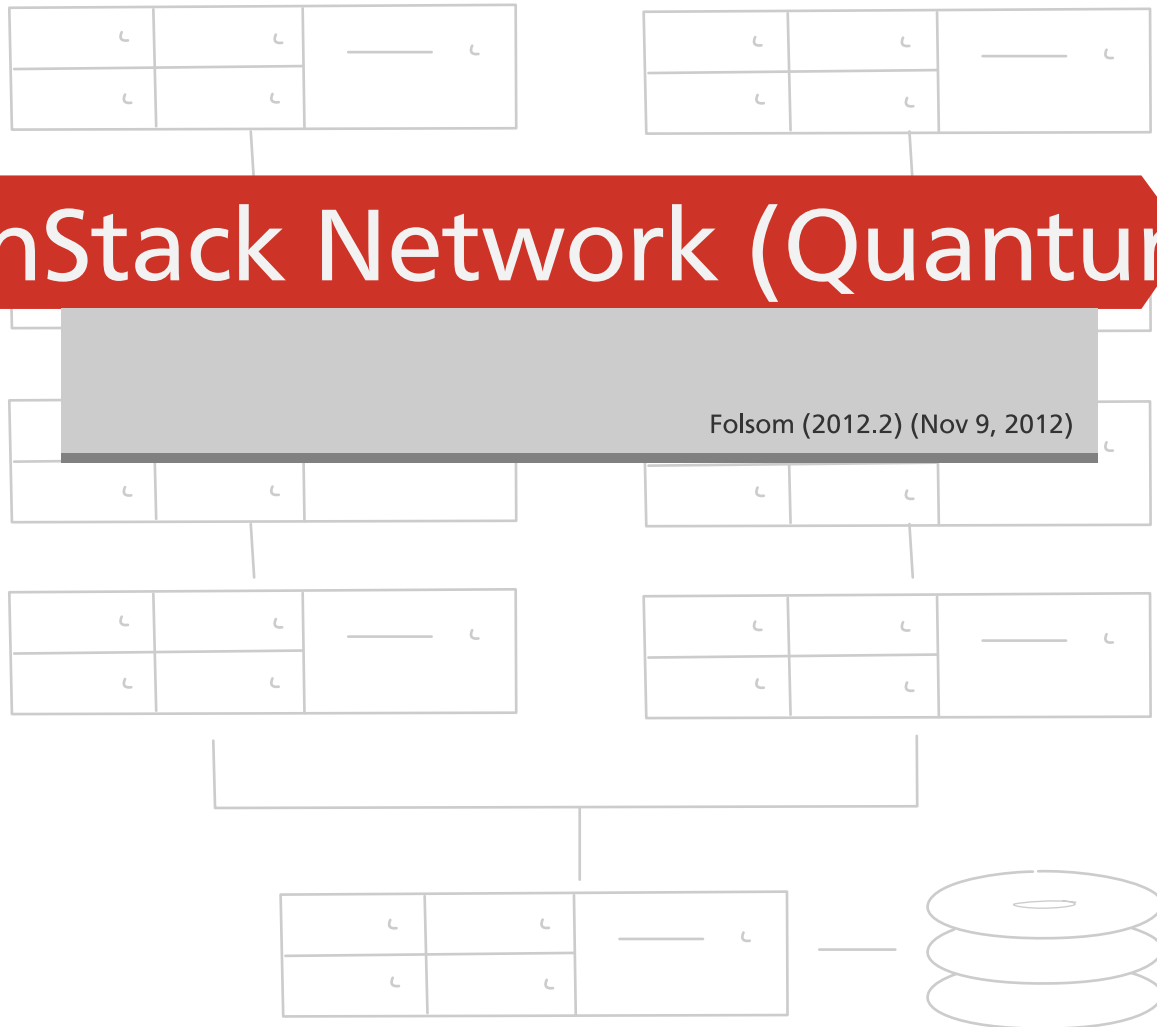


OpenStack Network (Quantum)



OpenStack Network (Quantum) Administration Guide

Folsom (2012.2) (2012-11-09)

Copyright © 2011, 2012 OpenStack All rights reserved.

This document is intended for administrators interested in running the OpenStack Quantum Virtual Network Service.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Table of Contents

Preface	vi
Intended Audience	vi
Document Change History	vi
Resources	vi
1. Overview	1
What is Quantum?	1
A Rich Network API	1
Flexibility to Choose Different Network Technologies	2
Quantum Architecture	2
Overview	2
Place Services on Physical Hosts	3
Network Connectivity for Physical Hosts	4
Quantum Deployment Use Cases	5
Use Case: Single Flat Network	5
Use Case: Multiple Flat Network	5
Use Case: Mixed Flat and Private Network	6
Use Case: Provider Router with Private Networks	7
Use Case: Per-tenant Routers with Private Networks	8
2. Quantum Installation	9
Install Packages (Ubuntu)	9
Install quantum-server	9
Install quantum-plugin-*-agent	11
Install quantum-dhcp-agent	12
Install quantum-l3-agent	12
Install quantum CLI Client	13
Init, Config, and Log File Locations	13
Installing Packages (Fedora)	13
RPC Setup	13
Install quantum-server and plugin	14
Install quantum-plugin-*-agent	14
Install quantum-dhcp-agent	15
Install quantum-l3-agent	15
Install quantum CLI client	15
Init, Config, and Log File Locations	15
3. Configuring other OpenStack Components	17
Keystone Configuration for Quantum	17
Running Nova with Quantum	18
Configuring Nova to reach the Quantum API	18
Configuring Vif-plugging in Nova	19
Example nova.conf (for nova-compute and nova-api)	20
4. Using Quantum	21
Core Quantum API Features	21
API Abstractions	21
Basic Workflow	23
Quantum CLI Guide	24
Admin API configuration	24
Advanced API Operations Examples	24
5. Advanced Features through API Extensions	26

Provider Networks	26
Terminology	26
Provider Attributes	27
Provider API Workflow	28
L3 Routing and NAT	29
L3 API Abstractions	29
Common L3 Workflow	30
Security Groups	32
Security Group API Abstractions	32
Common Security Group Commands	33
6. Advanced Configuration Options	35
Quantum Server with Plugin	35
DHCP Agent	36
Namespace	36
L3 Agent	37
Namespace	37
Multiple Floating IP Pools	38
Nova Metadata Server Support	39
7. Authentication and Authorization	41
8. Advanced Operational Features	45
Logging Settings	45
Notifications	45
Notification Options	45
Setting Cases	46
Quotas	47
Basic quota configuration	48
Per-tenant quota configuration	48
9. High Availability	52
Quantum High Availability with Pacemaker	52
10. Limitations	53
A. Demos Setup	55
Single Flat Network	55
Installations	57
Logical Network Configuration	58
Provider Router with Private Networks	62
Installations	64
Logical Network Configuration	66
Scale and HA of agents	70
Configuration	71
Commands in component extension	72
B. Core Configuration File Options	80
quantum.conf	80
ovs_quantum_plugin.ini:	83
linuxbridge_conf.ini:	84
dhcp_agent.ini	85
l3_agent.ini	86

List of Tables

4.1. Network	21
4.2. Subnet	22
4.3. Port	23
5.1. Provider Network Attributes	27
5.2. Router	29
5.3. Floating IP	29
5.4. Security Group Attributes	32
5.5. Security Group Rules	33
6.1. Settings	36
6.2. Basic settings	36
6.3. Basic settings	37
A.1. Nodes for Demo	56
A.2. Nodes for Demo	63
A.3. Hosts for Demo	71
B.1. Debugging Options	80
B.2. Logging Options	80
B.3. Service Options	80
B.4. Base Plugin Options	81
B.5. Common RPC Message Options	81
B.6. Rabbit RPC Options	81
B.7. QPID RPC Options	82
B.8. Notification Options	82
B.9. Quota Options	82
B.10. Database Access by Plugin	83
B.11. OVS Options Common to Plugin and Agent	83
B.12. Agent Options	84
B.13. Database Access by Plugin	84
B.14. VLAN Configurations	84
B.15. Networking Options on the Agent	84
B.16. Agent Options	85
B.17. DHCP Specific Options	85
B.18. Device Manager Options	85
B.19. dnsmasq Options	86
B.20. L3 Specific Options	86
B.21. Device Manager Options	86

Preface

The Quantum project was created to provide a rich and tenant-facing API for defining network connectivity and addressing in the cloud. The Quantum project gives operators the ability to leverage different networking technologies to power their cloud networking.

Intended Audience

This guide assists OpenStack administrators in leveraging different networking technologies to power their cloud networking. This document covers how to install, configure, and run Quantum.

The user should also have access to a plugin providing the implementation of the Quantum service. Several plugins are included in the Quantum distribution:

- **Openvswitch.** Implements Quantum with Open vSwitch for the KVM and XenServer compute platforms.
- **Cisco.** Implements Quantum for deployments by using Cisco UCS blades and Nexus switches for KVM compute platforms.
- **Linux Bridge** Implements Quantum with the Linux Bridge for the KVM compute platforms.
- **Nicira NVP.** Implements Quantum by using the Nicira Network Virtualization Platform (NVP) for KVM and XenServer compute platforms.
- **Ryu.** Implements Quantum by using the Ryu OpenFlow Controller for KVM compute platforms.
- **NEC OpenFlow.** Implements Quantum by using Trema Sliceable Switch OpenFlow Controller or NEC ProgrammableFlow Controller for KVM compute platforms.

Plugins can also be distributed separately from Quantum.

You should also be familiar with running the OpenStack Nova compute service as described in the Nova documentation.

Document Change History

The most recent changes are described in the table below:

Revision Date	Summary of Changes
Nov 9, 2012	<ul style="list-style-type: none">• Folsom release of this document.
Sep 18, 2012	<ul style="list-style-type: none">• First edition of this document.

Resources

For more information on Quantum and the other network-related projects, see the Quantum page on the OpenStack wiki (wiki.openstack.org/Quantum).

For information about programming against the Quantum API, see the [Quantum API Guide \(v2.0\)](#).

We welcome feedback, comments, and bug reports at bugs.launchpad.net/Quantum.

1. Overview

This chapter describes the high-level concepts and components of a Quantum deployment.

What is Quantum?

The Quantum project was created to provide a rich and tenant-facing API for defining network connectivity and addressing in the cloud. The Quantum project gives operators the ability to leverage different networking technologies to power their cloud networking.

For a detailed description of the Quantum API abstractions and their attributes, see the [Quantum API Guide \(v2.0\)](#).

A Rich Network API

Quantum is a virtual network service that provides a powerful API to define the network connectivity and addressing used by devices from other services, such as OpenStack Nova.

The OpenStack Nova API has a virtual server abstraction to describe compute resources. Similarly, the Quantum 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 Quantum 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 Nova to attach virtual devices to ports on these networks. In particular, Quantum 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 these capabilities to tenants directly, the Quantum API can be very useful even when used as an admin API because it provides significantly more flexibility for the cloud administrator to customize network offerings.

The Quantum API also provides a mechanism that lets cloud administrators expose additional API capabilities through API extensions. Commonly, new capabilities are first introduced as an API extension, and over time become part of the core Quantum API.

Flexibility to Choose Different Network Technologies

Enhancing traditional networking solutions to provide rich cloud networking is challenging. Traditional networking is not designed to scale to cloud proportions or to configure automatically.

The original Nova network implementation assumed a very basic model of performing all isolation through Linux VLANs and IP tables. Quantum introduces the concept of a *plugin*, which is a pluggable back-end implementation of the Quantum API. A plugin can use a variety of technologies to implement the logical API requests. Some Quantum plugins 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.

The current set of plugins include:

- **Open vSwitch.** <http://www.openvswitch.org/openstack/documentation>
- **Cisco.** quantum/plugins/cisco/README and <http://wiki.openstack.org/cisco-quantum>
- **Linux Bridge.** quantum/plugins/linuxbridge/README and <http://wiki.openstack.org/Quantum-Linux-Bridge-Plugin>
- **Nicira NVP.** quantum/plugins/nicira/nicira_nvp_plugin/README and <http://www.nicira.com/support>.
- **Ryu.** quantum/plugins/ryu/README and http://www.osrg.net/ryu/using_with_openstack.html
- **NEC OpenFlow.** <http://wiki.openstack.org/Quantum-NEC-OpenFlow-Plugin>

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

Quantum Architecture

This section describes the high-level components of a Quantum deployment.

Overview

Before you deploy Quantum, it is useful to understand the different components that make up the solution and how those components interact with each other and with other OpenStack services.

Quantum is a standalone service, just like other OpenStack services such as Nova, Glance, Keystone, and Horizon. Like those services, a deployment of Quantum often involves deploying several processes on a variety of hosts.

The main process of the Quantum Service is `quantum-server`, which is a Python daemon that exposes the Quantum API and passes user requests to the configured Quantum plugin for additional processing. Typically, the plugin requires access to a database for persistent storage, similar to other OpenStack services.

If your deployment uses a *controller host* to run centralized OpenStack Nova components, you can deploy Quantum on that same host. However, Quantum is entirely standalone and can be deployed on its own server as well. Quantum also includes additional agents that might be required depending on your deployment:

- **plugin agent** (`quantum-*agent`). Runs on each hypervisor to perform local vswitch configuration. Agent to be run depends on which plugin you are using, as some plugins do not require an agent.
- **dhcp agent** (`quantum-dhcp-agent`). Provides DHCP services to tenant networks. This agent is the same across all plugins.
- **I3 agent** (`quantum-l3-agent`). Provides L3/NAT forwarding to provide external network access for VMs on tenant networks. This agent is the same across all plugins.

These agents interact with the main `quantum-server` process in the following ways:

- Through RPC. For example, `rabbitmq` or `qpidd`.
- Through the standard Quantum API.

Quantum relies on the OpenStack Identity Project (Keystone) for authentication and authorization of all API request.

OpenStack Nova interacts with Quantum through calls to its standard API. As part of creating a VM, `nova-compute` communicates with the Quantum API to plug each virtual NIC on the VM into a particular Quantum network.

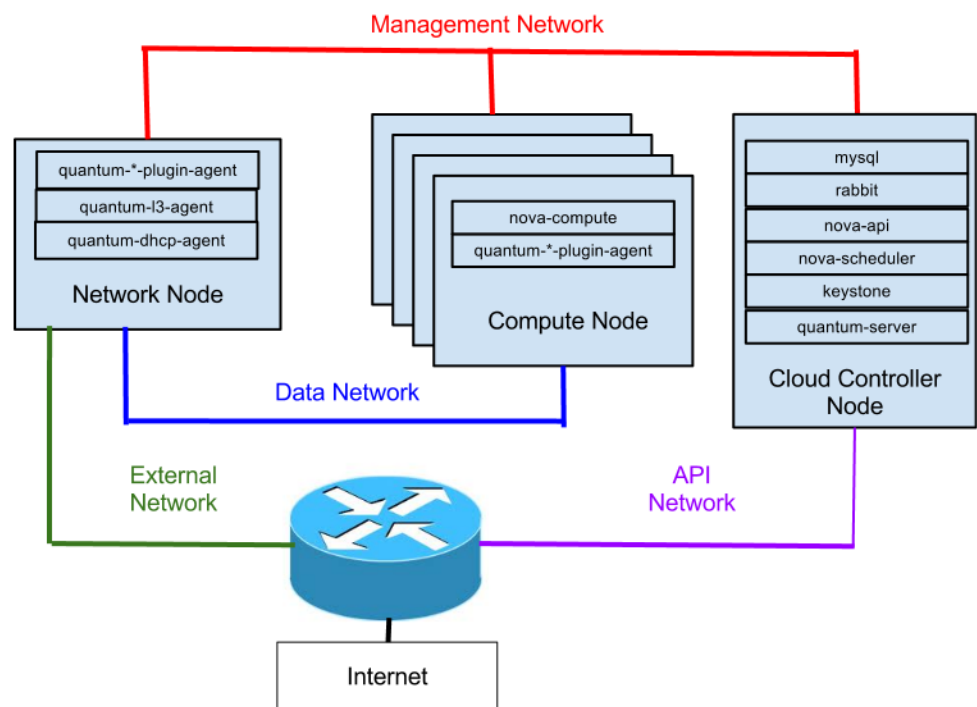
Horizon has basic integration with the Quantum API, allowing tenants to create networks and subnets through the GUI, and specify the set of networks that a VM should connect to when it is launched. See the *Horizon Administrator Guide*.

Place Services on Physical Hosts

Like other OpenStack services, Quantum provides cloud administrators with significant flexibility in deciding which individual services should run on which physical devices. One extreme, all services including Nova, Quantum, Keystone, and so on can be run on a single physical host for evaluation purposes. On the other, each service could have its own physical hosts, and some cases be replicated across multiple hosts for redundancy. See [Chapter 9, High Availability \[52\]](#)

In this guide, we focus primarily on a standard architecture that includes a “cloud controller” host, a “network gateway” host, and a set of hypervisors for running VMs. The “cloud controller” and “network gateway” can be combined in simple deployments, though if you expect VMs to send significant amounts of traffic to or from the Internet, a dedicated network gateway host is suggested to avoid potential CPU contention between packet forwarding performed by the `quantum-l3-agent` and other OpenStack services.

Network Connectivity for Physical Hosts



A standard Quantum setup has up to four distinct physical data center networks:

- **Management network.** Used for internal communication between OpenStack Components. The IP addresses on this network should be reachable only within the data center.
- **Data network.** Used for VM data communication within the cloud deployment. The IP addressing requirements of this network depend on the Quantum plugin in use.
- **External network.** Used to provide VMs with Internet access in some deployment scenarios. The IP addresses on this network should be reachable by anyone on the Internet.
- **API network.** Exposes all OpenStack APIs, including the Quantum API, to tenants. The IP addresses on this network should be reachable by anyone on the Internet. This may be the same network as the external network, as it is possible to create a quantum subnet for the external network that uses IP allocation ranges to use only less than the full range of IP addresses in an IP block.



Note

If the Nova metadata service is being used, any address space used on tenant networks must be routable on both the API network and on the External network, since the host running nova-api must be able to reply to HTTP requests with the un-SNATed IP address of a VM.

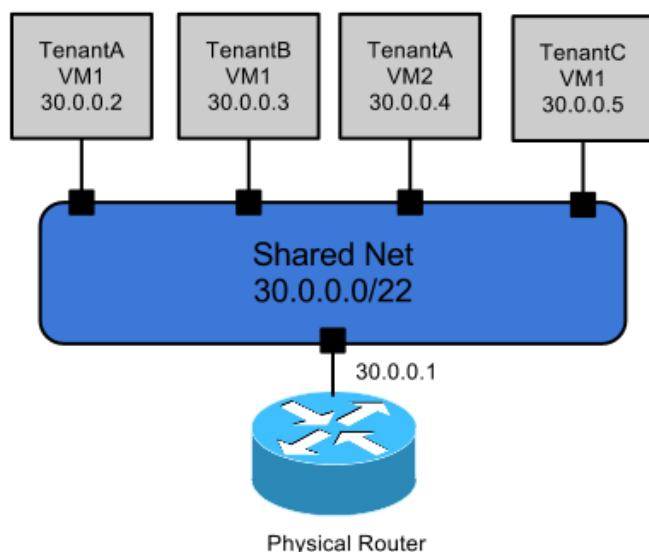
Quantum Deployment Use Cases

This section describes some of the common Quantum deployment use cases for Quantum. These examples are not exhaustive and can be combined with each other to create more complex use cases.

Use Case: Single Flat Network

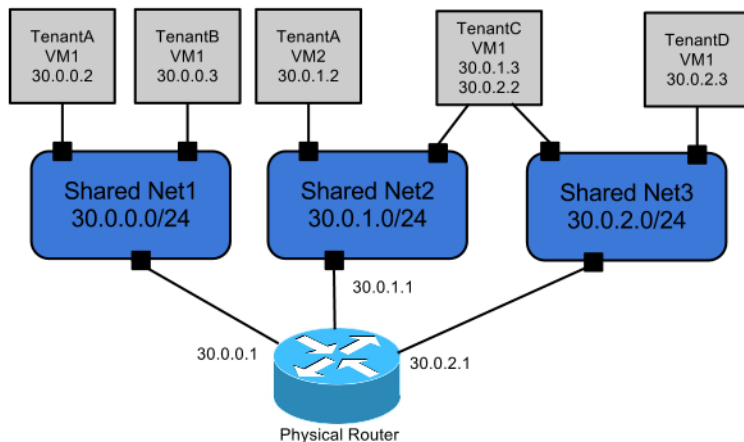
In the simplest use case, a single Quantum network exists. This is a "shared" network, meaning it is visible to all tenants via the Quantum API. Tenant VMs have a single NIC, and receive a fixed IP address from the subnet(s) associated with that network. This essentially maps to the FlatManager and FlatDHCPManager models provided by Nova. Floating IPs are not supported.

It is common that such a Quantum network is a "provider network", meaning it was created by the OpenStack administrator to map directly to an existing physical network in the data center. This allows the provider to use a physical router on that data center network as the gateway for VMs to reach the outside world. For each subnet on an external network, the gateway configuration on the physical router must be manually configured outside of OpenStack.



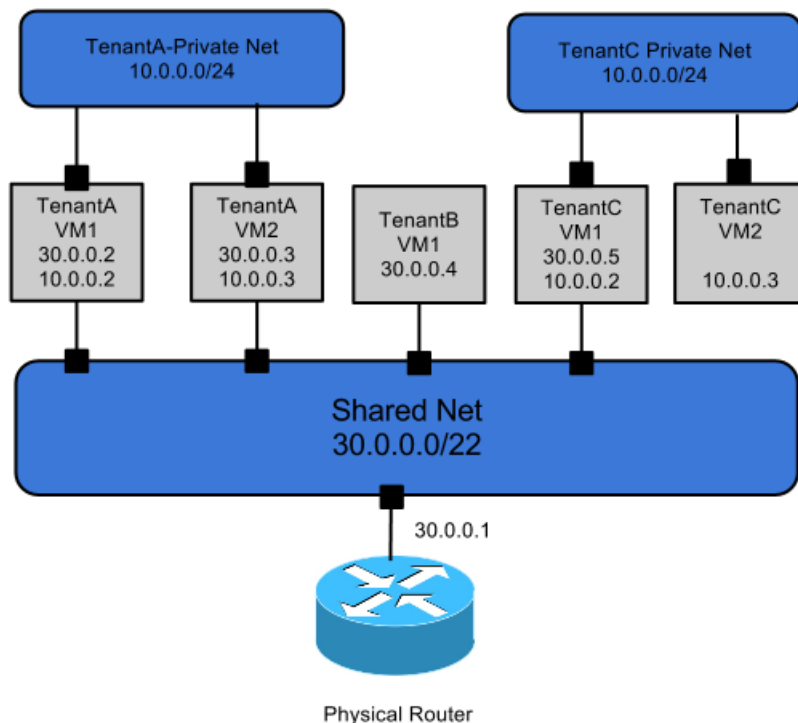
Use Case: Multiple Flat Network

This use case is very similar to the above Single Flat Network use case, except that tenants see multiple shared networks via the Quantum API and can choose which network (or networks) to plug into.



Use Case: Mixed Flat and Private Network

This use case is an extension of the above flat network use cases, in which tenants also optionally have access to private per-tenant networks. In addition to seeing one or more shared networks via the quantum API, tenants can create additional networks that are only visible to users of that tenant. When creating VMs, those VMs can have NICs on any of the shared networks and/or any of the private networks belonging to the tenant. This enables the creation of "multi-tier" topologies using VMs with multiple NICs. It also supports a model where a VM acting as a gateway can provide services such as routing, NAT, or load balancing.

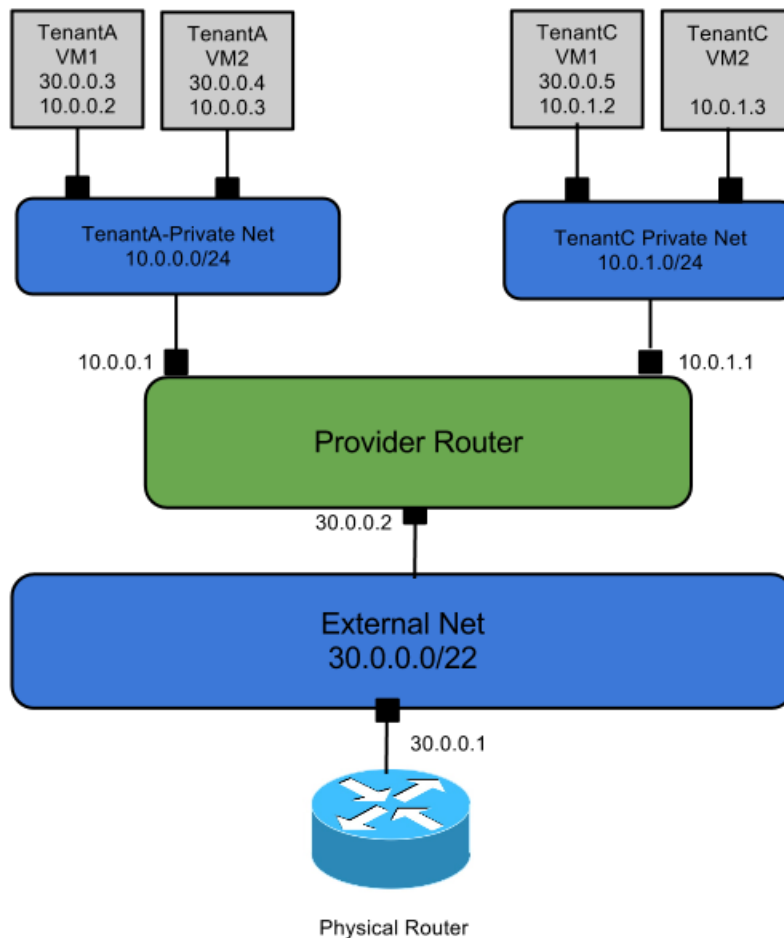


Use Case: Provider Router with Private Networks

This use provides each tenant with one or more private networks, which connect to the outside world via a Quantum router. The case where each tenant gets exactly one network in this form maps to the same logical topology as the VlanManager in Nova (of course, Quantum doesn't require VLANs). Using the Quantum API, the tenant would only see a network for each private network assigned to that tenant. The router object in the API is created and owned by the cloud admin.

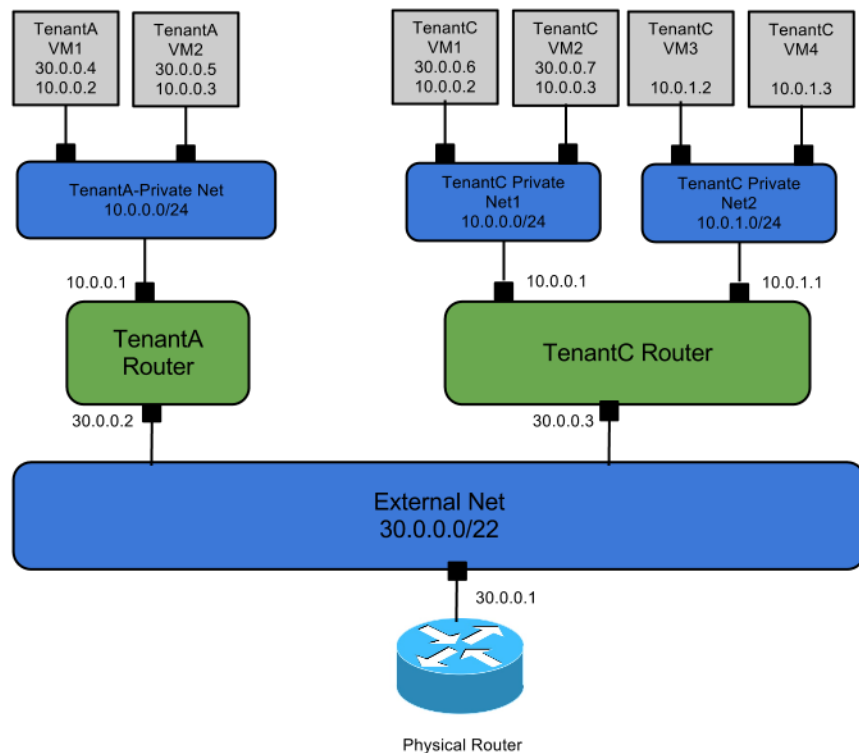
This model supports giving VMs public addresses using "floating IPs", in which the router maps public addresses from the external network to fixed IPs on private networks. Hosts without floating IPs can still create outbound connections to the external network, as the provider router performs SNAT to the router's external IP. The IP address of the physical router is used as the gateway_ip of the external network subnet, so the provider has a default router for Internet traffic.

The router provides L3 connectivity between private networks, meaning that different tenants can reach each others instances unless additional filtering (e.g., security groups) is used. Because there is only a single router, tenant networks cannot use overlapping IPs. Thus, it is likely that the admin would create the private networks on behalf of tenants.



Use Case: Per-tenant Routers with Private Networks

A more advanced router scenario in which each tenant gets at least one router, and potentially has access to the Quantum API to create additional routers. The tenant can create their own networks, potentially uplinking those networks to a router. This model enables tenant-defined multi-tier applications, with each tier being a separate network behind the router. Since there are multiple routers, tenant subnets can be overlapping without conflicting, since access to external networks all happens via SNAT or Floating IPs. Each router uplink and floating IP is allocated from the external network subnet.



2. Quantum Installation

This chapter describes how to install the Quantum Service and get it up and running.

If you are building a host from scratch to use for Quantum, we strongly recommend using Ubuntu 12.04/12.10 or Fedora 17/18 as these platforms have quantum packages and receive significant testing.

Quantum requires at least dnsmasq 2.59, to support all the options it requires.

Install Packages (Ubuntu)



Note

We are using Ubuntu cloud archive you can read more about it [Explanation of each possible sources.list entry can be found here: http://bit.ly/Q8OJ9M](http://bit.ly/Q8OJ9M)

Point to Folsom PPAs:

```
# echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/  
folsom main >> /etc/apt/sources.list.d/folsom.list  
# apt-get install ubuntu-cloud-keyring  
# apt-get update  
# apt-get upgrade
```



Note

Please use "sudo" in order to install and configure packages with superuser privileges.

Install quantum-server

Install quantum-server and CLI for accessing the API:

```
apt-get -y install quantum-server python-cliff python-pyparsing
```

You will also want to install the plugin you choose to use, for example:

```
apt-get -y install quantum-plugin-openvswitch
```

Most plugins require a database to be installed and configured in a plugin configuration file. For example:

```
apt-get -y install mysql-server python-mysqldb python-sqlalchemy
```

A database that you are already using for other OpenStack services will work fine for this. Simply create a 'quantum' database:

```
mysql -u <user> -p <pass> -e "create database quantum"
```

And then configure the plugin's configuration file to use this database. Find the plugin configuration file in `/etc/quantum/plugins` (For example, `/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini`) and set:


```
sql_connection = mysql://<user>:<password>@localhost/quantum?charset=utf8
```



Note

Different plugins can use different database schemas, so when switching a plugin, you must always drop the quantum database and start fresh.

RPC Setup

Quantum uses RPC to allow DHCP agents and any plugin agents to communicate with the main quantum-server process. Commonly, this can use the same RPC mechanism used by other OpenStack components like Nova.

To use RabbitMQ as the message bus for RPC, make sure that rabbit is installed on a host reachable via the management network (if this is already the case because of deploying another service like Nova, this existing RabbitMQ setup is sufficient):

```
apt-get install rabbitmq-server  
rabbitmqctl change_password guest <password>
```

Then update `/etc/quantum/quantum.conf` with these values:

```
rabbit_host=<mgmt-IP-of-rabbit-host>  
rabbit_password=<password>  
  
rabbit_userid=guest
```



Important

This `/etc/quantum/quantum.conf` file should be copied to and used on all hosts running quantum-server, quantum-dhcp-agent, quantum-openvswitch-agent, or quantum-linuxbridge-agent (see below).

Configuring Open vSwitch Plugin

Using the Open vSwitch (OVS) plugin in a deployment with multiple hosts requires the using of either tunneling or vlans in order to isolate traffic from multiple networks.

Tunneling is easier to deploy, as it does not require configuring VLANs on network switches, so that is what we describe here. More advanced deployment options are described in the ???

Edit `/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini` to specify the following values:

```
enable_tunneling=True  
tenant_network_type=gre  
tunnel_id_ranges=1:1000  
# only if node is running the agent  
local_ip=<data-net-IP-address-of-node>
```

After performing that change on the node running quantum-server, restart quantum-server to pick up the new settings.

```
service quantum-server restart
```

Configuring Nicira NVP Plugin

To configure Quantum to use the NVP plugin first edit `/etc/quantum/quantum.conf` and set:

```
core_plugin = quantum.plugins.nicira.nicira_nvp_plugin.QuantumPlugin.  
NvpPluginV2
```

Edit `/etc/quantum/plugins/nicira/nvp.ini` in order to configure the plugin. The quantum database created previously will be used by setting:

```
sql_connection = mysql://<user>:<password>@localhost/quantum?charset=utf8
```

In order to tell quantum about a controller cluster, create a `[CLUSTER:<name>]` section underneath the `[NVP]` section. Under this new cluster specify the Transport Zone by setting:

```
default_tz_uuid = <uuid_of_the_transport_zone>
```

Specify the controllers in this cluster by setting:

```
nvp_controller_connection =  
<ip>:<port>:<user>:<pw>:<req_timeout>:<http_timeout>:<retries>:<redirects>
```

one for each controller.

Lastly, restart quantum-server to pick up the new settings.

```
service quantum-server restart
```

Install quantum-plugin-* -agent

Some plugins utilize an agent that runs on each node that handles data packets. This includes any node running nova-compute, as well as nodes running dedicated quantum agents like quantum-dhcp-agent and quantum-l3-agent (see below). If your plugin uses an agent, this section describes how to run the agent for this plugin, as well as the basic configuration options.

Open vSwitch Agent

Install the OVS agent:

```
apt-get -y install quantum-plugin-openvswitch-agent
```

The `ovs_quantum_plugin.ini` created in the above step must be replicated on all nodes quantum-plugin-openvswitch-agent. When using tunneling, each node running quantum-plugin-openvswitch agent should have an IP address configured on the Data Network, and that IP address should be specified using the `local_ip` value in the `ovs_quantum_plugin.ini` file.

Then restart the agent

```
service quantum-plugin-openvswitch-agent restart
```

All hosts running quantum-plugin-openvswitch-agent also requires that an OVS bridge named "br-int" exists. To create it, run:

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

Install quantum-dhcp-agent

The host running quantum-server requires a network interface with an IP address on the "management network" and another interface on the "data network".

```
apt-get -y install quantum-dhcp-agent
```

Install the agent specific to the plugin (see plugin specific agent section above).

Install quantum-l3-agent

```
apt-get -y install quantum-l3-agent
```

Install the agent specific to the plugin (see plugin specific agent section above).

Create a bridge "br-ex" that will be used to uplink this node running quantum-l3-agent to the external network, then attach the NIC attached to the external network to this bridge.



Warning

OpenStack does not manage this routing for you, so you need to make sure that your host running the metadata service always has a route to reach each private network's subnet via the external network IP of that subnet's quantum router. To do this, you can run quantum without namespaces, and run the quantum-l3-agent on the same host as nova-api. Alternatively, you can identify an IP prefix that includes all private network subnet's (e.g., 10.0.0.0/8) and then make sure that your metadata server has a route for that prefix with the quantum router's external IP address as the next hop. For more validation information, refer to [Advanced configuration](#)

For example, with Open vSwitch and NIC eth1 connect to the external network, run:

```
ovs-vsctl add-br br-ex  
ovs-vsctl add-port br-ex eth1
```

The node running quantum-l3-agent should not have an IP address manually configured on the NIC connected to the external network. Rather, you must have a range of IP addresses from the external network that can be used by Quantum for routers that uplink to the external network. This range must be large enough to have an IP address for each router in the deployment, as well as each floating IP.

The quantum-l3-agent uses the Linux IP stack and iptables to perform L3 forwarding and NAT. In order to support multiple routers with potentially overlapping IP addresses, quantum-l3-agent defaults to using Linux network namespaces to provide isolated forwarding contexts. As a result, the IP addresses of routers will not be visible simply by running "ip addr list" or "ifconfig" on the node. Similarly, you will not be able to directly ping fixed IPs. To do either of these things, you must run the command within a particular router's network namespace. The namespace will have the name "qrouter-<UUID of the router>". The following commands are examples of running commands in the namespace of a router with UUID 47af3868-0fa8-4447-85f6-1304de32153b:

```
ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ip addr
list
ip netns exec qrouter-47af3868-0fa8-4447-85f6-1304de32153b ping <fixed-ip>
```

Install quantum CLI Client

Install the quantum CLI client:

```
apt-get -y install python-pyparsing python-cliff python-quantumclient
```

Init, Config, and Log File Locations

Services can be started and stopped using the 'service' command. For example:

```
service quantum-server stop
service quantum-server status
service quantum-server start
service quantum-server restart
```

Log files are found in /var/log/quantum.

Configuration files are in /etc/quantum.

Installing Packages (Fedora)

The OpenStack packages for Fedora can be retrieved from: <https://apps.fedoraproject.org/packages/s/openstack>.

RPC Setup

Quantum uses RPC to allow DHCP agents and any plugin agents to communicate with the main quantum-server process. Commonly, this can use the same RPC mechanism used by other OpenStack components like Nova.

To use QPID AMQP as the message bus for RPC, make sure that QPID is installed on a host reachable via the management network (if this is already the case because of deploying another service like Nova, this existing QPID setup is sufficient):

```
sudo yum -y install qpid-cpp-server qpid-cpp-server-daemon
sudo chkconfig qpid on
sudo service qpid start
```

Then update /etc/quantum/quantum.conf with these values:

```
rpc_backend = quantum.openstack.common.rpc.impl_qpid
qpid_hostname = <mgmt-IP-of-qpid-host>
```



Important

The Fedora packaging has a number of utility scripts that configure all of the necessary configuration files. The scripts can also be used to understand what needs to be configured for the specific Quantum services. The scripts

will be described below. Please note that the scripts make use of the package `openstack-utils`. Please install:

```
sudo yum install -y openstack-utils
```

Install quantum-server and plugin

Install `quantum-server` and plugin. **Note** the client is installed as a dependency for the Quantum service.

```
sudo yum install -y openstack-quantum
sudo yum install -y openstack-quantum-openvswitch
```

Most plugins require a database to be installed and configured in a plugin configuration file. The Fedora packaging for Quantum a server setup utility scripts that will take care of this. For example:

```
sudo quantum-server-setup --plugin openvswitch
```

Enable and start the service:

```
sudo systemctl enable quantum-server.service
sudo systemctl start quantum-server.service
```



Note

Different plugins can use different database schemas, so when switching a plugin, you must always drop the quantum database and start fresh.

Install quantum-plugin-*-agent

Some plugins utilize an agent that runs on each node that handles data packets. This includes any node running `nova-compute`, as well as nodes running dedicated quantum agents like `quantum-dhcp-agent` and `quantum-l3-agent` (see below). If your plugin uses an agent, this section describes how to run the agent for this plugin, as well as the basic configuration options.

Open vSwitch Agent

Install the OVS agent:

```
sudo yum install -y openstack-quantum-openvswitch
```

Run the agent setup script:

```
sudo quantum-node-setup --plugin openvswitch
```

All hosts running `quantum-plugin-openvswitch-agent` also requires that an OVS bridge named "br-int" exists. To create it, run:

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

Enable and start the agent:

```
sudo systemctl enable quantum-openvswitch-agent.service
```

```
sudo systemctl start quantum-openvswitch-agent.service
```

Install quantum-dhcp-agent

The DHCP agent is part of the openstack-quantum package.

```
sudo yum install -y openstack-quantum
```

Run the agent setup script:

```
sudo quantum-dhcp-setup --plugin openvswitch
```

Enable and start the agent:

```
sudo systemctl enable quantum-dhcp-agent.service  
sudo systemctl start quantum-dhcp-agent.service
```

Install quantum-l3-agent

The L3 agent is part of the openstack-quantum package.

Create a bridge "br-ex" that will be used to uplink this node running quantum-l3-agent to the external network, then attach the NIC attached to the external network to this bridge. For example, with Open vSwitch and NIC eth1 connect to the external network, run:

```
sudo ovs-vsctl add-br br-ex  
sudo ovs-vsctl add-port br-ex eth1
```

The node running quantum-l3-agent should not have an IP address manually configured on the NIC connected to the external network. Rather, you must have a range of IP addresses from the external network that can be used by Quantum for routers that uplink to the external network. This range must be large enough to have an IP address for each router in the deployment, as well as each floating IP.

```
sudo yum install -y openstack-quantum
```

Run the agent setup script:

```
sudo quantum-l3-setup --plugin openvswitch
```

Enable and start the agent:

```
sudo systemctl enable quantum-l3-agent.service  
sudo systemctl start quantum-l3-agent.service
```

Install quantum CLI client

Install the quantum CLI client:

```
sudo yum install -y python-quantumclient
```

Init, Config, and Log File Locations

Services can be started and stopped using the 'service' command. For example:

```
sudo service quantum-server stop
```

```
sudo service quantum-server status  
sudo service quantum-server start  
sudo service quantum-server restart
```

Log files are found in `/var/log/quantum`.

Configuration files are in `/etc/quantum`.

3. Configuring other OpenStack Components

Keystone Configuration for Quantum

Procedure 3.1. To Configure Keystone for Quantum

1. To Create a Quantum Service Entry

Quantum needs to be available in the Keystone service catalog. The steps for this depend on whether you are using Keystone's SQL catalog driver or the template catalog driver.

With the *SQL driver*, for a given region (\$REGION), IP address of the Quantum server (\$IP), and service ID (\$ID) returned by the Keystone service catalog, run:

```
keystone service-create --name quantum --type network --description 'OpenStack Networking Service'
```

Make a note of the ID returned by keystone and put it in the \$ID location.

```
keystone endpoint-create --region $REGION --service-id $ID --publicurl 'http://$IP:9696/' --adminurl 'http://$IP:9696/' --internalurl 'http://$IP:9696/'
```

Here's an example with real values:

```
$ keystone service-create --name quantum --type network --description 'OpenStack Networking Service'
```

Property	Value
description	OpenStack Networking Service
id	26a55b340e254ad5bb78c0b14391e153
name	quantum
type	network

```
$ keystone endpoint-create --region myregion --service-id 26a55b340e254ad5bb78c0b14391e153 \ --publicurl "http://10.211.55.17:9696/" --adminurl "http://10.211.55.17:9696/" --internalurl "http://10.211.55.17:9696/"
```

With the *template driver*, for a given region (\$REGION) and IP address of the Quantum server (\$IP), add the following content to your keystone catalog template file (default_catalog.templates).

```
catalog.$REGION.network.publicURL = http://$IP:9696  
catalog.$REGION.network.adminURL = http://$IP:9696  
catalog.$REGION.network.internalURL = http://$IP:9696  
catalog.$REGION.network.name = Network Service
```

Here is an example with real values:


```
catalog.$Region.network.publicURL = http://10.211.55.17:9696
catalog.$Region.network.adminURL = http://10.211.55.17:9696
catalog.$Region.network.internalURL = http://10.211.55.17:9696
catalog.$Region.network.name = Network Service
```

2. Create Quantum Service User

For Nova to speak to the Quantum API, and for some internal components of Quantum to communicate with the Quantum API, you need to provide them with admin user credentials that they can use when accessing the Quantum API. The suggested approach is to create a special 'service' tenant, create a 'quantum' user within this tenant, and to assign this user an 'admin' role. Kindly check the ID for user, role and tenant.

For example:

```
$ ADMIN_ROLE=$(get_id keystone role-create --name=admin)

$ QUANTUM_USER=$(get_id keystone user-create --name=quantum --pass=
"$QUANTUM_PASSWORD" --email=demo@example.com --tenant-id service)

$ keystone user-role-add --user_id $QUANTUM_USER --role_id $ADMIN_ROLE
--tenant_id service
```

See the Keystone Administrator Guide for more details about creating service entries and service users.

Running Nova with Quantum

Unlike traditional Nova deployments, when Quantum is in use, Nova should not run a nova-network. Instead, Nova delegates almost all of the network-related decisions to Quantum. This means many of the network-related CLI command and configuration options you are familiar with from using Nova do not work with Quantum.

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. If a Nova CLI command or configuration option related to networking is not mentioned in this guide, it is likely not supported for use with Quantum. 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 Quantum.



Note

It is strongly recommended that you uninstall nova-network and reboot any physical nodes that had been running nova-network before using them to run Quantum. Inadvertently running the nova-network process while using Quantum can cause problems, as can stale iptables rules pushed down by previously running nova-network.

Configuring Nova to reach the Quantum API

Each time a VM is provisioned or deprovisioned in Nova, Nova communicates with Quantum via the standard API. To do so, it requires the following items in the nova.conf used by each nova-compute and nova-api instance:

- `network_api_class`: must be modified from default to `nova.network.quantumv2.api.API` indicate that Quantum should be used rather than the traditional `nova-network` networking model.
- `quantum_url`: must include the hostname/IP and port of the Quantum server for this deployment.
- `quantum_auth_strategy`: should be kept as default 'keystone' for all production deployments.
- `quantum_admin_tenant_name`: must be modified to be the name of the service tenant created in the above section on Keystone configuration.
- `quantum_admin_username`: must be modified to be the name of the user created in the above section on Keystone configuration.
- `quantum_admin_password`: must be modified to be the password of the user created in the above section on Keystone configuration.
- `quantum_admin_auth_url`: must be modified to point to the keystone server IP and port. This is the Identity (keystone) admin API server IP and port value, and not the Identity service API IP and port.

Configuring Vif-plugging in Nova

When `nova-compute` creates a VM, it must "plug" each of the VM's vNICs into a Quantum controlled virtual switch, and inform the virtual switch about the Quantum port-id associated with each vNIC. This is done by specifying a field in the `nova.conf` of the `nova-compute` instance indicating what type of "vif-plugging" should be used. The exact field(s) you need to set depend on your plugin. For plugins not listed below, see the plugin's own documentation.

Vif-plugging with Open vSwitch Plugin

The choice of vif-plugging for the Open vSwitch plugin depends on what version of `libvirt` you are using, as well as whether you are using Nova security filtering (i.e., security groups, provider firewall, VM spoofing prevention).

- When using `libvirt` (any version) with Nova security filtering:
`libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver`
- When using `libvirt` (version < 0.9.11) without Nova security filtering:
`libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver`
- When using `libvirt` (version >= 0.9.11) without Nova security groups:
`libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver`



Checking your libvirt version

To check your `libvirt` version, use `libvirtd --version`.

Vif-plugging with Linux Bridge Plugin

- When using `libvirt` (any version):
`libvirt_vif_driver=nova.virt.libvirt.vif.QuantumLinuxBridgeVIFDriver`

Vif-plugging with Nicira NVP Plugin

The choice of vif-plugging for the NVP Plugin depends on what version of libvirt you are using (this assumes you are using NVP for security groups and VM spoof prevention).

- When using libvirt (version < 0.9.11):
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver
- When using libvirt (version >= 0.9.11):
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver
- When using XenServer: xenapi_vif_driver=nova.virt.xenapi.vif.XenAPIOpenVswitchDriver

Vif-plugging with NEC OpenFlow Plugin

The choice of vif-plugging for the NEC plugin depends on what version of libvirt you are using, as well as whether you are using Nova security filtering (i.e., security groups, provider firewall, VM spoofing prevention).

- When using libvirt (any version) with Nova security filtering:
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybirdOVSBridgeDriver
- When using libvirt (version < 0.9.11) without Nova security filtering:
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver
- When using libvirt (version >= 0.9.11) without Nova security groups:
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchVirtualPortDriver

Example nova.conf (for nova-compute and nova-api)

Example values for the above settings, assuming a cloud controller node running Keystone and Quantum with an IP address of 192.168.1.2 and vif-plugging using the LibvirtOpenVswitchDriver with virtio enabled.

```
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.1.2:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.1.2:35357/v2.0

# needed only for nova-compute
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtOpenVswitchDriver
libvirt_use_virtio_for_bridges=True
```

4. Using Quantum

There are two main approaches to using Quantum. The first is to expose the Quantum API to cloud tenants, allowing them to build rich network topologies. The second is to have the cloud administrator, or another tool run the the cloud admin, create network connectivity on behalf of tenants. In this document, we commonly describe operations as being performed by a tenant, but they might also be performed by the cloud admin on behalf of the tenant.

Core Quantum API Features

Once Quantum is installed and running, both tenants and admins primarily interact with the service via create-read-update-delete (CRUD) API operations performed either directly against the API, or more commonly via the 'quantum' CLI tool. Like other OpenStack CLI tools, the 'quantum' tool is just a basic wrapper around the Quantum API, so any operation that can be performed via the CLI has an equivalent API call that can be performed programmatically.

The CLI supports many options for filtering results, limiting fields show, etc. For details, refer to the Quantum CLI documentation.

API Abstractions

The Quantum v2.0 API provides control over both L2 network topologies and the IP addresses used on those networks (IP Address Management or IPAM). There is also an extension to cover basic L3 forwarding and NAT, providing capabilities similar to nova-network.

In the Quantum API, an isolated L2 network segment (similar to a VLAN) is called a 'Network' and forms the basis for describing the L2 network topology available in a given Quantum deployment.

The Quantum API uses the notion of a 'Subnet', which associates a block of IP addresses and other network configuration (e.g., default gateway, dns-servers) with a Quantum Network. Each subnet represents an IPv4 or IPv6 address block and each Quantum Network can have multiple subnets, if desired.

A 'Port' represents an attachment port to a L2 Quantum Network. When a port is created on the network, by default it will be allocated an available fixed IP address out of one of the designated Subnets for each IP version (if one exists). When the Port is destroyed, the allocated addresses return to the pool of available IPs on the subnet(s). Users of the Quantum API can either choose a specific IP address from the block, or let Quantum choose the first available IP address.

The table below summarizes the attributes available for each of these abstractions. For more operations about API abstraction and operations, please refer to the [Quantum v2.0 API Developer Guide](#).

Table 4.1. Network

Attribute name	Type	Default Value	Description
id	uuid-str	generated	UUID for the network.

Attribute name	Type	Default Value	Description
name	String	None	Human-readable name for the network. Might not be unique.
admin_state_up	Bool	True	The administrative state of network. If false (down), the network does not forward packets.
status	String	N/A	Indicates whether network is currently operational.
subnets	list(uuid-str)	Empty list	Subnets associated with this network.
shared	Bool	False	Specifies whether the network resource can be accessed by any tenant or not.
tenant_id	uuid-str	N/A	Owner of network. Only admin users can specify a tenant_id other than its own.

Table 4.2. Subnet

Attribute	Type	Default Value	Description
id	uuid-str	generated	UUID representing the subnet
network_id	uuid-str	N/A	network this subnet is associated with.
name	String	None	Human-readable name for the subnet. Might not be unique.
ip_version	int	4	IP version
cidr	string	N/A	cidr representing IP range for this subnet, based on IP version
gateway_ip	string	first address in <i>cidr</i>	default gateway used by devices in this subnet
dns_nameservers	list(str)	Empty list	DNS name servers used by hosts in this subnet.
allocation_pools	list(dict)	Every address in <i>cidr</i> , excluding <i>gateway_ip</i> if configured	Sub-ranges of cidr available for dynamic allocation to ports [{ "start": "10.0.0.2", "end": "10.0.0.254" }]
host_routes	list(dict)	Empty List	Routes that should be used by devices with IPs from this subnet (not including local subnet route).
enable_dhcp	Bool	True	Specifies whether DHCP is enabled for this subnet or not.
tenant_id	uuid-str	N/A	Owner of network. Only admin users can specify a tenant_id other than its own.

Table 4.3. Port

Attribute	Type	Default Value	Description
id	uuid-str	generated	UUID for the port.
network_id	uuid-str	N/A	Network that this port is associated with.
name	String	None	Human-readable name for the port. Might not be unique.
admin_state_up	bool	true	Administrative state of port. If false (down), port does not forward packets.
status	string	N/A	Indicates whether network is currently operational.
mac_address	string	generated	Mac address to use on this port.
fixed_ips	list(dict)	automatically allocated from pool	Specifies IP addresses for the port thus associating the port itself with the subnets where the IP addresses are picked from
device_id	str	None	identifies the device (e.g., virtual server) using this port.
device_owner	str	None	Identifies the entity (e.g.: dhcp agent) using this port.
tenant_id	uuid-str	N/A	Owner of network. Only admin users can specify a tenant_id other than its own.

Basic Workflow

(include output in examples)

```
quantum net-create net1
quantum subnet-create net1 10.0.0.0/24
```

View current networks:

```
quantum net-list
```

Boot the VM with a single NIC on the selected network:

```
nova boot --image <img> --flavor <flavor> --nic net-id=<net-id> <vm-name>
```

Congrats, you have booted a VM on a quantum network.

There is now a Quantum port on 'net1' that corresponds to the VM Nic. You can view it with the following command, which searches for all ports with a "device_id" corresponding to the Nova instance UUID:

```
quantum port-list -- --device_id=<vm-id>
```

To view only a few fields of the port, you can limit output using -c. For example to see only the mac_address of the port, use:

```
quantum port-list -c mac_address -- --device_id=<vm-id>
```

You could temporarily disable the port from sending traffic by updating it to have `admin_state_up=False`:

```
quantum port-update <port-id> --admin_state_up=False
```

When we delete the Nova VM, the underlying Quantum port is automatically deleted:

```
quantum port-list -c mac_address -- --device_id=<vm-id>
```

Quantum CLI Guide

Before going further, we **STRONGLY** suggest that you quickly read the few pages in the [OpenStack CLI Guide](#) that are specific to Quantum. Quantum's CLI has some advanced capabilities that are described only in that guide.

Admin API configuration

These same calls can be performed by the cloud admin on half of the tenants by specifying a `tenant_id` in the request, for example:

```
quantum net-create net1 --tenant_id=<tenant-id>
```

This `tenant_id` should be the tenant ID from keystone. To view all keystone tenant IDs, run the following command as a keystone admin user:

```
keystone tenant-list
```

Advanced API Operations Examples

Network, Subnet & Port Creation

- Create a "shared" network (i.e., a network that can be used by all tenants)

```
quantum net-create public-net --shared True
```

- Create a subnet that has a specific gateway IP address.

```
quantum subnet-create --gateway 10.0.0.254 net1 10.0.0.0/24
```

- Create a subnet that has no gateway IP address.

```
quantum subnet-create --no-gateway net1 10.0.0.0/24
```

- Create a subnet for which DHCP is disabled.

```
quantum subnet-create net1 10.0.0.0/24 --enable_dhcp False
```

- Create subnet with a specific set of host routes:

```
quantum subnet-create test-net1 40.0.0.0/24 --host_routes type=dict list=true destination=40.0.1.0/24,nexthop=40.0.0.2
```

- Create subnet with a specific set of dns nameserver:

```
quantum subnet-create test-net1 40.0.0.0/24 --dns_nameservers list=true 8.8.8.7 8.8.8.8
```

Searches

- Find all Ports/IPs allocated on a network

```
quantum port-list -- --network_id <net-id>
```

Advanced VM creation & port operations.

- booting a VM with multiple NICs

```
nova boot --image <img> --flavor <flavor> --nic net-id=<net1-id> --nic net-id=<net2-id> <vm-name>
```

- specifying a port-id, rather than a net-id

```
quantum port-create --fixed-ip subnet_id=<subnet-id>,ip_address=192.168.57.101 net1  
nova boot --image <img> --flavor <flavor> --nic port-id=<port-id> <vm-name>
```

- booting a VM with no `--nic` option specified. In this case the launched VM connects to all networks that are accessible to the tenant who submits the request.

```
nova boot --image <img> --flavor <flavor> <vm-name>
```


5. Advanced Features through API Extensions

This section discusses two API extensions implemented by several plugins. We include them in this guide as they provide capabilities similar to what was available in nova-network and are thus likely to be relevant to a large portion of the OpenStack community.

Provider Networks

Provider networks allow cloud administrators to create Quantum networks that map directly to physical networks in the data center. This is commonly used to give tenants direct access to a "public" network that can be used to reach the Internet. It may also be used to integrate with VLANs in the network that already have a defined meaning (e.g., allow a VM from the "marketing" department to be placed on the same VLAN as bare-metal marketing hosts in the same data center).

The provider extension allows administrators to explicitly manage the relationship between Quantum virtual networks and underlying physical mechanisms such as VLANs and tunnels. When this extension is supported, Quantum client users with administrative privileges see additional provider attributes on all virtual networks, and are able to specify these attributes in order to create provider networks.

As of the Folsom release, the provider extension is supported by the openvswitch and linuxbridge plugins. Configuration of these plugins requires familiarity with this extension.

Terminology

A number of terms are used in the provider extension and in the configuration of plugins supporting the provider extension:

- **virtual network** - A Quantum L2 network (identified by a UUID and optional name) whose ports can be attached as vNICs to Nova instances and to various Quantum agents. The openvswitch and linuxbridge plugins each support several different mechanisms to realize virtual networks.
- **physical network** - A network connecting virtualization hosts (i.e. Nova compute nodes) with each other and with other network resources. Each physical network may support multiple virtual networks. The provider extension and the plugin configurations identify physical networks using simple string names.
- **tenant network** - A "normal" virtual network created by/for a tenant. The tenant is not aware of how that network is physically realized.
- **provider network** - A virtual network administratively created to map to a specific network in the data center, typically to enable direct access to non-OpenStack resources on that network. Tenants can be given access to provider networks.
- **VLAN network** - A virtual network realized as packets on a specific physical network containing IEEE 802.1Q headers with a specific VID field value. VLAN networks sharing

the same physical network are isolated from each other at L2, and can even have overlapping IP address spaces. Each distinct physical network supporting VLAN networks is treated as a separate VLAN trunk, with a distinct space of VID values. Valid VID values are 1 through 4094.

- **flat network** - A virtual network realized as packets on a specific physical network containing no IEEE 802.1Q header. Each physical network can realize at most one flat network.
- **local network** - A virtual network that allows communication within each host, but not across a network. Local networks are intended mainly for single-node test scenarios, but may have other uses.
- **GRE network** - A virtual network realized as network packets encapsulated using GRE. GRE networks are also referred to as "tunnels". GRE tunnel packets are routed by the host's IP routing table, so GRE networks are not associated by Quantum with specific physical networks.

Both the openvswitch and linuxbridge plugins support VLAN networks, flat networks, and local networks. Only the openvswitch plugin currently supports GRE networks, provided that the host's Linux kernel supports the required Open vSwitch features.

Provider Attributes

The provider extension extends the Quantum network resource with the following three additional attributes:

Table 5.1. Provider Network Attributes

Attribute name	Type	Default Value	Description
provider:network_type	String	N/A	The physical mechanism by which the virtual network is realized. Possible values are "flat", "vlan", "local", and "gre", corresponding to flat networks, VLAN networks, local networks, and GRE networks as defined above. All types of provider networks can be created by administrators, while tenant networks can be realized as "vlan", "gre", or "local" network types depending on plugin configuration.
provider:physical_network	String	If a physical network named "default" has been configured, and if provider:network_type is "flat" or "vlan", then "default" is used.	The name of the physical network over which the virtual network is realized for flat and VLAN networks. Not applicable to the "local" or "gre" network types.
provider:segmentation_id	Integer	N/A	For VLAN networks, the VLAN VID on the physical network that realizes the virtual network. Valid VLAN VIDs are 1 through 4094.

Attribute name	Type	Default Value	Description
			For GRE networks, the tunnel ID. Valid tunnel IDs are any 32 bit unsigned integer. Not applicable to the "flat" or "local" network types.

The provider attributes are returned by Quantum network API operations when the client is authorized for the `extension:provider_network:view` action via the Quantum policy configuration. The provider attributes are only accepted for network API operations if the client is authorized for the `extension:provider_network:set` action. The default Quantum API policy configuration authorizes both actions for users with the admin role. See [Chapter 7, Authentication and Authorization \[41\]](#) for details on policy configuration.

Provider API Workflow

Show all attributes of a network, including provider attributes when invoked with the admin role:

```
quantum net-show <name or net-id>
```

Create a local provider network (admin-only):

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type  
local
```

Create a flat provider network (admin-only):

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type flat  
--provider:physical_network <phys-net-name>
```

Create a VLAN provider network (admin-only):

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type vlan  
--provider:physical_network <phys-net-name> --provider:segmentation_id <VID>
```

Create a GRE provider network (admin-only):

```
quantum net-create <name> --tenant_id <tenant-id> --provider:network_type gre  
--provider:segmentation_id <tunnel-id>
```

When creating flat networks or VLAN networks, `<phys-net-name>` must be known to the plugin. See [the section called "ovs_quantum_plugin.ini:" \[83\]](#) and [the section called "linuxbridge_conf.ini:" \[84\]](#) for details on configuring `network_vlan_ranges` to identify all physical networks. When creating VLAN networks, `<VID>` can fall either within or outside any configured ranges of VLAN IDs from which tenant networks are allocated. Similarly, when creating GRE networks, `<tunnel-id>` can fall either within or outside any tunnel ID ranges from which tenant networks are allocated.

Once provider networks have been created, subnets can be allocated and they can be used similarly to other virtual networks, subject to authorization policy based on the specified `<tenant_id>`.

L3 Routing and NAT

Just like the core Quantum API provides abstract L2 network segments that are decoupled from the technology used to implement the L2 network, Quantum includes an API extension that provides abstract L3 routers that API users can dynamically provision and configure. These Quantum routers can connect multiple L2 Quantum networks, and can also provide a "gateway" that connects one or more private L2 networks to a shared "external" network (e.g., a public network for access to the Internet). See [the section called "Use Case: Provider Router with Private Networks" \[7\]](#) and [the section called "Use Case: Per-tenant Routers with Private Networks" \[8\]](#) for details on common models of deploying Quantum L3 routers.

The L3 router provides basic NAT capabilities on "gateway" ports that uplink the router to external networks. This router SNATs all traffic by default, and supports "Floating IPs", which creates a static one-to-one mapping from a public IP on the external network to a private IP on one of the other subnets attached to the router. This allows a tenant to selectively expose VMs on private networks to other hosts on the external network (and often to all hosts on the Internet). Floating IPs can be allocated and then mapped from one Quantum port to another, as needed.

L3 API Abstractions

Table 5.2. Router

Attribute name	Type	Default Value	Description
id	uuid-str	generated	UUID for the router.
name	String	None	Human-readable name for the router. Might not be unique.
admin_state_up	Bool	True	The administrative state of router. If false (down), the router does not forward packets.
status	String	N/A	Indicates whether router is currently operational.
tenant_id	uuid-str	N/A	Owner of the router. Only admin users can specify a tenant_id other than its own.
external_gateway_info	dict contain 'network_id' key-value pair	Null	External network that this router connects to for gateway services (e.g., NAT)

Table 5.3. Floating IP

Attribute name	Type	Default Value	Description
id	uuid-str	generated	UUID for the floating IP.
floating_ip_address	string (IP address)	allocated by Quantum	The external network IP address available to be mapped to an internal IP address.
floating_network_id	uuid-str	N/A	The network indicating the set of subnets from which

Attribute name	Type	Default Value	Description
			the floating IP should be allocated
router_id	uuid-str	N/A	Read-only value indicating the router that connects the external network to the associated internal port, if a port is associated.
port_id	uuid-str	Null	Indicates the internal Quantum port associated with the external floating IP.
fixed_ip_address	string (IP address)	Null	Indicates the IP address on the internal port that is mapped to by the floating IP (since a Quantum port might have more than one IP address).
tenant_id	uuid-str	N/A	Owner of the Floating IP. Only admin users can specify a tenant_id other than its own.

Common L3 Workflow

Create external networks (admin-only)

```
quantum net-create public --router:external=True  
quantum subnet-create public 172.16.1.0/24
```

Viewing external networks:

```
quantum net-list -- --router:external=True
```

Creating routers

Internal-only router to connect multiple L2 networks privately.

```
quantum net-create net1  
quantum subnet-create net1 10.0.0.0/24  
quantum net-create net2  
quantum subnet-create net2 10.0.1.0/24  
quantum router-create router1  
quantum router-interface-add router1 <subnet1-uuid>  
quantum router-interface-add router1 <subnet2-uuid>
```

The router will get an interface with the gateway_ip address of the subnet, and this interface will be attached to a port on the L2 Quantum network associated with the subnet. The router will also get an gateway interface to the specified external network. This will provide SNAT connectivity to the external network as well as support for floating IPs allocated on that external networks (see below). Commonly an external network maps to a network in the provider

A router can also be connected to an "external network", allowing that router to act as a NAT gateway for external connectivity.

```
quantum router-gateway-set router1 <ext-net-id>
```

Viewing routers:

List all routers:

```
quantum router-list
```

Show a specific router:

```
quantum router-show <router_id>
```

Show all internal interfaces for a router:

```
quantum port-list -- --device_id=<router_id>
```

Associating / Disassociating Floating IPs:

First, identify the port-id representing the VM NIC that the floating IP should map to:

```
quantum port-list -c id -c fixed_ips -- --device_id=ZZZ
```

This port must be on a Quantum subnet that is attached to a router uplinked to the external network that will be used to create the floating IP. Conceptually, this is because the router must be able to perform the Destination NAT (DNAT) rewriting of packets from the Floating IP address (chosen from a subnet on the external network) to the internal Fixed IP (chosen from a private subnet that is “behind” the router).

Create floating IP unassociated, then associate

```
quantum floatingip-create <ext-net-id>  
quantum floatingip-associate <floatingip-id> <internal VM port-id>
```

create floating IP and associate in a single step

```
quantum floatingip-create --port_id <internal VM port-id> <ext-net-id>
```

Viewing Floating IP State:

```
quantum floatingip-list
```

Find floating IP for a particular VM port:

```
quantum floatingip-list -- --port_id=ZZZ
```

Disassociate a Floating IP:

```
quantum floatingip-disassociate <floatingip-id>
```

L3 Tear Down

Delete the Floating IP:

```
quantum floatingip-delete <floatingip-id>
```

Then clear the any gateway:

```
quantum router-gateway-clear router1
```

Then remove the interfaces from the router (deleting the network and subnet will do this as well):

```
quantum router-interface-remove router1 <subnet-id>
```

Finally, delete the router:

```
quantum router-delete router1
```

Security Groups

Security groups and security group rules allows administrators and tenants the ability to specify the type of traffic and direction (ingress/egress) that is allowed to pass through a port. A security group is a container for security group rules.

When a port is created in quantum it is associated with a security group. If a security group is not specified the port will be associated with a default security group. By default this group will drop all ingress traffic and allow all egress. Rules can be added to this group in order to change the behaviour.

In order to use nova security groups in quantum, configure the following files. In `/etc/quantum/quantum.conf`, set `proxy_mode True` within the `SECURITYGROUP` section. Ensure that the `quantum-server /etc/quantum/quantum.conf` file has `proxy_mode=False` so that the quantum API service does not work as a proxy for nova API calls. Then in `/etc/nova/nova.conf` set `security_group_handler` to be `nova.network.sg.SecurityGroupHandlerQuantumProxy` and `quantum_port_security=True` in the `DEFAULT` section. Once the files are changed, restart `quantum-server`, `nova-api`, and the `nova-compute` services in order to pick up the changes.



Note

If `quantum-server` is set to `proxy_mode=True` then you will not be able to use security groups through the quantum api and you will have to control security groups through nova configuration and commands.

Security Group API Abstractions

Table 5.4. Security Group Attributes

Attribute name	Type	Default Value	Description
id	uuid-str	generated	UUID for the security group.
name	String	None	Human-readable name for the security group. Might not be unique. Cannot be named default as that is automatically created for a tenant.
description	String	None	Human-readable description of a security group.
external_id	Integer	N/A	This is used to coordinate nova security groups with quantum security groups for backwards compatibility.
tenant_id	uuid-str	N/A	Owner of the security group. Only admin users can specify a <code>tenant_id</code> other than their own.

Table 5.5. Security Group Rules

Attribute name	Type	Default Value	Description
id	uuid-str	generated	UUID for the security group rule.
security_group_id	uuid-str or Integer	allocated by Quantum	The security group to associate rule with.
direction	String	N/A	The direction the traffic is allow (ingress/egress) from a VM.
protocol	String	None	IP Protocol (icmp, tcp, udp, etc).
port_range_min	Integer	None	Port at start of range
port_range_max	Integer	None	Port at end of range
ethertype	String	None	ethertype in L2 packet (IPv4, IPv6, etc)
source_ip_prefix	string (IP cidr)	None	CIDR for address range
source_group_id	uuid-str or Integer	allocated by Quantum or Nova	Source security group to apply to rule.
external_id	Integer	N/A	This is used to coordinate nova security groups with quantum security groups for backwards compatibility.
tenant_id	uuid-str	N/A	Owner of the security group rule. Only admin users can specify a tenant_id other than its own.

Common Security Group Commands

Create a security group for our web servers:

```
quantum security-group-create webservers --description "security group for webservers"
```

Viewing security groups:

```
quantum security-group-list
```

Creating security group rule to allow port 80 ingress:

```
quantum security-group-rule-create --direction ingress --protocol tcp --port_range_min 80 --port_range_max 80 <security_group_uuid>
```

List security group rules:

```
quantum security-group-rule-list
```

Delete a security group rule:

```
quantum security-group-rule-delete <security_group_rule_uuid>
```

Delete security group:

```
quantum security-group-delete <security_group_uuid>
```


Create a port associated with security group:

```
quantum port-create <network_id> --security_groups list=true  
<security_group_id> <security_group_id>
```

Remove security groups from a port:

```
quantum port-update <port_id> --security_groups=None
```

6. Advanced Configuration Options

This section describes advanced configurations options for various system components (i.e. config options where the default is usually ok, but that the user may want to tweak). After installing from packages, \$QUANTUM_CONF_DIR is /etc/quantum.

Quantum Server with Plugin

This is the Quantum service that runs the Quantum API Web Server. It is responsible for loading a plugin and passing the API calls to the plugin for processing. The quantum-server should receive one or more configuration files as its input, for example:

```
quantum-server --config-file <quantum config> --config-file <plugin config>
```

The quantum config contains the common quantum configuration parameters. The plugin config contains the plugin specific flags. The plugin that is run on the service is loaded via the configuration parameter 'core_plugin'. In some cases a plugin may have an agent that performs the actual networking. Specific configuration details can be seen in the Appendix - Configuration File Options.

Most plugins require a SQL database. After installing and starting the database server, set a password for the root account and delete the anonymous accounts:

```
$> mysql -u root
mysql> update mysql.user set password = password('iamroot') where user =
'root';
mysql> delete from mysql.user where user = '';
```

Create a database and user account specifically for plugin:

```
mysql> create database <database-name>;
mysql> create user '<user-name>'@'localhost' identified by '<user-name>';
mysql> create user '<user-name>'@'%' identified by '<user-name>';
mysql> grant all on <database-name>.* to '<user-name>'@'%';
```

Once the above is done you can update the settings in the relevant plugin configuration files. The plugin specific configuration files can be found at \$QUANTUM_CONF_DIR/plugins.

Some plugins have a L2 agent that performs the actual networking. That is, the agent will attach the virtual machine NIC to the Quantum network. Each node should have a Quantum agent running on it. Note that the agent receives the following input parameters:

```
quantum-plugin-agent --config-file <quantum config> --config-file <plugin
config>
```

Two things need to be done prior to working with the plugin:

1. Ensure that the core plugin is updated.
2. Ensure that the database connection is correctly set.

The table below contains examples for these settings. Some linux packages may provide installation utilities that configure these.

Table 6.1. Settings

Parameter	Value
Open vSwitch	
core_plugin (\$QUANTUM_CONF_DIR/ quantum.conf)	quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
sql_connection (in the plugin configuration file)	mysql://<username>:<password>@localhost/ovs_quantum?charset=utf8
Plugin Configuration File	\$QUANTUM_CONF_DIR/plugins/openvswitch/ovs_quantum_plugin.ini
Agent	quantum-openvswitch-agent
Linux Bridge	
core_plugin (\$QUANTUM_CONF_DIR/ quantum.conf)	quantum.plugins.linuxbridge.lb_quantum_plugin.LinuxBridgePluginV2
sql_connection (in the plugin configuration file)	mysql://<username>:<password>@localhost/quantum_linux_bridge? charset=utf8
Plugin Configuration File	\$QUANTUM_CONF_DIR/plugins/linuxbridge/linuxbridge_conf.ini
Agent	quantum-linuxbridge-agent

All of the plugin configuration files options can be found in the [Appendix - Configuration File Options](#).

DHCP Agent

There is an option to run a DHCP server that will allocate IP addresses to virtual machines running on the network. When a Quantum subnet is created, by default, the subnet has DHCP enabled.

The node that runs the DHCP agent should run:

```
quantum-dhcp-agent --config-file <quantum config>
--config-file <dhcp config>
```

Currently the DHCP agent uses dnsmasq to perform that static address assignment.

A driver needs to be configured that matches the plugin running on the service.

Table 6.2. Basic settings

Parameter	Value
Open vSwitch	
interface_driver (\$QUANTUM_CONF_DIR/dhcp_agent.ini)	quantum.agent.linux.interface.OVSInterfaceDriver
Linux Bridge	
interface_driver (\$QUANTUM_CONF_DIR/dhcp_agent.ini)	quantum.agent.linux.interface.BridgeInterfaceDriver

All of the DHCP agent configuration options can be found in the [Appendix - Configuration File Options](#).

Namespace

By default the DHCP agent makes use of linux network namespaces in order to support overlapping IP addresses. Requirements for network namespaces support are described in the [Limitation](#) section.

If the linux installation does not support network namespace, you must disable using network namespace in the DHCP agent config file (The default value of use_namespaces is True).

```
use_namespaces = False
```

L3 Agent

There is an option to run a L3 agent that will give enable layer 3 forwarding and floating IP support. The node that runs the L3 agent should run:

```
quantum-l3-agent --config-file <quantum config>  
--config-file <l3 config>
```

A driver needs to be configured that matches the plugin running on the service. The driver is used to create the routing interface.

Table 6.3. Basic settings

Parameter	Value
Open vSwitch	
interface_driver (\$QUANTUM_CONF_DIR/l3_agent.ini)	quantum.agent.linux.interface.OVSInterfaceDriver
external_network_bridge (\$QUANTUM_CONF_DIR/l3_agent.ini)	br-ex
Linux Bridge	
interface_driver (\$QUANTUM_CONF_DIR/l3_agent.ini)	quantum.agent.linux.interface.BridgeInterfaceDriver
external_network_bridge (\$QUANTUM_CONF_DIR/l3_agent.ini)	This field must be empty (or the bridge name for the external network).

The L3 agent communicates with the Quantum server via the Quantum API, so the following configuration is required:

1. Keystone authentication:

```
auth_url="$KEYSTONE_SERVICE_PROTOCOL://$KEYSTONE_AUTH_HOST:  
$KEYSTONE_AUTH_PORT/v2.0"
```

For example,

```
http://10.56.51.210:5000/v2.0
```

2. Admin user details:

```
admin_tenant_name $SERVICE_TENANT_NAME  
admin_user $Q_ADMIN_USERNAME  
admin_password $SERVICE_PASSWORD
```

All of the L3 agent configuration options can be found in the [Appendix - Configuration File Options](#).

Namespace

By default the L3 agent makes use of Linux network namespaces in order to support overlapping IP addresses. Requirements for network namespaces support are described in the [Limitation](#) section.

If the linux installation does not support network namespace, you must disable using network namespace in the L3 agent config file (The default value of use_namespaces is True).

```
use_namespaces = False
```

When use_namespaces is set as False, only one router ID can be supported per node. This must be configured via the configuration variable *router_id*.

```
# If use_namespaces is set as False then the agent can only configure one
router.
# This is done by setting the specific router_id.
router_id = 1064ad16-36b7-4c2f-86f0-daa2bcd6b2a
```

To configure it, you need to run the Quantum service and create a router, and then set an ID of the router created to *router_id* in the L3 agent configuration file.

```
$ quantum router-create myrouter1
Created a new router:
+-----+-----+
| Field          | Value                                     |
+-----+-----+
| admin_state_up | True                                     |
| external_gateway_info |                                         |
| id             | 338d42d7-b22e-42c5-9df6-f3674768fe75   |
| name           | myrouter1                               |
| status         | ACTIVE                                  |
| tenant_id      | 0c236f65baa04e6f9b4236b996555d56     |
+-----+-----+
```

Multiple Floating IP Pools

Quantum L3 supports multiple floating IP pools. In Quantum, a floating IP pool is represented as an external network and a floating IP is allocated from a subnet associated with the external network. Since each L3 agent can be associated with at most one external network, we need to invoke multiple L3 agent to define multiple floating IP pools. 'gateway_external_network_id' in L3 agent configuration file indicates the external network that the L3 agent handles. You can run multiple L3 agent instances on one host.

In addition, when you run multiple L3 agents, make sure that **handle_internal_only_routers** is set to **True** only for one L3 agent in a quantum deployment and set to **False** for all other L3 agents. Since the default value of this parameter is True, you need to configure it carefully.

Before starting L3 agents, you need to create routers and external networks, then update the configuration files with UUID of external networks and start L3 agents.

For the first agent, invoke it with the following l3_agent.ini where **handle_internal_only_routers** is True.

```
handle_internal_only_routers = True
gateway_external_network_id = 2118b11c-011e-4fa5-a6f1-2ca34d372c35
external_network_bridge = br-ex
```

```
python /opt/stack/quantum/bin/quantum-l3-agent
--config-file /etc/quantum/quantum.conf
```

```
--config-file=/etc/quantum/l3_agent.ini
```

For the second (or later) agent, invoke it with the following `l3_agent.ini` where `handle_internal_only_routers` is `False`.

```
handle_internal_only_routers = False
gateway_external_network_id = e828e54c-850a-4e74-80a8-8b79c6a285d8
external_network_bridge = br-ex-2
```

Nova Metadata Server Support

To use Nova metadata service, `metadata_ip` and `metadata_port` in the L3 agent configuration file need to be configured. Accessing from VMs to Nova metadata service is forwarded to an external network through Quantum L3 router. Nova metadata service must be reachable from the external network. As the [Limitations section](#) says, note that Quantum overlapping IPs support and Nova metadata service cannot be used together.

Allowing VMs to reach the metadata service is a big point of confusion with Quantum. We need to make sure instructions for how to set this up are displayed more prominently than they already are, and that there are instructions for how to validate and troubleshoot in this scenario.

Example validation includes:

VALIDATION STEP #1

- on network node(`l3_agent` running) ping to `metadata_ip` that specified by `l3_agent.py`
if you are not using namespaces, just run:

```
ping <metadata_ip>
```

for example, if the metadata server IP is `172.16.10.5`, run:

```
$ping 172.16.10.5
```

if you using namespaces, identify the UUID of the router and run:

```
ip netns exec qrouter-<router uuid> ping <metadata_ip>
```

for example, if the router uuid is `d7e9ec57-77c2-4046-aebf-d978ed4a4f83` and the metadata server IP is `172.16.10.5`, run:

```
$ ip netns exec qrouter-d7e9ec57-77c2-4046-aebf-d978ed4a4f83 ping 172.16.10.5
```

VALIDATION STEP #2

- on metadata server(`nova_api` server) check connection to vm's subnets using an un-NATed IP address of the VM, such as `10.0.0.2`:

```
$ping 10.0.0.2
```

NOTE

OpenStack does not manage this routing for you, so you need to make sure that your host running the metadata service always has a route to reach each private network's subnet

via the external network IP of that subnet's quantum router. To do this, you can either run quantum without namespaces, and run the quantum-l3-agent on the same host as nova-api. Otherwise, you can identify an IP prefix that includes all private network subnet's (e.g., 10.0.0.0/8) and then make sure that your metadata server has a route for that prefix with the quantum router's external IP address as the next hop.

7. Authentication and Authorization

Quantum uses the Keystone identity service (openstack.keystone.org) as the default authentication service. When keystone is enabled Users submitting requests to the Quantum service must provide an authentication token in X-Auth-Token request header. The aforementioned token should have been obtained by authenticating with the keystone endpoint. For more information concerning authentication with Keystone, please refer to the Keystone documentation. When keystone is enabled, it is not mandatory to specify `tenant_id` for resources in create requests, as the tenant identifier will be derived from the Authentication token. Please note that the default authorization settings only allow administrative users to create resources on behalf of a different tenant. Quantum uses information received from Keystone to authorize user requests. Quantum handles two kind of authorization policies:

- **Operation-based:** policies specify access criteria for specific operations, possibly with fine-grained control over specific attributes;
- **Resource-based:** whether access to specific resource might be granted or not according to the permissions configured for the resource (currently available only for the network resource). The actual authorization policies enforced in Quantum might vary from deployment to deployment.

The policy engine reads entries from the `policy.json` file. The actual location of this file might vary from distribution to distribution. Entries can be updated while the system is running, and no service restart is required. That is to say, everytime the policy file is updated, the policies will be automatically reloaded. Currently the only way of updating such policies is to edit the policy file. Please note that in this section we will use both the terms "policy" and "rule" to refer to objects which are specified in the same way in the policy file; in other words, there are no syntax differences between a rule and a policy. We will define a policy something which is matched directly from the quantum policy engine, whereas we will define a rule as the elements of such policies which are then evaluated. For instance in `create_subnet: [["admin_or_network_owner"]]`, `create_subnet` is regarded as a policy, whereas `admin_or_network_owner` is regarded as a rule.

Policies are triggered by the Quantum policy engine whenever one of them matches a Quantum API operation or a specific attribute being used in a given operation. For instance the `create_subnet` policy is triggered every time a `POST /v2.0/subnets` request is sent to the Quantum server; on the other hand `create_network:shared` is triggered every time the `shared` attribute is explicitly specified (and set to a value different from its default) in a `POST /v2.0/networks` request. It is also worth mentioning that policies can be also related to specific API extensions; for instance `extension:provider_network:set` will be triggered if the attributes defined by the Provider Network extensions are specified in an API request.

An authorization policy can be composed by one or more rules. If more rules are specified, evaluation policy will be successful if any of the rules evaluates successfully; if an API operation matches multiple policies, then all the policies must evaluate successfully. Also, authorization rules are recursive. Once a rule is matched, the rule(s) can be resolved to another rule, until a terminal rule is reached.

The Quantum policy engine currently defines the following kinds of terminal rules:

- **Role-based rules:** evaluate successfully if the user submitting the request has the specified role. For instance "role:admin" is successful if the user submitting the request is an administrator.
- **Field-based rules:** evaluate successfully if a field of the resource specified in the current request matches a specific value. For instance "field:networks:shared=True" is successful if the attribute *shared* of the *network* resource is set to true.
- **Generic rules:** compare an attribute in the resource with an attribute extracted from the user's security credentials and evaluates successfully if the comparison is successful. For instance "tenant_id:%(tenant_id)s" is successful if the tenant identifier in the resource is equal to the tenant identifier of the user submitting the request.

The following is an extract from the default policy.json file:

```
{  
  
"admin_or_owner": [{"role:admin"}, {"tenant_id:%(tenant_id)s"}],  
[1]  
  
"admin_or_network_owner": [{"role:admin"}, {"tenant_id:  
%(network_tenant_id)s"}],  
  
"admin_only": [{"role:admin"}], "regular_user": [],  
  
"shared": [{"field:networks:shared=True"}],  
  
"default": [{"rule:admin_or_owner"}], [2]  
  
"create_subnet": [{"rule:admin_or_network_owner"}],  
  
"get_subnet": [{"rule:admin_or_owner"}, {"rule:shared"}],  
  
"update_subnet": [{"rule:admin_or_network_owner"}],  
  
"delete_subnet": [{"rule:admin_or_network_owner"}],  
  
"create_network": [],  
  
"get_network": [{"rule:admin_or_owner"}, {"rule:shared"}], [3]  
  
"create_network:shared": [{"rule:admin_only"}], [4]  
  
"update_network": [{"rule:admin_or_owner"}],  
  
"delete_network": [{"rule:admin_or_owner"}],  
  
"create_port": [],  
  
"create_port:mac_address": [{"rule:admin_or_network_owner"}], [5]  
  
"create_port:fixed_ips": [{"rule:admin_or_network_owner"}],  
  
"get_port": [{"rule:admin_or_owner"}],  
  
"update_port": [{"rule:admin_or_owner"}],
```

```
"delete_port": [{"rule:admin_or_owner"}]
}
```

[1] is a rule which evaluates successfully if the current user is an administrator or the owner of the resource specified in the request (tenant identifier is equal).

[2] is the default policy which is always evaluated if an API operation does not match any of the policies in policy.json.

[3] This policy will evaluate successfully if either *admin_or_owner*, or *shared* evaluates successfully.

[4] This policy will restrict the ability of manipulating the *shared* attribute for a network to administrators only.

[5] This policy will restrict the ability of manipulating the *mac_address* attribute for a port only to administrators and the owner of the network where the port is attached.

In some cases, some operations should be restricted to administrators only; therefore, as a further example, let us consider how this sample policy file should be modified in a scenario where tenants are allowed only to define networks and see their resources, and all the other operations can be performed only in an administrative context:

```
{
"admin_or_owner": [{"role:admin"}, {"tenant_id:%(tenant_id)s"}],
"admin_only": [{"role:admin}], "regular_user": [],
"default": [{"rule:admin_only}],
"create_subnet": [{"rule:admin_only}],
"get_subnet": [{"rule:admin_or_owner}],
"update_subnet": [{"rule:admin_only}],
"delete_subnet": [{"rule:admin_only}],
"create_network": [],
"get_network": [{"rule:admin_or_owner}],
"create_network:shared": [{"rule:admin_only}],
"update_network": [{"rule:admin_or_owner}],
"delete_network": [{"rule:admin_or_owner}],
"create_port": [{"rule:admin_only}],
"get_port": [{"rule:admin_or_owner}],
"update_port": [{"rule:admin_only}],
```

```
"delete_port": [{"rule:admin_only"}]  
}
```

8. Advanced Operational Features

Logging Settings

Quantum components use Python logging module to do logging. Logging configuration can be provided in quantum.conf or as command line options. Command options will override ones in quantum.conf.

Two ways to specify the logging configuration for Quantum components:

1. Provide logging settings in a logging configuration file.

Please see [Python Logging HOWTO](#) for logging configuration file.

2. Provide logging setting in quantum.conf

```
[DEFAULT]
# Default log level is WARNING
# Show debugging output in logs (sets DEBUG log level output)
# debug = False

# Show more verbose log output (sets INFO log level output) if debug is
False
# verbose = False

# log_format = %(asctime)s %(levelname)8s [%(name)s] %(message)s
# log_date_format = %Y-%m-%d %H:%M:%S

# use_syslog = False
# syslog_log_facility = LOG_USER

# if use_syslog is False, we can set log_file and log_dir.
# if use_syslog is False and we do not set log_file,
# the log will be printed to stdout.
# log_file =
# log_dir =
```

Notifications

Notification Options

Notifications can be sent when Quantum resources such as network, subnet and port are created, updated or deleted. To support DHCP agent, rpc_notifier driver must be set. To set up the notification, edit notification options in quantum.conf:

```
# ===== Notification System Options =====

# Notifications can be sent when network/subnet/port are create, updated or
deleted.
# There are three methods of sending notifications: logging (via the
```

```
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
# notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver
notification_driver = quantum.openstack.common.notifier.rpc_notifier

# default_notification_level is used to form actual topic names or to set
  logging level
# default_notification_level = INFO

# default_publisher_id is a part of the notification payload
# host = myhost.com
# default_publisher_id = $host

# Defined in rpc_notifier for rpc way, can be comma separated values.
# The actual topic names will be %s.%(default_notification_level)s
notification_topics = notifications
```

Setting Cases

Logging and RPC

The options below will make Quantum server send notifications via logging and RPC. The logging options are described in [Logging Settings](#). RPC notifications will go to 'notifications.info' queue binded to a topic exchange defined by 'control_exchange' in quantum.conf.

```
# ===== Notification System Options =====

# Notifications can be sent when network/subnet/port are create, updated or
  deleted.
# There are three methods of sending notifications: logging (via the
# log_file directive), rpc (via a message queue) and
# noop (no notifications sent, the default)

# Notification_driver can be defined multiple times
# Do nothing driver
# notification_driver = quantum.openstack.common.notifier.no_op_notifier
# Logging driver
notification_driver = quantum.openstack.common.notifier.log_notifier
# RPC driver
notification_driver = quantum.openstack.common.notifier.rpc_notifier

# default_notification_level is used to form actual topic names or to set
  logging level
default_notification_level = INFO

# default_publisher_id is a part of the notification payload
# host = myhost.com
# default_publisher_id = $host
```

```
# Defined in rpc_notifier for rpc way, can be comma separated values.  
# The actual topic names will be %s.%(default_notification_level)s  
notification_topics = notifications
```

Multiple RPC Topics

The options below will make Quantum server send notifications to multiple RPC topics. RPC notifications will go to 'notifications_one.info' and 'notifications_two.info' queues binded to a topic exchange defined by 'control_exchange' in quantum.conf.

```
# ===== Notification System Options =====  
  
# Notifications can be sent when network/subnet/port are create, updated or  
# deleted.  
# There are three methods of sending notifications: logging (via the  
# log_file directive), rpc (via a message queue) and  
# noop (no notifications sent, the default)  
  
# Notification_driver can be defined multiple times  
# Do nothing driver  
# notification_driver = quantum.openstack.common.notifier.no_op_notifier  
# Logging driver  
# notification_driver = quantum.openstack.common.notifier.log_notifier  
# RPC driver  
notification_driver = quantum.openstack.common.notifier.rabbit_notifier  
  
# default_notification_level is used to form actual topic names or to set  
# logging level  
default_notification_level = INFO  
  
# default_publisher_id is a part of the notification payload  
# host = myhost.com  
# default_publisher_id = $host  
  
# Defined in rpc_notifier for rpc way, can be comma separated values.  
# The actual topic names will be %s.%(default_notification_level)s  
notification_topics = notifications_one,notifications_two
```

Quotas

Quota is a function to limit number of resources. You can enforce default quota for all tenant. You will get error when you try to create resources more than the number of quota value.

```
$ quantum net-create test_net  
Quota exceeded for resources: ['network']
```

Per-tenant quota configuration is also supported by quota extension API. See [Per-tenant quota configuration](#) for details.

Basic quota configuration

In Quantum default quota mechanism, all tenants have a same quota value, i.e., a number of resources that a tenant can create. This is enabled by default.

The value of quota is defined in the Quantum configuration file (quantum.conf). If you want to disable quota for a specific resource (e.g., network, subnet, port), remove a corresponding item from 'quota_items'. Each of the quota values in the example below is the default value.

```
[QUOTAS]
# resource name(s) that are supported in quota features
quota_items = network,subnet,port

# number of networks allowed per tenant, and minus means unlimited
quota_network = 10

# number of subnets allowed per tenant, and minus means unlimited
quota_subnet = 10

# number of ports allowed per tenant, and minus means unlimited
quota_port = 50

# default driver to use for quota checks
quota_driver = quantum.quota.ConfDriver
```

Quantum also supports quotas for L3 resources: router and floating IP. You can configure them by adding the following lines to 'QUOTAS' section in quantum.conf. (Note that 'quota_items' does not affect these quotas.)

```
[QUOTAS]
# number of routers allowed per tenant, and minus means unlimited
quota_router = 10

# number of floating IPs allowed per tenant, and minus means unlimited
quota_floatingip = 50
```

Per-tenant quota configuration

Quantum also supports per-tenant quota limit by quota extension API. To enable per-tenant quota, you need to set 'quota_driver' in 'quantum.conf'.

```
quota_driver = quantum.extensions._quotav2_driver.DbQuotaDriver
```

When per-tenant quota is enabled, the output of the following command contains 'quotas'.

```
$ quantum ext-list -c alias -c name
+-----+-----+
| alias  | name                               |
+-----+-----+
| router | Quantum L3 Router                 |
| quotas | Quotas for each tenant            |
| provider | Provider Network                  |
+-----+-----+
```

**Note**

In Folsom release, per-tenant quota is supported by Open vSwitch plugin, Linux Bridge plugin, and Nicira NVP plugin and cannot be used with other plugins.

There are four CLI commands to manage per-tenant quota.

quota-delete	Delete defined quotas of a given tenant.
quota-list	List defined quotas of all tenants.
quota-show	Show quotas of a given tenant
quota-update	Define tenant's quotas not to use defaults.

Only users with 'admin' role can change a quota value. Note that the default set of quotas are enforced for all tenants by default, so there is no 'quota-create' command.

'quota-list' displays a list of tenants for which per-tenant quota is enabled. The tenants who have the default set of quota limits are not listed. This command is permitted to only 'admin' users.

```
$ quantum quota-list
+-----+-----+-----+-----+-----+
+-----+
| floatingip | network | port | router | subnet | tenant_id
|           |         |     |       |        |
+-----+-----+-----+-----+-----+
|           |         |     |       |        |
| 20        | 5       | 20  | 10   | 5     |
| 6f88036c45344d9999a1f971e4882723 |
| 25        | 10      | 30  | 10   | 10    |
| bff5c9455ee24231b5bc713c1b96d422 |
+-----+-----+-----+-----+-----+
+-----+
```

'quota-show' reports the current set of quota limits for the specified tenant. Regular (non-admin) users can call this command (without `--tenant_id` parameter). If per-tenant quota limits are not defined for the tenant, the default set of quotas are displayed.

```
$ quantum quota-show --tenant_id 6f88036c45344d9999a1f971e4882723
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 20    |
| network    | 5     |
| port       | 20    |
| router     | 10    |
| subnet     | 5     |
+-----+-----+
```

The below is an example called by a non-admin user.

```
$ quantum quota-show
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 20    |
| network    | 5     |
| port       | 20    |
| router     | 10    |
| subnet     | 5     |
+-----+-----+
```



```
+-----+-----+
```

You can update a quota of the given tenant by 'quota-update' command.

Update the limit of network quota.

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 --network 5
```

```
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 50    |
| network    | 5     |
| port       | 50    |
| router     | 10    |
| subnet     | 10    |
+-----+-----+
```

You can update quotas of multiple resources in one command.

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 --subnet 5 --port 20
```

```
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 50    |
| network    | 5     |
| port       | 20    |
| router     | 10    |
| subnet     | 5     |
+-----+-----+
```

To update the limits of L3 resource (router, floating IP), we need to specify new values of the quotas after '-'. The example below updates the limit of the number of floating IPs for the given tenant.

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 -- --floatingip 20
```

```
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 20    |
| network    | 5     |
| port       | 20    |
| router     | 10    |
| subnet     | 5     |
+-----+-----+
```

You can update the limits of multiple resources including L2 resources and L3 resource in one command.

```
$ quantum quota-update --tenant_id 6f88036c45344d9999a1f971e4882723 --network 3 --subnet 3 --port 3 -- --floatingip 3 --router 3
```

```
+-----+-----+
| Field      | Value |
+-----+-----+
| floatingip | 3     |
| network    | 3     |
| port       | 3     |
| router     | 3     |
+-----+-----+
```

```
| subnet      | 3      |  
+-----+-----+
```

To clear per-tenant quota limits, use 'quota-delete'. After 'quota-delete', quota limits enforced to the tenant are reset to the default set of quotas.

```
$ quantum quota-delete --tenant_id 6f88036c45344d9999a1f971e4882723  
Deleted quota: 6f88036c45344d9999a1f971e4882723  
$ quantum quota-show --tenant_id 6f88036c45344d9999a1f971e4882723  
+-----+-----+  
| Field      | Value  |  
+-----+-----+  
| floatingip | 50     |  
| network    | 10     |  
| port       | 50     |  
| router     | 10     |  
| subnet     | 10     |  
+-----+-----+
```

9. High Availability

Several aspects of a Quantum deployment benefit from high-availability to withstand individual node failures. In general, quantum-server and quantum-dhcp-agent can be run in an active-active fashion. quantum-l3-agent can be run only as active/passive, to avoid IP conflicts with respect to gateway IP addresses.

Quantum High Availability with Pacemaker

You can run some Quantum services into a cluster (Active / Passive or Active / Active for Quantum Server only) with Pacemaker.

Here you can download the latest Resources Agents :

- quantum-server: <https://github.com/madkiss/openstack-resource-agents/blob/master/ocf/quantum-server>
- quantum-dhcp-agent : <https://github.com/madkiss/openstack-resource-agents/blob/master/ocf/quantum-agent-dhcp>
- quantum-l3-agent : <https://github.com/madkiss/openstack-resource-agents/blob/master/ocf/quantum-agent-l3>



Note

If you need more informations about "*How to build a cluster*", please refer to [Pacemaker documentation](#).

10. Limitations

- *Quantum overlapping IPs do not work with Nova security groups or Nova metadata server:* Nova was designed assuming that a particular IP address will only ever be used by a single VM at any time. Quantum supports overlapping IPs if the `allow_overlapping_ips` config value is set to 'True'. We default this value to false to prevent unintentionally running Nova security groups or metadata server with overlapping IPs. If you enable this flag, you must disable both Nova security groups and the Nova metadata service.
- *No equivalent for nova-network –multi_host flag:* Nova-network has a model where the L3, NAT, and DHCP processing happen on the compute node itself, rather than a dedicated networking node. Quantum does not have an equivalent configuration, but is likely to add a similar capability in the future. However, since the nova-network multi_host design has some significant limitations in terms of deployment scale (limited to a single physical L2) and L3 forwarding behavior (does not map to a single L3 router for traffic between instances on separate networks), the Quantum feature may not be an exact match, or may be limited to a subnet of all Quantum deployment scenarios.
- *Linux network namespace required on nodes running quantum-l3-agent or quantum-dhcp-agent:* . In order to support overlapping IP addresses, the Quantum DHCP and L3 agents use Linux network namespaces by default. The hosts running these processes must support network namespaces. To support network namespaces, the following are required:
 - Linux kernel 2.6.24 or newer (with `CONFIG_NET_NS=y` in kernel configuration) and
 - iproute2 utilities ('ip' command) version 3.1.0 (aka 20111117) or newer

To check whether your host supports namespaces try running the following as root:

```
ip netns create test-ns
ip netns exec test-ns ifconfig
```

If the preceding commands do not produce errors, your platform is likely sufficient to use the dhcp-agent or l3-agent with namespace. In our experience, Ubuntu 12.04 or later support namespaces as does Fedora 17 and new, but some older RHEL platforms do not by default. It may be possible to upgrade the iproute2 package on a platform that does not support namespaces by default.

If you need to disable namespaces, make sure the quantum.conf used by quantum-server has the following setting:

```
allow_overlapping_ips=False
```

and that the dhcp_agent.ini and l3_agent.ini have the following setting:

```
use_namespaces=False
```

If you run both L3 + DHCP services on the same node, you should enable namespaces to avoid conflicts with routes :

```
use_namespaces=True
```

- *No IPv6 support for L3 agent:* The quantum-l3-agent supports only IPv4 forwarding. Currently, There are no errors provided if you configure IPv6 addresses via the API.
- *L3 Agent supports limited scale for Quantum Routers:* The L3 agent polls the Quantum API to learn about changes to L3 configuration. If there are a large number of routers or router ports, this can lead to heavy load on the database used by a Quantum plugin. The suggested work-around is to increase the polling_interval value in l3_agent.ini . This will increase the possible time between when a L3 configuration change happens via the API and when it affects data forwarding.
- *ZeroMQ support is experimental:* Some agents, including quantum-dhcp-agent, quantum-openvswitch-agent, and quantum-linuxbridge-agent use RPC to communicate. ZeroMQ is an available option in the configuration file, but has not been tested and should be considered experimental. In particular, there are believed to be issues with ZeroMQ and the dhcp agent.
- *MetaPlugin is experimental:* This release includes a "MetaPlugin" that is intended to support multiple plugins at the same time for different API requests, based on the content of those API requests. This functionality has not been widely reviewed or tested by the core team, and should be considered experimental until further validation is performed.
- *Horizon does not support Routers/Floating IPs with Quantum:* Horizon support is limited to operations on Quantum Networks, Subnets, and Ports. Routers and Floating IPs must be configured via CLI.
- L3 Router Extension does not support IPv6.

Appendix A. Demos Setup

This section describes how to configure the Quantum service and its components for some typical use cases.

Single Flat Network

This section describes how to install the Quantum service and its components for the "[Use Case: Single Flat Network](#)". The demo assumes the following:

OpenStack Service Node

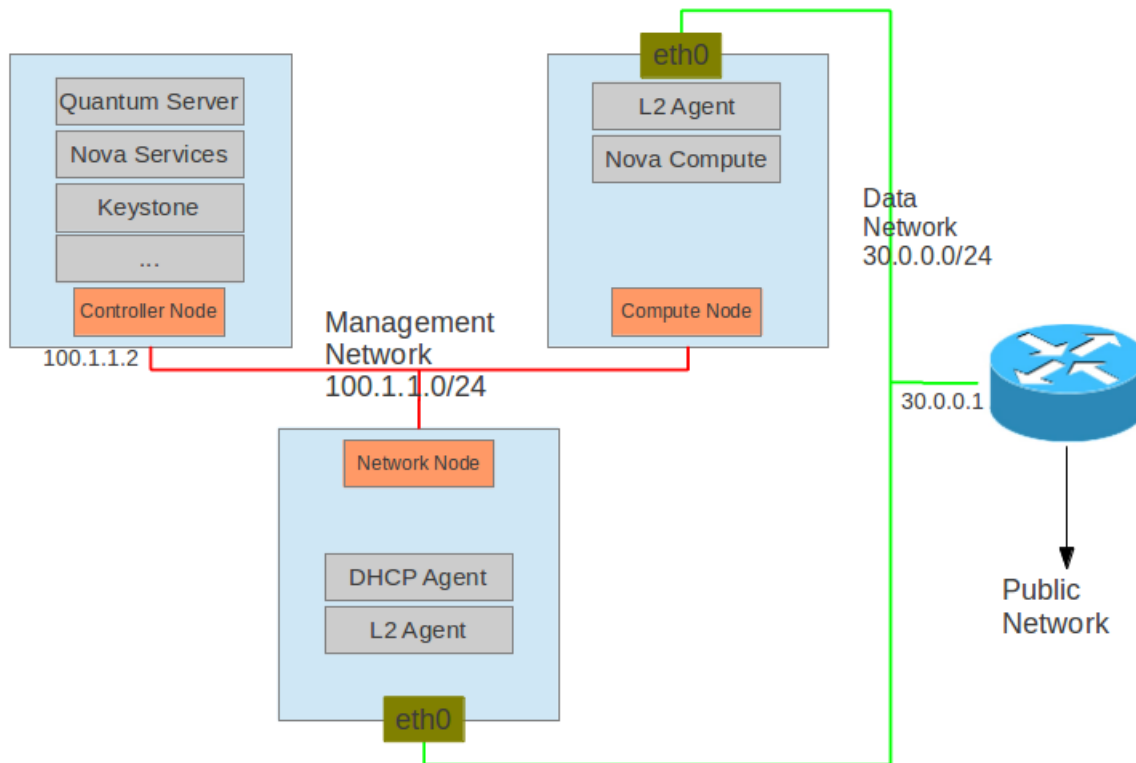
1. Relevant Nova services are installed, configured and running.
2. Glance is installed, configured and running. In addition to this there should be an image.
3. Keystone is installed, configured and running. A quantum user **quantum** should be created on tenant **servicetenant** with password **servicepassword**.
4. Additional services
 - RabbitMQ is running with default guest and its password
 - MySQL server (user is **root** and password is **root**)

Compute Nodes

1. Nova compute is installed and configured

The diagram below shows the setup. For simplicity all of the nodes should have one interface for management traffic and one or more interfaces for traffic to and from VMs. The management network is 100.1.1.0/24 with controller node at 100.1.1.2. The example uses the Open vSwitch plugin and agents.

Note the setup can be tweaked to make use of another supported plugin and its agents.



There will be some nodes in the setup.

Table A.1. Nodes for Demo

Node	Description
OpenStack Controller Node	Runs the Quantum service, Keystone and all of the Nova services that are required to deploy a VM. The service must have at least one network interface, this should be connected to the "Management Network". The IP address of the interface should be configured as 100.1.1.2/24. This will be used to communicate with the compute and network nodes. Note nova-network should not be running. This is replaced by Quantum.
Compute Node	Runs Nova compute and the Quantum L2 agent. This node will not have access the public network. The node must have at least two network interfaces. The first is used to communicate with the controller node, via the management network. The second interface will be used for the VM traffic, this is via the Data network. The VM will be able to receive its IP address from the DHCP agent on this network.
Network Node	Runs Quantum L2 agent and the DHCP agent. This node will have access the public network. The DHCP agent will allocate IP addresses to the VMs on the network. The node must have at least two network interfaces. The first is used to communicate with the controller node, this is via the management network. The second interface will be used for the VM traffic, this is on the data network.
Router	Router has IP 30.0.0.1, which is the default gateway for all VMs. The router should have ability to access public networks.

Installations

• Controller Node - Quantum Service

1. Install the Quantum service.
2. Create plugin database `ovs_quantum`. See the section on the [Core Plugins](#) for the exact details.
3. Update the Quantum configuration file, `/etc/quantum/quantum.conf`:

```
[DEFAULT]
core_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.
OVSQuantumPluginV2
control_exchange = quantum
rabbit_host = controlnode
notification_driver = quantum.openstack.common.notifier.rabbit_notifier
```

4. Update the plugin configuration file, `/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini`:

```
[DATABASE]
sql_connection = mysql://root:root@controlnode:3306/ovs_quantum?charset=
utf8
[OVS]
network_vlan_ranges = physnet1
bridge_mappings = physnet1:br-eth0
```

5. Update the api-paste configuration file to include the keystone user, `/etc/quantum/api-paste.ini`:

```
[filter:authtoken]
admin_tenant_name=servicetenant
admin_user=quantum
admin_password=servicepassword
```

6. Start the Quantum service

• Compute Node - Nova compute

1. Install the Nova compute.
2. Update the nova configuration file, `/etc/nova/nova.conf`. **Make sure the following is at the end of this file:**

```
network_api_class=nova.network.quantumv2.api.API

quantum_admin_username=quantum
quantum_admin_password=servicepassword
quantum_admin_auth_url=http://controlnode:35357/v2.0/
quantum_auth_strategy=keystone
quantum_admin_tenant_name=servicetenant
quantum_url=http://controlnode:9696/

libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
```

3. Restart the nova service

- **Compute and Network Node - L2 Agent**

1. Install the L2 agent.
2. Add the integration bridge to the Open vSwitch:

```
sudo ovs-vsctl add-br br-int
```

3. Update the Quantum configuration file, `/etc/quantum/quantum.conf`:

```
[DEFAULT]
core_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.
OVSQuantumPluginV2
control_exchange = quantum
rabbit_host = controlnode
notification_driver = quantum.openstack.common.notifier.rabbit_notifier
```

4. Update the plugin configuration file, `/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini`:

```
[DATABASE]
sql_connection = mysql://root:root@controlnode:3306/ovs_quantum?charset=
utf8
[OVS]
network_vlan_ranges = physnet1
bridge_mappings = physnet1:br-eth0
```

5. Create the network bridge **br-eth0** (All VM communication between the nodes will be done via eth0):

```
sudo ovs-vsctl add-br br-eth0
sudo ovs-vsctl add-port br-eth0 eth0
```

6. Start the Quantum L2 agent

- **Network Node - DHCP Agent**

1. Install the DHCP agent.
2. Update the Quantum configuration file, `/etc/quantum/quantum.conf`:

```
[DEFAULT]
core_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.
OVSQuantumPluginV2
control_exchange = quantum
rabbit_host = controlnode
notification_driver = quantum.openstack.common.notifier.rabbit_notifier
```

3. Update the DHCP configuration file, `/etc/quantum/dhcp_agent.ini`:

```
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
```

4. Start the DHCP agent

Logical Network Configuration

All of the commands below can be done on the controller node.

Note please ensure that the following environment variables are set. These are used by the various clients to access Keystone.

```
export OS_USERNAME=admin
export OS_PASSWORD=adminpassword
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://127.0.0.1:5000/v2.0/
```

1. Get the tenant ID (Used as \$TENANT_ID later):

```
keystone tenant-list
```

id	name	enabled
247e478c599f45b5bd297e8ddb9b6a	TenantA	True
2b4fec24e62e4ff28a8445ad83150f9d	TenantC	True
3719a4940bf24b5a8124b58c9b0a6ee6	TenantB	True
5fcfbc3283a142a5bb6978b549a511ac	demo	True
b7445f221cda4f4a8ac7db6b218b1339	admin	True

2. Get the User information:

```
keystone user-list
```

id	name	enabled	email
5a9149ed991744fa85f71e4aa92eb7ec	demo	True	
5b419c74980d46a1ab184e7571a8154e	admin	True	admin@example.com
8e37cb8193cb4873a35802d257348431	UserC	True	
c11f6b09ed3c45c09c21cbbc23e93066	UserB	True	
ca567c4f6c0942bdac0e011e97bddbe3	UserA	True	

3. Create a internal shared network on the demo tenant (\$TENANT_ID will be 5fcfbc3283a142a5bb6978b549a511ac):

```
quantum net-create --tenant-id $TENANT_ID sharednet1 --shared --
provider:network_type flat --provider:physical_network physnet1
```

Created a new network:

Field	Value
admin_state_up	True
id	04457b44-e22a-4a5c-be54-a53a9b2818e7
name	sharednet1
provider:network_type	flat
provider:physical_network	physnet1
provider:segmentation_id	
router:external	False
shared	True
status	ACTIVE
subnets	
tenant_id	b7445f221cda4f4a8ac7db6b218b1339

4. Create a subnet on the network:

```
quantum subnet-create --tenant-id $TENANT_ID sharednet1 30.0.0.0/24
```

Created a new subnet:

```
+-----+-----+
| Field          | Value                                     |
+-----+-----+
| allocation_pools | {"start": "30.0.0.2", "end": "30.0.0.254"} |
| cidr            | 30.0.0.0/24                             |
| dns_nameservers |                                           |
| enable_dhcp     | True                                      |
| gateway_ip      | 30.0.0.1                                 |
| host_routes     |                                           |
| id              | b8e9a88e-ded0-4e57-9474-e25fa87c5937    |
| ip_version      | 4                                         |
| name            |                                           |
| network_id      | 04457b44-e22a-4a5c-be54-a53a9b2818e7    |
| tenant_id       | 5fcfbc3283a142a5bb6978b549a511ac       |
+-----+-----+
```

5. Create a server for tenant A:

```
nova --os-tenant-name TenantA --os-username UserA --os-password password
--os-auth-url=http://localhost:5000/v2.0 boot --image tty --flavor 1 --nic
net-id=04457b44-e22a-4a5c-be54-a53a9b2818e7 TenantA_VM1
```

```
nova --os-tenant-name TenantA --os-username UserA --os-password password --
os-auth-url=http://localhost:5000/v2.0 list
```

```
+-----+-----+-----+-----+
| ID          | Name          | Status | Networks |
+-----+-----+-----+-----+
| 09923b39-050d-4400-99c7-e4b021cdc7c4 | TenantA_VM1 | ACTIVE | sharednet1=
30.0.0.3 |
+-----+-----+-----+-----+
```

6. Ping the server of tenant A:

```
sudo ip addr flush eth0
sudo ip addr add 30.0.0.201/24 dev br-eth0
ping 30.0.0.3
```

Note: if nova filter is using, please set right security group rule.

7. Ping the public network within the server of tenant A:

```
ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=1.74 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=1.50 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=1.23 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
```

```
rtt min/avg/max/mdev = 1.234/1.495/1.745/0.211 ms
```

Note: The 192.168.1.1 is an IP on public network that the router is connecting.

8. Create servers for other tenants

We can create servers for other tenants with similar commands. Since all these VMs share the same subnet, they will be able to access each other.

Provider Router with Private Networks

This section describes how to install the Quantum service and its components for the "[Use Case: Provider Router with Private Networks](#)". The demo assumes the following:

OpenStack Service Node

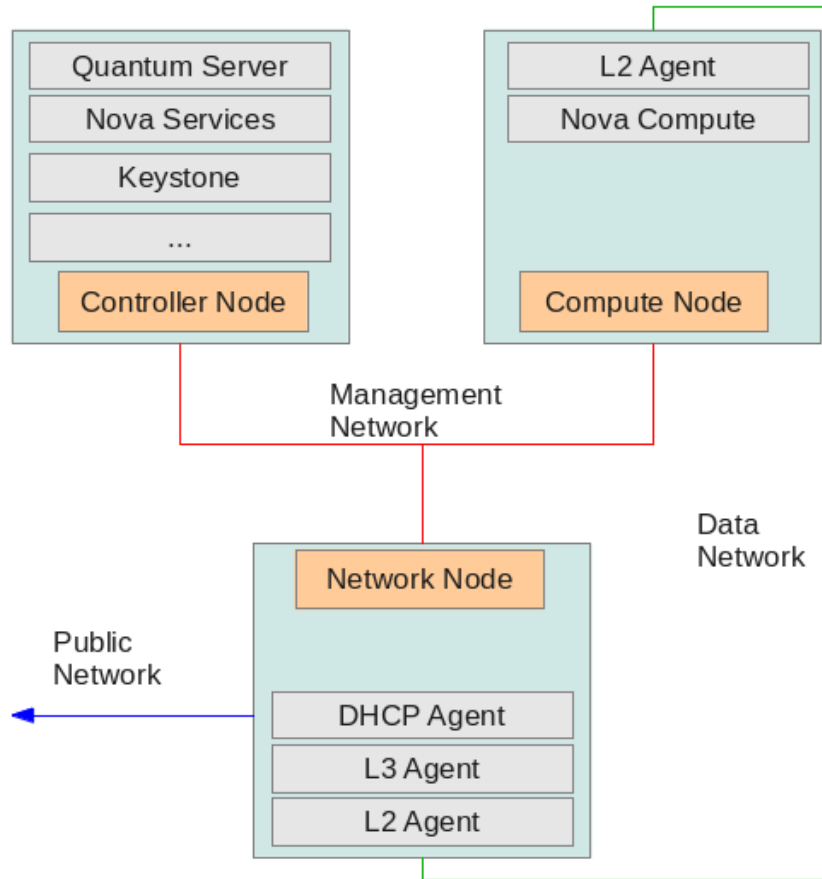
1. Relevant Nova services are installed, configured and running.
2. Glance is installed, configured and running. In addition to this there should be an image.
3. Keystone is installed, configured and running. A quantum user **quantum** should be created on tenant **servicetenant** with password **servicepassword**.
4. Additional services
 - RabbitMQ (password is **openstack**)
 - MySQL server (user is **quantum** and password is **openstack**)

Compute Nodes

1. Nova compute is installed and configured

The diagram below shows the setup. For simplicity all of the nodes should have one interface for management traffic and one or more interfaces for traffic to and from VMs. The management network is 100.1.1.0/24. The example uses the Open vSwitch plugin and agents.

Note the setup can be tweaked to make use of another supported plugin and its agents.



There will be three nodes in the setup.

Table A.2. Nodes for Demo

Node	Description
OpenStack Controller Node	Runs the Quantum service, Keystone and all of the Nova services that are required to deploy a VM. The service must have at least one network interface, this should be connected to the "Management Network". The IP address of the interface should be configured as 100.1.1.10/24. This will be used to communicate with the compute and network nodes. Note nova-network should not be running. This is replaced by Quantum.
Compute Node	Runs Nova compute and the Quantum L2 agent. This node will not have access the public network. The node must have at least two network interfaces. The first is used to communicate with the controller node, via the management network. The IP address of this interface should be configured to 100.1.1.11/24. The second interface will be used for the VM traffic, this is via the Data network. The VM will be able to receive its IP address from the DHCP agent on this network.
Network Node	Runs Quantum L2 agent, the DHCP agent and the L3 agent. This node will have access the public network. The DHCP agent will allocate IP addresses to the VMs on the network. The L3 agent will perform NAT and enable the VMs to access the public network. The node must

Node	Description
	have at least three network interfaces. The first is used to communicate with the controller node, this is via the management network. The IP address of this interface should be configured to 100.1.1.12/24. The second interface will be used for the VM traffic, this is on the data network. The third interface will be used to connect to the external gateway on the network. This interface will be bridged to the Open vSwitch bridge "br-ex" interface.

Installations

• Quantum Service

1. Install the Quantum service.
2. Create plugin database **ovs_quantum**. See the section on the [Core Plugins](#) for the exact details. Create user **quantum** with password **openstack**.
3. Update the Quantum configuration file, **/etc/quantum/quantum.conf**:

```
[DEFAULT]
core_plugin=quantum.plugins.openvswitch.ovs_quantum_plugin.
OVSQuantumPluginV2
rabbit_password = openstack
```

4. Update the plugin configuration file, **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini**:

```
[DATABASE]
sql_connection=mysql://quantum:openstack@localhost/ovs_quantum?charset=utf8

[OVS]
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
```

5. Update the api-paste configuration file to include the keystone user, **/etc/quantum/api-paste.ini**:

```
[filter:authtoken]
admin_tenant_name=servicetenant
admin_user=quantum
admin_password=servicepassword
```

6. Start the Quantum service

• Compute Nodes - L2 Agent

1. Install the L2 agent.
2. Add the integration bridge to the Open vSwitch:

```
sudo ovs-vsctl add-br br-int
```

3. Update the Quantum configuration file, **/etc/quantum/quantum.conf**:

```
rabbit_password = openstack
```

```
rabbit_host = 100.1.1.10
```

4. Update the plugin configuration file, `/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini`:

```
[OVS]
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

5. Create the network bridge `br-eth1` (All VM communication between the nodes will be done via eth1):

```
sudo ovs-vsctl add-br br-eth1
sudo ovs-vsctl add-port br-eth1 eth1
```

6. Update the nova configuration file, `/etc/nova/nova.conf`:

```
[DEFAULT]
network_api_class=nova.network.quantumv2.api.API

quantum_admin_username=quantum
quantum_admin_password=servicepassword
quantum_admin_auth_url=http://100.1.1.10:35357/v2.0/
quantum_auth_strategy=keystone
quantum_admin_tenant_name=servicetenant
quantum_url=http://100.1.1.10:9696/

libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
```

7. Restart the nova service

8. Start the Quantum L2 agent

- **Compute Node A - DHCP Agent**

1. Install the DHCP agent.
2. Update the Quantum configuration file, `/etc/quantum/quantum.conf`:

```
rabbit_password = openstack
rabbit_host = 100.1.1.10
```

3. Update the DHCP configuration file, `/etc/quantum/dhcp_agent.ini`:

```
interface_driver = quantum.agent.linux.interface.OVSInterfaceDriver
```

4. Start the DHCP agent

- **Compute Node B - L3 Agent**

1. Install the L3 agent.
2. Add the external network bridge to the Open vSwitch:

```
sudo ovs-vsctl add-br br-ex
```

3. Add the physical interface, for example eth2, that is connected to the outside network to this bridge:


```
sudo ovs-vsctl add-port br-ex eth2
```

4. Update the L3 configuration file, `/etc/quantum/l3_agent.ini`:

```
[DEFAULT]

auth_url=http://100.1.1.10:35357/v2.0/
admin_user=quantum
admin_password=servicepassword
admin_tenant_name=servicetenant

interface_driver=quantum.agent.linux.interface.OVSInterfaceDriver
```

5. Start the L3 agent

Logical Network Configuration

All of the commands below can be done on the service node.

Note please ensure that the following environment variables are set. These are used by the various clients to access Keystone.

```
export OS_USERNAME=admin
export OS_PASSWORD=adminpassword
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://127.0.0.1:5000/v2.0/
```

• Internal Networking Configuration

1. Get the tenant ID (Used as `$TENANT_ID` later).

```
keystone tenant-list
```

id	name	enabled
48fb81ab2f6b409bafac8961a594980f	admin	True
cbb574ac1e654a0a992bfc0554237abf	service	True
e371436fe2854ed89cca6c33ae7a83cd	invisible_to_admin	True
e40fa60181524f9f9ee7aa1038748f08	demo	True

2. Create a internal network on the demo tenant (`$TENANT_ID` will be `e40fa60181524f9f9ee7aa1038748f08`):

```
quantum net-create --tenant-id $TENANT_ID net1 --provider:network_type
vlan --provider:physical_network physnet1 --provider:segmentation_id 1024
```

Field	Value
admin_state_up	True
id	e99a361c-0af8-4163-9feb-8554d4c37e4f
name	net1
provider:network_type	vlan
provider:physical_network	physnet1
provider:segmentation_id	1024
router:external	False

shared	False
status	ACTIVE
subnets	
tenant_id	e40fa60181524f9f9ee7aa1038748f08

3. Create a subnet on the network (Used as \$SUBNET_ID later):

```
quantum subnet-create --tenant-id $TENANT_ID net1 10.0.0.0/24
```

Field	Value
allocation_pools	{"start": "10.0.0.2", "end": "10.0.0.254"}
cidr	10.0.0.0/24
dns_nameservers	
enable_dhcp	True
gateway_ip	10.0.0.1
host_routes	
id	c395cb5d-ba03-41ee-8a12-7e792d51a167
ip_version	4
name	
network_id	e99a361c-0af8-4163-9feb-8554d4c37e4f
tenant_id	e40fa60181524f9f9ee7aa1038748f08

• External Networking Configuration

1. Create a router (Used as \$ROUTER_ID later):

```
quantum router-create --tenant-id $TENANT_ID router1
```

Field	Value
admin_state_up	True
external_gateway_info	
id	685f64e7-a020-4fdf-a8ad-e41194ael24b
name	router1
status	ACTIVE
tenant_id	e40fa60181524f9f9ee7aa1038748f08

2. Add the router to the subnet:

```
quantum router-interface-add $ROUTER_ID $SUBNET_ID
```

```
Added interface to router 685f64e7-a020-4fdf-a8ad-e41194ael24b
```

3. Create the external network (Used as \$EXTERNAL_NETWORK_ID). Note this is on a different tenant to \$TENANT_ID:

```
quantum net-create ext_net --tenant-id $TENANT_ID --router:external=True
```

Field	Value
admin_state_up	True
id	8858732b-0400-41f6-8e5c-25590e67ffeb
name	ext_net
provider:network_type	vlan

```

| provider:physical_network | physnet1
| provider:segmentation_id | 1
| router:external          | True
| shared                   | False
| status                    | ACTIVE
| subnets                 |
| tenant_id                 | cbb574ac1e654a0a992bfc0554237abf
+-----+-----+

```

4. Create the subnet for floating IPs. **Note** the DHCP service is disabled for this subnet:

```
quantum subnet-create ext_net 172.24.4.224/28 -- --enable_dhcp=False
```

```

+-----+-----+
| Field          | Value
+-----+-----+
| allocation_pools | {"start": "172.24.4.226", "end": "172.24.4.238"}
| cidr            | 172.24.4.224/28
| dns_nameservers |
| enable_dhcp     | False
| gateway_ip      | 172.24.4.225
| host_routes     |
| id              | aef60b55-cbff-405d-a81d-406283ac6cff
| ip_version      | 4
| name            |
| network_id      | 8858732b-0400-41f6-8e5c-25590e67ffeb
| tenant_id       | cbb574ac1e654a0a992bfc0554237abf
+-----+-----+

```

5. Set the router for the external network:

```
quantum router-gateway-set $ROUTER_ID $EXTERNAL_NETWORK_ID
```

```
Set gateway for router 685f64e7-a020-4fdf-a8ad-e41194ae124b
```

- **Floating IP Allocation**

1. After a VM is deployed a floating IP address can be associated to the VM. A VM that is created will be allocated a Quantum port (\$PORT_ID). The port ID for the VM can be retrieved as follows:

```

nova list
+-----+-----+-----+-----+
| ID | Name | Status | Networks |
+-----+-----+-----+-----+
| 1cdc671d-a296-4476-9a75-f9cald92fd26 | testvm | ACTIVE | net1=10.0.0.3 |
+-----+-----+-----+-----+

quantum port-list -- --device_id 1cdc671d-a296-4476-9a75-f9cald92fd26
+-----+-----+-----+-----+
| id | name | mac_address |
+-----+-----+-----+-----+

```

```
| 9aa47099-b87b-488c-8c1d-32f993626a30 |          | fa:16:3e:b4:d6:6c |
{"subnet_id": "c395cb5d-ba03-41ee-8a12-7e792d51a167", "ip_address": "10.
0.0.3"} |
+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
```

2. Allocate a floating IP (Used as \$FLOATING_ID):

```
quantum floatingip-create ext_net

+-----+-----+-----+
| Field          | Value                               |
+-----+-----+-----+
| fixed_ip_address |                                     |
| floating_ip_address | 172.24.4.227                       |
| floating_network_id | 8858732b-0400-41f6-8e5c-25590e67ffeb |
| id              | 40952c83-2541-4d0c-b58e-812c835079a5 |
| port_id         |                                     |
| router_id       |                                     |
| tenant_id       | e40fa60181524f9f9ee7aa1038748f08  |
+-----+-----+-----+
```

3. Associate a floating IP to a VM (in the case of the example it is 9aa47099-b87b-488c-8c1d-32f993626a30):

```
quantum floatingip-associate $FLOATING_ID $PORT_ID

Associated floatingip 40952c83-2541-4d0c-b58e-812c835079a5
```

Show the floating IP:

```
quantum floatingip-show $FLOATING_ID

+-----+-----+-----+
| Field          | Value                               |
+-----+-----+-----+
| fixed_ip_address | 10.0.0.3                           |
| floating_ip_address | 172.24.4.227                       |
| floating_network_id | 8858732b-0400-41f6-8e5c-25590e67ffeb |
| id              | 40952c83-2541-4d0c-b58e-812c835079a5 |
| port_id         | 9aa47099-b87b-488c-8c1d-32f993626a30 |
| router_id       | 685f64e7-a020-4fdf-a8ad-e41194ae124b |
| tenant_id       | e40fa60181524f9f9ee7aa1038748f08  |
+-----+-----+-----+

ping 172.24.4.227

PING 172.24.4.227 (172.24.4.227) 56(84) bytes of data.
64 bytes from 172.24.4.227: icmp_req=2 ttl=64 time=0.152 ms
64 bytes from 172.24.4.227: icmp_req=3 ttl=64 time=0.049 ms
```

Scale and HA of agents

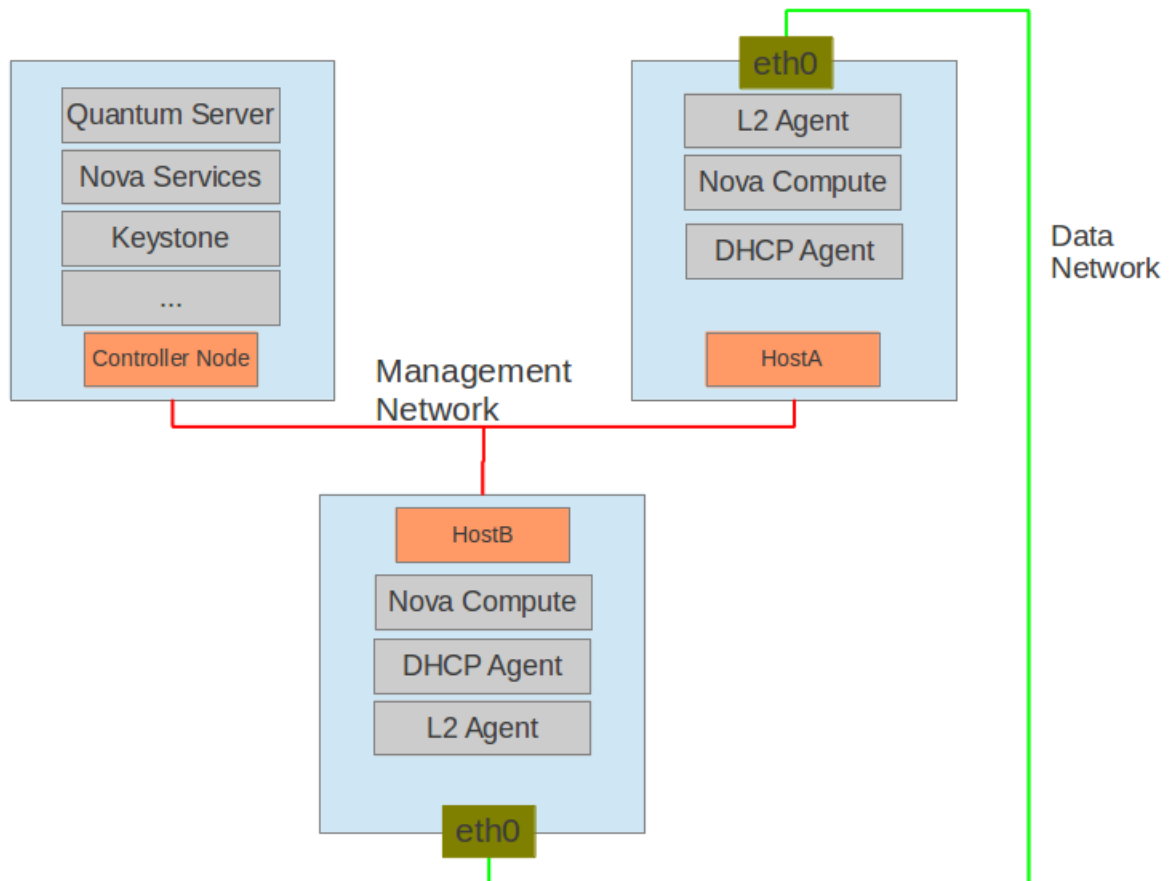
This section describes how to use component extension for agents scalability and HA



Note

We can use quantum client command to check if the component extension enabled:

```
gongysh@gongysh-laptop:~$ quantum ext-list -c name -c alias
+-----+-----+
| alias  | name  |
+-----+-----+
| component | component |
| router   | Quantum L3 Router |
| binding  | Port Binding  |
| quotas   | Quotas for each tenant |
| provider | Provider Network |
+-----+-----+
```



There will be three hosts in the setup.

Table A.3. Hosts for Demo

Host	Description
OpenStack Controller host - controlnode	Runs the Quantum service, Keystone and all of the Nova services that are required to deploy a VM. The service must have at least one network interface, this should be connected to the "Management Network". Note nova-network should not be running. This is replaced by Quantum.
HostA	Runs Nova compute and the Quantum L2 agent and DCHP agent
HostB	Same as HostA

Configuration

• controlnode - Quantum Server

1. Quantum configuration file `/etc/quantum/quantum.conf`:

```
core_plugin = quantum.plugins.openvswitch.ovs_quantum_plugin.  
OVSQuantumPluginV2  
core_plugin = quantum.plugins.linuxbridge.lb_quantum_plugin.  
LinuxBridgePluginV2  
rabbit_host = controlnode  
notification_driver = quantum.openstack.common.notifier.rabbit_notifier  
allow_overlapping_ips = True  
host = controlnode
```

2. Update the plugin configuration file `/etc/quantum/plugins/linuxbridge/linuxbridge_conf.ini`:

```
[VLANS]  
tenant_network_type = vlan  
network_vlan_ranges = physnet1:1000:2999  
[DATABASE]  
sql_connection = mysql://root:root@127.0.0.1:3306/quantum_linux_bridge  
reconnect_interval = 2  
[LINUX_BRIDGE]  
physical_interface_mappings = physnet1:eth0
```

• HostA and HostB - L2 Agent

1. Quantum configuration file `/etc/quantum/quantum.conf`:

```
rabbit_host = controlnode  
rabbit_password = openstack  
# host = HostB on hostb  
host = HostA
```

2. Update the plugin configuration file `/etc/quantum/plugins/linuxbridge/linuxbridge_conf.ini`:

```
[VLANS]  
tenant_network_type = vlan  
network_vlan_ranges = physnet1:1000:2999  
[DATABASE]
```

```
sql_connection = mysql://root:root@127.0.0.1:3306/quantum_linux_bridge
reconnect_interval = 2
[LINUX_BRIDGE]
physical_interface_mappings = physnet1:eth0
```

3. Update the nova configuration file `/etc/nova/nova.conf`:

```
[DEFAULT]
network_api_class=nova.network.quantumv2.api.API

quantum_admin_username=quantum
quantum_admin_password=servicepassword
quantum_admin_auth_url=http://controlnode:35357/v2.0/
quantum_auth_strategy=keystone
quantum_admin_tenant_name=servicetenant
quantum_url=http://100.1.1.10:9696/
firewall_driver=nova.virt.firewall.NoopFirewallDriver
libvirt_vif_driver=nova.virt.libvirt.vif.QuantumLinuxBridgeVIFDriver
```

• HostA and HostB - DHCP Agent

1. Update the DHCP configuration file `/etc/quantum/dhcp_agent.ini`:

```
[DEFAULT]
interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = quantum.agent.linux.dhcp.Dnsmasq
use_component_ext = True
```

Commands in component extension

All component extension commands below can be done on any hosts under only admin role.



Note

please ensure that the following environment variables are set. These are used by the various clients to access Keystone.

```
export OS_USERNAME=admin
export OS_PASSWORD=adminpassword
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controlnode:5000/v2.0/
```

• Settings

We need some VMs and a quantum network to experiment. Here they are:

```
gongysh@gongysh-laptop:~$ nova list
+-----+-----+-----+-----+
| ID                                     | Name       | Status  | Networks |
+-----+-----+-----+-----+
| c394fcd0-0baa-43ae-a793-201815c3e8ce | myserver1  | ACTIVE  | net1=10.0.1.3 |
| 2d604e05-9a6c-4ddb-9082-8a1fbdcc797d | myserver2  | ACTIVE  | net1=10.0.1.4 |
```

```
| c7c0481c-3db8-4d7a-a948-60ce8211d585 | myserver3 | ACTIVE | net1=10.0.1.5  
+-----+-----+-----+-----+  
+  
gongysh@gongysh-laptop:~$ quantum net-list  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| id | name | subnets  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| 89dca1c6-c7d4-4f7a-b730-549af0fb6e34 | net1 | f6c832e3-9968-46fd-8e45-  
d5cf646db9d1 |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+
```

- **Manage components in quantum deployment**

Component is a name for logic parts running in a quantum deployment. They are divided into groups, such as agent, plugin and service. Each group will have types. For example, there are 'Linux bridge agent, DHCP agent, Open vSwitch agent' in agent group. There are many kinds of plugins in plugin group too, such as coreplugin, lbaas plugin.

1. List all components

```
quantum component-list  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| id | type | host |  
disabled | state |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| 1b69828d-6a9b-4826-87cd-1757f0e27f31 | Linux bridge agent | HostA |  
False | XXX |  
| a0c1c21c-d4f4-4577-9ec7-908f2d48622d | DHCP agent | HostA |  
False | :- ) |  
| ed96b856-ae0f-4d75-bb28-40a47ffd7695 | Linux bridge agent | HostB |  
False | :- ) |  
| f28aa126-6edb-4ea5-a81e-8850876bc0a8 | DHCP agent | HostB |  
False | :- ) |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+
```



Note

'host' is just a logic name for each component.

Just as shown, we have four agents now, and they have reported their state. This command has a 'service_down_time' optional argument with default value 60 seconds. The state will be ':-)' if the component reported the state within this time. Otherwise the state is 'xxx'. So let us fix the 'Linux bridge agent' on HostA and get a ':-)' for it.

2. List the DHCP agents hosting a given network

In some deployments, one DHCP agent are not enough to hold all the network data. In addition, we should have backup for it even when the deployment is small one.

The same network can be assigned to more than DHCP agent and one DHCP agent can host more than one networks. Let's first go with command that lists DHCP agents hosting a given network.

```
gongysh@gongysh-laptop:~$ quantum dhcp-agent-list net1
+-----+-----+-----+-----+
| id                | host  | disabled | state |
+-----+-----+-----+-----+
| a0c1c21c-d4f4-4577-9ec7-908f2d48622d | HostA | False   | :-)  |
+-----+-----+-----+-----+
```

3. List the networks hosted by a given DHCP agent

This command is the sister one with previous.

```
gongysh@gongysh-laptop:~$ quantum network-dhcp-agent-list a0c1c21c-
d4f4-4577-9ec7-908f2d48622d
+-----+-----+-----+
| id                | name  | subnets |
+-----+-----+-----+
| 89dca1c6-c7d4-4f7a-b730-549af0fb6e34 | net1  | f6c832e3-9968-46fd-8e45-
d5cf646db9d1 |
+-----+-----+-----+
```

4. Show the component's detail information

'component-list' is just giving very general information for components. To know the detail information of a component, we can use 'component-show'.

```
gongysh@gongysh-laptop:~$ quantum component-show a0c1c21c-
d4f4-4577-9ec7-908f2d48622d
+-----+
| Field          | Value |
+-----+
| _current_time  | 2012-12-18T08:54:08.503839 |
| binary         | quantum-dhcp-agent |
| configurations | {
|                 |     "subnets": "1",
|                 |     "use_namespaces": "True",
|                 |     "dhcp_driver": "quantum.agent.linux.dhcp.
Dnsmasq",
|                 |     "networks": "1",
|                 |     "dhcp_lease_time": "120",
|                 |
+-----+
```


Just as shown, we can see bridge-mapping, and the number of VM's virtual nics on this L2 agent.

- **Manage assignment of networks to DHCP agent**

We have shown 'network-dhcp-agent-list' and 'dhcp-agent-list' command. Now let's look at how to add a network to a DHCP agent and remove one from it.

1. Default scheduling

When a network is created, we will try to schedule it to an active DHCP agent. If there are many active DHCP agents, we select one randomly. (We can design more sophisticated scheduling algorithm just like we do in nova-schedule.)

```
gongysh@gongysh-laptop:~$ quantum net-create net2
Created a new network:
+-----+
| Field          | Value                                     |
+-----+
| admin_state_up | True                                     |
| id              | 9b96b14f-71b8-4918-90aa-c5d705606b1a   |
| name           | net2                                     |
| provider:network_type | vlan                                     |
| provider:physical_network | physnet1                                 |
| provider:segmentation_id | 1001                                     |
| router:external | False                                    |
| shared         | False                                    |
| status         | ACTIVE                                   |
| subnets      |                                           |
| tenant_id     | b7445f221cda4f4a8ac7db6b218b1339     |
+-----+

gongysh@gongysh-laptop:~$ quantum dhcp-agent-list net2
+-----+
| id              | host  | disabled | state |
+-----+
| a0c1c21c-d4f4-4577-9ec7-908f2d48622d | HostA | False   | :- ) |
+-----+

gongysh@gongysh-laptop:~$ quantum subnet-create net2 9.0.1.0/24 --name
subnet2
Created a new subnet:
+-----+
| Field          | Value                                     |
+-----+
| allocation_pools | {"start": "9.0.1.2", "end": "9.0.1.254"} |
| cidr            | 9.0.1.0/24                               |
| dns_nameservers |                                           |
| enable_dhcp     | True                                       |
| gateway_ip      | 9.0.1.1                                   |
| host_routes     |                                           |
| id              | 6979b71a-0ae8-448c-aa87-65f68eedcaaa   |
| ip_version      | 4                                         |
| name            | subnet2                                   |
| network_id     | 9b96b14f-71b8-4918-90aa-c5d705606b1a   |
| tenant_id     | b7445f221cda4f4a8ac7db6b218b1339     |
+-----+
```

We can see it is still allocated to DHCP agent on HostA. If we want to validate the behavior via dnsmasq, don't forget to create a subnet for the network since DHCP agent starts the dnsmasq only if there is a DHCP enabled subnet on it.

2. Assign a network to a given DHCP agent

Since we have two DHCP agents, we want another DHCP agent to host the network too.

```
gongysh@gongysh-laptop:~$ quantum dhcp-agent-add-network
f28aa126-6edb-4ea5-a81e-8850876bc0a8 net2
Added network net2 to dhcp agent
gongysh@gongysh-laptop:~$ quantum dhcp-agent-list net2
```

id	host	disabled	state
a0c1c21c-d4f4-4577-9ec7-908f2d48622d	HostA	False	: -)
f28aa126-6edb-4ea5-a81e-8850876bc0a8	HostB	False	: -)

We can see Both DHCP agents are hosting 'net2' network.

3. Remove a network from a given DHCP agent

This command is the sister one of previous command. Let's remove 'net2' from HostA's DHCP agent.

```
gongysh@gongysh-laptop:~$ quantum dhcp-agent-remove-network a0c1c21c-
d4f4-4577-9ec7-908f2d48622d net2
Removed network net2 to dhcp agent
gongysh@gongysh-laptop:~$ quantum dhcp-agent-list net2
```

id	host	disabled	state
f28aa126-6edb-4ea5-a81e-8850876bc0a8	HostB	False	: -)

We can see now only HostB's DHCP agent is hosting 'net2' network.

• Experiment HA of DHCP agent

First we will boot a VM on net2, then we let both DHCP agents host 'net2'. After that, we fail the agent in turn and to see if the VM can still get the IP during that time.

1. Boot a VM on net2

```
gongysh@gongysh-laptop:~$ quantum net-list
```

id	name	subnets
89dca1c6-c7d4-4f7a-b730-549af0fb6e34	net1	f6c832e3-9968-46fd-8e45-d5cf646db9d1

```

| 9b96b14f-71b8-4918-90aa-c5d705606b1a | net2 | 6979b71a-0ae8-448c-
aa87-65f68eedcaaa |
+-----+-----+-----+-----+
+-----+
gongysh@gongysh-laptop:~$ nova boot --image tty --flavor 1 myserver4 --nic
net-id=9b96b14f-71b8-4918-90aa-c5d705606b1a
gongysh@gongysh-laptop:~$ nova list
+-----+-----+-----+-----+
+-----+
| ID | Name | Status | Networks
+-----+-----+-----+-----+
| c394fcd0-0baa-43ae-a793-201815c3e8ce | myserver1 | ACTIVE | net1=10.0.1.
3 |
| 2d604e05-9a6c-4ddb-9082-8a1fbdcc797d | myserver2 | ACTIVE | net1=10.0.1.
4 |
| c7c0481c-3db8-4d7a-a948-60ce8211d585 | myserver3 | ACTIVE | net1=10.0.1.
5 |
| f62f4731-5591-46b1-9d74-f0c901de567f | myserver4 | ACTIVE | net2=9.0.1.
2 |
+-----+-----+-----+-----+
+-----+

```

2. Make sure both DHCP agents hosting 'net2'

We can use commands shown before to assign the network to agents.

```

gongysh@gongysh-laptop:~$ quantum dhcp-agent-list net2
+-----+-----+-----+-----+
| id | host | disabled | state |
+-----+-----+-----+-----+
| a0c1c21c-d4f4-4577-9ec7-908f2d48622d | HostA | False | :- ) |
| f28aa126-6edb-4ea5-a81e-8850876bc0a8 | HostB | False | :- ) |
+-----+-----+-----+-----+

```

3. Testing the HA

Step 1. Login to the myserver4 VM, and run 'udhcpd'

Step 2. Stop the DHCP agent on HostA (Beside stoping the quantum-dhcp-agent binary, we must make sure dnsmasq processes are gone too.)

Step 3. Run 'udhcpd' in VM. We can see it can get the wanted IP.

Step 4. Stop the DHCP agent on HostB too.

Step 5. Run 'udhcpd' in VM. We can see it cannot get the wanted IP.

Step 6. Start DHCP agent on HostB. We can see VM can get the wanted IP again.

- Disable and remove a component

Amin user wants to disable a component if there is a system upgrade planned, whatever hardware or software. Some components which support scheduling support disable or enable too, such as 'L3 agent' and 'DHCP agent'. Once the component is disabled, the

scheduler will not schedule new resources to the component. After the component is disabled, we can remove the component. However, we need to remove the resources on the component before we remove the component itself.

To run the commands below, we need first stop the DHCP agent on HostA.

```
gongysh@gongysh-laptop:~/git/python-quantumclient$ quantum component-update
--disable a0c1c21c-d4f4-4577-9ec7-908f2d48622d
Updated component: a0c1c21c-d4f4-4577-9ec7-908f2d48622d
gongysh@gongysh-laptop:~/git/python-quantumclient$ quantum component-list
+-----+-----+
| id                    | type                | host |
| disabled | state |
+-----+-----+
| 1b69828d-6a9b-4826-87cd-1757f0e27f31 | Linux bridge agent | HostA | False
| :-) |
| a0c1c21c-d4f4-4577-9ec7-908f2d48622d | DHCP agent          | HostA | True
| XXX |
| ed96b856-ae0f-4d75-bb28-40a47ffd7695 | Linux bridge agent | HostB | False
| :-) |
| f28aa126-6edb-4ea5-a81e-8850876bc0a8 | DHCP agent          | HostB | False
| :-) |
+-----+-----+
gongysh@gongysh-laptop:~/git/python-quantumclient$ quantum component-delete
a0c1c21c-d4f4-4577-9ec7-908f2d48622d
Deleted component: a0c1c21c-d4f4-4577-9ec7-908f2d48622d
gongysh@gongysh-laptop:~/git/python-quantumclient$ quantum component-list
+-----+-----+
| id                    | type                | host |
| disabled | state |
+-----+-----+
| 1b69828d-6a9b-4826-87cd-1757f0e27f31 | Linux bridge agent | HostA | False
| :-) |
| ed96b856-ae0f-4d75-bb28-40a47ffd7695 | Linux bridge agent | HostB | False
| :-) |
| f28aa126-6edb-4ea5-a81e-8850876bc0a8 | DHCP agent          | HostB | False
| :-) |
+-----+-----+
```

After deletion, if we restart the DHCP agent, it will in component list again.

Appendix B. Core Configuration File Options

quantum.conf

The configuration settings for the quantum services are found in `/etc/quantum/quantum.conf`.

Table B.1. Debugging Options

Configuration option=Default value	(Type) Description
debug=False	(BoolOpt) Print debugging output
verbose=False	(BoolOpt) Print more verbose output

Table B.2. Logging Options

Configuration option=Default value	(Type) Description
log_config=	(StrOpt) If this option is specified, the logging configuration file specified is used and overrides any other logging options specified. Please see the Python logging module documentation for details on logging configuration files. Print debugging output
log_format=%(asctime)s %(levelname)8s [% (name)s] %(message)s	(StrOpt) A logging.Formatter log message format string which may use any of the available logging.LogRecord attributes.
log_date_format=%Y-%m-%d %H:%M:%S	(StrOpt) Format string for %(asctime)s in log records.
log_file=	(StrOpt) (Optional) Name of log file to output to. If not set, logging will go to stdout.
log_dir=	(StrOpt) (Optional) The directory to keep log files in (will be prepended to <code>-logfile</code>).
use_syslog=False	(BoolOpt) Use syslog for logging.
syslog_log_facility=LOG_USER	(StrOpt) syslog facility to receive log lines

Table B.3. Service Options

Configuration option=Default value	(Type) Description
bind_host=0.0.0.0	(StrOpt) Server listening IP.
bind_port=9696	(IntOpt) Server listening port.
api_paste_config=api-paste.ini	(StrOpt) The paste configuration file. This is used to configure the WSGI application.
api_extensions_path=	(StrOpt) Enables custom addition to be made to the above configuration.
policy_file=policy.json	(StrOpt) JSON file representing policies to access and view data. The usage and format is discussed in more detail in the Authentication and Authorization section.
auth_strategy=keystone	(StrOpt) The strategy used for authentication. The supported values are 'keystone' and 'noauth'.
core_plugin=quantum.plugins.sample.SamplePlugin.FakePlugin	(StrOpt) The plugin to be loaded by the service.

Table B.4. Base Plugin Options

Configuration option=Default value	(Type) Description
base_mac=fa:16:3e:00:00:00	(StrOpt) MAC addresses for a port are generated. The first 3 octets will remain unchanged. If the 4h octet is not 00, it will also be used. The others will be randomly generated.
mac_generation_retries=16	(IntOpt) The number of times the plugin attempts to generate a unique MAC address.
allow_bulk=True	(BoolOpt) Enable or disable bulk create/update/delete operations.
max_dns_nameservers=5	(IntOpt) The maximum amount of DNS nameservers that can be configured per subnet.
max_subnet_host_routes=20	(IntOpt) The maximum amount of host routes that can be configured per subnet.
state_path=.	(StrOpt) Top level directory for configuration files.
dhcp_lease_duration=120	(IntOpt) The default expiration time for a DHCP address. This is in seconds.

Table B.5. Common RPC Message Options

Configuration option=Default value	(Type) Description
control_exchange=quantum	(StrOpt) AMQP exchange to connect to if using RabbitMQ or QPID
rpc_back_end=quantum.openstack.common.rpc.impl_kombu	(StrOpt) The messaging module to use, defaults to kombu. For qpid, make use of quantum.openstack.common.rpc.impl_qpid
rpc_thread_pool_size=64	(IntOpt) Size of RPC thread pool.
rpc_conn_pool_size=30	(IntOpt) Size of RPC connection pool.
rpc_response_timeout=60	(IntOpt) Seconds to wait for a response from call or multical
allowed_rpc_exception_modules='quantum.openstack.common.rpc.exception', 'nova.exception'	(ListOpt) Modules of exceptions that are permitted to be recreated upon receiving exception data from an rpc call.
fake_rabbit=False	(BoolOpt) If passed, use a fake RabbitMQ provider

Table B.6. Rabbit RPC Options

Configuration option=Default value	(Type) Description
kombu_ssl_version=	(StrOpt) SSL version to use (valid only if SSL enabled).
kombu_ssl_keyfile=	(StrOpt) SSL key file (valid only if SSL enabled)
kombu_ssl_certfile=	(StrOpt) SSL cert file (valid only if SSL enabled)
kombu_ssl_ca_certs=	(StrOpt) SSL certification authority file (valid only if SSL enabled)
rabbit_host=localhost	(StrOpt) IP address of the RabbitMQ installation
rabbit_password=guest	Password of the RabbitMQ server
rabbit_port=5672	(IntOpt) Port where RabbitMQ server is running/listening
rabbit_userid=guest	(StrOpt) User ID used for RabbitMQ connections
rabbit_virtual_host=/	(StrOpt) Location of a virtual RabbitMQ installation.
rabbit_max_retries=0	(IntOpt) Maximum retries with trying to connect to RabbitMQ. The default of 0 implies an infinite retry count
rabbit_retry_interval=1	(IntOpt) RabbitMQ connection retry interval

Table B.7. QPID RPC Options

Configuration option=Default value	(Type) Description
qpid_hostname=localhost	(StrOpt) Qpid broker hostname
qpid_port=5672	(IntOpt) Qpid broker port
qpid_username=	(StrOpt) Username for qpid connection
qpid_password=	(StrOpt) Password for qpid connection
qpid_sasl_mechanisms=	(StrOpt) Space separated list of SASL mechanisms to use for auth
qpid_reconnect=True	(BoolOpt) Automatically reconnect
qpid_reconnect_timeout=0	(IntOpt) The number of seconds to wait before deciding that a reconnect attempt has failed
qpid_reconnect_limit=0	(IntOpt) The limit for the number of times to reconnect before considering the connection to be failed.
qpid_reconnect_interval_min=0	(IntOpt) Minimum seconds between reconnection attempts
qpid_reconnect_interval_max=0	(IntOpt) Maximum seconds between reconnection attempts
qpid_reconnect_interval=0	(IntOpt) Equivalent to setting max and min to the same value
qpid_heartbeat=60	(IntOpt) Seconds between connection keepalive heartbeats
qpid_protocol=tcp	(StrOpt) Transport to use, either 'tcp' or 'ssl'
qpid_tcp_nodelay=True	(BoolOpt) Disable Nagle algorithm

Table B.8. Notification Options

Configuration option=Default value	(Type) Description
notification_driver=quantum.openstack.common.notifier.list_notifier	(MultiStrOpt) Driver or drivers to handle sending notifications. The default is set as notifier as the DHCP agent makes use of the notifications.
default_notification_level=INFO	(StrOpt) Default notification level for outgoing notifications
default_publisher_id=\$host	(StrOpt) Default publisher_id for outgoing notifications
list_notifier_drivers='quantum.openstack.common.notifier.notification_driver'	(MultiStrOpt) List of drivers to send notifications
notification_topics='notifications'	(ListOpt) AMQP topic used for openstack notifications


Table B.9. Quota Options

Configuration option=Default value	(Type) Description
quota_driver=quantum.quota.ConfDriver	(StrOpt) Default driver to use for quota checks. If the default driver is used then the configuration values below are in effect. To limit quotas per tenant then use: quantum.extensions._quotav2_driver.DbQuotaDriver
quota_items=network,subnet,port	(ListOpt) Resource names that are supported by the Quotas feature.
default_quota=-1	(IntOpt) Default number of resources allowed per tenant, minus for unlimited
quota_network=10	(IntOpt) Number of networks allowed per tenant, and minus means unlimited
quota_subnet=10	(IntOpt) Number of subnets allowed per tenant, and minus means unlimited
quota_port=50	(IntOpt) Number of ports allowed per tenant, and minus means unlimited

ovs_quantum_plugin.ini:

For information about the Open vSwitch plugin configurations, see <http://wiki.openstack.org/ConfigureOpenvswitch>.

Table B.10. Database Access by Plugin

Configuration option=Default value	(Type) Description
sql_connection=sqlite://	<p>(StrOpt) The details of the database connection. For example <code>mysql://root:nova@127.0.0.1:3306/ovs_quantum</code>. Replace 127.0.0.1 above with the IP address of the database used by the main quantum server. (Leave it as is if the database runs on this host.).</p> <div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;">  </div> <div> <p>Note</p> <p>This line must be changed to ensure that the database values are persistent. The <code>sqlite</code> is used for testing.</p> </div> </div>
sql_max_retries=10	(IntOpt) Database re-connection retry times. This is in the event connectivity is lost with the database. -1 implies an infinite retry count.
reconnect_interval=2	(IntOpt) Database reconnection interval in seconds - in event connectivity is lost.

OVS Options (This is in the section OVS)

This section deals with the specific OVS options common to the plugin and agent.

Table B.11. OVS Options Common to Plugin and Agent

Configuration option=Default value	(Type) Description
network_vlan_ranges=default:2000:3999	(ListOpt) Comma-separated list of <code><physical_network>:<vlan_min>:<vlan_max></code> tuples enumerating ranges of VLAN IDs on named physical networks that are available for allocation.
tunnel_id_ranges=	(ListOpt) Comma-separated list of <code><tun_min>:<tun_max></code> tuples enumerating ranges of GRE tunnel IDs that are available for allocation.
integration_bridge=br-int	(StrOpt) This is the name of the OVS integration bridge. There is one per hypervisor. The integration bridge acts as a virtual "patch port". All VM VIFs are attached to this bridge and then "patched" according to their network connectivity. Do not change this parameter unless you have a good reason to.
tunnel_bridge=br-tun	(StrOpt) Specifies the name of the OVS tunnel bridge used by the agent for GRE tunnels. Only used if <code>tunnel_id_ranges</code> is not empty.
bridge_mappings=default:br-eth1	(ListOpt) Comma-separated list of <code><physical_network>:<bridge></code> tuples mapping physical network names to agent's node-specific OVS bridge names. Each bridge must exist, and should have physical network # interface configured as a port.
local_ip=10.0.0.3	(StrOpt) Set local-ip to be the local IP address of this hypervisor. This is used only when <code>tunnel_id_ranges</code> are used.
enable_tunneling=False	(BoolOpt) A flag indicating if tunneling is supported. Not all systems that support Open vSwitch support its GRE tunneling feature, that is, it is not supported in the Linux

Configuration option=Default value	(Type) Description
	kernel source tree. This applies to both the server and agent

(This is in the section AGENT)

Table B.12. Agent Options

Configuration option=Default value	(Type) Description
rpc=True	(BoolOpt) If this is True then the agent will communicate with the plugin via the OpenStack RPC (configured in quantum.conf). If this is False then the agent will poll the database for changes. In the event that this is False then please update the relevant database settings on the agent so that it is able to access the database.
polling_interval=2	(IntOpt) Agent's polling interval in seconds.
root_helper=sudo	(StrOpt) This enables limiting commands that can be run. Please refer to rootwrap section for more details.

linuxbridge_conf.ini:

Table B.13. Database Access by Plugin


Configuration option=Default value	(Type) Description
sql_connection=sqlite://	<p>(StrOpt) The details of the database connection. For example mysql://root:nova@127.0.0.1:3306/ovs_quantum. Replace 127.0.0.1 above with the IP address of the database used by the main quantum server. (Leave it as is if the database runs on this host.)</p> <div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;">  </div> <div> <p>Note</p> <p>This line must be changed to ensure that the database values are persistent. The sqlite is used for testing.</p> </div> </div>
sql_max_retries=10	(IntOpt) Database re-connection retry times. This is in the event connectivity is lost with the database. -1 implies an infinite retry count.
reconnect_interval=2	(IntOpt) Database reconnection interval in seconds - in event connectivity is lost.

Table B.14. VLAN Configurations

Configuration option=Default value	(Type) Description
network_vlan_ranges=default:1000:2999	(ListOpt) Comma-separated list of <physical_network>:<vlan_min>:<vlan_max> tuples enumerating ranges of VLAN IDs on named physical networks that are available for allocation.

Table B.15. Networking Options on the Agent

Configuration option=Default value	(Type) Description
physical_interface_mappings=default:eth1	(ListOpt) Comma-separated list of <physical_network>:<physical_interface> tuples mapping

Configuration option=Default value	(Type) Description
	physical network names to agent's node-specific physical network interfaces. Server uses physical network names for validation but ignores interfaces.

Table B.16. Agent Options

Configuration option=Default value	(Type) Description
rpc=True	(BoolOpt) If this is True then the agent will communicate with the plugin via the OpenStack RPC (configured in quantum.conf). If this is False then the agent will poll the database for changes. In the event that this is False then please update the relevant database settings on the agent so that it is able to access the database.
polling_interval=2	(IntOpt) Agent's polling interval in seconds.
root_helper=sudo	(StrOpt) This enables limiting commands that can be run. Please refer to rootwrap section for more details.

dhcp_agent.ini

Table B.17. DHCP Specific Options

Configuration option=Default value	(Type) Description
root_helper=sudo	(StrOpt) This enables limiting commands that can be run. Please refer to rootwrap section for more details.
dhcp_driver=quantum.agent.linux.dhcp.Dnsmasq	(StrOpt) The driver used to manage the DHCP server.
use_namespaces=True	(BoolOpt) Allow overlapping IP. (Note: If running multiple agents with different IP addresses on the same host this should be "True" if not you will get routing problems.)

Table B.18. Device Manager Options

Configuration option=Default value	(Type) Description
admin_user=	(StrOpt) Admin Quantum user defined in keystone
admin_password=	(StrOpt) The users password
admin_tenant_name=	(StrOpt) The admin tenant name
auth_url=	(StrOpt) The URL used to validate tokens. `auth_protocol`:`auth_host`:`auth_port`/v2.0
auth_strategy=keystone	(StrOpt) The strategy to use for authentication. Supports noauth or keystone.
auth_region=	(StrOpt) The authentication region
interface_driver=	(StrOpt) The driver used to manage the virtual interface.
dhcp_lease_relay_socket=\$state_path/dhcp/lease_relay	(StrOpt) Location to DHCP lease relay UNIX domain socket
ovs_integration_bridge=br-int	(StrOpt) Name of Open vSwitch bridge to use. Only relevant if using Open vSwitch.
network_device_mtu=	(StrOpt) MTU setting for device. Only relevant if using Open vSwitch.
ryu_api_host=127.0.0.1:8080	(StrOpt) OpenFlow Ryu REST API host:port. Only relevant if using Ryu.

Configuration option=Default value	(Type) Description
meta_flavor_driver_mappings=	(StrOpt). Mappings between flavors and drivers. Only relevant if using MetaPlugin.
resync_interval=30	(IntOpt) If there is an exception on the Quantum service then the DHCP agent will ensure that it syncs with the Quantum configuration. The validation about syncing is done every resync_interval seconds.

Table B.19. dnsmasq Options

Configuration option=Default value	(Type) Description
dhcp_confs=\$state_path/dhcp	(StrOpt) Location to store DHCP server config files
dhcp_lease_time=120	(IntOpt) Lifetime of a DHCP lease in seconds
dhcp_domain=openstacklocal	(StrOpt) Domain to use for building the hostnames
dnsmasq_config_file=	(StrOpt) Override the default dnsmasq settings with this file
dnsmasq_dns_server=	(StrOpt) Use another DNS server before any in /etc/resolv.conf.

I3_agent.ini

Table B.20. L3 Specific Options

Configuration option=Default value	(Type) Description
root_helper=sudo	(StrOpt) This enables limiting commands that can be run. Please refer to rootwrap section for more details.
external_network_bridge=br-ex	(StrOpt) Name of bridge used for external network traffic.
use_namespaces=True	(BoolOpt) Allow overlapping IP. (Note: If running multiple agents with different IP addresses on the same host this should be "True" if not you will get routing problems.)
polling_interval=3	(IntOpt) The time in seconds between state poll requests.
metadata_ip=	(StrOpt) IP address used by Nova metadata server.
metadata_port=8775	(IntOpt) TCP Port used by Nova metadata server.
router_id=	(StrOpt) If namespaces is disabled, the I3 agent can only configure a router whose ID matches this parameter.
handle_internal_only_routers=True	(BoolOpt) Agent should implement routers with no gateway.
gateway_external_network_id=	(StrOpt) UUID of external network for routers implemented by the agents.

Table B.21. Device Manager Options

Configuration option=Default value	(Type) Description
admin_user=	(StrOpt) Admin Quantum user defined in keystone
admin_password=	(StrOpt) The users password
admin_tenant_name=	(StrOpt) The admin tenant name
auth_url=	(StrOpt) The URL used to validate tokens. `auth_protocol`:`auth_host`:`auth_port`/v2.0
auth_strategy=keystone	(StrOpt) The strategy to use for authentication. Supports noauth or keystone.

Configuration option=Default value	(Type) Description
auth_region=	(StrOpt) The authentication region
interface_driver=	(StrOpt) The driver used to manage the virtual interface.