OpenShift Container Platform 4.6

Deploying installer-provisioned clusters on bare metal

Installing IPI OpenShift Container Platform bare metal clusters

Last Updated: 2020-11-30

OpenShift Container Platform 4.6 Deploying installer-provisioned clusters on bare metal

Installing IPI OpenShift Container Platform bare metal clusters

Abstract

This document provides instructions for installing OpenShift Container Platform clusters on bare metal infrastructure with installer-provisioned clusters.

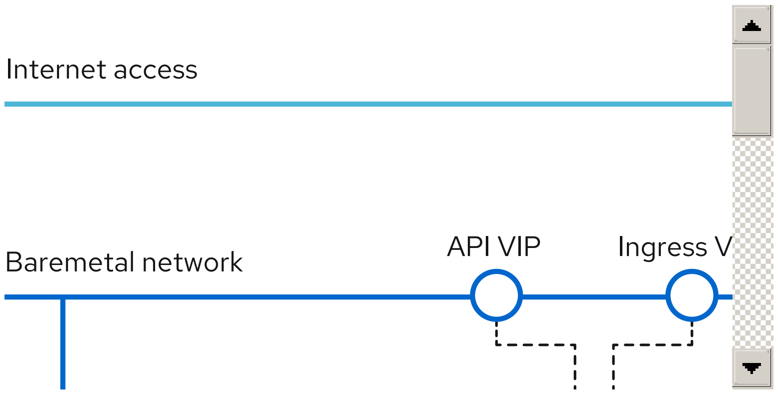
Table of Contents

CHAPTER 1. DEPLOYING INSTALLER-PROVISIONED CLUSTERS ON BARE METAL

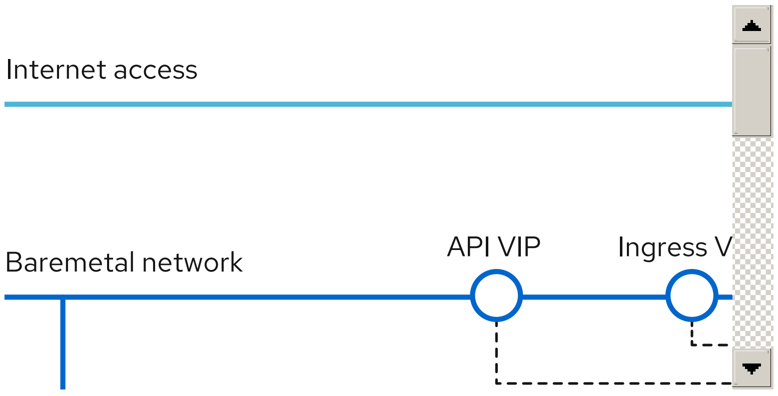
1.1. OVERVIEW

Installer-provisioned installation provides support for installing OpenShift Container Platform on bare metal nodes. This guide provides a methodology to achieving a successful installation.

During installer-provisioned installation on bare metal, the installer on the bare metal node labeled as **provisioner** creates a bootstrap VM. The role of the bootstrap VM is to assist in the process of deploying an OpenShift Container Platform cluster. The bootstrap VM connects to the **baremetal** network and to the **provisioning** network, if present, via the network bridges.



When the installation of OpenShift Container Platform control plane nodes is complete and fully operational, the installer destroys the bootstrap VM automatically and moves the virtual IP addresses (VIPs) to the appropriate nodes accordingly. The API VIP moves to the control plane nodes and the Ingress VIP moves to the worker nodes.



1.2. PREREQUISITES

Installer-provisioned installation of OpenShift Container Platform requires:

1. One provisioner node with RHEL 8.1 installed.
2. Three Control Plane nodes.
3. Baseboard Management Controller (BMC) access to each node.
4. At least one network:
    a. One **required** routable network

b. One **optional** network for provisioning nodes; and,

c. One **optional** management network.

Before starting an installer-provisioned installation of OpenShift Container Platform, ensure the hardware environment meets the following requirements.

1.2.1. Node requirements

Installer-provisioned installation involves a number of hardware node requirements:

- **CPU architecture:** All nodes must use **x86_64** CPU architecture.
- **Similar nodes:** Red Hat recommends nodes have an identical configuration per role. That is, Red Hat recommends nodes be the same brand and model with the same CPU, memory and storage configuration.
- **Baseboard Management Controller:** The **provisioner** node must be able to access the baseboard management controller (BMC) of each OpenShift Container Platform cluster node. You may use IPMI, RedFish, or a proprietary protocol.
- **Latest generation:** Nodes must be of the most recent generation. Installer-provisioned installation relies on BMC protocols, which must be compatible across nodes. Additionally, RHEL 8 ships with the most recent drivers for RAID controllers. Ensure that the nodes are recent enough to support RHEL 8 for the **provisioner** node and RHCOS 8 for the control plane and worker nodes.
- **Registry node:** (Optional) If setting up a disconnected mirrored registry, it is recommended the registry reside in its own node.
- **Provisioner node:** Installer-provisioned installation requires one **provisioner** node.
- **Control plane:** Installer-provisioned installation requires three control plane nodes for high availability.
- **Worker nodes:** While not required, a typical production cluster has one or more worker nodes. Smaller clusters are more resource efficient for administrators and developers during development and testing.
- **Network interfaces:** Each node must have at least one 10GB network interface for the routable **baremetal** network. Each node must have one 10GB network interface for a **provisioning** network **when using the provisioning network** for deployment. Using the **provisioning** network is the default configuration. Network interface names must follow the same naming convention across all nodes. For example, the first NIC name on a node, such as **eth0** or **eno1**, must be the same name on all of the other nodes. The same principle applies to the remaining NICs on each node.
- **Unified Extensible Firmware Interface (UEFI):** Installer-provisioned installation requires UEFI boot on all OpenShift Container Platform nodes when using IPv6 addressing on the **provisioning** network. In addition, UEFI Device PXE Settings must be set to use the IPv6 protocol on the **provisioning** network NIC, but **omitting the provisioning network removes this requirement.**

1.2.2. Network requirements

Installer-provisioned installation of OpenShift Container Platform involves several network requirements by default. First, installer-provisioned installation involves a non-routable **provisioning** network for provisioning the operating system on each bare metal node and a routable **baremetal** network. Since installer-provisioned installation deploys **ironic-dnsmasq**, the networks should have no other DHCP servers running on the same broadcast domain. Network administrators must reserve IP addresses for each node in the OpenShift Container Platform cluster.

Network Time Protocol (NTP)

Each OpenShift Container Platform node in the cluster must have access to an NTP server.

Configuring NICs

OpenShift Container Platform deploys with two networks:

- **provisioning**: The **provisioning** network is an **optional** non-routable network used for provisioning the underlying operating system on each node that is a part of the OpenShift Container Platform cluster. When deploying using the **provisioning** network, the first NIC on each node, such as **eth0** or **eno1**, **must** interface with the **provisioning** network.

- **baremetal**: The **baremetal** network is a routable network. When deploying using the **provisioning** network, the second NIC on each node, such as **eth1** or **eno2**, **must** interface with the **baremetal** network. When deploying without a **provisioning** network, you can use any NIC on each node to interface with the **baremetal** network.

IMPORTANT

Each NIC should be on a separate VLAN corresponding to the appropriate network.

Configuring the DNS server

Clients access the OpenShift Container Platform cluster nodes over the **baremetal** network. A network administrator must configure a subdomain or subzone where the canonical name extension is the cluster name.

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

For example:

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

Reserving IP Addresses for Nodes with the DHCP Server

For the **baremetal** network, a network administrator must reserve a number of IP addresses, including:

1. Two virtual IP addresses.
   - One IP address for the API endpoint
   - One IP address for the wildcard ingress endpoint
2. One IP address for the provisioner node.
3. One IP address for each control plane (master) node.
4. One IP address for each worker node, if applicable.

The following table provides an exemplary embodiment of hostnames for each node in the OpenShift Container Platform cluster.

Usage	Hostname	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>*
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>*

Additional requirements with no provisioning network

All installer-provisioned installations require a **baremetal** network. The **baremetal** network is a routable network used for external network access to the outside world. In addition to the IP address supplied to the OpenShift Container Platform cluster node, installations without a **provisioning** network require the following:

- Setting an available IP address from the **baremetal** network to the **bootstrapProvisioningIP** configuration setting within the **install-config.yaml** configuration file.
- Setting an available IP address from the **baremetal** network to the **provisioningHostIP** configuration setting within the **install-config.yaml** configuration file.
- Deploying the OpenShift Container Platform cluster using RedFish Virtual Media/iDRAC Virtual Media.

NOTE

Configuring additional IP addresses for **bootstrapProvisioningIP** and **provisioningHostIP** is not required when using a **provisioning** network.

1.2.3. Configuring nodes

Configuring nodes when using the provisioning network

Each node in the cluster requires the following configuration for proper installation.

WARNING

A mismatch between nodes will cause an installation failure.

While the cluster nodes can contain more than two NICs, the installation process only focuses on the first two NICs:

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

NIC1 is a non-routable network (**provisioning**) that is only used for the installation of the OpenShift Container Platform cluster.

The RHEL 8.x installation process on the provisioner node might vary. To install RHEL 8.x using a local Satellite server or a PXE server, PXE-enable NIC2.

PXE	Boot order
NIC1 PXE-enabled **provisioning** network	1
NIC2 **baremetal** network. PXE-enabled is optional.	2

NOTE

Ensure PXE is disabled on all other NICs.

Configure the control plane and worker nodes as follows:

PXE	Boot order
NIC1 PXE-enabled (provisioning network)	1

Configuring nodes without the **provisioning** network

The installation process requires one NIC:

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

NICx is a routable network (**baremetal**) that is used for the installation of the OpenShift Container Platform cluster, and routable to the internet.

1.2.4. Out-of-band management

Nodes will typically have an additional NIC used by the Baseboard Management Controllers (BMCs). These BMCs must be accessible from the **provisioner** node.

Each node must be accessible via out-of-band management. When using an out-of-band management network, the **provisioner** node requires access to the out-of-band management network for a successful OpenShift Container Platform 4 installation.

The out-of-band management setup is out of scope for this document. We recommend setting up a separate management network for out-of-band management. However, using the **provisioning** network or the **baremetal** network are valid options.

1.2.5. Required data for installation

Prior to the installation of the OpenShift Container Platform cluster, gather the following information from all cluster nodes:

- Out-of-band management IP
  - Examples
    - Dell (iDRAC) IP
    - HP (iLO) IP

When using the provisioning network

- NIC1 (**provisioning**) MAC address
- NIC2 (**baremetal**) MAC address

When omitting the provisioning network

- NICx (**baremetal**) MAC address

1.2.6. Validation checklist for nodes

When using the provisioning network

❑ NIC1 VLAN is configured for the **provisioning** network.

❑ NIC2 VLAN is configured for the **baremetal** network.

❑ NIC1 is PXE-enabled on the provisioner, Control Plane (master), and worker nodes.

❑ PXE has been disabled on all other NICs.

❑ Control plane and worker nodes are configured.

❑ All nodes accessible via out-of-band management.

❑ A separate management network has been created. (optional)

❑ Required data for installation.

When omitting the provisioning network

❑ NICx VLAN is configured for the **baremetal** network.

❑ Control plane and worker nodes are configured.

❑ All nodes accessible via out-of-band management.

❑ A separate management network has been created. (optional)

❑ Required data for installation.

1.3. SETTING UP THE ENVIRONMENT FOR AN OPENSHIFT INSTALLATION

1.3.1. Installing RHEL on the provisioner node

With the networking configuration complete, the next step is to install RHEL 8.x on the provisioner node. The installer uses the provisioner node as the orchestrator while installing the OpenShift Container Platform cluster. For the purposes of this document, installing RHEL on the provisioner node is out of scope. However, options include but are not limited to using a RHEL Satellite server, PXE, or installation media.

1.3.2. Preparing the provisioner node for OpenShift Container Platform installation

Perform the following steps to prepare the environment.

Procedure

1. Log in to the provisioner node via **ssh**.
2. Create a non-root user (**kni**) and provide that user with **sudo** privileges.

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

3. Create an **ssh** key for the new user.

```
[root@provisioner ~]# su - kni -c "ssh-keygen -t rsa -f /home/kni/.ssh/id_rsa -N ''"
```

4. Log in as the new user on the provisioner node.

```
[root@provisioner ~]# su - kni
[kni@provisioner ~]$
```

5. Use Red Hat Subscription Manager to register the provisioner node.

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

NOTE

For more information about Red Hat Subscription Manager, see Using and Configuring Red Hat Subscription Manager.

6. Install the following packages.

```
[kni@provisioner ~]$ sudo dnf install -y libvirt qemu-kvm mkisofs python3-devel jq ipmitool
```

7. Modify the user to add the **libvirt** group to the newly created user.

```
[kni@provisioner ~]$ sudo usermod --append --groups libvirt <user>
```

8. Restart **firewalld** and enable the **http** service.

```
[kni@provisioner ~]$ sudo systemctl start firewalld
[kni@provisioner ~]$ sudo firewall-cmd --zone=public --add-service=http --permanent
[kni@provisioner ~]$ sudo firewall-cmd --reload
```

9. Start and enable the **libvirtd** service.

```
[kni@provisioner ~]$ sudo systemctl start libvirtd
[kni@provisioner ~]$ sudo systemctl enable libvirtd --now
```

10. Create the **default** storage pool and start it.

```
[kni@provisioner ~]$ sudo virsh pool-define-as --name default --type dir --target
/var/lib/libvirt/images
[kni@provisioner ~]$ sudo virsh pool-start default
[kni@provisioner ~]$ sudo virsh pool-autostart default
```

11. Configure networking.

NOTE

This step can also be run from the web console.

```
[kni@provisioner ~]$ export PUB_CONN=<baremetal_nic_name>
[kni@provisioner ~]$ export PROV_CONN=<prov_nic_name>
[kni@provisioner ~]$ 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 "$PUB_CONN" master baremetal
    nmcli con down "$PUB_CONN";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
'
```

NOTE

The **ssh** connection might disconnect after executing this step.

The IPv6 address can be any address as long as it is not routable via the **baremetal** network.

Ensure that UEFI is enabled and UEFI PXE settings are set to the IPv6 protocol when using IPv6 addressing.

12. **ssh** back into the **provisioner** node (if required).

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

13. Verify the connection bridges have been properly created.

```
[kni@provisioner ~]$ 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
```

14. Create a **pull-secret.txt** file.

```
[kni@provisioner ~]$ vim pull-secret.txt
```

In a web browser, navigate to Install on Bare Metal with user-provisioned infrastructure , and scroll down to the **Downloads** section. Click **Copy pull secret**. Paste the contents into the **pull-secret.txt** file and save the contents in the **kni** user’s home directory.

1.3.3. Retrieving the OpenShift Container Platform installer

Use the **latest-4.x** version of the installer to deploy the latest generally available version of OpenShift Container Platform:

```
[kni@provisioner ~]$ 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}')
```

1.3.4. Extracting the OpenShift Container Platform installer

After retrieving the installer, the next step is to extract it.

Procedure

1. Set the environment variables:

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

2. Get the **oc** binary:

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

3. Extract the installer:

```
[kni@provisioner ~]$ sudo cp oc /usr/local/bin
[kni@provisioner ~]$ oc adm release extract --registry-config "${pullsecret_file}" --
command=$cmd --to "${extract_dir}" ${RELEASE_IMAGE}
```

1.3.5. Creating an RHCOS images cache (optional)

To employ image caching, you must download two images: the RHCOS image used by the bootstrap VM and the RHCOS image used by the installer to provision the different nodes. Image caching is optional, but especially useful when running the installer on a network with limited bandwidth.

If you are running the installer on a network with limited bandwidth and the RHCOS images download takes more than 15 to 20 minutes, the installer will timeout. Caching images on a web server will help in such scenarios.

Use the following steps to install a container that contains the images.

1. Install **podman**.

```
[kni@provisioner ~]$ sudo dnf install -y podman
```

2. Open firewall port **8080** to be used for RHCOS Image caching.

```
[kni@provisioner ~]$ sudo firewall-cmd --add-port=8080/tcp --zone=public --permanent
```

3. Create a directory to store the **bootstraposimage** and **clusterosimage**.

```
[kni@provisioner ~]$ mkdir /home/kni/rhcos_image_cache
```

4. Set the appropriate SELinux context for the newly created directory.

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

5. Get the commit ID from the installer. The ID determines which images the installer needs to download.

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

6. Get the URI for the RHCOS image that the installer will deploy on the nodes.

```
[kni@provisioner ~]$ export RHCOS_OPENSTACK_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq
.images.openstack.path | sed 's/"//g')
```

7. Get the URI for the RHCOS image that the installer will deploy on the bootstrap VM.

```
[kni@provisioner ~]$ export RHCOS_QEMU_URI=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq
.images.qemu.path | sed 's/"//g')
```

8. Get the path where the images are published.

```
[kni@provisioner ~]$ export RHCOS_PATH=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json | jq
.baseURI | sed 's/"//g')
```

9. Get the SHA hash for the RHCOS image that will be deployed on the bootstrap VM.

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

10. Get the SHA hash for the RHCOS image that will be deployed on the nodes.

```
[kni@provisioner ~]$ export RHCOS_OPENSTACK_SHA_COMPRESSED=$(curl -s -S
https://raw.githubusercontent.com/openshift/installer/$COMMIT_ID/data/data/rhcos.json  | jq
-r '.images.openstack.sha256')
```

11. Download the images and place them in the **/home/kni/rhcos_image_cache** directory.

```
[kni@provisioner ~]$ curl -L ${RHCOS_PATH}${RHCOS_QEMU_URI} -o
/home/kni/rhcos_image_cache
[kni@provisioner ~]$ curl -L ${RHCOS_PATH}${RHCOS_OPENSTACK_URI} -o
/home/kni/rhcos_image_cache
```

12. Confirm SELinux type is of **httpd_sys_content_t** for the newly created files.

```
[kni@provisioner ~]$ ls -Z /home/kni/rhcos_image_cache
```

13. Create the pod.

```
[kni@provisioner ~]$ podman run -d --name rhcos_image_cache \
-v /home/kni/rhcos_image_cache:/var/www/html \
-p 8080:8080/tcp \
registry.centos.org/centos/httpd-24-centos7:latest
```

■

1.3.6. Configuration files

1.3.6.1. Configuring the install-config.yaml file

The **install-config.yaml** file requires some additional details. Most of the information is teaching the installer and the resulting cluster enough about the available hardware so that it is able to fully manage it.

1. Configure **install-config.yaml**. Change the appropriate variables to match the environment, including **pullSecret** and **sshKey**.

```
apiVersion: v1
baseDomain: <domain>
metadata:
  name: <cluster-name>
networking:
  machineCIDR: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2
controlPlane:
  name: master
  replicas: 3
  platform:
    baremetal: {}
platform:
  baremetal:
    apiVIP: <api-ip>
    ingressVIP: <wildcard-ip>
    provisioningBridge: provisioning
    provisioningNetworkCIDR: 172.22.0.0/24
    hosts:
      - name: openshift-master-0
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
      - name: openshift-master-1
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
      - name: openshift-master-2
        role: master
        bmc:
          address: ipmi://<out-of-band-ip>
          username: <user>
          password: <password>
        bootMACAddress: <NIC1-mac-address>
    - name: openshift-worker-0
      role: worker
      bmc:
        address: ipmi://<out-of-band-ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1-mac-address>
    - name: openshift-worker-1
      role: worker
      bmc:
        address: ipmi://<out-of-band-ip>
        username: <user>
        password: <password>
      bootMACAddress: <NIC1-mac-address>
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

2. Create a directory to store cluster configs.

```
[kni@provisioner ~]$ mkdir ~/clusterconfigs
[kni@provisioner ~]$ cp install-config.yaml ~/clusterconfigs
```

3. Ensure all bare metal nodes are powered off prior to installing the OpenShift Container Platform cluster.

```
[kni@provisioner ~]$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

4. Remove old bootstrap resources if any are left over from a previous deployment attempt.

```
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 default;
  sudo virsh vol-delete $i.ign --pool default;
done
```

1.3.6.2. Modifying the **install-config.yaml** file for no **provisioning** network (optional)

To deploy an OpenShift Container Platform cluster without a **provisioning** network, make the following changes to the **install-config.yaml** file.

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

NOTE

Requires providing two IP addresses from the **baremetal** network for the **provisioningHostIP** and **bootstrapProvisioningIP** configuration settings, and removing the **provisioningBridge** and **provisioningNetworkCIDR** configuration settings.

1.3.6.3. Additional **install-config** parameters

See the following tables for the required parameters, the **hosts** parameter, and the **bmc** parameter for the **install-config.yaml** file.

Table 1.1. Required parameters

Parameters	Default	Description
baseDomain		The domain name for the cluster. For example, **example.com**.
sshKey		The **sshKey** configuration setting contains the key in the **~/.ssh/id_rsa.pub** file required to access the control plane nodes and worker nodes. Typically, this key is from the **provisioner** node.
pullSecret		The **pullSecret** configuration setting contains a copy of the pull secret downloaded from the Install OpenShift on Bare Metal page when preparing the provisioner node.

`metadata:`
`    name:`
		The name to be given to the OpenShift Container Platform cluster. For example, **openshift**.

`networking:`
`    machineCIDR:`
		The public CIDR (Classless Inter-Domain Routing) of the external network. For example, **10.0.0.0/24** .

`compute:`
`  - name: worker`
		The OpenShift Container Platform cluster requires a name be provided for worker (or compute) nodes even if there are zero nodes.

`compute:`
`    replicas: 2`
		Replicas sets the number of worker (or compute) nodes in the OpenShift Container Platform cluster.

Parameters	Default	Description

`controlPlane:`
`  name: master`
		The OpenShift Container Platform cluster requires a name for control plane (master) nodes.

`controlPlane:`
`  replicas: 3`
		Replicas sets the number of control plane (master) nodes included as part of the OpenShift Container Platform cluster.

provisioningNetworkInterface		The name of the network interface on control plane nodes connected to the provisioning network.
defaultMachinePlatform		The default configuration used for machine pools without a platform configuration.
apiVIP	**api.<clustername.clusterdomain>**	The VIP to use for internal API communication. This setting must either be provided or pre-configured in the DNS so that the default name resolves correctly.
disableCertificateVerification	**False**	**redfish** and **redfish-virtualmedia** need this parameter to manage BMC addresses. The value should be **True** when using a self-signed certificate for BMC addresses.
ingressVIP	**test.apps.<clustername.clusterdomain>**	The VIP to use for ingress traffic.

Table 1.2. Optional Parameters

Parameters	Default	Description
provisioningDHCPRange	**172.22.0.10,172.22.0.100**	Defines the IP range for nodes on the **provisioning** network.
provisioningNetworkCIDR	**172.22.0.0/24**	The CIDR for the network to use for provisioning. This option is required when not using the default address range on the **provisioning** network.

Parameters	Default	Description
clusterProvisioningIP	The third IP address of the **provisioningNetworkCIDR**.	The IP within the cluster where the provisioning services run. Defaults to the 3rd IP of the **provisioning** subnet. For example, **172.22.0.3**.
bootstrapProvisioningIP	The second IP address of the **provisioningNetworkCIDR**.	The IP on the bootstrap VM where the provisioning services run while the the installer is deploying the control plane (master) nodes. Defaults to the second IP of the **provisioning** subnet. For example, **172.22.0.2** . When using no **provisioning** network, set this value to an IP address that is available on the **baremetal** network.
externalBridge	**baremetal**	The name of the **baremetal** bridge of the hypervisor attached to the **baremetal** network.
provisioningBridge	**provisioning**	The name of the **provisioning** bridge on the **provisioner** host attached to the **provisioning** network.
defaultMachinePlatform		The default configuration used for machine pools without a platform configuration.
bootstrapOSImage		A URL to override the default operating system image for the bootstrap node. The URL must contain a SHA-256 hash of the image. For example: **https://mirror.openshift.com/rhcos-<version>-qemu.qcow2.gz?sha256=<uncompressed_sha256>** .
clusterOSImage		A URL to override the default operating system for cluster nodes. The URL must include a SHA-256 hash of the image. For example, **https://mirror.openshift.com/images/rhcos-<version>-openstack.qcow2.gz?sha256=<compressed_sha256>**.
provisioningNetwork		Set this parameter to **Disabled** to disable the requirement for a **provisioning** network. User may only do virtual media based provisioning, or bring up the cluster using assisted installation. If using power management, BMC's must be accessible from the machine networks. User must provide two IP addresses on the external network that are used for the provisioning services. Set this parameter to **managed**, which is the default, to fully manage the provisioning network, including DHCP, TFTP, and so on. Set this parameter to **unmanaged** to still enable the provisioning network but take care of manual configuration of DHCP. Virtual Media provisioning is recommended but PXE is still available if required.

Parameters	Default	Description
provisioningHostingIp		Set this parameter to an available IP address on the **baremetal** network when the **provisioningNetwork** configuration setting is set to **Disabled**.
httpProxy		Set this parameter to the appropriate HTTP proxy used within your environment.
httpsProxy		Set this parameter to the appropriate HTTPS proxy used within your environment.
noProxy		Set this parameter to the appropriate list of exclusions for proxy usage within your environment.

Hosts

The **hosts** parameter is a list of separate bare metal assets used to build the cluster.

Name	Default	Description
name		The name of the **BareMetalHost** resource to associate with the details. For example, **openshift-master-0**.
role		The role of the bare metal node. Either **master** or **worker**.
bmc		Connection details for the baseboard management controller. See the BMC addressing section for additional details.
bootMACAddress		The MAC address of the NIC the host will use to boot on the **provisioning** network.

1.3.6.4. BMC addressing

The **address** field for each **bmc** entry is a URL for connecting to the OpenShift Container Platform cluster nodes, including the type of controller in the URL scheme and its location on the network.

IPMI

IPMI hosts use **ipmi://<out-of-band-ip>:<port>** and defaults to port **623** if not specified. The following example demonstrates an IPMI configuration within the **install-config.yaml** file.

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

RedFish for HPE

To enable RedFish, use **redfish://** or **redfish+http://** to disable TLS. The installer requires both the hostname or the IP address and the path to the system ID. The following example demonstrates a RedFish configuration within the **install-config.yaml** file.

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

While it is recommended to have a certificate of authority for the out-of-band management addresses, you must include **disableCertificateVerification: True** in the **bmc** configuration if using self-signed certificates. The following example demonstrates a RedFish configuration using the **disableCertificateVerification: True** configuration parameter within the **install-config.yaml** file.

```
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

To enable RedFish, use **redfish://** or **redfish+http://** to disable TLS. The installer requires both the hostname or the IP address and the path to the system ID. The following example demonstrates a RedFish configuration within the **install-config.yaml** file.

```
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>
```

While it is recommended to have a certificate of authority for the out-of-band management addresses, you must include **disableCertificateVerification: True** in the **bmc** configuration if using self-signed certificates. The following example demonstrates a RedFish configuration using the **disableCertificateVerification: True** configuration parameter within the **install-config.yaml** file.

```
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
```

NOTE

Currently RedFish is only supported on Dell with iDRAC firmware version **4.20.20.20** or higher for installer-provisioned installations of OpenShift Container Platform on bare metal deployments.

RedFish Virtual Media for HPE

To enable RedFish Virtual Media for HPE servers, use **redfish-virtualmedia://** in the **address** setting. The following example demonstrates using RedFish Virtual Media within the **install-config.yaml** file.

```
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

For RedFish Virtual Media on Dell servers, use **idrac-virtualmedia://** in the **address** setting.

NOTE

RedFish Virtual Media on Dell servers has a known issue in OpenShift Container Platform 4.6. The 4.6.1 point release will resolve the issue.

The following example demonstrates using iDRAC Virtual Media within the **install-config.yaml** file.

```
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>
```

NOTE

idrac-virtualmedia requires iDRAC firmware version 4.20.20.20 or higher.

Ensure the OpenShift Container Platform cluster nodes have AutoAttach Enabled through the iDRAC console. The menu path is: **Configuration→Virtual Media→Attach Mode→AutoAttach**.

1.3.6.5. Root device hints

The **rootDeviceHints** parameter enables the installer to provision the Red Hat Enterprise Linux CoreOS (RHCOS) image to a particular device. The installer examines the devices in the order it discovers them, and compares the discovered values with the hint values. The installer uses the first discovered device that matches the hint value. The configuration can combine multiple hints, but a device must match all hints for the installer to select it.

Table 1.3. Subfields

Subfield	Description
deviceName	A string containing a Linux device name like **/dev/vda**. The hint must match the actual value exactly.
hctl	A string containing a SCSI bus address like **0:0:0:0**. The hint must match the actual value exactly.
model	A string containing a vendor-specific device identifier. The hint can be a substring of the actual value.
vendor	A string containing the name of the vendor or manufacturer of the device. The hint can be a sub-string of the actual value.
serialNumber	A string containing the device serial number. The hint must match the actual value exactly.
minSizeGigabytes	An integer representing the minimum size of the device in gigabytes.
wwn	A string containing the unique storage identifier. The hint must match the actual value exactly.

Subfield	Description
wwnWithExtension	A string containing the unique storage identifier with the vendor extension appended. The hint must match the actual value exactly.
wwnVendorExtension	A string containing the unique vendor storage identifier. The hint must match the actual value exactly.
rotational	A Boolean indicating whether the device should be a rotating disk (true) or not (false).

Example usage

```
     - 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"
```

1.3.6.6. Creating the OpenShift Container Platform manifests

1. Create the OpenShift Container Platform manifests.

   ```
   [kni@provisioner ~]$ ./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. Creating a disconnected registry (optional)

In some cases, you might want to install an OpenShift KNI cluster using a local copy of the installation registry. This could be for enhancing network efficiency because the cluster nodes are on a network that does not have access to the internet.

A local, or mirrored, copy of the registry requires the following:

- A certificate for the registry node. This can be a self-signed certificate.
- A webserver - this will be served by a container on a system.
- An updated pull secret that contains the certificate and local repository information.

NOTE

Creating a disconnected registry on a registry node is optional. The subsequent sections indicate that they are optional since they are steps you need to execute only when creating a disconnected registry on a registry node. You should execute all of the subsequent sub-sections labeled "(optional)" when creating a disconnected registry on a registry node.

1.3.7.1. Preparing the registry node to host the mirrored registry (optional)

Make the following changes to the registry node.

Procedure

1. Open the firewall port on the registry node.

```
[user@registry ~]$ sudo firewall-cmd --add-port=5000/tcp --zone=libvirt  --permanent
[user@registry ~]$ sudo firewall-cmd --add-port=5000/tcp --zone=public   --permanent
[user@registry ~]$ sudo firewall-cmd --reload
```

2. Install the required packages for the registry node.

```
[user@registry ~]$ sudo yum -y install python3 podman httpd httpd-tools jq
```

3. Create the directory structure where the repository information will be held.

```
[user@registry ~]$ sudo mkdir -p /opt/registry/{auth,certs,data}
```

1.3.7.2. Generating the self-signed certificate (optional)

Generate a self-signed certificate for the registry node and put it in the **/opt/registry/certs** directory.

Procedure

1. Adjust the certificate information as appropriate.

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

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

NOTE

When replacing **<Country Name>**, ensure that it only contains two letters. For example, **US**.

2. Update the registry node's **ca-trust** with the new certificate.

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

1.3.7.3. Creating the registry podman container (optional)

The registry container uses the **/opt/registry** directory for certificates, authentication files, and to store its data files.

The registry container uses **httpd** and needs an **htpasswd** file for authentication.

Procedure

1. Create an **htpasswd** file in **/opt/registry/auth** for the container to use.

```
[user@registry ~]$ htpasswd -bBc /opt/registry/auth/htpasswd <user> <passwd>
```

Replace **<user>** with the user name and **<passwd>** with the password.

2. Create and start the registry container.

```
[user@registry ~]$ 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
```

```
[user@registry ~]$ podman start ocpdiscon-registry
```

1.3.7.4. Copy and update the pull-secret (optional)

Copy the pull secret file from the provisioner node to the registry node and modify it to include the authentication information for the new registry node.

Procedure

1. Copy the **pull-secret.txt** file.

```
[user@registry ~]$ scp kni@provisioner:/home/kni/pull-secret.txt pull-secret.txt
```

2. Update the **host_fqdn** environment variable with the fully qualified domain name of the registry node.

```
[user@registry ~]$ host_fqdn=$( hostname --long )
```

3. Update the **b64auth** environment variable with the base64 encoding of the **http** credentials used to create the **htpasswd** file.

```
[user@registry ~]$ b64auth=$( echo -n '<username>:<passwd>' | openssl base64 )
```

   Replace **<username>** with the user name and **<passwd>** with the password.

4. Set the **AUTHSTRING** environment variable to use the **base64** authorization string. The **$USER** variable is an environment variable containing the name of the current user.

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

5. Update the pull-secret file.

```
[user@registry ~]$ jq ".auths += $AUTHSTRING" < pull-secret.json > pull-secret-update.json
```

1.3.7.5. Mirroring the repository (optional)

Procedure

1. Copy the **oc** binary from the provisioner node to the registry node.

```
[user@registry ~]$ sudo scp kni@provisioner:/usr/local/bin/oc /usr/local/bin
```

2. Mirror the remote install images to the local repository.

```
[user@registry ~]$ /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
```

1.3.7.6. Modify the install-config.yaml file to use the disconnected registry (optional)

On the provisioner node, the **install-config.yaml** file should use the newly created pull-secret from the **pull-secret-update.json** file. The **install-config.yaml** file must also contain the disconnected registry node's certificate and registry information.

Procedure

1. Add the disconnected registry node's certificate to the **install-config.yaml** file. The certificate should follow the **"additionalTrustBundle: |"** line and be properly indented, usually by two spaces.

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

2. Add the mirror information for the registry to the **install-config.yaml** file.

```
[kni@provisioner ~]$ echo "imageContentSources:" >> install-config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-
config.yaml
[kni@provisioner ~]$ echo "  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev" >>
install-config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-
config.yaml
[kni@provisioner ~]$ echo "  source: registry.svc.ci.openshift.org/ocp/release" >> install-
config.yaml
[kni@provisioner ~]$ echo "- mirrors:" >> install-config.yaml
[kni@provisioner ~]$ echo "  - registry.example.com:5000/ocp4/openshift4" >> install-
config.yaml
[kni@provisioner ~]$ echo "  source: quay.io/openshift-release-dev/ocp-release" >> install-
config.yaml
```

NOTE

Replace **registry.example.com** with the registry's fully qualified domain name.

1.3.8. Deploying routers on worker nodes

During installation, the installer deploys router pods on worker nodes. By default, the installer installs two router pods. If the initial cluster has only one worker node, or if a deployed cluster requires additional routers to handle external traffic loads destined for services within the OpenShift Container Platform cluster, you can create a **yaml** file to set an appropriate number of router replicas.

NOTE

By default, the installer deploys two routers. If the cluster has at least two worker nodes, you can skip this section. For more information on the Ingress Operator see: Ingress Operator in OpenShift Container Platform.

NOTE

If the cluster has no worker nodes, the installer deploys the two routers on the control plane nodes by default. If the cluster has no worker nodes, you can skip this section.

Procedure

1. Create a **router-replicas.yaml** file.

```
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: ""
```

NOTE

Replace **<num-of-router-pods>** with an appropriate value. If working with just one worker node, set **replicas:** to **1**. If working with more than 3 worker nodes, you can increase **replicas:** from the default value **2** as appropriate.

2. Save and copy the **router-replicas.yaml** file to the **clusterconfigs/openshift** directory.

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

1.3.9. Validation checklist for installation

❑ OpenShift Container Platform installer has been retrieved.

❑ OpenShift Container Platform installer has been extracted.

❑ Required parameters for the **install-config.yaml** have been configured.

❑ The **hosts** parameter for the **install-config.yaml** has been configured.

❑ The **bmc** parameter for the **install-config.yaml** has been configured.

❑ Conventions for the values configured in the **bmc address** field have been applied.

❑ Created a disconnected registry (optional).

❑ (optional) Validate disconnected registry settings if in use.

❑ (optional) Deployed routers on worker nodes.

1.3.10. Deploying the cluster via the OpenShift Container Platform installer

Run the OpenShift Container Platform installer:

```
[kni@provisioner ~]$ ./openshift-baremetal-install --dir ~/clusterconfigs --log-level debug create
cluster
```

1.3.11. Following the installation

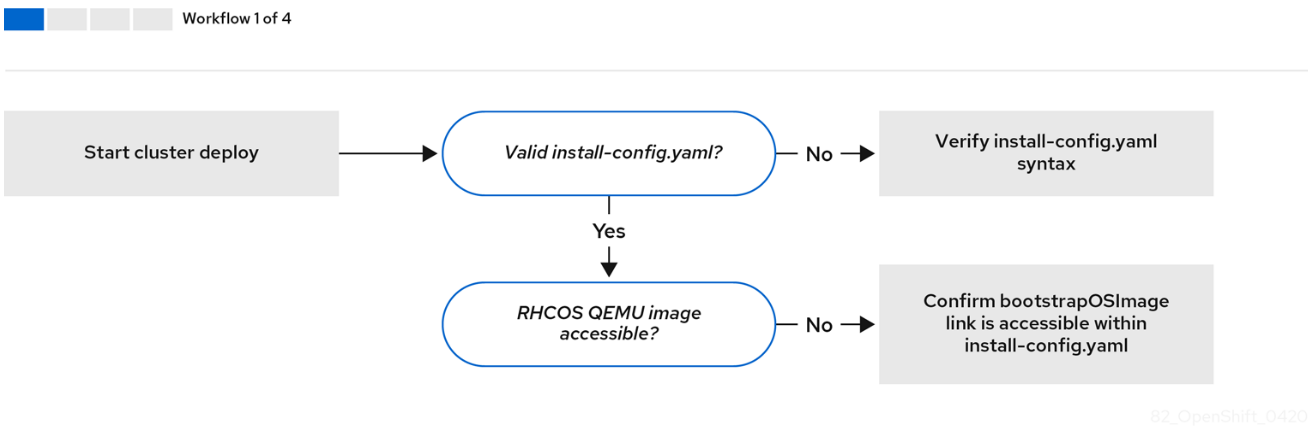
During the deployment process, you can check the installation's overall status by issuing the **tail** command to the **.openshift_install.log** log file in the install directory folder.

```
[kni@provisioner ~]$ tail -f /path/to/install-dir/.openshift_install.log
```

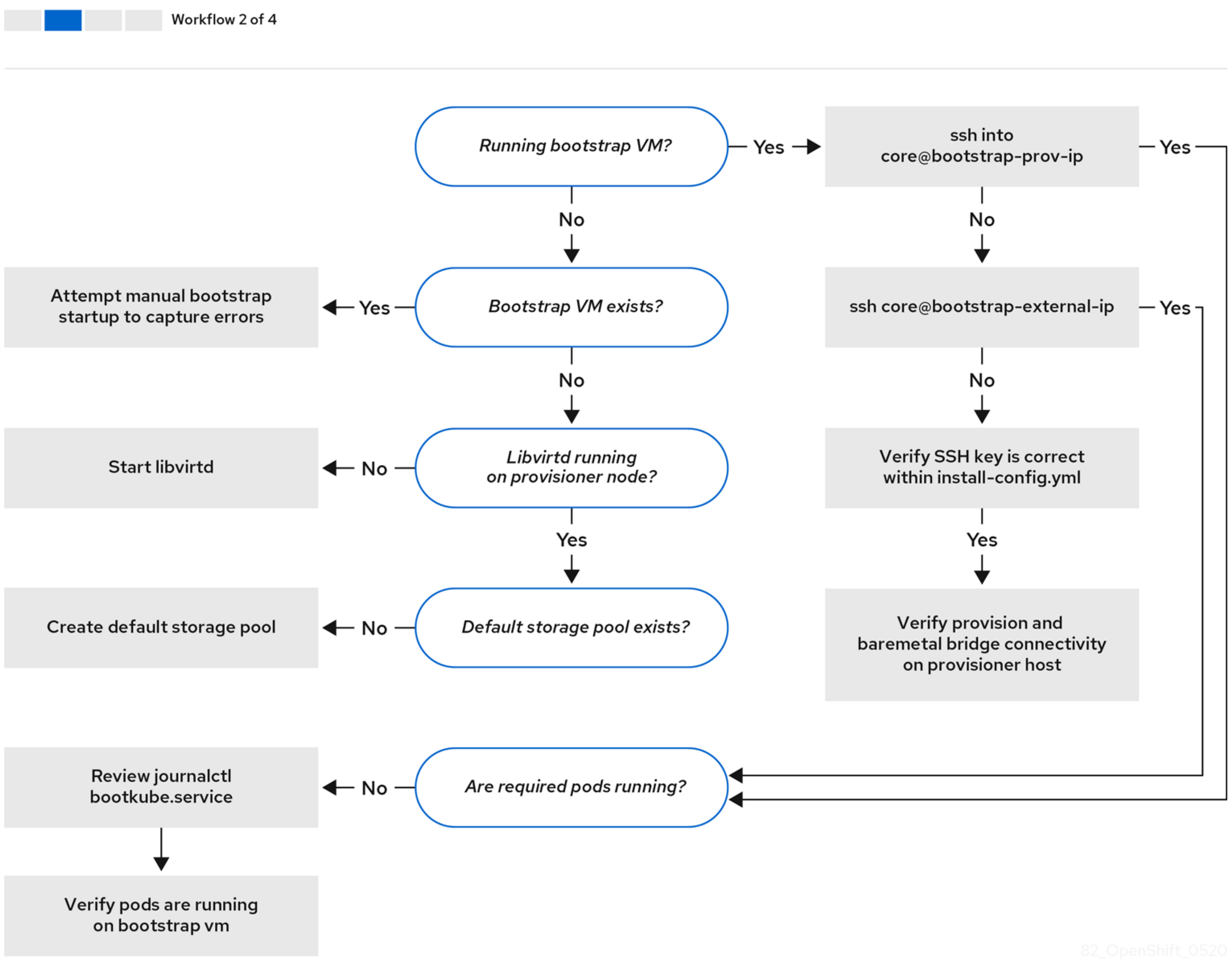
1.4. TROUBLESHOOTING

1.4.1. Troubleshooting the installer workflow

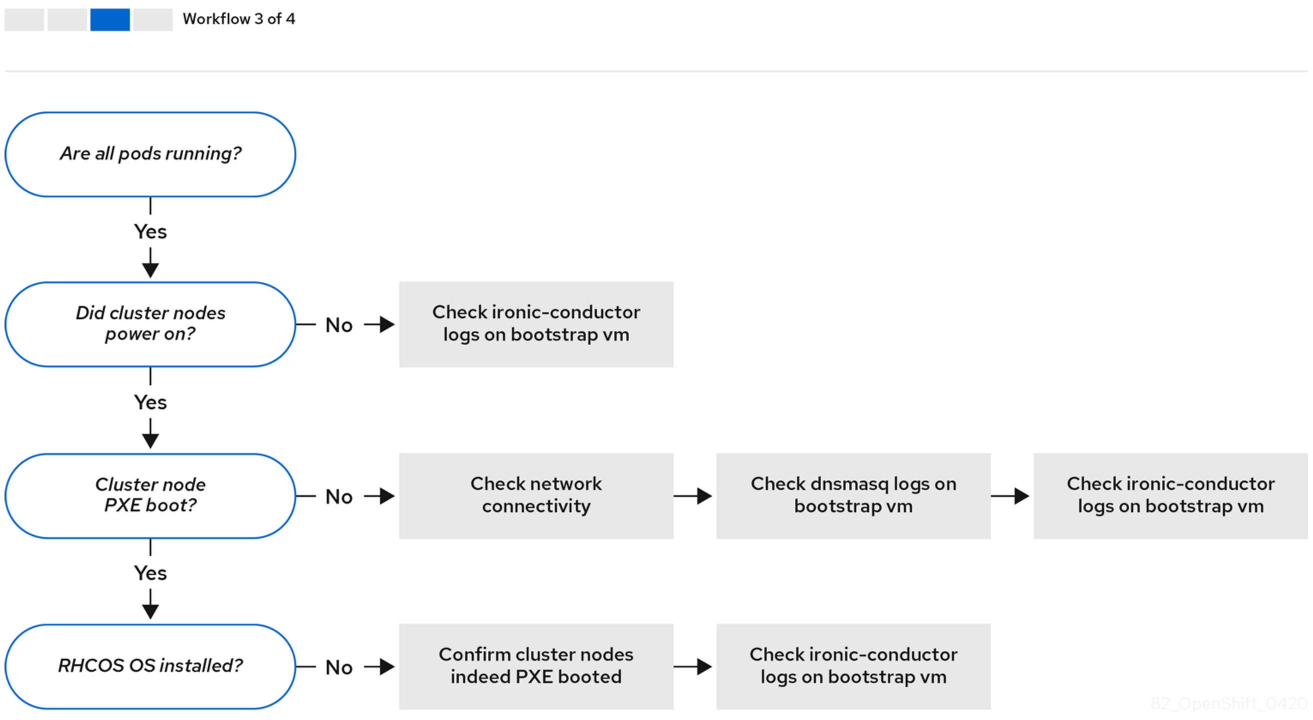
Prior to troubleshooting the installation environment, it is critical to understand the overall flow of the installer-provisioned installation on bare metal. The diagrams below provide a troubleshooting flow with a step-by-step breakdown for the environment.


Workflow 1 of 4
Start cluster deploy
Valid install-config.yaml?
No
Verify install-config.yaml syntax
Yes
RHCOS QEMU image accessible?
No
Confirm bootstrapOSImage link is accessible within install-config.yaml
82_OpenShift_0420

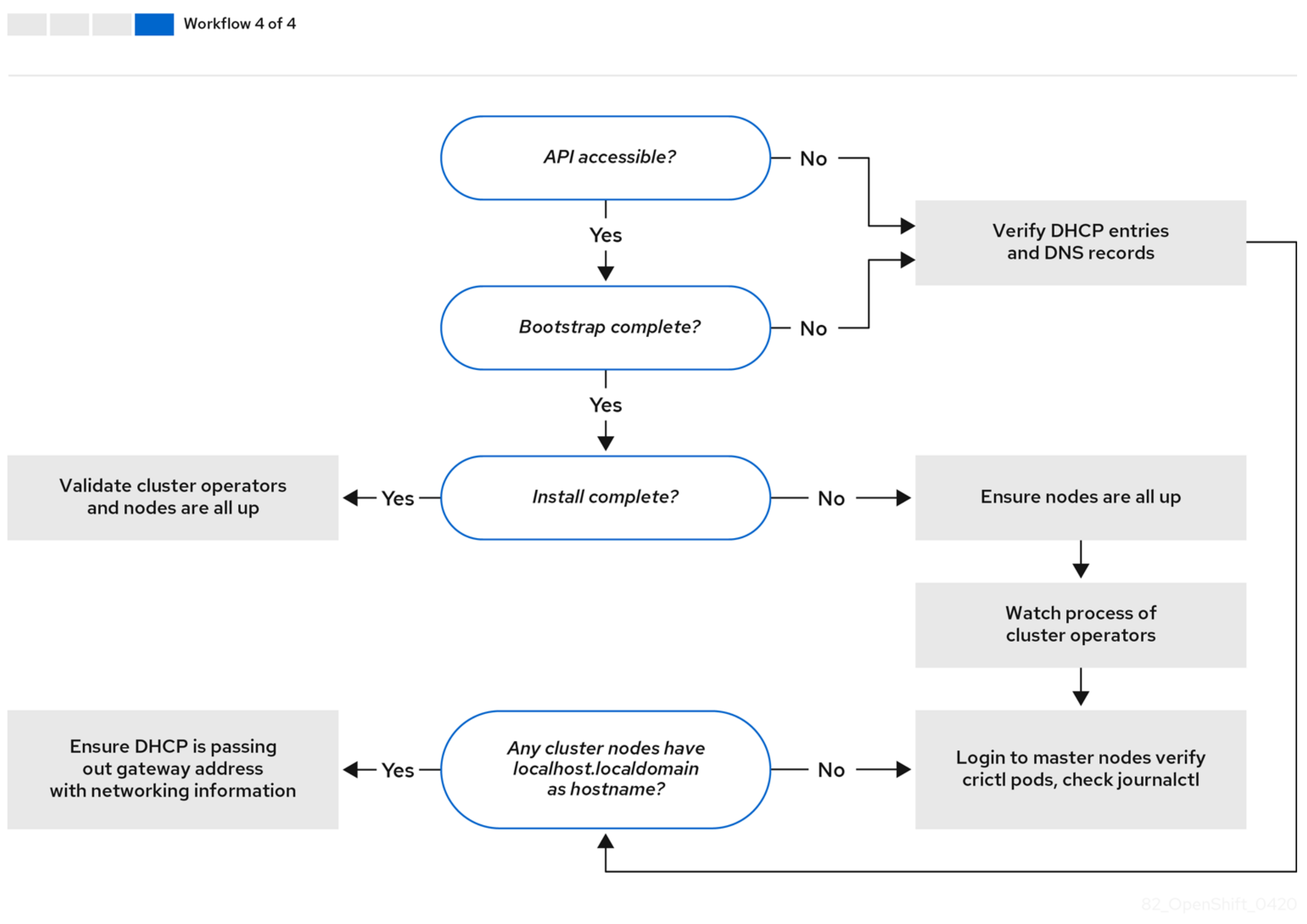
Workflow 1 of 4 illustrates a troubleshooting workflow when the **install-config.yaml** file has errors or the Red Hat Enterprise Linux CoreOS (RHCOS) images are inaccessible. Troubleshooting suggestions can be found at Troubleshooting **install-config.yaml**.


Workflow 2 of 4
Running bootstrap VM?
Yes
ssh into core@bootstrap-prov-ip
Yes
No
No
Attempt manual bootstrap startup to capture errors
Yes
Bootstrap VM exists?
ssh core@bootstrap-external-ip
Yes
No
No
Start libvirtd
No
Libvirtd running on provisioner node?
Verify SSH key is correct within install-config.yml
Yes
Yes
Create default storage pool
No
Default storage pool exists?
Verify provision and baremetal bridge connectivity on provisioner host
Review journalctl bootkube.service
No
Are required pods running?
Verify pods are running on bootstrap vm
82_OpenShift_0520

Workflow 2 of 4 illustrates a troubleshooting workflow for bootstrap VM issues, bootstrap VMs that cannot boot up the cluster nodes, and inspecting logs.


Workflow 3 of 4
Are all pods running?
Yes
Did cluster nodes power on?
No
Check ironic-conductor logs on bootstrap vm
Yes
Cluster node PXE boot?
No
Check network connectivity
Check dnsmasq logs on bootstrap vm
Check ironic-conductor logs on bootstrap vm
Yes
RHCOS OS installed?
No
Confirm cluster nodes indeed PXE booted
Check ironic-conductor logs on bootstrap vm
82_OpenShift_0420

Workflow 3 of 4 illustrates a troubleshooting workflow for cluster nodes that will not PXE boot.


Workflow 4 of 4
API accessible?
No
Yes
Verify DHCP entries and DNS records
Bootstrap complete?
No
Yes
Validate cluster operators and nodes are all up
Yes
Install complete?
No
Ensure nodes are all up
Watch process of cluster operators
Ensure DHCP is passing out gateway address with networking information
Yes
Any cluster nodes have localhost.localdomain as hostname?
No
Login to master nodes verify crictl pods, check journalctl
82_OpenShift_0420

Workflow 4 of 4 illustrates a troubleshooting workflow from a non-accessible API to a validated installation.

1.4.2. Troubleshooting **install-config.yaml**

The **install-config.yaml** configuration file represents all of the nodes that are part of the OpenShift Container Platform cluster. The file contains the necessary options consisting of but not limited to **apiVersion**, **baseDomain**, **imageContentSources** and virtual IP addresses. If errors occur early in the deployment of the OpenShift Container Platform cluster, the errors are likely in the **install-config.yaml** configuration file.

Procedure

1. Use the guidelines in YAML-tips.
2. Verify the YAML syntax is correct using syntax-check.
3. Verify the Red Hat Enterprise Linux CoreOS (RHCOS) QEMU images are properly defined and accessible via the URL provided in the **install-config.yaml**. For example:

```
[kni@provisioner ~]$ 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
```

If the output is **200**, there is a valid response from the webserver storing the bootstrap VM image.

1.4.3. Bootstrap VM issues

The OpenShift Container Platform installer spawns a bootstrap node virtual machine, which handles provisioning the OpenShift Container Platform cluster nodes.

Procedure

1. About 10 to 15 minutes after triggering the installer, check to ensure the bootstrap VM is operational using the **virsh** command:

```
[kni@provisioner ~]$ sudo virsh list
```

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

NOTE

The name of the bootstrap VM is always the cluster name followed by random set of characters and ending in the word "bootstrap."

If the bootstrap VM is not running after 10-15 minutes, troubleshoot why it was not created. Possible issues include:

1. Verify **libvirtd** is running on the system:

```
[kni@provisioner ~]$ 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
```

If the bootstrap VM is operational, log into it.

2. Use the **virsh console** command to find the IP address of the bootstrap VM:

```
[kni@provisioner ~]$ 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+SNcJ3a2g23o0lnIio (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:
```

3. Once you obtain the IP address, log in to the bootstrap VM using the **ssh** command:

   NOTE

   In the console output of the previous step, the IPv6 IP provided by **ens3** or the IPv4 IP provided by **ens4** can be used.

```
[kni@provisioner ~]$ ssh core@172.22.0.2
```

If you are not successful logging in to the bootstrap VM, you have likely encountered of the following scenarios:

- You cannot reach the **172.22.0.0/24** network. Verify network connectivity on the **provisioner** host specifically around the **provisioning** network bridge.
- You cannot reach the bootstrap VM via the public network. When attempting to SSH via **baremetal** network, verify connectivity on the **provisioner** host specifically around the **baremetal** network bridge.
- You encountered **Permission denied (publickey,password,keyboard-interactive)**. When attempting to access the bootstrap VM a **Permission denied** error might occur. Verify that the SSH key for the user attempting to log into the VM is set within the **install-config.yaml** file.

1.4.3.1. Bootstrap VM cannot boot up the cluster nodes

During the deployment, it is possible for the bootstrap VM to fail to boot the cluster nodes, which prevents the VM from provisioning the nodes with the RHCOS image. This scenario can arise due to:

- A problem with the **install-config.yaml** file.
- Issues with out-of-band network access via the baremetal network.

To verify the issue, there are three containers related to **ironic**:

- **ironic-api**
- **ironic-conductor**
- **ironic-inspector**

Procedure

1. Log in to the bootstrap VM:

```
[kni@provisioner ~]$ ssh core@172.22.0.2
```

2. To check the container logs, execute the following:

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

Replace **<container-name>** with one of **ironic-api**, **ironic-conductor**, or **ironic-inspector**. If you encounter an issue where the master nodes are not booting up via PXE, check the **ironic-conductor** pod. The **ironic-conductor** pod contains the most detail about the attempt to boot the cluster nodes, because it attempts to log in to the node over IPMI.

Potential reason

The cluster nodes might be in the **ON** state when deployment started.

Solution

Power off the OpenShift Container Platform cluster nodes before you begin the installation over IPMI:

```
[kni@provisioner ~]$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip> power off
```

1.4.3.2. Inspecting logs

When experiencing issues downloading or accessing the RHCOS images, first verify that the URL is correct in the **install-config.yaml** configuration file.

Example of internal webserver hosting RHCOS images

```
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
```

The **ipa-downloader** and **coreos-downloader** containers download resources from a webserver or the external quay.io registry, whichever is specified in the **install-config.yaml** configuration file. Verify the following two containers are up and running and inspect their logs as needed:

- **ipa-downloader**
- **coreos-downloader**

Procedure

1. Log in to the bootstrap VM:

   ```
   [kni@provisioner ~]$ ssh core@172.22.0.2
   ```

2. Check the status of the **ipa-downloader** and **coreos-downloader** containers within the bootstrap VM:

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

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

   If the bootstrap VM cannot access the URL to the images, use the **curl** command to verify that the VM can access the images.

3. [illegible]

3. To inspect the **bootkube** logs that indicate if all the containers launched during the deployment phase, execute the following:

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

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

4. Verify all the pods, including **dnsmasq**, **mariadb**, **httpd**, and **ironic**, are running:

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

5. If there are issues with the pods, check the logs of the containers with issues. To check the log of the **ironic-api**, execute the following:

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

1.4.4. Cluster nodes will not PXE boot

When OpenShift Container Platform cluster nodes will not PXE boot, execute the following checks on the cluster nodes that will not PXE boot.

Procedure

1. Check the network connectivity to the **provisioning** network.

2. Ensure PXE is enabled on the NIC for the **provisioning** network and PXE is disabled for all other NICs.

3. Verify that the **install-config.yaml** configuration file has the proper hardware profile and boot MAC address for the NIC connected to the **provisioning** network. For example:

   Master node settings

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

   Worker node settings

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

1.4.5. The API is not accessible

When the cluster is running and clients cannot access the API, domain name resolution issues might impede access to the API.

Procedure

1. **Hostname Resolution:** Check the cluster nodes to ensure they have a fully qualified domain name, and not just **localhost.localdomain**. For example:

   ```
   [kni@provisioner ~]$ hostname
   ```

If a hostname is not set, set the correct hostname. For example:

```
[kni@provisioner ~]$ hostnamectl set-hostname <hostname>
```

2. **Incorrect Name Resolution:** Ensure that each node has the correct name resolution in the DNS server using **dig** and **nslookup**. For example:

```
[kni@provisioner ~]$ 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
```

The output in the foregoing example indicates that the appropriate IP address for the **api.<cluster-name>.example.com** VIP is **10.19.13.86**. This IP address should reside on the **baremetal** network.

1.4.6. Cleaning up previous installations

In the event of a previous failed deployment, remove the artifacts from the failed attempt before attempting to deploy OpenShift Container Platform again.

Procedure

1. Power off all bare metal nodes prior to installing the OpenShift Container Platform cluster:

```
[kni@provisioner ~]$ ipmitool -I lanplus -U <user> -P <password> -H <management-server-ip> power off
```

2. Remove all old bootstrap resources if any are left over from a previous deployment attempt:

```
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 default;
  sudo virsh vol-delete $i.ign --pool default;
done
```

3. Remove the following from the **clusterconfigs** directory to prevent Terraform from failing:

```
[kni@provisioner ~]$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform*
~/clusterconfigs/tls ~/clusterconfigs/metadata.json
```

1.4.7. Issues with creating the registry

When creating a disconnected registry, you might encounter a "User Not Authorized" error when attempting to mirror the registry. This error might occur if you fail to append the new authentication to the existing **pull-secret.txt** file.

Procedure

1. Check to ensure authentication is successful:

```
[user@registry ~]$ /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
```

NOTE

Example output of the variables used to mirror the install images:

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

The values of **RELEASE_IMAGE** and **VERSION** were set during the **Retrieving OpenShift Installer** step of the **Setting up the environment for an OpenShift installation** section.

2. After mirroring the registry, confirm that you can access it in your disconnected environment:

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

1.4.8. Miscellaneous issues

1.4.8.1. Addressing the runtime network not ready error

After the deployment of a cluster you might receive the following error:

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

The Cluster Network Operator is responsible for deploying the networking components in response to a special object created by the installer. It runs very early in the installation process, after the Control Plane (master) nodes have come up, but before the bootstrap control plane has been torn down. It can be indicative of more subtle installer issues, such as long delays in bringing up Control Plane (master) nodes or issues with **apiserver** communication.

Procedure

1. Inspect the pods in the **openshift-network-operator** namespace:

   ```
   [kni@provisioner ~]$ oc get all -n openshift-network-operator
   ```

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

2. On the **provisioner** node, determine that the network configuration exists:

   ```
   [kni@provisioner ~]$ 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
   ```

   If it does not exist, the installer did not create it. To determine why the installer did not create it, execute the following:

   ```
   [kni@provisioner ~]$ openshift-install create manifests
   ```

3. Check that the **network-operator** is running:

   ```
   [kni@provisioner ~]$ kubectl -n openshift-network-operator get pods
   ```

4. Retrieve the logs:

   ```
   [kni@provisioner ~]$ kubectl -n openshift-network-operator logs -l "name=network-operator"
   ```

   On high availability clusters with three or more Control Plane (master) nodes, the Operator will perform leader election and all other Operators will sleep. For additional details, see Troubleshooting.

1.4.8.2. Cluster nodes not getting the correct IPv6 address over DHCP

If the cluster nodes are not getting the correct IPv6 address over DHCP, check the following:

1. Ensure the reserved IPv6 addresses reside outside the DHCP range.
2. In the IP address reservation on the DHCP server, ensure the reservation specifies the correct DHCP Unique Identifier (DUID). For example:

   ```
   # 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. Ensure that Route Announcements are working.
4. Ensure that the DHCP server is listening on the required interfaces serving the IP address ranges.

1.4.8.3. Cluster nodes not getting the correct hostname over DHCP

During IPv6 deployment, cluster nodes must get their hostname over DHCP. Sometimes the **NetworkManager** does not assign the hostname immediately. A Control Plane (master) node might report an error such as:

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

This error indicates that the cluster node likely booted without first receiving a hostname from the DHCP server, which causes **kubelet** to boot with a **localhost.localdomain** hostname. To address the error, force the node to renew the hostname.

Procedure

1. Retrieve the **hostname**:

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

   If the hostname is **localhost**, proceed with the following steps.

   NOTE

   Where **X** is the master node number.

2. Force the cluster node to renew the DHCP lease:

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

   Replace **<bare-metal-nic>** with the wired connection corresponding to the **baremetal** network.

3. Check **hostname** again:

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

4. If the hostname is still **localhost.localdomain**, restart **NetworkManager**:

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

5. If the hostname is still **localhost.localdomain**, wait a few minutes and check again. If the hostname remains **localhost.localdomain**, repeat the previous steps.

6. Restart the **nodeip-configuration** service:

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

   This service will reconfigure the **kubelet** service with the correct hostname references.

7. Reload the unit files definition since the kubelet changed in the previous step:

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

8. Restart the **kubelet** service:

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

9. Ensure **kubelet** booted with the correct hostname:

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

If the cluster node is not getting the correct hostname over DHCP after the cluster is up and running, such as during a reboot, the cluster will have a pending **csr**. **Do not** approve a **csr**, or other issues might arise.

Addressing a **csr**

1. Get CSRs on the cluster:

   ```
   [kni@provisioner ~]$ oc get csr
   ```

2. Verify if a pending **csr** contains **Subject Name: localhost.localdomain**:

   ```
   [kni@provisioner ~]$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' | base64 -d |
   openssl req -noout -text
   ```

3. Remove any **csr** that contains **Subject Name: localhost.localdomain**:

   ```
   [kni@provisioner ~]$ oc delete csr <wrong_csr>
   ```

1.4.8.4. Routes do not reach endpoints

During the installation process, it is possible to encounter a Virtual Router Redundancy Protocol (VRRP) conflict. This conflict might occur if a previously used OpenShift Container Platform node that was once part of a cluster deployment using a specific cluster name is still running but not part of the current OpenShift Container Platform cluster deployment using that same cluster name. For example, a cluster was deployed using the cluster name **openshift**, deploying three Control Plane (master) nodes and three worker nodes. Later, a separate install uses the same cluster name **openshift**, but this redeployment only installed three Control Plane (master) nodes, leaving the three worker nodes from a previous deployment in an **ON** state. This might cause a Virtual Router IDentifier (VRID) conflict and a VRRP conflict.

1. Get the route:

```
[kni@provisioner ~]$ oc get route oauth-openshift
```

2. Check the service endpoint:

```
[kni@provisioner ~]$ 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. Attempt to reach the service from a Control Plane (master) node:

```
[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. Identify the **authentication-operator** errors from the **provisioner** node:

```
[kni@provisioner ~]$ 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"
```

Solution

1. Ensure that the cluster name for every deployment is unique, ensuring no conflict.

2. Turn off all the rogue nodes which are not part of the cluster deployment that are using the same cluster name. Otherwise, the authentication Pod of the OpenShift Container Platform cluster might never start successfully.

1.4.8.5. Failed Ignition during Firstboot

During the Firstboot, the Ignition configuration may fail.

Procedure

1. Connect to the node where the Ignition configuration failed:

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

2. Restart the **machine-config-daemon-firstboot** service:

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

1.4.8.6. NTP out of sync

The deployment of OpenShift Container Platform clusters depends on NTP synchronized clocks among the cluster nodes. Without synchronized clocks, the deployment may fail due to clock drift if the time difference is greater than two seconds.

Procedure

1. Check for differences in the **AGE** of the cluster nodes. For example:

```
[kni@provisioner ~]$ 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. Check for inconsistent timing delays due to clock drift. For example:

```
[kni@provisioner ~]$ oc get bmh -n openshift-machine-api
```

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

```
[kni@provisioner ~]$ 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
```

Addressing clock drift in existing clusters

1. Create a **chrony.conf** file and encode it as **base64** string. For example:

```
[kni@provisioner ~]$ 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) Replace **<NTP-server>** with the IP address of the NTP server. Copy the output.

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

2. Create a **MachineConfig** file, replacing the **base64** string with the **[text-in-base-64]** string generated in the output of the previous step. The following example adds the file to the Control Plane (master) nodes. You can modify the file for worker nodes or make an additional **MachineConfig** for the worker role.

```
[kni@provisioner ~]$ 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
  osImageURL: ""
```

1 Replace **[text-in-base-64]** with the base64 string.

3. Make a backup copy of the configuration file. For example:

```
[kni@provisioner ~]$ cp 99_masters-chrony-configuration.yaml 99_masters-chrony-configuration.yaml.backup
```

4. Apply the configuration file:

```
[kni@provisioner ~]$ oc apply -f ./masters-chrony-configuration.yaml
```

5. Ensure the **System clock synchronized** value is **yes**:

```
[kni@provisioner ~]$ 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
```

To setup clock synchronization prior to deployment, generate the manifest files and add this file to the **openshift** directory. For example:

```
[kni@provisioner ~]$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-chrony-configuration.yaml
```

Then, continue to create the cluster.

1.4.9. Reviewing the installation

After installation, ensure the installer deployed the nodes and pods successfully.

Procedure

1. When the OpenShift Container Platform cluster nodes are installed appropriately, the following **Ready** state is seen within the **STATUS** column:

```
[kni@provisioner ~]$ 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. Confirm all pods are successfully deployed. The following command removes any pods that are still running or have completed as part of the output.

```
[kni@provisioner ~]$ oc get pods --all-namespaces | grep -iv running | grep -iv complete
```