 的不同。例如：

```
$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.16.2
master-1.cloud.example.com   Ready    master   135m   v1.16.2
master-2.cloud.example.com   Ready    master   145m   v1.16.2
worker-2.cloud.example.com   Ready    worker   100m   v1.16.2
```

2. 检查因为时钟偏移导致的时间延迟。例如：

```
$ oc get bmh -n openshift-machine-api
```

```
master-1   error registering master-1  ipmi://<out-of-band-ip>
```

```
$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
           Universal time: Tue 2020-03-10 18:20:02 UTC
                 RTC time: Tue 2020-03-10 18:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

处理现有集群中的时钟偏移

1.

1. 创建 chrony.conf 文件并将其编码为 base64 字符串。例如：

```
$ cat << EOF | base 64
server <NTP-server> iburst 1
stratumweight 0
driftfile /var/lib/chrony/drift
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony
EOF
```

1 将 <NTP-server> 替换为 NTP 服务器的 IP 地址。复制输出。

```
[text-in-base-64]
```

2. 创建 MachineConfig 对象，将 base64 字符串替换为上一步输出中生成的 [text-in-base-64] 字符串。以下示例将文件添加到 Control Plane（master）节点。您可以修改 worker 节点的文件，或为 worker 角色创建额外的机器配置。

```
$ cat << EOF > ./99_masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  creationTimestamp: null
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-etc-chrony-conf
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 3.1.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
```

```
      source: data:text/plain;charset=utf-8;base64,[text-in-base-64] 1
    group:
      name: root
    mode: 420
    overwrite: true
    path: /etc/chrony.conf
    user:
      name: root
oslmageURL: ""
```

1 将 [text-in-base-64] 替换为 base64 字符串。

3. 对配置文件做一个副本备份。例如：

```
$ cp 99_masters-chrony-configuration.yaml 99_masters-chrony-
configuration.yaml.backup
```

4. 应用配置文件：

```
$ oc apply -f ./masters-chrony-configuration.yaml
```

5. 确定 System clock synchronized 的值为 yes：

```
$ sudo timedatectl
```

```
           Local time: Tue 2020-03-10 19:10:02 UTC
       Universal time: Tue 2020-03-10 19:10:02 UTC
             RTC time: Tue 2020-03-10 19:36:53
            Time zone: UTC (UTC, +0000)
System clock synchronized: yes
          NTP service: active
      RTC in local TZ: no
```

要在部署前设置时钟同步，请生成清单文件并将该文件添加到 openshift 目录中。例如：

```
$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-
configuration.yaml
```

然后继续创建集群。

1.4.9. 检查安装

安装后，请确保安装程序成功部署节点和 Pod。

流程

1. 当正确安装 OpenShift Container Platform 集群节点时，在 STATUS 栏中会显示 Ready：

```
$ oc get nodes
```

```
NAME                 STATUS   ROLES          AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.16.2
master-1.example.com   Ready    master,worker   4h   v1.16.2
master-2.example.com   Ready    master,worker   4h   v1.16.2
```

2. 确认安装程序已成功部署所有 Pod。以下命令的输出中会排除仍在运行或已完成的 pod。

```
$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```