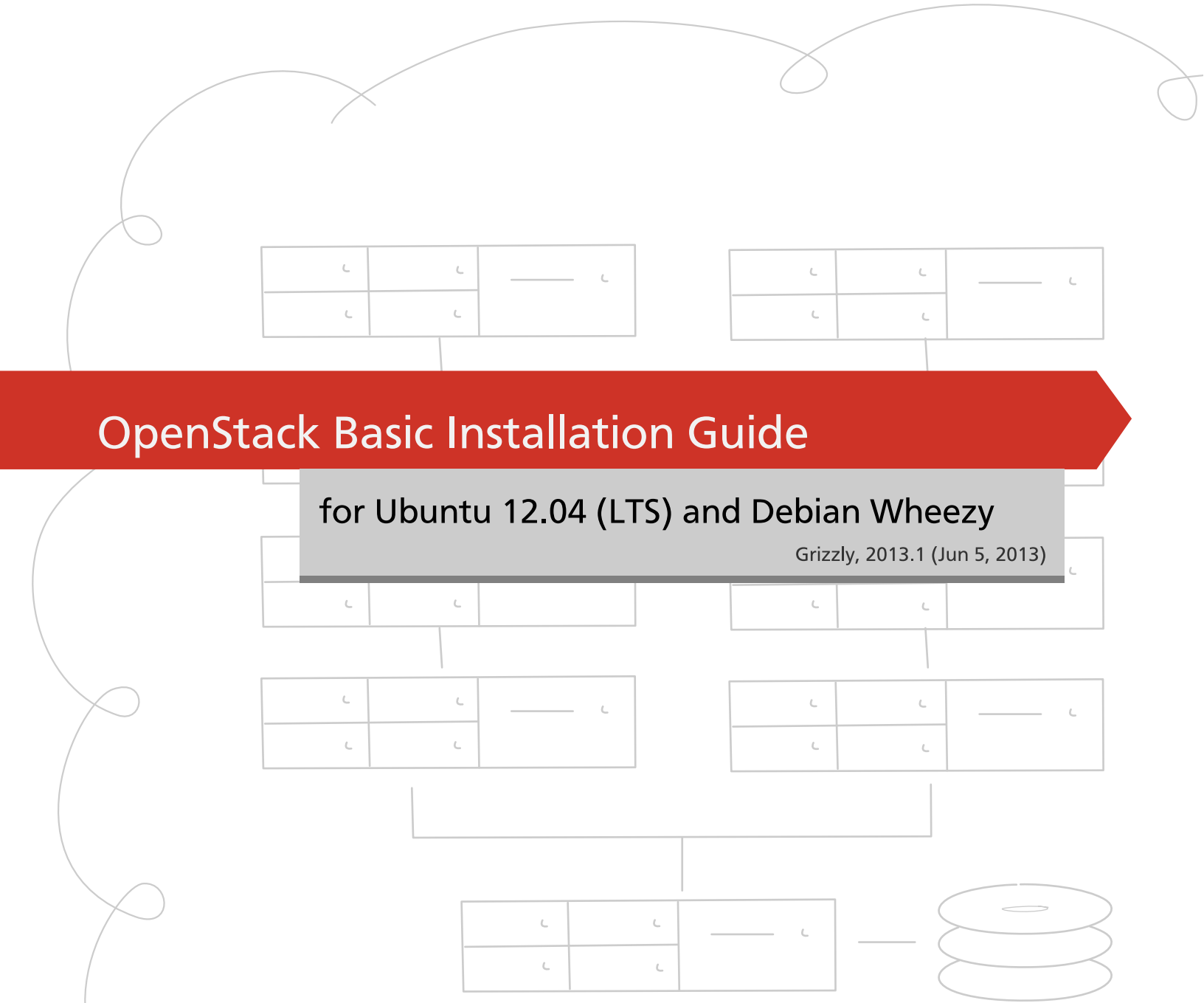
OpenStack Basic Installation Guide

for Ubuntu 12.04 (LTS) and Debian Wheezy

Grizzly, 2013.1 (Jun 5, 2013)



openstack™

docs.openstack.org

OpenStack Basic Installation Guide for Ubuntu 12.04 (LTS) and Debian Wheezy

Grizzly, 2013.1 (2013-06-05)

This document is intended for software developers interested in developing applications using the OpenStack Compute Application Programming Interface (API).

Table of Contents

List of Figures

List of Tables

Preface

This document helps you deploy OpenStack Grizzly for development purposes with Ubuntu 12.04 LTS (using the Ubuntu Cloud Archive).

Note

The Grizzly version is available on the current Ubuntu development series, which is 13.04 (Raring Ringtail) and the most recent LTS (Long Term Support) version which is 12.04 (Precise Pangolin), via the Ubuntu Cloud Archive. At this time, there are not packages available for 12.10.

We are going to install a three-node setup with one controller, one network and one compute node.

Of course, you can setup as many computes nodes as you want. Use this document to learn to install a testing infrastructure.

The OpenStack configuration files contain several commented options. These options specify the default setting. You only need to uncomment these lines if you are changing the setting to a non-default value. Additionally, the only options that will be shown throughout this guide are options that are being modified from their default value.

Finally, please be aware that the use of `password` as a password throughout this guide is for simplicity and testing purposes. Please ensure you use proper passwords when configuring your OpenStack environment in production.

At this time, this guide does not cover floating IPs.

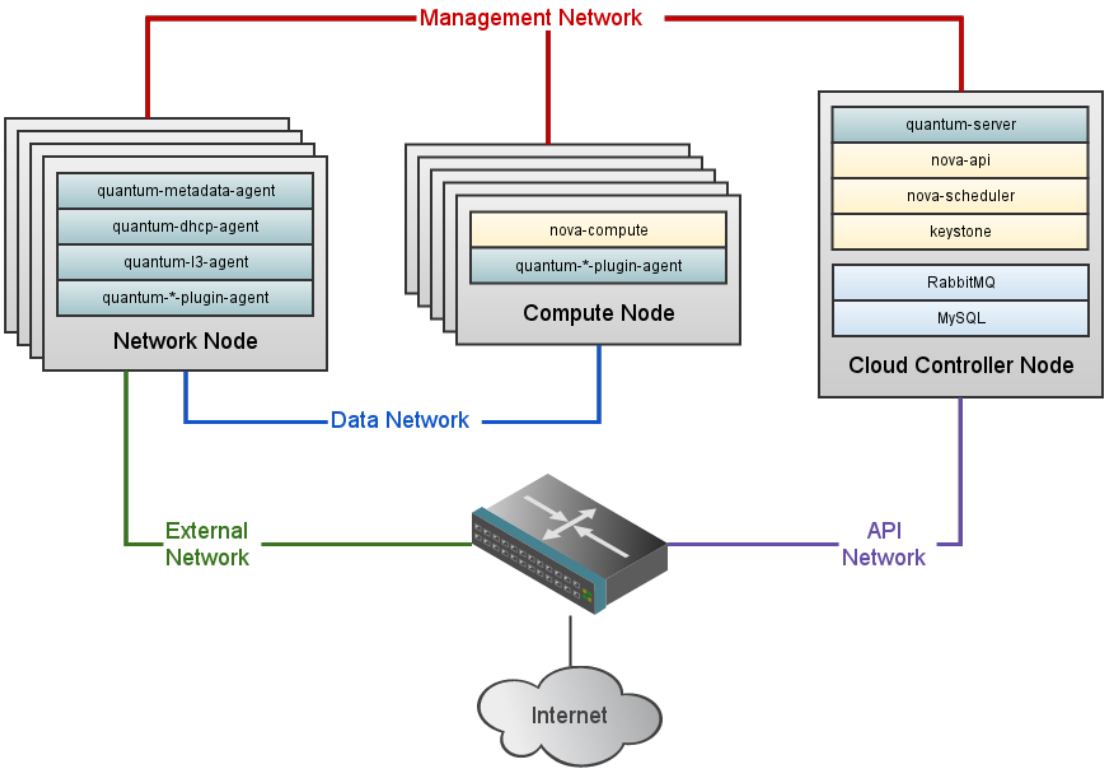
Document Change History

This version of the guide replaces and obsoletes all previous versions. The following table describes the latest changes: The most recent changes:

Revision Date	Summary of Changes
May 8, 2013	• Updated the title for consistency.
Jan 1, 2009	• First edition of this guide.

1. Architecture

Figure 1.1. Physical network diagram



This guide creates the following types of OpenStack servers:

- **The Cloud Controller**. Provides all functionality of the cloud except actually hosting virtual machines or providing network services. See the "Compute Node" and "Network Controller" for details about those roles. This server will host the OpenStack Image Service, the OpenStack Block Storage Service, the OpenStack Identity Service, and the OpenStack Dashboard. It will also run portions of the OpenStack Compute service such as the API server, the scheduler, conductor, console authenticator, and VNC service. Finally, it hosts the API endpoint for the OpenStack Network service.

- **The Network Controller**. Provides the bulk of the OpenStack Network services such as DHCP, layer 2 switching, layer 3 routing, floating IPs (which this guide does not configure), and metadata connectivity.

- **Compute Node**. Runs the OpenStack Compute service as well as the OpenStack Network service agent (in this case, the Open vSwitch plugin agent). This server also manages an OpenStack-compatible hypervisor such as KVM or Xen. This server will host the actual virtual machines (instances).

Note

OpenStack provides great flexibility with regard to how its individual services can be hosted. For example, the services that run on the Network Controller can easily be installed on the Cloud Controller. As another example, the OpenStack Image service can be installed on its own server (or many servers to provide a more highly available service).

With regard to cloud networking, a standard OpenStack Network setup can have up to four distinct physical data center networks. Note that these networks can be combined and re-used. For example, the Management, Data, and API networks are commonly the same network. For simplicity, this will be done in this guide.

- **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 OpenStack Networking plugin in use.
- **External network**. Provides 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 OpenStack Networking 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.

2. Install

Table of Contents

Requirements

You need at least three machines, virtual or physical, with Ubuntu 12.04 LTS installed.

Table 2.1. Architecture and node information

	Nodes		
	controller	**network**	**compute**
Hostname	cloud	network	c01
Services	MySQL, RabbitMQ , Nova, Cinder, Glance, Keystone, Quantum	Quantum-L3-agent, Quantum-DHCP-agent, Quantum Agent with Open-vSwitch	nova-compute, KVM, nova-api, Quantum Agent with Open-vSwitch
Minimum number of disks	2	1	1
External	10.0.0.10/24	10.0.0.9/24	-
Internal network	10.10.10.10/24	10.10.10.9/24	10.10.10.11/24
Total number of NIC	**2**	**2**	**1**

Controller Node

Introduction

The Controller node will provide :

- Databases (with MySQL)
- Queues (with RabbitMQ)
- Keystone
- Glance
- Nova (without nova-compute)
- Cinder
- Quantum Server (with Open-vSwitch plugin)
- Dashboard (with Horizon)

Common services

Operating System

1. Install Ubuntu 12.04 or 13.04. The exact installation procedure is outside the scope of this document, but please note the following configurations:

   - Time zone: **UTC**
   - Hostname: **cloud**
   - Packages: **OpenSSH-Server**, **wget**

   Once installation has finished, the server will reboot.

2. Since the default OpenStack release in Ubuntu 12.04 LTS is older, we are going to use the Ubuntu Cloud Archive for Grizzly:

   ```
   # apt-get install ubuntu-cloud-keyring
   ```

   Edit **/etc/apt/sources.list.d/cloud-archive.list**:

   ```
   deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/grizzly
    main
   ```

   Upgrade the system (and reboot if you need):

   ```
   # apt-get update && apt-get dist-upgrade
   ```

3. Configure the network:

   - Edit **/etc/network/interfaces**:

```
# Internal Network
auto eth0
    iface eth0 inet static
    address 10.10.10.10
    netmask 255.255.255.0

# External Network
    auto eth1
    iface eth1 inet static
    address 10.0.0.10
    netmask 255.255.255.0
    gateway 10.0.0.1
    dns-nameservers 8.8.8.8
```

- Edit **/etc/sysctl.conf**:

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Then, restart the network service:

```
# service networking restart
```

- Edit the **/etc/hosts** file and add **cloud**, **network**, and **c01** hostnames with correct IP.

```
127.0.0.1       localhost
10.10.10.10     cloud
10.10.10.9      network
10.10.10.11     c01
```

Note

While manually specifying host entries is acceptable for a simple or testing environment, it is highly recommended to use proper DNS entries, or at a minimum a configuration management system such as Puppet, to maintain your IP to host mappings.

4. Install NTP. NTP will ensure that the server has the correct time. This is important because if an OpenStack server's time is not correct, it will be removed from the rest of the cloud.

- `# apt-get install ntp`

MySQL Database Service

The various OpenStack components store persistent data in a relational database. MySQL is the most popular choice.

1. Install the packages:

```
# apt-get install python-mysqldb mysql-server
```

Note

`apt-get` will prompt you to set the MySQL root password.

2. By default, MySQL will only accept connections from localhost. This needs changed so that the compute nodes can access the OpenStack Networking service. Database requests for the OpenStack Compute service are proxied through the `nova-conductor` service.

```
# sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
```

3. Restart the service:

```
# service mysql restart
```

4. The various databases that the OpenStack services require need to be created. Additionally, MySQL accounts to access those databases need to be created:

```
# mysql -u root -p <<EOF
CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
IDENTIFIED BY 'password';
CREATE DATABASE cinder;
GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
IDENTIFIED BY 'password';
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
IDENTIFIED BY 'password';
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'password';
CREATE DATABASE quantum;
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'localhost' \
IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'10.10.10.9' \
IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON quantum.* TO 'quantum'@'10.10.10.11' \
IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
EOF
```

RabbitMQ Messaging Service

The OpenStack components also communicate through a queuing service. For example, the Cloud Controller places a request to launch an instance on the queue. The Compute Node then picks this request up and launches the instance. OpenStack can work with several different queuing services.

1. Install the packages:

```
# apt-get install rabbitmq-server
```

2. Change the default password:

```
# rabbitmqctl change_password guest password
```

Note

In addition to choosing another password in a production environment, you should also disable the guest account and use a proper RabbitMQ account. Please see the RabbitMQ documentation for further details.

OpenStack Identity Service

The OpenStack Identity Service provides the cloud environment with an authentication and authorization system. In this system, users are a part of one or more projects. In each of these projects, they hold a specific role.

1. Install the packages:

```
# apt-get install keystone python-keystone python-keystoneclient
```

2. Edit **/etc/keystone/keystone.conf**:

```
[DEFAULT]
admin_token = password
debug = True
verbose = True

[sql]
connection = mysql://keystone:password@localhost/keystone
```

3. Restart Keystone and create the tables in the database:

```
# service keystone restart
# keystone-manage db_sync
```

Note

Check the `/var/log/keystone/keystone.log` file for errors that would prevent the Identity Service from successfully starting.

4. Create an `openrc` File

   - Create a file called **~/openrc**. This file contains the OpenStack admin credentials that will be used when interacting with the OpenStack environment on the command line.

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=password
```

   - Source the credentials into your environment:

```
source ~/openrc
```

   - Configure the Bash shell to load these credentials upon each login:

```
echo "source ~/openrc" >> ~/.bashrc
```

5. The following bash script will populate Keystone with some initial data:

   - Projects: admin and services
   - Roles: admin, Member
   - Users: admin, demo, nova, glance, quantum, and cinder

- Services: compute, volume, image, identity, ec2, and network

```
#!/bin/bash

# Modify these variables as needed
ADMIN_PASSWORD=${ADMIN_PASSWORD:-password}
SERVICE_PASSWORD=${SERVICE_PASSWORD:-$ADMIN_PASSWORD}
DEMO_PASSWORD=${DEMO_PASSWORD:-$ADMIN_PASSWORD}
export OS_SERVICE_TOKEN="password"
export OS_SERVICE_ENDPOINT="http://localhost:35357/v2.0"
SERVICE_TENANT_NAME=${SERVICE_TENANT_NAME:-service}
#
MYSQL_USER=keystone
MYSQL_DATABASE=keystone
MYSQL_HOST=localhost
MYSQL_PASSWORD=password
#
KEYSTONE_REGION=RegionOne
KEYSTONE_HOST=10.10.10.10

# Shortcut function to get a newly generated ID
function get_field() {
    while read data; do
        if [ "$1" -lt 0 ]; then
            field="(\$(NF$1))"
        else
            field="\$$(($1 + 1))"
        fi
        echo "$data" | awk -F'[ \t]*\\|[ \t]*' "{print $field}"
    done
}

# Tenants
ADMIN_TENANT=$(keystone tenant-create --name=admin | grep " id " | get_field
 2)
DEMO_TENANT=$(keystone tenant-create --name=demo | grep " id " | get_field
 2)
SERVICE_TENANT=$(keystone tenant-create --name=$SERVICE_TENANT_NAME | grep "
 id " | get_field 2)

# Users
ADMIN_USER=$(keystone user-create --name=admin --pass="$ADMIN_PASSWORD" --
email=admin@domain.com | grep " id " | get_field 2)
DEMO_USER=$(keystone user-create --name=demo --pass="$DEMO_PASSWORD" --
email=demo@domain.com --tenant-id=$DEMO_TENANT | grep " id " | get_field 2)
NOVA_USER=$(keystone user-create --name=nova --pass="$SERVICE_PASSWORD" --
tenant-id $SERVICE_TENANT --email=nova@domain.com | grep " id " | get_field
 2)
GLANCE_USER=$(keystone user-create --name=glance --pass="$SERVICE_PASSWORD"
 --tenant-id $SERVICE_TENANT --email=glance@domain.com | grep " id " |
 get_field 2)
QUANTUM_USER=$(keystone user-create --name=quantum --pass=
"$SERVICE_PASSWORD" --tenant-id $SERVICE_TENANT --email=quantum@domain.com |
 grep " id " | get_field 2)
CINDER_USER=$(keystone user-create --name=cinder --pass="$SERVICE_PASSWORD"
 --tenant-id $SERVICE_TENANT --email=cinder@domain.com | grep " id " |
 get_field 2)

# Roles
```

```
ADMIN_ROLE=$(keystone role-create --name=admin | grep " id " | get_field 2)
MEMBER_ROLE=$(keystone role-create --name=Member | grep " id " | get_field
 2)

# Add Roles to Users in Tenants
keystone user-role-add --user-id $ADMIN_USER --role-id $ADMIN_ROLE --tenant-
id $ADMIN_TENANT
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $NOVA_USER --
role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $GLANCE_USER --
role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $QUANTUM_USER
 --role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $SERVICE_TENANT --user-id $CINDER_USER --
role-id $ADMIN_ROLE
keystone user-role-add --tenant-id $DEMO_TENANT --user-id $DEMO_USER --role-
id $MEMBER_ROLE

# Create services
COMPUTE_SERVICE=$(keystone service-create --name nova --type compute --
description 'OpenStack Compute Service' | grep " id " | get_field 2)
VOLUME_SERVICE=$(keystone service-create --name cinder --type volume --
description 'OpenStack Volume Service' | grep " id " | get_field 2)
IMAGE_SERVICE=$(keystone service-create --name glance --type image --
description 'OpenStack Image Service' | grep " id " | get_field 2)
IDENTITY_SERVICE=$(keystone service-create --name keystone --type identity
 --description 'OpenStack Identity' | grep " id " | get_field 2)
EC2_SERVICE=$(keystone service-create --name ec2 --type ec2 --description
 'OpenStack EC2 service' | grep " id " | get_field 2)
NETWORK_SERVICE=$(keystone service-create --name quantum --type network --
description 'OpenStack Networking service' | grep " id " | get_field 2)

# Create endpoints
keystone endpoint-create --region $KEYSTONE_REGION --service-id
 $COMPUTE_SERVICE --publicurl 'http://'"$KEYSTONE_HOST"':8774/v2/
$(tenant_id)s' --adminurl 'http://'"$KEYSTONE_HOST"':8774/v2/$(tenant_id)s'
 --internalurl 'http://'"$KEYSTONE_HOST"':8774/v2/$(tenant_id)s'
keystone endpoint-create --region $KEYSTONE_REGION --service-id
 $VOLUME_SERVICE --publicurl 'http://'"$KEYSTONE_HOST"':8776/v1/
$(tenant_id)s' --adminurl 'http://'"$KEYSTONE_HOST"':8776/v1/$(tenant_id)s'
 --internalurl 'http://'"$KEYSTONE_HOST"':8776/v1/$(tenant_id)s'
keystone endpoint-create --region $KEYSTONE_REGION --service-id
 $IMAGE_SERVICE --publicurl 'http://'"$KEYSTONE_HOST"':9292/v2' --
adminurl 'http://'"$KEYSTONE_HOST"':9292/v2' --internalurl 'http://
'"$KEYSTONE_HOST"':9292/v2'
keystone endpoint-create --region $KEYSTONE_REGION --service-id
 $IDENTITY_SERVICE --publicurl 'http://'"$KEYSTONE_HOST"':5000/v2.0' --
adminurl 'http://'"$KEYSTONE_HOST"':35357/v2.0' --internalurl 'http://
'"$KEYSTONE_HOST"':5000/v2.0'
keystone endpoint-create --region $KEYSTONE_REGION --service-id $EC2_SERVICE
 --publicurl 'http://'"$KEYSTONE_HOST"':8773/services/Cloud' --adminurl
 'http://'"$KEYSTONE_HOST"':8773/services/Admin' --internalurl 'http://
'"$KEYSTONE_HOST"':8773/services/Cloud'
keystone endpoint-create --region $KEYSTONE_REGION --service-id
 $NETWORK_SERVICE --publicurl 'http://'"$KEYSTONE_HOST"':9696/' --
adminurl 'http://'"$KEYSTONE_HOST"':9696/' --internalurl 'http://
'"$KEYSTONE_HOST"':9696/'
```

Note

If you make a mistake during this guide, you can reset the Keystone database by performing the following steps:

```
# mysql -u root -p keystone -e "drop database keystone"
# mysql -u root -p keystone -e "create database keystone"
# mysql -u root -p keystone -e "grant all privileges on keystone.*
 TO 'keystone'@'localhost' identified by 'password'"
# keystone-manage db_sync
```

And finally, re-run the above bash script.

OpenStack Image Service

The Image Service provides the cloud environment with a catalog of virtual machine "templates". These templates are used as the basis of instances. For example, if the catalog contains an image for an Ubuntu 12.04 distribution, the users of the cloud will be able to launch Ubuntu 12.04 instances.

1. Install the Glance packages:

```
# apt-get install glance glance-api glance-registry python-glanceclient
 glance-common
```

2. Configure Glance:

   - Glance consists of two services: `glance-api` and `glance-registry`. For a basic setup, they are configured identically, however, be aware that they provide two distinct services.

     Edit **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf**:

     ```
     [DEFAULT]
     sql_connection = mysql://glance:password@localhost/glance
     [keystone_authtoken]
     admin_tenant_name = service
     admin_user = glance
     admin_password = password
     ```

   - Restart both Glance services:

     ```
     # service glance-api restart && service glance-registry restart
     ```

     #### Note

     Check the `/var/log/glance/*.log` files for errors that would prevent the Image Service from successfully starting.

   - Create Glance tables into the database:

     ```
     # glance-manage db_sync
     ```

   - Download and import Ubuntu 12.04 LTS UEC Image:

     ```
     # wget http://uec-images.ubuntu.com/releases/12.04/release/ubuntu-12.04-
     server-cloudimg-amd64-disk1.img
     ```

```
# glance image-create --is-public true --disk-format qcow2 --container-
format bare --name "Ubuntu" < ubuntu-12.04-server-cloudimg-amd64-disk1.img
```

Download and import Cirros QCOW2 Image:

```
# wget http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
# glance image-create --is-public true --disk-format qcow2 --container-
format bare --name "Cirros 0.3.1" < cirros-0.3.1-x86_64-disk.img
```

- Check if the images have been introduced in the index:

```
# glance image-list

+--------------------------------------+---------------------------------
+-------------+-----------------+-----------+--------+
| ID                                   | Name                            |
 Disk Format | Container Format | Size      | Status |
+--------------------------------------+---------------------------------
+-------------+-----------------+-----------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | Cirros 0.3.1                    |
 qcow2       | bare             | 13147648  | active |
| 10ccdf86-e59e-41ac-ab41-65af91ea7a9c | cirros-0.3.0-x86_64-uec         |
 ami         | ami              | 25165824  | active |
| 8473f43f-cd1f-47cc-8d88-ccd9a62e566f | cirros-0.3.0-x86_64-uec-kernel  |
 aki         | aki              | 4731440   | active |
| 75c1bb27-a406-462c-a379-913e4e6221c9 | cirros-0.3.0-x86_64-uec-ramdisk |
 ari         | ari              | 2254249   | active |
| 62f9278e-a26e-4fa0-9537-1eb503aa2f01 | Ubuntu                          |
 qcow2       | bare             | 251985920 | active |
+--------------------------------------+---------------------------------
+-------------+-----------------+-----------+--------+
```

OpenStack Compute (Cloud Controller services)

The OpenStack Compute Service provides the cloud environment with the ability to manage the scheduling, creation and deletion of virtual machines (instances).

1. Install the Nova packages:

```
# apt-get install nova-api nova-cert nova-common nova-conductor \
    nova-scheduler python-nova python-novaclient nova-consoleauth novnc \
    nova-novncproxy
```

2. Configure Nova:

- Edit `/etc/nova/api-paste.ini`:

```
admin_tenant_name = service
admin_user = nova
admin_password = password
```

- Add the following to the **/etc/nova/nova.conf** file. This file is the main configuration file of Nova. There is a large amount of configuration options that can go in this file. This guide illustrates the minimum needed for a simple environment. Note that the nova.conf file supplied by your distribution will have some options already set. Leave them as-is.

```
[DEFAULT]
```

```
sql_connection=mysql://nova:password@localhost/nova
rabbit_password=password
auth_strategy=keystone

# Networking
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://10.10.10.10:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://10.10.10.10:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver

# Security Groups
firewall_driver=nova.virt.firewall.NoopFirewallDriver
security_group_api=quantum

# Metadata
quantum_metadata_proxy_shared_secret=password
service_quantum_metadata_proxy=true
metadata_listen = 10.10.10.10
metadata_listen_port = 8775

# Cinder
volume_api_class=nova.volume.cinder.API

# Glance
glance_api_servers=10.10.10.10:9292
image_service=nova.image.glance.GlanceImageService

# novnc
novnc_enable=true
novncproxy_port=6080
novncproxy_host=10.0.0.10
vncserver_listen=0.0.0.0
```

- Create Nova tables into the database:

```
# nova-manage db sync
```

- Restart Nova services:

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

Note

Check the `/var/log/nova/nova-*` files for any errors that would prevent the Compute Service from successfully starting.

OpenStack Block Storage

While Cinder contains many different storage drivers, the most common and basic configuration uses LVM and iSCSI. This guide illustrates how to use one disk (`/dev/sdb`) in an LVM Volume Group called `cinder-volumes`. When a user requests a block storage volume, a Logical Volume is created from this Volume Group and then mounted on the user's instance by way of iSCSI.

1. Install the Cinder packages:

```
# apt-get install cinder-api cinder-scheduler cinder-volume iscsitarget \
    open-iscsi iscsitarget-dkms python-cinderclient linux-headers-`uname -r`
```

2. Configure & start the iSCSI services:

```
# sed -i 's/false/true/g' /etc/default/iscsitarget
# service iscsitarget start
# service open-iscsi start
```

3. Configure Cinder:

   - Edit `/etc/cinder/cinder.conf`:

```
[DEFAULT]
sql_connection = mysql://cinder:password@localhost/cinder
rabbit_password = password

[keystone_authtoken]
admin_tenant_name = service
admin_user = cinder
admin_password = password
service_protocol = http
service_host = localhost
service_port = 5000
auth_host = localhost
auth_port = 35357
auth_protocol = http
```

   - Edit `/etc/cinder/api-paste.ini` in the `filter_authtoken` section so it does not contain any of the variables defined in the `keystone_authtoken` section of `cinder.conf`:

```
[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
# This section should not contain any other config options
```

   - Create the LVM Physical Volume and Logical Volume:

```
# pvcreate /dev/sdb
# vgcreate cinder-volumes /dev/sdb
```

   - Create Cinder tables into the database:

```
# cinder-manage db sync
```

   - Restart the services:

```
# service cinder-api restart
# service cinder-scheduler restart
```

```
# service cinder-volume restart
```

OpenStack Network Service (Cloud Controller)

The OpenStack Network Service provides a comprehensive and extendible networking service to the cloud. Some features include, but are not limited to, the ability for instances to reach an external network outside of the cloud as well as the ability for each user of the cloud to create multiple internal subnets of their own.

1. Install the Quantum Server:

```
# apt-get install quantum-server
```

2. Configure the Quantum service:

   - Edit **/etc/quantum/quantum.conf**:

```
[DEFAULT]
verbose = True
rabbit_password = password
[keystone_authtoken]
admin_tenant_name = service
admin_user = quantum
admin_password = password
```

   - Edit **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini**:

```
[DATABASE]
sql_connection = mysql://quantum:password@localhost/quantum
[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
local_ip = 10.10.10.10
[SECURITYGROUP]
firewall_driver = quantum.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver
```

Note

It's more handy to choose **tunnel mode** since you don't have to configure your physical switches for VLANs.

3. Enable the OVS plugin:

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

4. Start the services:

```
# service quantum-server restart
```

OpenStack Dashboard

The OpenStack Dashboard service provides users of the cloud environment with a web-accessible GUI as an alternative to using the command-line tools.

To enable it, install the Horizon package and its dependencies:

```
# apt-get install openstack-dashboard memcached python-memcache
```

Note

Optional, but recommended: remove the `openstack-dashboard-ubuntu-theme` package. This theme prevents several menus as well as the network map from rendering correctly:

```
apt-get remove --purge openstack-dashboard-ubuntu-theme
```

OpenStack Dashboard is now available at **http://cloud/horizon**. We can login with the **admin** / **password** credentials or **demo** / **password**.

Note

Check the `/var/log/apache/error.log` file for errors that wold prevent either the Apache service or the Dashboard service from successfully starting.

Network Node

Introduction

The Network node will provide:

- Virtual Bridging (Open-vSwitch + Quantum Agent) with tunneling
- DHCP Server (Quantum DHCP Agent)
- Virtual Routing (Quantum L3 Agent)

Note

It is entirely possible to install all of these services on the Cloud Controller. If you are short of resources, this is a good alternative.

Common services

Operating System

1. Install Ubuntu 12.04. The exact installation procedure is outside the scope of this document, but please note the following configurations:

   - Time zone: **UTC**
   - Hostname: **network**
   - Packages: **OpenSSH-Server**

   Once installation has finished, the server will reboot.

2. Since the default OpenStack release in Ubuntu 12.04 LTS is older, we are going to use the Ubuntu Cloud Archive for Grizzly:

   ```
   # apt-get install ubuntu-cloud-keyring
   ```

Edit **/etc/apt/sources.list.d/cloud-archive.list**:

```
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/grizzly
 main
```

Upgrade the system (and reboot if needed):

```
# apt-get update && apt-get dist-upgrade
```

3. Configure the network:

- Edit **/etc/network/interfaces**:

Note

This will change later on in the guide when Open vSwitch is configured.

```
# Internal Network
auto eth0
iface eth0 inet static
    address 10.10.10.9
    netmask 255.255.255.0

# External Network
auto eth1
iface eth1 inet static
    address 10.0.0.9
    netmask 255.255.255.0
    gateway 10.0.0.1
    dns-nameservers 8.8.8.8
```

- Edit **/etc/sysctl.conf**:

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Then, restart the network service:

```
service networking restart
```

- Edit the **/etc/hosts** file and add **controller**, **networknode** and **compute1** hostnames with correct IP.

```
127.0.0.1       localhost
10.10.10.10     cloud
10.10.10.9      network
10.10.10.11     c01
```

4. Install NTP:

- `# apt-get install ntp`

OpenStack Networking (Network Controller)

Open vSwitch

1. Install the packages:

```
# apt-get install quantum-plugin-openvswitch-agent \
quantum-dhcp-agent quantum-l3-agent
```

2. Start Open vSwitch:

```
# service openvswitch-switch start
```

3. Create an internal and external network bridge. The purposes of these bridges are described in the Introduction of this guide.

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

4. Configure the bridges:

   - Change the `eth1` entry in **/etc/network/interfaces** to look like:

```
auto eth1
iface eth1 inet manual
    up ip address add 0/0 dev $IFACE
    up ip link set $IFACE up
    down ip link set $IFACE down
```

   - Add `br-ex` to **/etc/network/interfaces**:

```
auto br-ex
iface br-ex inet static
    address 10.0.0.9
    netmask 255.255.255.0
    gateway 10.0.0.1
```

   - Remove the IP address from `eth1` add it to `br-ex`:

```
# ip addr del 10.0.0.9/24 dev eth1
# ip addr add 10.0.0.9/24 dev br-ex
```

   - Restart networking:

```
# service networking restart
```

5. Finally, enable a simple NAT service so that the Compute Node(s) can access the Internet through the Cloud Controller:

```
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
iptables -A FORWARD -i eth0 -o br-ex -s 10.10.10.0/24 -m conntrack --ctstate
 NEW -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A POSTROUTING -s 10.10.10.0/24 -t nat -j MASQUERADE
```

Quantum

Configure the Quantum services:

- Edit **/etc/quantum/quantum.conf**:

```
[DEFAULT]
verbose = True
rabbit_password = password
rabbit_host = 10.10.10.10
[keystone_authtoken]
auth_host = 10.10.10.10
admin_tenant_name = service
admin_user = quantum
admin_password = password
```

- Edit **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini**:

```
[DATABASE]
sql_connection = mysql://quantum:password@10.10.10.10/quantum
[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
local_ip = 10.10.10.9
[SECURITYGROUP]

firewall_driver = quantum.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver
```

Note

It's more handy to choose **tunnel mode** since you don't have to configure your physical switches for VLANs.

- Edit **/etc/quantum/dhcp_agent.ini**:

```
[DEFAULT]
enable_isolated_metadata = True
enable_metadata_network = True
```

- Edit **/etc/quantum/metadata_agent.ini**:

```
[DEFAULT]
auth_url = http://10.10.10.10:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = quantum
admin_password = password
nova_metadata_ip = 10.10.10.10
metadata_proxy_shared_secret = password
```

Start the services:

```
# service quantum-plugin-openvswitch-agent start
# service quantum-dhcp-agent restart
# service quantum-metadata-agent restart
# service quantum-l3-agent restart
```

Note

Check the `/var/log/quantum/*.log` files for errors that would prevent the Networking Service from successfully starting.

Virtual Networking

Create Virtual Networking

1. Create an `openrc` File

   - Create a file called **~/openrc**. This file contains the OpenStack admin credentials that will be used when interacting with the OpenStack environment on the command line.

     ```
     export OS_TENANT_NAME=admin
     export OS_USERNAME=admin
     export OS_PASSWORD=password
     export OS_AUTH_URL="http://10.10.10.10:5000/v2.0/"
     export SERVICE_ENDPOINT="http://10.10.10.10:35357/v2.0"
     export SERVICE_TOKEN=password
     ```

   - Source the credentials into your environment:

     ```
     source ~/openrc
     ```

   - Configure the Bash shell to load these credentials upon each login:

     ```
     echo "source ~/openrc" >> ~/.bashrc
     ```

2. The following bash script will create an internal network for the "demo" project.

```
#!/bin/bash
TENANT_NAME="demo"
TENANT_NETWORK_NAME="demo-net"
TENANT_SUBNET_NAME="${TENANT_NETWORK_NAME}-subnet"
TENANT_ROUTER_NAME="demo-router"
FIXED_RANGE="10.5.5.0/24"
NETWORK_GATEWAY="10.5.5.1"
TENANT_ID=$(keystone tenant-list | grep " $TENANT_NAME " | awk '{print $2}')

TENANT_NET_ID=$(quantum net-create --tenant_id $TENANT_ID
 $TENANT_NETWORK_NAME --provider:network_type gre  --
provider:segmentation_id 1 | grep " id " | awk '{print $4}')
TENANT_SUBNET_ID=$(quantum subnet-create --tenant_id $TENANT_ID --ip_version
 4 --name $TENANT_SUBNET_NAME $TENANT_NET_ID $FIXED_RANGE --gateway
 $NETWORK_GATEWAY --dns_nameservers list=true 8.8.8.8 | grep " id " | awk
 '{print $4}')
ROUTER_ID=$(quantum router-create --tenant_id $TENANT_ID $TENANT_ROUTER_NAME
 | grep " id " | awk '{print $4}')
quantum router-interface-add $ROUTER_ID $TENANT_SUBNET_ID
```

L3 Configuration

The Quantum L3 service enables instances to have external network access. If this service is not configured, your instances will only be able to communicate with each other. Please note that this configuration is highly dependant on your environment. For example, make note of the `subnet-create` command below. You will need to verify your own network settings for the external subnet (`10.0.0.0/24` in this case) as well as an allocation pool. The allocation pool is used to provide each Project with an IP address to access the external network. The pool consists of 50 IPs and therefore only 50 projects will be able to get a gateway IP.

- Create an external network:

```
quantum net-create public --router:external=True
```

- Create a subnet for the external network:

```
quantum subnet-create --ip_version 4 --gateway 10.0.0.1 public 10.0.0.0/
24 --allocation-pool start=10.0.0.200,end=10.0.0.250 --disable-dhcp --name
 public-subnet
```

- Set the gateway of the demo router to the public network:

```
quantum router-gateway-set demo-router public
```

Compute Node

Introduction

The Compute node will provide :

- Hypervisor (KVM)
- nova-compute
- Quantum OVS Agent

Common services

Operating System

1. Install Ubuntu 12.04. Just like with the Cloud Controller, the exact steps are outside the scope of this document, but please note the following options:

   - Time zone: **UTC**
   - Hostname: **c01**
   - Packages: **OpenSSH-Server**

   Once installation has finished, the server will reboot.

2. Since the default OpenStack release in Ubuntu 12.04 LTS is older, we are going to use the Ubuntu Cloud Archive for Grizzly:

   ```
   # apt-get install ubuntu-cloud-keyring
   ```

   Edit **/etc/apt/sources.list.d/cloud-archive.list**:

   ```
   deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/grizzly
    main
   ```

   Upgrade the system (and reboot if you need):

   ```
   # sudo apt-get update && apt-get upgrade
   ```

3. Configure the network:

Note

This will change later on in the guide when Open vSwitch is configured

- Edit **/etc/network/interfaces**:

```
# Internal Network
auto eth0
    iface eth0 inet static
    address 10.10.10.11
    netmask 255.255.255.0
    gateway 10.10.10.9
    dns-nameservers 8.8.8.8
```

- Edit **/etc/sysctl.conf**:

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Then, restart the network service:

```
service networking restart
```

- Edit the **/etc/hosts** file and add **cloud**, **network** and **c01** hostnames with correct IP.

```
127.0.0.1       localhost
10.10.10.10     cloud
10.10.10.9      network
10.10.10.11     c01
```

4. Install NTP:

- `# apt-get install ntp`

OpenStack Compute (Compute Node services)

Just like with the Cloud Controller, the OpenStack Compute service is installed on the Compute Node. However, this time the `nova-compute` service is installed. This provides the Compute Node the capability to host virtual machines.

1. Install the Nova Compute package:

```
# apt-get install nova-compute-kvm
```

Note

`nova-compute-kvm` requires that your CPU supports hardware-assisted virtualization (HVM) such as Intel VT-x or AMD-V. If your CPU does not support this, or if you are already running in a virtualized environment, you can instead use the `nova-compute-qemu` package. This package provides software-based virtualization.

2. Configure Nova:

- Edit **/etc/nova/api-paste.ini**:

```
[filter:authtoken]
auth_host = 10.10.10.10
admin_tenant_name = service
admin_user = nova
admin_password = password
```

- Edit **/etc/nova/nova.conf**:

```
[DEFAULT]

# General
verbose=True
rabbit_host=10.10.10.10
rabbit_password=password

auth_strategy=keystone
ec2_host=10.10.10.10
ec2_url=http://10.10.10.10:8773/services/Cloud

# Networking
libvirt_use_virtio_for_bridges=True
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://10.10.10.10:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://10.10.10.10:35357/v2.0

# Security Groups
firewall_driver=nova.virt.firewall.NoopFirewallDriver
security_group_api=quantum

# Compute #
compute_driver=libvirt.LibvirtDriver
connection_type=libvirt

# Cinder
volume_api_class=nova.volume.cinder.API

# Glance
glance_api_servers=10.10.10.10:9292
image_service=nova.image.glance.GlanceImageService

# novnc
vnc_enabled=true
vncserver_proxyclient_address=10.10.10.11
novncproxy_base_url=http://10.0.0.10:6080/vnc_auto.html
vncserver_listen=0.0.0.0
```

- Restart Nova services:

```
# service nova-compute restart
```

OpenStack Networking (Compute Node)

Open vSwitch

1. Install the packages:

```
# apt-get install openvswitch-switch
```

2. Start Open vSwitch service

```
# service openvswitch-switch start
```

3. Create an internal bridge. Just as described in the Introduction to this guide, the Compute Node does not provide an external bridge. This enforces all instances' network traffic to go through the Network Controller. This is known as a "single-node" networking setup.

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

Quantum

1. Install the packages:

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

2. Edit **/etc/quantum/quantum.conf**:

```
[DEFAULT]
rabbit_host = 10.10.10.10
rabbit_password = password
verbose = True
```

3. Edit **/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini**:

```
[DATABASE]
sql_connection = mysql://quantum:password@10.10.10.1/quantum
[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
local_ip = 10.10.10.11
enable_tunneling = True
[SECURITYGROUP]
firewall_driver = quantum.agent.linux.iptables_firewall.
OVSHybridIptablesFirewallDriver
```

4. Start the Agent:

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

5. Ensure the cleanup utility is started on future boots:

Note

Check the `/var/log/quantum/openvswitch-agent.log` file for errors that would prevent the Networking service from successfully starting.

3. Create Your First VM

You can use OpenStack API or the Dashboard to manage your own IaaS: go to http://10.0.0.10/horizon with demo / password credentials.

In the Dashboard:

1. Edit the security group "Default" to allow ICMP and SSH.
2. Create a personal keypair.
3. Go to "Instances" and click "Launch Instance" for spawning a new VM.

Alternatively, you can do this all on the command line at the Cloud Controller:

1. Copy the `~/openrc` file as `~/demorc` and edit the contents to suit the demo user.
2. Create an SSH keypair and add it to Nova:

```
# ssh-keygen -f ~/.ssh/id_rsa -t rsa -N ''
# nova keypair-add --pub_key ~/.ssh/id_rsa.pub default_key
```

3. Edit the "default" Security Group to allow SSH and ICMP:

```
# nova secgroup-add-rule default tcp 22 22 0.0.0.0/24
# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/24
```

4. Query the Image Service and note the ID of the image you want to launch:

```
# nova image-list
```

5. Launch an instance:

```
# nova boot --flavor 1 --image <image_id> --key-name default_key my_instance
```

6. Wait a few seconds and check the status of your instance:

```
# nova show my_instance
```

7. If the status is in Error state, check the `nova-scheduler` log:

```
# tail /var/log/nova/nova-scheduler.log
```

8. If the status is in Available state, check the instance's console for booting status:

```
# nova console-log my_instance
```

After your instance has successfully booted and the console log is showing a log in prompt, you may now SSH into your instance. In order to do this, though, you have to work with the Linux network namespaces on the Network Controller:

1. Print a list of all namespaces:

```
# ip netns
```

   The output should contain two lines: one beginning with `qrouter` and the other beginning with `qdhcp`

2. Run SSH inside the qdhcp namespace:

```
# ip netns exec qdhcp-c73d082f-d7ed-4b53-ac93-7a6a4c3fa3aa ssh 10.5.5.2
```

4. Conclusion

We have built a basic architecture for advanced testing purpose. This kind of architecture is close to the production, without High Availability (HA) and some services such as those for running OpenStack Object Storage. You can of course add as many compute nodes as you want. If you need more specific help, please read the official documentation of each project or write a post to an OpenStack mailing list.