Red Hat OpenStack Platform 10
Manual Installation Procedures

Manual Installation Procedures for Red Hat OpenStack Platform

OpenStack Documentation Team

Red Hat OpenStack Platform 10 Manual Installation Procedures

Manual Installation Procedures for Red Hat OpenStack Platform

OpenStack Documentation Team
Red Hat Customer Content Services
rhos-docs@redhat.com

Abstract

This document is a reference for how components are installed and configured in Red Hat OpenStack Platform. It provides an instructional walkthrough of the deployment process.

Table of Contents

Chapter 1. Introduction **4**
1.1. Subscribe to the Required Channels 4
1.2. Installation Prerequisites Checklists 5

Chapter 2. Prerequisites **10**
2.1. Configure the Firewall 10
2.2. Install the Database Server 12
2.3. Install the Message Broker 14
2.4. Network Time Protocol 18
2.5. Install the OpenStack Command Line Client 18

Chapter 3. Install the Identity Service **20**
3.1. Install the Identity Service Packages 20
3.2. Create the Identity Database 20
3.3. Configure the Identity Service 21
3.4. Start the Identity Service 24
3.5. Create an Administrator Account and the Identity Service Endpoint 24
3.6. Create a Regular User Account 26
3.7. Create the Services Tenant 27
3.8. Validate the Identity Service Installation 29

Chapter 4. Install the Object Service **31**
4.1. Object Storage Service Requirements 31
4.2. Configure rsyncd 31
4.3. Install the Object Storage Service Packages 33
4.4. Configure the Object Storage Service 34
4.5. Validate the Object Storage Service Installation 41

Chapter 5. Install the Image Service **42**
5.1. Image Service Requirements 42
5.2. Install the Image Service Packages 42
5.3. Create the Image Service Database 42
5.4. Configure the Image Service 43
5.5. Launch the Image API and Registry Services 51
5.6. Validate the Image Service Installation 52

Chapter 6. Install the Block Storage Service **55**
6.1. Install the Block Storage Service Packages 55
6.2. Create the Block Storage Service Database 55
6.3. Configure the Block Storage Service 56
6.4. Configure the Volume Service 61
6.5. Launch the Block Storage Services 63
6.6. Validate the Block Storage Service Installation 63

Chapter 7. Install OpenStack Networking **66**
7.1. Install the OpenStack Networking Packages 66
7.2. Configure OpenStack Networking 66
7.3. Configure the DHCP Agent 76
7.4. Configure the Plug-in Agent 77
7.5. Create an External Network 78
7.6. Configure the L3 Agent 80
7.7. Validate the OpenStack Networking Installation 83

Chapter 8. Install the Compute Service **85**

Chapter 8. Install the Compute Service 85
8.1. Install a Compute VNC Proxy 85
8.2. Install a Compute Node 88

Chapter 9. Install the Orchestration Service 101
9.1. Install the Orchestration Service Packages 101
9.2. Configure the Orchestration Service 101
9.3. Launch the Orchestration Service 108
9.4. Deploy a Stack Using Orchestration Templates 109
9.5. Integrate Telemetry and Orchestration Services 110

Chapter 10. Install the Dashboard 111
10.1. Dashboard Service Requirements 111
10.2. Install the Dashboard Packages 111
10.3. Launch the Apache Web Service 111
10.4. Configure the Dashboard 112
10.5. Validate Dashboard Installation 117

Chapter 11. Install the Data Processing Service 119
11.1. Install the Data Processing Service Packages 119
11.2. Configure the Data Processing Service 119
11.3. Configure and Launch the Data Processing Service 122

Chapter 12. Install the Telemetry Service 124
12.1. Overview of Telemetry Service Deployment 124
12.2. Install the Telemetry Service Packages 124
12.3. Configure the MongoDB Back End and Create the Telemetry Database 125
12.4. Configure the Telemetry Service Database Connection 126
12.5. Create the Telemetry Identity Records 127
12.6. Configure Telemetry Service Authentication 128
12.7. Configure the Firewall to Allow Telemetry Service Traffic 130
12.8. Configure RabbitMQ Message Broker Settings for the Telemetry Service 130
12.9. Configure the Compute Node 131
12.10. Configure Monitored Services 132
12.11. Launch the Telemetry API and Agents 133

Chapter 13. Install the Telemetry Alarming Service 134
13.1. Install the Telemetry Alarming Service Packages 134
13.2. Create the Telemetry Alarming Service Database 134
13.3. Configure the Telemetry Alarming Service 135
13.4. Launch the Telemetry Alarming Service 138

Chapter 14. Install Time-Series-Database-as-a-Service 140
14.1. Install the Time-Series-Database-as-a-Service Packages 140
14.2. Initialize Time-Series-Database-as-a-Service 141
14.3. Configure Time-Series-Database-as-a-Service 141
14.4. Create Time-Series-Database-as-a-Service Database 142
14.5. Set Time-Series-Database-as-a-Service as the Backend for Telemetry Service 143

Chapter 15. Install the Shared File System Service 144
15.1. Shared File System Service Back End Requirements 144
15.2. Install the Shared File System Service Packages 144
15.3. Create the Shared File System Service Identity Records 144
15.4. Configure Basic Shared File System Service Settings 145
15.5. Create the Shared File System Service Database 147
15.6. Define the Shared File System Service Back End 148

15.7. Launch the Shared File System Service 148
15.8. Create a Share Type for the Defined Back End 149

Chapter 16. Install Database-as-a-Service (Technology Preview) 150
16.1. Database-as-a-Service Requirements 150
16.2. Install the Database-as-a-Service Packages 151
16.3. Configure Database-as-a-Service 151

Chapter 17. Install OpenStack Integration Test Suite 158
17.1. Install the OpenStack Integration Test Suite Packages 158
17.2. Configure the OpenStack Integration Test Suite 159

Chapter 18. Install OpenStack Benchmarking Service 162
18.1. Install the OpenStack Benchmarking Service Packages 162
18.2. Configure the OpenStack Benchmarking Service 163
18.3. Extend the Tenant Network for the Benchmarking Service 164
18.4. Validate the OpenStack Benchmarking Service 167

Chapter 1. Introduction

This document provides a reference for how components in a Red Hat OpenStack Platform environment are installed and configured. Installation and configuration information is grouped by component for the following components:

- The MariaDB Database Service
- The RabbitMQ Message Broker
- The Identity Service
- The Object Storage Service
- The Image Service
- The Block Storage Service
- OpenStack Networking
- The Compute Service
- The Orchestration Service
- The Dashboard
- The Data Processing Service
- The Telemetry Service
- The Time-Series-as-a-Service
- The File Share Service (Technology Preview)
- The Database-as-a-Service (Technology Preview)

Note

For an overview of the OpenStack components and their interfaces, see the *Architecture Guide* (https://access.redhat.com/documentation/en/red-hat-openstack-platform/).

The document includes tasks, such as database setup and firewall configuration, that are common to all components, and tasks that are specific to configuring each component.

1.1. Subscribe to the Required Channels

To install Red Hat OpenStack Platform, you must register all systems in the OpenStack environment with Red Hat Subscription Manager, and subscribe to the required channels.

Procedure 1.1. 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. Obtain detailed information about the Red Hat OpenStack Platform subscription available to you:

   ```
   # subscription-manager list --available --matches '*OpenStack
   Platform*'
   ```

   This command should print output similar to the following:

   ```
   +-------------------------------------------+
       Available Subscriptions
   +-------------------------------------------+
   Subscription Name:   Red Hat Enterprise Linux OpenStack Platform,
   Standard (2-sockets)
   Provides:            Red Hat Beta
   ...
                        Red Hat OpenStack
   ...
   SKU:                 ABC1234
   Contract:            12345678
   Pool ID:             0123456789abcdef0123456789abcdef
   Provides Management: No
   Available:           Unlimited
   Suggested:           1
   Service Level:       Standard
   Service Type:        L1-L3
   Subscription Type:   Stackable
   Ends:                12/31/2099
   System Type:         Virtual
   ```

3. Use the **Pool ID** printed by this command to attach the Red Hat OpenStack Platform entitlement:

   ```
   # subscription-manager attach --pool=Pool ID
   ```

4. Disable any irrelevant and enable the required channels:

   ```
   # subscription-manager repos --disable=* \
   --enable=rhel-7-server-rpms \
   --enable=rhel-7-server-openstack-10-rpms \
   --enable=rhel-7-server-rh-common-rpms \
   --enable=rhel-7-server-extras-rpms
   ```

5. Run the **yum update** command and reboot to ensure that the most up-to-date packages, including the kernel, are installed and running.

   ```
   # yum update
   # reboot
   ```

You have successfully configured your system to receive Red Hat OpenStack Platform packages. You may use the **yum repolist** command to confirm the repository configuration again at any time.

1.2. Installation Prerequisites Checklists

The following tables describe prerequisites for successfully installing a Red Hat OpenStack Platform environment. Checklist items are the minimum that should be known or verified before the installation is started.

The *Value/Verified* column can be used to provide the appropriate value or a 'check' that the item has been verified.

Note

If you are installing single components after the initial Red Hat OpenStack Platform installation, ensure that you have the following permissions:

- `root` access to the host machine (to install components and perform other administrative tasks such as updating the firewall).
- Administrative access to the Identity service.
- Administrative access to the database (ability to add both databases and users).

Table 1.1. OpenStack Installation: General

Item	Description	Value/Verified
Hardware requirements	Hardware requirements must be verified.	Yes | No
Operating system	Red Hat Enterprise Linux 7.1 Server	Yes | No
Red Hat subscription	You must have a subscription that entitles your systems to receive the following updates: - Package updates from the Content Delivery Network or an equivalent source such as a Red Hat Satellite server - Software updates for both Red Hat Enterprise Linux 7.1 Server and Red Hat OpenStack Platform	Yes | No
Administrative access on all installation machines	Almost all procedures in this guide must be performed as the root user, so you must have root access.	Yes | No
Red Hat subscription user name and password	You must know the Red Hat subscription user name and password.	- Name: - Password:

Item	Description	Value/Verified
Machine addresses	You must know the IP address or host name of the server or servers on which any OpenStack components and supporting software will be installed.	Provide host addresses for the following services: • Identity service • OpenStack Networking • Block Storage service • Compute service • Image service • Object Storage service • Dashboard service • Database server or servers

Table 1.2. OpenStack Identity Service

Item	Description	Value
Host access	The system hosting the Identity service must have access to the following components: • Content Delivery Network or equivalent service • Network interface addressable by all OpenStack hosts • Network access to the database server or servers • If using LDAP, network access to the directory server	Verify whether the system has access to the following components: • Yes | No • Yes | No • Yes | No • Yes | No
SSL certificates	If you are using external SSL certificates, you must know where the database and certificates are located, and have access to them.	Yes | No
LDAP information	If you are using LDAP, you must have administrative access to configure a new directory server schema.	Yes | No
Connections	The system hosting the Identity service must have a connection to all other OpenStack services.	Yes | No

Table 1.3. OpenStack Object Storage Service

Item	Description	Value
File system	Red Hat currently supports the **`XFS`** and **`ext4`** file systems for object storage; one of these must be available.	• XFS • ext4
Mount point	The **`/srv/node`** mount point must be available.	Yes | No
Connections	The system hosting the Object Storage service requires a connection to the Identity service.	Yes | No

Table 1.4. OpenStack Image Service

Item	Description	Value

Item	Description	Value
Back-end storage	The Image service supports a number of storage back ends. You must decide on one of the following: - File (local directory) - Object Storage service	Storage type:
Connections	The server hosting the Image service must have a connection to the Identity service, the dashboard service, and the Compute services. The server must also have access to the Object Storage service if it is using Object Storage as its back end.	Yes | No

Table 1.5. OpenStack Block Storage Service

Item	Description	Value
Back-end storage	The Block Storage service supports a number of storage back ends. You must decide on one of the following: - Red Hat Ceph - LVM/iSCSI - ThinLVM - NFS - NetApp - Dell EqualLogic - Dell Storage Center	Storage type:
Connections	The server hosting the Block Storage service must have a connection to the Identity service, the dashboard service, and the Compute services.	Yes | No

Table 1.6. OpenStack Networking

Item	Description	Value
Plug-in agents	In addition to the standard OpenStack Networking components, a number of plug-in agents are also available that implement various networking mechanisms. You must decide which of these apply to your network and install them.	Circle the appropriate plug-in: - Open vSwitch - Cisco UCS/Nexus - Linux Bridge - VMware NSX virtualized network platform - Ryu OpenFlow Controller - NEC OpenFlow - Big Switch Controller Plugin - Cloudbase Hyper-V - MidoNet - Brocade Neutron Plugin - PLUMgrid
Connections	The server hosting OpenStack Networking must have a connection to the Identity service, the dashboard service, and the Compute services.	Yes | No

Table 1.7. OpenStack Compute Service

Item	Description	Value
Hardware virtualization support	The Compute service requires hardware virtualization support.	Yes | No
VNC client	The Compute service supports Virtual Network Computing (VNC) console access to instances through a web browser. You must decide whether this will be provided to your users.	Yes | No
Resources: CPU and memory	OpenStack supports overcommitting of CPU and memory resources on Compute nodes: • The default CPU overcommit ratio of 16 means that up to 16 virtual cores can be assigned to a node for each physical core. • The default memory overcommit ratio of 1.5 means that instances can be assigned to a physical node if the total instance memory usage is less than 1.5 times the amount of physical memory available.	Decide: • CPU setting: • Memory setting:
Resources: host	You can reserve resources for the host, to prevent a given amount of memory and disk resources from being automatically assigned to other resources on the host.	Decide: • Host Disk (default 0MB): • Host Memory (default 512MB):
libvirt version	You must know the version of libvirt that you are using in order to configure Virtual Interface Plugging.	Version:
Connections	The server or servers hosting the Compute service must have a connection to all other OpenStack services.	Yes | No

Table 1.8. OpenStack Dashboard Service

Item	Description	Value
Host software	The system hosting the dashboard service must have the following packages already installed: • httpd • mod_wsgi • mod_ssl	Yes | No
Connections	The system hosting the dashboard service must have a connection to all other OpenStack services.	Yes | No

Chapter 2. Prerequisites

This chapter outlines how to configure all nodes to use **iptables** to provide firewall capabilities. It also explains how to install the database service and message broker used by all components in the Red Hat OpenStack Platform environment. The MariaDB database service provides the tools to create and access the databases required for each component. The RabbitMQ message broker allows internal communication between the components. Messages can be sent from and received by any component that is configured to use the message broker.

Note

Prior to deploying Red Hat OpenStack Platform, it is important to consider the characteristics of the available deployment methods. For more information, refer to the Installing and Managing Red Hat OpenStack Platform.

2.1. Configure the Firewall

Configure the server or servers hosting each component to use **iptables**. This involves disabling the Network Manager service, and configuring the server to use the firewall capabilities provided by **iptables** instead of those provided by **firewalld**. All further firewall configuration in this document uses **iptables**.

2.1.1. Disable Network Manager

OpenStack Networking does not work on systems that have the Network Manager service enabled. All steps in this procedure must be performed on each server in the environment that will handle network traffic, while logged in as the **root** user. This includes the server that will host OpenStack Networking, all network nodes, and all Compute nodes.

Procedure 2.1. Disabling the Network Manager Service

1. Verify whether Network Manager is currently enabled:

   ```
   # systemctl status NetworkManager.service | grep Active:
   ```

   A. The system displays an error if the Network Manager service is not currently installed. If this error is displayed, no further action is required to disable the Network Manager service.

   B. The system displays **Active: active (running)** if Network Manager is running, or **Active: inactive (dead)** if it is not. If Network Manager is inactive, no further action is required.

2. If Network Manager is running, stop it and then disable it:

   ```
   # systemctl stop NetworkManager.service
   # systemctl disable NetworkManager.service
   ```

3. Open each interface configuration file on the system in a text editor. Interface configuration files are found in the **/etc/sysconfig/network-scripts/** directory and have names in the format **ifcfg-*X***, where *X* is replaced by the name of the interface. Valid interface names include **eth0**, **p1p5**, and **em1**.

To ensure that the standard network service takes control of the interfaces and automatically activates them on boot, confirm that the following keys are set in each interface configuration file, or add them manually:

```
NM_CONTROLLED=no
ONBOOT=yes
```

4. Start the standard network service:

```
# systemctl start network.service
```

5. Configure the network service to start at boot time:

```
# systemctl enable network.service
```

2.1.2. Disable the firewalld Service

Disable the **`firewalld`** service for Compute and OpenStack Networking nodes, and enable the **`iptables`** service.

Procedure 2.2. Disabling the firewalld Service

1. Install the **`iptables`** service:

```
# yum install iptables-services
```

2. Review the iptables rules defined in **`/etc/sysconfig/iptables`**:

Note

You can review your current **`firewalld`** configuration:

```
# firewall-cmd --list-all
```

3. When you are satisfied with the **`iptables`** rules, disable **`firewalld`**:

```
# systemctl disable firewalld.service
```

4. Stop the **`firewalld`** service and start the **`iptables`** services:

```
# systemctl stop firewalld.service; systemctl start iptables.service;
systemctl start ip6tables.service
```

5. Configure the **`iptables`** services to start at boot time:

```
# systemctl enable iptables.service
# systemctl enable ip6tables.service
```

2.2. Install the Database Server

Each OpenStack component requires a running MariaDB database service. You must deploy the database service before deploying a full Red Hat OpenStack Platform environment or installing any single OpenStack component.

2.2.1. Install the MariaDB Database Packages

The following packages are required by the MariaDB database service:

mariadb-galera-server

Provides the MariaDB database service.

mariadb-galera-common

Provides the MariaDB service shared files. This package is installed as a dependency of the *mariadb-galera-server* package.

galera

Installs the Galera wsrep (Write Set REPlication) provider. This package is installed as a dependency of the *mariadb-galera-server* package.

Install the packages:

```
# yum install mariadb-galera-server
```

2.2.2. Configure the Firewall to Allow Database Traffic

All components in the OpenStack environment use the database server, and must be able to access it. The firewall on the server hosting the database service must be configured to allow network traffic on the required port. All steps in this procedure must be performed on the server hosting the database service, while logged in as the **`root`** user.

Procedure 2.3. Configuring the Firewall to Allow Database Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port **`3306`** to the file. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 3306 -j ACCEPT
   ```

3. Save the changes to the **`/etc/sysconfig/iptables`** file.
4. Restart the **`iptables`** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

2.2.3. Start the Database Service

All steps in this procedure must be performed on the server hosting the database service, while logged in as the **`root`** user.

Procedure 2.4. Starting the Database Service

1. Start the **`mariadb`** service:

   ```
   # systemctl start mariadb.service
   ```

2. Configure the **`mariadb`** service to start at boot time:

   ```
   # systemctl enable mariadb.service
   ```

2.2.4. Configure the Database Administrator Account

By default, MariaDB creates a database user account named **`root`** that provides access to the MariaDB service from the machine on which the MariaDB service was installed. You must set a password for this account to secure access to the server hosting the MariaDB service. You must also enable access to the MariaDB service from machines other than the machine on which the MariaDB server is installed. It is also recommended that you remove the anonymous user and test database that are created during installation.

Procedure 2.5. Configuring the Database Administrator Account

1. Log in to the machine on which the MariaDB service is installed.

2. Use the **`mysql_secure_installation`** to set the **`root`** password, allow remote root login, and remove the anonymous user account and test database:

   ```
   # mysql_secure_installation
   ```

Note

Change the password of a database user, if required. In the following example, replace *OLDPASS* with the existing password of the user and *NEWPASS* with a new password, leaving no space between **`-p`** and the old password:

```
# mysqladmin -u root -pOLDPASS password NEWPASS
```

2.2.5. Test Connectivity

To ensure that a database user account has been correctly configured, test the connectivity of that user account with the MariaDB database service from the machine on which the MariaDB service is installed (local connectivity), and from a machine other than the machine on which the MariaDB service is installed (remote connectivity).

2.2.5.1. Test Local Connectivity

Test whether you can connect to the server hosting the database service from the machine on which the MariaDB service is installed.

Procedure 2.6. Testing Local Connectivity

1. Connect to the database service, replacing **`USER`** with the user name with which to connect:

```
# mysql -u USER -p
```

2. Enter the password of the database user when prompted.

```
Enter password:
```

If the permissions for the database user are correctly configured, the connection succeeds and the MariaDB welcome screen and prompt are displayed. If the permissions for the database user are not correctly configured, an error message is displayed that explains that the database user is not allowed to connect to the database service.

2.2.5.2. Test Remote Connectivity

Test whether you can connect to the database service from a machine other than the machine on which the MariaDB service is installed.

Procedure 2.7. Testing Remote Connectivity

1. Install the MySQL client tools:

```
# yum install mysql
```

2. Connect to the database service, replacing *USER* with the database user name and *HOST* with the IP address or host name of the server hosting the database service:

```
# mysql -u USER -h HOST -p
```

3. Enter the password of the database user when prompted:

```
Enter password:
```

If the permissions for the database user are correctly configured, the connection succeeds and the MariaDB welcome screen and prompt are displayed. If the permissions for the database user are not correctly configured, an error message is displayed that explains that the database user is not allowed to connect to the database service.

2.3. Install the Message Broker

If you are deploying a full Red Hat OpenStack Platform environment, you must set up a working message broker for the following OpenStack components:

- Block Storage service
- Compute service
- OpenStack Networking
- Orchestration service
- Image service
- Telemetry service

2.3.1. Install the RabbitMQ Message Broker Package

RabbitMQ is the default (and recommended) message broker. The RabbitMQ messaging service is provided by the *rabbitmq-server* package.

Install RabbitMQ:

```
# yum install rabbitmq-server
```

2.3.2. Configure the Firewall for Message Broker Traffic

Before installing and configuring the message broker, allow incoming connections on the port it will use. The default port for message broker (AMQP) traffic is **`5672`**. All steps in this procedure must be performed on the server hosting the messaging service, while logged in as the **`root`** user.

Procedure 2.8. Configuring the Firewall for Message Broker Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.
2. Add an INPUT rule allowing incoming connections on port **`5672`**. The new rule must appear before any INPUT rules that REJECT traffic.

   ```
   -A INPUT -p tcp -m tcp --dport 5672  -j ACCEPT
   ```

3. Save the changes to the **`/etc/sysconfig/iptables`** file.
4. Restart the **`iptables`** service for the firewall changes to take effect:

   ```
   # systemctl restart iptables.service
   ```

2.3.3. Launch and Configure the RabbitMQ Message Broker

Procedure 2.9. Launching and Configuring the RabbitMQ Message Broker for Use with OpenStack

1. Launch the **`rabbitmq-server`** service and configure it to start at boot time:

   ```
   # systemctl start rabbitmq-server.service
   # systemctl enable rabbitmq-server.service
   ```

2. When the *rabbitmq-server* package is installed, a **`guest`** user with a default **`guest`** password is automatically created for the RabbitMQ service. Red Hat strongly advises that you change this default password, especially if you have IPv6 available. With IPv6, RabbitMQ may be accessible from outside the network. Change the default guest password:

   ```
   # rabbitmqctl change_password guest NEW_RABBITMQ_PASS
   ```

   Replace *NEW_RABBITMQ_PASS* with a more secure password.

3. Create a RabbitMQ user account for the Block Storage service, the Compute service, OpenStack Networking, the Orchestration service, the Image service, and the Telemetry service:

   ```
   # rabbitmqctl add_user cinder CINDER_PASS
   # rabbitmqctl add_user nova NOVA_PASS
   # rabbitmqctl add_user neutron NEUTRON_PASS
   # rabbitmqctl add_user heat HEAT_PASS
   ```

```
# rabbitmqctl add_user glance GLANCE_PASS
# rabbitmqctl add_user ceilometer CEILOMETER_PASS
```

Replace *CINDER_PASS*, *NOVA_PASS*, *NEUTRON_PASS*, *HEAT_PASS*, *GLANCE_PASS*, and *CEILOMETER_PASS* with secure passwords for each service.

4. Grant each of these RabbitMQ users read and write permissions to all resources:

```
# rabbitmqctl set_permissions cinder ".*" ".*" ".*"
# rabbitmqctl set_permissions nova ".*" ".*" ".*"
# rabbitmqctl set_permissions neutron ".*" ".*" ".*"
# rabbitmqctl set_permissions heat ".*" ".*" ".*"
# rabbitmqctl set_permissions glance ".*" ".*" ".*"
# rabbitmqctl set_permissions ceilometer ".*" ".*" ".*"
```

2.3.4. Enable SSL on the RabbitMQ Message Broker

The RabbitMQ message broker features built-in support for SSL, which you can use to secure traffic. Create the certificates required for SSL communication, and configure SSL on RabbitMQ through the **/etc/rabbitmq/rabbitmq.config** configuration file.

Procedure 2.10. Enabling SSL on the RabbitMQ Message Broker

1. Create a directory in which to store the required certificates:

```
# mkdir /etc/pki/rabbitmq
```

2. Choose a secure certificate password and store it in a file within the **/etc/pki/rabbitmq** directory:

```
# echo SSL_RABBITMQ_PW > /etc/pki/rabbitmq/certpw
```

Replace *SSL_RABBITMQ_PW* with a certificate password. This password will be used later for further securing the necessary certificates.

3. Set the permissions for the certificate directory and password file:

```
# chmod 700 /etc/pki/rabbitmq
# chmod 600 /etc/pki/rabbitmq/certpw
```

4. Create the certificate database files (***.db**) in the **/etc/pki/rabbitmq** directory, using the password in the **/etc/pki/rabbitmq/certpw** file:

```
# certutil -N -d /etc/pki/rabbitmq -f /etc/pki/rabbitmq/certpw
```

5. For a production environment, it is recommended that you use a reputable third-party Certificate Authority (CA) to sign your certificates. Create a Certificate Signing Request (CSR) for a third-party CA:

```
# certutil -R -d /etc/pki/rabbitmq -s "CN=RABBITMQ_HOST" \
 -a -f /etc/pki/rabbitmq/certpw > RABBITMQ_HOST.csr
```

Replace *RABBITMQ_HOST* with the IP or host name of the server hosting the RabbitMQ message broker. This command produces a CSR named ***RABBITMQ_HOST*.csr** and a key file (*keyfile.key*). The key file will be used later when configuring the RabbitMQ message broker to use SSL.

Note

Some CAs may require additional values other than **"CN=*RABBITMQ_HOST*"**.

6. Provide ***RABBITMQ_HOST*.csr** to your third-party CA for signing. Your CA should provide you with a signed certificate (*server.crt*) and a CA file (*ca.crt*). Add these files to your certificate database:

```
 #  certutil -A -d /etc/pki/rabbitmq -n RABBITMQ_HOST -f
/etc/pki/rabbitmq/certpw \
 -t u,u,u -a -i /path/to/server.crt
 #  certutil -A -d /etc/pki/rabbitmq -n "Your CA certificate" \
 -f /etc/pki/rabbitmq/certpw -t CT,C,C -a -i /path/to/ca.crt
```

7. Configure the RabbitMQ message broker to use the certificate files for secure communications. Open the **/etc/rabbitmq/rabbitmq.config** configuration file in a text editor, and edit the **rabbit** section as follows:

   a. Find the line that reads:

   ```
      %% {ssl_listeners, [5671]},
   ```

   Uncomment the setting by removing the percent signs:

   ```
      {ssl_listeners, [5671]},
   ```

   b. Scroll down to the line that reads:

   ```
      %% {ssl_options, [{cacertfile,
    "/path/to/testca/cacert.pem"},
   ```

   Replace this line and the next few lines which comprise the **ssl_options** section with the following content:

   ```
      {ssl_options, [{cacertfile,           "/path/to/ca.crt"},
                     {certfile,
    "/path/to/server.crt"},
                     {keyfile,
    "/path/to/keyfile.key"},
                     {verify,               verify_peer},
                     {versions,
    ['tlsv1.2','tlsv1.1',tlsv1]},
                     {fail_if_no_peer_cert, false}]}
   ```

   - Replace */path/to/ca.crt* with the absolute path to the CA certificate.
   - Replace */path/to/server.crt* with the absolute path to the signed certificate.
   - Replace */path/to/keyfile.key* with the absolute path to the key file.

8. Disable SSLv3 by editing the **rabbitmq.config** to include support for only specific TLS encryption versions:

```
{rabbit, [
{ssl_options, [{versions, ['tlsv1.2','tlsv1.1',tlsv1]}]},
]}
```

9. Restart the RabbitMQ service for the change to take effect:

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

2.3.5. Export an SSL Certificate for Clients

When SSL is enabled on a server, the clients require a copy of the SSL certificate to establish a secure connection.

The following example commands can be used to export a client certificate and the private key from the message broker's certificate database:

```
 # pk12util -o <p12exportfile> -n <certname> -d <certdir> -w <p12filepwfile>
 # openssl pkcs12 -in <p12exportfile> -out <clcertname> -nodes -clcerts -
passin pass:<p12pw>
```

For more information on SSL commands and options, see the OpenSSL Documentation. On Red Hat Enterprise Linux, see the **openssl** manual page.

2.4. Network Time Protocol

Use Network Time Protocol (NTP) on each system in your OpenStack environment to synchronize all the services. Start by configuring NTP on the controller node, and make sure the same external NTP servers that are commonly used within your organization are also set here. Then, set the rest of the systems in your OpenStack environment to take their synchronization information from the controller node.

Important

Use external NTP servers that are synchronized from various sources and routed via different networks.

If multiple controller nodes are present in your OpenStack environment, pay special attention to the synchronization of their clocks, as even a small drift can cause problems for the other systems. In such an environment, it is also worthwhile for the systems to take their synchronization information from multiple controller nodes in case one of them becomes unavailable.

Instructions on how to configure NTP are available in the System Administrator's Guide for Red Hat Enterprise Linux 7.

2.5. Install the OpenStack Command Line Client

To be able to configure OpenStack services and create users and projects using the **openstack** command line client, make sure the *python-openstackclient* package is installed:

```
# yum install python-openstackclient
```

Chapter 3. Install the Identity Service

This chapter outlines how to install and configure the OpenStack Identity service, and set up the basic user accounts and tenants required to use the service.

3.1. Install the Identity Service Packages

The Identity service requires the following packages:

openstack-keystone

Provides the OpenStack Identity service.

openstack-utils

Provides supporting utilities to assist with a number of tasks, including the editing of configuration files.

openstack-selinux

Provides OpenStack-specific SELinux policy modules.

Install the packages:

```
# yum install -y openstack-keystone \
      openstack-utils \
      openstack-selinux
```

3.2. Create the Identity Database

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

Procedure 3.1. Creating the Identity Service Database

1. Connect to the database service:

   ```
   # mysql -u root -p
   ```

2. Create the **`keystone`** database:

   ```
   mysql> CREATE DATABASE keystone;
   ```

3. Create a **`keystone`** database user and grant the user access to the**`keystone`** database:

   ```
   mysql> GRANT ALL ON keystone.* TO 'keystone'@'%' IDENTIFIED BY
   'PASSWORD';
   mysql> GRANT ALL ON keystone.* TO 'keystone'@'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
   ```

3.3. Configure the Identity Service

3.3.1. Configure the Identity Service Database Connection

The database connection string used by the Identity service is defined in the **`/etc/keystone/keystone.conf`** file. It must be updated to point to a valid database server before starting the service.

All steps in this procedure must be performed on the server hosting the Identity service, while logged in as the **`root`** user.

Procedure 3.2. Configuring the Identity Service SQL Database Connection

- Set the value of the **`connection`** configuration key:

  ```
  # openstack-config --set /etc/keystone/keystone.conf \
    sql connection mysql://USER:PASS@IP/DB
  ```

  Replace the following values:

  - Replace *USER* with the Identity service database user name, usually **`keystone`**.
  - Replace *PASS* with the password of the database user.
  - Replace *IP* with the IP address or host name of the database server.
  - Replace *DB* with the name of the Identity service database, usually **`keystone`**.

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the keystone database user was granted access when creating the keystone database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the keystone database, you must enter 'localhost'.

3.3.2. Configure the Public Key Infrastructure

3.3.2.1. Public Key Infrastructure Overview

The Identity service generates tokens, which are cryptographically signed documents that users and other services use for authentication. The tokens are signed using a private key, while the public key is made available in an X509 certificate.

The certificates and relevant configuration keys are automatically generated by the **`keystone-manage pki_setup`** command. It is, however, possible to manually create and sign the required certificates using a third party certificate authority. If using third party certificates the Identity service configuration must be manually updated to point to the certificates and supporting files.

The configuration keys relevant to PKI setup appear in the **`[signing]`** section of the **`/etc/keystone/keystone.conf`** configuration file. These keys are:

ca_certs

Specifies the location of the certificate for the authority that issued the certificate denoted by the **`certfile`** configuration key. The default value is **`/etc/keystone/ssl/certs/ca.pem`**.

ca_key

Specifies the key of the certificate authority that issued the certificate denoted by the **`certfile`** configuration key. The default value is **`/etc/keystone/ssl/certs/cakey.pem`**.

ca_password

Specifies the password, if applicable, required to open the certificate authority file. The default action if no value is specified is not to use a password.

certfile

Specifies the location of the certificate that must be used to verify tokens. The default value of **`/etc/keystone/ssl/certs/signing_cert.pem`** is used if no value is specified.

keyfile

Specifies the location of the private key that must be used when signing tokens. The default value of **`/etc/keystone/ssl/private/signing_key.pem`** is used if no value is specified.

token_format

Specifies the algorithm to use when generating tokens. Possible values are **`UUID`** and **`PKI`**. The default value is **`PKI`**.

3.3.2.2. Create the Public Key Infrastructure Files

Create and configure the PKI files to be used by the Identity service. All steps in this procedure must be performed on the server hosting the Identity service, while logged in as the **`root`** user.

Procedure 3.3. Creating the PKI Files to be Used by the Identity Service

1. Run the **`keystone-manage pki_setup`** command:

```
# keystone-manage pki_setup \
  --keystone-user keystone \
  --keystone-group keystone
```

2. Ensure that the **`keystone`** user owns the **`/var/log/keystone/`** and **`/etc/keystone/ssl/`** directories:

```
# chown -R keystone:keystone /var/log/keystone \
  /etc/keystone/ssl/
```

3.3.2.3. Configure the Identity Service to Use Public Key Infrastructure Files

After generating the PKI files for use by the Identity service, you must enable the Identity service to use them.

Set the values of the attributes in the **/etc/keystone/keystone.conf** file:

```
# openstack-config --set /etc/keystone/keystone.conf \
 signing token_format PKI
# openstack-config --set /etc/keystone/keystone.conf \
 signing certfile /etc/keystone/ssl/certs/signing_cert.pem
# openstack-config --set /etc/keystone/keystone.conf \
 signing keyfile /etc/keystone/ssl/private/signing_key.pem
# openstack-config --set /etc/keystone/keystone.conf \
 signing ca_certs /etc/keystone/ssl/certs/ca.pem
# openstack-config --set /etc/keystone/keystone.conf \
 signing key_size 1024
# openstack-config --set /etc/keystone/keystone.conf \
 signing valid_days 3650
# openstack-config --set /etc/keystone/keystone.conf \
 signing ca_password None
```

You can also update these values directly by editing the **/etc/keystone/keystone.conf** file.

3.3.3. Configure the Firewall to Allow Identity Service Traffic

Each component in the OpenStack environment uses the Identity service for authentication and must be able to access the service.

The firewall on the system hosting the Identity service must be altered to allow network traffic on the required ports. All steps in this procedure must be run on the server hosting the Identity service, while logged in as the **root** user.

Procedure 3.4. Configuring the Firewall to Allow Identity Service Traffic

1. Open the **/etc/sysconfig/iptables** file in a text editor.
2. Add an INPUT rule allowing TCP traffic on ports **5000** and **35357** to the file. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 5000,35357 -j ACCEPT
   ```

3. Save the changes to the **/etc/sysconfig/iptables** file.
4. Restart the **iptables** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

3.3.4. Populate the Identity Service Database

Populate the Identity service database after you have successfully configured the Identity service database connection string.

Procedure 3.5. Populating the Identity Service Database

1. Log in to the system hosting the Identity service.

2. Switch to the **keystone** user and initialize and populate the database identified in **/etc/keystone/keystone.conf**:

```
# su keystone -s /bin/sh -c "keystone-manage db_sync"
```

3.3.5. Limit the Number of Entities in a Collection

Use this procedure to set a limit on the number of results returned by list commands. You can use a lower limit to avoid problems when the number of results is larger than available memory or to avoid a long list's response times.

Procedure 3.6. Limiting the Number of Entities in a Collection

1. Open the **/etc/keystone/keystone.conf** in a text editor.

2. Set a global value using **list_limit** in the **[DEFAULT]** section.

3. Optionally override the global value with a specific limit in individual sections. For example:

```
[assignment]
list_limit = 100
```

If a response to a **list_{entity}** call has been truncated, the response status code will still be 200 (OK), but the **truncated** attribute in the collection will be set to **true**.

3.4. Start the Identity Service

All steps in this procedure must be performed on the server hosting the Identity service, while logged in as the **root** user.

Procedure 3.7. Launching the Identity Service

1. Start the **httpd** service:

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

2. Configure the **httpd** service to start at boot time:

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

3.5. Create an Administrator Account and the Identity Service Endpoint

The following procedure creates an administrative user and an associated tenant and role. The Identity service endpoint is created at the same time.

All steps in this procedure must be performed on the system hosting the Identity service.

Procedure 3.8. Creating an Administrator Account and the Identity Service Endpoint

1. Create an **admin** user, role, and tenant:

```
 #  keystone-manage bootstrap \
--bootstrap-password PASSWORD \
--bootstrap-username admin \
--bootstrap-project-name admin \
--bootstrap-role-name admin \
--bootstrap-service-name keystone \
--bootstrap-region-id RegionOne \
--bootstrap-admin-url http://IP:35357 \
--bootstrap-public-url http://IP:5000 \
--bootstrap-internal-url http://IP:5000
```

Replace *PASSWORD* with the password of the **admin** user, and replace *IP* with the IP address or host name of the Identity server.

2. The newly-created **admin** account will be used for future management of the Identity service. To facilitate authentication, create a **keystonerc_admin** file in a secure location such as the home directory of the **root** user.

   Add these lines to the file to set the environment variables that will be used for authentication:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_PASSWORD=PASSWORD
export OS_AUTH_URL=http://IP:35357/v2.0/
export PS1='[\u@\h \W(keystone_admin)]\$ '
```

   Again, replace *PASSWORD* with the password of the **admin** user, and replace *IP* with the IP address or host name of the Identity server.

3. Load the environment variables used for authentication:

```
 #  source ~/keystonerc_admin
```

Note

Red Hat recommends creating the administrator account using the **keystone-manage bootstrap** command, which obsoletes previously used *administration tokens*. If you still want to set up and use an administration token, see the instructions in the Create an Administrator Account and the Identity Service Endpoint section of the *Manual Installation Procedures* for Red Hat OpenStack Platform 10.

3.5.1. Service Regions

Each service cataloged in the Identity service is identified by its region, which typically represents a geographical location, and its endpoint. In a Red Hat OpenStack Platform environment with multiple Compute deployments, regions allow for the discrete separation of services, and are a robust way to share some infrastructure between Compute installations, while allowing for a high degree of failure tolerance.

Administrators determine which services are shared between regions and which services are used only with a specific region. By default, when an endpoint is defined and no region is specified, it is created in the region named **RegionOne**.

To begin using separate regions, specify the ***--region*** argument when adding service endpoints:

```
[(keystone_admin)]# openstack endpoint create --region REGION \
   --publicurl PUBLICURL \
   --adminurl ADMINURL \
   --internalurl INTERNALURL \
   SERVICENAME
```

Replace *REGION* with the name of the region to which the endpoint belongs. When sharing an endpoint between regions, create an endpoint entry containing the same URLs for each applicable region. For information on setting the URLs for each service, see the Identity service configuration information of the service in question.

Example 3.1. Endpoints Within Discrete Regions

In this example, the **APAC** and **EMEA** regions share an Identity server (**identity.example.com**) endpoint, while providing region specific compute API endpoints:

```
 $ openstack endpoint list --long
+--------+-----------+-------------+-------------+------------------
----------------------------------+...
| ID     | Region    | Service Name | Service Type | PublicURL
|...
+--------+-----------+-------------+-------------+------------------
----------------------------------+...
| b02... | APAC      | compute     | compute      | http://nova-
apac.example.com:8774/v2/%(tenant_id)s  |...
| c46... | APAC      | keystone    | identity     |
http://identity.example.com:5000/v3                  |...
| 31d... | EMEA      | compute     | compute      | http://nova-
emea.example.com:8774/v2/%(tenant_id)s  |...
| 727... | EMEA      | keystone    | identity     |
http://identity.example.com:5000/v3                  |...
+--------+-----------+-------------+-------------+------------------
----------------------------------+...
```

3.6. Create a Regular User Account

Create a regular user and tenant. Tenants are used to aggregate service resources, and are also known as projects.

All steps in this procedure must be performed on the system hosting the Identity service.

Procedure 3.9. Creating a Regular User Account

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

   ```
   # source ~/keystonerc_admin
   ```

2. Create a tenant:

   ```
   [(keystone_admin)]# openstack project create TENANT
   +-------------+----------------------------------+
   | Field       | Value                            |
   ```

```
+-------------+----------------------------------+
| description | None                             |
| enabled     | True                             |
| id          | 99c674e3fead4237ace2a0d86dab76e4 |
| name        | TENANT                           |
+-------------+----------------------------------+
```

Replace *TENANT* with a name for the tenant.

3. Create a regular user:

```
[(keystone_admin)]# openstack user create --project TENANT --password
PASSWORD USER
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| email      | None                             |
| enabled    | True                             |
| id         | 246b1342a8684bf39d7cc5165ef835d4 |
| name       | USER                             |
| project_id | 99c674e3fead4237ace2a0d86dab76e4 |
| username   | USER                             |
+------------+----------------------------------+
```

Replace *USER* with a user name for the account. Replace *TENANT* with the tenant name that you used in the previous step. Replace *PASSWORD* with a secure password for the account.

Note

The user is associated with Identity's default **`_member_`** role automatically thanks to the **`--project`** option.

4. To facilitate authentication, create a **`keystonerc_`*`user`*** file in a secure location (for example, the home directory of the **`root`** user).

Set the following environment variables to be used for authentication:

```
export OS_USERNAME=USER
export OS_TENANT_NAME=TENANT
export OS_PASSWORD=PASSWORD
export OS_AUTH_URL=http://IP:5000/v2.0/
export PS1='[\u@\h \W(keystone_user)]\$ '
```

Replace *USER*, *TENANT*, and *PASSWORD* with the values specified during tenant and user creation. Replace *IP* with the IP address or host name of the Identity server.

3.7. Create the Services Tenant

Per tenant, quota controls can be used to limit the numbers of resources.

Note

For more information about quotas, see the "Manage Projects" section in the Red Hat OpenStack Platform *Administration Guide*. This document is available from the following page:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

Each user is assigned to a tenant. For regular users, the tenant typically represents their group, project, or organisation. For service users (the entity accessing the Identity service on behalf of the service), the tenant represents a service's geographical region. If the services in your environment are distributed, typically one service tenant is created for each endpoint on which services are running (excepting the Identity and dashboard services). If the services in your environment are deployed on a single node, only one service tenant is required, though it is possible to create more for administrative purposes.

The service setup examples in this guide assume that all services are deployed on one node, therefore only one service tenant is required. All such examples use the **`services`** tenant.

Note

Because administrators, regular users, and service users all need a tenant, at least three tenants are typically created, one for each group. To create administrative and regular users and tenants, see Section 3.5, “Create an Administrator Account and the Identity Service Endpoint” and Section 3.6, “Create a Regular User Account”.

Procedure 3.10. Creating the Services Tenant

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

```
# source ~/keystonerc_admin
```

2. Create the **`services`** tenant:

```
[(keystone_admin)]# openstack project create --description "Services
Tenant" services
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Services Tenant                  |
| enabled     | True                             |
| id          | 42e1efb4bd5e49a49cb2b346078d6325 |
| name        | services                         |
+-------------+----------------------------------+
```

Note

To obtain a list of all Identity service tenants and their IDs, run:

```
 [(keystone_admin)]# openstack project list
+----------------------------------+----------+
| ID                               | Name     |
+----------------------------------+----------+
| 42e1efb4bd5e49a49cb2b346078d6325 | services |
| 65d8216d98c64399b8f44929b634bc3f | admin    |
| 99c674e3fead4237ace2a0d86dab76e4 | test     |
+----------------------------------+----------+
```

3.8. Validate the Identity Service Installation

Verify that an Identity service installation is functioning correctly. All steps in this procedure must be performed on the Identity server or on another server in the environment. The logged-in user must have access to **keystonerc_admin** and **keystonerc_user** files containing the environment variables required to authenticate as the administrative user and a regular user respectively. Also, the system must have the following already installed: httpd, mod_wsgi, and mod_ssl (for security purposes).

Procedure 3.11. Validating the Identity Service Installation

1. Set up the shell to access keystone as the adminstrative user:

   ```
   # source ~/keystonerc_admin
   ```

2. List the users defined in the system:

   ```
    [(keystone_admin)]# openstack user list
   +----------------------------------+-------+
   | ID                               | Name  |
   +----------------------------------+-------+
   | 23c56d02d3bc4b88b034e0b3720fcd1b | admin |
   | 246b1342a8684bf39d7cc5165ef835d4 | USER  |
   +----------------------------------+-------+
   ```

   The list of users defined in the system is displayed. If the list is not displayed, there is an issue with the installation.

   a. If the message returned indicates a permissions or authorization issue, check that the administrative user account, tenant, and role were created properly. Also ensure that the three objects are linked correctly.

   b. If the message returned indicates a connectivity issue (**Connection refused**), verify that the **openstack-keystone** service is running and that the firewall service is configured to allow connections on ports **5000** and **35357**.

3. Set up the shell to access keystone as the regular Identity service user:

   ```
   # source ~/keystonerc_user
   ```

4. Attempt to list the users defined in the system:

   ```
   [(keystone_user)]# openstack user list
   You are not authorized to perform the requested action: admin_required
   (HTTP 403) (Request-ID: req-1cfd3869-ac97-424d-bd00-f835a6ab9be6)
   ```

   An error message is displayed indicating that the user is not an administrator. If the error message is not displayed, but the user list appears instead, then the regular user account was incorrectly attached to the **admin** role.

3.8.1. Troubleshoot Identity Client (keystone) Connectivity Problems

When the Identity client (**keystone**) is unable to contact the Identity service, it returns an error:

```
Unable to communicate with identity service: [Errno 113] No route to host.
(HTTP 400)
```

To debug the issue, check for these common causes:

Identity service is down

On the system hosting the Identity service, check the service status:

```
# systemctl status openstack-keystone
● openstack-keystone.service - OpenStack Identity Service (code-named
Keystone)
   Loaded: loaded (/usr/lib/systemd/system/openstack-
keystone.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2016-06-07 02:31:14 EDT; 5h
29min ago
 Main PID: 23236 (keystone-all)
   CGroup: /system.slice/openstack-keystone.service
           ├─23236 /usr/bin/python2 /usr/bin/keystone-all
           ├─23247 /usr/bin/python2 /usr/bin/keystone-all
           ├─23248 /usr/bin/python2 /usr/bin/keystone-all
           ├─23249 /usr/bin/python2 /usr/bin/keystone-all
           └─23250 /usr/bin/python2 /usr/bin/keystone-all

Jun 07 02:31:13 mitaka.localdomain systemd[1]: Starting OpenStack
Identity Service (code-named Keystone)...
Jun 07 02:31:14 mitaka.localdomain systemd[1]: Started OpenStack
Identity Service (code-named Keystone).
```

If the service is not running (the output reads **Active: inactive (dead)**), log in as the **root** user and start it:

```
# systemctl start openstack-keystone
```

Firewall is not configured properly

The firewall might not be configured to allow TCP traffic on ports **5000** and **35357**. See Section 3.3.3, "Configure the Firewall to Allow Identity Service Traffic" for instructions on how to correct this.

Chapter 4. Install the Object Service

4.1. Object Storage Service Requirements

The following items are requirements for installing the Object Storage service:

Supported Filesystems

The Object Storage service stores objects in filesystems. Currently, **`XFS`** and **`ext4`** are supported. Your filesystem must be mounted with *Extended Attributes* (**`xattr`**) enabled.

It is recommended that you use **`XFS`**. Configure this in **`/etc/fstab`**:

Example 4.1. Sample /etc/fstab Entry for One XFS Storage Disk

```
/dev/sdb1 /srv/node/d1 xfs inode64,noatime,nodiratime 0 0
```

Note

Extended Attributes are already enabled on **`XFS`** by default. As such, you do not need to specify **`user_xattr`** in your **`/etc/fstab`** entry.

Acceptable Mountpoints

The Object Storage service expects devices to be mounted at **`/srv/node/`**.

4.2. Configure rsyncd

To ensure replication, you must set up **`rsyncd`** for your filesystems before you install and configure the Object Storage service. The following procedure must be performed on each storage node, while logged in as the **`root`** user. The procedure assumes that at least two XFS storage disks have been mounted on each storage node.

Example 4.2. Sample /etc/fstab Entry for Two XFS Storage Disks

```
/dev/sdb1 /srv/node/d1 xfs inode64,noatime,nodiratime 0 0
/dev/sdb2 /srv/node/d2 xfs inode64,noatime,nodiratime 0 0
```

Procedure 4.1. Configuring rsyncd

1. Copy addresses from the controller's **`/etc/hosts`** file, and add storage node IP addresses. Also ensure that all nodes have all addresses in their **`/etc/hosts`** file.

2. Install the *rsync* and *xinetd* packages:

```
# yum install rsync xinetd
```

3. Open the **/etc/rsyncd.conf** file in a text editor, and add the following lines:

```
##assumes 'swift' has been used as the Object Storage user/group
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
##address on which the rsync daemon listens
address = LOCAL_MGT_NETWORK_IP

[account]
max connections = 2
path = /srv/node/
read only = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
lock file = /var/lock/object.lock
```

Note

Multiple account, container, and object sections can be used.

4. Open the **/etc/xinetd.d/rsync** file, and add the following lines:

```
service rsync
 {
     port              = 873
     disable           = no
     socket_type       = stream
     protocol          = tcp
     wait              = no
```

```
        user                = root
        group               = root
        groups              = yes
        server              = /usr/bin/rsync
        bind                = LOCAL_MGT_NETWORK_IP
        server_args = --daemon --config /etc/rsync.conf
    }
```

5. Start the **`xinetd`** service, and configure it to start at boot time:

```
# systemctl start xinetd.service
# systemctl enable xinetd.service
```

4.3. Install the Object Storage Service Packages

The following packages provide the components of the Object Storage service:

Primary OpenStack Object Storage Packages

openstack-swift-proxy

Proxies requests for objects.

openstack-swift-object

Stores data objects of up to 5GB.

openstack-swift-container

Maintains a database that tracks all of the objects in each container.

openstack-swift-account

Maintains a database that tracks all of the containers in each account.

OpenStack Object Storage Dependencies

openstack-swift

Contains code common to the specific services.

openstack-swift-plugin-swift3

The swift3 plugin for OpenStack Object Storage.

memcached

Soft dependency of the proxy server, caches authenticated clients rather than making them reauthorize at every interaction.

openstack-utils

Provides utilities for configuring OpenStack.

python-swiftclient

Provides the **`swift`** command-line tool.

Procedure 4.2. Installing the Object Storage Service Packages

- Install the required packages:

```
# yum install -y openstack-swift-proxy \
  openstack-swift-object \
  openstack-swift-container \
  openstack-swift-account \
  openstack-utils \
  memcached \
  python-swiftclient
```

4.4. Configure the Object Storage Service

4.4.1. Create the Object Storage Service Identity Records

Create and configure Identity service records required by the Object Storage service. These entries provide authentication for the Object Storage service, and guide other OpenStack services attempting to locate and access the functionality provided by the Object Storage service.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 4.3. Creating Identity Records for the Object Storage Service

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

   ```
   # source ~/keystonerc_admin
   ```

2. Create the **`swift`** user:

   ```
   [(keystone_admin)]# openstack user create --password PASSWORD swift
   +----------+----------------------------------+
   | Field    | Value                            |
   +----------+----------------------------------+
   | email    | None                             |
   | enabled  | True                             |
   | id       | 00916f794cec438ea7f14ee0769e6964 |
   | name     | swift                            |
   | username | swift                            |
   +----------+----------------------------------+
   ```

   Replace PASSWORD with a secure password that will be used by the Object Storage service when authenticating with the Identity service.

3. Link the **`swift`** user and the **`admin`** role together within the context of the **`services`** tenant:

```
[(keystone_admin)]# openstack role add --project services --user swift
admin
```

4. Create the **swift** Object Storage service entry:

```
[(keystone_admin)]# openstack service create --name swift \
    --description "Swift Storage Service" \
    object-store
```

5. Create the **swift** endpoint entry:

```
[(keystone_admin)]# openstack endpoint create \
    --publicurl 'http://IP:8080/v1/AUTH_%(tenant_id)s' \
    --adminurl 'http://IP:8080/v1' \
    --internalurl 'http://IP:8080/v1/AUTH_%(tenant_id)s' \
    --region RegionOne \
    swift
```

Replace *IP* with the IP address or fully qualified domain name of the server hosting the Object Storage Proxy service.

4.4.2. Configure the Object Storage Service Storage Nodes

The Object Storage service stores objects on the filesystem, usually on a number of connected physical storage devices. All of the devices that will be used for object storage must be formatted **ext4** or **XFS**, and mounted under the **/srv/node/** directory. All of the services that will run on a given node must be enabled, and their ports opened.

Although you can run the proxy service alongside the other services, the proxy service is not covered in this procedure.

Procedure 4.4. Configuring the Object Storage Service Storage Nodes

1. Format your devices using the **ext4** or **XFS** filesystem. Ensure that **xattr**s are enabled.

2. Add your devices to the **/etc/fstab** file to ensure that they are mounted under **/srv/node/** at boot time. Use the **blkid** command to find your device's unique ID, and mount the device using its unique ID.

Note

If using **ext4**, ensure that extended attributes are enabled by mounting the filesystem with the **user_xattr** option. (In **XFS**, extended attributes are enabled by default.)

3. Configure the firewall to open the TCP ports used by each service running on each node. By default, the account service uses port 6202, the container service uses port 6201, and the object service uses port 6200.

   a. Open the **/etc/sysconfig/iptables** file in a text editor.

b. Add an **`INPUT`** rule allowing TCP traffic on the ports used by the account, container, and object service. The new rule must appear before any **`reject-with icmp-host-prohibited`** rule:

```
-A INPUT -p tcp -m multiport --dports 6200,6201,6202,873 -j
ACCEPT
```

c. Save the changes to the **`/etc/sysconfig/iptables`** file.

d. Restart the **`iptables`** service for the firewall changes to take effect:

```
# systemctl restart iptables.service
```

4. Change the owner of the contents of **`/srv/node/`** to **`swift:swift`**:

```
# chown -R swift:swift /srv/node/
```

5. Set the **`SELinux`** context correctly for all directories under **`/srv/node/`**:

```
# restorecon -R /srv
```

6. Add a hash prefix to the **`/etc/swift/swift.conf`** file:

```
# openstack-config --set /etc/swift/swift.conf swift-hash
swift_hash_path_prefix \
   $(openssl rand -hex 10)
```

7. Add a hash suffix to the **`/etc/swift/swift.conf`** file:

```
# openstack-config --set /etc/swift/swift.conf swift-hash
swift_hash_path_suffix \
   $(openssl rand -hex 10)
```

8. Set the IP address that the storage services will listen on. Run the following commands for every service on every node in your Object Storage cluster:

```
# openstack-config --set /etc/swift/object-server.conf \
   DEFAULT bind_ip NODE_IP_ADDRESS
# openstack-config --set /etc/swift/account-server.conf \
   DEFAULT bind_ip NODE_IP_ADDRESS
# openstack-config --set /etc/swift/container-server.conf \
   DEFAULT bind_ip NODE_IP_ADDRESS
```

Replace *NODE_IP_ADDRESS* with the IP address of the node you are configuring.

9. Copy **`/etc/swift/swift.conf`** from the node you are currently configuring to all of your Object Storage service nodes.

Important

The **`/etc/swift/swift.conf`** file must be identical on all of your Object Storage service nodes.

10. Start the services that will run on the node:

    ```
    # systemctl start openstack-swift-account.service
    # systemctl start openstack-swift-container.service
    # systemctl start openstack-swift-object.service
    ```

11. Configure the services to start at boot time:

    ```
    # systemctl enable openstack-swift-account.service
    # systemctl enable openstack-swift-container.service
    # systemctl enable openstack-swift-object.service
    ```

4.4.3. Configure the Object Storage Service Proxy Service

The Object Storage proxy service determines to which node **`gets`** and **`puts`** are directed.

Although you can run the account, container, and object services alongside the proxy service, only the proxy service is covered in the following procedure.

Note

Because the SSL capability built into the Object Storage service is intended primarily for testing, it is not recommended for use in production. In a production cluster, Red Hat recommends that you use the load balancer to terminate SSL connections.

Procedure 4.5. Configuring the Object Storage Service Proxy Service

1. Update the configuration file for the proxy server with the correct authentication details for the appropriate service user:

   ```
   # openstack-config --set /etc/swift/proxy-server.conf \
      filter:authtoken auth_host IP
   # openstack-config --set /etc/swift/proxy-server.conf \
      filter:authtoken admin_tenant_name services
   # openstack-config --set /etc/swift/proxy-server.conf \
      filter:authtoken admin_user swift
   # openstack-config --set /etc/swift/proxy-server.conf \
      filter:authtoken admin_password PASSWORD
   ```

   Replace the following values:

   - Replace *IP* with the IP address or host name of the Identity server.
   - Replace *services* with the name of the tenant that was created for the Object Storage service (previous examples set this to **`services`**).

- Replace *swift* with the name of the service user that was created for the Object Storage service (previous examples set this to **`swift`**).
- Replace *PASSWORD* with the password associated with the service user.

2. Start the **`memcached`** and **`openstack-swift-proxy`** services:

```
# systemctl start memcached.service
# systemctl start openstack-swift-proxy.service
```

3. Configure the **`memcached`** and **`openstack-swift-proxy`** services to start at boot time:

```
# systemctl enable memcached.service
# systemctl enable openstack-swift-proxy.service
```

4. Allow incoming connections to the server hosting the Object Storage proxy service. Open the **`/etc/sysconfig/iptables`** file in a text editor, and Add an INPUT rule allowing TCP traffic on port 8080. The new rule must appear before any INPUT rules that REJECT traffic: :

```
-A INPUT -p tcp -m multiport --dports 8080 -j ACCEPT
```

Important

This rule allows communication from all remote hosts to the system hosting the Swift proxy on port **`8080`**. For information regarding the creation of more restrictive firewall rules, see the*Red Hat Enterprise Linux Security Guide*:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

5. Restart the **`iptables`** service to ensure that the change takes effect:

```
# systemctl restart iptables.service
```

4.4.4. Object Storage Service Rings

Rings determine where data is stored in a cluster of storage nodes. Ring files are generated using the **swift-ring-builder** tool. Three ring files are required, one each for the **object**, **container**, and **account** services.

Each storage device in a cluster is divided into partitions, with a recommended minimum of 100 partitions per device. Each partition is physically a directory on disk. A configurable number of bits from the MD5 hash of the filesystem path to the partition directory, known as the *partition power*, is used as a partition index for the device. The *partition count* of a cluster that has 1000 devices, where each device has 100 partitions on it, is 100,000.

The partition count is used to calculate the partition power, where 2 to the partition power is the partition count. If the partition power is a fraction, it is rounded up. If the partition count is 100,000, the part power is 17 (16.610 rounded up). This can be expressed mathematically as: $2^{\text{partition power}}$ = partition count.

4.4.5. Build Object Storage Service Ring Files

Three ring files need to be created: one to track the objects stored by the Object Storage Service, one to track the containers in which objects are placed, and one to track which accounts can access which containers. The ring files are used to deduce where a particular piece of data is stored.

Ring files are generated using four possible parameters: partition power, replica count, zone, and the amount of time that must pass between partition reassignments.

Table 4.1. Parameters Used when Building Ring Files

Ring File Parameter	Description
part_power	$2^{\text{partition power}}$ = partition count. The partition is rounded up after calculation.
replica_count	The number of times that your data will be replicated in the cluster.
min_part_hours	Minimum number of hours before a partition can be moved. This parameter increases availability of data by not moving more than one copy of a given data item within that min_part_hours amount of time.
zone	Used when adding devices to rings (optional). Zones are a flexible abstraction, where each zone should be separated from other zones as possible in your deployment. You can use a zone to represent sites, cabinet, nodes, or even devices.

Procedure 4.6. Building Object Storage Service Ring Files

1. Build one ring for each service. Provide a builder file, a *partition power*, a *replica count*, and the *minimum hours between partition reassignment*:

```
# swift-ring-builder /etc/swift/object.builder create part_power
replica_count min_part_hours
# swift-ring-builder /etc/swift/container.builder create part_power
replica_count min_part_hours
# swift-ring-builder /etc/swift/account.builder create part_power
replica_count min_part_hours
```

2. When the rings are created, add devices to the account ring:

```
# swift-ring-builder /etc/swift/account.builder add
zX-SERVICE_IP:6202/dev_mountpt part_count
```

Replace the following values:

- Replace *X* with the corresponding integer of a specified zone (for example, `z1` would correspond to Zone One).
- Replace *SERVICE_IP* with the IP on which the account, container, and object services should listen. This IP should match the `bind_ip` value set during the configuration of the Object Storage service storage nodes.
- Replace *dev_mountpt* with the `/srv/node` subdirectory under which your device is mounted.
- Replace *part_count* with the partition count you used to calculate your partition power.

Note

Repeat this step for each device (on each node in the cluster) you want added to the ring.

3. Add each device to both the container and object rings:

```
# swift-ring-builder /etc/swift/container.builder add
zX-SERVICE_IP:6201/dev_mountpt part_count
# swift-ring-builder /etc/swift/object.builder add
zX-SERVICE_IP:6200/dev_mountpt part_count
```

   Replace the variables with the same ones used in the previous step.

   Note

   Repeat these commands for each device (on each node in the cluster) you want added to the ring.

4. Distribute the partitions across the devices in the ring:

```
# swift-ring-builder /etc/swift/account.builder rebalance
# swift-ring-builder /etc/swift/container.builder rebalance
# swift-ring-builder /etc/swift/object.builder rebalance
```

5. Check to see that you now have three ring files in the directory **/etc/swift**:

```
# ls /etc/swift/*gz
```

   The files should be listed as follows:

```
/etc/swift/account.ring.gz  /etc/swift/container.ring.gz
/etc/swift/object.ring.gz
```

6. Restart the **openstack-swift-proxy** service:

```
# systemctl restart openstack-swift-proxy.service
```

7. Ensure that all files in the **/etc/swift/** directory, including those that you have just created, are owned by the **root** user and the **swift** group:

   Important

   All mount points must be owned by **root**; all roots of mounted file systems must be owned by **swift**. Before running the following command, ensure that all devices are already mounted and owned by **root**.

```
# chown -R root:swift /etc/swift
```

8. Copy each ring builder file to each node in the cluster, storing them under **`/etc/swift/`**.

4.5. Validate the Object Storage Service Installation

After installing and configuring the Object Storage service, you must validate it. The following procedure must be performed on the server hosting the proxy service, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the *python-swiftclient* package is installed.

Procedure 4.7. Validating the Object Storage Service Installation

1. On the proxy server node, turn on debug level logging:

```
 # openstack-config --set /etc/swift/proxy-server.conf DEFAULT
log_level debug
```

2. Restart the **`rsyslog`** service and the **`openstack-swift-proxy`** service:

```
 # systemctl restart rsyslog.service
 # systemctl restart openstack-swift-proxy.service
```

3. Set up the shell to access Keystone as the administrative user:

```
 # source ~/keystonerc_admin
```

4. Ensure that you can connect to the proxy server:

```
 [(keystone_admin)]# swift list
  Message from syslogd@example-swift-01 at Jun 14 02:46:00 ...
  135 proxy-server Server reports support for api versions: v3.0, v2.0
```

5. Upload some files to your Object Storage service nodes:

```
 [(keystone_admin)]# head -c 1024 /dev/urandom > data1.file ; swift
upload c1 data1.file
 [(keystone_admin)]# head -c 1024 /dev/urandom > data2.file ; swift
upload c1 data2.file
 [(keystone_admin)]# head -c 1024 /dev/urandom > data3.file ; swift
upload c1 data3.file
```

6. List the objects stored in the Object Storage service cluster:

```
 [(keystone_admin)]# swift list
 [(keystone_admin)]# swift list c1
data1.file
data2.file
data3.file
```

Chapter 5. Install the Image Service

5.1. Image Service Requirements

To install the Image service, you must have access to the following credentials and information:

- The root credentials and IP address of the server hosting the MariaDB database service
- The administrative user credentials and endpoint URL of the Identity service

If you are using the OpenStack Object Storage service as the storage back end, you will also need to know that service's endpoint public URL. This endpoint is configured as part of Section 4.4.1, “Create the Object Storage Service Identity Records”.

5.2. Install the Image Service Packages

The OpenStack Image service requires the following packages:

openstack-glance

Provides the OpenStack Image service.

openstack-utils

Provides supporting utilities to assist with a number of tasks, including the editing of configuration files.

openstack-selinux

Provides OpenStack-specific SELinux policy modules.

Install the packages:

```
# yum install -y openstack-glance openstack-utils openstack-selinux
```

5.3. Create the Image Service Database

Create the database and database user used by the Image service. All steps must be performed on the database server, while logged in as the **`root`** user.

Procedure 5.1. Creating the Image Service Database

1. Connect to the database service:

   ```
   # mysql -u root -p
   ```

2. Create the **`glance`** database:

   ```
   mysql> CREATE DATABASE glance;
   ```

3. Create a **`glance`** database user and grant the user access to the**`glance`** database:

```
mysql> GRANT ALL ON glance.* TO 'glance'@'%' IDENTIFIED BY
'PASSWORD';
mysql> GRANT ALL ON glance.* TO 'glance'@'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
```

5.4. Configure the Image Service

To configure the Image service, the following tasks must be completed:

- Configure the Identity service for Image service authentication (create database entries, set connection strings, and update configuration files).
- Configure the disk-image storage back end (this guide uses the Object Storage service).
- Configure the firewall for Image service access.
- Configure TLS/SSL.
- Populate the Image service database.

5.4.1. Configure the Image Service Database Connection

The database connection string used by the Image service is defined in the **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf** files. It must be updated to point to a valid database server before starting the service.

All steps in this procedure must be performed on the server hosting the Image service, while logged in as the **root** user.

Procedure 5.2. Configuring the Image Service SQL Database Connection

1. Set the value of the **sql_connection** configuration key in the **glance-api.conf** file:

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace the following values:

- Replace *USER* with the Image service database user name, usually **glance**.
- Replace *PASS* with the password of the database user.
- Replace *IP* with the IP address or host name of the server hosting the database service.

- Replace *DB* with the name of the Image service database, usually **`glance`**.

2. Set the value of the **`sql_connection`** configuration key in the **`glance-registry.conf`** file:

```
# openstack-config --set /etc/glance/glance-registry.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace *USER*, *PASS*, *IP*, and *DB* with the same values used in the previous step.

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Image service database user was granted access when creating the Image service database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the Image service database, you must enter 'localhost'.

5.4.2. Create the Image Service Identity Records

Create and configure Identity service records required by the Image service. These entries assist other OpenStack services attempting to locate and access the volume functionality provided by the Image service.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 5.3. Creating Identity Records for the Image Service

1. Set up the shell to access Keystone as the admin user:

```
# source ~/keystonerc_admin
```

2. Create the **`glance`** user:

```
[(keystone_admin)]# openstack user create --password PASSWORD glance
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | b1f665b15a7943ccb4668c9e78e98a7c |
| name     | glance                           |
| username | glance                           |
+----------+----------------------------------+
```

Replace *PASSWORD* with a secure password that will be used by the Image Service when authenticating with the Identity service.

3. Link the **glance** user and the **admin** role together within the context of the **services** tenant:

```
[(keystone_admin)]# openstack role add --project services --user
glance admin
```

4. Create the **glance** Image service entry:

```
[(keystone_admin)]# openstack service create --name glance \
        --description "Glance Image Service" \
        image
```

5. Create the **glance** endpoint entry:

```
[(keystone_admin)]# openstack endpoint create \
        --publicurl 'http://IP:9292' \
        --adminurl 'http://IP:9292' \
        --internalurl 'http://IP:9292' \
        --region RegionOne \
        glance
```

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

5.4.3. Configure Image Service Authentication

Configure the Image service to use the Identity service for authentication. All steps in this procedure must be performed on each node hosting the Image service, while logged in as the **root** user.

Procedure 5.4. Configuring the Image Service to Authenticate through the Identity Service

1. Configure the **glance-api** service:

```
# openstack-config --set /etc/glance/glance-api.conf \
  paste_deploy flavor keystone
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken auth_host IP
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken auth_port 35357
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken auth_protocol http
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_tenant_name services
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_user glance
# openstack-config --set /etc/glance/glance-api.conf \
  keystone_authtoken admin_password PASSWORD
```

2. Configure the **glance-registry** service:

```
# openstack-config --set /etc/glance/glance-registry.conf \
  paste_deploy flavor keystone
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken auth_host IP
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken auth_port 35357
```

```
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken auth_protocol http
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_tenant_name services
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_user glance
# openstack-config --set /etc/glance/glance-registry.conf \
  keystone_authtoken admin_password PASSWORD
```

Replace the following values:

- Replace *IP* with the IP address or host name of the Identity server.
- Replace *services* with the name of the tenant that was created for the use of the Image service (previous examples set this to **services**).
- Replace *glance* with the name of the service user that was created for the Image service (previous examples set this to **glance**).
- Replace *PASSWORD* with the password associated with the service user.

5.4.4. Use the Object Storage Service for Image Storage

By default, the Image service uses the local file system (**file**) for its storage back end; however, either of the following storage back ends can be used to store uploaded disk images:

- **file** - Local file system of the Image server (**/var/lib/glance/images/** directory)
- **swift** - OpenStack Object Storage service

Note

The configuration procedure below uses the **openstack-config** command; however, you can also manually update the **/etc/glance/glance-api.conf** file. If manually updating the file, ensure that the **default_store** parameter is set to the correct back end (for example, '**default_store=rbd**'), and update the parameters in that back end's section (for example, under '**RBD Store Options**').

Procedure 5.5. Configuring the Image Service to use the Object Storage Service

1. Set the **default_store** configuration key to **swift**:

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT default_store swift
```

2. Set the **swift_store_auth_address** configuration key to the public endpoint for the Identity service:

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT swift_store_auth_address http://IP:5000/v2.0/
```

3. Add the container for storing images in the Object Storage service:

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT swift_store_create_container_on_put True
```

4. Set the **swift_store_user** configuration key, in the format *TENANT*:*USER*, to contain the tenant and user to use for authentication:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT swift_store_user services:swift
   ```

   - If you followed the instructions in this guide to deploy Object Storage, replace these values with the **services** tenant and the **swift** user respectively (as shown in the command example above).
   - If you did not follow the instructions in this guide to deploy Object Storage, replace these values with the appropriate Object Storage tenant and user for your environment.

5. Set the **swift_store_key** configuration key to the password that was set for the **swift** user when deploying the Object Storage service:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT swift_store_key PASSWORD
   ```

5.4.5. Configure the Firewall to Allow Image Service Traffic

The Image service must be accessible over the network through port **9292**. All steps in this procedure must be performed on the server hosting the Image service, while logged in as the **root** user.

Procedure 5.6. Configuring the Firewall to Allow Image Service Traffic

1. Open the **/etc/glance/glance-api.conf** file in a text editor, and remove any comment characters preceding the following parameters:

   ```
   bind_host = 0.0.0.0
   bind_port = 9292
   ```

2. Open the **/etc/sysconfig/iptables** file in a text editor.

3. Add an INPUT rule allowing TCP traffic on port **9292**. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 9292 -j ACCEPT
   ```

4. Save the changes to the **/etc/sysconfig/iptables** file.

5. Restart the **iptables** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

5.4.6. Configure RabbitMQ Message Broker Settings for the Image Service

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 server hosting the Image service, while logged in as the **`root`** user.

Procedure 5.7. Configuring the Image Service (glance) to Use the RabbitMQ Message Broker

1. Set RabbitMQ as the notifier:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT notification_driver messaging
   ```

2. Set the name of the RabbitMQ host:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT rabbit_host RABBITMQ_HOST
   ```

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

3. Set the message broker port to **`5672`**:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT rabbit_port 5672
   ```

4. Set the RabbitMQ user name and password created for the Image service when RabbitMQ was configured:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT rabbit_userid glance
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT rabbit_password GLANCE_PASS
   ```

   Replace **`glance`** and *GLANCE_PASS* with the RabbitMQ user name and password created for the Image service.

5. When RabbitMQ was launched, the **`glance`** user was granted read and write permissions to all resources: specifically, through the virtual host `/`. Configure the Image service to connect to this virtual host:

   ```
   # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT rabbit_virtual_host /
   ```

5.4.7. Configure the Image Service to Use SSL

Use the following options in the **`glance-api.conf`** file to configure SSL.

Table 5.1. SSL Options for the Image Service

Configuration Option	Description
`cert_file`	The path to the certificate file to use when starting the API server securely.
`key_file`	The path to the private key file to use when starting the API server securely.
`ca_file`	The path to the CA certificate file to use to verify connecting clients.

5.4.8. Populate the Image Service Database

Populate the Image service database after you have successfully configured the Image service database connection string.

Procedure 5.8. Populating the Image Service Database

1. Log in to the system hosting the Image service.

2. Switch to the **glance** user:

```
#  su glance -s /bin/sh
```

3. Initialize and populate the database identified in **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf**:

```
$  glance-manage db_sync
```

5.4.9. Enable Image Loading Through the Local File System

By default, the Image service provides images to instances using the HTTP protocol. Specifically, image data is transmitted from the image store to the local disk of the Compute node using HTTP. This process is typical for most deployments where the Image and Compute services are installed on different hosts.

Note

You can use direct image access if the Image service and the Compute service are not installed on the same host, but are sharing a shared file system. In this case, the file system must be mounted in the same location.

In deployments where both services are installed on the same host (and, consequently, share the same file system), it is more efficient to skip the HTTP steps altogether. Instead, you must configure both the Image service and the Compute service to send and receive images using the local file system.

The Image file system metadata generated for this procedure will only apply to new images. Any existing images will not use this metadata.

Procedure 5.9. Configuring Image and Compute Services to Send and Receive Images through the Local File System

1. Create a JSON document that exposes the Image file system metadata required by **openstack-nova-compute**.

2. Configure the Image service to use the JSON document.

3. Configure **openstack-nova-compute** to use the file system metadata provided by the Image service.

5.4.9.1. Configure the Image Service to Provide Images Through the Local File System

To enable image loading through the local file system (as opposed to HTTP), the Image service must first expose its local file-system metadata to the **openstack-nova-compute** service.

Procedure 5.10. Configuring the Image Service to Expose Local File System Metadata to the Compute Service

1. Determine the mount point of the file system used by the Image service:

   ```
    # df
   Filesystem     1K-blocks     Used Available Use% Mounted on
   /dev/sda3       51475068 10905752  37947876  23% /
   devtmpfs         2005504        0   2005504   0% /dev
   tmpfs            2013248      668   2012580   1% /dev/shm
   ```

   For example, if the Image service uses the **/dev/sda3** file system, its corresponding mount point is **/**.

2. Create a unique ID for the mount point:

   ```
    # uuidgen
   ad5517ae-533b-409f-b472-d82f91f41773
   ```

   Note the output of the **uuidgen**, as this will be used in the next step.

3. Create a file with the **.json** extension.

4. Open the file in a text editor, and add the following information:

   ```
   {
   "id": "UID",
   "mountpoint": "MOUNTPT"
   }
   ```

   Replace the following values:

   - Replace *UID* with the unique ID created in the previous step.
   - Replace *MOUNTPT* with the mount point of the Image service's file system, as determined in the first step.

5. Configure the Image service to use this JSON file:

   ```
    # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT show_multiple_locations True
    # openstack-config --set /etc/glance/glance-api.conf \
     DEFAULT filesystem_store_metadata_file JSON_PATH
   ```

   Replace *JSON_PATH* with the full path to the JSON file.

   Important

   If configured without the proper policy settings, a non-admin user of the Image Service can replace active image data (that is, switch out a current image without other users knowing). See the OSSN announcement (recommended actions) for configuration information: https://wiki.openstack.org/wiki/OSSN/OSSN-0065

6. Restart the Image service (if it is already running):

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

The Image file-system metadata generated for this procedure only applies to new images. Any existing images will not use this metadata.

5.4.9.2. Configure the Compute Service to Use Local File System Metadata

After configuring the Image service to expose local file-system metadata, configure the Compute service to use this metadata. This allows **`openstack-nova-compute`** to load images from the local file system.

Procedure 5.11. Configuring the Compute Service to use File System Metadata Provided by the Image Service

1. Configure **`openstack-nova-compute`** to enable the use of direct URLs that have the **`file://`** scheme:

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

2. Create an entry for the Image service's file system:

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

   Replace *FSENTRY* with a name to assign to the Image service's file system.

3. Open the **`.json`** file used by the Image service to expose its local file-system metadata. The information in this file will be used in the next step.

4. Associate the entry for Image service's file system to the file system metadata exposed by the Image service:

   ```
   # openstack-config --set /etc/nova/nova.conf \
    image_file_url:FSENTRY id UID
   # openstack-config --set /etc/nova/nova.conf \
    image_file_url:FSENTRY mountpoint MOUNTPT
   ```

   Replace the following values:

   - Replace *UID* with the unique ID used by the Image service. In the **`.json`** file used by the Image service, the *UID* is the **`"id"`** value.
   - Replace *MOUNTPT* with the mount point used by the Image service's file system. In the **`.json`** file used by the Image service, the *MOUNTPT* is the **`"mountpoint"`** value.

5.5. Launch the Image API and Registry Services

After Glance has been configured, start the **`glance-api`** and **`glance-registry`** services, and configure each service to start at boot time:

```
# systemctl start openstack-glance-registry.service
# systemctl start openstack-glance-api.service
# systemctl enable openstack-glance-registry.service
# systemctl enable openstack-glance-api.service
```

5.6. Validate the Image Service Installation

This section outlines the steps required to upload a disk image to the Image service. This image can be used as a basis for launching virtual machines in your OpenStack environment.

5.6.1. Obtain a Test Disk Image

Download from Red Hat a disk image that can be used to test the import of images into the Image service. A new image is provided with each minor Red Hat Enterprise Linux 7 release, and is available on the **Product Downloads** page for Red Hat Enterprise Linux.

Procedure 5.12. Downloading a Test Disk Image

1. Go to https://access.redhat.com, and log in to the Red Hat Customer Portal using your customer account details.
2. Click **Downloads** in the menu bar.
3. Click **`A-Z`** to sort the product downloads alphabetically.
4. Click **`Red Hat Enterprise Linux`** to access the **Product Downloads** page.
5. Click the **`KVM Guest Image`** download link.

5.6.2. Upload a Disk Image

To launch instances based on images stored in the Image service, you must first upload one or more images into the Image service. You must have access to images suitable for use in the OpenStack environment.

Important

It is recommended that you run the **`virt-sysprep`** command on all Linux-based virtual machine images prior to uploading them to the Image service. The **`virt-sysprep`** command reinitializes a disk image in preparation for use in a virtual environment. Default operations include the removal of SSH keys, removal of persistent MAC addresses, and removal of user accounts.

The **`virt-sysprep`** command is provided by the Red Hat Enterprise Linux *libguestfs-tools* package. Install the package, and reinitialize the disk image:

```
# yum install -y libguestfs-tools
# virt-sysprep --add FILE
```

For information on enabling and disabling specific operations, see the **`virt-sysprep`** manual page.

Procedure 5.13. Uploading a Disk Image to the Image Service

1. Set up the shell to access keystone as a configured user (an administrative account is not required):

   ```
   #  source ~/keystonerc_userName
   ```

2. Import the disk image:

   ```
   [(keystone_userName)]#  glance image-create --name "NAME" \
           --is-public IS_PUBLIC \
           --disk-format DISK_FORMAT \
           --container-format CONTAINER_FORMAT \
           --file IMAGE
   ```

   Replace the following values:

   - Replace *NAME* with a name by which users will refer to the disk image.
   - Replace *IS_PUBLIC* with either **`true`** or **`false`**:
     - **`true`** - All users are able to view and use the image.
     - **`false`** - Only administrators are able to view and use the image.
   - Replace *DISK_FORMAT* with the disk image's format. Valid values include: **`aki`**, **`ami`**, **`ari`**, **`iso`**, **`qcow2`**, **`raw`**, **`vdi`**, **`vhd`**, and **`vmdk`**. If the format of the virtual machine disk image is unknown, use the **`qemu-img info`** command to try and identify it.
   - Replace *CONTAINER_FORMAT* with the container format of the image. The container format is **`bare`** unless the image is packaged in a file format, such as **`ovf`** or **`ami`**, that includes additional metadata related to the image.
   - Replace *IMAGE* with the local path to the image file (for uploading). If the image being uploaded is not locally accessible but is available using a remote URL, provide the URL using the **`--location`** parameter instead of the **`--file`** parameter. Note that you must also specify the **`--copy-from`** argument to copy the image into the object store, otherwise the image will be accessed remotely each time it is required.

   For more information about the **`glance image-create`** syntax, see the help page:

   ```
   [(keystone_userName)]#  glance help image-create
   ```

   Note the unique identifier for the image in the output of the command above.

3. Verify that your image was successfully uploaded:

   ```
    [(keystone_userName)]#  glance image-show IMAGE_ID
   +------------------+--------------------------------------+
   | Property         | Value                                |
   +------------------+--------------------------------------+
   | checksum         | 2f81976cae15c16ef0010c51e3a6c163     |
   | container_format | bare                                 |
   | created_at       | 2013-01-25T14:45:48                  |
   | deleted          | False                                |
   | disk_format      | qcow2                                |
   | id               | 0ce782c6-0d3e-41df-8fd5-39cd80b31cd9 |
   | is_public        | True                                 |
   | min_disk         | 0                                    |
   ```

```
| min_ram          | 0                                   |
| name             | RHEL 6.6                            |
| owner            | b1414433c021436f97e9e1e4c214a710    |
| protected        | False                               |
| size             | 25165824                            |
| status           | active                              |
| updated_at       | 2013-01-25T14:45:50                 |
+------------------+-------------------------------------+
```

Replace *IMAGE_ID* with the unique identifier for the image.

The disk image can now be used as the basis for launching virtual machine instances in your OpenStack environment.

Chapter 6. Install the Block Storage Service

6.1. Install the Block Storage Service Packages

The OpenStack Block Storage service requires the following packages:

openstack-cinder

Provides the Block Storage services and associated configuration files.

openstack-utils

Provides supporting utilities to assist with a number of tasks including the editing of configuration files.

openstack-selinux

Provides OpenStack specific SELinux policy modules.

device-mapper-multipath

Provides tools to manage multipath devices using device-mapper. These tools are necessary for proper block storage operations.

Install the packages:

```
# yum install -y openstack-cinder openstack-utils openstack-selinux device-mapper-multipath
```

6.2. Create the Block Storage Service Database

Create the database and database user used by the Block Storage services. All steps must be performed on the database server, while logged in as the **root** user.

Procedure 6.1. Creating the Block Storage Services Database

1. Connect to the database service:

   ```
   # mysql -u root -p
   ```

2. Create the **cinder** database:

   ```
   mysql> CREATE DATABASE cinder;
   ```

3. Create a **cinder** database user and grant the user access to the**cinder** database:

   ```
   mysql> GRANT ALL ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'PASSWORD';
   mysql> GRANT ALL ON cinder.* TO 'cinder'@'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
   ```

6.3. Configure the Block Storage Service

6.3.1. Configure the Block Storage Service Database Connection

The database connection string used by the Block Storage services is defined in the **/etc/cinder/cinder.conf** file. It must be updated to point to a valid database server before starting the service.

Set the value of the **sql_connection** configuration key on each system hosting Block Storage services:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace the following values:

- Replace *USER* with the Block Storage service database user name, usually **cinder**.
- Replace *PASS* with the password of the database user.
- Replace *IP* with the IP address or host name of the server hosting the database service.
- Replace *DB* with the name of the Block Storage service database, usually **cinder**.

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Block Storage service database user was granted access when creating the Block Storage service database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the Block Storage service database, you must enter 'localhost'.

6.3.2. Create the Block Storage Service Identity Records

Create and configure Identity service records required by the Block Storage service. These entries provide authentication for the Block Storage services, and guide other OpenStack services attempting to locate and access the volume functionality provided by Block Storage.

This procedure assumes that you have already created an administrative user account and a **services** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **keystonerc_admin** file and on which the **keystone** command-line utility is installed.

Procedure 6.2. Creating Identity Records for the Block Storage Service

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

```
 #  source ~/keystonerc_admin
```

2. Create the **cinder** user:

```
 [(keystone_admin)]# openstack user create --password PASSWORD cinder
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | cb4dafc849df4ea180e9e29346a29038 |
| name     | cinder                           |
| username | cinder                           |
+----------+----------------------------------+
```

   Replace *PASSWORD* with a secure password that will be used by the Block Storage service when authenticating with the Identity service.

3. Link the **cinder** user and the **admin** role together within the context of the **services** tenant:

```
 [(keystone_admin)]# openstack role add --project services --user
cinder admin
```

4. Create the **cinder** and **cinderv2** Block Storage service entries:

```
 [(keystone_admin)]# openstack service create --name cinder \
        --description "Cinder Volume Service" \
        volume
```

5. Create the **cinder** endpoint entry:

```
 [(keystone_admin)]# openstack endpoint create \
   --publicurl 'http://IP:8776/v1/%(tenant_id)s' \
   --adminurl 'http://IP:8776/v1/%(tenant_id)s' \
   --internalurl 'http://IP:8776/v1/%(tenant_id)s'\
   --region RegionOne \
   cinder
```

   Replace *IP* with the IP address or host name of the server hosting the Block Storage API service (**openstack-cinder-api**). To install and run multiple instances of the API service, repeat this step for the IP address or host name of each instance.

6.3.3. Configure Block Storage Service Authentication

Configure the Block Storage service to use the Identity service for authentication. All steps in this procedure must be performed on each server hosting Block Storage services, while logged in as the **root** user.

Procedure 6.3. Configuring the Block Storage Service to Authenticate Through the Identity Service

1. Set the authentication strategy to **`keystone`**:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT auth_strategy keystone
   ```

2. Set the Identity service host that the Block Storage services must use:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     keystone_authtoken auth_host IP
   ```

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

3. Set the Block Storage services to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     keystone_authtoken admin_tenant_name services
   ```

   Replace *services* with the name of the tenant created for the use of OpenStack Networking. Examples in this guide use **`services`**.

4. Set the Block Storage services to authenticate using the **`cinder`** administrative user account:

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

5. Set the Block Storage services to use the correct **`cinder`** administrative user account password:

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

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

6.3.4. Configure the Firewall to Allow Block Storage Service Traffic

Each component in the OpenStack environment uses the Identity service for authentication and must be able to access the service. The firewall on the system hosting the Block Storage service must be altered to allow network traffic on the required ports. All steps in this procedure must be run on each server hosting Block Storage services, while logged in as the **`root`** user.

Procedure 6.4. Configuring the Firewall to Allow Block Storage Service Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.

2. Add an INPUT rule allowing TCP traffic on ports **`3260`** and **`8776`** to the file. The new rule must appea before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 3260,8776 -j ACCEPT
   ```

3. Save the changes to the **`/etc/sysconfig/iptables`** file.

4. Restart the **`iptables`** service to ensure that the change takes effect:

```
#  systemctl restart iptables.service
```

6.3.5. Configure the Block Storage Service to Use SSL

Use the following options in the **cinder.conf** file to configure SSL.

Table 6.1. SSL options for Block Storage

Configuration Option	Description
backlog	The number of backlog requests with which to configure the socket.
tcp_keepidle	Sets the value of TCP_KEEPIDLE in seconds for each server socket.
ssl_ca_file	The CA certificate file to use to verify connecting clients.
ssl_cert_file	The certificate file to use when starting the server securely.
ssl_key_file	The private key file to use when starting the server securely.

6.3.6. Configure RabbitMQ Message Broker Settings for the Block Storage Service

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 server hosting the Block Storage service, while logged in as the **root** user.

Procedure 6.5. Configuring the Block Storage Service to use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

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

2. Set the name of the RabbitMQ host:

```
#  openstack-config --set /etc/cinder/cinder.conf \
   DEFAULT rabbit_host RABBITMQ_HOST
```

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

3. Set the message broker port to **5672**:

```
#  openstack-config --set /etc/cinder/cinder.conf \
   DEFAULT rabbit_port 5672
```

4. Set the RabbitMQ username and password created for the Block Storage service when RabbitMQ was configured:

```
#  openstack-config --set /etc/cinder/cinder.conf \
   DEFAULT rabbit_userid cinder
```

```
#  openstack-config --set /etc/cinder/cinder.conf \
   DEFAULT rabbit_password CINDER_PASS
```

Replace **cinder** and *CINDER_PASS* with the RabbitMQ user name and password created for the Block Storage service.

5. When RabbitMQ was launched, the **cinder** user was granted read and write permissions to all resources: specifically, through the virtual host **/**. Configure the Block Storage service to connect to this virtual host:

```
# openstack-config --set /etc/cinder/cinder.conf \
   DEFAULT rabbit_virtual_host /
```

6.3.7. Enable SSL Communication Between the Block Storage Service and the Message Broker

If you enabled SSL on the message broker, you must configure the Block Storage service accordingly. This procedure requires the exported client certificates and key file. See Section 2.3.5, "Export an SSL Certificate for Clients" for instructions on how to export these files.

1. Enable SSL communication with the message broker:

```
# openstack-config --set /etc/cinder/cinder.conf \
 DEFAULT rabbit_use_ssl True
# openstack-config --set /etc/cinder/cinder.conf \
 DEFAULT kombu_ssl_certfile /path/to/client.crt
# openstack-config --set /etc/cinder/cinder.conf \
 DEFAULT kombu_ssl_keyfile /path/to/clientkeyfile.key
```

Replace the following values:

- Replace */path/to/client.crt* with the absolute path to the exported client certificate.
- Replace */path/to/clientkeyfile.key* with the absolute path to the exported client key file.

2. If your certificates were signed by a third-party Certificate Authority (CA), you must also run the following command:

```
# openstack-config --set /etc/cinder/cinder.conf \
 DEFAULT kombu_ssl_ca_certs /path/to/ca.crt
```

Replace */path/to/ca.crt* with the absolute path to the CA file provided by the third-party CA (see Section 2.3.4, "Enable SSL on the RabbitMQ Message Broker" for more information).

6.3.8. Populate the Block Storage Database

Populate the Block Storage database after you have successfully configured the Block Storage service database connection string.

Important

This procedure must be followed only once to initialize and populate the database. You do not need to perform these steps again when adding additional systems hosting Block Storage services.

Procedure 6.6. Populating the Block Storage Service Database

1. Log in to the system hosting one of the Block Storage services.
2. Switch to the **cinder** user:

   ```
   # su cinder -s /bin/sh
   ```

3. Initialize and populate the database identified in **/etc/cinder/cinder.conf**:

   ```
   $ cinder-manage db sync
   ```

6.3.9. Increase the Throughput of the Block Storage API Service

By default, the Block Storage API service (**openstack-cinder-api**) runs in one process. This limits the number of API requests that the Block Storage service can process at any given time. In a production environment, you should increase the Block Storage API throughput by allowing **openstack-cinder-api** to run in as many processes as the machine capacity allows.

The Block Storage API service option, ***osapi_volume_workers***, allows you to specify the number of API service workers (or OS processes) to launch for **openstack-cinder-api**.

To set this option, run the following command on the **openstack-cinder-api** host:

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT osapi_volume_workers CORES
```

Replace *CORES* with the number of CPU cores/threads on a machine.

6.4. Configure the Volume Service

6.4.1. Block Storage Driver Support

The volume service (**openstack-cinder-volume**) requires access to suitable block storage. Red Hat OpenStack Platform provides volume drivers for the following supported block storage types:

- Red Hat Ceph
- LVM/iSCSI
- ThinLVM
- NFS
- NetApp
- Dell EqualLogic
- Dell Storage Center

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

For instructions on how to set up an LVM back end, refer to Section 6.4.2, “Configure OpenStack Block Storage to Use an LVM Storage Back End”.

6.4.2. Configure OpenStack Block Storage to Use an LVM Storage Back End

The **openstack-cinder-volume** service can make use of a volume group attached directly to the server on which the service runs. This volume group must be created exclusively for use by the Block Storage service, and the configuration updated to point to the name of the volume group.

All steps in the following procedure must be performed on the server hosting the **openstack-cinder-volume** service, while logged in as the **root** user.

Procedure 6.7. Configuring openstack-cinder-volume to Use LVM Storage as a Back End

1. Create a physical volume:

   ```
   # pvcreate DEVICE
     Physical volume "DEVICE" successfully created
   ```

   Replace *DEVICE* with the path to a valid, unused, device. For example:

   ```
   # pvcreate /dev/sdX
   ```

2. Create a volume group:

   ```
   # vgcreate cinder-volumes DEVICE
     Volume group "cinder-volumes" successfully created
   ```

   Replace *DEVICE* with the path to the device used when creating the physical volume. Optionally replace *cinder-volumes* with an alternative name for the new volume group.

3. Set the **volume_group** configuration key to the name of the volume group created in the previous step:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT volume_group cinder-volumes
   ```

4. Ensure that the correct volume driver for accessing LVM storage is in use by setting the **volume_driver** configuration key to **cinder.volume.drivers.lvm.LVMVolumeDriver**:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT volume_driver cinder.volume.drivers.lvm.LVMVolumeDriver
   ```

6.4.3. Configure the SCSI Target Daemon

The **openstack-cinder-volume** service uses a SCSI target daemon for mounting storage. You must install a SCSI target daemon on each server hosting an instance of the **openstack-cinder-volume** service, while logged in as the **root** user.

Procedure 6.8. Configure a SCSI Target Daemon

1. Install the *targetcli* package:

   ```
   # yum install targetcli
   ```

2. Launch the **target** daemon and configure it to start at boot time:

```
# systemctl start target.service
# systemctl enable target.service
```

3. Configure the volume service to use the **`lioadm`** iSCSI target user-land tool:

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT iscsi_helper lioadm
   ```

4. Set the IP address on which the iSCSI daemon must listen (*ISCSIIP*):

   ```
   # openstack-config --set /etc/cinder/cinder.conf \
     DEFAULT iscsi_ip_address ISCSIIP
   ```

   Replace *ISCSI_IP* with the IP address or host name of the server hosting the **`openstack-cinder-volume`** service.

6.5. Launch the Block Storage Services

To enable the Block Storage functionality at least one instance of each of the three services must be started:

- The API service (**`openstack-cinder-api`**).
- The scheduler service (**`openstack-cinder-scheduler`**).
- The volume service (**`openstack-cinder-volume`**).

The services do not need to be located on the same system, but must be configured to communicate using the same message broker and database instance. When the services are running, the API accepts incoming volume requests, the scheduler assigns them as appropriate, and the volume service actions them.

Procedure 6.9. Launching Block Storage Services

1. Log in as the **`root`** user on each server where you intend to run the API, start the API service, and configure it to start at boot time:

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

2. Log in as the **`root`** user on each server where you intend to run the scheduler, start the scheduler service, and configure it to start at boot time:

   ```
   # systemctl start openstack-cinder-scheduler.service
   # systemctl enable openstack-cinder-scheduler.service
   ```

3. Log in as the **`root`** user on each server to which Block Storage has been attached, start the volume service, and configure it to start at boot time:

   ```
   # systemctl start openstack-cinder-volume.service
   # systemctl enable openstack-cinder-volume.service
   ```

6.6. Validate the Block Storage Service Installation

6.6.1. Validate the Block Storage Service Installation Locally

Create and remove a volume locally to validate that the block storage installation is complete and ready for use. Perform this test on the server hosting the Block Storage API service, while logged in as the **`root`** user or a user with access to the **`keystonerc_admin`** file. Copy the **`keystonerc_admin`** file to the system before proceeding.

Procedure 6.10. Validating the Block Storage Service Installation Locally

1. Populate the environment variables used for identifying and authenticating the administrative user:

   ```
   # source ~/keystonerc_admin
   ```

2. Verify that no errors are returned in the output of this command:

   ```
   # cinder list
   ```

3. Create a volume:

   ```
   # cinder create SIZE
   ```

   Replace *SIZE* with the size of the volume to create in Gigabytes (GB).

4. Remove the volume:

   ```
   # cinder delete ID
   ```

   Replace *ID* with the identifier returned when the volume was created.

6.6.2. Validate the Block Storage Service Installation Remotely

Create and remove a volume using a remote machine to validate that the block storage installation is complete and ready for use. Perform this test on a server other than the server hosting the Block Storage API service, while logged in as the **`root`** user or a user with access to the **`keystonerc_admin`** file. Copy the **`keystonerc_admin`** file to the system before proceeding.

Procedure 6.11. Validating the Block Storage Service Installation Remotely

1. Install the *python-cinderclient* package:

   ```
   # yum install -y python-cinderclient
   ```

2. Populate the environment variables used for identifying and authenticating the administrative user:

   ```
   # source ~/keystonerc_admin
   ```

3. Verify that no errors are returned in the output of this command:

   ```
   # cinder list
   ```

4. Create a volume:

   ```
   # cinder create SIZE
   ```

Replace *SIZE* with the size of the volume to create in gigabytes (GB).

5. Remove the volume:

   ```
   # cinder delete ID
   ```

   Replace *ID* with the identifier returned when the volume was created.

Chapter 7. Install OpenStack Networking

7.1. Install the OpenStack Networking Packages

OpenStack Networking requires the following packages:

openstack-neutron

Provides OpenStack Networking and associated configuration files.

openstack-neutron-ml2>

Provides an OpenStack Networking plug-in.

openstack-utils

Provides supporting utilities to assist with a number of tasks, including the editing of configuration files.

openstack-selinux

Provides OpenStack-specific SELinux policy modules.

The packages must be installed on all systems that will handle network traffic. This includes the OpenStack Networking node, all network nodes, and all Compute nodes.

Install the packages:

```
# yum install -y openstack-neutron \
  openstack-neutron-ml2 \
  openstack-utils \
  openstack-selinux
```

Replace *PLUGIN* with **`ml2`**,**`openvswitch`**, or **`linuxbridge`** to determine which plug-in is installed.

7.2. Configure OpenStack Networking

Important

In an unmodified Red Hat Openstack Platform installation, the **`vif_plugging_is_fatal`** option is commented out in the **`[DEFAULT]`** section of the **`/etc/nova/nova.conf`** file, and defaults to **`True`**. This option controls whether instances should fail to boot if VIF plugging fails. Similarly, the **`notify_nova_on_port_status_changes`** and **`notify_nova_on_port_data_changes`** options are commented out in the **`[DEFAULT]`** section of the **`/etc/neutron/neutron.conf`** file, and default to **`False`**. These options control whether notifications should be sent to nova on port status or data changes. However, this combination of values can prevent instances from booting. To allow instances to boot correctly, set all of these options to either **`True`** or **`False`**. To set **`True`**, run the following commands:

```
 # openstack-config --set /etc/nova/nova.conf \
DEFAULT vif_plugging_is_fatal True
 # openstack-config --set /etc/neutron/neutron.conf \
DEFAULT notify_nova_on_port_status_changes True
 # openstack-config --set /etc/neutron/neutron.conf \
DEFAULT notify_nova_on_port_data_changes True
```

To set **`False`**, run the following commands instead:

```
 # openstack-config --set /etc/nova/nova.conf \
DEFAULT vif_plugging_is_fatal False
 # openstack-config --set /etc/neutron/neutron.conf \
DEFAULT notify_nova_on_port_status_changes False
 # openstack-config --set /etc/neutron/neutron.conf \
DEFAULT notify_nova_on_port_data_changes False
```

7.2.1. Set the OpenStack Networking Plug-in

OpenStack Networking plug-ins can be referenced in **`neutron.conf`** by their nominated short names, instead of their lengthy class names. For example:

```
core_plugin = neutron.plugins.ml2.plugin:Ml2Plugin
```

will become:

```
core_plugin = ml2
```

Take care not to introduce errant whitespace characters, as these could result in parse errors.

The **`service_plugins`** option accepts a comma-delimited list of multiple service plugins.

Table 7.1. service_plugins

Short name	Class name
dummy	neutron.tests.unit.dummy_plugin:DummyServicePlugin
router	neutron.services.l3_router.l3_router_plugin:L3RouterPlugin
firewall	neutron.services.firewall.fwaas_plugin:FirewallPlugin
lbaas	neutron.services.loadbalancer.plugin:LoadBalancerPlugin

Short name	Class name
metering	neutron.services.metering.metering_plugin:MeteringPlugin

7.2.1.1. Enable the ML2 Plug-in

Enable the ML2 plug-in on the node running the **neutron-server** service.

Procedure 7.1. Enabling the ML2 Plug-in

1. Create a symbolic link to direct OpenStack Networking to the **ml2_conf.ini** file:

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

2. Set the tenant network type. Supported values are **gre**, **local**, **vlan**, and **vxlan**. The default value is **local**, but this is not recommended for enterprise deployments:

```
# openstack-config --set /etc/neutron/plugin.ini \
  ml2 tenant_network_types TYPE
```

   Replace *TYPE* with the tenant network type.

3. If you chose **flat** or **vlan** networking, you must also map physical networks to VLAN ranges:

```
# openstack-config --set /etc/neutron/plugin.ini \
  ml2 network_vlan_ranges NAME:START:END
```

   Replace the following values:

   - Replace *NAME* with the name of the physical network.
   - Replace *START* with the VLAN identifier that starts the range.
   - Replace *END* with the VLAN identifier that ends the range.

   Multiple ranges can be specified using a comma-delimited list, for example:

```
physnet1:1000:2999,physnet2:3000:3999
```

4. Set the driver types. Supported values are **local**, **flat**, **vlan**, **gre**, and **vxlan**:

```
# openstack-config --set /etc/neutron/plugin.ini \
  ml2 type_drivers TYPE
```

   Replace *TYPE* with the driver type. Specify multiple drivers using a comma-delimited list.

5. Set the mechanism drivers. Available values are **openvswitch**, **linuxbridge**, and **l2population**:

```
# openstack-config --set /etc/neutron/plugin.ini \
  ml2 mechanism_drivers TYPE
```

Replace *TYPE* with the mechanism driver type. Specify multiple mechanism drivers using a comma-delimited list.

6. Enable L2 population:

```
# openstack-config --set /etc/neutron/plugin.ini \
  agent l2_population True
```

7. Set the firewall driver in the **`/etc/neutron/plugins/ml2/openvswitch_agent.ini`** file or the **`/etc/neutron/plugins/ml2/linuxbridge_agent.ini`** file, depending on which plug-in agent you are using:

   a. **Open vSwitch Firewall Driver**

   ```
    # openstack-config --set
   /etc/neutron/plugins/ml2/openvswitch_agent.ini
      securitygroup firewall_driver
   neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallD
   river
   ```

   b. **Linux Bridge Firewall Driver**

   ```
    # openstack-config --set
   /etc/neutron/plugins/ml2/linuxbridge_agent.ini
      securitygroup firewall_driver
   neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
   ```

8. Enable the ML2 plug-in and the L3 router:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT core_plugin ml2
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT service_plugins router
```

7.2.2. Create the OpenStack Networking Database

Create the database and database user used by OpenStack Networking. All steps in this procedure must be performed on the database server, while logged in as the **`root`** user, and prior to starting the **`neutron-server`** service.

Procedure 7.2. Creating the OpenStack Networking Database

1. Connect to the database service:

```
# mysql -u root -p
```

2. Create the database with one of the following names:

   - If you are using the ML2 plug-in, the recommended database name is **`neutron_ml2`**
   - If you are using the Open vSwitch plug-in, the recommended database name is **`ovs_neutron`**.

- If you are using the Linux Bridge plug-in, the recommended database name is **`neutron_linux_bridge`**.

This example creates the ML2 **`neutron_ml2`** database:

```
mysql> CREATE DATABASE neutron_ml2 character set utf8;
```

3. Create a **`neutron`** database user and grant the user access to the**`neutron_ml2`** database:

```
mysql> GRANT ALL ON neutron_ml2.* TO 'neutron'@'%' IDENTIFIED BY
'PASSWORD';
mysql> GRANT ALL ON neutron_ml2.* TO 'neutron'@'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
```

7.2.3. Configure the OpenStack Networking Database Connection

The database connection string used by OpenStack Networking is defined in the **`/etc/neutron/plugin.ini`** file. It must be updated to point to a valid database server before starting the service. All steps in this procedure must be performed on the server hosting OpenStack Networking, while logged in as the **`root`** user.

Procedure 7.3. Configuring the OpenStack Networking SQL Database Connection

1. Set the value of the **`connection`** configuration key.

```
# openstack-config --set /etc/neutron/plugin.ini \
  DATABASE sql_connection mysql://USER:PASS@IP/DB
```

Replace the following values:

- Replace *USER* with the OpenStack Networking database user name, usually **`neutron`**.
- Replace *PASS* with the password of the database user.
- Replace *IP* with the IP address or host name of the database server.
- Replace *DB* with the name of the OpenStack Networking database.

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the OpenStack Networking database user was granted access when creating the OpenStack Networking database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the database, you must enter 'localhost'.

2. Upgrade the OpenStack Networking database schema:

```
 #  neutron-db-manage --config-file /usr/share/neutron/neutron-
dist.conf \
   --config-file /etc/neutron/neutron.conf --config-file
/etc/neutron/plugin.ini upgrade head
```

7.2.4. Create the OpenStack Networking Identity Records

Create and configure Identity service records required by OpenStack Networking. These entries assist other OpenStack services attempting to locate and access the functionality provided by OpenStack Networking.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 7.4. Creating Identity Records for OpenStack Networking

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

```
 #  source ~/keystonerc_admin
```

2. Create the **`neutron`** user:

```
 [(keystone_admin)]#  openstack user create --password PASSWORD neutron
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 8f0d819a4ae54bf9b12d01d0fb095805 |
| name     | neutron                          |
| username | neutron                          |
+----------+----------------------------------+
```

Replace *PASSWORD* with a secure password that will be used by OpenStack Networking when authenticating with the Identity service.

3. Link the **`neutron`** user and the **`admin`** role together within the context of the **`services`** tenant:

```
[(keystone_admin)]# openstack role add --project services --user
neutron admin
```

4. Create the **neutron** OpenStack Networking service entry:

```
[(keystone_admin)]# openstack service create --name neutron \
   --description "OpenStack Networking" \
   network
```

5. Create the **neutron** endpoint entry:

```
[(keystone_admin)]# openstack endpoint create
   --publicurl 'http://IP:9696' \
   --adminurl 'http://IP:9696' \
   --internalurl 'http://IP:9696' \
   --region RegionOne \
   neutron
```

   Replace *IP* with the IP address or host name of the server that will act as the OpenStack Networking node.

7.2.5. Configure OpenStack Networking Authentication

Configure OpenStack Networking to use the Identity service for authentication. All steps in this procedure must be performed on the server hosting OpenStack Networking, while logged in as the **root** user.

Procedure 7.5. Configuring the OpenStack Networking Service to Authenticate through the Identity Service

1. Set the authentication strategy to **keystone**:

```
# openstack-config --set /etc/neutron/neutron.conf \
   DEFAULT auth_strategy keystone
```

2. Set the Identity service host that OpenStack Networking must use:

```
# openstack-config --set /etc/neutron/neutron.conf \
   keystone_authtoken auth_host IP
```

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

3. Set OpenStack Networking to authenticate as the correct tenant:

```
# openstack-config --set /etc/neutron/neutron.conf \
   keystone_authtoken admin_tenant_name services
```

   Replace *services* with the name of the tenant created for the use of OpenStack Networking. Examples in this guide use **services**.

4. Set OpenStack Networking to authenticate using the **neutron** administrative user account:

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

5. Set OpenStack Networking to use the correct **`neutron`** administrative user account password:

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

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

7.2.6. Configure the Firewall to Allow OpenStack Networking Traffic

OpenStack Networking receives connections on TCP port **`9696`**. The firewall on the OpenStack Networking node must be configured to allow network traffic on this port. All steps in this procedure must be performed on the server hosting OpenStack Networking, while logged in as the **`root`** user.

Procedure 7.6. Configuring the Firewall to Allow OpenStack Networking Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port **`9696`**.

```
-A INPUT -p tcp -m multiport --dports 9696 -j ACCEPT
```

3. Add an INPUT rule for the firewall to accept VXLAN connections on port **`4789`**. The new rules must appear before any INPUT rules that REJECT traffic:

```
-A INPUT -p udp -m udp --dport 4789 -j ACCEPT
```

4. Save the changes to the **`/etc/sysconfig/iptables`** file.
5. Restart the **`iptables`** service to ensure that the change takes effect:

```
# systemctl restart iptables.service
```

7.2.7. Configure RabbitMQ Message Broker Settings for OpenStack Networking

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 system hosting OpenStack Networking, while logged in as the **`root`** user.

Procedure 7.7. Configuring the OpenStack Networking Service to use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

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

2. Set OpenStack Networking to connect to the RabbitMQ host:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_host RABBITMQ_HOST
```

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

3. Set the message broker port to **5672**:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_port 5672
```

4. Set the RabbitMQ user name and password created for OpenStack Networking when RabbitMQ was configured:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_userid neutron
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_password NEUTRON_PASS
```

Replace **neutron** and *NEUTRON_PASS* with the RabbitMQ user name and password created for OpenStack Networking.

5. When RabbitMQ was launched, the **neutron** user was granted read and write permissions to all resources: specifically, through the virtual host **/**. Configure the Networking service to connect to this virtual host:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_virtual_host /
```

7.2.8. Enable SSL Communication Between OpenStack Networking and the Message Broker

If you enabled SSL on the message broker, you must configure OpenStack Networking accordingly. This procedure requires the exported client certificates and key file. See Section 2.3.5, “Export an SSL Certificate for Clients” for instructions on how to export these files.

Procedure 7.8. Enabling SSL Communication Between OpenStack Networking and the RabbitMQ Message Broker

1. Enable SSL communication with the message broker:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT rabbit_use_ssl True
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT kombu_ssl_certfile /path/to/client.crt
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT kombu_ssl_keyfile /path/to/clientkeyfile.key
```

Replace the following values:

- Replace */path/to/client.crt* with the absolute path to the exported client certificate.
- Replace */path/to/clientkeyfile.key* with the absolute path to the exported client key file.

2. If your certificates were signed by a third-party Certificate Authority (CA), you must also run the following command:

```
# openstack-config --set /etc/neutron/neutron.conf \
 DEFAULT kombu_ssl_ca_certs /path/to/ca.crt
```

Replace */path/to/ca.crt* with the absolute path to the CA file provided by the third-party CA (see Section 2.3.4, “Enable SSL on the RabbitMQ Message Broker” for more information).

7.2.9. Configure OpenStack Networking to Communicate with the Compute Service

Configure OpenStack Networking to communicate with the Compute service about network topology changes.

Procedure 7.9. Configuring OpenStack Networking to Communicate with the Compute Service

1. Set OpenStack Networking to connect to the Compute controller node:

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

   Replace *CONTROLLER_IP* with the IP address or host name of the Compute controller node.

2. Set the user name, password, and tenant for the **nova** user:

   ```
   # openstack-config --set /etc/neutron/neutron.conf \
      nova username nova
   # openstack-config --set /etc/neutron/neutron.conf \
     nova auth_type password
   # openstack-config --set /etc/neutron/neutron.conf \
     nova password PASSWORD
    # openstack-config --set /etc/neutron/neutron.conf \
     nova project_name SERVICES
   ```

   Replace the *SERVICES* with the correct name of the nova project. Replace *PASSWORD* with the password set when the **nova** user was created.

3. Set OpenStack Networking to connect to the Compute controller node in an administrative context:

   ```
   # openstack-config --set /etc/neutron/neutron.conf \
     DEFAULT nova auth_url http://CONTROLLER_IP:35357/v3
   ```

   Replace *CONTROLLER_IP* with the IP address or host name of the Compute controller node.

4. Set OpenStack Networking to use the correct region for the Compute controller node:

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

7.2.10. Launch OpenStack Networking

Launch the **neutron-server** service and configure it to start at boot time:

```
# systemctl start neutron-server.service
# systemctl enable neutron-server.service
```

Important

By default, OpenStack Networking does not enforce Classless Inter-Domain Routing (CIDR) checking of IP addresses. This is to maintain backwards compatibility with previous releases. If you require such checks set the value of the **`force_gateway_on_subnet`** configuration key to **`True`** in the **`/etc/neutron/neutron.conf`** file.

7.3. Configure the DHCP Agent

Configure the DHCP agent. All steps in this procedure must be performed on the server hosting OpenStack Networking, while logged in as the **`root`** user.

Procedure 7.10. Configuring the DHCP Agent

1. Configure the DHCP agent to use the Identity service for authentication.

   a. Set the authentication strategy to **`keystone`**:

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
     DEFAULT auth_strategy keystone
   ```

   b. Set the Identity service host that the DHCP agent must use:

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
     keystone_authtoken auth_host IP
   ```

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

   c. Set the DHCP agent to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
     keystone_authtoken admin_tenant_name services
   ```

   Replace **`services`** with the name of the tenant created for the use of OpenStack Networking. Examples in this guide use **`services`**.

   d. Set the DHCP agent to authenticate using the **`neutron`** administrative user account:

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
     keystone_authtoken admin_user neutron
   ```

   e. Set the DHCP agent to use the correct **`neutron`** administrative user account password:

   ```
   # openstack-config --set /etc/neutron/dhcp_agent.ini \
     keystone_authtoken admin_password PASSWORD
   ```

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

2. Set the interface driver in the **`/etc/neutron/dhcp_agent.ini`** file based on the OpenStack Networking plug-in being used. If you are using ML2, select either driver. Use the command that applies to the plug-in used in your environment:

A. **Open vSwitch Interface Driver**

```
# openstack-config --set /etc/neutron/dhcp_agent.ini \
  DEFAULT interface_driver
neutron.agent.linux.interface.OVSInterfaceDriver
```

B. **Linux Bridge Interface Driver**

```
# openstack-config --set /etc/neutron/dhcp_agent.ini \
  DEFAULT interface_driver \
  neutron.agent.linux.interface.BridgeInterfaceDriver
```

3. Start the **`neutron-dhcp-agent`** service and configure it to start at boot time:

```
# systemctl start neutron-dhcp-agent.service
# systemctl enable neutron-dhcp-agent.service
```

7.4. Configure the Plug-in Agent

Configure the agent associated with the plug-in used in your environment. If you are using the ML2 plug-in configure the Open vSwitch agent.

7.4.1. Configure the Open vSwitch Plug-in Agent

You must install and enable the ML2 plug-in before configuring it. See Section 7.2.1.1, “Enable the ML2 Plug-in”.

The Open vSwitch plug-in has a corresponding agent. When the Open vSwitch plug-in is in use, all nodes in the environment that handle data packets must have the agent installed and configured. This includes all Compute nodes and systems hosting the dedicated DHCP and L3 agents.

Note

Open vSwitch support for TCP segmentation offload (TSO) and Generic Segmentation Offload (GSO) to VXLAN and GRE is enabled by default.

Procedure 7.11. Configuring the Open vSwitch Plug-in Agent

1. Start the **`openvswitch`** service:

```
# systemctl start openvswitch.service
```

2. Configure the **`openvswitch`** service to start at boot time:

```
# systemctl enable openvswitch.service
```

3. Each host running the Open vSwitch agent requires an Open vSwitch bridge called **`br-int`**, which is used for private network traffic. This bridge is created automatically.

Warning

The **br-int** bridge is required for the agent to function correctly. Once created, do not remove or otherwise modify the **br-int** bridge.

4. Set the value of the **bridge_mappings** configuration key to a comma-separated list of physical networks and the network bridges associated with them:

   ```
   # openstack-config --set
   /etc/neutron/plugins/ml2/openvswitch_agent.ini \
      ovs bridge_mappings PHYSNET:BRIDGE
   ```

   Replace *PHYSNET* with the name of a physical network, and replace *BRIDGE* with the name of the network bridge.

5. Start the **neutron-openvswitch-agent** service:

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

6. Configure the **neutron-openvswitch-agent** service to start at boot time:

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

7. Configure the **neutron-ovs-cleanup** service to start at boot time. This service ensures that the OpenStack Networking agents maintain full control over the creation and management of tap devices

   ```
   # systemctl enable neutron-ovs-cleanup.service
   ```

7.5. Create an External Network

OpenStack Networking provides two mechanisms for connecting the Layer 3 (L3) agent to an external network. The first, attaching it to an external bridge (**br-ex**) directly, is only supported when the Open vSwitch plug-in (or its functionality, implemented through ML2) is in use. The second method, which is supported by the ML2 plug-in, the Open vSwitch plug-in, and the Linux Bridge plug-in, is to use an external provider network.

All steps in this procedure must be performed on a server with the OpenStack Networking command-line interface (provided by the *python-neutronclient* package) installed. You must also have access to a **keystonerc_admin** file containing the authentication details of the Identity service administrative user.

Take note of the unique identifiers generated by the steps listed in this procedure. These identifiers will be required when configuring the L3 agent.

Procedure 7.12. Creating and Configuring an External Network

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

   ```
   # source ~/keystonerc_admin
   ```

2. Create a new provider network:

```
[(keystone_admin)]# neutron net-create EXTERNAL_NAME \
   --router:external \
   --provider:network_type TYPE \
   --provider:physical_network PHYSNET \
   --provider:segmentation_id VLAN_TAG
```

Replace the following values:

- Replace *EXTERNAL_NAME* with a name for the new external network provider.
- Replace *TYPE* with the type of provider network to use. Supported values are **`flat`** (for flat networks), **`vlan`** (for VLAN networks), and **`local`** (for local networks).
- Replace *PHYSNET* with a name for the physical network. This is not applicable if you intend to use a local network type. *PHYSNET* must match one of the values defined under **`bridge_mappings`** in the **`/etc/neutron/plugin.ini`** file.
- Replace *VLAN_TAG* with the VLAN tag that will be used to identify network traffic. The VLAN tag specified must have been defined by the network administrator. If the **`network_type`** was set to a value other than **`vlan`**, this parameter is not required.

Take note of the unique external network identifier returned; this is required in subsequent steps.

3. Create a new subnet for the external provider network:

```
[(keystone_admin)]# neutron subnet-create --gateway GATEWAY \
   --allocation-pool start=IP_RANGE_START,end=IP_RANGE_END \
   --disable-dhcp EXTERNAL_NAME EXTERNAL_CIDR
```

Replace the following values:

- Replace *GATEWAY* with the IP address or hostname 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 *EXTERNAL_CIDR*, but outside of the block of IP addresses specified by the range started by *IP_RANGE_START* and ended by *IP_RANGE_END*.
- Replace *IP_RANGE_START* 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 *IP_RANGE_END* 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.
- Replace *EXTERNAL_NAME* with the name of the external network the subnet is to be associated with. This must match the name that was provided to the **`net-create`** action in the previous step.
- Replace *EXTERNAL_CIDR* with the Classless Inter-Domain Routing (CIDR) representation of the block of IP addresses the subnet represents. An example is **`192.168.100.0/24`**. The block of IP addresses specified by the range started by *IP_RANGE_START* and ended by *IP_RANGE_END* **must** fall within the block of IP addresses specified by *EXTERNAL_CIDR*.

Take note of the unique subnet identifier returned; this is required in subsequent steps.

4. Create a new router:

```
[(keystone_admin)]# neutron router-create NAME
```

Replace *NAME* with a name for the new router. Take note of the unique router identifier returned; this is required in subsequent steps, and when configuring the L3 agent.

5. Link the router to the external provider network:

```
[(keystone_admin)]# neutron router-gateway-set ROUTER NETWORK
```

Replace *ROUTER* with the unique identifier of the router, and replace *NETWORK* with the unique identifier of the external provider network.

6. Link the router to each private network subnet:

```
[(keystone_admin)]# neutron router-interface-add ROUTER SUBNET
```

Replace *ROUTER* with the unique identifier of the router, and replace *SUBNET* with the unique identifier of a private network subnet. Perform this step for each existing private network subnet to which to link the router.

7.6. Configure the L3 Agent

Configure the Layer 3 agent. All steps in this procedure must be performed on the server hosting OpenStack Networking, while logged in as the **root** user.

Procedure 7.13. Configuring the L3 Agent

1. Configure the L3 agent to use the Identity service for authentication.

   a. Set the authentication strategy to **keystone**:

   ```
   # openstack-config --set /etc/neutron/metadata_agent.ini \
     DEFAULT auth_strategy keystone
   ```

   b. Set the Identity service host that the L3 agent must use:

   ```
   # openstack-config --set /etc/neutron/metadata_agent.ini \
     keystone_authtoken auth_host IP
   ```

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

   c. Set the L3 agent to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/neutron/metadata_agent.ini \
     keystone_authtoken admin_tenant_name services
   ```

   Replace *services* with the name of the tenant created for the use of OpenStack Networking. Examples in this guide use **services**.

   d. Set the L3 agent to authenticate using the **neutron** administrative user account:

   ```
   # openstack-config --set /etc/neutron/metadata_agent.ini \
     keystone_authtoken admin_user neutron
   ```

   e. Set the L3 agent to use the correct **neutron** administrative user account password:

```
# openstack-config --set /etc/neutron/metadata_agent.ini \
  keystone_authtoken admin_password PASSWORD
```

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

f. If the **neutron-metadata-agent** service and the **nova-metadata-api** service are not installed on the same server, set the address of the **nova-metadata-api** service:

```
# openstack-config --set /etc/neutron/metadata_agent.ini \
  DEFAULT nova_metadata_ip IP
```

Replace *IP* with the IP address of the server hosting the **nova-metadata-api** service.

2. Set the interface driver in the **/etc/neutron/l3_agent.ini** file based on the OpenStack Networking plug-in being used. Use the command the applies to the plug-in used in your environment:

A. **Open vSwitch Interface Driver**

```
# openstack-config --set /etc/neutron/l3_agent.ini \
  DEFAULT interface_driver
neutron.agent.linux.interface.OVSInterfaceDriver
```

B. **Linux Bridge Interface Driver**

```
# openstack-config --set /etc/neutron/l3_agent.ini \
  DEFAULT interface_driver
neutron.agent.linux.interface.BridgeInterfaceDriver
```

3. The L3 agent connects to external networks using either an external bridge or an external provider network. When using the Open vSwitch plug-in, either approach is supported. When using the Linux Bridge plug-in, only the use of an external provider network is supported. Set up the option that is most appropriate for your environment.

A. **Using an External Bridge**

Create and configure an external bridge and configure OpenStack Networking to use it. Perform these steps on each system hosting an instance of the L3 agent.

a. Create the external bridge, **br-ex**:

```
# ovs-vsctl add-br br-ex
```

b. Ensure that the **br-ex** device persists on reboot by creating a **/etc/sysconfig/network-scripts/ifcfg-br-ex** file, and adding the following lines:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
ONBOOT=yes
BOOTPROTO=none
```

c. Ensure that the L3 agent will use the external bridge:

```
# openstack-config --set /etc/neutron/l3_agent.ini \
  DEFAULT external_network_bridge br-ex
```

B. **Using a Provider Network**

To connect the L3 agent to external networks using a provider network, you must first have created the provider network. You must also have created a subnet and router to associate with it. The unique identifier of the router is required to complete these steps.

Set the value of the **`external_network_bridge`** configuration to be blank. This ensures that the L3 agent does not attempt to use an external bridge:

```
# openstack-config --set /etc/neutron/l3_agent.ini \
  DEFAULT external_network_bridge ""
```

4. Start the **`neutron-l3-agent`** service and configure it to start at boot time:

```
# systemctl start neutron-l3-agent.service
# systemctl enable neutron-l3-agent.service
```

5. The OpenStack Networking metadata agent allows virtual machine instances to communicate with the Compute metadata service. It runs on the same hosts as the L3 agent. Start the **`neutron-metadata-agent`** service and configure it to start at boot time:

```
# systemctl start neutron-metadata-agent.service
# systemctl enable neutron-metadata-agent.service
```

6. The **`leastrouter`** scheduler enumerates L3 Agent router assignment, and consequently schedules the router to the L3 Agent with the fewest routers. This differs from the ChanceScheduler behavior, which randomly selects from the candidate pool of L3 Agents.

a. Enable the **`leastrouter`** scheduler:

```
# openstack-config --set /etc/neutron/neutron.conf \
  DEFAULT router_scheduler_driver
neutron.scheduler.l3_agent_scheduler.LeastRoutersScheduler
```

b. Set up the shell to access keystone as the administrative user:

```
# source ~/keystonerc_admin
```

c. The router is scheduled once connected to a network. Unschedule the router:

```
[(keystone_admin)]# neutron l3-agent-router-remove L3_NODE_ID
ROUTER_ID
```

Replace *L3_NODE_ID* with the unique identifier of the agent on which the router is currently hosted, and replace *ROUTER_ID* with the unique identifier of the router.

d. Assign the router:

```
[(keystone_admin)]# neutron l3-agent-router-add L3_NODE_ID
ROUTER_ID
```

Replace *L3_NODE_ID* with the unique identifier of the agent on which the router is to be assigned, and replace *ROUTER_ID* with the unique identifier of the router.

7.7. Validate the OpenStack Networking Installation

To begin using OpenStack Networking, you must deploy networking components to Compute nodes. You must also define initial networks and routers. It is, however, possible to perform basic sanity checking of the OpenStack Networking deployment by following the steps outlined in this procedure.

Procedure 7.14. Validate the OpenStack Networking Installation

1. **On All Nodes**

   a. Verify that the customized Red Hat Enterprise Linux kernel intended for use with Red Hat OpenStack Platform is running:

   ```
   # uname --kernel-release
   2.6.32-358.6.2.openstack.el6.x86_64
   ```

   If the kernel release value returned does not contain the string **openstack**, update the kernel and reboot the system.

   b. Ensure that the installed IP utilities support network namespaces:

   ```
   # ip netns
   ```

   If an error indicating that the argument is not recognised or supported is returned, update the system using **yum**.

2. **On Service Nodes**

   a. Ensure that the **neutron-server** service is running:

   ```
   # openstack-status | grep neutron-server
   neutron-server:                         active
   ```

3. **On Network Nodes**

   Ensure that the following services are running:

   - DHCP agent (**neutron-dhcp-agent**)
   - L3 agent (**neutron-l3-agent**)
   - Plug-in agent, if applicable (**neutron-openvswitch-agent** or **neutron-linuxbridge-agent**)
   - Metadata agent (**neutron-metadata-agent**)

   ```
   # openstack-status | grep SERVICENAME
   ```

7.7.1. Troubleshoot OpenStack Networking Issues

This section discusses the commands you can use and procedures you can follow to troubleshoot OpenStack Networking issues.

Debugging Networking Device

- Use the **`ip a`** command to display all the physical and virtual devices.
- Use the **`ovs-vsctl show`** command to display the interfaces and bridges in a virtual switch.
- Use the **`ovs-dpctl show`** command to show datapaths on the switch.

Tracking Networking Packets

- Check where packets are not getting through:

  ```
  # tcpdump -n -i INTERFACE -e -w FILENAME
  ```

  Replace *INTERFACE* with the name of the network interface to check. The interface name can be the name of the bridge or host Ethernet device.

  The **`-e`** flag ensures that the link-level header is printed (in which the **`vlan`** tag will appear).

  The **`-w`** flag is optional. Use it if you want to write the output to a file. If not, the output is written to the standard output (**`stdout`**).

  For more information about **`tcpdump`**, see its manual page.

Debugging Network Namespaces

- Use the **`ip netns list`** command to list all known network namespaces.
- Show routing tables inside specific namespaces:

  ```
  # ip netns exec NAMESPACE_ID bash
  # route -n
  ```

  Start the **`ip netns exec`** command in a bash shell so that subsequent commands can be invoked without the **`ip netns exec`** command.

Chapter 8. Install the Compute Service

8.1. Install a Compute VNC Proxy

8.1.1. Install the Compute VNC Proxy Packages

The VNC proxy is available to users of the Compute service. Two types of VNC proxy server packages are available. The *openstack-nova-novncproxy* package provides VNC support to instances through a web browser (using Websockets), while the *openstack-nova-console* package provides access to instances through a traditional VNC client (through the **`openstack-nova-xvpvncproxy`** service).

The console authentication service, also provided by the *openstack-nova-console* package, is used to authenticate the VNC connections. Typically the console authentication service and the proxy utilities are installed on the same host as the Compute API service.

The following steps must be performed while logged in as the **`root`** user.

Procedure 8.1. Installing the Compute VNC proxy packages

- Install the VNC proxy utilities and the console authentication service:

  A. Install the *openstack-nova-novncproxy* package using the **`yum`** command:

  ```
  # yum install -y openstack-nova-novncproxy
  ```

  B. Install the *openstack-nova-console* package using the **`yum`** command:

  ```
  # yum install -y openstack-nova-console
  ```

The VNC proxy packages and the console authentication service are now installed and ready for configuration.

8.1.2. Configure the Firewall to Allow Compute VNC Proxy Traffic

The node that hosts VNC access to instances must be configured to allow VNC traffic through its firewall. By default, the **`openstack-nova-novncproxy`** service listens on TCP port 6080 and the **`openstack-nova-xvpvncproxy`** service listens on TCP port 6081.

The following procedure allows traffic on TCP port 6080 to traverse through the firewall for use by the *openstack-nova-novncproxy* package:

The following steps must be performed while logged in as the **`root`** user.

Procedure 8.2. Configuring the firewall to allow Compute VNC proxy traffic

1. Edit the **`/etc/sysconfig/iptables`** file and add the following on a new line underneath the *-A INPUT -i lo -j ACCEPT* line and before any *-A INPUT -j REJECT* rules:

   ```
   -A INPUT -m state --state NEW -m tcp -p tcp --dport 6080 -j ACCEPT
   ```

2. Save the file and exit the editor.

- Similarly, when using the **openstack-nova-xvpvncproxy** service, enable traffic on TCP port 6081 with the following on a new line in the same location:

  ```
  -A INPUT -m state --state NEW -m tcp -p tcp --dport 6081 -j ACCEPT
  ```

Once the file has been edited with the new firewall rule or rules, run the following commands as the **root** user to apply the changes:

```
# service iptables restart
```

```
# iptables-save
```

The firewall is now configured to allow VNC proxy traffic.

8.1.3. Configure the VNC Proxy Service

VNC access to instances is available over a web browser or with a traditional VNC client. The **/etc/nova/nova.conf** file holds the following VNC options:

- *vnc_enabled* - Default is true.
- *vncserver_listen* - The IP address to which VNC services will bind.
- *vncserver_proxyclient_address* - The IP address of the compute host used by proxies to connect to instances.
- *novncproxy_base_url* - The browser address where clients connect to instance.
- *novncproxy_port* - The port listening for browser VNC connections. Default is 6080.
- *xvpvncproxy_port* - The port to bind for traditional VNC clients. Default is 6081.

As the **root** user, use the **service** command to start the console authentication service:

```
# service openstack-nova-consoleauth start
```

Use the **chkconfig** command to permanently enable the service:

```
# chkconfig openstack-nova-consoleauth on
```

As the **root** user, use the **service** command on the nova node to start the browser-based service:

```
# service openstack-nova-novncproxy start
```

Use the **chkconfig** command to permanently enable the service:

```
# chkconfig openstack-nova-novncproxy on
```

To control access to the VNC service that uses a traditional client (non browser-based), substitute *openstack-nova-xvpvncproxy* into the previous commands.

8.1.4. Configure Live Migration

Red Hat OpenStack Platform supports live migration using either shared storage migration or block migration. The following sections provide general prerequisites for both types. For detailed configuration steps for both types, see "Migrate a Live (running) Instance".

8.1.4.1. General Requirements

General requirements for migration include:

- Access to the cloud environment on the command line as an administrator (all steps in this procedure are carried out on the command line). To execute commands, first load the user's authentication variables:

  ```
  # source ~/keystonerc_admin
  ```

- Both source and destination nodes must be located in the same subnet, and have the same processor type.
- All compute servers (controller and nodes) must be able to perform name resolution with each other.
- The UID and GID of the Compute service and libvirt users must be identical between compute nodes.
- The compute nodes must be using KVM with libvirt.

8.1.4.2. Multipathing Requirements

When migrating an instance with multipathing configured, you need to ensure consistent multipath device naming between the source and destination nodes. The migration will fail if the instance cannot resolve multipath device names in the destination node.

You can ensure consistent multipath device naming by forcing both source and destination nodes to use device WWIDs. To do this, disable *user-friendly names* and restart **multipathd** by running these commands on both source and destination nodes:

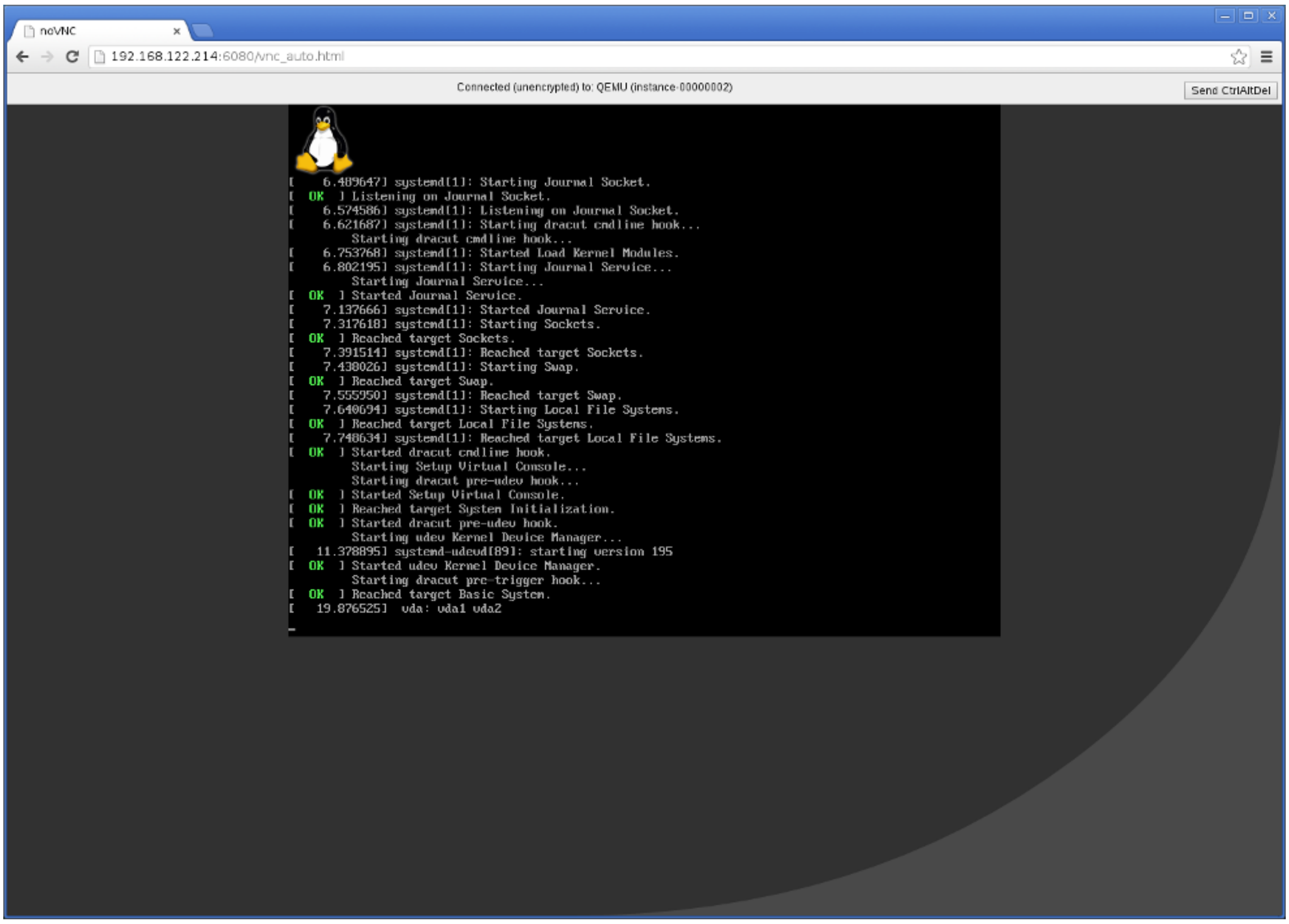
```
# mpathconf --enable --user_friendly_names n
# service multipathd restart
```

For more information, see Consistent Multipath Device Names in a Cluster from the DM Multipath guide.

8.1.5. Access Instances with the Compute VNC Proxy

Browse to the *novncproxy_base_url* URL provided in the **/etc/nova/nova.conf** file to access instance consoles.

The following image shows VNC access to a Fedora Linux instance with a web browser. It is provided only as an example, and settings such as IP addresses will be different in your environment.



Connected (unencrypted) to: QEMU (instance-00000002)

```
[    6.489647] systemd[1]: Starting Journal Socket.
[  OK  ] Listening on Journal Socket.
[    6.574586] systemd[1]: Listening on Journal Socket.
[    6.621687] systemd[1]: Starting dracut cmdline hook...
         Starting dracut cmdline hook...
[    6.753768] systemd[1]: Started Load Kernel Modules.
[    6.802195] systemd[1]: Starting Journal Service...
         Starting Journal Service...
[  OK  ] Started Journal Service.
[    7.137666] systemd[1]: Started Journal Service.
[    7.317618] systemd[1]: Starting Sockets.
[  OK  ] Reached target Sockets.
[    7.391514] systemd[1]: Reached target Sockets.
[    7.438026] systemd[1]: Starting Swap.
[  OK  ] Reached target Swap.
[    7.555950] systemd[1]: Reached target Swap.
[    7.640694] systemd[1]: Starting Local File Systems.
[  OK  ] Reached target Local File Systems.
[    7.748634] systemd[1]: Reached target Local File Systems.
[  OK  ] Started dracut cmdline hook.
         Starting Setup Virtual Console...
         Starting dracut pre-udev hook...
[  OK  ] Started Setup Virtual Console.
[  OK  ] Reached target System Initialization.
[  OK  ] Started dracut pre-udev hook.
         Starting udev Kernel Device Manager...
[   11.378895] systemd-udevd[89]: starting version 195
[  OK  ] Started udev Kernel Device Manager.
         Starting dracut pre-trigger hook...
[  OK  ] Reached target Basic System.
[   19.876525]  vda: vda1 vda2

_
```

Figure 8.1. VNC instance access

8.2. Install a Compute Node

8.2.1. Install the Compute Service Packages

The OpenStack Compute service requires the following packages:

openstack-nova-api

Provides the OpenStack Compute API service. At least one node in the environment must host an instance of the API service. This must be the node pointed to by the Identity service endpoint definition for the Compute service.

openstack-nova-compute

Provides the OpenStack Compute service.

openstack-nova-conductor

Provides the Compute conductor service. The conductor handles database requests made by Compute nodes, ensuring that individual Compute nodes do not require direct database access. At least one node in each environment must act as a Compute conductor.

openstack-nova-scheduler

Provides the Compute scheduler service. The scheduler handles scheduling of requests made to the API across the available Compute resources. At least one node in each environment must act as a Compute scheduler.

python-cinderclient

Provides client utilities for accessing storage managed by the Block Storage service. This package is not required if you do not intend to attach block storage volumes to your instances or you intend to manage such volumes using a service other than the Block Storage service.

Install the packages:

```
# yum install -y openstack-nova-api openstack-nova-compute \
  openstack-nova-conductor openstack-nova-scheduler \
  python-cinderclient
```

Note

In the example above, all Compute service packages are installed on a single node. In a production deployment, Red Hat recommends that the API, conductor, and scheduler services be installed on a separate controller node or on separate nodes entirely. The Compute service itself must be installed on each node that is expected to host virtual machine instances.

8.2.2. Create the Compute Service Database

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

Procedure 8.3. Creating the Compute Service Databases

1. Connect to the database service:

```
# mysql -u root -p
```

2. Create the **nova** and **nova_api** databases:

```
mysql> CREATE DATABASE nova;
```

```
mysql> CREATE DATABASE nova_api;
```

3. Create a **nova** database user and grant the user access to the**nova** and **nova_api** databases:

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

```
mysql> GRANT ALL ON nova.* TO 'nova'@'%' IDENTIFIED BY 'PASSWORD';
```

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

```
mysql> GRANT ALL ON nova_api.* TO 'nova'@'%' 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
```

8.2.3. Configure the Compute Service Database Connection

The database connection string used by the Compute service is defined in the **/etc/nova/nova.conf** file. It must be updated to point to a valid database server before starting the service.

The database connection string only needs to be set on nodes that are hosting the conductor service (**openstack-nova-conductor**). Compute nodes communicate with the conductor using the messaging infrastructure; the conductor orchestrates communication with the database. As a result, individual Compute nodes do not require direct access to the database. There must be at least one instance of the conductor service in any Compute environment.

All steps in this procedure must be performed on the server or servers hosting the Compute conductor service, while logged in as the **root** user.

Procedure 8.4. Configuring the Compute Service SQL Database Connection

- Set the value of the **sql_connection** configuration key:

```
# openstack-config --set /etc/nova/nova.conf \
  DEFAULT sql_connection mysql://USER:PASS@IP/DB
```

Replace the following values:

- Replace *USER* with the Compute service database user name, usually **nova**.
- Replace *PASS* with the password of the database user.
- Replace *IP* with the IP address or host name of the database server.
- Replace *DB* with the name of the Compute service database, usually **nova**.

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Compute service database user was granted access when creating the Compute service database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the Compute service database, you must enter 'localhost'.

8.2.4. Create the Compute Service Identity Records

8.2.4. Create the Compute Service Identity Records

Create and configure Identity service records required by the Compute service. These entries assist other OpenStack services attempting to locate and access the functionality provided by the Compute service.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, "Create an Administrator Account and the Identity Service Endpoint"
- Section 3.7, "Create the Services Tenant"

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 8.5. Creating Identity Records for the Compute Service

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

   ```
    #  source ~/keystonerc_admin
   ```

2. Create the **`compute`** user:

   ```
    [(keystone_admin)]# openstack user create --password PASSWORD compute
   +----------+----------------------------------+
   | Field    | Value                            |
   +----------+----------------------------------+
   | email    | None                             |
   | enabled  | True                             |
   | id       | 421665cdfaa24f93b4e904c81ff702ad |
   | name     | compute                          |
   | username | compute                          |
   +----------+----------------------------------+
   ```

   Replace *PASSWORD* with a secure password that will be used by the Compute service when authenticating with the Identity service.

3. Link the **`compute`** user and the **`admin`** role together within the context of the **`services`** tenant:

   ```
    [(keystone_admin)]# openstack role add --project services --user
   compute admin
   ```

4. Create the **`compute`** service entry:

   ```
    [(keystone_admin)]# openstack service create --name compute \
      --description "OpenStack Compute Service" \
      compute
   ```

5. Create the **`compute`** endpoint entry:

   ```
    [(keystone_admin)]# openstack endpoint create \
      --publicurl "http://IP:8774/v2/%(tenant_id)s" \
      --adminurl "http://IP:8774/v2/%(tenant_id)s" \
      --internalurl "http://IP:8774/v2/%(tenant_id)s" \
      --region RegionOne \
      compute
   ```

Replace *IP* with the IP address or host name of the system hosting the Compute API service.

8.2.5. Configure Compute Service Authentication

Configure the Compute service to use the Identity service for authentication. All steps in this procedure must be performed on each system hosting Compute services, while logged in as the **`root`** user.

Procedure 8.6. Configuring the Compute Service to Authenticate Through the Identity Service

1. Set the authentication strategy to **`keystone`**:

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

2. Set the Identity service host that the Compute service must use:

   ```
   # openstack-config --set /etc/nova/api-paste.ini \
     filter:authtoken auth_host IP
   ```

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

3. Set the Compute service to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/nova/api-paste.ini \
     filter:authtoken admin_tenant_name services
   ```

   Replace *services* with the name of the tenant created for the use of the Compute service. Examples in this guide use **`services`**.

4. Set the Compute service to authenticate using the **`compute`** administrative user account:

   ```
   # openstack-config --set /etc/nova/api-paste.ini \
     filter:authtoken admin_user compute
   ```

5. Set the Compute service to use the correct **`compute`** administrative user account password:

   ```
   # openstack-config --set /etc/nova/api-paste.ini \
     filter:authtoken admin_password PASSWORD
   ```

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

8.2.6. Configure the Firewall to Allow Compute Service Traffic

Connections to virtual machine consoles, whether direct or through the proxy, are received on ports **`5900`** to **`5999`**. Connections to the Compute API service are received on port **`8774`**. The firewall on the service node must be configured to allow network traffic on these ports. All steps in this procedure must be performed on each Compute node, while logged in as the **`root`** user.

Procedure 8.7. Configuring the Firewall to Allow Compute Service Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.

2. Add an INPUT rule allowing TCP traffic on ports in the ranges **5900** to **5999**. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 5900:5999 -j ACCEPT
   ```

3. Add an INPUT rule allowing TCP traffic on port **8774**. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 8774 -j ACCEPT
   ```

4. Save the changes to the **/etc/sysconfig/iptables** file.

5. Restart the **iptables** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

8.2.7. Configure the Compute Service to Use SSL

Use the following options in the **nova.conf** file to configure SSL.

Table 8.1. SSL Options for Compute

Configuration Option	Description
enabled_ssl_apis	A list of APIs with enabled SSL.
ssl_ca_file	The CA certificate file to use to verify connecting clients.
ssl_cert_file	The SSL certificate of the API server.
ssl_key_file	The SSL private key of the API server.
tcp_keepidle	Sets the value of TCP_KEEPIDLE in seconds for each server socket. Defaults to 600.

8.2.8. Configure RabbitMQ Message Broker Settings for the Compute Service

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 systems hosting the Compute controller service and Compute nodes, while logged in as the **root** user.

Procedure 8.8. Configuring the Compute Service to use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

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

2. Set the Compute service to connect to the RabbitMQ host:

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

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

3. Set the message broker port to **5672**:

   ```
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT rabbit_port 5672
   ```

4. Set the RabbitMQ user name and password created for the Compute service when RabbitMQ was configured:

   ```
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT rabbit_userid nova
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT rabbit_password NOVA_PASS
   ```

   Replace **nova** and *NOVA_PASS* with the RabbitMQ user name and password created for the Compute service.

5. When RabbitMQ was launched, the **nova** user was granted read and write permissions to all resources: specifically, through the virtual host `/`. Configure the Compute service to connect to this virtual host:

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

8.2.9. Enable SSL Communication Between the Compute Service and the Message Broker

If you enabled SSL on the message broker, you must configure the Compute service accordingly. This procedure requires the exported client certificates and key file. See Section 2.3.5, "Export an SSL Certificate for Clients" for instructions on how to export these files.

Procedure 8.9. Enabling SSL Communication Between the Compute Service and the RabbitMQ Message Broker

1. Enable SSL communication with the message broker:

   ```
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT rabbit_use_ssl True
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT kombu_ssl_certfile /path/to/client.crt
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT kombu_ssl_keyfile /path/to/clientkeyfile.key
   ```

   Replace the following values:

   - Replace */path/to/client.crt* with the absolute path to the exported client certificate.
   - Replace */path/to/clientkeyfile.key* with the absolute path to the exported client key file.

2. If your certificates were signed by a third-party Certificate Authority (CA), you must also run the following command:

   ```
   # openstack-config --set /etc/nova/nova.conf \
     DEFAULT kombu_ssl_ca_certs /path/to/ca.crt
   ```

Replace */path/to/ca.crt* with the absolute path to the CA file provided by the third-party CA (see Section 2.3.4, “Enable SSL on the RabbitMQ Message Broker” for more information).

8.2.10. Configure Resource Overcommitment

OpenStack supports overcommitting of CPU and memory resources on Compute nodes. Overcommitting is a technique of allocating more virtualized CPUs and/or memory than there are physical resources.

Important

Overcommitting increases the number of instances you are able to run, but reduces instance performance.

CPU and memory overcommit settings are represented as a ratio. OpenStack uses the following ratios by default:

- The default CPU overcommit ratio is 16. This means that up to 16 virtual cores can be assigned to a node for each physical core.
- The default memory overcommit ratio is 1.5. This means that instances can be assigned to a physical node if the total instance memory usage is less than 1.5 times the amount of physical memory available.

Use the **`cpu_allocation_ratio`** and **`ram_allocation_ratio`** directives in **`/etc/nova/nova.conf`** to change these default settings.

8.2.11. Reserve Host Resources

You can reserve host memory and disk resources so that they are always available to OpenStack. To prevent a given amount of memory and disk resources from being considered as available to be allocated for usage by virtual machines, edit the following directives in **`/etc/nova/nova.conf`**:

- **`reserved_host_memory_mb`**. Defaults to 512MB.
- **`reserved_host_disk_mb`**. Defaults to 0MB.

8.2.12. Configure Compute Networking

8.2.12.1. Compute Networking Overview

Unlike Nova-only deployments, when OpenStack Networking is in use, the **`nova-network`** service **must** not run. Instead all network related decisions are delegated to the OpenStack Networking Service.

Therefore, it is very important that you refer to this guide when configuring networking, rather than relying on Nova networking documentation or past experience with Nova networking. In particular, using CLI tools like **`nova-manage`** and **`nova`** to manage networks or IP addressing, including both fixed and floating IPs, is not supported with OpenStack Networking.

Important

It is strongly recommended that you uninstall **nova-network** and reboot any physical nodes that were running **nova-network** before using these nodes to run OpenStack Network. Problems can arise from inadvertently running the **nova-network** process while using OpenStack Networking service; for example, a previously running **nova-network** could push down stale firewall rules.

8.2.12.2. Update the Compute Configuration

Each time a Compute instance is provisioned or deprovisioned, the service communicates with OpenStack Networking through its API. To facilitate this connection, you must configure each Compute node with the connection and authentication details outlined in this procedure.

All steps in the following procedure must be performed on each Compute node, while logged in as the **root** user.

Procedure 8.10. Updating the Connection and Authentication Settings of Compute Nodes

1. Modify the **network_api_class** configuration key to indicate that OpenStack Networking is in use:

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

2. Set the Compute service to use the endpoint of the OpenStack Networking API:

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

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

3. Set the name of the project used by the OpenStack Networking service. Examples in this guide use *SERVICES*:

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

4. Set the authentication method that will be used by the OpenStack Networking service:

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

5. Set the name of the OpenStack Networking administrative user:

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

6. Set the password associated with the OpenStack Networking administrative user:

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

7. Set the URL associated with the Identity service endpoint:

   ```
   # openstack-config --set /etc/nova/nova.conf \
                       neutron auth_url http://IP:35357/
   ```

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

8. Enable the metadata proxy and configure the metadata proxy secret:

   ```
   # openstack-config --set /etc/nova/nova.conf \
     neutron service_metadata_proxy true
   # openstack-config --set /etc/nova/nova.conf \
     neutron metadata_proxy_shared_secret METADATA_SECRET
   ```

   Replace *METADATA_SECRET* with the string that the metadata proxy will use to secure communication.

9. Configure the metadata proxy secret for neutron as well:

   ```
   # openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
     metadata_proxy_shared_secret METADATA_SECRET
   ```

   Replace *METADATA_SECRET* with the string that the metadata proxy will use to secure communication.

10. Enable the use of OpenStack Networking security groups:

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

11. Set the firewall driver to **`nova.virt.firewall.NoopFirewallDriver`**:

    ```
    # openstack-config --set /etc/nova/nova.conf \
      DEFAULT firewall_driver nova.virt.firewall.NoopFirewallDriver
    ```

    This **must** be done when OpenStack Networking security groups are in use.

12. Open the **`/etc/sysctl.conf`** file in a text editor, and add or edit the following kernel networking parameters:

    ```
    net.ipv4.ip_forward = 1
    net.ipv4.conf.all.rp_filter = 0
    net.ipv4.conf.default.rp_filter = 0
    net.bridge.bridge-nf-call-arptables = 1
    net.bridge.bridge-nf-call-ip6tables = 1
    net.bridge.bridge-nf-call-iptables = 1
    ```

13. Load the updated kernel parameters:

    ```
    # sysctl -p
    ```

8.2.12.3. Configure the L2 Agent

Each compute node must run an instance of the Layer 2 (L2) agent appropriate to the networking plug-in that is in use.

- Section 7.4.1, “Configure the Open vSwitch Plug-in Agent”

8.2.12.4. Configure Virtual Interface Plugging

When **`nova-compute`** creates an instance, it must 'plug' each of the vNIC associated with the instance into a OpenStack Networking controlled virtual switch. Compute must also inform the virtual switch of the OpenStack Networking port identifier associated with each vNIC.

A generic virtual interface driver, **`nova.virt.libvirt.vif.LibvirtGenericVIFDriver`**, is provided in Red Hat OpenStack Platform. This driver relies on OpenStack Networking being able to return the type of virtual interface binding required. The following plug-ins support this operation:

- Linux Bridge
- Open vSwitch
- NEC
- BigSwitch
- CloudBase Hyper-V
- Brocade

To use the generic driver, execute the **`openstack-config`** command to set the value of the **`vif_driver`** configuration key appropriately:

```
# openstack-config --set /etc/nova/nova.conf \
  libvirt vif_driver \
  nova.virt.libvirt.vif.LibvirtGenericVIFDriver
```

Important

Considerations for Open vSwitch and Linux Bridge deployments:

- If running Open vSwitch with security groups enabled, use the Open vSwitch specific driver, **`nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver`**, instead of the generic driver.
- For Linux Bridge environments, you must add the following to the **`/etc/libvirt/qemu.conf`** file to ensure that the virtual machine launches properly:

```
user = "root"
group = "root"
cgroup_device_acl = [
   "/dev/null", "/dev/full", "/dev/zero",
   "/dev/random", "/dev/urandom",
   "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
   "/dev/rtc", "/dev/hpet", "/dev/net/tun",
]
```

8.2.13. Populate the Compute Service Database

Populate the Compute Service database after you have successfully configured the Compute Service database connection string.

Important

This procedure must be followed only once to initialize and populate the database. You do not need to perform these steps again when adding additional systems hosting Compute services.

Procedure 8.11. Populating the Compute Service Database

1. Log in to a system hosting an instance of the **`openstack-nova-conductor`** service.
2. Switch to the **`nova`** user:

```
# su nova -s /bin/sh
```

3. Initialize and populate the database identified in **`/etc/nova/nova.conf`**:

```
$ nova-manage db sync
```

8.2.14. Launch the Compute Services

Procedure 8.12. Launching Compute Services

1. Libvirt requires that the **`messagebus`** service be enabled and running. Start the service:

```
# systemctl start messagebus.service
```

2. The Compute service requires that the **`libvirtd`** service be enabled and running. Start the service and configure it to start at boot time:

```
# systemctl start libvirtd.service
# systemctl enable libvirtd.service
```

3. Start the API service on each system that is hosting an instance of it. Note that each API instance should either have its own endpoint defined in the Identity service database or be pointed to by a load balancer that is acting as the endpoint. Start the service and configure it to start at boot time:

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

4. Start the scheduler on each system that is hosting an instance of it. Start the service and configure it to start at boot time:

```
# systemctl start openstack-nova-scheduler.service
# systemctl enable openstack-nova-scheduler.service
```

5. Start the conductor on each system that is hosting an instance of it. Note that it is recommended that this service is not run on every Compute node as this eliminates the security benefits of restricting direct database access from the Compute nodes. Start the service and configure it to start at boot time:

```
# systemctl start openstack-nova-conductor.service
# systemctl enable openstack-nova-conductor.service
```

6. Start the Compute service on every system that is intended to host virtual machine instances. Start the service and configure it to start at boot time:

```
# systemctl start openstack-nova-compute.service
# systemctl enable openstack-nova-compute.service
```

7. Depending on your environment configuration, you may also need to start the following services:

   `openstack-nova-cert`

   The X509 certificate service, required if you intend to use the EC2 API to the Compute service.

   Note

   To use the EC2 API to the Compute service, you must set the options in the **`nova.conf`** configuration file. For more information, see *Configuring the EC2 API* section in the *Red Hat OpenStack Platform Configuration Reference Guide*. This document is available from the following link:

   https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform

   `openstack-nova-network`

   The Nova networking service. Note that you **must** not start this service if you have installed and configured, or intend to install and configure, OpenStack Networking.

   `openstack-nova-objectstore`

   The Nova object storage service. It is recommended that the Object Storage service (Swift) is used for new deployments.

Chapter 9. Install the Orchestration Service

9.1. Install the Orchestration Service Packages

The Orchestration service requires the following packages:

openstack-heat-api

Provides the OpenStack-native REST API to the Orchestration engine service.

openstack-heat-api-cfn

Provides the AWS CloudFormation-compatible API to the Orchestration engine service.

openstack-heat-common

Provides components common to all Orchestration services.

openstack-heat-engine

Provides the OpenStack API for launching templates and submitting events back to the API.

openstack-heat-api-cloudwatch

Provides the AWS CloudWatch-compatible API to the Orchestration engine service.

heat-cfntools

Provides the tools required on **heat**-provisioned cloud instances.

python-heatclient

Provides a Python API and command-line script, both of which make up a client for the Orchestration API service.

openstack-utils

Provides supporting utilities to assist with a number of tasks, including the editing of configuration files.

Install the packages:

```
# yum install -y openstack-heat-* python-heatclient openstack-utils
```

9.2. Configure the Orchestration Service

To configure the Orchestration service, you must complete the following tasks:

- Configure a database for the Orchestration service.
- Bind each Orchestration API service to a corresponding IP address.
- Create and configure the Orchestration service Identity records.
- Configure how Orchestration services authenticate with the Identity service.

The following sections describe each procedure in detail.

9.2.1. Create the Orchestration Service Database

Create the database and database user used by the Orchestration service. The database connection string used by the Orchestration service is defined in the **/etc/heat/heat.conf** file. It must be updated to point to a valid database server before the service is started. All steps in this procedure must be performed on the database server, while logged in as the **root** user.

Procedure 9.1. Configuring the Orchestration Service Database

1. Connect to the database service:

   ```
   # mysql -u root -p
   ```

2. Create the **heat** database:

   ```
   mysql> CREATE DATABASE heat;
   ```

3. Create a database user named **heat** and grant the user access to the **heat** database:

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

   Replace *PASSWORD* with a secure password that will be user 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
   ```

6. Set the value of the **sql_connection** configuration key:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT sql_connection mysql://heat:PASSWORD@IP/heat
   ```

   Replace the following values:

   - Replace *PASSWORD* with the password of the **heat** database user.
   - Replace *IP* with the IP address or host name of the database server.

7. As the **heat** user, sync the database:

   ```
   # runuser -s /bin/sh heat -c "heat-manage db_sync"
   ```

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Orchestration service database user was granted access when creating the Orchestration service database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the Orchestration service database, you must enter 'localhost'.

9.2.2. Restrict the Bind Addresses of Each Orchestration API Service

After configuring the database, set the **`bind_host`** setting of each Orchestration API service. This setting controls which IP address a service should use for incoming connections.

Set the **`bind_host`** setting for each Orchestration API service:

```
# openstack-config --set /etc/heat/heat.conf
  heat_api bind_host IP
# openstack-config --set /etc/heat/heat.conf
  heat_api_cfn bind_host IP
# openstack-config --set /etc/heat/heat.conf
  heat_api_cloudwatch bind_host IP
```

Replace *IP* with the IP address that the corresponding API should use.

9.2.3. Create the Orchestration Service Identity Records

Create and configure Identity service records required by the Orchestration service. These entries assist other OpenStack services attempting to locate and access the functionality provided by the Orchestration service.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 9.2. Creating Identity Records for the Orchestration Service

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

```
# source ~/keystonerc_admin
```

2. Create the **`heat`** user:

```
[(keystone_admin)]# openstack user create --password PASSWORD heat
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 5902673a8d3a4552b813dff31717476b |
| name     | heat                             |
| username | heat                             |
+----------+----------------------------------+
```

Replace *PASSWORD* with a password that will be used by the Orchestration service when authenticating with the Identity service.

3. Link the **heat** user and the **admin** role together within the context of the **services** tenant:

```
[(keystone_admin)]# openstack role add --project services --user heat
admin
```

4. Create the **heat** and **heat-cfn** service entries:

```
[(keystone_admin)]# openstack service create --name heat orchestration
[(keystone_admin)]# openstack service create --name heat-cfn
cloudformation
```

5. Create endpoint entries for the **heat** service and the **heat-cfn** service:

```
[(keystone_admin)]# openstack endpoint create \
   --publicurl 'HEAT_CFN_IP:8000/v1' \
   --adminurl 'HEAT_CFN_IP:8000/v1' \
   --internalurl 'HEAT_CFN_IP:8000/v1' \
   --region RegionOne \
   heat-cfn
[(keystone_admin)]# openstack endpoint create \
   --publicurl 'HEAT_IP:8004/v1/%(tenant_id)s' \
   --adminurl 'HEAT_IP:8004/v1/%(tenant_id)s' \
   --internalurl 'HEAT_IP:8004/v1/%(tenant_id)s' \
   --region RegionOne \
   heat
```

Replace the following values:

- Replace *HEAT_CFN_IP* with the IP or host name of the system hosting the **heat-cfn** service.
- Replace *HEAT_IP* with the IP or host name of the system hosting the **heat** service.

Important

Include the **http://** prefix for *HEAT_CFN_IP* and *HEAT_IP* values.

9.2.3.1. Create the Required Identity Domain for the Orchestration Service

The Orchestration service requires its own Identity domain, through which users can be created and associated with credentials deployed inside instances owned by **heat** stacks. Using a separate domain allows for separation between the instances and the user deploying the stack. This allows regular users without administrative rights to deploy **heat** stacks that require such credentials.

Procedure 9.3. Creating an Identity Service Domain for the Orchestration Service

1. Create the **heat** domain:

   ```
   # openstack --os-url=http://IDENTITY_IP:5000/v3 \
     --os-identity-api-version=3 \
     --description "Owns users and projects created by heat"
     domain create heat
   ```

   Replace *IDENTITY_IP* with the IP or host name of the server hosting the Identity service.

   This command returns the domain ID of the **heat** domain. This ID (*HEAT_DOMAIN_ID*) is used in the next step.

2. Create a user named **heat_domain_admin** that can have administrative rights within the **heat** domain:

   ```
   # openstack --os-url=http://IDENTITY_IP:5000/v3 \
     --os-identity-api-version=3 user create heat_domain_admin \
     --password PASSWORD \
     --domain HEAT_DOMAIN_ID \
     --description "Manages users and projects created by heat"
   ```

   Replace *PASSWORD* with a password for this user. This command returns a user ID (*DOMAIN_ADMIN_ID*), which is used in the next step.

3. Grant the **heat_domain_admin** user administrative rights within the **heat** domain:

   ```
   # openstack --os-url=http://IDENTITY_IP:5000/v3 \
     --os-identity-api-version=3 role add --user DOMAIN_ADMIN_ID \
     --domain HEAT_DOMAIN_ID admin
   ```

4. On the server hosting the Orchestration service, configure the service to use the **heat** domain and user:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT stack_domain_admin_password DOMAIN_PASSWORD
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT stack_domain_admin heat_domain_admin
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT stack_user_domain HEAT_DOMAIN_ID
   ```

9.2.4. Configure Orchestration Service Authentication

Configure the Orchestration service to use the Identity service for authentication. All steps in this procedure must be performed on each system hosting Orchestration services, while logged in as the **root** user.

Procedure 9.4. Configuring the Orchestration Service to Authenticate Through the Identity Service

1. Set the Orchestration services to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     keystone_authtoken admin_tenant_name services
   ```

Replace *services* is the name of the tenant created for the use of the Orchestration service. Examples in this guide use **services**.

2. Set the Orchestration services to authenticate using the **heat** administrative user account:

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

3. Set the Orchestration services to use the correct **heat** administrative user account password:

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

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

4. Set the Identity service host that the Orchestration services must use:

```
# openstack-config --set /etc/heat/heat.conf \
  keystone_authtoken service_host KEYSTONE_HOST
# openstack-config --set /etc/heat/heat.conf \
  keystone_authtoken auth_host KEYSTONE_HOST
# openstack-config --set /etc/heat/heat.conf \
  keystone_authtoken auth_uri http://KEYSTONE_HOST:35357/v2.0
# openstack-config --set /etc/heat/heat.conf \
  keystone_authtoken keystone_ec2_uri http://KEYSTONE_HOST:35357/v2.0
```

Replace *KEYSTONE_HOST* with the IP address or host name of the server hosting the Identity service. If the Identity service is hosted on the same system, use **127.0.0.1**.

5. Configure the **heat-api-cfn** and **heat-api-cloudwatch** service host names to which virtual machine instances will connect:

```
# openstack-config --set /etc/heat/heat.conf \
  DEFAULT heat_metadata_server_url HEAT_CFN_HOST:8000
# openstack-config --set /etc/heat/heat.conf \
  DEFAULT heat_waitcondition_server_url
HEAT_CFN_HOST:8000/v1/waitcondition
# openstack-config --set /etc/heat/heat.conf \
  DEFAULT heat_watch_server_url HEAT_CLOUDWATCH_HOST:8003
```

Replace the following values:

- Replace *HEAT_CFN_HOST* with the IP address or host name of the server hosting the **heat-api-cfn** service.
- Replace *HEAT_CLOUDWATCH_HOST* with the IP address or host name of the server hosting the **heat-api-cloudwatch** service.

Important

Even if all services are hosted on the same system, *do not* use **127.0.0.1** for either service host name. This IP address refers to the local host of each instance, and would therefore prevent the instance from reaching the actual service.

6. Application templates use wait conditions and signaling for orchestration. Define the Identity role for users that should receive progress data. By default, this role is **heat_stack_user**:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT heat_stack_user_role heat_stack_user
   ```

9.2.5. Configure RabbitMQ Message Broker Settings for the Orchestration Service

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 system hosting the Orchestration controller service, while logged in as the **root** user.

Procedure 9.5. Configuring the Orchestration Service to use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

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

2. Set the Orchestration service to connect to the RabbitMQ host:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_host RABBITMQ_HOST
   ```

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

3. Set the message broker port to **5672**:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_port 5672
   ```

4. Set the RabbitMQ user name and password created for the Orchestration service when RabbitMQ was configured:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_userid heat
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_password HEAT_PASS
   ```

   Replace **heat** and *HEAT_PASS* with the RabbitMQ user name and password created for the Orchestration service.

5. When RabbitMQ was launched, the **heat** user was granted read and write permissions to all resources: specifically, through the virtual host **/**. Configure the Orchestration service to connect to this virtual host:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_virtual_host /
   ```

9.2.6. Enable SSL Communication Between the Orchestration Service and the Message Broker

If you enabled SSL on the message broker, you must configure the Orchestration service accordingly. This procedure requires the exported client certificates and key file. See Section 2.3.5, “Export an SSL Certificate for Clients” for instructions on how to export these files.

Procedure 9.6. Enabling SSL Communication Between the Orchestration Service and the RabbitMQ Message Broker

1. Enable SSL communication with the message broker:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT rabbit_use_ssl True
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT kombu_ssl_certfile /path/to/client.crt
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT kombu_ssl_keyfile /path/to/clientkeyfile.key
   ```

   Replace the following values:

   - Replace */path/to/client.crt* with the absolute path to the exported client certificate.
   - Replace */path/to/clientkeyfile.key* with the absolute path to the exported client key file.

2. If your certificates were signed by a third-party Certificate Authority (CA), you must also run the following command:

   ```
   # openstack-config --set /etc/heat/heat.conf \
     DEFAULT kombu_ssl_ca_certs /path/to/ca.crt
   ```

   Replace */path/to/ca.crt* with the absolute path to the CA file provided by the third-party CA (see Section 2.3.4, “Enable SSL on the RabbitMQ Message Broker” for more information).

9.3. Launch the Orchestration Service

Procedure 9.7. Launching Orchestration Services

1. Start the Orchestration API service, and configure it to start at boot time:

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

2. Start the Orchestration AWS CloudFormation-compatible API service, and configure it to start at boot time:

   ```
   # systemctl start openstack-heat-api-cfn.service
   # systemctl enable openstack-heat-api-cfn.service
   ```

3. Start the Orchestration AWS CloudWatch-compatible API service, and configure it to start at boot time:

   ```
   # systemctl start openstack-heat-api-cloudwatch.service
   # systemctl enable openstack-heat-api-cloudwatch.service
   ```

4. Start the Orchestration API service for launching templates and submitting events back to the API, and configure it to start at boot time:

```
#  systemctl start openstack-heat-engine.service
#  systemctl enable openstack-heat-engine.service
```

9.4. Deploy a Stack Using Orchestration Templates

The Orchestration engine service uses templates (defined as **`.template`** files) to launch instances, IPs, volumes, or other types of stacks. The **heat** utility is a command-line interface that allows you to create, configure, and launch stacks.

Note

The *openstack-heat-templates* package provides sample templates that you can use to test core Orchestration features. It also contains template-related scripts and conversion tools. To install this package, run the following command:

```
#  yum install -y openstack-heat-templates
```

Some Orchestration templates launch instances that require access to the **`openstack-heat-api-cfn`** service. Such instances must be able to communicate with the **`openstack-heat-api-cloudwatch`** service and the **`openstack-heat-api-cfn`** service. The IPs and ports used by these services are the values set in the **`/etc/heat/heat.conf`** file as **`heat_metadata_server_url`** and **`heat_watch_server_url`**.

To allow access to these services, you must open the ports used by **`openstack-heat-api-cloudwatch`** (8003), **`openstack-heat-api-cfn`** (8000), and **`openstack-api`** (8004).

Procedure 9.8. Deploying a Stack Using Orchestration Templates

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.
2. Add the following INPUT rules to allow TCP traffic on ports **`8003`**, **`8000`**, and **`8004`**:

   ```
   -A INPUT -i BR -p tcp --dport 8003 -j ACCEPT
   -A INPUT -i BR -p tcp --dport 8000 -j ACCEPT
   -A INPUT -p tcp -m multiport --dports 8004 -j ACCEPT
   ```

   Replace *BR* with the interface of the bridge used by the instances launched from Orchestration templates. Do not include the **`-i`** ***BR*** parameter in the **`INPUT`** rules if you are not using **`nova-network`**, or if the Orchestration service and **`nova-compute`** are not hosted on the same server.
3. Save the changes to the **`/etc/sysconfig/iptables`** file.
4. Restart the **`iptables`** service for the firewall changes to take effect:

   ```
   #  systemctl restart iptables.service
   ```
5. Launch an application:

```
#  heat stack-create STACKNAME \
   --template-file=PATH_TEMPLATE \
 --parameters="PARAMETERS"
```

Replace the following values:

- Replace *STACKNAME* with the name to assign to the stack. This name will appear when you run the **`heat stack-list`** command.
- Replace *PATH_TEMPLATE* with the path to your **`.template`** file.
- Replace *PARAMETERS* with a semicolon-delimited list of stack creation parameters to use. Supported parameters are defined in the template file itself.

9.5. Integrate Telemetry and Orchestration Services

The Orchestration service can use the Telemetry service (and its alarms) to monitor the resource usage of stacks created using the **`heat stack-create`** command. To enable this, the Orchestration service must be installed and configured accordingly (see Section 12.1, “Overview of Telemetry Service Deployment” for more information).

Telemetry service alarms are used by the *autoscaling* feature. This feature allows the Orchestration service to automatically create stacks when the usage of a specific resource reaches a certain level. To allow Orchestration to use Telemetry alarms, uncomment or add the following line in the **`resource_registry`** section of **`/etc/heat/environment.d/default.yaml`**:

```
"AWS::CloudWatch::Alarm":
"file:///etc/heat/templates/AWS_CloudWatch_Alarm.yaml"
```

Chapter 10. Install the Dashboard

10.1. Dashboard Service Requirements

The system hosting the dashboard service must be configured in the following way:

- The *httpd*, *mod_wsgi*, and *mod_ssl* packages must be installed (for security purposes):

  ```
  # yum install -y mod_wsgi httpd mod_ssl
  ```

- The system must have a connection to the Identity service, as well as to the other OpenStack API services (Compute, Block Storage, Object Storage, Image, and Networking services).

- You must know the URL of the Identity service endpoint.

10.2. Install the Dashboard Packages

Install the packages required by the dashboard service.

Note

The dashboard service uses a configurable back-end session store. This installation uses **`memcached`** as the session store.

The following package is required:

openstack-dashboard

Provides the OpenStack dashboard service.

If you are using **`memcached`**, the following packages must also be installed:

memcached

Memory-object caching system that speeds up dynamic web applications by alleviating database load.

python-memcached

Python interface to the **`memcached`** daemon.

Procedure 10.1. Installing the Dashboard Packages

1. If required, install the **`memcached`** object caching system:

   ```
   # yum install -y memcached python-memcached
   ```

2. Install the dashboard package:

   ```
   # yum install -y openstack-dashboard
   ```

10.3. Launch the Apache Web Service

The dashboard is a Django (Python) web application; it is hosted by the **httpd** service. Start the service, and configure it to start at boot time:

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

10.4. Configure the Dashboard

10.4.1. Configure Connections and Logging

Before users connect to the dashboard for the first time, the following parameters must be configured in the **/etc/openstack-dashboard/local_settings** file (sample files are available in the *Configuration Reference Guide* at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform):

Procedure 10.2. Configuring Connections and Logging for the Dashboard

1. Set the **ALLOWED_HOSTS** parameter with a comma-separated list of host/domain names that the application can serve. For example:

   ```
   ALLOWED_HOSTS = ['horizon.example.com', 'localhost', '192.168.20.254',
   ]
   ```

2. Update the **CACHES** settings with the **memcached** values:

   ```
   SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
   CACHES = {
    'default': {
     'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
     'LOCATION' : 'memcacheURL:port',
    }
   }
   ```

   Replace the following values:

   - Replace *memcacheURL* with IP address of the host on which **memcached** was installed.
   - Replace *port* with the value from the **PORT** parameter in the **/etc/sysconfig/memcached** file.

3. Specify the host URL for the Identity service endpoint. For example:

   ```
   OPENSTACK_KEYSTONE_URL="127.0.0.1"
   ```

4. Update the dashboard's time zone:

   ```
   TIME_ZONE="UTC"
   ```

   The time zone can also be updated using the dashboard GUI.

5. To ensure the configuration changes take effect, restart the Apache service:

   ```
   # systemctl restart httpd.service
   ```

Note

The **`HORIZON_CONFIG`** dictionary contains all the settings for the dashboard. Whether or not a service is in the dashboard depends on the Service Catalog configuration in the Identity service.

Note

It is recommended that you use the **`django-secure`** module to ensure that most of the recommended practices and modern browser protection mechanisms are enabled. For more information http://django-secure.readthedocs.org/en/latest/ (*django-secure*).

10.4.2. Configure the Dashboard to Use HTTPS

Although the default installation uses a non-encrypted channel (HTTP), it is possible to enable SSL support for the dashboard.

Procedure 10.3. Configuring the Dashboard to use HTTPS

1. Open the **`/etc/openstack-dashboard/local_settings`** file in a text editor, and uncomment the following parameters:

   ```
   SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTOCOL', 'https')
   CSRF_COOKIE_SECURE = True
   SESSION_COOKIE_SECURE = True
   ```

   The latter two settings instruct the browser to only send dashboard cookies over HTTPS connections, ensuring that sessions will not work over HTTP.

2. Open the **`/etc/httpd/conf/httpd.conf`** file in a text editor, and add the following line:

   ```
   NameVirtualHost *:443
   ```

3. Open the **`/etc/httpd/conf.d/openstack-dashboard.conf`** file in a text editor.

   a. Delete the following lines:

      ```
      WSGIDaemonProcess dashboard
      WSGIProcessGroup dashboard
      WSGISocketPrefix run/wsgi

      WSGIScriptAlias /dashboard /usr/share/openstack-
      dashboard/openstack_dashboard/wsgi/django.wsgi
      Alias /static /usr/share/openstack-dashboard/static/

      <Directory /usr/share/openstack-
      dashboard/openstack_dashboard/wsgi>
          <IfModule mod_deflate.c>
            SetOutputFilter DEFLATE
            <IfModule mod_headers.c>
              # Make sure proxies donâ  t deliver the wrong content
              Header append Vary User-Agent env=!dont-vary
      ```

```
      </IfModule>
    </IfModule>

  Order allow,deny
  Allow from all
</Directory>
<Directory /usr/share/openstack-dashboard/static>
  <IfModule mod_expires.c>
    ExpiresActive On
    ExpiresDefault "access 6 month"
  </IfModule>
  <IfModule mod_deflate.c>
    SetOutputFilter DEFLATE
  </IfModule>

  Order allow,deny
  Allow from all
</Directory>

  RedirectMatch permanent ^/$
https://xxx.xxx.xxx.xxx:443/dashboard
```

b. Add the following lines:

```
WSGIDaemonProcess dashboard
WSGIProcessGroup dashboard
WSGISocketPrefix run/wsgi
LoadModule ssl_module modules/mod_ssl.so

<VirtualHost *:80>
  ServerName openstack.example.com
  RedirectPermanent / https://openstack.example.com/
</VirtualHost>

<VirtualHost *:443>
    ServerName openstack.example.com
    SSLEngine On
    SSLCertificateFile /etc/httpd/SSL/openstack.example.com.crt
    SSLCACertificateFile
/etc/httpd/SSL/openstack.example.com.crt
    SSLCertificateKeyFile
/etc/httpd/SSL/openstack.example.com.key
    SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-
shutdown
    WSGIScriptAlias / /usr/share/openstack-
dashboard/openstack_dashboard/wsgi/django.wsgi
    WSGIDaemonProcess horizon user=apache group=apache
processes=3 threads=10
    RedirectPermanent /dashboard https://openstack.example.com
    Alias /static /usr/share/openstack-dashboard/static/
    <Directory /usr/share/openstack-
dashboard/openstack_dashboard/wsgi>
      Order allow,deny
      Allow from all
    </Directory>
</VirtualHost>
```

```
<Directory /usr/share/openstack-dashboard/static>
  <IfModule mod_expires.c>
    ExpiresActive On
    ExpiresDefault "access 6 month"
  </IfModule>
  <IfModule mod_deflate.c>
    SetOutputFilter DEFLATE
  </IfModule>

Order allow,deny
Allow from all
</Directory>

RedirectMatch permanent ^/$ /dashboard/
```

In the new configuration, Apache listens on port 443 and redirects all non-secured requests to the HTTPS protocol. The **`<VirtualHost *:443>`** section defines the required options for this protocol, including private key, public key, and certificates.

4. Restart the Apache service and the **`memcached`** service:

```
# systemctl restart httpd.service
# systemctl restart memcached.service
```

When using the HTTP version of the dashboard (through the browser), the user is redirected to the HTTPS version of the page.

10.4.3. Change the Default Role for the Dashboard

By default, the dashboard service uses the Identity role, **`_member_`**, which is created automatically by the Identity service. This is adequate for regular users. If you choose to create a different role and set the dashboard to use this role, you must create this role in the Identity service prior to using the dashboard, then configure the dashboard to use it.

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **`keystone`** command-line utility is installed.

Procedure 10.4. Changing the Default Role for the Dashboard

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

```
# source ~/keystonerc_admin
```

2. Create the new role:

```
[(keystone_admin)]# keystone role-create --name NEW_ROLE
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
| id       | 8261ac4eabcc4da4b01610dbad6c038a |
| name     |             NEW_ROLE             |
+----------+----------------------------------+
```

Replace *NEW_ROLE* with a name for the role.

3. Open the **`/etc/openstack-dashboard/local_settings`** file in a text editor, and change the value of the following parameter:

   ```
   OPENSTACK_KEYSTONE_DEFAULT_ROLE = 'NEW_ROLE'
   ```

   Replace *NEW_ROLE* with the name of the role you created in the previous step.

4. Restart the Apache service for the change to take effect:

   ```
   # systemctl restart httpd.service
   ```

10.4.4. Configure SELinux

SELinux is a security feature of Red Hat Enterprise Linux that provides access control. SELinux status values are 'Enforcing', 'Permissive', and 'Disabled'. If SELinux is in 'Enforcing' mode, you must modify the SELinux policy to allow connections from the **`httpd`** service to the Identity server. This is also recommended if SELinux is configured in 'Permissive' mode.

Procedure 10.5. Configuring SELinux to Allow Connections from the Apache Service

1. Check the status of SELinux on the system:

   ```
   # getenforce
   ```

2. If the resulting value is 'Enforcing' or 'Permissive', allow connections between the **`httpd`** service and the Identity service:

   ```
   # setsebool -P httpd_can_network_connect on
   ```

10.4.5. Configure the Dashboard Firewall

To allow users to connect to the dashboard, you must configure the system firewall to allow connections. The **`httpd`** service and the dashboard support both HTTP and HTTPS connections. All steps in this procedure must be performed on the server hosting the **`httpd`** service, while logged in as the **`root`** user.

Note

To protect authentication credentials and other data, it is highly recommended that you enable only HTTPS connections.

Procedure 10.6. Configuring the Firewall to Allow Dashboard Traffic

1. Open the **`/etc/sysconfig/iptables`** configuration file in a text editor:

   - To allow incoming connections using only HTTPS, add the following firewall rule:

     ```
     -A INPUT -p tcp --dport 443 -j ACCEPT
     ```

   - To allow incoming connections using both HTTP and HTTPS, add the following firewall rule:

     ```
     -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
     ```

2. Restart the **`iptables`** service for the changes to take effect:

```
# systemctl restart iptables.service
```

Important

These rules allow communication on ports 80 and 443 from all remote hosts to the server running the dashboard service. For information regarding the creation of more restrictive firewall rules, see the *Red Hat Enterprise Linux Security Guide* at the following link:

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

10.5. Validate Dashboard Installation

After the dashboard has been successfully installed and configured, access the user interface with your web browser. Replace *HOSTNAME* with the host name or IP address of the server on which you installed the dashboard service:

- HTTPS

```
https://HOSTNAME/dashboard/
```

- HTTP

```
http://HOSTNAME/dashboard/
```

When prompted, log in using the credentials of your OpenStack user.

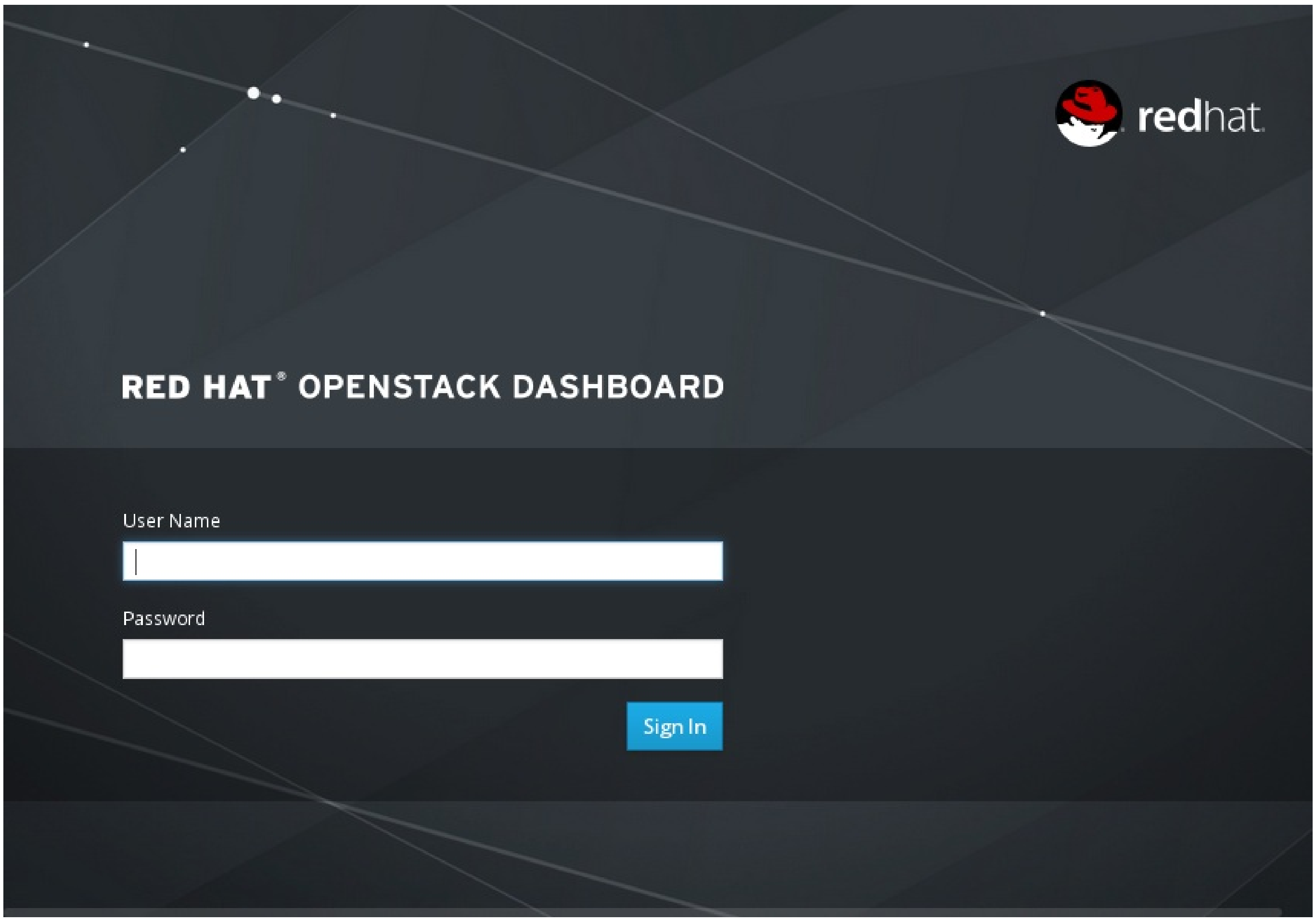


Figure 10.1. Dashboard Login Screen

Chapter 11. Install the Data Processing Service

11.1. Install the Data Processing Service Packages

On the server hosting the Data Processing service, install the *openstack-sahara-api* and *openstack-sahara-engine* packages:

```
# yum install openstack-sahara-api openstack-sahara-engine
```

This package provides the Data Processing CLI clients (**sahara** and **sahara-db-manage**) and the **`openstack-sahara-api`** service.

11.2. Configure the Data Processing Service

To configure the Data Processing service (Sahara), you must complete the following tasks:

- Configure the Data Processing service database connection.
- Configure the Data Processing API service to authenticate with the Identity service.
- Configure the firewall to allow service traffic for the Data Processing service (through port **`8386`**).

11.2.1. Create the Data Processing Service Database

Create the database and database user used by the Data Processing API service. The database connection string used by the Data Processing service is defined in the **`/etc/sahara/sahara.conf`** file. It must be updated to point to a valid database server before starting the Data Processing API service (**`openstack-sahara-api`**).

Procedure 11.1. Creating and Configuring a Database for the Data Processing API Service

1. Connect to the database service:

   ```
   # mysql -u root -p
   ```

2. Create the **`sahara`** database:

   ```
   mysql> CREATE DATABASE sahara;
   ```

3. Create a **`sahara`** database user and grant the user access to the**`sahara`** database:

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

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

4. Exit the **`mysql`** client:

   ```
   mysql> quit
   ```

5. Set the value of the **sql_connection** configuration key:

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

   Replace the following values:

   - Replace *PASS* with the password of the database user.
   - Replace *IP* with the IP address or host name of the server hosting the database service.

6. Configure the schema of the **sahara** database:

```
#  sahara-db-manage --config-file /etc/sahara/sahara.conf upgrade head
```

Important

The IP address or host name specified in the connection configuration key must match the IP address or host name to which the Data Processing service database user was granted access when creating the Data Processing service database. Moreover, if the database is hosted locally and you granted permissions to 'localhost' when creating the Data Processing service database, you must enter 'localhost'.

11.2.2. Create the Data Processing Service Identity Records

Create and configure Identity service records required by the Data Processing service. These entries assist other OpenStack services attempting to locate and access the functionality provided by the Data Processing service.

This procedure assumes that you have already created an administrative user account and a **services** tenant. For more information, see:

- Section 3.5, “Create an Administrator Account and the Identity Service Endpoint”
- Section 3.7, “Create the Services Tenant”

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **keystonerc_admin** file and on which the **keystone** command-line utility is installed.

Procedure 11.2. Creating Identity Records for the Data Processing Service

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

```
#  source ~/keystonerc_admin
```

2. Create the **sahara** user:

```
[(keystone_admin)]#  openstack user create --password PASSWORD sahara
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | 1fc5b7ac48b646ab850854e565ac1cfc |
| name     | sahara                           |
| username | sahara                           |
+----------+----------------------------------+
```

Replace *PASSWORD* with a password that will be used by the Data Processing service when authenticating with the Identity service.

3. Link the **sahara** user and the **admin** role together within the context of the **services** tenant:

```
[(keystone_admin)]# openstack role add --project services --user
sahara admin
```

4. Create the **sahara** service entry:

```
[(keystone_admin)]# openstack service create --name sahara \
--description "OpenStack Data Processing" \
data-processing
```

5. Create the **sahara** endpoint entry:

```
[(keystone_admin)]# openstack endpoint create \
   --publicurl 'http://SAHARA_HOST:8386/v1.1/%(tenant_id)s' \
   --adminurl 'http://SAHARA_HOST:8386/v1.1/%(tenant_id)s' \
   --internalurl 'http://SAHARA_HOST:8386/v1.1/%(tenant_id)s' \
   --region RegionOne \
   sahara
```

Replace *SAHARA_HOST* with the IP address or fully qualified domain name of the server hosting the Data Processing service.

Note

By default, the endpoint is created in the default region, **RegionOne**. This is a case-sensitive value. To specify a different region when creating an endpoint, use the ***--region*** argument to provide it.

See Section 3.5.1, “Service Regions” for more information.

11.2.3. Configure Data Processing Service Authentication

Configure the Data Processing API service (**openstack-sahara-api**) to use the Identity service for authentication. All steps in this procedure must be performed on the server hosting the Data Processing API service, while logged in as the **root** user.

Procedure 11.3. Configuring the Data Processing API Service to Authenticate through the Identity Service

1. Set the Identity service host that the Data Processing API service must use:

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

Replace *IP* with the IP address of the server hosting the Identity service.

2. Set the Data Processing API service to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/sahara/sahara.conf \
     keystone_authtoken admin_tenant_name services
   ```

   Replace *services* with the name of the tenant created for the use of the Data Processing service. Examples in this guide use **services**.

3. Set the Data Processing API service to authenticate using the **sahara** administrative user account:

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

4. Set the Data Processing API service to use the correct **sahara** administrative user account password:

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

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

11.2.4. Configure the Firewall to Allow OpenStack Data Processing Service Traffic

The Data Processing service receives connections on port **8386**. The firewall on the service node must be configured to allow network traffic on this port. All steps in this procedure must be performed on the server hosting the Data Processing service, while logged in as the **root** user.

Procedure 11.4. Configuring the Firewall to Allow Data Processing Service Traffic

1. Open the **/etc/sysconfig/iptables** file in a text editor.

2. Add an INPUT rule allowing TCP traffic on port **8386**. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 8386 -j ACCEPT
   ```

3. Save the changes to the **/etc/sysconfig/iptables** file.

4. Restart the **iptables** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

11.3. Configure and Launch the Data Processing Service

Procedure 11.5. Launching the Data Processing Service

1. If your OpenStack deployment uses OpenStack Networking (**`neutron`**), you must configure the Data Processing service accordingly:

   ```
   # openstack-config --set /etc/sahara/sahara.conf \
     DEFAULT use_neutron true
   ```

2. Start the Data Processing services and configure them to start at boot time:

   ```
   # systemctl start openstack-sahara-api.service
   # systemctl start openstack-sahara-engine.service
   # systemctl enable openstack-sahara-api.service
   # systemctl enable openstack-sahara-engine.service
   ```

Chapter 12. Install the Telemetry Service

12.1. Overview of Telemetry Service Deployment

The Telemetry service is composed of an API service, three **openstack-ceilometer** agents. The API service (provided by the *openstack-ceilometer-api* package) runs on one or more central management servers to provide access to the Telemetry database.

Note

At present, **mongod** is the only database service supported by the Telemetry service.

The three Telemetry agents (and their respective packages) are listed below:

- The Central agent (provided by *openstack-ceilometer-central*) runs on a central management server to poll public REST APIs for utilization statistics about resources that are not visible (either through notifications or from the hypervisor layer).
- The Collector (provided by *openstack-ceilometer-collector*) runs on one or more central management servers to receive notifications on resource usage. The Collector also parses resource usage statistics and saves them as datapoints in the Telemetry database.
- The Compute agent (provided by *openstack-ceilometer-compute*) runs on each Compute service node to poll for instance utilization statistics. You must install and configure the Compute service before installing the *openstack-ceilometer-compute* package on any node.

You must configure the following settings for each of the components:

- Authentication, including the Identity service tokens and Telemetry secret
- The database connection string, for connecting to the Telemetry database

The authentication settings and database connection string for these components are all configured in **/etc/ceilometer/ceilometer.conf**. As such, components deployed on the same host will share the same settings. If Telemetry components are deployed on multiple hosts, you must replicate any authentication changes to these hosts by copying the **ceilometer.conf** file to each host after applying the new settings.

Once the Telemetry service (all of its components, wherever each is hosted) is deployed and configured, you must configure each monitored service (Image, Networking, Object Storage, Block Storage, and each Compute node) to submit data to the Telemetry service. The related settings are configured in each service's configuration file.

12.2. Install the Telemetry Service Packages

The Telemetry service requires the following packages:

mongodb

Provides the MongoDB database service. The Telemetry service uses MongoDB as its back-end data repository.

openstack-ceilometer-api

Provides the **ceilometer** API service.

openstack-ceilometer-central

Provides the Central **ceilometer** agent.

openstack-ceilometer-collector

Provides the **ceilometer** Collector agent.

openstack-ceilometer-common

Provides components common to all **ceilometer** services.

openstack-ceilometer-compute

Provides the **ceilometer** agent that must run on each Compute node.

openstack-ceilometer-notification

Provides the **ceilometer** Notification agent. This agent provides metrics to the Collector agent from different OpenStack services.

python-ceilometer

Provides the **ceilometer** Python library.

python-ceilometerclient

Provides the **ceilometer** command-line tool and a Python API (specifically, the **ceilometerclient** module).

You can deploy the API Server, Central agent, MongoDB database service, and Collector on different hosts. Each Compute node must also have a Compute agent installed; this agent gathers detailed usage metrics on instances running on the Compute node.

Install the required packages on the same host:

```
# yum install -y mongodb openstack-ceilometer-* python-ceilometer python-ceilometerclient
```

12.3. Configure the MongoDB Back End and Create the Telemetry Database

The Telemetry service uses MongoDB as its back-end data repository. Before starting the **mongod** service, optionally configure **mongod** to run with the **--smallfiles** parameter. This parameter configures MongoDB to use a smaller default data file and journal size. MongoDB will limit the size of each data file, creating and writing to a new one when it reaches 512MB.

Procedure 12.1. Configuring the MongoDB Back End and Creating the Telemetry Database

1. Optionally configure **mongod** to run with the **--smallfiles** parameter. Open the **/etc/sysconfig/mongod** file in a text editor, and add the following line:

   ```
   OPTIONS="--smallfiles /etc/mongodb.conf"
   ```

MongoDB uses the parameters specified in the **OPTIONS** section when **mongod** launches.

2. Start the MongoDB service:

```
# systemctl start mongod.service
```

3. If the database must be accessed from a server other than its local host, open the **/etc/mongod.conf** file in a text editor, and update the **bind_ip** with the IP address of your MongoDB server:

```
bind_ip = MONGOHOST
```

4. Open the **/etc/sysconfig/iptables** file in a text editor and add an INPUT rule allowing TCP traffic on port **27017**. The new rule must appear before any INPUT rules that REJECT traffic:

```
-A INPUT -p tcp -m multiport --dports 27017 -j ACCEPT
```

5. Restart the **iptables** service to ensure that the change takes effect:

```
# systemctl restart iptables.service
```

6. Create a database for the Telemetry service:

```
# mongo --host MONGOHOST --eval '
  db = db.getSiblingDB("ceilometer");
  db.addUser({user: "ceilometer",
   pwd: "MONGOPASS",
   roles: [ "readWrite", "dbAdmin" ]})'
```

This also creates a database user named **ceilometer**. Replace *MONGOHOST* with the IP address or host name of the server hosting the MongoDB database. Replace *MONGOPASS* with a password for the **ceilometer** user.

12.4. Configure the Telemetry Service Database Connection

The database connection URL used by the Telemetry service is defined in the **/etc/ceilometer/ceilometer.conf** file. It must be updated to point to a valid database server before starting the Telemetry API service (**openstack-ceilometer-api**), Notification agent (**openstack-ceilometer-notification**), and Collector agent (**openstack-ceilometer-collector**).

All steps in this procedure must be performed on the server or servers hosting the **openstack-ceilometer-api** service and the **openstack-ceilometer-collector** service, while logged in as the **root** user.

Procedure 12.2. Configuring the Telemetry Service Database Connection

- Set the database connection string:

```
# openstack-config --set /etc/ceilometer/ceilometer.conf \
  database connection mongodb://ceilometer:MONGOPASS@MONGOHOST/ceilometer
```

Replace the following values:

- Replace *MONGOPASS* with the password of the **`ceilometer`** user; it is required by the Telemetry service to log in to the database server. Supply these credentials only when required by the database server (for example, when the database server is hosted on another system or node).
- Replace *MONGOHOST* with the IP address or host name and the port of the server hosting the database service.

If MongoDB is hosted locally on the same host, use the following database connection string:

```
mongodb://localhost:27017/ceilometer
```

12.5. Create the Telemetry Identity Records

Create and configure Identity service records required by the Telemetry service. These entries assist other OpenStack services attempting to locate and access the functionality provided by the Telemetry service.

This procedure assumes that you have already created an administrative user account and a **`services`** tenant. For more information, see:

- Section 3.5, "Create an Administrator Account and the Identity Service Endpoint"
- Section 3.7, "Create the Services Tenant"

Perform this procedure on the Identity service server, or on any machine onto which you have copied the **`keystonerc_admin`** file and on which the **keystone** command-line utility is installed.

Procedure 12.3. Creating Identity Records for the Telemetry Service

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

   ```
    #  source ~/keystonerc_admin
   ```

2. Create the **`ceilometer`** user:

   ```
    [(keystone_admin)]#  openstack user create --password PASSWORD --email
   CEILOMETER_EMAIL  ceilometer
   ```

   Replace the following values:

   - Replace *PASSWORD* with the password that will be used by the Telemetry service when authenticating with the Identity service.
   - Replace *CEILOMETER_EMAIL* with the email address used by the Telemetry service.

3. Create the **`ResellerAdmin`** role:

   ```
    [(keystone_admin)]#  openstack role create ResellerAdmin
   +-----------+----------------------------------+
   | Field     | Value                            |
   +-----------+----------------------------------+
   | domain_id | None                             |
   | id        | 9276cfe40bca485bacc1775d10ef98f6 |
   | name      | ResellerAdmin                    |
   +-----------+----------------------------------+
   ```

4. Link the **ceilometer** user and the **ResellerAdmin** role together within the context of the **services** tenant:

```
[(keystone_admin)]# openstack role add --project services --user
ceilometer ResellerAdmin
```

5. Link the **ceilometer** user and the **admin** role together within the context of the **services** tenant:

```
[(keystone_admin)]# openstack role add --project services --user
ceilometer admin
```

6. Create the **ceilometer** service entry:

```
[(keystone_admin)]# openstack service create --name ceilometer \
   --description "OpenStack Telemetry Service" \
    metering
```

7. Create the **ceilometer** endpoint entry:

```
[(keystone_admin)]# openstack endpoint create \
   --publicurl 'IP:8777' \
   --adminurl 'IP:8777' \
   --internalurl 'IP:8777' \
   --region RegionOne \
   ceilometer
```

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

Note

By default, the endpoint is created in the default region, **RegionOne**. This is a case-sensitive value. To specify a different region when creating an endpoint, use the ***--region*** argument to provide it.

See Section 3.5.1, "Service Regions" for more information.

12.6. Configure Telemetry Service Authentication

Configure the Telemetry API service (**openstack-ceilometer-api**) to use the Identity service for authentication. All steps in this procedure must be performed on the server hosting the Telemetry API service, while logged in as the **root** user.

Procedure 12.4. Configuring the Telemetry Service to Authenticate Through the Identity Service

1. Set the Identity service host that the Telemetry API service must use:

```
# openstack-config --set /etc/ceilometer/ceilometer.conf \
   keystone_authtoken auth_host IP
```

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

2. Set the authentication port that the Telemetry API service must use:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken auth_port PORT
   ```

   Replace *PORT* with the authentication port used by the Identity service, usually **35357**.

3. Set the Telemetry API service to use the **http** protocol for authenticating:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken auth_protocol http
   ```

4. Set the Telemetry API service to authenticate as the correct tenant:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     keystone_authtoken admin_tenant_name services
   ```

   Replace *services* with the name of the tenant created for the use of the Telemetry service. Examples in this guide use **services**.

5. Set the Telemetry service to authenticate using the **ceilometer** administrative user account:

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

6. Set the Telemetry service to use the correct **ceilometer** administrative user account password:

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

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

7. The Telemetry secret is a string used to help secure communication between all components of the Telemetry service across multiple hosts (for example, between the Collector agent and a Compute node agent). Set the Telemetry secret:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     publisher_rpc metering_secret SECRET
   ```

   Replace *SECRET* with the string that all Telemetry service components should use to sign and verify messages that are sent or received over AMQP.

8. Configure the service endpoints to be used by the Central agent, Compute agents, and Evaluator on the host where each component is deployed:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT os_auth_url http://IP:35357/v2.0
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT os_username ceilometer
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT os_tenant_name services
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT os_password PASSWORD
   ```

Replace the following values:

- Replace *IP* with the IP address or host name of the server hosting the Identity service.
- Replace *PASSWORD* with the password set when the **`ceilometer`** user was created.

12.7. Configure the Firewall to Allow Telemetry Service Traffic

The Telemetry service receives connections on port **`8777`**. The firewall on the service node must be configured to allow network traffic on this port. All steps in this procedure must be performed on the server hosting the Telemetry service, while logged in as the **`root`** user.

Procedure 12.5. Configuring the Firewall to Allow Telemetry Service Traffic

1. Open the **`/etc/sysconfig/iptables`** file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port **`8777`**. The new rule must appear before any INPUT rules that REJECT traffic:

   ```
   -A INPUT -p tcp -m multiport --dports 8777 -j ACCEPT
   ```

3. Save the changes to the **`/etc/sysconfig/iptables`** file.
4. Restart the **`iptables`** service to ensure that the change takes effect:

   ```
   # systemctl restart iptables.service
   ```

12.8. Configure RabbitMQ Message Broker Settings for the Telemetry Service

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 system hosting the Telemetry service, while logged in as the **`root`** user.

Procedure 12.6. Configuring the Telemetry Service to Use the RabbitMQ Message Broker

1. Set RabbitMQ as the RPC back end:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rpc_backend rabbit
   ```

2. Set the Telemetry service to connect to the RabbitMQ host:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rabbit_host RABBITMQ_HOST
   ```

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

3. Set the message broker port to **`5672`**:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rabbit_port 5672
   ```

4. Set the RabbitMQ user name and password created for the Telemetry service when RabbitMQ was configured:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rabbit_userid ceilometer
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rabbit_password CEILOMETER_PASS
   ```

   Replace **ceilometer** and *CEILOMETER_PASS* with the RabbitMQ user name and password created for the Telemetry service.

5. When RabbitMQ was launched, the **ceilometer** user was granted read and write permissions to all resources: specifically, through the virtual host **/**. Configure the Telemetry service to connect to this virtual host:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf \
     DEFAULT rabbit_virtual_host /
   ```

12.9. Configure the Compute Node

The Telemetry service monitors each node by collecting usage data from the Compute agent (**openstack-ceilometer-compute**) installed on that node. You can configure a node's Compute agent by replicating the **/etc/ceilometer/ceilometer.conf** file from another host whose Telemetry components have already been configured.

You must also configure the Compute node itself to enable notifications.

Procedure 12.7. Enabling Notifications on a Compute Node

1. Install *openstack-ceilometer-compute*, *python-ceilometer* and *python-ceilometerclient* on the node:

   ```
   # yum install openstack-ceilometer-compute python-ceilometer python-
   ceilometerclient
   ```

2. Enable auditing on the node:

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

3. Configure the audit frequency:

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

4. Configure what type of state changes should trigger a notification:

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

5. Set the node to use the correct notification drivers. Open the **/etc/nova/nova.conf** file in a text editor, and add the following lines in the **DEFAULT** section:

```
notification_driver = messagingv2
notification_driver = ceilometer.compute.nova_notifier
```

The Compute node requires two different notification drivers, which are defined using the same configuration key. You cannot use **openstack-config** to set these values.

6. Start the Compute agent:

```
# systemctl start openstack-ceilometer-compute.service
```

7. Configure the agent to start at boot time:

```
# systemctl enable openstack-ceilometer-compute.service
```

8. Restart the **openstack-nova-compute** service to apply all changes to **/etc/nova/nova.conf**:

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

12.10. Configure Monitored Services

The Telemetry service can also monitor the Image service, OpenStack Networking, the Object Storage service, and the Block Storage service. You must configure each service to submit samples to the Collector services. Before configuring any of these services, you must install the *python-ceilometer* and *python-ceilometerclient* packages on the node hosting the service:

```
# yum install python-ceilometer python-ceilometerclient
```

Note

Restart each service after configuring it to be monitored by the Telemetry service.

Image service (glance)

```
# openstack-config --set /etc/glance/glance-api.conf \
  DEFAULT notifier_strategy NOTIFYMETHOD
```

Replace *NOTIFYMETHOD* with a notification queue: **rabbit** (to use a **rabbitmq** queue) or **qpid** (to use a **qpid** message queue).

Block Storage service (cinder)

```
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT notification_driver messagingv2
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT rpc_backend cinder.openstack.common.rpc.impl_kombu
# openstack-config --set /etc/cinder/cinder.conf \
  DEFAULT control_exchange cinder
```

Object Storage service (swift)

The Telemetry service collects samples from the Object Storage service (**`swift`**) through the **`ResellerAdmin`** role that was created when configuring the required Identity records for Telemetry. You must also configure the Object Storage service to process traffic from **`ceilometer`**.

1. Open the **`/etc/swift/proxy-server.conf`** file in a text editor, and add or update the following lines:

   ```
   [filter:ceilometer]
   use = egg:ceilometer#swift

   [pipeline:main]
   pipeline = healthcheck cache authtoken keystoneauth ceilometer
   proxy-server
   ```

2. Add the **`swift`** user to the **`ceilometer`** group:

   ```
    # usermod -a -G ceilometer swift
   ```

3. Allow the Object Storage service to output logs to **`/var/log/ceilometer/swift-proxy-server.log`**:

   ```
    # touch /var/log/ceilometer/swift-proxy-server.log
    # chown ceilometer:ceilometer /var/log/ceilometer/swift-proxy-
   server.log
    # chmod 664 /var/log/ceilometer/swift-proxy-server.log
   ```

OpenStack Networking (`neutron`)

Telemetry supports the use of labels for distinguishing IP ranges. Enable OpenStack Networking integration with Telemetry:

```
 # openstack-config --set /etc/neutron/neutron.conf \
   DEFAULT notification_driver messagingv2
```

12.11. Launch the Telemetry API and Agents

Launch the corresponding service for each component of the Telemetry service, and configure each service to start at boot time:

```
 # systemctl start SERVICENAME.service
 # systemctl enable SERVICENAME.service
```

Replace *SERVICENAME* with the corresponding name of each Telemetry component service:

- openstack-ceilometer-compute
- openstack-ceilometer-central
- openstack-ceilometer-collector
- openstack-ceilometer-api
- openstack-ceilometer-notification

Chapter 13. Install the Telemetry Alarming Service

The Telemetry Alarming service (**Aodh**) triggers defined actions based on the data collected by the Telemetry service.

13.1. Install the Telemetry Alarming Service Packages

The following packages provide the components of the Telemetry Alarming service:

openstack-aodh-api

Provides the main OpenStack Telemetry Alarming service API

openstack-aodh-common

Provides the files that are common to all OpenStack Telemetry Alarming service components

openstack-aodh-evaluator

Provides the alarm evaluator, which determines when alarms are to trigger

openstack-aodh-expirer

Provides the expirer component, which clears expired alarm history data

openstack-aodh-listener

Provides the listener daemon, which executes the defined actions

openstack-aodh-notifier

Provides the notifier daemon, which allows alarms to be set

python-aodh

Provides the OpenStack Telemetry Alarming service Python libraries

python-aodhclient

Provides the OpenStack Telemetry Alarming service command-line tool and a Python API (specifically, the aodhclient module).

Install all the required packages:

```
 # yum install openstack-aodh-api openstack-aodh-evaluator openstack-aodh-
expirer openstack-aodh-listener openstack-aodh-notifier python-aodhclient
```

This will install the *openstack-aodh-common* and *python-aodh* packages automatically because they are dependencies.

13.2. Create the Telemetry Alarming Service Database

Create the database and database user used by the Telemetry Alarming service. All steps in this procedure must be performed on the database server, while logged in as the `root` user.

Procedure 13.1. Creating the Telemetry Alarming Service Database

1. Connect to the database service:

```
# mysql -u root -p
```

2. Create the **aodh** database:

```
mysql> CREATE DATABASE aodh;
```

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

```
mysql> GRANT ALL ON aodh.* TO 'aodh'@'%' IDENTIFIED BY
'AODH_PASSWORD';
mysql> GRANT ALL ON aodh.* TO 'aodh'@'localhost' IDENTIFIED BY
'AODH_PASSWORD';
```

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

4. Exit the **mysql** client:

```
mysql> quit
```

13.3. Configure the Telemetry Alarming Service

You must configure the Telemetry Alarming service before you start the individual daemons.

13.3.1. Configure the Telemetry Alarming Service Database Connection

The database connection URL used by the Telemetry Alarming service is defined in the **/etc/aodh/aodh.conf** file. Set the URL by running the following command:

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

Replace *AODH_PASSWORD* with your Telemetry Alarming service password and *IP* with the IP address or host name of the server hosting the database service.

13.3.2. Create the Telemetry Alarming Service Identity Records

Create and configure Identity service records required by the Telemetry Alarming service. These entries assist other OpenStack services attempting to locate and access the functionality provided by the Telemetry Alarming service.

Procedure 13.2. Creating Identity Records for the Telemetry Alarming Service

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

```
# source ~/keystonerc_admin
```

2. Create the **aodh** user:

```
[(keystone_admin)]# openstack user create --password AODH_PASSWORD
aodh
+----------+----------------------------------+
| Field    | Value                            |
+----------+----------------------------------+
| email    | None                             |
| enabled  | True                             |
| id       | f55915b5ca2d451d8b4109251976a4bc |
| name     | aodh                             |
| username | aodh                             |
+----------+----------------------------------+
```

Replace *AODH_PASSWORD* with your Telemetry Alarming service password.

3. Add the **aodh** user to the **services** project, as a member of the **admin** role:

```
[(keystone_admin)]# openstack role add --project services --user aodh
admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 63aa6177a61b44aca25dd88a917353bc |
| name      | admin                            |
+-----------+----------------------------------+
```

4. Create the **aodh** service entity:

```
[(keystone_admin)]# openstack service create --name aodh \
--description "Telemetry Alarming Service" \
alarming
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Telemetry Alarming Service       |
| enabled     | True                             |
| id          | 67bb52266ae84c1f88877bbb4bf5d587 |
| name        | aodh                             |
| type        | alarming                         |
+-------------+----------------------------------+
```

5. Create the Telemetry Alarming service endpoint:

```
[(keystone_admin)]# openstack endpoint create \
--publicurl "http://IP:8042" \
--adminurl "http://IP:8042" \
--internalurl "http://IP:8042" \
--region RegionOne \
alarming
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| adminurl     | http://IP:8042                   |
| id           | ac2735777336400baa38e2d408d26392 |
| internalurl  | http://IP:8042                   |
| publicurl    | http://IP:8042                   |
| region       | RegionOne                        |
| service_id   | 67bb52266ae84c1f88877bbb4bf5d587 |
| service_name | aodh                             |
| service_type | alarming                         |
+--------------+----------------------------------+
```

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

13.3.3. Configure the Firewall to Allow Telemetry Alarming Service Traffic

The Telemetry Alarming service receives connections on port **8042**. The firewall on the service node must be configured to allow network traffic on this port. All steps in this procedure must be performed on the server hosting the Telemetry Alarming service, while logged in as the **root** user.

Procedure 13.3. Configuring the Firewall to Allow Telemetry Alarming Service Traffic

1. Open the **/etc/sysconfig/iptables** file in a text editor.

2. Add an INPUT rule allowing TCP traffic on port **8042**. The new rule must appear before any INPUT rules that REJECT traffic:

```
-A INPUT -p tcp -m multiport --dports 8042 -j ACCEPT
```

3. Save the changes to the **/etc/sysconfig/iptables** file.

4. Restart the **iptables** service to ensure that the change takes effect:

```
# systemctl restart iptables.service
```

13.3.4. Configure Telemetry Alarming Service Authentication

Configure the Telemetry Alarming service to use the Identity service for authentication. All steps in this procedure must be performed on each server hosting Telemetry Alarming services, while logged in as the **root** user.

Procedure 13.4. Configuring the Telemetry Alarming Service to Authenticate Through the Identity Service

1. Configure service credentials:

```
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials auth_type password
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials auth_url = http://CONTROLLER:5000/v3
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials interface internalURL
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials password AODH_PASSWORD
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials project_domain_name default
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials project_name service
# openstack-config --set /etc/aodh/aodh.conf \
service_credentials region_name RegionOne
 #  openstack-config --set /etc/aodh/aodh.conf \
service_credentials user_domain_name default
 #  openstack-config --set /etc/aodh/aodh.conf \
service_credentials username aodh
```

Replace *CONTROLLER* with the host name or IP address of your Identity server and *AODH_PASSWORD* with your Telemetry Alarming service password.

2. Configure access to the Identity service:

```
 #  openstack-config --set /etc/aodh/aodh.conf \
DEFAULT auth_strategy keystone
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken auth_type password
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken auth_uri http://CONTROLLER:5000
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken auth_url http://CONTROLLER:35357
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken memcached_servers CONTROLLER:11211
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken password AODH_PASSWORD
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken project_domain_name default
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken project_name service
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken user_domain_name default
 #  openstack-config --set /etc/aodh/aodh.conf \
keystone_authtoken username aodh
```

Again, replace *CONTROLLER* with the host name or IP address of your Identity server and *AODH_PASSWORD* with your Telemetry Alarming service password.

13.3.5. Configure RabbitMQ Message Broker Settings for the Telemetry Alarming Service

Configure access to the RabbitMQ message queue:

```
 #  openstack-config --set /etc/aodh/aodh.conf \
DEFAULT rpc_backend rabbit
 #  openstack-config --set /etc/aodh/aodh.conf \
oslo_messaging_rabbit rabbit_host CONTROLLER
 #  openstack-config --set /etc/aodh/aodh.conf \
oslo_messaging_rabbit rabbit_userid USER
 #  openstack-config --set /etc/aodh/aodh.conf \
oslo_messaging_rabbit rabbit_password RABBITMQ_PASSWORD
```

Replace *CONTROLLER* with the host name or IP address of your RabbitMQ broker, *USER* with your RabbitMQ user ID, and *RABBITMQ_PASSWORD* with your RabbitMQ password.

13.4. Launch the Telemetry Alarming Service

Start the Telemetry Alarming service daemons and configure them to start at boot time:

```
 # systemctl start openstack-aodh-api.service \
openstack-aodh-evaluator.service \
openstack-aodh-notifier.service \
openstack-aodh-listener.service
 # systemctl enable openstack-aodh-api.service \
openstack-aodh-evaluator.service \
openstack-aodh-notifier.service \
openstack-aodh-listener.service
```

Chapter 14. Install Time-Series-Database-as-a-Service

Time-Series-Database-as-a-Service (**`gnocchi`**) is a multi-tenant, metrics and resource database. It is designed to store metrics at a very large scale while providing access to metrics and resources information to operators and users.

For more information on using the Time-Series-Database-as-a-Service, see the Using the Time-Series-Database-as-a-Service section in the *Logging, Monitoring, and Troubleshooting Guide*.

Time-Series-as-a-Service is built around the following drivers:

storage

The **`storage`** driver is responsible for storing measures of created metrics. It receives timestamps and values and computes aggregations according to the defined archive policies.

indexer

The **`indexer`** driver is responsible for storing the index of all resources, along with their types and their properties. Time-Series-as-a-Service only knows resource types from the OpenStack project, but also provides a generic type so you can create basic resources and handle the resource properties yourself. The **`indexer`** is also responsible for linking resources with metrics.

The REST API exposed to the user manipulates both these drivers to provide all the features that are needed to provide correct infrastructure measurement.

14.1. Install the Time-Series-Database-as-a-Service Packages

The following packages provide the components of the Time-Series-Database-as-a-Service:

openstack-gnocchi-api

Provides the main OpenStack Time-Series-Database-as-a-Service API

openstack-gnocchi-carbonara

Provides the OpenStack Time-Series-Database-as-a-Service carbonara

openstack-gnocchi-doc

Provides the OpenStack Time-Series-Database-as-a-Service documentation

openstack-gnocchi-indexer-sqlalchemy

Provides the OpenStack Time-Series-Database-as-a-Service indexer SQLAlchemy

openstack-gnocchi-statsd

Provides the OpenStack Time-Series-Database-as-a-Service stats deamon

python-gnocchi

Provides the OpenStack Time-Series-Database-as-a-Service Python libraries

Install all the packages on the controller node:

```
# yum install openstack-gnocchi\* -y
```

14.2. Initialize Time-Series-Database-as-a-Service

Initialize the **indexer**:

```
# gnocchi-upgrade
```

14.3. Configure Time-Series-Database-as-a-Service

When manually installing the Time-Series-Database-as-a-Service packages, the service's configuration file (namely, **/etc/gnocchi/gnocchi.conf**) will have no settings configured. You will need to manually add and configure each setting as required.

- In the **[DEFAULT]** section, enable logging and verbose output:

```
[DEFAULT]
debug = true
verbose = true
```

- In the **[API]** section, list the number of workers:

```
[api]
workers = 1
```

- In the **[database]** section, set backend to **sqlalchemy**:

```
[database]
backend = sqlalchemy
```

- In the **[indexer]** section, configure the SQL database by passing the user name, password, and the IP address:

```
[indexer]
url = mysql://USER_NAME:PASSWORD@192.0.2.10/gnocchi2?charset=utf8
```

Note

The database has to be created before starting **gnocchi-api**

- In the **[keystone_authtoken]** section, update the authentication parameters. For example:

```
[keystone_authtoken]
auth_uri = http://192.0.2.7:5000/v2.0
signing_dir = /var/cache/gnocchi
auth_host = 192.0.2.7
auth_port = 35357
auth_protocol = http
```

```
identity_uri = http://192.0.2.7:35357/
admin_user = admin
admin_password = 5179f4d3c5b1a4c51269cad2a23dbf336513efeb
admin_tenant_name = admin
```

- In the **`[statsd]`** section, include the following parameter values:

```
[statsd]
resource_id = RESOURCE_ID
user_id = USER_ID
project_id = PROJECT_ID
archive_policy_name = low
flush_delay = 5
```

  Replace the values for **`RESOURCE_ID`**, **`USER_ID`**, and **`PROJECT_ID`** with values for your deployment.

- In the **`[storage]`** section, manually add the **`coordination_url`** and **`file_basepath`** and set the **`driver`** value to file:

```
[storage]
coordination_url = file:///var/lib/gnocchi/locks
driver = file
file_basepath = /var/lib/gnocchi
```

- Restart the **`gnocchi`** service to ensure that the change takes effect:

```
# systemctl restart openstack-gnocchi-api.service
# systemctl restart openstack-gnocchi-metricd.service
# systemctl restart openstack-gnocchi-statsd.service
```

14.4. Create Time-Series-Database-as-a-Service Database

Create the database and database user used by the Time-Series-Database-as-a-Service service. All steps in this procedure must be performed on the database server, while logged in as the **`root`** user.

- Connect to the database service:

```
# mysql -u root -p
```

- Create the Time-Series-Database-as-a-Service database:

```
mysql> CREATE DATABASE gnocchi;
```

- Create a Time-Series-Database-as-a-Service database user and grant the user access to the Time-Series-Database-as-a-Service database:

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

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

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

  ```
  mysql> FLUSH PRIVILEGES;
  ```

- Exit the **mysql** client:

  ```
  mysql> quit
  ```

14.5. Set Time-Series-Database-as-a-Service as the Backend for Telemetry Service

Telemetry service, by default only saves metering data in a database. To allow Telemetry to send metering data to other systems in addition to the database, multiple dispatchers can be developed and enabled by modifying the Telemetry configuration file. The **gnocchi** dispatcher posts the meters onto TDSaaS backend.

For the **gnocchi** dispatcher, add the following configuration settings to the **/etc/ceilometer/ceilometer.conf** file:

```
[DEFAULT]
dispatcher = gnocchi

[dispatcher_gnocchi]
filter_project = gnocchi_swift
filter_service_activity = True
archive_policy = low
url = http://localhost:8041
```

The **url** in the above configuration is a TDSaaS endpoint URL and depends on your deployment.

Note

If the **gnocchi** dispatcher is enabled, Ceilometer API calls will return a **410**, with an empty result. The TDSaaS API should be used instead to access the data.

Restart the **gnocchi** service to ensure that the change takes effect:

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

Chapter 15. Install the Shared File System Service

OpenStack's Shared File System service provides the means to easily provision shared file systems that can be consumed by multiple instances. These shared file systems are provisioned from pre-existing back end resources.

15.1. Shared File System Service Back End Requirements

The OpenStack Shared File System service allows you to create shared file systems on demand. However, the back end resources for the shares must already exist. This chapter describes how to deploy the service with NetApp's unified **`manila`** driver (**`manila.share.drivers.netapp.common.NetAppDriver`**).

15.2. Install the Shared File System Service Packages

The following packages provide the components of the Shared File System service:

openstack-manila

Provides the main OpenStack Shared File System service.

openstack-manila-share

Provides the service necessary for exporting provisioned shares.

python-manilaclient

Provides the client library and CLI for the Shared File System service

Install the packages on the Controller node:

```
 #  yum install -y openstack-manila openstack-manila-share python-
manilaclient
```

15.3. Create the Shared File System Service Identity Records

After installing the necessary packages, create the Identity records required for the Shared File System service. Perform the following procedure on the Identity service host, or on any machine onto which you have copied the **`keystonerc_admin`** file.

Note

For more details about the **`keystonerc_admin`** file, see Section 3.5, "Create an Administrator Account and the Identity Service Endpoint".

Procedure 15.1. Creating Identity Records for the Shared File System Service

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

```
#  source ~/keystonerc_admin
```

2. Create the **manila** service user:

   ```
    [(keystone_admin)]# openstack user create --password MANILAPASS --
   email manila@localhost manila
   ```

   Replace *MANILAPASS* with a password that will be used by the Shared File System service when authenticating with the Identity service.

3. Add the **admin** role to the **manila** user.

   ```
    [(keystone_admin)]# openstack role add --project services --user
   manila admin
   ```

4. Create the **manila** service entities:

   ```
    [(keystone_admin)]# openstack service create --name manila --
   description "OpenStack Shared Filesystems" share
   ```

5. Create the **manila** endpoint entry:

   ```
    [(keystone_admin)]# openstack endpoint create \
   --publicurl 'http://MANILAIP:8786/v1/%(tenant_id)s' \
   --internalurl 'http://MANILAIP:8786/v1/%(tenant_id)s' \
   --adminurl 'http://MANILAIP:8786/v1/%(tenant_id)s' \
   --region RegionOne \
   manila
   ```

   Replace *MANILAIP* with the IP of the Controller node.

15.4. Configure Basic Shared File System Service Settings

When manually installing the Shared File System service packages, the service's configuration file (namely, **/etc/manila/manila.conf**) will have no settings configured. You will need to manually uncomment/add and configure each setting as required.

The following code snippet is the basic configuration required for deploying the Shared File System service. You can copy its contents to **/etc/manila/manila.conf**, replacing the necessary variables when you do:

```
[DEFAULT]

osapi_share_listen=0.0.0.0
sql_connection=mysql://manila:MANILADBPASS@CONTROLLERIP/manila # (1) (2)
api_paste_config=/etc/manila/api-paste.ini
state_path=/var/lib/manila
sql_idle_timeout=3600
storage_availability_zone=nova
rootwrap_config=/etc/manila/rootwrap.conf
auth_strategy=keystone
nova_catalog_info=compute:nova:publicURL
nova_catalog_admin_info=compute:nova:adminURL
nova_api_insecure=False
nova_admin_username=nova
```

```
nova_admin_password=NOVAADMINPASS # 3
nova_admin_tenant_name=services
nova_admin_auth_url=http://localhost:5000/v2.0
network_api_class=manila.network.neutron.neutron_network_plugin.NeutronNetwo
rkPlugin
debug=False
verbose=True
log_dir=/var/log/manila
use_syslog=False
rpc_backend=manila.openstack.common.rpc.impl_kombu
control_exchange=openstack
amqp_durable_queues=False

[oslo_messaging_rabbit]
rabbit_ha_queues=False
rabbit_userid=guest
rabbit_password=guest
rabbit_port=5672
rabbit_use_ssl=False
rabbit_virtual_host=/
rabbit_host=CONTROLLERIP # 4
rabbit_hosts=CONTROLLERIP:5672 # 5

[oslo_concurrency]
lock_path=/tmp/manila/manila_locks
```

Replace the following values:

1 *MANILADBPASS* is the database password of the Shared File System service, which you used in Section 15.5, “Create the Shared File System Service Database”.

2 *CONTROLLERIP* is the IP address of the Controller node.

4

5

3 *NOVAADMINPASS* is the admin password of the Compute service. This is identical to the value of **`nova_admin_password`** in **`/etc/neutron/neutron.conf`**.

Note

If you deployed OpenStack using the Director, you can also find this password in the undercloud's **`/home/stack/tripleo-overcloud-passwords`** file.

As of this release, some Shared File System service settings are still defined in **`/etc/manila/api-paste.ini`**. Update this file with the following code snippet:

```
[filter:keystonecontext]
paste.filter_factory =
manila.api.middleware.auth:ManilaKeystoneContext.factory
```

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
service_protocol = http
service_host = localhost
service_port = 5000
auth_host = localhost
auth_port = 35357
auth_protocol = http
admin_tenant_name = services
admin_user = manila
admin_password = MANILAPASS # (1)
signing_dir = /var/lib/manila
auth_uri=http://CONTROLLERIP:5000/v2.0 # (2)
identity_uri=http://CONTROLLERIP:35357 # (3)
```

(1) *MANILAPASS* is the admin password of the **manila** service user (which you used in Section 15.3, "Create the Shared File System Service Identity Records").

(2) *CONTROLLERIP* is the IP address of the Controller node.

(3)

15.5. Create the Shared File System Service Database

Create a database and database user for the Shared File System service. All steps must be performed on the database server, while logged in as the **root** user.

Procedure 15.2. Creating the Shared File System Service Database

1. Connect to the database service:

   ```
   # mysql -u root
   ```

2. Create the **manila** database:

   ```
   mysql> CREATE DATABASE manila;
   ```

3. Create a **manila** database user and grant the user access to the**manila** database:

   ```
   mysql> GRANT ALL ON manila.* TO 'manila'@'%' IDENTIFIED BY
   'MANILADBPASS';
   mysql> GRANT ALL ON manila.* TO 'manila'@'localhost' IDENTIFIED BY
   'MANILADBPASS';
   ```

   Replace *MANILADBPASS* with a secure password that the **manila** service can use to authenticate with the database server. You will use this same password later in Section 15.4, "Configure Basic Shared File System Service Settings".

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

```
mysql> FLUSH PRIVILEGES;
```

5. Exit the **mysql** client:

```
mysql> quit
```

6. Create the Shared File System service tables and apply all necessary migrations:

```
# manila-manage db sync
```

15.6. Define the Shared File System Service Back End

The Shared File System service requires a back end. The back end is defined in its own section in **/etc/manila/manila.conf**. NetApp provides detailed information on how to define the Shared File System service back end. See OpenStack Shared File System Service (Manila) from NetApp's OpenStack Deployment and Operations Guide.

15.7. Launch the Shared File System Service

At this point, the Shared File System service should be fully configured. To apply the required settings, restart all the Shared File System services:

```
# systemctl start openstack-manila-api
# systemctl start openstack-manila-share
# systemctl start openstack-manila-scheduler
```

Afterwards, enable them as well:

```
# systemctl enable openstack-manila-api
# systemctl enable openstack-manila-share
# systemctl enable openstack-manila-scheduler
```

To check whether each service were launched and enabled successfully, run:

```
# systemctl status openstack-manila-api
openstack-manila-api.service - OpenStack Manila API Server
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-api.service;
enabled)
Active: active (running) since Mon 2015-07-27 17:02:49 AEST; 1 day 18h ago
[...]

# systemctl status openstack-manila-share
openstack-manila-share.service - OpenStack Manila Share Service
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-share.service;
enabled)
Active: active (running) since Mon 2015-07-27 17:02:49 AEST; 1 day 18h ago
[...]

# systemctl status openstack-manila-scheduler
openstack-manila-scheduler.service - OpenStack Manila Scheduler
```

```
Loaded: loaded (/usr/lib/systemd/system/openstack-manila-scheduler.service;
enabled)
Active: active (running) since Mon 2015-07-27 17:02:49 AEST; 1 day 18h ago
[...]
```

15.8. Create a Share Type for the Defined Back End

The Shared File System service allows you to define *share types* that you can use to create shares with specific settings. Share types work exactly like Block Storage volume types: each type has associated settings (namely, *extra specifications*), and invoking the type during share creation applies those settings.

When creating a share on a non-default back end, you need to explicitly specify which back end to use. To make the process seamless for users, create a share type and associate it with the **share_backend_name** value of your back end (whichever you chose in Section 15.6, “Define the Shared File System Service Back End”).

To create a share type named *TYPENAME*, run the following as an OpenStack admin:

```
# manila type-create TYPENAME SHAREHANDLING
```

SHAREHANDLING specifies whether or not the share type will use the driver to handle the life cycle of shares. This should be the value set in **driver_handles_share_servers** of your back end definition. With **driver_handles_share_servers=False**, *SHAREHANDLING* should also be **false**. So, to create a share type called **share1**:

```
# manila type-create share1 false
```

Next, associate the **share1** type to a specific back end. You can specify the back end through its **share_backend_name** value. For example, to associate the share type **share1** to a back end named **GENERIC**, run:

```
# manila type-key share1 set share_backend_name='GENERIC'
```

Users should now be able to invoke the **SHARE1** type to create a share from the **GENERIC** back end.

Chapter 16. Install Database-as-a-Service (Technology Preview)

The OpenStack Database-as-a-Service (trove) allows users to easily provision single-tenant databases, and bypass much of the traditional administrative overhead involved in deploying, using, managing, monitoring, and scaling databases.

Warning

DEPRECATION NOTICE: Beginning in Red Hat OpenStack Platform 10, the OpenStack Trove service will no longer be included in the Red Hat OpenStack Platform distribution. We are working with a trusted partner to provide our customers with a production ready DBaaS service. Please contact your sales account manager to learn more about this option.

Warning

The OpenStack Database-as-a-Service is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

16.1. Database-as-a-Service Requirements

The following steps are requirements for using the Database-as-a-Service:

1. Update the admin user's password:

```
 # keystone user-password-update --pass ADMIN_PASSWORD admin
```

2. Update the `/root/keystonerc_admin` file with the new password:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_PASSWORD=ADMIN_PASSWORD
export OS_AUTH_URL=http://keystone IP:5000/v2.0/
export PS1='[\u@\h \W(keystone_admin)]\$ '
```

3. Load the environment variables and make sure the **admin** user has the **admin** role in the **services** tenant:

```
 # source keystonerc_admin
 ~(keystone_admin)]# keystone user-role-add --user admin --tenant
services --role admin
 ~(keystone_admin)]# keystone user-role-list --user admin --tenant
services
+----------------------------------+-------+----------------------
----------+----------------------------------+
|                id                | name |           user_id
|            tenant_id             |
+----------------------------------+-------+----------------------
```

```
-----------+----------------------------------+
| 4501ce8328324ef5bf1ed93ceb5494e6 | admin |
4db867e819ad40e4bf79681bae269084 | 70cd02c84f86471b8dd934db46fb484f |
+----------------------------------+-------+-----------------------
-----------+----------------------------------+
```

16.2. Install the Database-as-a-Service Packages

The following packages provide the components of the Database-as-a-Service:

openstack-trove-api

Provides the main OpenStack Database-as-a-Service API.

openstack-trove-conductor

Provides the OpenStack Database-as-a-Service conductor service.

openstack-trove-guestagent

Provides the OpenStack Database-as-a-Service guest agent service.

openstack-trove-taskmanager

Provides the OpenStack Database-as-a-Service task manager service.

openstack-trove-images

Provides the OpenStack Database-as-a-Service image creation tool.

python-trove

Provides the OpenStack Database-as-a-Service Python library.

python-troveclient

Provides a client for the Database-as-a-Service API.

Install all the Database-as-a-Service packages on the controller node:

```
# yum install openstack-trove\*
```

16.3. Configure Database-as-a-Service

1. Create a keystone user and add role for the Database-as-a-Service:

```
 [root@rhosp-trove ~(keystone_admin)]# openstack user create --
password trove --email trove@localhost --project services trove
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| email      | trove@localhost                  |
| enabled    | True                             |
| id         | 8740fd0cba314fe68cf0ca95144d2766 |
| name       | trove                            |
| project_id | 42e1efb4bd5e49a49cb2b346078d6325 |
| username   | trove                            |
+------------+----------------------------------+
 [root@rhosp-trove ~(keystone_admin)]# openstack role add --project
services --user trove admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 63aa6177a61b44aca25dd88a917353bc |
| name      | admin                            |
+-----------+----------------------------------+
 [root@rhosp-trove ~(keystone_admin)]# openstack user role list --
project services trove
+----------------------------------+----------+----------+-------+
| ID                               | Name     | Project  | User  |
+----------------------------------+----------+----------+-------+
| 63aa6177a61b44aca25dd88a917353bc | admin    | services | trove |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ | services | trove |
+----------------------------------+----------+----------+-------+
```

2. Optionally, set up verbose debug information in all configuration files:

```
 [root@rhosp-trove ~(keystone_admin)]# for conf_file in {trove,trove-
conductor,trove-taskmanager,trove-guestagent}; do
> openstack-config --set /etc/trove/$conf_file.conf DEFAULT verbose
True;
> openstack-config --set /etc/trove/$conf_file.conf DEFAULT debug
True;
> done
```

3. Create the **`api-paste.ini`** file (if not present):

```
 [root@rhosp-trove ~(keystone_admin)]# cp /usr/share/trove/trove-dist-
paste.ini /etc/trove/api-paste.ini
```

4. Update keystone authtoken in **`api-paste.ini`**:

```
[filter:authtoken]
paste.filter_factory =
keystoneclient.middleware.auth_token:filter_factory
auth_uri = http://127.0.0.1:35357/
identity_uri = http://127.0.0.1:35357/
admin_password = TROVE_PASSWORD
admin_user = trove
admin_tenant_name = services
```

```
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove.conf DEFAULT api_paste_config /etc/trove/api-paste.ini
```

5. Update **`trove.conf`** with the same information as **`api-paste.ini`**:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove.conf keystone_authtoken auth_uri http://127.0.0.1:35357/
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
```

```
trove.conf keystone_authtoken identity_uri http://127.0.0.1:35357/
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove.conf keystone_authtoken admin_password TROVE_PASSWORD
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove.conf keystone_authtoken admin_user trove
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove.conf keystone_authtoken admin_tenant_name = services
```

6. Set up **nova_proxy** information in **trove-taskmanager.conf**. This needs to be the actual admin user as the Database-as-a-Service will use this user's credentials to issue nova commands:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove-taskmanager.conf DEFAULT nova_proxy_admin_user admin
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove-taskmanager.conf DEFAULT nova_proxy_admin_password
ADMIN_PASSWORD
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
trove-taskmanager.conf DEFAULT nova_proxy_admin_tenant_name services
```

7. Update the configuration files with RabbitMQ host information:

```
 [root@rhosp-trove trove(keystone_admin)]# cat
/etc/rabbitmq/rabbitmq.config
% This file managed by Puppet
% Template Path: rabbitmq/templates/rabbitmq.config
[
  {rabbit, [
    {default_user, <<"guest">>},
    {default_pass, <<"RABBITMQ_GUEST_PASSWORD">>}
  ]},
```

```
 [root@rhosp-trove trove(keystone_admin)]# for conf_file in trove.conf
trove-taskmanager.conf trove-conductor.conf ; do
> openstack-config --set /etc/trove/$conf_file DEFAULT rabbit_host
127.0.0.1;
> openstack-config --set /etc/trove/$conf_file DEFAULT rabbit_password
RABBITMQ_GUEST_PASSWORD;
> done
```

8. Add service URLs to all the configuration files:

```
 [root@rhosp-trove trove(keystone_admin)]# for conf_file in trove.conf
trove-taskmanager.conf trove-conductor.conf ; do
> openstack-config --set /etc/trove/$conf_file DEFAULT trove_auth_url
http://127.0.0.1:5000/v2.0
> openstack-config --set /etc/trove/$conf_file DEFAULT
nova_compute_url http://127.0.0.1:8774/v2
> openstack-config --set /etc/trove/$conf_file DEFAULT cinder_url
http://127.0.0.1:8776/v1
> openstack-config --set /etc/trove/$conf_file DEFAULT swift_url
http://127.0.0.1:8080/v1/AUTH_
> openstack-config --set /etc/trove/$conf_file DEFAULT sql_connection
```

```
mysql://trove:trove@127.0.0.1/trove
> openstack-config --set /etc/trove/$conf_file DEFAULT
notifier_queue_hostname 127.0.0.1
> done
```

Note that the commands above add a MySQL connection that does not work yet; those permissions are added next.

9. Update the task manager configuration with **cloud-init** information:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
/etc/trove/trove-taskmanager.conf DEFAULT cloud-init_loaction
/etc/trove/cloudinit
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
/etc/trove/trove-taskmanager.conf DEFAULT taskmanager_manager
trove.taskmanager.manager.Manager
 [root@rhosp-trove trove(keystone_admin)]# mkdir /etc/trove/cloudinit
```

10. Update `trove.conf` with the default datastore (database type), and set the name of the OpenStack Networking network to which instances will be attached. In this case, that network was named `private`:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
/etc/trove/trove.conf DEFAULT default_datastore mysql
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
/etc/trove/trove.conf DEFAULT add_addresses True
 [root@rhosp-trove trove(keystone_admin)]# openstack-config --set
/etc/trove/trove.conf DEFAULT network_label_regex ^private$
```

11. Create the Database-as-a-Service database and add permissions for the `trove` user:

```
 [root@rhosp-trove trove(keystone_admin)]# mysql -u root
MariaDB [(none)]> create database trove;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on trove.* to trove@'localhost' identified
by 'TROVE_PASSWORD';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> grant all on trove.* to trove@'%' identified by
'TROVE_PASSWORD';
Query OK, 0 rows affected (0.00 sec)
```

12. Populate the new database and create the initial datastore:

```
 [root@rhosp-trove trove(keystone_admin)]# trove-manage db_sync
 [root@rhosp-trove trove(keystone_admin)]# trove-manage
datastore_update mysql ''
```

13. Create the **cloud-init** file that will be used with an image.

Note

When an instance is created by the Database-as-a-Service, it will use whatever **`image_id`** you have set in the database to build the instance. Additionally, based on the datastore specified, it will also now look in **`/etc/trove/cloudinit/`** for a **`.cloudinit`** file to attach as user data. For example, if you choose **`mysql`** as the datastore for a new instance, nova will look for a **`mysql.cloudinit`** file in **`/etc/trove/cloudinit/`** to attach as a user-data script. This is used to register and install MySQL at build time.

Create the **`/etc/trove/cloudinit/mysql.cloudinit`** file with the following content, replacing each occurrence of *PASSWORD* with a suitable password, *RHN_USERNAME*, *RHN_PASSWORD* and *POOL_ID* with your Red Hat credentials and subscription pool ID, and *host SSH public key* with the key for passwordless SSH login:

```
#!/bin/bash

sed -i'.orig' -e's/without-password/yes/' /etc/ssh/sshd_config
echo "PASSWORD" | passwd --stdin cloud-user
echo "PASSWORD" | passwd --stdin root
systemctl restart sshd

subscription-manager register --username=RHN_USERNAME --
password=RHN_PASSWORD
subscription-manager attach --pool POOL_ID
subscription-manager repos --disable=*
subscription-manager repos --enable=rhel-7-server-optional-rpms
subscription-manager repos --enable=rhel-7-server-rpms
subscription-manager repos --enable=rhel-server-rhscl-7-rpms
yum install -y openstack-trove-guestagent mysql55

cat << EOF > /etc/trove/trove-guestagent.conf
rabbit_host = 172.1.0.12
rabbit_password = RABBITMQ_GUEST_PASSWORD
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASSWORD
nova_proxy_admin_tenant_name = services
trove_auth_url = http://172.1.0.12:35357/v2.0
control_exchange = trove
EOF

echo "host SSH public key" >> /root/.ssh/authorized_keys

echo "host SSH public key" >> /home/cloud-user/.ssh/authorized_keys

systemctl stop trove-guestagent
systemctl enable trove-guestagent
systemctl start trove-guestagent
```

Note

The above is written as a bash script, which is supported by **`cloud-init`**. This can also be done using **`cloud-init`**'s YAML-style layout.

14. Upload a cloud image, specified as the parameter of the **`--file`** option, using glance:

```
 [root@rhosp-trove trove(keystone_admin)]# glance image-create --name
rhel7 \
> --file image.qcow2 \
> --disk_format qcow2 \
> --container_format bare \
> --is-public True \
> --owner trove

 [root@rhosp-trove trove(keystone_admin)]# glance image-list
+--------------------------------------+--------+-------------+----
--------------+-----------+--------+
| ID                                   | Name   | Disk Format |
Container Format | Size      | Status |
+--------------------------------------+--------+-------------+----
--------------+-----------+--------+
| b88fa633-7219-4b80-87fa-300840575f91 | cirros | qcow2       | bare
| 13147648  | active |
| 9bd48cdf-52b4-4463-8ce7-ce81f44205ae | rhel7  | qcow2       | bare
| 435639808 | active |
+--------------------------------------+--------+-------------+----
--------------+-----------+--------+
```

15. Update the Database-as-a-Service database with a reference to the Red Hat Enterprise Linux 7 image; use the ID from the output of the previous command:

```
 [root@rhosp-trove trove(keystone_admin)]# trove-manage --config-
file=/etc/trove/trove.conf datastore_version_update \
> mysql mysql-5.5 mysql 9bd48cdf-52b4-4463-8ce7-ce81f44205ae mysql55 1
```

Note

The syntax is: **`trove-manage datastore_version_update`** ***`datastore version_name manager image_id packages active`***

16. Create the Database-as-a-Service service using keystone to make OpenStack aware of its presence:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack service create --
name trove \
> --description "OpenStack DBaaS" \
> database
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | OpenStack DBaaS                  |
| enabled     | True                             |
| id          | b05b564d5ac049f49984a827d820c5a5 |
| name        | trove                            |
| type        | database                         |
+-------------+----------------------------------+
```

17. Add URL endpoints for the Database-as-a-Service API:

```
 [root@rhosp-trove trove(keystone_admin)]# openstack endpoint create \
> --publicurl 'http://127.0.0.1:8779/v1.0/%(tenant_id)s' \
> --internalurl 'http://127.0.0.1:8779/v1.0/%(tenant_id)s' \
> --adminurl 'http://127.0.0.1:8779/v1.0/%(tenant_id)s' \
> --region RegionOne \
> database
```

18. Start the three Database-as-a-Service services and enable them to start at boot:

```
 [root@rhosp-trove trove(keystone_admin)]# systemctl start openstack-
trove-{api,taskmanager,conductor}
 [root@rhosp-trove trove(keystone_admin)]# systemctl enable openstack-
trove-{api,taskmanager,conductor}
ln -s '/usr/lib/systemd/system/openstack-trove-api.service'
'/etc/systemd/system/multi-user.target.wants/openstack-trove-
api.service'
ln -s '/usr/lib/systemd/system/openstack-trove-taskmanager.service'
'/etc/systemd/system/multi-user.target.wants/openstack-trove-
taskmanager.service'
ln -s '/usr/lib/systemd/system/openstack-trove-conductor.service'
'/etc/systemd/system/multi-user.target.wants/openstack-trove-
conductor.service'
```

Important

Run **`systemctl status openstack-trove-{api,taskmanager,conductor}`** to make sure these services have started properly. If they have failed due to an error with **`/var/log/trove`**, you can run these commands to solve the issue:

```
 [root@rhosp-trove trove(keystone_admin)]# chown -R trove:trove
/var/log/trove
 [root@rhosp-trove trove(keystone_admin)]# systemctl restart
openstack-trove-{api,taskmanager,conductor}
```

Chapter 17. Install OpenStack Integration Test Suite

OpenStack Integration Test Suite (tempest) is a set of integration tests that are to be run against a live OpenStack cluster. The Integration Test Suite has a set of tests for OpenStack API validation, scenarios and other specific tests useful in validating your Red Hat OpenStack Platform deployment.

To run the OpenStack Integration Test Suite, you need to first install the necessary packages and create a configuration file that will tell the Integration Test Suite where to find the various OpenStack services and other testing behaviour switches. The location of this configuration file and how you interact with it, will determine how you run the Integration Test Suite.

There are two menthods of using the Integration Test Suite:

- In this method, the Integration Test Suite runs as a system installed program. This is the newer of the two methods and Red Hat recommends that you use this method while using the test suite.
- In the second method, the Integration Test Suite assumes your working directory is the actual test suite source repo and some assumptions associated with that.

17.1. Install the OpenStack Integration Test Suite Packages

Before you begin:

- On the controller node, as a **root** user, create a virtual machine titled **tempest** using the **virt-manager** that installs Red Hat Enterprise Linux 7.2. For more information, see Creating Guests with Virt-Manager
- Also, before installing the OpenStack Integration Test Suite, create the following networks within the Red Hat OpenStack Platfor environment. The OpenStack Integration Test Suite requires one of the networks to be considered **external**:

```
 #  neutron net-create private --shared
 #  neutron subnet-create private PRIVATE_NETWORK_ADDRESS MASK
 #  neutron router-create router
 #  neutron router-interface-add ROUTER_ID PRIVATE_SUBNET_ID
 #  neutron net-create public --router:external --provider:network_type
flat
 #  neutron subnet-create public --allocation-pool
start=START_IP_ADDRESS,end=END_IP_ADDRESS --gateway=DEFAULT_GATEWAY --
enable_dhcp=False PUBLIC_NETWORK_ADDRESS/MASK
 #  neutron router-gateway-set ROUTER_ID PUBLICH_NETWORK_ID
```

The OpenStack Integration Test Suite contains a list of Design Principles for validating the OpenStack Cloud. The main objective of OpenStack Integration Test Suite is to validate any OpenStack installation by explicitly testing OpenStack scenarios using public interfaces to determine whether the OpenStack cloud is running as intended. The following sections provide detailed steps in installing and configuring the OpenStack Integration Test Suite within the **tempest** virtual machine and showing how to run different OpenStack scenarios.

Procedure 17.1. Installing the Integration Test Suite

1. Install the packages listed above related to the OpenStack Integration Test Suite:

```
 #  yum install openstack-tempest
```

However, the command will not install any tempest plug-ins. These have to be installed manually, depending on your OpenStack installation.

2. The following command will show you all OpenStack components installed on your machine.

```
# openstack-status
```

3. For each component you can install an appropriate tempest plug-in, for example:

```
# yum install python-glance-tests python-keystone-tests python-
horizon-tests-tempest python-neutron-tests python-cinder-tests python-
nova-tests python-swift-tests python-ceilometer-tests python-gnocchi-
tests python-aodh-tests
```

You can also use the following command that will find the installed OpenStack components automatically and install the required test packages.

```
# /usr/share/openstack-tempest-*/tools/install_test_packages.py
```

17.2. Configure the OpenStack Integration Test Suite

Within the **`tempest`** virtual machine, run the following steps to configure your environment:

1. Create a working directory to run the OpenStack Integration Test Suite tools, for example, **`mytempest`**:

```
# mkdir -p /path/to/mytempest
```

2. Within the **`mytempest`** directory, run the **`configure-tempest-directory`** command to setup the OpenStack Integration Test Suite within the **`mytempest`** directory.

```
# cd /path/to/mytempest
# /usr/share/openstack-tempest/tools/configure-tempest-directory
```

3. Export the Red Hat OpenStack Platform environment variable with the proper credentials. The credentials for this environment are as follows:

```
# export OS_USERNAME=admin
# export OS_TENANT_NAME=admin
# export OS_PASSWORD=
# export OS_AUTH_URL=http://IP:35357/v2
```

Note

It is not necessary to have admin credetials to run the OpenStack Intergration Test Suite. With **`admin`** credentials, you can perform the following actions:

- Run tests for admin APIs
- Generate test credentials on the go.

4. With a saved resource state, run the **config_tempest.py** script with the following credentials to properly create the **tempest.conf** configuration file that resides within the **/etc** directory.

```
# tools/config_tempest.py --debug --create identity.uri $OS_AUTH_URL
identity.admin_username $OS_USERNAME identity.admin_password
$OS_PASSWORD identity.admin_tenant_name $OS_TENANT_NAME object-
storage.operator_role Member
```

5. Before you run any tests, it is critical to preserve a clean resouce state of the existing OpenStack cloud:

```
# python -m tempest.cmd.cleanup --init-saved-state
```

Note

Cleanup with the OpenStack Integration Test Suire does not cleanup the entire environment. It is possible that manual intervention is required to clean up the existing OpenStack cloud.

6. Review the **etc/tempest.conf** file located within the **mytempest** directory to ensure it meets all your Red Hat OpenStack Platform environment needs.

7. Verify the **run-test.sh** script properly runs, by using one of the **tempest** tests labeled **tempest.api.compute.flavors**.

```
# tools/run-tests.sh --concurrency 4 tempest.api.compute.flavors
```

Note

The option **concurrency 4** prevents the OpenStack Integration Test Suite from creating race conditions and/or unexpected errors.

8. After successful verification, run the **run-tests.sh** scripts as follows:

```
# tools/run-tests.sh --concurrency 4 --skip-file tools/ra-skip-file |
tee ra-out.txt
```

Note

The **`ra-skip-file`** contains specific tempest tests excluded when running the **`run-tests.sh`** script. The reason specific tests are excluded is due to your environment not running certain scenarios, for example, floating IP tests and third party tests. A sample **`ra-skip file`** can be as follows:

```
-tempest\.api\.compute\.floating_ips.* -
tempest\.thirdparty\.boto.*
-tempest\.api\.compute\.floating_ips.* -
tempest\.api\.network\.test_floating_ips.*
-tempest\.api\.network\.admin\.test_floating_ips_admin_action
```

Every Red Hat OpenStack Environment is different and therefore the test scenario cases to skip may vary for each environment.

9. Once the OpenStack Integration Test Suite tests that verify your OpenStack deployment are complete, cleanup the environment to return it to its original resource state.

```
# python -m tempest.cmd.cleanup
```

Chapter 18. Install OpenStack Benchmarking Service

The OpenStack Benchmarking Service helps answer the question of "How does OpenStack work at scale?". It is able to achieve this up automating the process that includes the OpenStack deployment, cloud verification, benchmarking and profiling. While the Benchmarking service is able to offer an assortment of actions to test and validate an OpenStack cloud, this chapter focuses on installing and configuring it to work with your OpenStack deployment.

Note

The OpenStack Benchmarking service is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.

18.1. Install the OpenStack Benchmarking Service Packages

Before you Begin:

- As a **root** user, create a virtual machine titled **rally** using the **virt-manager** that installs Red Hat Enterprise Linux 7.2. For more information, see Creating a Red Hat Enterprise Linux 7 guest with virt-manager.
- The Benchmarking service virtual machine requires access to the Internal API network, the tenant network and the Controller network.

  Note

  For more information, see Creating a Red Hat Enterprise Linux 7 guest with virt-manager.

- Make sure you are subscribed to the necessary channels on the Benchmarking service virtual machine.

Procedure 18.1. Install the Benchmarking Service

1. On the Benchmarking service virtual machine, as a **root** user create a working directory:

   ```
   # mkdir /path/to/myrally
   ```

2. Clone the repository:

   ```
   # git clone https://github.com/redhat-openstack/rally.git
   ```

3. Navigate to the working directory:

   ```
   # cd /path/to/myrally/rally
   ```

4. As a **root** user, run the **install_rally.sh** script and follow the prompts to install the required dependencies.

```
# //path/to/myrally/rally/install_rally.sh

[ ... Output Abbreviated ...]
Installing rally-manage script to /usr/bin
Installing rally script to /usr/bin
======================================================================
Information about your Rally installation:
* Method: system
* Database at: /var/lib/rally/database
* Configuration file at: /etc/rally
===================================================
```

18.2. Configure the OpenStack Benchmarking Service

When you have successfully installed the Benchmarking service, you need to configure the Benchmarking service environment by providing the service access to your OpenStack deployment.

Procedure 18.2. Confifuring the OpenStack Benchmarking Service

1. Export the OpenStack environment variables with the proper credentials. The credentials are as follows:

```
# export OS_USERNAME=admin
# export OS_TENANT_NAME=admin
# export OS_PASSWORD=
# export OS_AUTH_URL=http://IP:35357/v2
```

2. Add the existing Red Hat OpenStack environment to the Benchmarking service database. Create a Benchmarking service deployment json file labeled **`osp-deployment.json`**:

```
{
  "type": "ExistingCloud",
    "auth_url": "http://DEPLOYMENT_IP:35357/v2.0/",
    "region_name": "RegionOne",
  "endpoint_type": "public",
    "admin": {
        "username": "admin",
        "password": "********",
        "tenant_name": "admin"
    },
    "users": [
        {
            "username": "admin",
            "password": "*****",
            "tenant_name": "admin"
        }
    ]
}
```

3. (Optional) If creating additional users outside of **`rally`** and adding them to the **`osp-deployment.json`** file, take a look at 1175432 for a known race condition.

4. Add the existing OpenStack deployment to the **`rally`** database. The following example names the entry *RA_Rally* and captures all the appropriate credentials required within the **`osp-`**

deployment.json file:

```
#  rally deployment create --name RA-Rally --file osp-deployment.json
--------------------------+-----------------------------+----------
+-----------------+--------+
| uuid                    | created_at                  | name      |
status           | active |
+-------------------------+-----------------------------+----------
+-----------------+--------+
| 496e3be9-6020-4543-82f6-e7eb90517153 | 2015-05-12 15:18:13.817669 |
RA-Rally | deploy->finished |     |
+-------------------------+-----------------------------+----------
+-----------------+--------+

Using deployment: 496e3be9-6020-4543-82f6-e7eb90517153
~/.rally/openrc was updated
[ ... Output Abbreviated ... ]
```

18.3. Extend the Tenant Network for the Benchmarking Service

The following procedure is **optional** and required only if the Benchmarking service scenario being run requires a direct access (**ssh**) to the guest. When running the **NovaServers.boot_server** scenario, extending the tenant network is not required as this is specific scenario does not **ssh** into the guests but simple launches the guests.

Perform the following steps on your Controller node, unless otherwise specified.

1. If your Red Hat OpenStack Platform deployment does not have any floating point IPs available, extend the *Tenant* network to the **rally** host. Run the following command to install the OpenStack Networking OpenvSwitch agent package:

```
#  yum install openstack-neutron-openvswitch
```

2. Copy the OpenStack Networking configuration files from the Compute node to the Controller node:

```
#  scp COMPUTE_NODE_IP:/etc/neutron/* /etc/neutron
```

3. Set an IP address to the interface that would have been associated with the *Tenant* network had it been part of the OpenStack cluster.

4. Edit the **ovs_neutron_plugin.ini** file and change the **local_ip** to the IP address that resides on the *Tenant* network:

```
[OVS]
vxlan_udp_port=4789
network_vlan_ranges=
tunnel_type=vxlan
tunnel_id_ranges=1:1000
tenant_network_type=vxlan
local_ip=IP_WITHIN_TENANT_NETWORK
enable_tunneling=True
integration_bridge=br-int
tunnel_bridge=br-tu
```

5. Edit the **/etc/neutron/plugin.ini** file and change the **local_ip** to the IP address that resides on the *Tenant* network:

```
[OVS]
vxlan_udp_port=4789
network_vlan_ranges=
tunnel_type=vxlan
tunnel_id_ranges=1:1000
tenant_network_type=vxlan
local_ip=IP_WITHIN_TENANT_NETWORK
enable_tunneling=True
integration_bridge=br-int
tunnel_bridge=br-tun
```

6. Restart the **neutron-openvswitch-agent**:

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

7. On the Controller node, verify the agent is properly setup by using the **neutron agent-list** command:

```
# neutron agent-list
... Output Abbreviated ... ]
| 8578499b-0873-47a9-9cae-9884d4abf768 | Open vSwitch agent |
Controller_Host | :-) | True | neutron-openvswitch-agent
```

8. On the Controller node, create the variables **netid** and **hostid** with the following information:

```
# netid=PRIVATE_NETWORK_ID
# echo $netid
# hostid=CONTROLLER_NODE_HOSTNAME
# echo $hostid
```

9. On the Controller node, create a **neutron** port labelled **rally-port** that binds to the **host_id** and creates the port within the network of the associated **netid**.

```
# neutron port-create --name rally-port --binding:host_id=$hostid
$netid
Created a new port:
+-----------------------
+------------------------------------------------------------------
----------------+
| Field                 | Value
|
+-----------------------
+------------------------------------------------------------------
----------------+
| admin_state_up        | True
|
| allowed_address_pairs |
|
| binding:host_id       | VISIONING_HOST
|
| binding:profile       | {}
|
| binding:vif_details   | {"port_filter": true, "ovs_hybrid_plug":
true}
|
| binding:vif_type      | ovs
|
| binding:vnic_type     | normal
|
| device_id             |
|
| device_owner          |
|
| extra_dhcp_opts       |
|
| fixed_ips             | {"subnet_id": "5279a66d-a5f5-4639-b664-
163c39f26838", "ip_address": "10.1.0.2"}
|
| id                    | 0a9e54da-6588-42ce-9011-d637c3387670
|
| mac_address           | fa:16:3e:44:c3:d9
|
| name                  | rally-port
|
| network_id            | 1cd7ae4f-d057-41bd-8c56-557771bf9f73
|
| security_groups       | 76fe37c7-debb-46b4-a57a-d7dbb9dfd0ed
|
| status                | DOWN
|
| tenant_id             | 6f98906137a2421da579d4e70714cac6
|
+-----------------------
+------------------------------------------------------------
----------------+
```

10. Within the Controller node, modify the **rally** virtual machine XML with the following:

```
 #  virsh edit rally
...
   <interface type='bridge'>
     <mac address='fa:16:3e:1a:3b:f1'/>
     <source bridge='br-int'/>
     <virtualport type='openvswitch'>
       <parameters interfaceid='neutron-port-id'/>
     </virtualport>
     <target dev='vnet4'/>
     <model type='virtio'/>
     <alias name='net1'/>
     <address type='pci' domain='0x0000' bus='0x00' slot='0x0f'
function='0x0'/>
   </interface>
...
```

Note

Make sure you update the **`neutron-port-id`** with the **`id`** located in the previous step when creating the **`rally-port`**.

11. After the XML file changes have been applied to the **`rally`** guest, reboot the guest.

18.4. Validate the OpenStack Benchmarking Service

In the **`rally`** virtual machine, run the following commands:

1. List the current status of the Red Hat OpenStack Platform deployment:

```
 #  rally deployment list
--------------------------+----------------------------+----------
+------------------+--------+
| uuid                                 | created_at
| name     | status            | active |
+------------------------------------+---------------------------
-+----------+-----------------+--------+
| 496e3be9-6020-4543-82f6-e7eb90517153 | 2015-05-12 15:18:13.817669 |
RA-Rally | deploy->finished | * |
+------------------------------------+---------------------------
-+----------+-----------------+--------+
```

2. Using the built-in sample **`keystone`** scenario labeled **`create-user.json`**, test the **`rally`** environment to confirm proper functionality of the Benchmarking service. The following **`create-user.json`** scenario creates 10 names at a time up to 100 times and shows the duration of creating a user in subsets of 10 until reaching the maximum total users to generate (up to 100). This sample test verifies that your Benchmarking service can access the OpenStack environment.

```
 #  rally task start
/path/to/myrally/rally/samples/tasks/scenarios/keystone/create-
user.json
```