OpenShift Container Platform 4.18

Installing IBM Cloud Bare Metal (Classic)

Installing OpenShift Container Platform on IBM Cloud Bare Metal (Classic)

Last Updated: 2025-06-04

OpenShift Container Platform 4.18 Installing IBM Cloud Bare Metal (Classic)

Installing OpenShift Container Platform on IBM Cloud Bare Metal (Classic)

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document describes how to install OpenShift Container Platform on IBM Cloud Bare Metal (Classic).

Table of Contents

CHAPTER 1. PREREQUISITES

You can use installer-provisioned installation to install OpenShift Container Platform on IBM Cloud® Bare Metal (Classic) nodes. This document describes the prerequisites and procedures when installing OpenShift Container Platform on IBM Cloud® nodes.

IMPORTANT

Red Hat supports IPMI and PXE on the provisioning network only. Red Hat has not tested Red Fish, virtual media, or other complementary technologies such as Secure Boot on IBM Cloud® deployments. A provisioning network is required.

Installer-provisioned installation of OpenShift Container Platform requires:

- One node with Red Hat Enterprise Linux CoreOS (RHCOS) 8.x installed, for running the provisioner
- Three control plane nodes
- One routable network
- One provisioning network

Before starting an installer-provisioned installation of OpenShift Container Platform on IBM Cloud® Bare Metal (Classic), address the following prerequisites and requirements.

1.1. SETTING UP IBM CLOUD BARE METAL (CLASSIC) INFRASTRUCTURE

To deploy an OpenShift Container Platform cluster on IBM Cloud® Bare Metal (Classic) infrastructure, you must first provision the IBM Cloud® nodes.

IMPORTANT

Red Hat supports IPMI and PXE on the **provisioning** network only. Red Hat has not tested Red Fish, virtual media, or other complementary technologies such as Secure Boot on IBM Cloud® deployments. The **provisioning** network is required.

You can customize IBM Cloud® nodes using the IBM Cloud® API. When creating IBM Cloud® nodes, you must consider the following requirements.

Use one data center per cluster

All nodes in the OpenShift Container Platform cluster must run in the same IBM Cloud® data center.

Create public and private VLANs

Create all nodes with a single public VLAN and a single private VLAN.

Ensure subnets have sufficient IP addresses

IBM Cloud® public VLAN subnets use a **/28** prefix by default, which provides 16 IP addresses. That is sufficient for a cluster consisting of three control plane nodes, four worker nodes, and two IP addresses for the API VIP and Ingress VIP on the **baremetal** network. For larger clusters, you might need a smaller prefix.

IBM Cloud® private VLAN subnets use a **/26** prefix by default, which provides 64 IP addresses. IBM

Cloud® Bare Metal (Classic) uses private network IP addresses to access the Baseboard Management Controller (BMC) of each node. OpenShift Container Platform creates an additional subnet for the **provisioning** network. Network traffic for the **provisioning** network subnet routes through the private VLAN. For larger clusters, you might need a smaller prefix.

Table 1.1. IP addresses per prefix

IP addresses	Prefix
32	**/27**
64	**/26**
128	**/25**
256	**/24**

Configuring NICs

OpenShift Container Platform deploys with two networks:

- **provisioning**: The **provisioning** network is a non-routable network used for provisioning the underlying operating system on each node that is a part of the OpenShift Container Platform cluster.
- **baremetal**: The **baremetal** network is a routable network. You can use any NIC order to interface with the **baremetal** network, provided it is not the NIC specified in the **provisioningNetworkInterface** configuration setting or the NIC associated to a node's **bootMACAddress** configuration setting for the **provisioning** network.

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

NIC	Network	VLAN
NIC1	**provisioning**	<provisioning_vlan>
NIC2	**baremetal**	<baremetal_vlan>

In the previous example, NIC1 on all control plane and worker nodes connects to the non-routable network (**provisioning**) that is only used for the installation of the OpenShift Container Platform cluster. NIC2 on all control plane and worker nodes connects to the routable **baremetal** network.

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

NOTE

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

Configuring canonical names

Clients access the OpenShift Container Platform cluster nodes over the **baremetal** network. Configure IBM Cloud® subdomains or subzones where the canonical name extension is the cluster name.

```
<cluster_name>.<domain>
```

For example:

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

Creating DNS entries

You must create DNS **A** record entries resolving to unused IP addresses on the public subnet for the following:

Usage	Host Name	IP
API	api.<cluster_name>.<domain>	<ip>
Ingress LB (apps)	*.apps.<cluster_name>.<domain>	<ip>

Control plane and worker nodes already have DNS entries after provisioning.

The following table provides an example of fully qualified domain names. The API and Nameserver addresses begin with canonical name extensions. The host names of the control plane and worker nodes are examples, so you can use any host naming convention you prefer.

Usage	Host Name	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>

Usage	Host Name	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>

OpenShift Container Platform includes functionality that uses cluster membership information to generate **A** records. This resolves the node names to their IP addresses. After the nodes are registered with the API, the cluster can disperse node information without using CoreDNS-mDNS. This eliminates the network traffic associated with multicast DNS.

IMPORTANT

After provisioning the IBM Cloud® nodes, you must create a DNS entry for the **api.<cluster_name>.<domain>** domain name on the external DNS because removing CoreDNS causes the local entry to disappear. Failure to create a DNS record for the **api.<cluster_name>.<domain>** domain name in the external DNS server prevents worker nodes from joining the cluster.

Network Time Protocol (NTP)

Each OpenShift Container Platform node in the cluster must have access to an NTP server. OpenShift Container Platform nodes use NTP to synchronize their clocks. For example, cluster nodes use SSL certificates that require validation, which might fail if the date and time between the nodes are not in sync.

IMPORTANT

Define a consistent clock date and time format in each cluster node's BIOS settings, or installation might fail.

Configure a DHCP server

IBM Cloud® Bare Metal (Classic) does not run DHCP on the public or private VLANs. After provisioning IBM Cloud® nodes, you must set up a DHCP server for the public VLAN, which corresponds to OpenShift Container Platform's **baremetal** network.

NOTE

The IP addresses allocated to each node do not need to match the IP addresses allocated by the IBM Cloud® Bare Metal (Classic) provisioning system.

See the "Configuring the public subnet" section for details.

Ensure BMC access privileges

The "Remote management" page for each node on the dashboard contains the node's intelligent platform management interface (IPMI) credentials. The default IPMI privileges prevent the user from making certain boot target changes. You must change the privilege level to **OPERATOR** so that Ironic can make those changes.

In the **install-config.yaml** file, add the **privilegelevel** parameter to the URLs used to configure each BMC. See the "Configuring the install-config.yaml file" section for additional details. For example:

```
ipmi://<IP>:<port>?privilegelevel=OPERATOR
```

Alternatively, contact IBM Cloud® support and request that they increase the IPMI privileges to **ADMINISTRATOR** for each node.

Create bare metal servers

Create bare metal servers in the IBM Cloud® dashboard by navigating to **Create resource → Bare Metal Servers for Classic**.

Alternatively, you can create bare metal servers with the **ibmcloud** CLI utility. For example:

```
$ ibmcloud sl hardware create --hostname <SERVERNAME> \
                              --domain <DOMAIN> \
                              --size <SIZE> \
                              --os <OS-TYPE> \
                              --datacenter <DC-NAME> \
                              --port-speed <SPEED> \
                              --billing <BILLING>
```

See Installing the stand-alone IBM Cloud® CLI for details on installing the IBM Cloud® CLI.

NOTE

IBM Cloud® servers might take 3-5 hours to become available.

CHAPTER 2. SETTING UP THE ENVIRONMENT FOR AN OPENSHIFT CONTAINER PLATFORM INSTALLATION

2.1. PREPARING THE PROVISIONER NODE ON IBM CLOUD(R) BARE METAL (CLASSIC) INFRASTRUCTURE

Perform the following steps to prepare the provisioner node.

Procedure

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

   ```
   # useradd kni
   ```

   ```
   # passwd kni
   ```

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

   ```
   # chmod 0440 /etc/sudoers.d/kni
   ```

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

   ```
   # su - kni -c "ssh-keygen -f /home/kni/.ssh/id_rsa -N ""
   ```

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

   ```
   # su - kni
   ```

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

   ```
   $ 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
   ```

   NOTE

   For more information about Red Hat Subscription Manager, see Registering a RHEL system with command-line tools.

6. Install the following packages:

   ```
   $ 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:

   ```
   $ sudo usermod --append --groups libvirt kni
   ```

–

8. Start **firewalld**:

```
$ sudo systemctl start firewalld
```

9. Enable **firewalld**:

```
$ sudo systemctl enable firewalld
```

10. Start the **http** service:

```
$ sudo firewall-cmd --zone=public --add-service=http --permanent
```

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

11. Start and enable the **libvirtd** service:

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

12. Set the ID of the provisioner node:

```
$ PRVN_HOST_ID=<ID>
```

You can view the ID with the following **ibmcloud** command:

```
$ ibmcloud sl hardware list
```

13. Set the ID of the public subnet:

```
$ PUBLICSUBNETID=<ID>
```

You can view the ID with the following **ibmcloud** command:

```
$ ibmcloud sl subnet list
```

14. Set the ID of the private subnet:

```
$ PRIVSUBNETID=<ID>
```

You can view the ID with the following **ibmcloud** command:

```
$ ibmcloud sl subnet list
```

15. Set the provisioner node public IP address:

```
$ PRVN_PUB_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | jq
.primaryIpAddress -r)
```

16. Set the CIDR for the public network:

```
$ PUBLICCIDR=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq .cidr)
```

17. Set the IP address and CIDR for the public network:

    ```
    $ PUB_IP_CIDR=$PRVN_PUB_IP/$PUBLICCIDR
    ```

18. Set the gateway for the public network:

    ```
    $ PUB_GATEWAY=$(ibmcloud sl subnet detail $PUBLICSUBNETID --output JSON | jq
    .gateway -r)
    ```

19. Set the private IP address of the provisioner node:

    ```
    $ PRVN_PRIV_IP=$(ibmcloud sl hardware detail $PRVN_HOST_ID --output JSON | \
                 jq .primaryBackendIpAddress -r)
    ```

20. Set the CIDR for the private network:

    ```
    $ PRIVCIDR=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq .cidr)
    ```

21. Set the IP address and CIDR for the private network:

    ```
    $ PRIV_IP_CIDR=$PRVN_PRIV_IP/$PRIVCIDR
    ```

22. Set the gateway for the private network:

    ```
    $ PRIV_GATEWAY=$(ibmcloud sl subnet detail $PRIVSUBNETID --output JSON | jq
    .gateway -r)
    ```

23. Set up the bridges for the **baremetal** and **provisioning** networks:

    ```
    $ sudo nohup bash -c "
       nmcli --get-values UUID con show | xargs -n 1 nmcli con delete
       nmcli connection add ifname provisioning type bridge con-name provisioning
       nmcli con add type bridge-slave ifname eth1 master provisioning
       nmcli connection add ifname baremetal type bridge con-name baremetal
       nmcli con add type bridge-slave ifname eth2 master baremetal
       nmcli connection modify baremetal ipv4.addresses $PUB_IP_CIDR ipv4.method manual
    ipv4.gateway $PUB_GATEWAY
       nmcli connection modify provisioning ipv4.addresses 172.22.0.1/24,$PRIV_IP_CIDR
    ipv4.method manual
       nmcli connection modify provisioning +ipv4.routes \"10.0.0.0/8 $PRIV_GATEWAY\"
       nmcli con down baremetal
       nmcli con up baremetal
       nmcli con down provisioning
       nmcli con up provisioning
       init 6
    "
    ```

    NOTE

    For **eth1** and **eth2**, substitute the appropriate interface name, as needed.

24. If required, SSH back into the **provisioner** node:

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

25. Verify the connection bridges have been properly created:

```
$ sudo nmcli con show
```

Example output

```
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-eth1  76a8ed50-c7e5-4999-b4f6-6d9014dd0812  ethernet  eth1
bridge-slave-eth2  f31c3353-54b7-48de-893a-02d2b34c4736  ethernet  eth2
```

26. Create a **pull-secret.txt** file:

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

In a web browser, navigate to Install on Bare Metal with user-provisioned infrastructure . In step 1, click **Download pull secret** Paste the contents into the **pull-secret.txt** file and save the contents in the **kni** user's home directory.

2.2. CONFIGURING THE PUBLIC SUBNET

All of the OpenShift Container Platform cluster nodes must be on the public subnet. IBM Cloud® Bare Metal (Classic) does not provide a DHCP server on the subnet. Set it up separately on the provisioner node.

You must reset the BASH variables defined when preparing the provisioner node. Rebooting the provisioner node after preparing it will delete the BASH variables previously set.

Procedure

1. Install **dnsmasq**:

```
$ sudo dnf install dnsmasq
```

2. Open the **dnsmasq** configuration file:

```
$ sudo vi /etc/dnsmasq.conf
```

3. Add the following configuration to the **dnsmasq** configuration file:

```
interface=baremetal
except-interface=lo
bind-dynamic
log-dhcp

dhcp-range=<ip_addr>,<ip_addr>,<pub_cidr> 1
```

```
dhcp-option=baremetal,121,0.0.0.0/0,<pub_gateway>,<prvn_priv_ip>,<prvn_pub_ip> 2

dhcp-hostsfile=/var/lib/dnsmasq/dnsmasq.hostsfile
```

1. Set the DHCP range. Replace both instances of **<ip_addr>** with one unused IP address from the public subnet so that the **dhcp-range** for the **baremetal** network begins and ends with the same the IP address. Replace **<pub_cidr>** with the CIDR of the public subnet.

2. Set the DHCP option. Replace **<pub_gateway>** with the IP address of the gateway for the **baremetal** network. Replace **<prvn_priv_ip>** with the IP address of the provisioner node's private IP address on the **provisioning** network. Replace **<prvn_pub_ip>** with the IP address of the provisioner node's public IP address on the **baremetal** network.

To retrieve the value for **<pub_cidr>**, execute:

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .cidr
```

Replace **<publicsubnetid>** with the ID of the public subnet.

To retrieve the value for **<pub_gateway>**, execute:

```
$ ibmcloud sl subnet detail <publicsubnetid> --output JSON | jq .gateway -r
```

Replace **<publicsubnetid>** with the ID of the public subnet.

To retrieve the value for **<prvn_priv_ip>**, execute:

```
$ ibmcloud  sl hardware detail <id> --output JSON | \
          jq .primaryBackendIpAddress -r
```

Replace **<id>** with the ID of the provisioner node.

To retrieve the value for **<prvn_pub_ip>**, execute:

```
$ ibmcloud sl hardware detail <id> --output JSON | jq .primaryIpAddress -r
```

Replace **<id>** with the ID of the provisioner node.

4. Obtain the list of hardware for the cluster:

```
$ ibmcloud sl hardware list
```

5. Obtain the MAC addresses and IP addresses for each node:

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '.networkComponents[] | \
  "\(.primaryIpAddress) \(.macAddress)"' | grep -v null
```

Replace **<id>** with the ID of the node.

Example output

```
"10.196.130.144 00:e0:ed:6a:ca:b4"
"141.125.65.215 00:e0:ed:6a:ca:b5"
```

Make a note of the MAC address and IP address of the public network. Make a separate note of the MAC address of the private network, which you will use later in the **install-config.yaml** file. Repeat this procedure for each node until you have all the public MAC and IP addresses for the public **baremetal** network, and the MAC addresses of the private **provisioning** network.

6. Add the MAC and IP address pair of the public **baremetal** network for each node into the **dnsmasq.hostsfile** file:

```
$ sudo vim /var/lib/dnsmasq/dnsmasq.hostsfile
```

Example input

```
00:e0:ed:6a:ca:b5,141.125.65.215,master-0
<mac>,<ip>,master-1
<mac>,<ip>,master-2
<mac>,<ip>,worker-0
<mac>,<ip>,worker-1
...
```

Replace **<mac>,<ip>** with the public MAC address and public IP address of the corresponding node name.

7. Start **dnsmasq**:

```
$ sudo systemctl start dnsmasq
```

8. Enable **dnsmasq** so that it starts when booting the node:

```
$ sudo systemctl enable dnsmasq
```

9. Verify **dnsmasq** is running:

```
$ sudo systemctl status dnsmasq
```

Example output

```
● dnsmasq.service - DNS caching server.
Loaded: loaded (/usr/lib/systemd/system/dnsmasq.service; enabled; vendor preset: disabled)
Active: active (running) since Tue 2021-10-05 05:04:14 CDT; 49s ago
Main PID: 3101 (dnsmasq)
Tasks: 1 (limit: 204038)
Memory: 732.0K
CGroup: /system.slice/dnsmasq.service
 └─3101 /usr/sbin/dnsmasq -k
```

10. Open ports **53** and **67** with UDP protocol:

```
$ sudo firewall-cmd --add-port 53/udp --permanent
$ sudo firewall-cmd --add-port 67/udp --permanent
```

11. Add **provisioning** to the external zone with masquerade:

```
$ sudo firewall-cmd --change-zone=provisioning --zone=external --permanent
```

This step ensures network address translation for IPMI calls to the management subnet.

12. Reload the **firewalld** configuration:

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

2.3. RETRIEVING THE OPENSHIFT CONTAINER PLATFORM INSTALLER

Use the **stable-4.x** version of the installation program and your selected architecture to deploy the generally available stable version of OpenShift Container Platform:

```
$ export VERSION=stable-4.18
```

```
$ export RELEASE_ARCH=<architecture>
```

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

2.4. EXTRACTING THE OPENSHIFT CONTAINER PLATFORM INSTALLER

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

Procedure

1. Set the environment variables:

```
$ export cmd=openshift-baremetal-install
```

```
$ export pullsecret_file=~/pull-secret.txt
```

```
$ export extract_dir=$(pwd)
```

2. Get the **oc** binary:

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

3. Extract the installer:

```
$ 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
```

2.5. 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 IBM Cloud® Bare Metal (Classic) hardware so that it is able to fully manage it. The material difference between installing on bare metal and installing on IBM Cloud® Bare Metal (Classic) is that you must explicitly set the privilege level for IPMI in the BMC section of the **install-config.yaml** file.

Procedure

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:
  machineNetwork:
  - cidr: <public-cidr>
  networkType: OVNKubernetes
compute:
- name: worker
  replicas: 2
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://10.196.130.145?privilegelevel=OPERATOR 1
          username: root
          password: <password>
        bootMACAddress: 00:e0:ed:6a:ca:b4 2
        rootDeviceHints:
          deviceName: "/dev/sda"
      - name: openshift-worker-0
        role: worker
        bmc:
          address: ipmi://<out-of-band-ip>?privilegelevel=OPERATOR 3
      username: <user>
      password: <password>
    bootMACAddress: <NIC1_mac_address> ④
    rootDeviceHints:
      deviceName: "/dev/sda"
pullSecret: '<pull_secret>'
sshKey: '<ssh_pub_key>'
```

① ③ The **bmc.address** provides a **privilegelevel** configuration setting with the value set to **OPERATOR**. This is required for IBM Cloud® Bare Metal (Classic) infrastructure.

② ④ Add the MAC address of the private **provisioning** network NIC for the corresponding node.

NOTE

You can use the **ibmcloud** command-line utility to retrieve the password.

```
$ ibmcloud sl hardware detail <id> --output JSON | \
  jq '"(.networkManagementIpAddress)
(.remoteManagementAccounts[0].password)"'
```

Replace **<id>** with the ID of the node.

2. Create a directory to store the cluster configuration:

```
$ mkdir ~/clusterconfigs
```

3. Copy the **install-config.yaml** file into the directory:

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

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

```
$ ipmitool -I lanplus -U <user> -P <password> -H <management_server_ip> power off
```

5. 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 $i;
  sudo virsh vol-delete $i.ign --pool $i;
  sudo virsh pool-destroy $i;
  sudo virsh pool-undefine $i;
done
```

2.6. 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 2.1. Required parameters

Parameters	Default	Description
baseDomain		The domain name for the cluster. For example, **example.com**.
bootMode	**UEFI**	The boot mode for a node. Options are **legacy**, **UEFI**, and **UEFISecureBoot**. If **bootMode** is not set, Ironic sets it while inspecting the node.

`platform:`
`  baremetal:`
`    bootstrapExternalStaticDNS`
		The static network DNS of the bootstrap node. You must set this value when deploying a cluster with static IP addresses when there is no Dynamic Host Configuration Protocol (DHCP) server on the bare-metal network. If you do not set this value, the installation program will use the value from **bootstrapExternalStaticGateway**, which causes problems when the IP address values of the gateway and DNS are different.

`platform:`
`  baremetal:`
`    bootstrapExternalStaticIP`
		The static IP address for the bootstrap VM. You must set this value when deploying a cluster with static IP addresses when there is no DHCP server on the bare-metal network.

`platform:`
`  baremetal:`
`    bootstrapExternalStaticGateway`
		The static IP address of the gateway for the bootstrap VM. You must set this value when deploying a cluster with static IP addresses when there is no DHCP server on the bare-metal network.

sshKey		The **sshKey** configuration setting has the key in the **~/.ssh/id_rsa.pub** file required to access the control plane nodes and compute nodes. Typically, this key is from the **provisioner** node.
pullSecret		The **pullSecret** configuration setting has a copy of the pull secret downloaded from the Install OpenShift on Bare Metal page when preparing the provisioner node.

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

Parameters	Default	Description

`networking:`
`  machineNetwork:`
`  - cidr:`
		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 you to provide a name for compute nodes even if there are zero nodes.

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

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

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

provisioningNetworkInterface		The name of the network interface on nodes connected to the provisioning network. For OpenShift Container Platform 4.9 and later releases, use the **bootMACAddress** configuration setting to enable Ironic to identify the IP address of the NIC instead of using the **provisioningNetworkInterface** configuration setting to identify the name of the NIC.
defaultMachinePlatform		The default configuration used for machine pools without a platform configuration.

Parameters	Default	Description
apiVIPs		(Optional) The virtual IP address for Kubernetes API communication. You must either provide this setting in the **install-config.yaml** file as a reserved IP from the **MachineNetwork** parameter or preconfigured in the DNS so that the default name resolves correctly. Use the virtual IP address and not the FQDN when adding a value to the **apiVIPs** configuration setting in the **install-config.yaml** file. The primary IP address must be from the IPv4 network when using dual stack networking. If not set, the installation program uses **api.<cluster_name>.<base_domain>** to derive the IP address from the DNS. NOTE Before OpenShift Container Platform 4.12, the cluster installation program only accepted an IPv4 address or an IPv6 address for the **apiVIP** configuration setting. From OpenShift Container Platform 4.12 or later, the **apiVIP** configuration setting is deprecated. Instead, use a list format for the **apiVIPs** configuration setting to specify an IPv4 address, an IPv6 address or both IP address formats.
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.

Parameters	Default	Description
ingressVIPs		(Optional) The virtual IP address for ingress traffic. You must either provide this setting in the **install-config.yaml** file as a reserved IP from the **MachineNetwork** parameter or preconfigured in the DNS so that the default name resolves correctly. Use the virtual IP address and not the FQDN when adding a value to the **ingressVIPs** configuration setting in the **install-config.yaml** file. The primary IP address must be from the IPv4 network when using dual stack networking. If not set, the installation program uses **test.apps.<cluster_name>.<base_domain>** to derive the IP address from the DNS. NOTE Before OpenShift Container Platform 4.12, the cluster installation program only accepted an IPv4 address or an IPv6 address for the **ingressVIP** configuration setting. In OpenShift Container Platform 4.12 and later, the **ingressVIP** configuration setting is deprecated. Instead, use a list format for the **ingressVIPs** configuration setting to specify an IPv4 addresses, an IPv6 addresses or both IP address formats.

Table 2.2. Optional Parameters

Parameters	Default	Description

platform:
  baremetal:
    additionalNTPServers:
      - <ip_address_or_domain_name>
		An optional list of additional NTP servers to add to each host. You can use an IP address or a domain name to specify each NTP server. Additional NTP servers are user-defined NTP servers that enable preinstallation clock synchronization when the cluster host clocks are out of synchronization.

provisioningDHCPRange	**172.22.0.10,172.22.0.100**	Defines the IP range for nodes on the provisioning network.

Parameters	Default	Description
provisioningNetworkCIDR	**172.22.0.0/24**	The CIDR for the network to use for provisioning. The installation program requires this option when not using the default address range on the provisioning network.
clusterProvisioningIP	The third IP address of the **provisioningNetworkCIDR**.	The IP address within the cluster where the provisioning services run. Defaults to the third IP address of the provisioning subnet. For example, **172.22.0.3**.
bootstrapProvisioningIP	The second IP address of the **provisioningNetworkCIDR**.	The IP address on the bootstrap VM where the provisioning services run while the installation program is deploying the control plane (master) nodes. Defaults to the second IP address of the provisioning subnet. For example, **172.22.0.2** or **2620:52:0:1307::2**.
externalBridge	**baremetal**	The name of the bare-metal bridge of the hypervisor attached to the bare-metal network.
provisioningBridge	**provisioning**	The name of the provisioning bridge on the **provisioner** host attached to the provisioning network.
architecture		Defines the host architecture for your cluster. Valid values are **amd64** or **arm64**.
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>**.

Parameters	Default	Description
provisioningNetwork		The **provisioningNetwork** configuration setting determines whether the cluster uses the provisioning network. If it does, the configuration setting also determines if the cluster manages the network. **Disabled**: Set this parameter to **Disabled** to disable the requirement for a provisioning network. When set to **Disabled**, you must only use virtual media based provisioning, or start the cluster by using the Assisted Installer. If set to **Disabled** and using power management, BMCs must be accessible from the bare-metal network. If set to **Disabled**, you must provide two IP addresses on the bare-metal network that the installation program uses for the provisioning services. **Managed**: Set this parameter to **Managed**, which is the default, to fully manage the provisioning network, including DHCP, TFTP, and so on. **Unmanaged**: Set this parameter to **Unmanaged** to enable the provisioning network but take care of manual configuration of DHCP. Virtual media provisioning is recommended but PXE is still available if required.
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.

Table 2.3. Hosts

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** (control plane node) or **worker** (compute node).
bmc		Connection details for the baseboard management controller. See the BMC addressing section for additional details.

Name	Default	Description
bootMACAddress		The MAC address of the NIC that the host uses for the provisioning network. Ironic retrieves the IP address using the **bootMACAddress** configuration setting. Then, it binds to the host. NOTE You must provide a valid MAC address from the host if you disabled the provisioning network.
networkConfig		Set this optional parameter to configure the network interface of a host. See "(Optional) Configuring host network interfaces" for additional details.

2.7. 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 2.4. Subfields

Subfield	Description
deviceName	A string containing a Linux device name such as **/dev/vda** or **/dev/disk/by-path/**. NOTE It is recommended to use the **/dev/disk/by-path/<device_path>** link to the storage location. 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.

Subfield	Description
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.
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"
```

2.8. CREATING THE OPENSHIFT CONTAINER PLATFORM MANIFESTS

1. Create the OpenShift Container Platform manifests.

```
$ ./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
```

2.9. DEPLOYING THE CLUSTER VIA THE OPENSHIFT CONTAINER PLATFORM INSTALLER

Run the OpenShift Container Platform installer:

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

2.10. FOLLOWING THE PROGRESS OF 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:

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