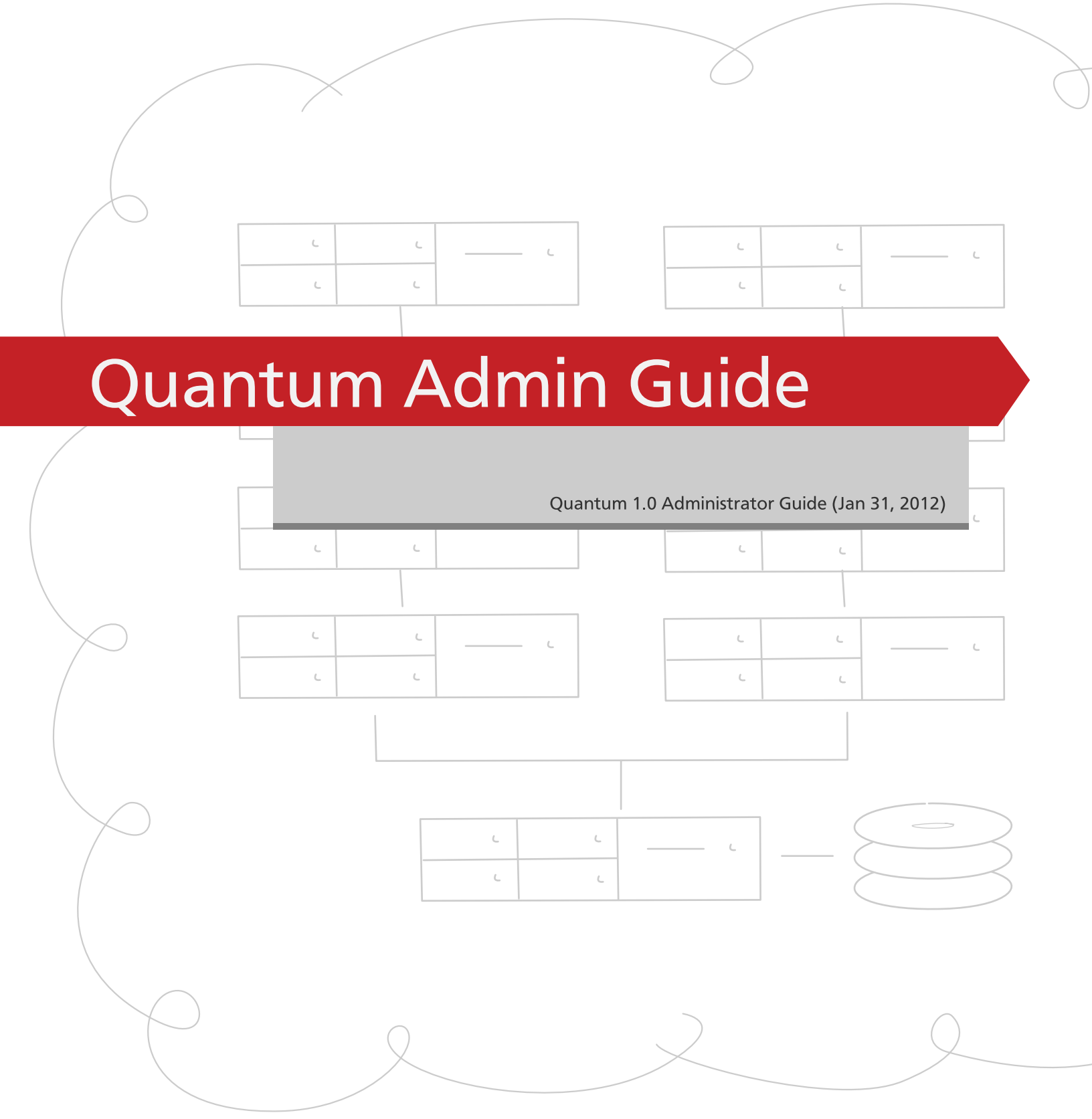

Quantum Admin Guide

Quantum 1.0 Administrator Guide (Jan 31, 2012)

openstack™

docs.openstack.org

Quantum Admin Guide

Quantum 1.0 Administrator Guide (2012-01-31)

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

Table of Contents

1. Preface

Quantum is a project to provide "network connectivity as a service" between devices managed by other OpenStack services (e.g., vNICs from the OpenStack Nova compute service). For more information on Quantum and the other network-related projects please check the Quantum home page (wiki.openstack.org/Quantum) and the NetStack home page (wiki.openstack.org/Network).

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

Intended Audience

This guide is intended to assist OpenStack administrators looking to run an OpenStack cloud that leverages Quantum for advanced networking. This document only covers the process of installing, configuring, and running Quantum. Information about programming against the Quantum API is found in the Quantum API Guide.

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

- Openvswitch - Implementing Quantum with Open vSwitch for KVM and XenServer compute platforms.
- Cisco - Implementing Quantum for deployments using Cisco UCS blades and Nexus switches.

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
Jan 31, 2012	• Essex-3 update (2).
Dec 15, 2011	• Essex-2 update.
Sep 22, 2011	• Initial draft.

2. Overview

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

What is Quantum?

Quantum is a "virtual network service" that aims to provide a powerful API to define the network connectivity between devices from other OpenStack services. The Quantum service has an API that presents a logical abstraction for describing network connectivity. The service relies on a Quantum plugin to manage virtual and/or physical switches within the cloud data center to make sure those devices forward packets according to the behavior defined in the logical API model. This notion of a plugin is similar to how Nova can use different types of hypervisors to implement the same virtual server API.

The Quantum API utilizes the following logical abstractions:

- Network: An isolated L2 segment, analogous to a single physical L2 switching device with an arbitrary number of ports.
- Port: Provide a connection point to a Quantum network. Ports can also be configured to apply various network security, QoS, monitoring, etc. policies. Such policies are currently exposed by API extensions.
- Attachment: Identifier of an interface device to be "plugged in" to a Quantum port, such as a vNIC from Nova.

For a detailed description of the Quantum API abstractions and their attributes, please see the Quantum API Guide.

Quantum Architecture

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

Switching Infrastructure

At its core, Quantum controls the network connectivity seen by other OpenStack resources like Nova vNICs by managing the switching infrastructure within the cloud data center. Exactly which switches need to be managed depends on the plugin in use. For example, a plugin may manage only the vswitch running on compute server, or it may manage both the vswitch and adjacent physical switches.

Quantum-Aware OpenStack Services

A Quantum-aware OpenStack service (e.g., Nova) must inform Quantum about individual interface devices (e.g., Nova vNICs) that can be plugged into Quantum networks. This interaction enables a Quantum plugin to associate a port on a managed virtual/physical switch with a particular interface device identifier (and thus, ultimately with a port on a Quantum network).

Quantum Service

The Quantum Service is a python process that runs the Quantum API webserver and is responsible for loading a Quantum plugin and passing each API call to the Quantum plugin for processing. This process is commonly run on a "controller" host along with other OpenStack services (e.g., nova-api, nova-scheduler), but can also be run stand-alone.

Quantum Plugin

The role of the Quantum plugin is to translate logical network modifications received from the Quantum Service API and map them to specific operations on the switching infrastructure. A plugin may be open source or proprietary, and may be specific to a single type of switching infrastructure, or work across switches of many different types/vendors. Plugins are able to expose advanced capabilities beyond L2 connectivity using API extensions.

3. Quantum Setup

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

Selecting a Host

The Quantum Service is a python process, similar to other OpenStack projects like OpenStack Nova. If your deployment uses a "controller host" to run centralize 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.

If you are building a host from scratch to use for Quantum, we recommend using a recent versions of Ubuntu (10.10 or newer) or Red Hat Enterprise Linux (version 6.x) as these are the best-tested platforms.

Installing Quantum

This section describes the different options for installing Quantum. Regardless of how it is installed, a Quantum installation provides two executables:

- quantum-server : a daemon that implements the Quantum virtual network service. An init script should be installed (e.g., /etc/init.d/quantum-server) for starting/stopping the service and logs are sent to /var/log/quantum/quantum-server.log .
- quantum : a CLI utility for manually driving the Quantum API (note: in most common use cases, the Quantum API is primarily driven by another service, for example OpenStack Nova, and this CLI is used only for simple testing or troubleshooting).

Quantum has two configuration files quantum.conf and plugins.ini . The location of these files, referred to as $QUANTUM_CONF_DIR in text below, depends on the installation option chosen and is described below.

Installing from Packages

As with other OpenStack components such as Nova, Glance, etc., the preferred mechanism for getting Quantum is installing it via packages that are compatible with your Linux platform. A list of Linux distributions that package Quantum is available at: http://wiki.openstack.org/QuantumPackages.

After installing from packages, $QUANTUM_CONF_DIR is /etc/quantum.

Installing from Source

Note

Quantum packaging changed significantly for Essex-3, with the CLI and API client code being separated into a different package. The instructions below will work only for releases Essex-3 and beyond.

If you cannot install Quantum from distribution packages, you can download the both the python-quantumclient-<version>.tar.gz and quantum-<version>.tar.gz files from https://launchpad.net/quantum/+download.

Note

The Quantum server code currently depends on code in python-quantumclient, so the client code must always be downloaded and installed.

First, install python setup tools. For example, on Ubuntu:

```
sudo apt-get install python-setuptools python-dev libxslt1-dev
```

Note

Due to a problem with the Essex-3 tarball, there is an extra step in the instructions below to copy a missing __init__.py file into the python-quantumclient directory.

Then install the client and server packages using setup tools. For example, on Ubuntu:

```
tar xzf python-quantumclient-<version>.tar.gz
tar xzf quantum-<version>.tar.gz
cp quantum-<version>/quantum/__init__.py python-quantumclient-<version>/
quantum/__init__.py
cd python-quantumclient-<version>
sudo python setup.py install
cd ..
cd quantum-<version>
sudo python setup.py install
```

After the install, the executables "quantum-server" and "quantum" should be in your path, just as if you had installed from packages.

Run setup.py with the "-h" option to see other install options.

After installing from source, files normally placed in /etc are instead placed in /usr/local/lib/python2.7/dist-packages/quantum-2012.1-py2.7.egg/etc/ . As a result, you MUST copy these files to /etc:

```
sudo cp -r /usr/local/lib/python2.7/dist-packages/quantum-2012.1-py2.7.egg/
etc/quantum /etc
sudo cp /usr/local/lib/python2.7/dist-packages/quantum-2012.1-py2.7.egg/etc/
init.d/quantum-server /etc/init.d
sudo chmod +x /etc/init.d/quantum-server
```

No Installation (Running from Source)

Note

Running from source with no installation is primarily useful for developers and is usually not relevant to users.

You only need to download the main quantum-<version>.tar.gz file from https://launchpad.net/quantum/+download, as the python-quantumclient code will be installed as a dependency.

First, install dependencies. For example, on Ubuntu:

```
sudo apt-get install python-setuptools python-pip python-virtualenv
```

Note

Due to a problem with the Essex-3 tarballs, the pip-requires file is not available in the tarball itself. It can be downloaded from https://github.com/openstack/quantum/blob/9a1823d72ba84f6f98a9870970a7cd6eab25822d/tools/pip-requires and manually placed in the tools directory referenced below.

Next, untar the code and install dependencies using pip. On Ubuntu:

```
tar xzf quantum-<version>.tar.gz
cd quantum-<version>
sudo pip install -r tools/pip-requires
```

Then, from within the Quantum source directory, you can run the unit tests to confirm that the dependencies are correctly installed:

```
./run_tests.sh -N
```

All tests should print a green 'OK'. The end of the test output may print a few pep8 errors depending on your version of pep8 installed. These can be safely ignored.

When running from source, be sure source that your PYTHONPATH environment variable includes the root directory of the Quantum source tree, and that the executables in the bin/ directory of the source tree are in your PATH environment variable.

The quantum-server binary will be run from the local bin/ directory within the source, and the quantum CLI client will be installed as part of the pip install.

When running from source, $QUANTUM_CONF_DIR is the etc/ directory within the Quantum source directory. For example, from the top of the Quantum source directory, run:

```
./bin/quantum-server
```

Selecting a Plugin

The plugin used by the Quantum Service is configured using the plugins.ini found in $QUANTUM_CONF_DIR. By default, Quantum is configured to use a "FakePlugin" that will let you make API calls, but does not actually manage any switches. To change to another plugin, edit the following line in the plugins.ini file to point to your plugin of choice:

```
provider = quantum.plugins.SamplePlugin.FakePlugin
```

Quantum plugins can be distributed with the main Quantum binaries or distributed separately.

The plugin should include its own documentation indicating plugin-specific dependencies, installation procedures, plugin-specific configuration files, and the "provider" value that should be specified in the main plugins.ini configuration file.

The documentation for the plugins shipped with Quantum are located in:

- Open vSwitch: http://www.openvswitch.org/openstack/documentation
- Cisco: quantum/plugins/cisco/README

Running the Service

To start the service, the recommend approach is to use the init script:

```
/etc/init.d/quantum-server start
```

The service can also be invoked directly:

```
quantum-server
```

A successful start of the Quantum service with default settings will have output that ends with a line similar to. If running from an init script, this output will be in the file /var/log/quantum/quantum-server.log .

```
DEBUG [eventlet.wsgi.server] (2094) wsgi starting up on http://0.0.0.0:9696/
```

By default the service uses TCP port 9696 and listens on all IP addresses. To edit this and other Quantum Service settings, edit quantum.conf in the Quantum configuration directory. Editing this file can also be used to enable any extensions supported by the plugin. See your plugin documentation for more details.

To use a non-standard location for quantum.conf, run quantum-server directly and provide it as an argument when running the server:

```
quantum-server /path/to/quantum.conf
```

Validating the Setup

You can use the basic Quantum API client CLI to validate your setup. While still running the Quantum service, run the following command from the same host to confirm that the client can communicate with Quantum service API running your plugin:

```
quantum create_net quantum-fake-tenant net1
```

This command creates a Quantum network named 'net1' for a tenant named 'quantum-fake-tenant'. The resulting output should resemble:

```
Created a new Virtual Network with ID: 0a649eca-3764-417c-91a7-eb51291d4bb9
for Tenant: quantum-fake-tenant
```

This validation confirms that the Quantum service is able to communicate with the Quantum plugin and that the Quantum plugin is able to perform basic API operations. It does not test whether the Quantum plugin is correctly communicating with your switch infrastructure. See your plugin documentation for additional validation steps.

To experiment more with basic Quantum API commands, invoke the CLI utility with no arguments to see all available commands:

```
list_nets [tenant-id]
rename_net [tenant-id] [net-id] [new-name]
show_port [tenant-id] [net-id] [port-id]
```

```
unplug_iface [tenant-id] [net-id] [port-id]
plug_iface [tenant-id] [net-id] [port-id] [iface-id]
show_net [tenant-id] [net-id]
delete_port [tenant-id] [net-id] [port-id]
delete_net [tenant-id] [net-id]
set_port_state [tenant-id] [net-id] [port-id] [new_state]
create_net [tenant-id] [net-name]
create_port [tenant-id] [net-id]
list_ports [tenant-id] [net-id]
```

Invoking the CLI utility with no arguments displays additional command-line options to, for example, specify an alternate Quantum server IP address or port.

4. Using Quantum with Nova Compute

This chapter covers how to configure the OpenStack Nova compute service to communicate with Quantum using a special network manager class called the Quantum Manager.

Note

The Quantum Manager code within Nova is rapidly adding new capabilities during the "essex" release cycle. As a result, some of the functionality documented in this section will only be available if you are running Essex-2 milestone or newer code.

The final section of this chapter describes another model of using Quantum with Nova that does not require Quantum Manager, but requires manually associating vNICs with Quantum networks.

Enabling Quantum Manager

Within Nova, the nova-network process manages VM network connectivity and related network capabilities (e.g., IP Address management, DHCP, L3 + NAT forwarding, VPN). Nova supports plugging in different implementation of "managers" for this network service, and Quantum takes advantage of this by creating a standard Quantum Manager that communicates with Quantum to allow Nova and Quantum to integrate.

To enable the Quantum Manager, your nova-network service must specify the following flag:

```
--network_manager=nova.network.quantum.manager.QuantumManager
```

Nova defaults to connecting to Quantum on localhost using the standard port. The "–quantum_connection_host" and "–quantum_connection_port" flags can override these defaults.

Quantum Compatible Vif-Plugging in Nova

As described in the Architecture section, a service like Nova must be made aware of Quantum such that a Quantum Plugin can associate an attachment-id passed in via the Quantum API with a port on a vswitch or physical switch managed by the plugin.

In Nova, this is done by specifying that the nova-compute service use a type of "vif-plugging" that is compatible with your Quantum plugin. A Quantum plugin's documentation should specify the nova-compute flags it requires for vif-plugging. Usually this will include setting the vif-driver flag for your virt-layer (e.g., "libvirt_vif_driver" or "xenapi_vif_driver") to point to a vif-driver class that is specified in your plugin documentation. Depending on the vif-driver selected, the plugin documentation may require that additional configuration flags are set as well (e.g., "libvirt_vif_type").

Defining VM Network Connectivity

With Nova, the network manager is responsible for much more than simply creating L2 connectivity. Among other things, the network manager is responsible for:

- Defining vNICs when a VM is first created.
- Attaching VM vNICs to an L2 network.
- Assigning each VM vNIC a fixed IP address from a subnet associated with its L2 network.
- Optionally providing a "gateway router" to connect private L2 networks to a public network using NAT.

By default, the Quantum Manager determines the set of vNICs and fixed IPs for a VM based on networks statically created by the cloud administrator using nova-manage (see below for instructions on creating such networks).

For each vNIC created, the Quantum Manager will create a port on the Quantum network and attach the vNIC. If the VM is later terminated, Quantum Manager will destroy the associated Quantum port.

Creating Networks

To create a network using Quantum Manager, run:

```
nova-manage network create --label=public --fixed_range_v4=8.8.8.0/24
```

When this command is invoked, the Quantum Manager will contact the Quantum Service to create a corresponding Quantum network, and likewise will create an IPAM subnet using the specified fixed range.

By default, networks are "global" in the sense that they are shared by all tenants (i.e. projects in Nova). To create a tenant-specific network, specify the "project_id" flag when creating the network. For example:

```
nova-manage network create --label=tenant2-private --
fixed_range_v4=10.0.0.0/24 --project_id=2
```

This project_id should be the integer 'tenant-id' value from keystone. To view a tenant's ID, run:

```
keystone-manage tenant list
```

When a new VM is created, it is given a vNIC for each global network, as well as a vNIC for each network owned by the tenant associated with the VM. This lets users create hybrid scenarios where a VM has a vNIC both on a shared public network and on a private tenant-specific network.

Network Priorities

To provide a mechanism for ordering multiple vNICs on a VM, Quantum Manager has a notion of an integer "priority" for each network. When a VM has multiple vNICs, the vNICs are added in ascending priority order (i.e., 0 is highest priority). The priority of a network can be specified at network creation. Consider a deployment where all VMs should have two vNICs: the first (e.g., eth0) connected to the shared public network and a second (e.g., eth1) connected to a private tenant network.

To do this, first create the shared network:

```
nova-manage network create --label=public --fixed_range_v4=8.8.8.0/24 --
priority=0
```

Then, for each with tenant with ID X, create a private network:

```
nova-manage network create --label=tenantX-private --
fixed_range_v4=10.0.0.0/24 --project_id=X --priority=1
```

Note

To allow the CIDR address range for different networks to overlap, you must be using Melange for IP Address Management (see: Using Melange IPAM below).

If a network is created with no "priority", the network has priority 0. Networks should be created such that a VM is never associated with multiple networks of the same priority, as the order of vNICs will be undefined.

Network Gateway Routers and Floating IPs

Note

For Essex-3, Quantum Manager will only implement gateway routers and floating IPs if DHCP is enabled. Quantum Manager will not reject calls to configure gateway routers or floating IPs, the the underlying forwarding will not happen. This will be fixed in Essex-4.

Optionally, each Quantum network can have a gateway router attached. Similar to a nova-network deployment with VLANManager, this gateway router functionality provide VMs on private networks with access to a public network. This gateway provides SNAT to a public network, as well as "floating IPs".

Currently, this gateway capability is implemented using the nova-network process that is running on one or more "network controller" nodes. This is essentially the same mechanism as used by Nova's "VLAN Manager". In this model, a network controller node has two NICs and acts as a NAT gateway between an "internal" network with compute hosts and an "external" network (e.g., used for Internet access).

With Quantum Manager, a gateway will be attached to a network if the "–gateway" flag is included in the nova-manage call to create the network. For example:

```
nova-manage network create --label=tenantX-private --
fixed_range_v4=10.0.0.0/24 --project_id=X --priority=1 --gateway=10.0.0.1
```

Similar to the VLAN Manager, the gateway implemented by the Quantum Manager also performs the rewriting of HTTP requests destined to the Metadata server IP address (169.254.169.254) to instead reach the IP address of Nova's meta-data service.

Floating IPs should be configured using the standard Nova commands for floating IPs. See: Nova Admin Manual on Floating IPs.

Leveraging Existing Quantum Networks

It is also possible to use nova-manage to create a network that refers to an existing Quantum network. The "uuid" flag passes an existing Quantum Network UUID to nova-manage, meaning that Quantum Manager does not need to contact the Quantum service to create a network. Instead the Quantum Manager will just create and delete ports on the network when VMs are created/deleted. For example:

```
nova-manage network create --label=public --fixed_range_v4=8.8.8.0/24 --
priority=0 --uuid=0a649eca-3764-417c-91a7-eb51291d4bb9
```

Specifying VM Network Connectivity on Boot

Note

This capability is broken due to a bug in the Essex-3 milestone, but is fixed in Essex-4.

Quantum Manager already supports more flexible vNIC-to-network associations using the 'os-create-server-ext' extension. The Quantum Manager supports the network association portion of this extension, but ignores any IP address data in the request.

IP Address Management (IPAM) and DHCP

Nova is responsible for IP Address Management (IPAM), assigning each VM vNIC a fixed IP address from a subnet associated with its L2 network.

By default, the Quantum Manager uses the existing IPAM database tables within Nova for managing fixed-ip assignments for VMs. Using this default is recommended for basic use.

Enabling Nova DHCP

Quantum Manager can leverage the same dnsmasq DHCP mechanism supported by other Nova Network Managers. DHCP is not enabled by default. It can be enabled system-wide with the following flag:

```
--quantum_use_dhcp=True
```

Using Melange IPAM

Quantum Manager also supports using the new standalone IPAM service from the Melange Project . Melange provides expert users with more flexibility in terms of managing how IP addresses are assigned to VMs or other devices on the network.

Using Melange requires downloading the Melange code (it is a separate project from both Quantum and Nova) and running the Melange service. To tell Nova to use Melange IPAM instead of the default, set the following flag:

```
--quantum_ipam_lib=nova.network.quantum.melange_ipam_lib
```

Nova defaults to connecting to Melange on localhost using the standard port. The "–melange_host" and "–melange_port" flags can override these defaults.

Quantum Manager Limitations

During Essex, Quantum has added many capabilities toward achieving "nova-network parity". The following limitations exist:

- Multi-host nova-network HA is not supported (i.e., running nova-network on each compute node for HA purposes). It is unlikely that Quantum Manager will support this mode in Essex.
- CloudPipe VPNs do not work when using Quantum Manager. Nova is developing an alternative to CloudPipe for Essex that should be compatible with Quantum Manager.

Quantum and Nova FlatManager

It is also possible to use Quantum and Nova together without using Quantum Manager, though it requires extra work on behalf of the user or orchestration software to associate vNICs with networks. The approach is to have nova-network use the FlatManager class with a single global IP subnet used by all VMs. This can be achieved using the following flags in the flags file for nova-network:

```
--network_manager=nova.network.manager.FlatManager
--flat_network_bridge=br100
--fixed_range=192.168.0.0/16
```

In addition, the flags file used by the compute node must specify the vif-driver flags (e.g., "libvirt_vif_driver") associated with your Quantum plugin. These flags should be described in your plugin's documentation. The "flat_network_bridge" flag must be specified as well, though commonly it is can be set to any value in the, all Quantum-aware vif-drivers ignore this flag (it must be specified however, otherwise the FlatManager code will produce an error). As mentioned in the OpenStack documentation, in Flat Mode, the IP addresses for VM instances are grabbed from the subnet specified in the "fixed_range" flag, and then injected into the image on launch. Each instance receives a fixed IP address from the pool of available addresses. The configuration injection currently only works on Linux-style systems that keep networking configuration in /etc/network/interfaces.

With this setup, VMs will be spawned with a single vNIC, but that vNIC will not be plugged into any Quantum port. You will need to create a Quantum port using a Quantum API client and associate the interface-id of the vNIC with that port. The interface-id for a Nova vNIC is exposed using a the 'os-virtual-interfaces' extension to the Nova API. Currently there are no CLI tools that list these interface-id in an easily consumable format. However, Quantum integration for the Dashboard project leverages this extension.

5. Quantum Keystone Authentication and Authorization

Requests to the Quantum API can be authenticated with the OpenStack Keystone identity service using a token-based authentication protocol. This is optional, and is only needed for production deployments that expose the Quantum API directly to tenants. Deployments where tenants only contact the Nova API, and Nova communicates with Quantum using the Quantum Manager (see above), do not require Quantum to use Keystone.

Keystone integration is disabled by default, as Quantum does not yet provide authorization, meaning that NOTHING IS DONE with existing Keystone identities by Quantum. We expect this to change in future Essex Milestones, but for the time being this portion of the document is purely experimental.

Configuring Keystone with Quantum

There are three steps to configuring Keystone with Quantum:

1.Running the Keystone Service

The Keystone identity service is a requirement. It must be installed, although not necessarily on the same machine where Quantum is running; both Keystone's admin API and service API should be running.

2.Enabling Authentication and Authorization within Quantum

Authentication and Authorization middleware should be enabled in the Quantum pipeline. To this aim, uncomment the following line in quantum.conf:

```
pipeline = authN extensions quantumapiapp
```

3. Configuring Quantum to Connect to Keystone

The final step concerns configuring access to Keystone. The following attributes must be specified in the [filter:authN] section of quantum.conf:

```
auth_host              IP address or host name of the server where Keystone is
 running
auth_port              Port where the Keystone Admin API is listening
auth_protocol       Protocol used for communicating with Keystone (http/https)
auth_version         Keystone API version (default: 2.0)
```

4. Setup Keystone Admin Credentials

In order to validate authentication tokens, Quantum uses Keystone's administrative API. It therefore requires credentials for an administrative user, which can be specified in Quantum's configuration file (quantum.conf) Either username and password, or an authentication token for an administrative user can be specified in the configuration file:

auth_admin_token Keystone token for administrative access auth_admin_user Keystone user with administrative rights auth_admin_password Password for the user specified with auth_admin_user

For example, using password:

```
auth_admin_user = admin auth_admin_password = secrete
```

Or using a token:

```
auth_admin_token = 9a82c95a-99e9-4c3a-b5ee-199f6ba7ff04
```

Note

auth_admin_token and auth_admin_user/password are exclusive. If both are specified, auth_admin_token has priority.

Client Authentication Requirements

A user should first authenticate with Keystone, supplying user credentials; the Keystone service will return an authentication token, together with information concerning token expirations and endpoint where that token can be used.

The authentication token must be included in every request for the Quantum API, in the 'X_AUTH_TOKEN' header. Quantum will look for the authentication token in this header, and validate it with the Keystone service.