Red Hat OpenStack Platform 9

Bare Metal Provisioning

Install, Configure, and Use Bare Metal Provisioning (Ironic)

Last Updated: 2018-01-20

Red Hat OpenStack Platform 9 Bare Metal Provisioning

Install, Configure, and Use Bare Metal Provisioning (Ironic)

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

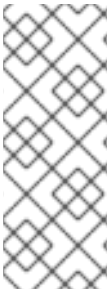
This guide provides procedures for installing, configuring, and using Bare Metal Provisioning in the Overcloud of a Red Hat OpenStack Platform environment.

Table of Contents

PREFACE

This document provides instructions for installing and configuring Bare Metal Provisioning (ironic) as an OpenStack service in the overcloud, and using the service to provision and manage physical machines for end users. Configuring this service allows users to launch instances on physical machines in the same way that they launch virtual machine instances.

The Bare Metal Provisioning components are also used by the Red Hat OpenStack Platform director, as part of the undercloud, to provision and manage the bare metal nodes that make up the OpenStack environment (the overcloud). For information on how the director uses Bare Metal Provisioning, see Director Installation and Usage.



NOTE

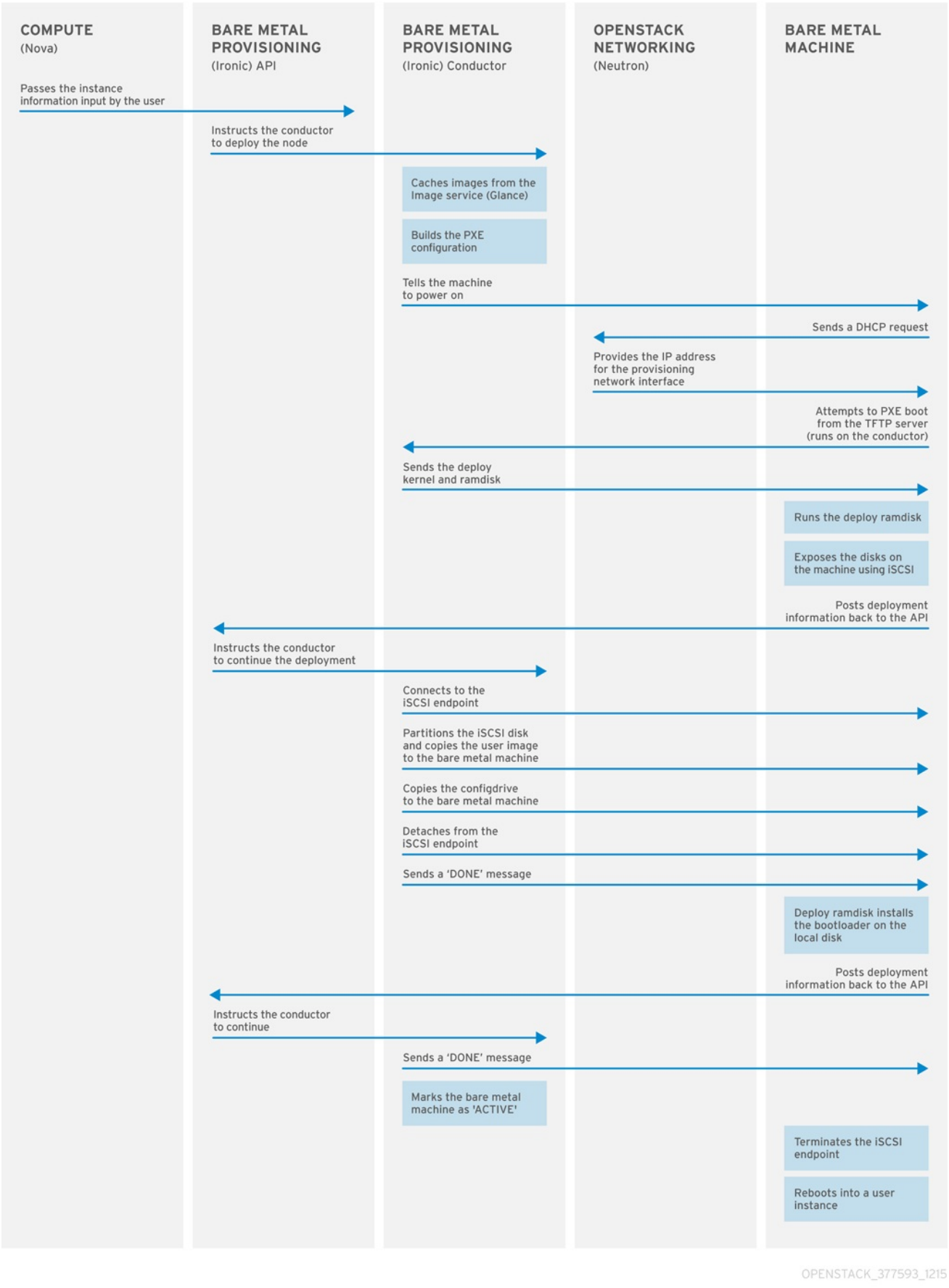
This document describes how to manually install Bare Metal Provisioning on an already deployed overcloud. This manual procedure differs from how Bare Metal Provisioning is integrated into later versions of Red Hat OpenStack Platform director. Attempting to upgrade an overcloud that has undergone this process is unsupported and the results are undefined. This document should be treated as a guide to preview what is integrated into later versions of Red Hat OpenStack Platform director.

CHAPTER 1. INSTALL AND CONFIGURE OPENSTACK BARE METAL PROVISIONING (IRONIC)

OpenStack Bare Metal Provisioning (ironic) provides the components required to provision and manage physical (bare metal) machines for end users. Bare Metal Provisioning in the overcloud interacts with the following OpenStack services:

- OpenStack Compute (nova) provides scheduling, tenant quotas, IP assignment, and a user-facing API for virtual machine instance management, while Bare Metal Provisioning provides the administrative API for hardware management. Choose a single, dedicated **openstack-nova-compute** host to use the Bare Metal Provisioning drivers and handle Bare Metal Provisioning requests.
- OpenStack Identity (keystone) provides request authentication and assists Bare Metal Provisioning in locating other OpenStack services.
- OpenStack Image service (glance) manages images and image metadata used to boot bare metal machines.
- OpenStack Networking (neutron) provides DHCP and network configuration for the required Bare Metal Provisioning networks.

Bare Metal Provisioning uses PXE to provision physical machines. The following diagram outlines how the OpenStack services interact during the provisioning process when a user launches a new bare metal machine.



1.1. REQUIREMENTS

This chapter outlines the requirements for setting up Bare Metal Provisioning, including installation assumptions, hardware requirements, and networking requirements.

1.1.1. Bare Metal Provisioning Installation Assumptions

Bare Metal Provisioning is a collection of components that can be configured to run on the same node or on separate nodes. The configuration examples in this guide install and configure all Bare Metal Provisioning components on a single node. This guide assumes that the services for OpenStack Identity, OpenStack Image, OpenStack Compute, and OpenStack Networking have already been installed and configured. Bare Metal Provisioning also requires the following external services, which must also be installed and configured as a prerequisite:

- A database server in which to store hardware information and state. This guide assumes that the MariaDB database service is configured for the Red Hat OpenStack Platform environment.
- A messaging service. This guide assumes that RabbitMQ is configured for the environment.

If you used the director to deploy your OpenStack environment, the database and messaging services are installed on a controller node in the overcloud.

Red Hat OpenStack Platform requires **iptables** instead of **firewalld** on Compute nodes and OpenStack Networking nodes running Red Hat Enterprise Linux 7. Firewall rules in this document are set using **iptables**.

NOTE

Hardware inspection (*ironic-inspector*) uses **iptables** to blacklist the MAC addresses of ironic nodes. In the event that another process has locked **iptables** while *ironic-inspector* is attempting to make a modification, *ironic-inspector* uses the **iptables** `-w` flag, where supported (version 1.4.21, or higher).

1.1.2. Bare Metal Provisioning Hardware Requirements

A node running all Bare Metal Provisioning components requires the following hardware:

- 64-bit x86 processor with support for the Intel 64 or AMD64 CPU extensions.
- A minimum of 6 GB of RAM.
- A minimum of 40 GB of available disk space.
- A minimum of two 1 Gbps Network Interface Cards. However, a 10 Gbps interface is recommended for Bare Metal Provisioning Network traffic, especially if you are provisioning a large number of bare metal machines.
- Red Hat Enterprise Linux 7.2 (or later) installed as the host operating system.

Alternatively, install and configure Bare Metal Provisioning components on a dedicated **openstack-nova-compute** node; see Compute Node Requirements in the *Director Installation and Usage* guide for hardware requirements.

1.1.3. Bare Metal Provisioning Networking Requirements

Bare Metal Provisioning requires at least two networks:

- Provisioning network: This is a private network that Bare Metal Provisioning uses to provision and manage bare metal machines. The Bare Metal Provisioning Network provides DHCP and PXE boot functions to help discover bare metal systems. This network should ideally use a

native VLAN on a trunked interface so that Bare Metal Provisioning serves PXE boot and DHCP requests. This is also the network used to control power management through out-of-band drivers on the bare metal machines to be provisioned.

- External network: A separate network for remote connectivity. The interface connecting to this network requires a routable IP address, either defined statically or dynamically through an external DHCP service.

1.1.4. Bare Metal Machine Requirements

Bare metal machines that will be provisioned require the following:

- Two NICs: one for the Bare Metal Provisioning Network, and one for external connectivity.
- A power management interface (e.g. IPMI) connected to the Bare Metal Provisioning Network. If you are using SSH for testing purposes, this is not required.
- PXE boot on the Bare Metal Provisioning Network at the top of the system's boot order, ahead of hard disks and CD/DVD drives. Disable PXE boot on all other NICs on the system.

Other requirements for bare metal machines that will be provisioned vary depending on the operating system you are installing. For Red Hat Enterprise Linux 7, see the Red Hat Enterprise Linux 7 Installation Guide. For Red Hat Enterprise Linux 6, see the Red Hat Enterprise Linux 6 Installation Guide.

1.2. CONFIGURE OPENSTACK FOR THE BARE METAL PROVISIONING SERVICE

Every OpenStack service has a user name and password that is used to authenticate it with the Identity service. Each service also needs to be defined with the OpenStack Identity service and have an endpoint URL associated with it for Internal, Admin and Public connectivity.

To configure the Bare Metal Provisioning Service from the director node:

1. Source the **overcloudrc** file:

```
# source ~stack/overcloudrc
```

2. Create the OpenStack Bare Metal Provisioning user:

```
# openstack user create --password IRONIC_PASSWORD --enable
IRONIC_USER
# openstack role add --project service --user IRONIC_USER admin
```

Here, **IRONIC_USER** is the user for the Bare Metal Provisioning service and **IRONIC_PASSWORD** is the password.

3. Create the OpenStack Bare Metal Provisioning service:

```
# openstack service create --name ironic --description "Ironic bare
metal provisioning service" baremetal
```

4. Verify the virtual IP (VIP) address that the other OpenStack services are using:

```
# openstack endpoint list -c "Service Name" -c "PublicURL" --long
```

The output of this command lists the services and their **`Public URL`**, which are usually all on the same server and use the same IP address.

5. Get the Internal API network address of the Compute node that you are installing the Bare Metal Provisioning service on:

```
# route -n
```

The output of this command lists the IP routing table with the IP addresses and the **`interface`** for each of the IP addresses.

The Internal API network address is then used to create a service endpoint.

6. You can check the IP address associated with the NIC to use for the Internal and Admin URLs as follows:

```
# ifconfig INTERFACE
```

7. Create the service endpoint:

```
# openstack endpoint create --publicurl http://VIP:6385 --
internalurl http://COMPUTE_INTERNAL_API_IP:6385 --adminurl
http://COMPUTE_INTERNAL_API_IP:6385 --region regionOne SERVICE_ID
```

Here, **`VIP`** is the virtual IP address configured in HAProxy, **`COMPUTE_INTERNAL_API_IP`** is the IP address for the Compute node running the Bare Metal Provisioning service that is connected to the Internal API network and **`SERVICE_ID`** is the ID of the Bare Metal Provisioning service created using the **`service create`** command.

Next, you must configure the HAProxy to make sure you receive requests for the Public URL for the endpoints you created in the previous procedure. To configure the HAProxy value, ensure that you are logged in as the **`root`** user on your controller nodes.

1. Edit the **`/etc/haproxy/haproxy.cfg`** file and add the following line at the end of the file:

```
listen ironic

  bind VIP:6385 transparent
  server SERVER_NAME COMPUTE_INTERNAL_API_IP:6385 check fall 5 inter
2000 rise 2
```

In this example:

- **`VIP`** is the virtual IP address.
- **`SERVER_NAME`** is the HAProxy identifying name for the Compute server where the Bare Metal Provisioning service will be installed and running.
- **`COMPUTE_INTERNAL_API_IP`** is the Internal API IP address of the Compute server where the Bare Metal Provisioning service will be installed and running.
- **`transparent`** allows the HAProxy to bind the IP address even if it does not exist on the Controller node so that in a clustered environment, the virtual IP address can move between controllers.

- **`check fall 5 inter 2000 rise 2`** refers to the following health checks on the back end server:
  - **`fall 5`** - the server is considered unavailable after 5 consecutive failed health checks.
  - **`inter 2000`** - the interval between health checks is 2000 ms or 2 seconds.
  - **`rise 2`** - the server is considered available after 2 consecutive successful health checks.

2. Restart the HAProxy to make sure the changes take effect:

```
# systemctl restart haproxy.service
```

You can get the following message stating the back end server is not available: *haproxy[4249]: proxy ironic has no server available!*. This message can be ignored for now, since you have not yet installed or configured the service.

1.3. CONFIGURE THE CONTROLLER NODES FOR BARE METAL PROVISIONING SERVICE

The following steps need to be performed on all the controller nodes in your Red Hat OpenStack Platform deployment as a **`root`** user, with the exception of the *Create the Bare Metal Provisioning Database* section. You must perform that procedure on one controller since they all share the database.

On the Controller nodes, you need to make sure your Bare Metal Provisioning Network is connected to Open vSwitch so your OpenStack deployment can reach it.

1. Add a bridge into Open vSwitch:

```
# ovs-vsctl add-br br-ironic
# ovs-vsctl add-port br-ironic IRONIC_PROVISIONING_NIC
# ovs-vsctl show
```

Here **`br-ironic`** is the name of the bridge and **`IRONIC_PROVISIONING_NIC`** is the NIC connected to the Bare Metal Provisioning Network.

With the **`ovs-vsctl show`** command, you can see that a new bridge is created with the associated port, however you will notice that the **`br-int`** integration bridge lacks a patch to the new bridge.

2. To get the new bridge added to the integration bridge, you need to update the following plugin files:

   a. Update the ML2 configuration file, **`/etc/neutron/plugins/ml2/ml2_conf.ini`** as follows:

   - For the **`type_drivers`** parameter, make sure **`flat`** is listed among the drivers, for example, **`type_drivers = vxlan,vlan,flat,gre`**. This is a comma delimited list.
   - For the **`mechanism_drivers`** parameter, make sure **`openvswitch`** option is listed among the drivers, for example, **`mechanism_drivers =openvswitch`**. This is a comma delimited list.
   - For the **`flat_networks`** parameters, create a name to refer to the Bare Metal

Provisioning Network, for example, **ironicnet**. Make sure this name is listed among the **flat_networks** listed, for example, **flat_networks =datacentre,ironicnet**. This is a comma delimited list.

- If you are using a VLAN for the Bare Metal Provisioning Network, add the **network_vlan_ranges** parameter with the following format: **ironicnet:VLAN_START:VLAN_END**, for example, **network_vlan_ranges =datacentre:1:1000**. This is a comma delimited list.

- The **enable_security_group** parameter should already be enabled. But if it is not set, change the value to **True**, for example, **enable_security_group = True**.

b. In the **/etc/neutron/plugins/ml2/openvswitch_agent.ini** file, find the **bridge_mappings** parameter and update as follows:

```
bridge_mappings =datacentre:br-ex,ironicnet:br-ironic
```

The value of this comma delimited key-value pair maps the name of the Bare Metal Provisioning Network to the physical device which is connected to the network.

3. Restart the **neutron-openvswitch-agent.service** to see the **br-ironic** bridge as a part of the integration bridge:

```
# systemctl restart neutron-openvswitch-agent.service
```

4. Restart the **neutron-server.service** so that it detects the new connection:

```
# systemctl restart neutron-server.service
```

NOTE

If you do not perform this step, trying to create the Bare Metal Provisioning Network within the OpenStack Networking service will fail with a message that the requested flat network does not exist.

1.3.1. Create the Bare Metal Provisioning Database

Create the database and database user used by Bare Metal Provisioning. All steps in this procedure must be performed on the database server, while logged in as the **root** user.

Creating the Bare Metal Provisioning Database

1. Connect to the database service:

```
# mysql -u root
```

2. Create the **ironic** database:

```
mysql> CREATE DATABASE ironic CHARACTER SET utf8;
```

3. Create an **ironic** database user and grant the user access to the**ironic** database:

```
mysql> GRANT ALL PRIVILEGES ON ironic.* TO 'ironic'@'%' IDENTIFIED
```

```
BY 'PASSWORD';
mysql> GRANT ALL PRIVILEGES ON ironic.* TO 'ironic'@'localhost'
IDENTIFIED BY 'PASSWORD';
```

Replace *PASSWORD* with a secure password that will be used to authenticate with the database server as this user.

4. Flush the database privileges to ensure that they take effect immediately:

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the mysql client:

```
mysql> quit
```

1.3.2. Configure OpenStack Compute Services For Bare Metal Provisioning

Configure Compute services for the Bare Metal Provisioning driver. Using this driver enables Compute to provision physical machines using the same API that is used to provision virtual machines. Only one driver can be specified for each **openstack-nova-compute** node; a node with the Bare Metal Provisioning driver can provision only physical machines. It is recommended that you allocate a single **openstack-nova-compute** node to provision all bare metal nodes using the Bare Metal Provisioning driver. All steps in the following procedure must be performed on a chosen compute node, while logged in as the **root** user.

Configuring OpenStack Compute for the Bare Metal Provisioning

1. Set Compute to use the Bare Metal Provisioning scheduler host manager:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT scheduler_host_manager
nova.scheduler.ironic_host_manager.IronicHostManager
```

2. Disable the Compute scheduler from tracking changes in instances:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT
scheduler_tracks_instance_changes false
```

3. Set the default filters as follows:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT
baremetal_scheduler_default_filters
AvailabilityZoneFilter,ComputeFilter,ComputeCapabilitiesFilter,Image
PropertiesFilter
```

4. Set Compute to use default Bare Metal Provisioning scheduling filters:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT scheduler_use_baremetal_filters True
```

5. Set Compute to use the correct authentication details for Bare Metal Provisioning:

```
# openstack-config --set /etc/nova/nova.conf \
```

```
   ironic admin_username ironic
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_password PASSWORD
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_url http://IDENTITY_IP:35357/v2.0
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf \
   ironic api_endpoint http://IRONIC_API_IP:6385/v1
```

Replace the following values:

- Replace *PASSWORD* with the password that Bare Metal Provisioning uses to authenticate with Identity.
- Replace *IDENTITY_IP* with the IP address or host name of the server hosting Identity.
- Replace *IRONIC_API_IP* with the IP address or host name of the server hosting the Bare Metal Provisioning API service.

6. Set the **nova** database credentials on the **ironic** Compute node:

```
# openstack-config --set /etc/nova/nova.conf database connection
"mysql+pymysql://nova:NOVA_DB_PASSWORD@DB_IP/nova"
```

7. Restart the Compute scheduler service on the Compute controller nodes:

```
# systemctl restart openstack-nova-scheduler.service
```

8. Restart the compute service on the compute nodes:

```
# systemctl restart openstack-nova-compute.service
```

1.3.3. Configure the OpenStack Networking DHCP Agent to Tag iPXE Requests

OpenStack Networking DHCP requests from iPXE need to have a DHCP tag called **ipxe** to let the DHCP server know that the client needs to perform an HTTP operation to get the **boot.ipxe** script. You can do this by adding a **dhcp-userclass** entry to the **dnsmasq** configuration file used by the OpenStack Networking DHCP Agent service.

1. On your overcloud controller, verify which **dnsmasq** file the DHCP Agent is using:

```
# grep ^dnsmasq_config_file /etc/neutron/dhcp_agent.ini

dnsmasq_config_file =/etc/neutron/dnsmasq-neutron.conf
```

2. Edit this file and add the following lines at the end of the file:

```
# Create the "ipxe" tag if request comes from iPXE user class

dhcp-userclass=set:ipxe,iPXE
```

3. Save the file and restart the OpenStack Networking DHCP Agent service:

```
# systemctl restart neutron-dhcp-agent.service
```

1.4. CONFIGURE THE COMPUTE NODE FOR BARE METAL PROVISIONING

The following instructions here apply **ONLY** to the Compute node that is also running the Bare Metal Provisioning service. You need to perform these steps as a **root** user on the Compute node.

On the Compute node, you have the Bare Metal Provisioning NIC, for example, **eth6**. The goals with this procedure are as follows:

1. To connect the Bare Metal Provisioning NIC, **eth6** in this example, to Open vSwitch.
2. To assign an IP address on this connection as the Bare Metal server needs to pull down the boot images from the bare metal node as a part of the iPXE process.

Connecting eth6 to Open vSwitch

1. As with the Controller node in Section 1.3, “Configure the Controller Nodes for Bare Metal Provisioning Service”, create a bridge within Open vSwitch on the Compute node running the Bare Metal Provisioning service:

   ```
   # ovs-vsctl add-br br-ironic
   # ovs-vsctl add-port br-ironic IRONIC_PROVISIONING_NIC
   ```

   Here, **br-ironic** is the name of the bridge and **IRONIC_PROVISIONING NIC** is the NIC connected to the Bare Metal Provisioning Network, for example, **eth6**.

   NOTE

   The only difference between this and Section 1.3, “Configure the Controller Nodes for Bare Metal Provisioning Service” is that you do not restart the OpenStack Networking service on the Compute node.

   This adds the bridge and port to the Open vSwitch, which you can verify using the **ovs-vsctl show** command. However, it does not connect it to the integration bridge (**br-int**) for use by OpenStack.

2. To create the connection, you need to update the OpenStack Networking plugin files as follows:

   a. Update the ML2 configuration file, **/etc/neutron/plugins/ml2/ml2_conf.ini** as follows:

   - For the **type_drivers** parameter, make sure **flat** is listed among the drivers, for example, **type_drivers = vxlan,vlan,flat,gre**. This is a comma delimited list.
   - For the **mechanism_drivers** parameter, make sure **openvswitch** option is listed among the drivers, for example, **mechanism_drivers =openvswitch**. This is a comma delimited list.
   - For the **flat_networks** parameters, create a name to refer to the Bare Metal Provisioning Network, for example, **ironicnet**. Make sure this name is listed among the **flat_networks** listed, for example, **flat_networks**

=datacentre,ironicnet. This is a comma delimited list.

- If you are using a VLAN for the Bare Metal Provisioning Network, add the **network_vlan_ranges** parameter with the following format: **ironicnet:VLAN_START:VLAN_END**, for example, **network_vlan_ranges =datacentre:1:1000**. This is a comma delimited list.

- The **enable_security_group** parameter should already be enabled. But if it is not set, change the value to **True**, for example, **enable_security_group = True**.

b. In the **/etc/neutron/plugins/ml2/openvswitch_agent.ini** file, find the **bridge_mappings** parameter and update as follows:

```
bridge_mappings =datacentre:br-ex,ironicnet:br-ironic
```

The value of this comma delimited key-value pair maps the name of the Bare Metal Provisioning Network to the physical device which is connected to the network.

3. Restart the OpenStack Networking Open vSwitch Agent service:

```
# systemctl restart neutron-openvswitch-agent.service
```

You have now achieved your first goal from this procedure. Next, you need to assign an IP address to your **br-ironic** bridge and make sure it persists after a reboot.

Assigning an IP address to the Bare Metal server

1. Create standard configuration files in the **/etc/sysconfig/network-scripts** location. You can copy the **ifcfg*** files already available in the tenant network and edit the following values: **device**, **ipaddr**, **ovs_bridge**, **bridge name** and **MAC** addresses for the **br-ironic** and **eth6**. When you have completed updating the new files, they should have the following values:
ifcfg-br-ironic

```
DEVICE=br-ironic
ONBOOT=yes
HOTPLUG=no
NM_CONTROLLED=no
PEERDNS=no
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=BARE_METAL_PROVISIONING_IP
NETMASK=255.255.255.0
OVS_EXTRA="set bridge br-ironic other-config:hwaddr=MAC_ADDRESS"
```

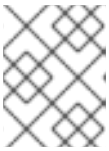
ifcfg-eth6

```
DEVICE=eth6
ONBOOT=yes
HOTPLUG=no
NM_CONTROLLED=no
PEERDNS=no
DEVICETYPE=ovs
```

```
TYPE=OVSPort
OVS_BRIDGE=br-ironic
BOOTPROTO=none
```

2. Restart the network bridge to make your IP address pingable.

```
# ifup br-ironic
```



NOTE

If you get disconnected from the node when you restart the network services, reboot the server.

1.4.1. Subscribe to the Required Channels

To install the Bare Metal Provisioning packages, you must register the server or servers with Red Hat Subscription Manager, and subscribe to the required channels. If you are installing Bare Metal Provisioning on a compute node, your server may already be appropriately subscribed. Run **yum repolist** to check whether the channels in the procedure below have been enabled.

Subscribing to the Required Channels

1. Register your system with the Content Delivery Network, entering your Customer Portal user name and password when prompted:

```
# subscription-manager register
```

2. Find entitlement pools containing the channels required to install Bare Metal Provisioning:

```
# subscription-manager list --available | grep -A13 "Red Hat
Enterprise Linux Server"
# subscription-manager list --available | grep -A13 "Red Hat
OpenStack Platform"
```

3. Use the pool identifiers located in the previous step to attach the **Red Hat Enterprise Linux 7 Server** and **Red Hat OpenStack Platform** entitlements:

```
# subscription-manager attach --pool=POOL_ID
```

4. Enable the required channels:

```
# subscription-manager repos --enable=rhel-7-server-rpms \
--enable=rhel-7-server-openstack-9-rpms \
--enable=rhel-7-server-rh-common-rpms --enable=rhel-7-server-
optional-rpms \
--enable=rhel-7-server-openstack-9-optools-rpms
```

1.4.2. Install the Bare Metal Provisioning Packages

Bare Metal Provisioning requires the following packages:

openstack-ironic-api

Provides the Bare Metal Provisioning API service.

openstack-ironic-conductor

Provides the Bare Metal Provisioning conductor service. The conductor allows adding, editing, and deleting nodes, powering on or off nodes with IPMI or SSH, and provisioning, deploying, and decommissioning bare metal nodes.

python-ironicclient

Provides a command-line interface for interacting with the Bare Metal Provisioning services.

Install the packages:

```
# yum install openstack-ironic-api openstack-ironic-conductor python-
ironicclient ipxe-bootimgs
```

1.4.3. Configure iPXE

1. Create the necessary directories for iPXE, map-files and copy the **undionly.kpxe** boot image, iPXE and **map-file** into place:

```
# mkdir /httpboot
# mkdir /tftpboot
# echo 'r ([/]) /tftpboot/\1' > /tftpboot/map-file
# echo 'r ^(/tftpboot/) /tftpboot/\2' >> /tftpboot/map-file
# cp /usr/share/ipxe/undionly.kpxe /tftpboot/
# chown -R ironic:ironic /httpboot
# chown -R ironic:ironic /tftpboot
```

2. By default, the Compute node deployed by the director runs SELinux in **Enforcing** mode. To avoid getting permission errors when trying iPXE boot, make sure you set the appropriate labels on these directories. To apply these labels, run the following commands as a **root** user:

```
# semanage fcontext -a -t httpd_sys_content_t "/httpboot(/.*)?"
# restorecon -Rv /httpboot
# semanage fcontext -a -t tftpdir_t "/tftpboot(/.*)?"
# restorecon -Rv /tftpboot
```

3. Configure HTTP so that it can serve requests for the images. The **httpd** package is already installed, so it is a matter of creating the appropriate virtual host entry and starting the service.

NOTE

The **/etc/httpd/conf.d** contains number of files. As Red Hat utilizes a single overcloud full image for all the nodes, it includes these files on all the nodes even though it is only used on the Controller node. You can delete the contents of **/etc/httpd/conf.d** or copy them somewhere else as they are not used.

Create a new file in for iPXE configuration. You can name this file anything, making sure it is in the **.conf** format and has the following contents:

```
# cat 10-ipxe_vhost.conf
```

```
Listen 8088
<VirtualHost *:8088>
    DocumentRoot "/httpboot"
    <Directory "/httpboot">
        Options Indexes FollowSymLinks
        AllowOverride None
        Order allow,deny
        Allow from all
        Require all granted
    </Directory>

    ## Logging
    ErrorLog "/var/log/httpd/ironic_error.log"
    ServerSignature Off
    CustomLog "/var/log/httpd/ironic_access.log" combined
</VirtualHost>
```

The above virtual host configuration configures HTTPD to listed on all addresses on port 8088 and sets the document root for all requests to that port to go to **`/httpboot`**.

4. Save this file and enable and restart HTTPD service on the Compute node:

```
# systemctl enable httpd.service
# systemctl start httpd.service
```

1.4.4. Configure the Bare Metal Provisioning Service

In this section, you will make the necessary changes to the **`/etc/ironic/ironic.conf`** file.

1.4.4.1. Configure Bare Metal Provisioning to Communicate with the Database Server

Set the value of the connection configuration key:

```
# openstack-config --set /etc/ironic/ironic.conf \
database connection mysql+pymysql://ironic:PASSWORD@IP/ironic
```

Here, *PASSWORD* is the password of the database server, *IP* is the IP address or host name of the database server.

IMPORTANT

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Bare Metal Provisioning database user was granted access when creating the Bare Metal Provisioning database in Section 1.3.1, “Create the Bare Metal Provisioning Database”. Moreover, if the database is hosted locally and you granted permissions to **`localhost`** when creating the database, you must enter **`localhost`**.

1.4.4.2. Configure Bare Metal Provisioning Authentication

Configure Bare Metal Provisioning to use Identity for authentication. All steps in this procedure must be performed on the server or servers hosting Bare Metal Provisioning, while logged in as the **root** user.

Configuring Bare Metal Provisioning to Authenticate Through Identity

1. Set the Identity public and admin endpoints that Bare Metal Provisioning must use:

```
# openstack-config --set /etc/ironic/ironic.conf \
   keystone_authtoken auth_uri http://IP:5000/v2.0
# openstack-config --set /etc/ironic/ironic.conf \
   keystone_authtoken identity_uri http://IP:35357/
```

   Replace *IP* with the IP address or host name of the Identity server.

2. Set Bare Metal Provisioning to authenticate as the **service** tenant:

```
# openstack-config --set /etc/ironic/ironic.conf \
   keystone_authtoken admin_tenant_name service
```

3. Set Bare Metal Provisioning to authenticate using the **ironic** administrative user account:

```
# openstack-config --set /etc/ironic/ironic.conf \
   keystone_authtoken admin_user ironic
```

4. Set Bare Metal Provisioning to use the correct **ironic** administrative user account password:

```
# openstack-config --set /etc/ironic/ironic.conf \
   keystone_authtoken admin_password PASSWORD
```

   Replace *PASSWORD* with the password set when the **ironic** user was created.

1.4.4.3. Configure RabbitMQ Message Broker Settings for Bare Metal Provisioning

RabbitMQ is the default (and recommended) message broker. The RabbitMQ messaging service is provided by the *rabbitmq-server* package. All steps in the following procedure must be performed on the Controller or Compute nodes hosting Bare Metal Provisioning, while logged in as the **root** user.

This procedure assumes that the RabbitMQ messaging service has been installed and configured, and an **ironic** user and associated password have been created on the server hosting the messaging service.

Configuring Bare Metal Provisioning to use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

```
# openstack-config --set /etc/ironic/ironic.conf \
   DEFAULT rpc_backend ironic.openstack.common.rpc.impl_kombu
```

2. Set the Bare Metal Provisioning to connect to the RabbitMQ host:

```
# openstack-config --set /etc/ironic/ironic.conf \
   oslo_messaging_rabbit rabbit_host RABBITMQ_HOST
```

   Replace *RABBITMQ_HOST* with the IP address or host name of the server hosting the message broker.

3. Set the message broker port to 5672:

```
# openstack-config --set /etc/ironic/ironic.conf \
   oslo_messaging_rabbit rabbit_port 5672
```

4. Set the RabbitMQ user name and password created for Bare Metal Provisioning when RabbitMQ was configured:

```
# openstack-config --set /etc/ironic/ironic.conf \
   oslo_messaging_rabbit rabbit_userid guest
# openstack-config --set /etc/ironic/ironic.conf \
   oslo_messaging_rabbit rabbit_password RABBIT_GUEST_PASSWORD
```

   Replace *RABBIT_GUEST_PASSWORD* with the RabbitMQ password for the guest user.

5. When RabbitMQ was launched, the **guest** user was granted read and write permissions to all resources: specifically, through the virtual host. Configure Bare Metal Provisioning to connect to this virtual host:

```
# openstack-config --set /etc/ironic/ironic.conf \
   oslo_messaging_rabbit rabbit_virtual_host /
```

1.4.4.4. Configure Bare Metal Provisioning Drivers

Bare Metal Provisioning supports multiple drivers for deploying and managing bare metal servers. Some drivers have hardware requirements, and require additional configuration or package installation. See Appendix A, *Bare Metal Provisioning Drivers* for detailed driver information. The first half of a driver's name specifies its deployment method (e.g. PXE), and the second half specifies its power management method (e.g. IPMI).

Configuring Bare Metal Provisioning Drivers

1. Specify the driver or drivers that you will use to provision bare metal servers. Specify multiple drivers using a comma-separated list:

```
# openstack-config --set /etc/ironic/ironic.conf \
   DEFAULT enabled_drivers DRIVER1,DRIVER2
```

   The following drivers are supported:

   - IPMI with PXE deploy
     - **pxe_ipmitool**
   - DRAC with PXE deploy
     - **pxe_drac**
   - iLO with PXE deploy
     - **pxe_ilo**
   - SSH with PXE deploy
     - **pxe_ssh**
   - iRMC with PXE

- **pxe_irmc**

- AMT with PXE
  - **pxe_amt**

- AMT with HTTP
  - **agent_amt**

2. Restart the Bare Metal conductor service:

```
# systemctl restart openstack-ironic-conductor.service
```

1.4.4.5. Configure the Bare Metal Provisioning Service to use PXE

1. Set the Bare Metal Provisioning service to use PXE templates:

```
# openstack-config --set /etc/ironic/ironic.conf \
pxe pxe_config_template
\$pybasedir/drivers/modules/ipxe_config.template
```

2. Set the Bare Metal Provisioning service to use **tftp_server**:

```
# openstack-config --set /etc/ironic/ironic.conf \
pxe tftp_server BARE_METAL_PROVISIONING_NETWORK_IP
```

3. Set the PXE **tftp_root**:

```
# openstack-config --set /etc/ironic/ironic.conf \
pxe tftp_root /tftpboot
```

4. Set the PXE boot file name:

```
# openstack-config --set /etc/ironic/ironic.conf \
pxe pxe_bootfile_name undionly.kpxe
```

5. Enable the Bare Metal Provisioning service to use iPXE:

```
# openstack-config --set /etc/ironic/ironic.conf \
pxe ipxe_enabled true
```

6. Set the URL for the **http** server:

```
# openstack-config --set /etc/ironic/ironic.conf deploy http_url
http://BARE_METAL_PROVISIONING_IP:8088
```

7. Restart the Bare Metal conductor service:

```
# systemctl restart openstack-ironic-conductor.service
```

1.4.4.6. Configure Bare Metal Provisioning to Communicate with OpenStack Networking and OpenStack Image

Bare Metal Provisioning uses OpenStack Networking for DHCP and network configuration, and uses the Image service for managing the images used to boot physical machines. Configure Bare Metal Provisioning to connect to and communicate with OpenStack Networking and the Image service. All steps in this procedure must be performed on the server hosting Bare Metal Provisioning, while logged in as the **root** user.

Configuring Bare Metal Provisioning to Communicate with OpenStack Networking and OpenStack Image

1. Set Bare Metal Provisioning to use the OpenStack Networking endpoint:

   ```
   # openstack-config --set /etc/ironic/ironic.conf \
      neutron url http://NEUTRON_IP:9696
   ```

   Replace *NEUTRON_IP* with the IP address or host name of the server hosting OpenStack Networking.

2. Set Bare Metal Provisioning to communicate with the Image service:

   ```
   # openstack-config --set /etc/ironic/ironic.conf \
      glance glance_host GLANCE_IP
   ```

   Replace *GLANCE_IP* with the IP address or host name of the server hosting the Image service.

3. Start the Bare Metal Provisioning API service, and configure it to start at boot time:

   ```
   # systemctl start openstack-ironic-api.service
   # systemctl enable openstack-ironic-api.service
   ```

4. Create the Bare Metal Provisioning database tables:

   ```
   # ironic-dbsync --config-file /etc/ironic/ironic.conf create_schema
   ```

5. Start the Bare Metal Provisioning conductor service, and configure it to start at boot time:

   ```
   # systemctl restart openstack-ironic-conductor.service
   # systemctl enable openstack-ironic-conductor.service
   ```

1.4.4.7. Configure a Ceph Object Gateway for the Image and Bare Metal Provisioning Services

Red Hat Ceph Storage is a distributed storage system that includes a Ceph object (RADOS) gateway with a Swift-compatible API. To use a RADOS gateway for the Image service, you need to:

- Configure the Image service to grant access to the RADOS gateway.
- Configure the Bare Metal Provisioning service to use the RADOS gateway Swift API to provide bare metal images.

Before You Begin

Ensure that you have configured your Red Hat Ceph Storage with a Ceph object gateway. See Ceph Object Gateway Installation for details.

Configure the Ceph Object Gateway Access for the Image Service

1. Create the access credentials for the Image Service on the Ceph object gateway **admin** host.

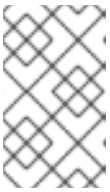
```
# radosgw-admin user create --uid=GLANCE_USERNAME --display-
name="User for Glance"

# radosgw-admin subuser create --uid=GLANCE_USERNAME --
subuser=GLANCE_USERNAME:swift --access
=full

# radosgw-admin key create --subuser=GLANCE_USERNAME:swift --key-
type=swift --secret=STORE_KEY

# radosgw-admin user modify --uid=GLANCE_USERNAME --temp-url-
key=TEMP_URL_KEY
```

   Replace *GLANCE_USERNAME* with a user name for the Image service access, and replace *STORE_KEY* and *TEMP_URL_KEY* with suitable keys.



NOTE

Do not use the **--gen-secret** CLI parameter because it will cause the **radosgw-admin** utility to generate keys with slash symbols which do not work with the OpenStack Image service.

2. Edit the **/etc/glance/glance-api.conf** file to configure the Image API service to use the Ceph object gateway Swift API as the backend.

```
[glance_store]

stores = file, http, swift
default_store = swift
swift_store_auth_version = 1
swift_store_auth_address = http://RADOS_IP:PORT/auth/1.0
swift_store_user = GLANCE_USERNAME:swift
swift_store_key = STORE_KEY
swift_store_container = glance
swift_store_create_container_on_put = True
```

   Replace *RADOS_IP* and *PORT* with the IP/port of the Ceph object gateway API service.

NOTE

The Ceph object gateway uses the FastCGI protocol for interacting with the HTTP server. See your HTTP server documentation if you want to enable HTTPS support.

3. Restart the Image API service.

```
# systemctl restart openstack-glance-api.service
```

Configure Bare Metal Provisioning to use a Ceph Object Gateway

1. Edit the **/etc/ironic/ironic.conf** file to configure the Bare Metal Provisioning conductor service to use the Ceph object gateway.

```
[glance]

swift_container = glance
swift_api_version = v1
swift_endpoint_url = http://RADOS_IP:PORT
swift_temp_url_key = TEMP_URL_KEY
temp_url_endpoint_type=radosgw
```

   Replace *TEMP_URL_KEY and _RADOS_IP:PORT* with the values used in the prior procedure.

2. Restart the Bare Metal Provisioning conductor service.

```
# systemctl restart openstack-ironic-conductor.service
```

1.4.5. Configure OpenStack Compute to Use Bare Metal Provisioning Service

In this section, you will update the **/etc/nova/nova.conf** file to configure the Compute service to use the Bare Metal Provisioning service:

Configuring OpenStack Compute to Use Bare Metal Provisioning

1. Set Compute to use the clustered compute manager:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT compute_manager
ironic.nova.compute.manager.ClusteredComputeManager
```

2. Set the virtual RAM to physical RAM allocation ratio:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT ram_allocation_ratio 1.0
```

3. Set the amount of disk space in MB to reserve for the host:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT reserved_host_memory_mb 0
```

4. Set Compute to use the Bare Metal Provisioning driver:

```
# openstack-config --set /etc/nova/nova.conf \
   DEFAULT compute_driver nova.virt.ironic.IronicDriver
```

5. Set Compute to use the correct authentication details for Bare Metal Provisioning:

```
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_username ironic
```

```
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_password PASSWORD
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_url http://IDENTITY_IP:35357/v2.0
# openstack-config --set /etc/nova/nova.conf \
   ironic admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf \
   ironic api_endpoint http://IRONIC_API_IP:6385/v1
```

Replace the following values:

- Replace *PASSWORD* with the password that Bare Metal Provisioning uses to authenticate with Identity.
- Replace *IDENTITY_IP* with the IP address or host name of the server hosting Identity.
- Replace *IRONIC_API_IP* with the IP address or host name of the server hosting the Bare Metal Provisioning API service.

6. Restart the Compute scheduler service on the Compute controller nodes:

```
# systemctl restart openstack-nova-scheduler.service
```

7. Restart the compute service on the compute nodes:

```
# systemctl restart openstack-nova-compute.service
```

CHAPTER 2. CONFIGURE BARE METAL DEPLOYMENT

Configure Bare Metal Provisioning, the Image service, and Compute to enable bare metal deployment in the OpenStack environment. The following sections outline the additional configuration steps required to successfully deploy a bare metal node.

2.1. CREATE OPENSTACK CONFIGURATIONS FOR BARE METAL PROVISIONING SERVICE

2.1.1. Configure the OpenStack Networking Configuration

Configure OpenStack Networking to communicate with Bare Metal Provisioning for DHCP, PXE boot, and other requirements. The procedure below configures OpenStack Networking for a single, flat network use case for provisioning onto bare metal. The configuration uses the ML2 plug-in and the Open vSwitch agent.

Ensure that the network interface used for provisioning is not the same network interface that is used for remote connectivity on the OpenStack Networking node. This procedure creates a bridge using the Bare Metal Provisioning Network interface, and drops any remote connections.

All steps in the following procedure must be performed on the server hosting OpenStack Networking, while logged in as the **root** user.

Configuring OpenStack Networking to Communicate with Bare Metal Provisioning

1. Set up the shell to access Identity as the administrative user:

```
# source ~stack/overcloudrc
```

2. Create the flat network over which to provision bare metal instances:

```
# neutron net-create --tenant-id TENANT_ID sharednet1 --shared \
--provider:network_type flat --provider:physical_network PHYSNET
```

Replace *TENANT_ID* with the unique identifier of the tenant on which to create the network. Replace *PHYSNET* with the name of the physical network.

3. Create the subnet on the flat network:

```
# neutron subnet-create sharednet1 NETWORK_CIDR --name SUBNET_NAME \
--ip-version 4 --gateway GATEWAY_IP --allocation-pool \
start=START_IP,end=END_IP --enable-dhcp
```

Replace the following values:

- Replace *NETWORK_CIDR* with the Classless Inter-Domain Routing (CIDR) representation of the block of IP addresses the subnet represents. The block of IP addresses specified by the range started by *START_IP* and ended by *END_IP* must fall within the block of IP addresses specified by *NETWORK_CIDR*.
- Replace *SUBNET_NAME* with a name for the subnet.
- Replace *GATEWAY_IP* with the IP address or host name of the system that will act as the gateway for the new subnet. This address must be within the block of IP addresses specified

by *NETWORK_CIDR*, but outside of the block of IP addresses specified by the range started by *START_IP* and ended by *END_IP*.

- Replace *START_IP* with the IP address that denotes the start of the range of IP addresses within the new subnet from which floating IP addresses will be allocated.
- Replace *END_IP* with the IP address that denotes the end of the range of IP addresses within the new subnet from which floating IP addresses will be allocated.

4. Attach the network and subnet to the router to ensure the metadata requests are served by the OpenStack Networking service.

```
# neutron router-create ROUTER_NAME
```

Replace **ROUTER_NAME** with a name for the router.

5. Add the Bare Metal subnet as an interface on this router:

```
# neutron router-interface-add ROUTER_NAME BAREMETAL_SUBNET
```

Replace *ROUTER_NAME* with the name of your router and **BAREMETAL_SUBNET** with the ID or subnet name that you previously created. This allows the metadata requests from **cloud-init** to be served and the node configured.

6. Update the **/etc/ironic/ironic.conf** file on the Compute node running the Bare Metal Provisioning service to utilize the same network for the cleaning service. Login to the Compute node where the Bare Metal Provisioning service is running and execute the following as a **root** user:

```
# openstack-config --set /etc/ironic/ironic.conf neutron
cleaning_network_uuid NETWORK_UUID
```

Replace the *NETWORK_UUID* with the ID of the Bare Metal Provisioning Network created in the previous steps.

7. Restart the Bare Metal Provisioning service:

```
# systemctl restart openstack-ironic-conductor.service
```

2.1.2. Create the Bare Metal Provisioning Flavor

You need to create a flavor to use as a part of the deployment which should have the specifications (memory, CPU and disk) that is equal to or less than what your bare metal node provides.

1. Set up the shell to access Identity as the administrative user:

```
# source ~stack/overcloudrc
```

2. List existing flavors:

```
# openstack flavor list
```

3. Create a new flavor for the Bare Metal Provisioning service:

```
# openstack flavor create --id auto --ram RAM --vcpus VCPU --disk
DISK --public baremetal
```

Replace **RAM** with the RAM memory, **VCPU** with the number of vCPUs and **DISK** with the disk storage value.

4. Verify that the new flavor is created with the respective values:

```
# openstack flavor list
```

2.1.3. Create the Bare Metal Images

Bare Metal Provisioning supports deploying whole-disk images or root partition images. The whole-disk image contains the partition table, kernel image, and final user image. Root partition images contains the root partition of the OS and requires the kernel and ramdisk image for the bare metal node to use to boot the final user image with. All supported bare metal agent drivers can deploy whole-disk or root partition images.

A whole-disk image requires one image that contains the partition table, boot loader, and user image. Bare Metal Provisioning does not control the subsequent reboot of a node deployed with a whole-disk image as the node supports localboot.

A root partition deployment requires two sets of images - **deploy** image and **user** image. Bare Metal Provisioning uses the **deploy** image to boot the node and copy the **user** image on to the bare metal node. After the **deploy** image is loaded into the Image service, you can use the bare metal node's properties to associate the **deploy** image to the bare metal node to set it to use the **deploy** image as the boot image. A subsequent reboot of the node uses net-boot to pull down the user image.

This section uses a root partition image to provision bare metal nodes in the examples. For a whole-disk image deployment, see Section 3.3, "Create a Whole Windows Image". For a partition-based deployment, you **do not** have to create the **deploy** image as it was already used when the overcloud was deployed by the undercloud. The **deploy** image consists of two images - the **kernel** image and the **ramdisk** image as follows:

```
ironic-python-agent.kernel
ironic-python-agent.initramfs
```

These images will be in the **/usr/share/rhosp-director-images/ironic-python-agent*.el7ost.tar** file if you have the **rhosp-director-images-ipa** package installed.

Extract the images and load them to the Image service:

```
# openstack image create --container-format aki --disk-format aki --public
--file ./ironic-python-agent.kernel bm-deploy-kernel
# openstack image create --container-format ari --disk-format ari --public
--file ./ironic-python-agent.initramfs bm-deploy-ramdisk
```

The final image that you need is the actual image that will be deployed on the Bare Metal Provisioning node. For example, you can download a **Red Hat Enterprise Linux KVM** image since it already has **cloud-init**.

Load the image to the Image service:

```
# openstack image create --container-format bare --disk-format qcow2  --
property kernel_id=DEPLOY_KERNEL_ID \
--property ramdisk_id=DEPLOY_RAMDISK_ID --public --file ./IMAGE_FILE rhel
```

Where *DEPLOY_KERNEL_ID* is the UUID associated with the deploy-kernel images uploaded to the Image service. And *DEPLOY_RAMDISK_ID* is the UUID associated with the deploy-ramdisk image uploaded to the Image service. Use **`openstack image list`** to find these UUIDs.

2.1.4. Add the Bare Metal Provisioning Node to the Bare Metal Provisioning Service

In order to add the Bare Metal Provisioning node to the Bare Metal Provisioning service, copy the section of the **`instackenv.json`** file that was used to instantiate the cloud and modify it according to your needs.

1. Source the **`overcloudrc`** file and import the **`.json`** file:

```
# source ~stack/overcloudrc
# openstack baremetal import --json ./baremetal.json
```

2. Update the bare metal node in the Bare Metal Provisioning service to use the deployed images as the initial boot image by specifying the **`deploy_kernel`** and **`deploy_ramdisk`** in the **`driver_info`** section of the node:

```
# ironic node-update NODE_UUID add
driver_info/deploy_kernel=DEPLOY_KERNEL_ID
driver_info/deploy_ramdisk=DEPLOY_RAMDISK_ID
```

Replace **`NODE_UUID`** with the UUID of the bare metal node. You can get this value by executing the **`ironic node-list`** command on the director node. Replace **`DEPLOY_KERNEL_ID`** with the ID of the deploy kernel image. You can get this value by executing the **`glance image-list`** command on the director node. Replace the **`DEPLOY_RAMDISK_ID`** with the ID of the deploy ramdisk image. You can get this value by executing the **`glance image-list`** command on the director node.

2.1.5. Configure Proxy Services For Image Deployment

You can optionally configure a bare metal node to use Object Storage with HTTP or HTTPS proxy services to download images to the bare metal node. This allows you to cache images in the same physical network segments as the bare metal nodes to reduce overall network traffic and deployment time.

Before you Begin

Configure the proxy server with the following additional considerations:

- Use content caching, even for queries contained in the requested URL.
- Raise the maximum cache size to accommodate your image sizes.

NOTE

Only configure the proxy server to store images as unencrypted if the images do not contain sensitive information.

Configure Image Proxy

1. Set up the shell to access Identity as the administrative user:

```
# source ~stack/overcloudrc
```

2. Configure the bare metal node driver to use HTTP or HTTPS:

```
# openstack baremetal node set NODE_UUID \
   --driver_info image_https_proxy=HTTPS://PROXYIP:PORT
```

This example uses the HTTPS protocol. Set the **driverinfo/image_http_proxy** parameter if you want to use HTTP instead of HTTPS.

3. Set Bare Metal Provisioning to reuse cached Object Storage temporary URLs when an image is requested.

```
# openstack-config --set /etc/ironic/ironic.conf glance
swift_temp_url_cache_enabled=true
```

The proxy server will not create new cache entries for the same image based on the query part of the URL when it contains some query parameters that change each time the request is regenerated.

4. Set the duration (in seconds) that the generated temporary URL remains valid:

```
# openstack-config --set /etc/ironic/ironic.conf glance
swift_temp_url_duration=DURATION
```

Only non-expired links to images will be returned from the Object Storage service temporary URLs cache. If swift_temp_url_duration=1200, then after 20 minutes a new image will be cached by the proxy server. The value of this option must be greater than or equal to swift_temp_url_expected_download_start_delay.

5. Set the download start delay (in seconds) for your hardware:

```
# openstack-config --set /etc/ironic/ironic.conf glance
swift_temp_url_expected_download_start_delay=DELAY
```

Set *DELAY* to cover the delay between when the deploy request is made (the temporary URL is generated) to when the URL is used to download an image to the bare metal node. This delay allows time for the ramdisk to boot and begin the image download. This value determines if a cached entry will still be valid when the image download starts.

2.1.6. Deploy the Bare Metal Provisioning Node

Deploy the Bare Metal Provisioning node using the **nova boot** command:

```
# nova  boot --image BAREMETAL_USER_IMAGE --flavor BAREMETAL_FLAVOR --nic
net-id=IRONIC_NETWORK_ID --key default MACHINE_HOSTNAME
```

Replace **BAREMETAL_USER_IMAGE** with image that was loaded to the Image service, **BAREMETAL_FLAVOR** with the flavor for the Bare Metal deployment, **IRONIC_NETWORK_ID** with the ID of the Bare Metal Provisioning Network in the OpenStack Networking service, and **MACHINE_HOSTNAME**

with the hostname of the machine you want it to be after it is deployed.

2.2. CONFIGURE HARDWARE INSPECTION

Hardware inspection allows Bare Metal Provisioning to discover hardware information on a node. Inspection also creates ports for the discovered Ethernet MAC addresses. Alternatively, you can manually add hardware details to each node; see Section 2.3.2, “Add a Node Manually” for more information. All steps in the following procedure must be performed on the server hosting the Bare Metal Provisioning conductor service, while logged in as the **root** user.

Hardware inspection is supported in-band using the following drivers:

- `pxe_drac`
- `pxe_ipmitool`
- `pxe_ssh`
- `pxe_amt`

Configuring Hardware Inspection

1. Obtain the **Ironic Python Agent** kernel and ramdisk images used for bare metal system discovery over PXE boot. These images are available in a TAR archive labeled **Ironic Python Agent Image for RHOSP director 9.0** at https://access.redhat.com/downloads/content/191/ver=9/rhel---7/9/x86_64/product-software . Download the TAR archive, extract the image files (***ironic-python-agent.kernel*** and ***ironic-python-agent.initramfs***) from it, and copy them to the ***/tftpboot*** directory on the TFTP server.

2. On the server that will host the hardware inspection service, enable the **Red Hat OpenStack Platform 9 director for RHEL 7 (RPMs)** channel:

```
# subscription-manager repos --enable=rhel-7-server-openstack-9-
director-rpms
```

3. Install the **openstack-ironic-inspector** package:

```
# yum install openstack-ironic-inspector
```

4. Enable inspection in the ***ironic.conf*** file:

```
# openstack-config --set /etc/ironic/ironic.conf \
   inspector enabled True
```

5. If the hardware inspection service is hosted on a separate server, set its URL on the server hosting the conductor service:

```
# openstack-config --set /etc/ironic/ironic.conf \
   inspector service_url http://INSPECTOR_IP:5050
```

   Replace *INSPECTOR_IP* with the IP address or host name of the server hosting the hardware inspection service.

6. Provide the hardware inspection service with authentication credentials:

```
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   keystone_authtoken identity_uri http://IDENTITY_IP:35357
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   keystone_authtoken auth_uri http://IDENTITY_IP:5000/v2.0
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   keystone_authtoken admin_user ironic
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   keystone_authtoken admin_password PASSWORD
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   keystone_authtoken admin_tenant_name services
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   ironic os_auth_url http://IDENTITY_IP:5000/v2.0
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   ironic os_username ironic
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   ironic os_password PASSWORD
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   ironic os_tenant_name service
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   firewall dnsmasq_interface br-ironic
# openstack-config --set /etc/ironic-inspector/inspector.conf \
   database connection sqlite:////var/lib/ironic-
inspector/inspector.sqlite
```

Replace the following values:

- Replace *IDENTITY_IP* with the IP address or host name of the Identity server.
- Replace *PASSWORD* with the password that Bare Metal Provisioning uses to authenticate with Identity.

7. Optionally, set the hardware inspection service to store logs for the ramdisk:

```
# openstack-config --set /etc/ironic-inspector/inspector.conf \
processing ramdisk_logs_dir /var/log/ironic-inspector/ramdisk
```

8. Optionally, enable an additional data processing plug-in that gathers block devices on bare metal machines with multiple local disks and exposes root devices. **`ramdisk_error`**, **`root_disk_selection`**, **`scheduler`**, and **`validate_interfaces`** are enabled by default, and should not be disabled. The following command adds **`root_device_hint`** to the list:

```
# openstack-config --set /etc/ironic-inspector/inspector.conf \
processing processing_hooks
'$default_processing_hooks,root_device_hint'
```

9. Generate the initial **`ironic inspector`** database:

```
# ironic-inspector-dbsync --config-file /etc/ironic-
inspector/inspector.conf upgrade
```

10. Update the inspector database file to be owned by ironic-inspector:

```
# chown ironic-inspector /var/lib/ironic-inspector/inspector.sqlite
```

11. Open the ***/etc/ironic-inspector/dnsmasq.conf*** file in a text editor, and configure the following PXE boot settings for the **openstack-ironic-inspector-dnsmasq** service:

```
port=0
interface=br-ironic
bind-interfaces
dhcp-range=START_IP,END_IP
enable-tftp
tftp-root=/tftpboot
dhcp-boot=pxelinux.0
```

Replace the following values:

- Replace *INTERFACE* with the name of the Bare Metal Provisioning Network interface.
- Replace *START_IP* with the IP address that denotes the start of the range of IP addresses from which floating IP addresses will be allocated.
- Replace *END_IP* with the IP address that denotes the end of the range of IP addresses from which floating IP addresses will be allocated.

12. Copy the **`syslinux bootloader`** to the **`tftp`** directory:

```
# cp /usr/share/syslinux/pxelinux.0 /tftpboot/pxelinux.0
```

13. Optionally, you can configure the hardware inspection service to store metadata in the *swift* section of the **`/etc/ironic-inspector/inspector.conf`** file.

```
[swift]
username = ironic
password = PASSWORD
tenant_name = service
os_auth_url = http://IDENTITY_IP:5000/v2.0
```

Replace the following values:

- Replace *IDENTITY_IP* with the IP address or host name of the Identity server.
- Replace *PASSWORD* with the password that Bare Metal Provisioning uses to authenticate with Identity.

14. Open the ***/tftpboot/pxelinux.cfg/default*** file in a text editor, and configure the following options:

```
default discover

label discover
kernel ironic-python-agent.kernel
append initrd=ironic-python-agent.initramfs \
ipa-inspection-callback-url=http://INSPECTOR_IP:5050/v1/continue
ipa-api-url=http://IRONIC_API_IP:6385

ipappend 3
```

Replace *INSPECTOR_IP* with the IP address or host name of the server hosting the hardware inspection service. Note that the text from **append** to **/continue** must be on a single line, as indicated by the \ in the block above.

15. Reset the security context for the ***/tftpboot/*** directory and its files:

```
# restorecon -R /tftpboot/
```

This step ensures that the directory has the correct SELinux security labels, and the dnsmasq service is able to access the directory.

16. Start the hardware inspection service and the dnsmasq service, and configure them to start at boot time:

```
# systemctl start openstack-ironic-inspector.service
# systemctl enable openstack-ironic-inspector.service
# systemctl start openstack-ironic-inspector-dnsmasq.service
# systemctl enable openstack-ironic-inspector-dnsmasq.service
```

Hardware inspection can be used on nodes after they have been registered with Bare Metal Provisioning.

2.3. ADD PHYSICAL MACHINES AS BARE METAL NODES

Add as nodes the physical machines onto which you will provision instances, and confirm that Compute can see the available hardware. Compute is not immediately notified of new resources, because Compute's resource tracker synchronizes periodically. Changes will be visible after the next periodic task is run. This value, **scheduler_driver_task_period**, can be updated in ***/etc/nova/nova.conf***. The default period is 60 seconds.

After systems are registered as bare metal nodes, hardware details can be discovered using hardware inspection, or added manually.

2.3.1. Add a Node with Hardware Inspection

Register a physical machine as a bare metal node, then use **openstack-ironic-inspector** to detect the node's hardware details and create ports for each of its Ethernet MAC addresses. All steps in the following procedure must be performed on the server hosting the Bare Metal Provisioning conductor service, while logged in as the **root** user.

Adding a Node with Hardware Inspection

1. Set up the shell to use Identity as the administrative user:

```
# source ~/keystonerc_admin
```

2. Add a new node:

```
# ironic node-create -d DRIVER_NAME
```

Replace *DRIVER_NAME* with the name of the driver that Bare Metal Provisioning will use to provision this node. You must have enabled this driver in the ***/etc/ironic/ironic.conf*** file. To create a node, you must, at a minimum, specify the driver name.

IMPORTANT

Note the unique identifier for the node.

3. You can refer to a node by a logical name or by its UUID. Optionally assign a logical name to the node:

```
# ironic node-update NODE_UUID add name=NAME
```

Replace *NODE_UUID* with the unique identifier for the node. Replace *NAME* with a logical name for the node.

4. Determine the node information that is required by the driver, then update the node driver information to allow Bare Metal Provisioning to manage the node:

```
# ironic driver-properties DRIVER_NAME
# ironic node-update NODE_UUID add \
   driver_info/PROPERTY=VALUE \
   driver_info/PROPERTY=VALUE
```

Replace the following values:

- Replace *DRIVER_NAME* with the name of the driver for which to show properties. The information is not returned unless the driver has been enabled in the ***/etc/ironic/ironic.conf*** file.
- Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.
- Replace *PROPERTY* with a required property returned by the **ironic driver-properties** command.
- Replace *VALUE* with a valid value for that property.

5. Specify the deploy kernel and deploy ramdisk for the node driver:

```
# ironic node-update NODE_UUID add \
  driver_info/deploy_kernel=KERNEL_UUID \
  driver_info/deploy_ramdisk=INITRAMFS_UUID
```

Replace the following values:

- Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.
- Replace *KERNEL_UUID* with the unique identifier for the ***.kernel*** image that was uploaded to the Image service.
- Replace *INITRAMFS_UUID* with the unique identifier for the ***.initramfs*** image that was uploaded to the Image service.

6. Configure the node to reboot after initial deployment from a local boot loader installed on the node's disk, instead of via PXE or virtual media. The local boot capability must also be set on the flavor used to provision the node. To enable local boot, the image used to deploy the node must contain **grub2**. Configure local boot:

```
# ironic node-update NODE_UUID add \
   properties/capabilities="boot_option:local"
```

Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.

7. Move the bare metal node to **manageable** state:

```
# ironic node-set-provision-state NODE_UUID manage
```

Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.

8. Start inspection:

```
# openstack baremetal introspection start NODE_UUID --discoverd-url
http://overcloud IP:5050
```

- Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name. The node discovery and inspection process must run to completion before the node can be provisioned. To check the status of node inspection, run **ironic node-list** and look for **Provision State**. Nodes will be in **available** state after successful inspection.
- Replace *overcloud IP* with the **service_url** value that was previously set in *ironic.conf*.

9. Validate the node's setup:

```
# ironic node-validate NODE_UUID
+------------+--------+----------------------------+
| Interface  | Result | Reason                     |
+------------+--------+----------------------------+
| console    | None   | not supported              |
| deploy     | True   |                            |
| inspect    | True   |                            |
| management | True   |                            |
| power      | True   |                            |
+------------+--------+----------------------------+
```

Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name. The output of the command above should report either **True** or **None** for each interface. Interfaces marked **None** are those that you have not configured, or those that are not supported for your driver.

2.3.2. Add a Node Manually

Register a physical machine as a bare metal node, then manually add its hardware details and create ports for each of its Ethernet MAC addresses. All steps in the following procedure must be performed on the server hosting the Bare Metal Provisioning conductor service, while logged in as the **root** user.

Adding a Node without Hardware Inspection

1. Set up the shell to use Identity as the administrative user:

```
# source ~/keystonerc_admin
```

2. Add a new node:

```
# ironic node-create -d DRIVER_NAME
```

Replace *DRIVER_NAME* with the name of the driver that Bare Metal Provisioning will use to provision this node. You must have enabled this driver in the ***/etc/ironic/ironic.conf*** file. To create a node, you must, at a minimum, specify the driver name.

IMPORTANT

Note the unique identifier for the node.

3. You can refer to a node by a logical name or by its UUID. Optionally assign a logical name to the node:

```
# ironic node-update NODE_UUID add name=NAME
```

Replace *NODE_UUID* with the unique identifier for the node. Replace *NAME* with a logical name for the node.

4. Determine the node information that is required by the driver, then update the node driver information to allow Bare Metal Provisioning to manage the node:

```
# ironic driver-properties DRIVER_NAME
# ironic node-update NODE_UUID add \
   driver_info/PROPERTY=VALUE \
   driver_info/PROPERTY=VALUE
```

Replace the following values:

- Replace *DRIVER_NAME* with the name of the driver for which to show properties. The information is not returned unless the driver has been enabled in the ***/etc/ironic/ironic.conf*** file.
- Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.
- Replace *PROPERTY* with a required property returned by the **ironic driver-properties** command.
- Replace *VALUE* with a valid value for that property.

5. Specify the deploy kernel and deploy ramdisk for the node driver:

```
# ironic node-update NODE_UUID add \
  driver_info/deploy_kernel=KERNEL_UUID \
  driver_info/deploy_ramdisk=INITRAMFS_UUID
```

Replace the following values:

- Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.
- Replace *KERNEL_UUID* with the unique identifier for the ***.kernel*** image that was uploaded to the Image service.

- Replace *INITRAMFS_UUID* with the unique identifier for the ***.initramfs*** image that was uploaded to the Image service.

6. Update the node's properties to match the hardware specifications on the node:

```
# ironic node-update NODE_UUID add \
   properties/cpus=CPU \
   properties/memory_mb=RAM_MB \
   properties/local_gb=DISK_GB \
   properties/cpu_arch=ARCH
```

   Replace the following values:

   - Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.
   - Replace *CPU* with the number of CPUs to use.
   - Replace *RAM_MB* with the RAM (in MB) to use.
   - Replace *DISK_GB* with the disk size (in GB) to use.
   - Replace *ARCH* with the architecture type to use.

7. Configure the node to reboot after initial deployment from a local boot loader installed on the node's disk, instead of via PXE or virtual media. The local boot capability must also be set on the flavor used to provision the node. To enable local boot, the image used to deploy the node must contain **grub2**. Configure local boot:

```
# ironic node-update NODE_UUID add \
   properties/capabilities="boot_option:local"
```

   Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name.

8. Inform Bare Metal Provisioning of the network interface cards on the node. Create a port with each NIC's MAC address:

```
# ironic port-create -n NODE_UUID -a MAC_ADDRESS
```

   Replace *NODE_UUID* with the unique identifier for the node. Replace *MAC_ADDRESS* with the MAC address for a NIC on the node.

9. Validate the node's setup:

```
# ironic node-validate NODE_UUID
+------------+--------+----------------------------+
| Interface  | Result | Reason                     |
+------------+--------+----------------------------+
| console    | None   | not supported              |
| deploy     | True   |                            |
| inspect    | None   | not supported              |
| management | True   |                            |
| power      | True   |                            |
+------------+--------+----------------------------+
```

Replace *NODE_UUID* with the unique identifier for the node. Alternatively, use the node's logical name. The output of the command above should report either **`True`** or **`None`** for each interface. Interfaces marked **`None`** are those that you have not configured, or those that are not supported for your driver.

2.3.3. Configure Manual Node Cleaning

When a bare metal server is initially provisioned or reprovisioned after the server is freed from a workload, Bare Metal Provisioning can automatically clean the server to ensure that the server is ready for another workload. You can also initiate a manual cleaning cycle when the server is in the manageable state. Manual cleaning cycles are useful for long running or destructive tasks. You can configure the specific cleaning steps for the bare metal server.

Configure a Cleaning Network

Bare Metal Provisioning uses the cleaning network to provide in-band cleaning steps for the bare metal server. You can create a separate network for this cleaning network or use the provisioning network.

To configure the Bare Metal Provisioning service cleaning network, follow these steps:

1. Set up the shell to access Identity as the administrative user.

```
# source ~stack/overcloudrc
```

2. Find the network UUID for the network you want Bare Metal Provisioning to use for cleaning bare metal servers.

```
# openstack network list
```

   Select the UUID from the *id* field in the **`neutron net-list`** output.

3. Set **`cleaning_network_uuid`** in the **`/etc/ironic/ironic.conf`** file to the cleaning network UUID.

```
# openstack-config --set /etc/ironic/ironic.conf neutron
cleaning_network_uuid CLEANING_NETWORK_UUID
```

   Replace *CLEANING_NETWORK_UUID* with the network *id* retrieved in the earlier step.

4. Restart the Bare Metal Provisioning Service.

```
# systemctl restart openstack-ironic-conductor
```

Configure Manual Cleaning

1. Ensure that the bare metal server is in the manageable state.

```
# ironic node-set-provision-state NODE_ID manage
```

   Replace *NODE_ID* with the bare metal server UUID or node name.

2. Set the bare metal server in cleaning state and provide the cleaning steps.

```
# ironic node-set-provision-state NODE_ID clean --clean-steps
CLEAN_STEPS
```

Replace *NODE_ID* with the bare metal server UUID or node name. Replace *CLEAN_STEPS* with the cleaning steps in JSON format, a path to a file that contains the cleaning steps, or directly from standard input. The following is an example of cleaning steps in JSON format:

```
'[{"interface": "deploy", "step": "erase_devices"}]'
```

See OpenStack - Node Cleaning for more details.

2.3.4. Specify the Preferred Root Disk on a Bare Metal Node

When the deploy **`ramdisk`** boots on a bare metal node, the first disk that Bare Metal Provisioning discovers becomes the **`root`** device (the device where the image is saved). If the bare metal node has more than one SATA, SCSI, or IDE disk controller, the order in which their corresponding disk devices are added is arbitrary and may change at each reboot. For example, devices such as **`/dev/sda`** and **`/dev/sdb`** may switch on each boot, which would result in Bare Metal Provisioning selecting a different disk each time the bare metal node is being deployed.

With disk hints, you can pass hints to the deploy **`ramdisk`** to specify which disk device Bare Metal Provisioning should deploy the image onto. The following table describes the hints you can use to select a preferred **`root`** disk on a bare metal node.

Table 2.1. Disk Hints

Hint	Type	Description
model	(STRING):	Disk device identifier.
vendor	(STRING):	Disk device vendor.
serial	(STRING):	Disk device serial number.
size	(INT):	Size of the disk device in GB. **NOTE**: The **`local_gb`** property of a node is often set to a value 1 GB less than the actual disk size to account for partitioning. However, in this case **`size`** should be the actual size. For example, for a 128 GB disk **`local_gb`** is 127, but the **`size`** hint is 128.
wwn	(STRING):	Unique storage identifier.
wwn_with_extension	(STRING):	Unique storage identifier with the vendor extension appended.
wwn_vendor_extension	(STRING):	Unique vendor storage identifier.

Hint	Type	Description
name	(STRING):	The device name, for example /dev/md0. **WARNING**: The root device hint name should only be used for devices with constant names (for example, RAID volumes). Do not use this hint for SATA, SCSI, and IDE disk controllers because the order in which the device nodes are added in Linux is arbitrary, resulting in devices such as /dev/sda and /dev/sdb switching at boot time. See Persistent Naming for details.

To associate one or more disk device hints with a bare metal node, update the node's properties with a **`root_device`** key, for example:

```
# ironic node-update <node-uuid> add properties/root_device='{"wwn":
"0x4000cca77fc4dba1"}'
```

This example guarantees that Bare Metal Provisioning picks the disk device that has the **`wwn`** equal to the specified WWN value, or fails the deployment if no disk device on that node has the specified WWN value.

The hints can have an operator at the beginning of the value string. If no operator is specified the default is **==** (for numerical values) and **s==** (for string values).

Table 2.2. Supported Operators

Type	Operator	Description
numerical	**=**	equal to or greater than (equivalent to **>=**)
	==	equal to
	!=	not equal to
	>=	greater than or equal to
	>	greater than
	<=	less than or equal to
	<	less than
string (python comparisons)	**s==**	equal to
	s!=	not equal to

Type	Operator	Description
	s>=	greater than or equal to
	s>	greater than
	s<=	less than or equal to
	s<	less than
	<in>	substring
collections	**<all-in>**	all elements contained in collection
	<or>	find one of these

The following examples show how to update bare metal node properties to select a particular disk:

- Find a non-rotational (SSD) disk greater than or equal to 60 GB:

```
# ironic node-update <node-uuid> add properties/root_device='{"size": ">=
60", "rotational": false}'
```

- Find a Samsung or Winsys disk:

```
# ironic node-update <node-uuid> add properties/root_device='{"vendor": "
<or> samsung <or> winsys"}'
```

NOTE

If multiple hints are specified, a disk device must satisfy all the hints.

2.4. USE HOST AGGREGATES TO SEPARATE PHYSICAL AND VIRTUAL MACHINE PROVISIONING

Host aggregates are used by OpenStack Compute to partition availability zones, and group nodes with specific shared properties together. Key value pairs are set both on the host aggregate and on instance flavors to define these properties. When an instance is provisioned, Compute's scheduler compares the key value pairs on the flavor with the key value pairs assigned to host aggregates, and ensures that the instance is provisioned in the correct aggregate and on the correct host: either on a physical machine or as a virtual machine on an **openstack-nova-compute** node.

If your Red Hat OpenStack Platform environment is set up to provision both bare metal machines and virtual machines, use host aggregates to direct instances to spawn as either physical machines or virtual machines. The procedure below creates a host aggregate for bare metal hosts, and adds a key value pair specifying that the host type is **baremetal**. Any bare metal node grouped in this aggregate inherits this key value pair. The same key value pair is then added to the flavor that will be used to provision the instance.

If the image or images you will use to provision bare metal machines were uploaded to the Image service with the **hypervisor_type=ironic** property set, the scheduler will also use that key pair value in its scheduling decision. To ensure effective scheduling in situations where image properties may not apply, set up host aggregates in addition to setting image properties. See Section 2.1.3, "Create the Bare Metal Images" for more information on building and uploading images.

Creating a Host Aggregate for Bare Metal Provisioning

1. Create the host aggregate for **baremetal** in the default **nova** availability zone:

```
# nova aggregate-create baremetal nova
```

2. Set metadata on the **baremetal** aggregate that will assign hosts added to the aggregate the **hypervisor_type=ironic** property:

```
# nova aggregate-set-metadata baremetal hypervisor_type=ironic
```

3. Add the **openstack-nova-compute** node with Bare Metal Provisioning drivers to the **baremetal** aggregate:

```
# nova aggregate-add-host baremetal COMPUTE_HOSTNAME
```

Replace *COMPUTE_HOSTNAME* with the host name of the system hosting the **openstack-nova-compute** service. A single, dedicated compute host should be used to handle all Bare Metal Provisioning requests.

4. Add the **ironic** hypervisor property to the flavor or flavors that you have created for provisioning bare metal nodes:

```
# nova flavor-key FLAVOR_NAME set hypervisor_type="ironic"
```

Replace *FLAVOR_NAME* with the name of the flavor.

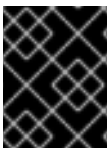
5. Add the following Compute filter scheduler to the existing list under **scheduler_default_filters** in ***/etc/nova/nova.conf***:

```
AggregateInstanceExtraSpecsFilter
```

This filter ensures that the Compute scheduler processes the key value pairs assigned to host aggregates.

2.5. EXAMPLE: TEST BARE METAL PROVISIONING WITH SSH AND VIRSH

Test the Bare Metal Provisioning setup by deploying instances on two virtual machines acting as bare metal nodes on a single physical host. Both virtual machines are virtualized using **libvirt** and **virsh**.



IMPORTANT

The SSH driver is for testing and evaluation purposes only. It is not recommended for Red Hat OpenStack Platform enterprise environments.

This scenario requires the following resources:

- A Red Hat OpenStack Platform environment with Bare Metal Provisioning services configured on an overcloud node. You must have completed all steps in this guide.
- One bare metal machine with Red Hat Enterprise Linux 7.2 and **libvirt** virtualization tools installed. This system acts as the host containing the virtualized bare metal nodes.
- One network connection between the Bare Metal Provisioning node and the host containing the virtualized bare metal nodes. This network acts as the Bare Metal Provisioning Network.

2.5.1. Create the Virtualized Bare Metal Nodes

Create two virtual machines that will act as the bare metal nodes in the test scenario. The nodes will be referred to as **`Node1`** and **`Node2`**.

Creating Virtualized Bare Metal Nodes

1. Access the Virtual Machine Manager from the **libvirt** host.
2. Create two virtual machines with the following configuration:
   - 1 vCPU
   - 2048 MB of memory
   - Network Boot (PXE)
   - 20 GB storage
   - **Network source**: **`Host device eth0: macvtap`** and **Source mode**: **`Bridge`**. Selecting macvtap sets the virtual machines to share the host's Ethernet network interface. This way the Bare Metal Provisioning node has direct access to the virtualized nodes.
3. Shut down both virtual machines.

2.5.2. Create an SSH Key Pair

Create an SSH key pair that will allow the Bare Metal Provisioning node to connect to the **libvirt** host.

Creating an SSH Key Pair

1. On the Bare Metal Provisioning node, create a new SSH key:

   ```
   # ssh-keygen -t rsa -b 2048 -C "user@domain.com" -f ./virtkey
   ```

   Replace *user@domain.com* with an email address or other comment that identifies this key. When the command prompts you for a passphrase, press **Enter** to proceed without a passphrase. The command creates two files: the private key (***virtkey***) and the public key (***virtkey.pub***).

2. Copy the contents of the public key into the ***/root/.ssh/authorized_keys*** file of the **libvirt** host's **root** user:

   ```
   # ssh-copy-id -i virtkey root@LIBVIRT_HOST
   ```

Replace *LIBVIRT_HOST* with the IP address or host name of the **libvirt** host.

The private key (***virtkey***) is used when the nodes are registered.

2.5.3. Add the Virtualized Nodes as Bare Metal Nodes

Add as nodes the virtual machines onto which you will provision instances. In this example, the driver details are provided manually and the node details are discovered using hardware inspection. Node details can also be added manually on a node-by-node basis. See Section 2.3.2, “Add a Node Manually” for more information.

Adding Virtualized Nodes as Bare Metal Nodes

1. On the Bare Metal Provisioning conductor service node, enable the **pxe_ssh** driver:

   ```
   # openstack-config --set /etc/ironic/ironic.conf \
      DEFAULT enabled_drivers pxe_ssh
   ```

   If you are adding **pxe_ssh** to a list of existing drivers, open the file and add the driver to the list in **enabled_drivers**, separated by a comma.

2. Set up the shell to use Identity as the administrative user:

   ```
   # source ~/keystonerc_admin
   ```

3. Add the first node, and register the SSH details for the **libvirt** host:

   ```
   # ironic node-create -d pxe_ssh -n Node1 \
      -i ssh_virt_type=virsh \
      -i ssh_username=root \
      -i ssh_key_filename=VIRTKEY_FILE_PATH \
      -i ssh_address=LIBVIRT_HOST_IP \
      -i deploy_kernel=KERNEL_UUID \
      -i deploy_ramdisk=INITRAMFS_UUID
   ```

   Replace the following values:

   - Replace *VIRTKEY_FILE_PATH* with the absolute file path of the **virtkey** SSH private key file.
   - Replace *LIBVIRT_HOST_IP* with the IP address or host name of the **libvirt** host.
   - Replace *KERNEL_UUID* with the unique identifier for the ***.kernel*** image that was uploaded to the Image service.
   - Replace *INITRAMFS_UUID* with the unique identifier for the ***.initramfs*** image that was uploaded to the Image service.

4. Add a second node, using the same command as above, and replacing **Node1** with **Node2**.

5. Configure the node to reboot after initial deployment from a local boot loader installed on the node’s disk, instead of via PXE or virtual media. The local boot capability must also have been set on the flavor you will use to provision the node. To enable local boot, the image used to deploy the node must contain **grub2**. Configure local boot:

```
# ironic node-update Node1 add \
   properties/capabilities="boot_option:local"
# ironic node-update Node2 add \
   properties/capabilities="boot_option:local"
```

6. Move the nodes to the manageable state:

```
# ironic node-set-provision-state Node1 manage
# ironic node-set-provision-state Node2 manage
```

7. Start inspection on the nodes:

```
# ironic node-set-provision-state Node1 inspect
# ironic node-set-provision-state Node2 inspect
```

   The node discovery and inspection process must run to completion before the node can be provisioned. To check the status of node inspection, run **ironic node-list** and look for **`Provision State`**. Nodes will be in the **`available`** state after successful inspection.

8. Validate the node's setup:

```
# ironic node-validate Node1
# ironic node-validate Node2
+------------+--------+----------------------------+
| Interface  | Result | Reason                     |
+------------+--------+----------------------------+
| console    | None   | not supported              |
| deploy     | True   |                            |
| inspect    | True   |                            |
| management | True   |                            |
| power      | True   |                            |
+------------+--------+----------------------------+
```

   The output of the command above should report either **`True`** or **`None`** for each interface. Interfaces marked **`None`** are those that you have not configured, or those that are not supported for your driver.

9. When the nodes have been successfully added, launch two instances using Chapter 3, *Launch Bare Metal Instances*.

CHAPTER 3. LAUNCH BARE METAL INSTANCES

Provision a physical machine on an enrolled bare metal node. Instances can be launched from the command line or in the OpenStack dashboard.

3.1. DEPLOY AN INSTANCE USING THE COMMAND LINE INTERFACE

Use the **nova** command line interface to deploy a bare metal instance.

Deploying an Instance on the Command Line

1. Set up the shell to access Identity as the administrative user:

```
# source ~/keystonerc_admin
```

2. Deploy the instance:

```
# nova boot --nic net-id=NETWORK_UUID --flavor FLAVOR_NAME --image
IMAGE_UUID INSTANCE_NAME
```

   Replace the following values:

   - Replace *NETWORK_UUID* with the unique identifier for the network that was created for use with Bare Metal Provisioning.
   - Replace *FLAVOR_NAME* with the name of the flavor that was created for the node.
   - Replace *IMAGE_UUID* with the unique identifier for the disk image that was uploaded to the Image service.
   - Replace *INSTANCE_NAME* with a name for the bare metal instance.

3. Check the status of the instance:

```
# nova list
```

3.2. DEPLOY AN INSTANCE USING THE DASHBOARD

Use the dashboard graphical user interface to deploy a bare metal instance.

Deploying an Instance in the Dashboard

1. Log in to the dashboard at **https://*DASHBOARD_IP*/dashboard**.
2. Click **Project > Compute > Instances**
3. Click **Launch Instance**.
4. In the **Details** tab, fill out the following fields:
   - Specify the **Instance Name**.
   - Select the **Flavor** that was created for the bare metal node.

- Select **1** from the **Instance Count** list.
- Select *Boot from image* from the **Instance Boot Source** list.
- Select the operating system disk image from the **Image Name** list.

5. In the **Networking** tab, drag and drop the required networks from **Available Networks** to **Selected Networks**. Ensure that the shared network created for Bare Metal Provisioning is selected here.

6. Click **Launch**.

3.3. CREATE A WHOLE WINDOWS IMAGE

This procedure creates a deployment image for Windows Server 2012. Perform the following steps on a Red Hat Enterprise Linux system:

1. Download the virtio-win drivers. Refer to the Red Hat customer portal for the required steps.
   As a result, the *virtio-win* .iso file is downloaded to */usr/share/virtio-win/*. For example: */usr/share/virtio-win/virtio-win-1.8.0.iso*

2. Create a Windows Server 2012 template VM using *virt-manager*:
   - Determine the virtual hardware requirements of your Windows template. This example uses 1 x CPU, 4GB RAM, 10GB HDD, 2 x NICs, and 2 x virtual CD-ROM drives.
   - Create the disk as a *qcow2* file.
   - Set both Virtual HDD and NIC drivers to **`virtio`**.
   - Attach 2x virtual CD-ROM drives, mounted to the Windows Server 2012 R2 *.iso* file, and the *virtio-win-1.8.0.iso* file. The two virtual CD-ROMs are required to allow the installation of the *virtio-win* drivers during the Windows installation process.

3. Install Windows manually from the ISO image:
   - Start the installation of Windows from the evaluation .iso image of Windows Server 2012 R2.
   - When given the opportunity to select a HDD driver, select the driver from the second virtual CD-ROM containing the virtio-win-1.8.0.iso file.

4. Perform post-install Windows checks:
   Open *Device Manager* and confirm that all devices are properly recognized, and that no *question mark* warnings are present. In particular, check that the NIC, serial, and balloon driver are using the VirtIO driver. If any devices are not correctly recognized, apply the driver from the virtio-win driver disc.

5. Run *sysprep*:
   Sysprep causes the Windows installation to become generic, removing installation information that was specific to the single installation performed previously. This allows you to use the virtual hard disk of the VM as a template for multiple installations to other systems.
   a. Launch **Sysprep** from *C:\Windows\System32\sysprep\sysprep.exe*
   b. Enter the following information into the Sysprep tool:
      - For **System Cleanup Action**, select **Enter System Out-of-Box-Experience (OOBE)**.

- Select the **Generalize** check box
- Under **Shutdown Options**, select **Shutdown**.

c. Click **OK** to complete the sysprep process. The virtual machine will shut down automatically upon completion.

6. Register the Windows image in Image Service (glance): This step registers the qcow2 HDD of the Windows installation in glance. This example uses the disk file: **`/root/win2012r2.qcow2`**.

```
$ glance image-create --file /root/win2012r2.qcow2 --disk-format
qcow2 --container-format bare --name win2012r2 --is-public True --
progress
[=============================>] 100%
```

3.3.1. Deploy Windows to a physical server

1. Register the physical node in ironic by specifying the hardware components. For example:

```
# ironic node-create -d fake_pxe -p cpus=1 -p memory_mb=1024 -p
local_gb=15 -p cpu_arch=amd64
```

2. Retrieve the new node's UUID:

```
# ironic node-list
```

3. Create a port for the node that matches the MAC address of the physical Windows node.

```
# ironic port-create -n [NODE_UUID] -a [NIC_MAC]
```

For example:

```
# ironic port-create -n 3b3d3fd4-4621-427f-a78d-054a1ef17a0a -a
52:54:00:85:76:53
```

4. Set the node to maintenance mode:

```
# ironic node-set-maintenance [NODE_UUID] true
# ironic node-set-provision-state [NODE_UUID] manage
```

5. Perform inspection of the new node:

```
# openstack baremetal introspection start [NODE_UUID] --discoverd-
url http://[KEYSTONE_URL]:5050
```

- Replace **`[KEYSTONE_URL]`** with the IP address of the keystone service.
  Note: If using the FAKE_PXE driver, power on the node manually once the *ironic node-list* state moves to **`deploy wait-callback`**.

6. Once inspection has completed, turn off maintenance mode:

```
# ironic node-set-maintenance [NODE UUID] false
# ironic node-set-provision-state [NODE UUID] provide
```

7. Create a keypair for authentication to the instance:

```
# nova keypair-add --pub_key ~/.ssh/id_rsa.pub [KEY_NAME]
```

8. Enable local boot for the node:

```
# ironic node-update [NODE_UUID] add
properties/capabilities="boot_option:local"
```

9. Use **nova** to boot the node:

```
# nova boot --nic net-id=[NETWORK_UUID] --flavor [FLAVOR_NAME] --
image [IMAGE_UUID/IMAGE_NAME] --keyname [INSTANCE_NAME]
```

   Replace the following values:

   - Replace **[NETWORK_UUID]** with the unique identifier for the network that was created for use with Bare Metal Provisioning.
   - Replace **[FLAVOR_NAME]** with the name of the flavor that was created for the node.
   - Replace **[IMAGE_UUID]** with the unique identifier for the disk image that was uploaded to the Image service.
   - Replace **[INSTANCE_NAME]** with a name for the bare metal instance.

10. Retrieve the instance password:

```
# nova get-password [INSTANCE_NAME]
/path/to/your/keypairs/private/key
```

11. Review the status of the instance:

```
# nova list
```

You can test this further by accessing the instance using the console in dashboard.

3.3.2. Enable Remote Desktop access

1. Add a security group rule allowing *Remote Desktop* traffic on **TCP/UDP 3389**.
2. Use the *nova* command to retrieve the security keys:

```
# nova get-password [INSTANCE_NAME]
/path/to/your/keypairs/private/key
```

   - Replace **[INSTANCE_NAME]** with the name of the bare metal instance.

CHAPTER 4. TROUBLESHOOT BARE METAL PROVISIONING

The following sections contain information and steps that may be useful for diagnosing issues in a Bare Metal Provisioning setup.

Bare Metal Provisioning with inspection uses four services: ***openstack-ironic-api***, ***openstack-ironic-conductor***, ***openstack-ironic-inspector***, and ***openstack-ironic-inspector-dnsmasq***. Logs for most OpenStack components can be found in the */**var**/**log*** directory.

4.1. TROUBLESHOOT PXE BOOT ERRORS

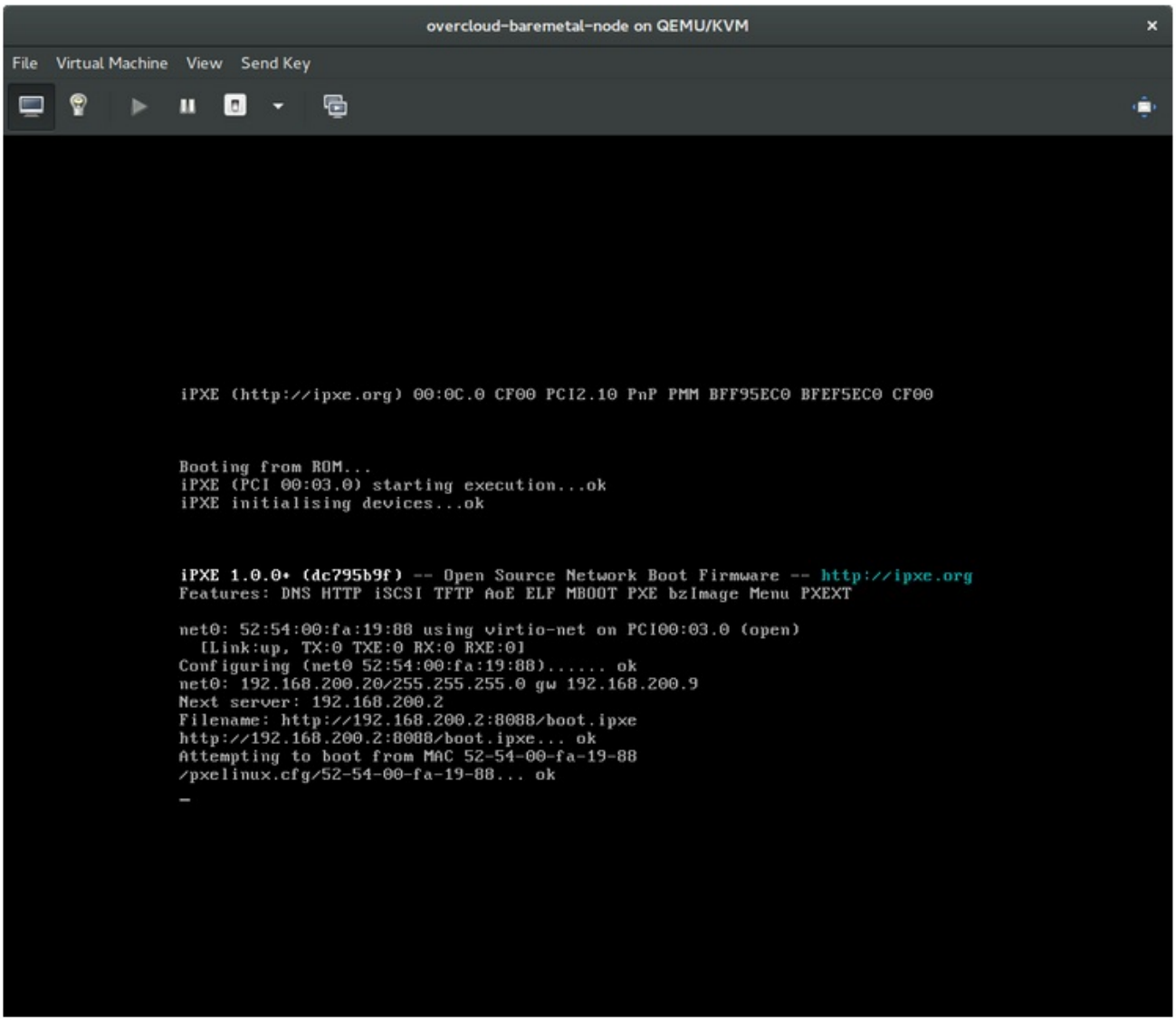
Permission Denied Errors

If you are getting a permission denied error on the console of your Bare Metal Provisioning node, make sure you applied the appropriate SELinux content to the **`/httpboot`** and **`/tftpboot`** directories as follows:

```
# semanage fcontext -a -t httpd_sys_content_t "/httpboot(/.*)?"
# semanage fcontext -a -t tftpdir_t "/tftpboot(/.*)?"
```

Boot Process Freezes at `/pxelinux.cfg/XX-XX-XX-XX-XX-XX`

On the console of your node, if it looks like you are getting an IP address and then the process stops as shown below:



This indicates that you might be using the wrong PXE boot template in your **`ironic.conf`** file.

```
# grep ^pxe_config_template ironic.conf
pxe_config_template=$pybasedir/drivers/modules/ipxe_config.template
```

The default template is **`pxe_config.template`**, so it is easy to miss the *i* to turn this into **`ipxe_config.template`**.

4.2. TROUBLESHOOT LOGIN ERRORS AFTER THE BARE METAL NODE BOOTS

When you try to log in at the login prompt on the console of the node with the **`root`** password that you set in the configurations steps, but are not able to, it indicates you are not booted in to the deployed image. You are probably stuck in the **`deploy-kernel/deploy-ramdisk`** image and the system has yet to get the correct image.

To fix this issue, verify the PXE Boot Configuration file in the **`/httpboot/pxelinux.cfg/MAC_ADDRESS`** on the Compute or Bare Metal Provisioning node and ensure that all the IP addresses listed in this file correspond to IP addresses on the Bare Metal Provisioning Network.

NOTE

The only network the Bare Metal Provisioning node knows about is the Bare Metal Provisioning Network. If one of the endpoints is not on the network, the endpoint will not be able to reach the Bare Metal Provisioning node as a part of the boot process.

For example, the kernel line in your file is as follows:

```
kernel http://192.168.200.2:8088/5a6cdbe3-2c90-4a90-b3c6-
85b449b30512/deploy_kernel selinux=0 disk=cciss/c0d0,sda,hda,vda
iscsi_target_iqn=iqn.2008-10.org.openstack:5a6cdbe3-2c90-4a90-b3c6-
85b449b30512 deployment_id=5a6cdbe3-2c90-4a90-b3c6-85b449b30512
deployment_key=VWDYDVVEFCQJNOSTO9R67HKUXUGP77CK
ironic_api_url=http://192.168.200.2:6385 troubleshoot=0 text nofb
nomodeset vga=normal boot_option=netboot ip=${ip}:${next-
server}:${gateway}:${netmask} BOOTIF=${mac}  ipa-api-
url=http://192.168.200.2:6385 ipa-driver-name=pxe_ssh boot_mode=bios
initrd=deploy_ramdisk coreos.configdrive=0 || goto deploy
```

Value in the above example `kernel` line	Corresponding information
http://192.168.200.2:8088	Parameter **`http_url`** in **`/etc/ironic/ironic.conf`** file. This IP address must be on the Bare Metal Provisioning Network.
5a6cdbe3-2c90-4a90-b3c6-85b449b30512	UUID of the baremetal node in **`ironic node-list`**.
deploy_kernel	This is the deploy kernel image in the Image service that is copied down as **`/httpboot/<NODE_UUID>/deploy_kernel`**.
http://192.168.200.2:6385	Parameter **`api_url`** in **`/etc/ironic/ironic.conf`** file. This IP address must be on the Bare Metal Provisioning Network.
pxe_ssh	The IPMI Driver in use by the Bare Metal Provisioning service for this node.
deploy_ramdisk	This is the deploy ramdisk image in the Image service that is copied down as **`/httpboot/<NODE_UUID>/deploy_ramdisk`**.

If any of these values do not correspond between the **`/httpboot/pxelinux.cfg/MAC_ADDRESS`** and the **`ironic.conf`** file, you need to update them in the **`ironic.conf`** file and restart the Bare Metal Provisioning service and then re-deploy the Bare Metal Provisioning node.

4.3. TROUBLESHOOT THE BARE METAL PROVISIONING SERVICE NOT GETTING THE RIGHT HOSTNAME

If your Bare Metal Provisioning system is not getting the right hostname, it means that **`cloud-init`** is

failing. To fix this, connect the Bare Metal Provisioning subnet to a router in the OpenStack Networking service. The requests to the meta-data agent should now be routed correctly.

4.4. TROUBLESHOOT INVALID OPENSTACK IDENTITY SERVICE CREDENTIALS WHEN EXECUTING BARE METAL PROVISIONING COMMANDS

If you are having trouble authenticating to the Identity service, check the **`identity_uri`** parameter in the **`ironic.conf`** file and make sure you remove the **`/v2.0`** from the **`keystone`** AdminURL. For example, **`identity_uri`** should be set to **`http://IP:PORT`**.

4.5. TROUBLESHOOT HARDWARE ENROLLMENT

Issues with enrolled hardware can be caused by incorrect node registration details. Ensure that property names and values have been entered correctly. Incorrect or mistyped property names will be successfully added to the node's details, but will be ignored.

Update a node's details. This example updates the amount of memory the node is registered to use to 2 GB:

```
# ironic node-update NODE_UUID replace properties/memory_mb=2048
```

4.6. TROUBLESHOOT *NO VALID HOST* ERRORS

If the Compute scheduler cannot find a suitable Bare Metal Provisioning node on which to boot an instance, a **`NoValidHost`** error can be seen in ***/var/log/nova/nova-conductor.log*** or immediately upon launch failure in the dashboard. This is usually caused by a mismatch between the resources Compute expects and the resources the Bare Metal Provisioning node provides.

1. Check the hypervisor resources that are available:

   ```
   # nova hypervisor-stats
   ```

   The resources reported here should match the resources that the Bare Metal Provisioning nodes provide.

2. Check that Compute recognizes the Bare Metal Provisioning nodes as hypervisors:

   ```
   # nova hypervisor-list
   ```

   The nodes, identified by UUID, should appear in the list.

3. Check the details for a Bare Metal Provisioning node:

   ```
   # ironic node-list
   # ironic node-show NODE_UUID
   ```

   Verify that the node's details match those reported by Compute.

4. Check that the selected flavor does not exceed the available resources of the Bare Metal Provisioning nodes:

```
nova flavor-show FLAVOR_NAME
```

5. Check the output of **ironic node-list** to ensure that Bare Metal Provisioning nodes are not in maintenance mode. Remove maintenance mode if necessary:

```
# ironic node-set-maintenance NODE_UUID off
```

6. Check the output of **ironic node-list** to ensure that Bare Metal Provisioning nodes are in an **`available`** state. Move the node to **`available`** if necessary:

```
# ironic node-set-provision-state NODE_UUID provide
```

4.7. TROUBLESHOOT HARDWARE INSPECTION

Hardware inspection can fail on Bare Metal Provisioning nodes in the **`available`** provision state.

1. Check the provision state for all nodes:

```
# ironic node-list
```

2. Move a node from **`available`** to **`manageable`** before starting inspection:

```
# ironic node-set-provision-state NODE_UUID manage
```

APPENDIX A. BARE METAL PROVISIONING DRIVERS

Bare Metal Provisioning can be configured to use one of many drivers. Each driver is made up of a provisioning method and a power management type. Some drivers require additional configuration. Each driver described in this section uses PXE for provisioning; drivers are listed by their power management type. Agent drivers support whole disk image and partition image deployment. To enable a driver or drivers for Bare Metal Provisioning, see Section 1.4.4.4, “Configure Bare Metal Provisioning Drivers”.

A.1. INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)

IPMI is an interface that provides out-of-band remote management features, including power management and server monitoring. To use this power management type, all Bare Metal Provisioning nodes require an IPMI that is connected to the shared Bare Metal Provisioning Network. Enable the **`pxe_ipmitool`** driver, and set the following information in the node’s **`driver_info`**:

- **`ipmi_address`** - The IP address of the IPMI NIC.
- **`ipmi_username`** - The IPMI user name.
- **`ipmi_password`** - The IPMI password.

A.2. DELL REMOTE ACCESS CONTROLLER (DRAC)

DRAC is an interface that provides out-of-band remote management features, including power management and server monitoring. To use this power management type, all Bare Metal Provisioning nodes require a DRAC that is connected to the shared Bare Metal Provisioning Network. Enable the **`pxe_drac`** driver, and set the following information in the node’s **`driver_info`**:

- **`drac_address`** - The IP address of the DRAC NIC.
- **`drac_username`** - The DRAC user name.
- **`drac_password`** - The DRAC password.

A.3. INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)

iRMC from Fujitsu is an interface that provides out-of-band remote management features including power management and server monitoring. To use this power management type on a Bare Metal Provisioning node, the node requires an iRMC interface that is connected to the shared Bare Metal Provisioning Network. Enable the **`pxe_irmc`** driver, and set the following information in the node’s **`driver_info`**:

- **`irmc_address`** - The IP address of the iRMC interface NIC.
- **`irmc_username`** - The iRMC user name.
- **`irmc_password`** - The iRMC password.

To use IPMI to set the boot mode or SCCI to get sensor data, you must complete the following additional steps:

1. Enable the sensor method in ***ironic.conf***:

```
# openstack-config --set /etc/ironic/ironic.conf \
   irmc sensor_method METHOD
```

Replace *METHOD* with **scci** or **ipmitool**.

2. If you enabled SCCI, install the **python-scciclient** package:

```
# yum install python-scciclient
```

3. Restart the Bare Metal Provisioning conductor service:

```
# systemctl restart openstack-ironic-conductor.service
```

NOTE

To use the iRMC driver, iRMC S4 or higher is required.

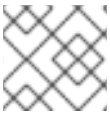
A.4. INTEGRATED LIGHTS-OUT (ILO)

iLO from Hewlett-Packard is an interface that provides out-of-band remote management features including power management and server monitoring. To use this power management type, all Bare Metal Provisioning nodes require an iLO interface that is connected to the shared Bare Metal Provisioning Network. Enable the **pxe_ilo** driver, and set the following information in the node's **driver_info**:

- **ilo_address** - The IP address of the iLO interface NIC.
- **ilo_username** - The iLO user name.
- **ilo_password** - The iLO password.

You must also install the **python-proliantutils** package and restart the Bare Metal Provisioning conductor service:

```
# yum install python-proliantutils
# systemctl restart openstack-ironic-conductor.service
```



NOTE

HP nodes must have a 2015 firmware version for successful inspection.

A.5. ACTIVE MANAGEMENT TECHNOLOGY (AMT)

AMT from Intel is an out-of-band remote management technology widely used to monitor and manage desktops, including controlling the desktop power, similar to how IPMI is used with servers.

AMT drivers use the WS-MAN protocol to interact with AMT clients.

AMT consists of two drivers:

- **pxe_amt** uses AMT for power management and deploys the user image over iSCSI from the conductor.
- **agent_amt** uses AMT for power management and deploys the user image with HTTP to the node.

WARNING

DEPRECATION NOTICE. Beginning in Red Hat OpenStack Platform 11, AMT drivers will be deprecated and no longer supported in a future release.

Before you Begin

Set up the desktop environment and AMT client. See Manually configuring the AMT Client for details.

Set up Your Environment

1. Install **openwsman-python** on the Bare Metal Provisioning service node:

```
# yum install openwsman-python
```

2. Enable the AMT driver in **ironic.conf**:

```
# openstack-config --set /etc/ironic/ironic.conf \
  enabled_drivers DRIVER
```

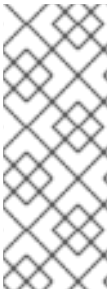
   Where *DRIVER* is either **pxe_amt** or **agent_amt**.

3. Restart the Bare Metal Provisioning conductor service:

```
# systemctl restart openstack-ironic-conductor.service
```

After you enable the AMT driver, you need to set the following information in the node's **driver_info**:

- **amt_address** - The IP address of the AMT NIC.
- **amt_username** - The AMT user name.
- **amt_password** - The AMT password.



NOTE

Bare metal nodes that use AMT drivers should be deployed with the local boot option enabled. AMT currently has no support for setting a persistent boot device. Nodes deployed without the local boot option could fail to boot if they are restarted outside the control of the Bare Metal Provisioning service. For example, a node rebooted by a local user will not attempt to PXE or network boot the kernel. An AMT node deployed with the local boot option enabled solves this issue.

A.6. SSH AND VIRSH

Bare Metal Provisioning can access a host that is running libvirt and use virtual machines as nodes. Virsh controls the power management of the nodes.

IMPORTANT

The SSH driver is for testing and evaluation purposes only. It is not recommended for Red Hat OpenStack Platform enterprise environments.

To use this power management type, Bare Metal Provisioning must have SSH access to an account with full access to the libvirt environment on the host where the virtual nodes will be set up. Enable the **pxe_ssh** driver, and set the following information in the node's **driver_info**:

- **ssh_virt_type** - Set this option to **virsh**.
- **ssh_address** - The IP address of the virsh host.
- **ssh_username** - The SSH user name.
- **ssh_key_contents** - The contents of the SSH private key on the Bare Metal Provisioning conductor node. The matching public key must be copied to the virsh host.