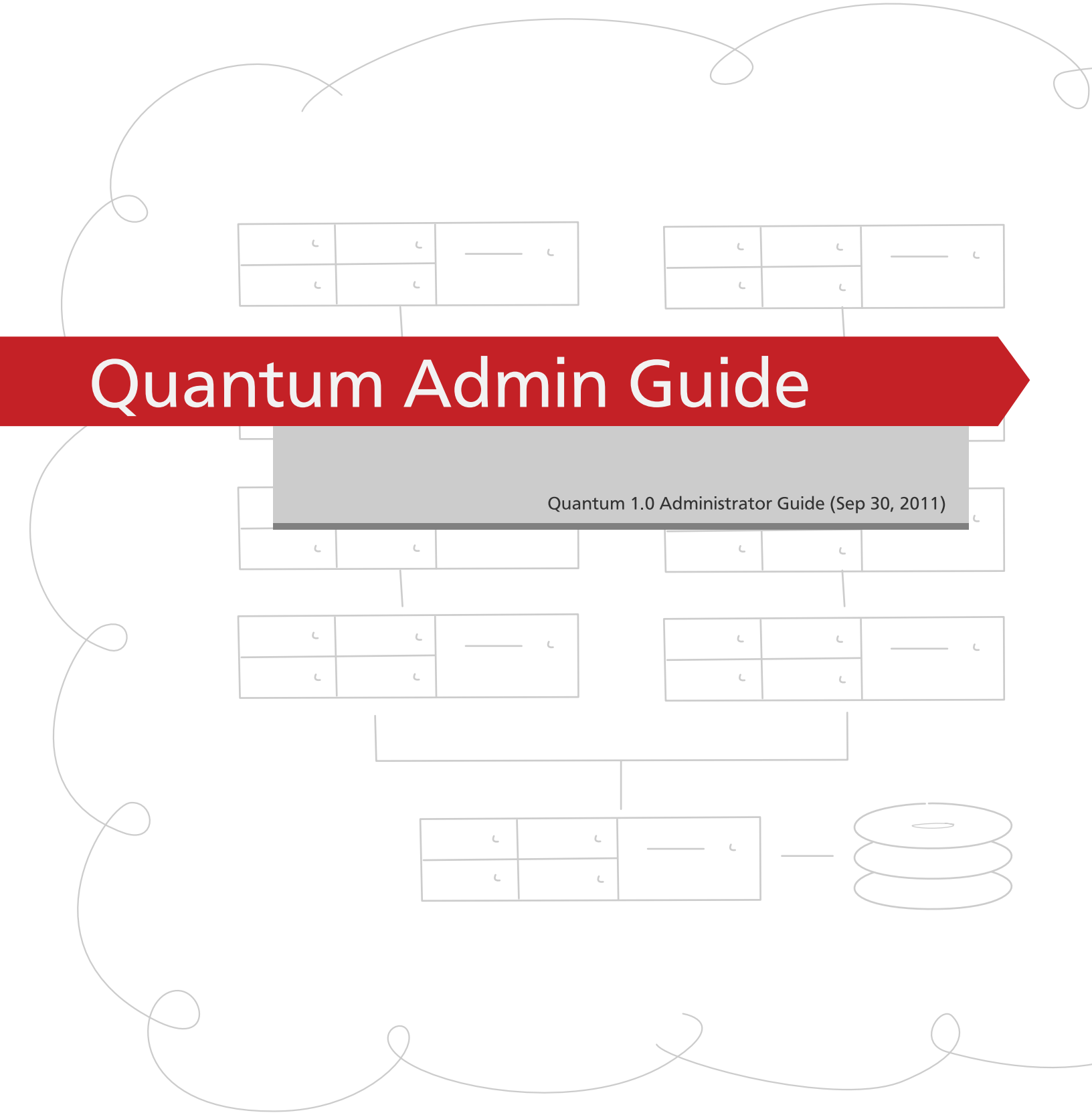

Quantum Admin Guide

Quantum 1.0 Administrator Guide (Sep 30, 2011)

openstack™

docs.openstack.org

Quantum Admin Guide

Quantum 1.0 Administrator Guide (2011-09-30)

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
September 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 platforms the most well-tested platforms.

Getting Quantum

There is currently no packaging for Quantum, so it must be deployed from source. The most recent official release of Quantum is the 2011.3 "diablo" release (9/2011), available at: http://launchpad.net/quantum/diablo/2011.3/+download/quantum-2011.3.tar.gz.

Since Quantum is still evolving rapidly, you may want to check the Quantum launchpad site for the most recent milestone release: http://launchpad.net/quantum.

Unarchive the file and change into the directory. All commands described in the rest of this document should be run from the top level of this directory unless otherwise stated. The Quantum distribution has the following key directories and files:

- bin: contains quantum service and CLI executables
- etc: contains service configuration files quantum.conf
- quantum: main source code directory
- quantum/plugins: directory containing plugins shipped with Quantum

Installing Dependencies

To install dependencies for the Quantum service, install 'pip' and use it to install all dependencies listed in tools/pip-requires . For example, on Ubuntu:

```
sudo apt-get install python-pip
sudo pip install tools/pip-requires
```

Please see the documentation for your Quantum plugin to see if there are any additional dependencies.

Selecting a Plugin

The plugin used by the Quantum Service is configured in the file quantum/plugins.ini . 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
```

Plugins distributed with Quantum are available in the quantum/plugins directory. You may also use a plugin that is distributed separately from the main Quantum code. If so, copy that plugin into the quantum/plugins directory.

The plugin should include its own documentation (e.g., a README file) indicating plugin-specific dependencies, plugin-specific configuration files, and the "provider" value that should be specified in the main quantum/plugins.ini configuration file. The README files for the plugins shipped with Quantum are located in:

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

Running the Service

To start the service, simply run:

```
bin/quantum
```

A successful start of the Quantum service with default settings will have output that ends with a line similar to:

```
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 etc/quantum.conf . Editing this file can also be used to enable any extensions supported by the plugin. See your plugin documentation for more details.

Validating the Setup

A good first step to validate the setup is to run the unit tests:

```
./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.

You can use the basic API client CLI that ships with Quantum 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:

```
```

```
bin/cli 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 help also 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.

The Quantum Manager code was added to Nova in the 2011.3 "diablo" release of Nova. The Quantum Manager instructions in this section will not work with any "diablo" milestone releases, or previous Nova releases.

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

The nova-network process manages VM network connectivity and related network services within Nova. 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, make sure your nova-network service is started with the following flag:

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

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 specied 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").

IP Address Management (IPAM)

By default, the Quantum Manager uses the existing IPAM database tables within Nova for managing fixed-ip assignments for VMs. Quantum Manager also supports using a new standalone IPAM service from the Melange project (http://wiki.openstack.org/Melange). To enable Melange IPAM, set the following flag:

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

Melange is still experimental and is being integrated into the Nova codebase early in the Essex cycle. See the Melange documentation for more information.

Defining VM 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:

- Determining the number and order of vNICs on each VM.
- Associating vNICs with an L2 network.
- IP Address Management (IPAM), assigning each vNIC a fixed IP address from a subnet associated with its Quantum network.

Similar to standard Nova multi-nic networking, 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.

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.

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 projects. To create a project-specific network, specify the "project_id" flag when creating the network. For example:

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

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 project 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.

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 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 projectX, create a private network:

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

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.

Note

In the future, Quantum Manager aims to extend the Nova API to expose a more flexible mechanism to determine the number of vNICs on each VM and their associations with Quantum networks. This will make it easier to create more advanced multi-tier network topologies.

Quantum Manager already supports more flexible vNIC-to-network associations using the 'os-create-server-ext' extension, though there are currently no user-friendly tools available to leverage this capability. The Quantum Manager supports the network association portion of this extension, but ignores any IP address data in the request.

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
```

Quantum Manager Limitations

The 2011.3 diablo release is the first release of QuantumManager, and only a subset of Nova networking functionality is supported in this initial release. Specifically, there are the following limitations:

- Nova's mechanism for providing DHCP using dnsmasq is not supported. This means that for Nova virt drivers that do not support IP injection (e.g., libvirt), VMs will not automatically receive IP addresses.
- Floating IP addresses are not supported.
- NAT gateway and EC2 metadata server are not supported.
- If Quantum Manager is controlling the networking for a tenant, you should not create or modify ports for that tenant using the Quantum CLI or other API clients.

We expect most of these limitations to be addressed by enhancements to Quantum Manager during the Essex timeframe.

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.