Chapter 9. Installing the OpenStack Networking Service

9.1. OpenStack Networking Installation Overview

9.1.1. OpenStack Networking Architecture

OpenStack Networking provides cloud administrators with flexibility in deciding which individual services should run on which physical systems. All service daemons can be run on a single physical host for evaluation purposes. Alternatively each service can have its own physical host or even be replicated across multiple hosts for redundancy.

This chapter focuses on an architecture that combines the role of cloud controller with that of the network host while allowing for multiple compute nodes on which virtual machine instances run. The networking services deployed on the cloud controller in this chapter may also be deployed on a separate network host entirely. This is recommended for environments where it is expected that significant amounts of network traffic will need to be routed from virtual machine instances to external networks.

Report a bug

9.1.2. OpenStack Networking API

OpenStack Networking provides a powerful API to define the network connectivity and addressing used by devices from other services, such as OpenStack Compute.

The OpenStack Compute API has a virtual server abstraction to describe compute resources. Similarly, the OpenStack Networking API has virtual network, subnet, and port abstractions to describe network resources. In more detail:

Network
An isolated L2 segment, analogous to VLAN in the physical networking world.

Subnet
A block of v4 or v6 IP addresses and associated configuration state.

Port
A connection point for attaching a single device, such as the NIC of a virtual server, to a virtual network. Also describes the associated network configuration, such as the MAC and IP addresses to be used on that port.

You can configure rich network topologies by creating and configuring networks and subnets, and then instructing other OpenStack services like OpenStack Compute to attach virtual devices to ports on these networks. In particular, OpenStack Networking supports each tenant having multiple private networks, and allows tenants to choose their own IP addressing scheme, even if those IP addresses overlap with those used by other tenants. This enables very advanced cloud networking use cases, such as building multi-tiered web applications and allowing applications to be migrated to the cloud without changing IP addresses.

Even if a cloud administrator does not intend to expose the above capabilities to tenants directly, the OpenStack Networking API can be very useful for administrative purposes. The API provides significantly more flexibility for the cloud administrator when customizing network offerings.

Report a bug

9.1.3. OpenStack Networking API Extensions

The OpenStack networking API allows plug-ins to provide extensions to enable additional networking functional not available in the core API itself.

Provider Networks
Provider networks allow the creation of virtual networks that map directly to networks in the physical data center. This allows allows the administrator to give tenants direct access to a public network such as the Internet or to integrate with existing VLANs in the physical networking environment that have a defined meaning or purpose.

When the provider extension is enabled OpenStack networking users with administrative privileges are able to see additional provider attributes on all virtual networks. In addition such users have the ability to specify provider attributes when creating new provider networks.

Both the Open vSwitch and Linux Bridge plug-ins support the provider networks extension.

Layer 3 (L3) Routing and Natural Address Translation (NAT)
The L3 routing API extensions provides abstract L3 routers that API users are able to dynamically provision and configure. These routers are able to connect to one or more Layer 2 (L2) OpenStack networking controlled

networks. Additionally the routers are able to provide a gateway that connects one or more private L2 networks to an common public or external network such as the Internet.

The L3 router provides basic NAT capabilities on gateway ports that connect the router to external networks. The router supports floating IP addresses which give a static mapping between a public IP address on the external network and the private IP address on one of the L2 networks attached to the router.

This allows the selective exposure of compute instances to systems on an external public network. Floating IP addresses are also able to be reallocated to different OpenStack networking ports as necessary.

Security Groups

Security groups and security group rules allow the specification of the specific type and direction of network traffic that is allowed to pass through a given network port. This provides an additional layer of security over and above any firewall rules that exist within a compute instance. The security group is a container object which can contain one or more security rules. A single security group can be shared by multiple compute instances.

When a port is created using OpenStack networking it is associated with a security group. If a specific security group was not specified then the port is associated with the **`default`** security group. By default this group will drop all inbound traffic and allow all outbound traffic. Additional security rules can be added to the **`default`** security group to modify its behaviour or new security groups can be created as necessary.

The Open vSwitch, Linux Bridge, Nicira NVP, NEC, and Ryu networking plug-ins currently support security groups.

Note

Unlike compute security groups OpenStack networking security groups are applied on a per port basis rather than on a per instance basis.

Report a bug

9.1.4. OpenStack Networking Plug-ins

The original OpenStack Compute network implementation assumed a basic networking model where all network isolation was performed through the use of Linux VLANs and IP tables. OpenStack Networking introduces the concept of a *plug-in*, which is a pluggable back-end implementation of the OpenStack Networking API. A plug-in can use a variety of technologies to implement the logical API requests. Some OpenStack Networking plug-ins might use basic Linux VLANs and IP tables, while others might use more advanced technologies, such as L2-in-L3 tunneling or OpenFlow, to provide similar benefits.

Plug-ins for these networking technologies are currently tested and supported for use with Red Hat OpenStack:

- Open vSwitch (*openstack-quantum-openvswitch*)
- Linux Bridge (*openstack-quantum-linuxbridge*)

Other plug-ins that are also packaged and available include:

- Cisco (*openstack-quantum-cisco*)
- NEC OpenFlow (*openstack-quantum-nec*)
- Nicira (*openstack-quantum-nicira*)
- Ryu (*openstack-quantum-ryu*)

Plug-ins enable the cloud administrator to weigh different options and decide which networking technology is right for the deployment.

Report a bug

9.1.5. OpenStack Networking Agents

As well as the OpenStack Networking service and the installed plug-in a number of components combine to provide networking functionality in the OpenStack environment.

L3 Agent

The L3 agent is part of the *openstack-quantum* package. It acts as an abstract L3 router that can connect to and provide gateway services for multiple L2 networks.

The nodes on which the L3 agent is to be hosted must not have a manually configured IP address on a network interface that is connected to an external network. Instead there must be a range of IP addresses from the external network that are available for use by OpenStack Networking. These IP addresses will be assigned to the

routers that provide the link between the internal and external networks.

The range selected must be large enough to provide a unique IP address for each router in the deployment as well as each desired floating IP.

DHCP Agent

The OpenStack Networking DHCP agent is capable of allocating IP addresses to virtual machines running on the network. If the agent is enabled and running when a subnet is created then by default that subnet has DHCP enabled.

Plug-in Agent

Many of the OpenStack Networking plug-ins, including Open vSwitch and Linux Bridge, utilize their own agent. The plug-in specific agent runs on each node that manages data packets. This includes all compute nodes as well as nodes running the dedicated agents **`quantum-dhcp-agent`** and **`quantum-l3-agent`**.

Report a bug

9.1.6. Recommended Networking Deployment

OpenStack Networking provides an extreme amount of flexibility when deploying networking in support of a compute environment. As a result, the exact layout of a deployment will depend on a combination of expected workloads, expected scale, and available hardware.

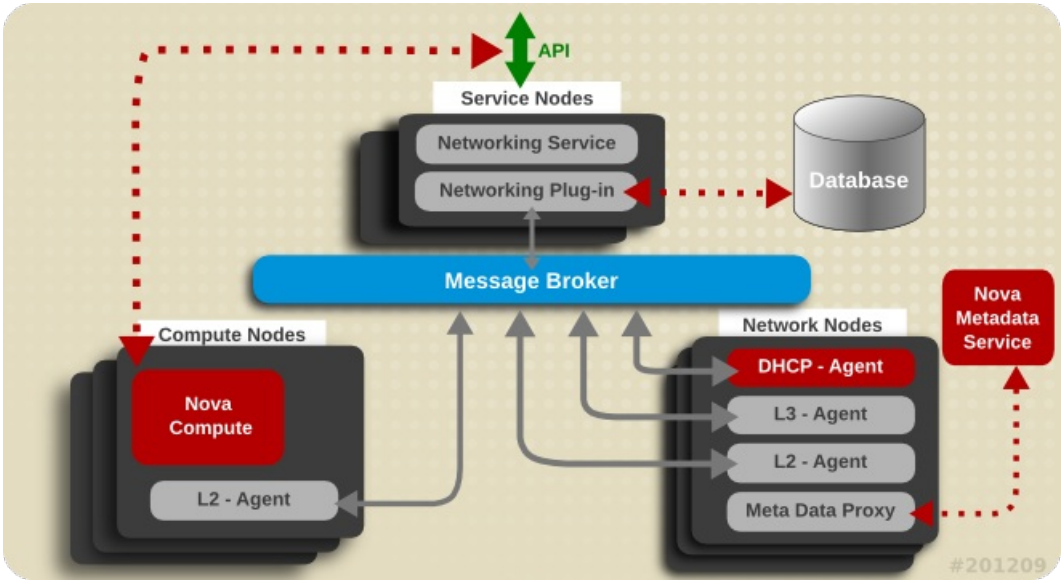

Figure 9.1. Deployment Architecture

For demonstration purposes, this chapter concentrates on a networking deployment that consists of these types of nodes:

Service Node

The service node exposes the networking API to clients and handles incoming requests before forwarding them to a message queue to be actioned by the other nodes. The service node hosts both the networking service itself and the active networking plug-in.

In environments that use *controller nodes* to host the client-facing APIs and schedulers for all services, the controller node would also fulfil the role of service node as it is applied in this chapter.

Network Node

The network node handles the majority of the networking workload. It hosts the DHCP agent, the Layer 3 (L3) agent, the Layer 2 (L2) Agent, and the metadata proxy. In addition to plug-ins that require an agent, it runs an instance of the plug-in agent (as do all other systems that handle data packets in an environment where such plug-ins are in use). Both the Open vSwitch and Linux Bridge plug-ins include an agent.

Compute Node

The compute hosts the compute instances themselves. To connect compute instances to the networking services, compute nodes must also run the L2 agent. Like all other systems that handle data packets it must also run an instance of the plug-in agent.

This deployment type and division of responsibilities is made only as a suggestion. Other divisions are equally valid, in particular in some environments the network and service nodes may be combined.

Warning

Environments that have been configured to use Compute networking, either using the **packstack** utility or manually, can be reconfigured to use OpenStack Networking. This is however currently **not** recommended for environments where Compute instances have already been created and configured to use Compute networking. If you wish to proceed with such a conversion. you must ensure that you stop the **openstack-nova-network** service on each Compute node using the **service** command before proceeding.

```
# service openstack-nova-network stop
```

You must also disable the **openstack-nova-network** service permanently on each node using the **chkconfig** command.

```
# chkconfig openstack-nova-network off
```

Important

When running 2 or more active controller nodes, do not run **nova-consoleauth** on more than one node. Running more than one instance of **nova-consoleauth** causes a conflict between nodes with regard to token requests which may cause errors.

See Also:

- Section 9.2, “Networking Prerequisite Configuration”
- Section 9.3, “Common Networking Configuration”
- Section 9.4, “Configuring the Networking Service”
- Section 9.5, “Configuring the DHCP Agent”
- Section 9.6, “Configuring a Provider Network”
- Section 9.7, “Configuring the L3 Agent”
- Section 9.8, “Configuring the Plug-in Agent”

Report a bug

9.2. Networking Prerequisite Configuration

9.2.1. Creating the OpenStack Networking Database

In this procedure the database and database user that will be used by the networking service will be created. These steps must be performed while logged in to the database server as the **root** user.

1. Connect to the database service using the **mysql** command.

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

2. Create the database. If you intend to use the:
   - Open vSwitch plug-in, the recommended database name is **ovs_quantum**.
   - Linux Bridge plug-in, the recommended database name is **quantum_linux_bridge**.

   This example uses the database name of '**ovs_quantum**'.

   ```
   mysql> CREATE DATABASE ovs_quantum;
   ```

3. Create a **quantum** database user and grant it access to the **ovs_quantum** database.

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

   ```
   mysql> GRANT ALL ON ovs_quantum.* TO 'quantum'@'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 command.

```
mysql> quit
```

The OpenStack Networking database 'ovs_quantum' has been created. The database will be populated during service configuration.

Report a bug

9.2.2. Creating the OpenStack Networking Identity Records

The steps outlined in this procedure cover the creation of these identity records to support the OpenStack networking service:

- A **quantum** user.
- A **network** tenant.
- An entry linking the **quantum** user, **admin** role, and **network** tenant.
- An **quantum** service entry.
- An endpoint entry associated with the **quantum** service.

These entries will assist other OpenStack services attempting to locate and access the networking functionality provided by the OpenStack Networking service.

1. Authenticate as the administrator of the identity service by running the **source** command on the **keystonerc_admin** file containing the required credentials.

```
# source ~/keystonerc_admin
```

2. Create a user named **quantum** for the OpenStack networking service to use.

```
# keystone user-create --name quantum --pass PASSWORD
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 1df18bcd14404fa9ad954f9d5eb163bc |
|   name   |              quantum             |
| tenantId |                                  |
+----------+----------------------------------+
```

   Replace ***PASSWORD*** with a secure password that will be used by the OpenStack networking service when authenticating with the identity service. Take note of the unique user identifier returned as it will be used in subsequent steps.

3. Locate the unique identifier of the **admin** role.

```
# keystone role-get admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 78035c5d3cd94e62812d6d37551ecd6a |
|   name   |              admin               |
+----------+----------------------------------+
```

   Take note of the unique role identifier returned as it will be used in subsequent steps.

4. Create a tenant named ***network*** for the OpenStack networking service to use.

```
# keystone tenant-create --name network
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |                                  |
|   enabled   |               True               |
|      id     | f466f9e136214ed0b4151d4f8e375345 |
|     name    |              network             |
+-------------+----------------------------------+
```

   Take note of the unique tenant identifier returned as it will be used in subsequent steps.

5. Use the **keystone user-role-add** command to link the **quantum** user, **admin** role, and **network** tenant together.

```
# keystone user-role-add --user-id USERID \
        --role-id ROLEID \
        --tenant-id TENANTID
```

   Replace:

   - ***USERID*** with the unique user identifier returned by **keystone user-create** when the user was created.
   - ***ROLEID*** with the unique user identifier returned by **keystone role-get** when identifying the **admin** role.
   - ***TENANTID*** with the unique tenant identifier returned by the **keystone tenant-create** when the tenant was created.

6. Create the **quantum** service entry.

```
# keystone service-create --name quantum \
        --type network \
        --description "OpenStack Networking Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |   OpenStack Networking Service   |
|      id     | 134e815915f442f89c39d2769e278f9b |
|     name    |              quantum             |
|     type    |              network             |
+-------------+----------------------------------+
```

   Take note of the unique service identifier returned as it will be used in subsequent steps.

7. Create the **network** endpoint entry.

```
# keystone endpoint-create --service-id SERVICEID \
        --publicurl "http://IP:9696" /
        --adminurl "http://IP:9696" /
        --internalurl "http://IP:9696"
```

   Replace ***SERVICEID*** with the identifier returned by the **keystone service-create** command. Replace ***IP*** with the IP address or host name of the system that will be acting as the network node.

All supporting identity service entries required by the OpenStack networking service have been created.

Report a bug

9.3. Common Networking Configuration

9.3.1. Upgrading the Kernel

Red Hat OpenStack includes a custom Red Hat Enterprise Linux kernel that supports the use of network namespaces. This kernel **must** be installed on nodes that will handle OpenStack networking traffic. Additionally the Open vSwitch plug-in will not work with kernels with versions lower than **2.6.32-*343*.el6.x86_64**.

Follow the steps listed in this procedure on each node in the environment that will handle OpenStack networking traffic. You must log in to each node as the **root** user to complete the procedure.

1. Use the **uname** command to identify the kernel that is currently in use on the system.

```
# uname --kernel-release
```

   A. If the output includes the text **openstack** then the system already has a network namespaces enabled kernel.

```
2.6.32-358.6.2.openstack.el6.x86_64
```

      No further action is required to install a network namespaces enabled kernel on this system.

   B. If the output does **not** include the text **openstack** then the system does not currently have a network namespaces enabled kernel and further action must be taken.

```
2.6.32-358.el6.x86_64
```

      Further action is required to install a network namespaces enabled kernel on this system. Follow the remaining steps outlined in this procedure to perform this task.

Note

Note that the release field may contain a higher value than ***358***. As new kernel updates are released this value is increased.

2. Install the updated kernel with network namespaces support using the **yum** command.

```
# yum install "kernel-2.6.*.openstack.el6.x86_64"
```

The use of the wildcard character (*****) ensures that the latest kernel release available will be installed.

3. Reboot the system to ensure that the new kernel is running before proceeding with OpenStack networking installation.

```
# reboot
```

4. Run the **uname** command again once the system has rebooted to confirm that the newly installed kernel is running.

```
# uname --kernel-release
2.6.32-358.6.2.openstack.el6.x86_64
```

The system is now running a kernel with network namespaces support, enabling advanced OpenStack networking configurations.

Report a bug

9.3.2. Disabling Network Manager

OpenStack networking currently does not work on systems that have the Network Manager (**NetworkManager**) service enabled. The Network Manager service is currently enabled by default on Red Hat Enterprise Linux installations where one of these package groups was selected during installation:

- Desktop
- Software Development Workstation

The Network Manager service is **not** currently enabled by default on Red Hat Enterprise Linux installations where one of these package groups was selected during installation:

- Basic Server
- Database Server
- Web Server
- Identity Management Server
- Virtualization Host
- Minimal Install

Follow the steps listed in this procedure while logged in as the **root** user on each system in the environment that will handle network traffic. This includes the system that will host the OpenStack Networking service, all network nodes, and all compute nodes.

These steps ensure that the **NetworkManager** service is disabled and replaced by the standard network service for all interfaces that will be used by OpenStack Networking.

1. Verify Network Manager is currently enabled using the **chkconfig** command.

```
# chkconfig --list NetworkManager
```

The output displayed by the **chkconfig** command inicates whether or not the Network Manager service is enabled.

A. The system displays an error if the Network Manager service is not currently installed:

```
error reading information on service NetworkManager: No such file or directory
```

If this error is displayed then no further action is required to disable the Network Manager service.

B. The system displays a list of numerical run levels along with a value of **on** or **off** indicating whether the Network Manager service is enabled when the system is operating in the given run level.

```
NetworkManager  0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

If the value displayed for all run levels is **off** then the Network Manager service is disabled and no further action is required. If the value displayed for any of the run levels is **on** then the Network Manager service is enabled

and further action is required.

2. Ensure that the Network Manager service is stopped using the **service** command.

```
# service NetworkManager stop
```

3. Ensure that the Network Manager service is disabled using the **chkconfig** command.

```
# chkconfig NetworkManager off
```

4. 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 of the form **ifcfg-*X*** where ***X*** is replaced by the name of the interface. Valid interface names include **eth0**, **p1p5**, and **em1**.

   In each file ensure that the **NM_CONTROLLED** configuration key is set to **no** and the **ON_BOOT** configuration key is set to **yes**.

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

   This action ensures that the standard network service will take control of the interfaces and automatically activate them on boot.

5. Ensure that the network service is started using the **service** command.

```
# service network start
```

6. Ensure that the network service is enabled using the **chkconfig** command.

```
# chkconfig network on
```

The Network Manager service has been disabled. The standard network service has been enabled and configured to control the required network interfaces.

Report a bug

9.3.3. Installing the Packages

The OpenStack Networking service requires the following packages:

openstack-quantum
: Provides the networking service and associated configuration files.

openstack-quantum-PLUGIN
: Provides a networking plug-in. Replace ***PLUGIN*** with one of the recommended plug-ins (**openvswitch** and **linuxbridge**).

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 service node, all network nodes, and all Compute nodes.

To install all of the above packages, execute the following command while logged in as the **root** user:

```
# yum install -y openstack-quantum \
   openstack-quantum-PLUGIN \
   openstack-utils \
   openstack-selinux
```

Replace ***PLUGIN*** with **openvswitch** or **linuxbridge** (determines which plug-in is installed).

The networking services are installed and ready to be configured.

Report a bug

9.3.4. Configuring the Firewall

Systems attempting to use the functionality provided by the networking service access it over the network using port **9696**.

To allow this the firewall on the service node must be altered to allow network traffic on this port. All steps in this procedure must be run while logged in to the server hosting the image storage service as the **root** user.

1. Open the **/etc/sysconfig/iptables** file in a text editor.
2. Add an INPUT rule allowing TCP traffic on port **9696** to the file. The new rule must appear before any INPUT rules that REJECT traffic.

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

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

```
# service iptables restart
```

The **iptables** firewall is now configured to allow incoming connections to the networking service on port **9696**.

Report a bug

9.4. Configuring the Networking Service

Prerequisites:

- Section 9.3, “Common Networking Configuration”

The configuration of the network service itself is contained in the **/etc/quantum/quantum.conf** file.

Repeat these configuration steps on each node that will host an instance of the networking service while logged in as the **root** user.

1. **Setting the Identity Values**

   The networking services must be explicitly configured to use the identity service for authentication.

   a. Set the authentication strategy (**auth_strategy**) configuration key to **keystone** using the **openstack-config** command.

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

   b. Set the authentication host (**auth_host** configuration key to the IP address or host name of the identity server.

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

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

   c. Set the administration tenant name (**admin_tenant_name**) configuration key to the name of the tenant that was created for the use of the networking services. This is typically **network**.

```
# openstack-config --set /etc/quantum/quantum.conf \
   keystone_authtoken admin_tenant_name network
```

   d. Set the administration user name (**admin_user**) configuration key to the name of the user that was created for the use of the networking services. This is typically **quantum**.

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

   e. Set the administration password (**admin_password**) configuration key to the password that is associated with the user specified in the previous step.

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

   The authentication keys used by the networking services have been set and will be used when the services are started.

2. **Setting the Message Broker**

   The networking services must be explicitly configured with the type, location, and authentication details of the message broker.

a. Use the **openstack-config** utility to set the value of the **rpc_backend** configuration key to Qpid.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT rpc_backend quantum.openstack.common.rpc.impl_qpid
```

b. Use the **openstack-config** utility to set the value of the **qpid_hostname** configuration key to the host name of the Qpid server.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_hostname IP
```

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

c. If you have configured Qpid to authenticate incoming connections, you must provide the details of a valid Qpid user in the networking configuration.

a. Use the **openstack-config** utility to set the value of the **qpid_username** configuration key to the username of the Qpid user that the networking services must use when communicating with the message broker.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_username USERNAME
```

Replace ***USERNAME*** with the required Qpid user name.

b. Use the **openstack-config** utility to set the value of the **qpid_password** configuration key to the password of the Qpid user that the networking services must use when communicating with the message broker.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_password PASSWORD
```

Replace ***PASSWORD*** with the password of the Qpid user.

d. If you configured Qpid to use SSL, you must inform the networking services of this choice. Use **openstack-config** utility to set the value of the **qpid_protocol** configuration key to **ssl**.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_protocol ssl
```

The value of the **qpid_port** configuration key must be set to **5671** as Qpid listens on this different port when SSL is in use.

```
# openstack-config --set /etc/quantum/quantum.conf \
  DEFAULT qpid_port 5671
```

Important

To communicate with a Qpid message broker that uses SSL the node must also have:

- The *nss* package installed.
- The certificate of the relevant certificate authority installed in the system NSS database (**/etc/pki/nssdb/**).

The **certtool** command is able to import certificates into the NSS database. See the **certtool** manual page for more information (**man certtool**).

The networking services have been configured to use the message broker and any authentication schemes that it presents.

3. **Setting the Plug-in**

Additional configuration settings must be applied to enable the desired plug-in.

A. **Open vSwitch**

a. Create a symbolic link between the **/etc/quantum/plugin.ini** path referred to by the networking service and the plug-in specific configuration file.

```
# ln -s /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini \
       /etc/quantum/plugin.ini
```

b. Update the value of the **tenant_network_type** configuration key in the **/etc/quantum/plugin.ini** file to refer to the type of network that must be used for tenant networks. Supported values are **flat**, **vlan**, and **local**.

The default is **local** but this is not recommended for real deployments.

```
# openstack-config --set /etc/quantum/plugin.ini \
   OVS tenant_network_type TYPE
```

Replace ***TYPE*** with the type chosen tenant network type.

c. If **flat** or **vlan** networking was chosen, the value of the **network_vlan_ranges** configuration key must also be set. This configuration key maps physical networks to VLAN ranges.

Mappings are of the form ***NAME***:***START***:***END*** where ***NAME*** is replaced by the name of the physical network, ***START*** is replaced by the VLAN identifier that starts the range, and ***END*** is replaced by the replaced by the VLAN identifier that ends the range.

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

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

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

d. Update the value of the **core_plugin** configuration key in the **/etc/quantum/quantum.conf** file to refer to the Open vSwitch plug-in.

```
# openstack-config --set /etc/quantum/quantum.conf \
   DEFAULT core_plugin \
   quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
```

B. **Linux Bridge**

a. Create a symbolic link between the **/etc/quantum/plugin.ini** path referred to by the networking service and the plug-in specific configuration file.

```
# ln -s /etc/quantum/plugins/linuxbridge/linuxbridge_conf.ini \
        /etc/quantum/plugin.ini
```

b. Update the value of the **tenant_network_type** configuration key in the **/etc/quantum/plugin.ini** file to refer to the type of network that must be used for tenant networks. Supported values are **flat**, **vlan**, and **local**.

The default is **local** but this is not recommended for real deployments.

```
# openstack-config --set /etc/quantum/plugin.ini \
   VLAN tenant_network_type TYPE
```

Replace ***TYPE*** with the type chosen tenant network type.

c. If **flat** or **vlan** networking was chosen, the value of the **network_vlan_ranges** configuration key must also be set. This configuration key maps physical networks to VLAN ranges.

Mappings are of the form ***NAME***:***START***:***END*** where ***NAME*** is replaced by the name of the physical network, ***START*** is replaced by the VLAN identifier that starts the range, and ***END*** is replaced by the replaced by the VLAN identifier that ends the range.

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

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

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

d. Update the value of the **core_plugin** configuration key in the **/etc/quantum/quantum.conf** file to refer to the Linux Bridge plug-in.

```
# openstack-config --set /etc/quantum/quantum.conf \
   DEFAULT core_plugin \
   quantum.plugins.linuxbridge.lb_quantum_plugin.LinuxBridgePluginV2
```

4. **Setting the Database Connection String**

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

a. Use the **openstack-config** command to set the value of the **connection** configuration key.

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

Replace:

- ***USER*** with the database user name the networking service is to use, usually **quantum**.
- ***PASS*** with the password of the chosen database user.
- ***IP*** with the IP address or host name of the database server.
- ***DB*** with the name of the database that has been created for use by the networking service (**ovs_quantum** was used as the example in the previous *Creating the OpenStack Networking Database* section).

5. **Start the Networking Service**
   a. Start the networking service using the **service** command.

```
# service quantum-server start
```

   b. Enable the networking service permanantly using the **chkconfig** command.

```
# chkconfig quantum-server on
```

The networking service is configured and running. Further action is however required to configure and run the various networking agents that are also fundamental to providing networking functionality.

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/quantum/quantum.conf** file.

Report a bug

9.5. Configuring the DHCP Agent

Prerequisites:

- Section 9.3, “Common Networking Configuration”

Follow the steps listed in this procedure to configure the DHCP Agent. All steps listed in this procedure must be performed on the network node while logged in as the **root** user on the system hosting the DHCP agent.

1. **Configuring Authentication**
   The DHCP agent must be explicitly configured to use the identity service for authentication.
   a. Set the authentication strategy (**auth_strategy**) configuration key to **keystone** using the **openstack-config** command.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
   DEFAULT auth_strategy keystone
```

   b. Set the authentication host (**auth_host** configuration key to the IP address or host name of the identity server.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
   keystone_authtoken auth_host IP
```

   Replace ***IP*** with the IP address or host name of the identity server.
   c. Set the administration tenant name (**admin_tenant_name**) configuration key to the name of the tenant that was created for the use of the networking services. This is typically ***network***.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
   keystone_authtoken admin_tenant_name network
```

   d. Set the administration user name (**admin_user**) configuration key to the name of the user that was created for the use of the networking services. This is typically ***quantum***.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
   keystone_authtoken admin_user quantum
```

   e. Set the administration password (**admin_password**) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/quantum/dhcp_agent.ini \
   keystone_authtoken admin_password PASSWORD
```

2. **Configuring the Interface Driver**

   Set the value of the **interface_driver** configuration key in the **/etc/quantum/dhcp_agent.ini** file based on the networking plug-in being used. Execute only the configuration step that applies to the plug-in used in your environment.

   A. **Open vSwitch Interface Driver**

   ```
   # openstack-config --set /etc/quantum/dhcp_agent.ini \
     DEFAULT interface_driver quantum.agent.linux.interface.OVSInterfaceDriver
   ```

   B. **Linux Bridge Interface Driver**

   ```
   # openstack-config --set /etc/quantum/dhcp_agent.ini \
     DEFAULT interface_driver quantum.agent.linux.interface.BridgeInterfaceDriver
   ```

3. **Starting the DHCP Agent**

   a. Use the **service** command to start the **quantum-dhcp-agent** service.

   ```
   # service quantum-dhcp-agent start
   ```

   b. Use the **chkconfig** command to ensure that the **quantum-dhcp-agent** service will be started automatically in the future.

   ```
   # chkconfig quantum-dhcp-agent on
   ```

The DHCP agent has been configured and started.

Report a bug

9.6. Configuring a Provider Network

Prerequisites:

- Section 9.3, “Common Networking Configuration”

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 is in use. The second method, which is supported by both the Open vSwitch plug-in and the Linux Bridge plug-in, is to use an external provider network.

To use an external provider network it is first necessary to create one. Follow the steps outlined in this procedure while logged in to a system with the OpenStack networking client - provided by the *python-quantumclient* package installed. You must also have access to a **keystonerc_admin** file containing the authentication details of the OpenStack 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.

1. Use the **source** command to load the credentials of the administrative user.

   ```
   $ source ~/keystonerc_admin
   ```

2. Use the **net-create** action of the **quantum** command line client to create a new provider network.

   ```
   $ quantum net-create EXTERNAL_NAME \
     --router:external True \
     --provider:network_type TYPE \
     --provider:physical_network PHYSICAL_NAME \
     --provider:segmentation_id VLAN_TAG
   ```

   Replace these strings with the appropriate values for your environment:

   - Replace ***EXTERNAL_NAME*** with a name for the new external network provider.
   - Replace ***PHYSICAL_NAME*** with a name for the physical network. This is not applicable if you intend to use a local network type.
   - Replace ***TYPE*** with the type of provider network you wish to use. Supported values are **flat** (for flat networks), **vlan** (for VLAN networks), and **local** (for local networks).
   - 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** then this parameter is not required.

   Take note of the unique external network identifier returned, this will be required in subsequent steps.

3. Use the **subnet-create** action of the command line client to create a new subnet for the new external provider network.

```
$ quantum subnet-create --gateway GATEWAY \
   --allocation-pool start=IP_RANGE_START,end=IP_RANGE_END \
   --disable-dhcp EXTERNAL_NAME EXTERNAL_CIDR
```

   Replace these strings with the appropriate values for your environment:

   - Replace ***GATEWAY*** with the IP address or hostname of the system that is to act as the gateway for the new subnet.
   - Replace ***IP_RANGE_START*** with the IP address that denotes the start of the range of IP addresses within the new subnet that floating IP addresses will be allocated from.
   - Replace ***IP_RANGE_END*** with the IP address that denotes the end of the range of IP addresses within the new subnet that floating IP addresses will be allocated from.
   - 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 would be **192.168.100.0/24**.

   Take note of the unique subnet identifier returned, this will be required in subsequent steps.

Important

The IP address used to replace the string ***GATEWAY*** **must** be within the block of IP addresses specified in place of the ***EXTERNAL_CIDR*** string but outside of the block of IP addresses specified by the range started by ***IP_RANGE_START*** and ended by ***IP_RANGE_END***.
The block of IP addresses specifed by the range started by ***IP_RANGE_START*** and ended by ***IP_RANGE_END*** **must** also fall within the block of IP addresses specified by ***EXTERNAL_CIDR***.

4. Use the **router-create** action of the **quantum** command line client to create a new router.

```
$ quantum router-create NAME
```

   Replace ***NAME*** with the name to give the new router. Take note of the unique router identifier returned, this will be required in subsequent steps.

5. Use the **router-gateway-set** action of the **quantum** command line client to link the newly created router to the external provider network.

```
$ quantum router-gateway-set ROUTER NETWORK
```

   Replace ***ROUTER*** with the unique identifier of the router, replace ***NETWORK*** with the unique identifier of the external provider network.

6. Use the **router-interface-add** action of the **quantum** command line client to link the newly created router to the subnet.

```
$ quantum router-interface-add ROUTER SUBNET
```

   Replace ***ROUTER*** with the unique identifier of the router, replace ***SUBNET*** with the unique identifier of the subnet.

An external provider network has been created. Use the unique identifier of the router when configuring the L3 agent.

Report a bug

9.7. Configuring the L3 Agent

Prerequisites:

- Section 9.3, “Common Networking Configuration”

Follow the steps listed in this procedure to configure the L3 agent. All steps listed in this procedure must be performed on the network node while logged in as the **root** user.

1. **Configuring Authentication**
   a. Set the authentication strategy (**auth_strategy**) configuration key to **keystone** using the **openstack-config** command.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
   DEFAULT auth_strategy keystone
```

b. Set the authentication host (**auth_host** configuration key to the IP address or host name of the identity server.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
   keystone_authtoken auth_host IP
```

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

c. Set the administration tenant name (**admin_tenant_name**) configuration key to the name of the tenant that was created for the use of the networking services. Usually this is ***network***.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
   keystone_authtoken admin_tenant_name network
```

d. Set the administration user name (**admin_user**) configuration key to the name of the user that was created for the use of the networking services. Usually this is ***network***.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
   keystone_authtoken admin_user network
```

e. Set the administration password (**admin_password**) configuration key to the password that is associated with the user specified in the previous step.

```
# openstack-config --set /etc/quantum/metadata_agent.ini \
   keystone_authtoken admin_password PASSWORD
```

2. **Configuring the Interface Driver**

Set the value of the **interface_driver** configuration key in the **/etc/quantum/l3_agent.ini** file based on the networking plug-in being used. Execute only the configuration step that applies to the plug-in used in your environment.

A. **Open vSwitch Interface Driver**

```
# openstack-config --set /etc/quantum/l3_agent.ini \
   DEFAULT interface_driver quantum.agent.linux.interface.OVSInterfaceDriver
```

B. **Linux Bridge Interface Driver**

```
# openstack-config --set /etc/quantum/l3_agent.ini \
   DEFAULT interface_driver quantum.agent.linux.interface.BridgeInterfaceDriver
```

3. **Configuring External Network Access**

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. Choose the approach that is most appropriate for the environment.

A. **Using an External Bridge**

To use an external bridge you must create and configure it. Finally the OpenStack networking configuration must be updated to use it. This must be done on each system hosting an instance of the L3 agent.

a. Use the **ovs-ctl** command to create the external bridge named **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 with these contents:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
ONBOOT=yes
BOOTPROTO=none
```

c. Ensure that the value of the **external_network_bridge** configuration key in the **/etc/quantum/l3_agent.ini** file is **br-ex**. This ensures that the L3 agent will use the external bridge.

```
# openstack-config --set /etc/quantum/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 will be required to complete these steps.

a. Ensure that the value of the **external_network_bridge** configuration key in the **/etc/quantum/l3_agent.ini** file is blank. This ensures that the L3 agent does not attempt to use an external bridge.

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT external_network_bridge ""
```

b. Set the value of the **router_id** configuration key in the **/etc/quantum/l3_agent.ini** file to the identifier of the external router that must be used by the L3 agent when accessing the external provider network.

```
# openstack-config --set /etc/quantum/l3_agent.ini \
  DEFAULT router_id ROUTER
```

Replace ***ROUTER*** with the unique identifier of the router that has been defined for use when accessing the external provider network.

4. **Starting the L3 Agent**
   a. Use the **service** command to start the **quantum-l3-agent** service.

```
# service quantum-l3-agent start
```

   b. Use the **chkconfig** command to ensure that the **quantum-l3-agent** service will be started automatically in the future.

```
# chkconfig quantum-l3-agent on
```

5. **Starting the Metadata Agent**
   The OpenStack networking metadata agent allows virtual machine instances to communicate with the compute metadata service. It runs on the same hosts as the Layer 3 (L3) agent.
   a. Use the **service** command to start the **quantum-metadata-agent** service.

```
# service quantum-metadata-agent start
```

   b. Use the **chkconfig** command to ensure that the **quantum-metadata-agent** service will be started automatically in the future.

```
# chkconfig quantum-metadata-agent on
```

The L3 agent has been configured and started.

Report a bug

9.8. Configuring the Plug-in Agent

9.8.1. Configuring the Open vSwitch Plug-in Agent

Prerequisites:

- Section 9.3, “Common Networking Configuration”

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.

1. Confirm that the *openvswitch* package is installed. This is normally installed as a dependency of the *quantum-plugin-openvswitch* package.

```
# rpm -qa | grep openvswitch
openvswitch-1.10.0-1.el6.x86_64
openstack-quantum-openvswitch-2013.1-3.el6.noarc
```

2. Start the **openvswitch** service.

```
# service openvswitch start
```

3. Enable the **openvswitch** service permanently.

```
# chkconfig openvswitch on
```

4. Each host running the Open vSwitch agent also requires an Open vSwitch bridge named **br-int**. This bridge is used for private network traffic. Use the **ovs-vsctl** command to create this bridge before starting the agent.

```
# ovs-vsctl add-br br-int
```

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.

5. Ensure that the **br-int** device persists on reboot by creating a **/etc/sysconfig/network-scripts/ifcfg-br-int** file with these contents:

```
DEVICE=br-int
DEVICETYPE=ovs
TYPE=OVSBridge
ONBOOT=yes
BOOTPROTO=none
```

6. Use the **service** command to start the **quantum-openvswitch-agent** service.

```
#  service quantum-openvswitch-agent start
```

7. Use the **chkconfig** command to ensure that the **quantum-openvswitch-agent** service is started automatically in the future.

```
# chkconfig quantum-openvswitch-agent on
```

8. Use the **chkconfig** command to ensure that the **quantum-ovs-cleanup** service is started automatically on boot. When started at boot time this service ensures that the OpenStack Networking agents maintain full control over the creation and management of tap devices.

```
# chkconfig quantum-ovs-cleanup on
```

The networking configuration has been updated to use the Open vSwitch plug-in.

Report a bug

9.8.2. Configuring the Linux Bridge Plug-in Agent

Prerequisites:

- Section 9.3, “Common Networking Configuration”

The Linux Bridge plug-in has a corresponding agent. When the Linux Bridge 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.

1. Use the service command to start the **quantum-linuxbridge-agent** service.

```
# service quantum-linuxbridge-agent start
```

2. Use the chkconfig command to ensure that the **quantum-linuxbridge-agent** service is started automatically in the future.

```
# chkconfig quantum-linuxbridge-agent on
```

The networking configuration has been updated to use the Linux Bridge plug-in.

Report a bug

9.9. Validating the OpenStack Networking Installation

To begin using OpenStack networking it is necessary to deploy networking components to compute nodes. Initial networks and routers must also be defined. It is however possible to perform basic santiy checking of the OpenStack networking deployment by following the steps outline in this procedure.

1. **All Nodes**
   a. Verify that the customized Red Hat Enterprise Linux kernel intended for use with Red Hat OpenStack 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** then 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 then update the system using **yum**.

2. **Service Nodes**

a. Ensure that the **quantum-server** service is running:

```
$ service quantum-server status
quantum-server (pid  3011) is running...
```

3. **Network Nodes**

a. Ensure that the DHCP agent is running:

```
$ service quantum-dhcp-agent status
quantum-dhcp-agent (pid  3012) is running...
```

b. Ensure that the L3 agent is running:

```
$ service quantum-l3-agent status
quantum-l3-agent (pid  3013) is running...
```

c. Ensure that the plug-in agent, if applicable, is running:

```
$ service quantum-PLUGIN-agent status
quantum-PLUGIN-agent (pid  3014) is running...
```

Replace ***PLUGIN*** with the appropriate plug-in for the environment. Valid values include **openvswitch** and **linuxbridge**.

d. Ensure that the metadata agent is running:

```
$ service quantum-metadata-agent status
quantum-metadata-agent (pid  3015) is running...
```

All required services on the service and network nodes are operational. Proceed to deploy some compute nodes, define networks, and define routers to begin using OpenStack networking.

Report a bug