Deploying OpenStack (for Ubuntu Server 14.04 LTS)

This documentation is aimed at System Administrators for deploying OpenStack on Ubuntu 14.04 LTS. This can be used as an ideal setup for a small office environment, or if properly scaled up can serve as a backbone for an enterprise environment.

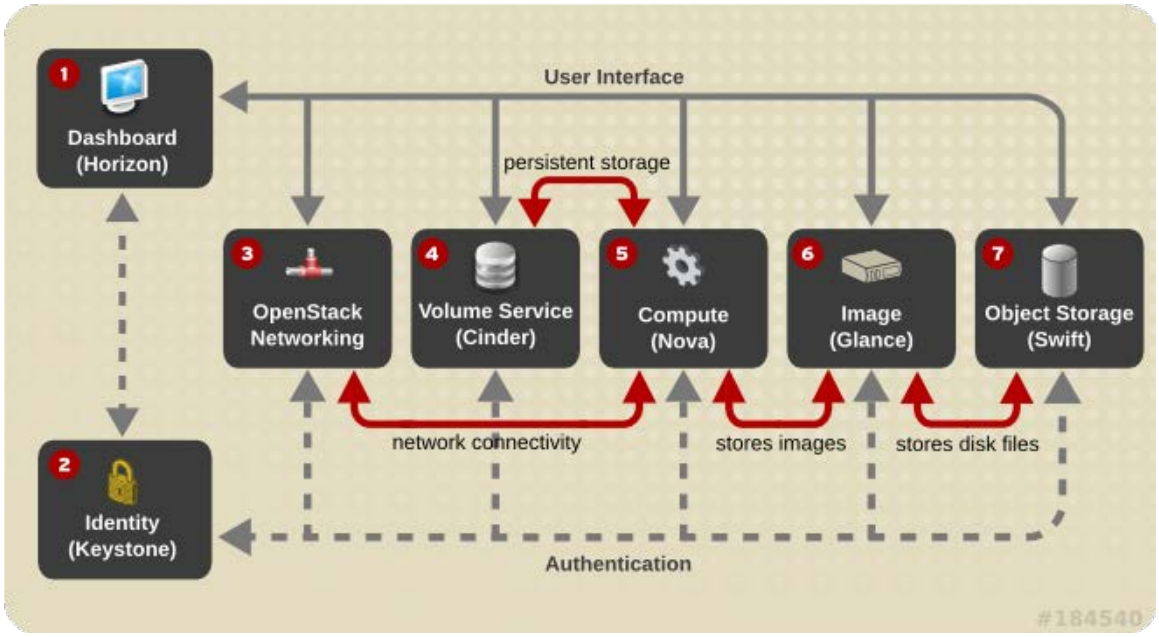
The contents of documentation are as follows.

Introduction to OpenStack

OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service (IaaS). The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center. One can either manage it through a web-based dashboard, through command-line tools, or through RESTful web services.

OpenStack has a modular architecture with various code names for its components.

- **Identity (Keystone)**
- **Image (Glance)**
- **Compute (Nova)**
- **Networking (Neutron)**
- **Dashboard (Horizon)**
- **Storage (Cinder/Swift)**



OpenStack architecture diagram

◘ Identity (Keystone)

OpenStack Identity service provides authentication and authorization service for other OpenStack services. It provides a catalog of endpoints for all OpenStack services.

Keystone is the identity service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API. It supports LDAP, OAuth, OpenID Connect, SAML and SQL.

◘ Image (Glance)

OpenStack Image service provides ability for users to upload and discover data assets that are meant to be used with other services. This currently includes images and metadata definitions. Thus it allows storing and retrieving virtual machine disk images. Compute service makes use of this during instance provisioning.

Glance is the image service that provide ability to discover, register, and retrieve virtual machine (VM) images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image. VM images made available through Glance can be stored in a variety of locations from simple filesystems to object-storage systems like the OpenStack Swift project.

◘ Compute (Nova)

OpenStack Compute service manages the lifecycle of compute instances in an OpenStack environment. The responsibilities include spawning, scheduling and decommissioning of virtual machines on demand. Thus it implements services and associated libraries to provide massively scalable, on demand, self service access to compute resources, including bare metal, virtual machines, and containers.

Nova is the OpenStack project that provides this functionality. Nova supports creating virtual machines, baremetal servers (through the use of ironic), and has limited support for system containers. Nova runs as a set of daemons on top of existing Linux servers to provide that service. It can also integrate with other services to include: persistent block storage, encrypted disks, and baremetal compute instances. Nova requires the additional OpenStack services Keystone, Glance and Neutron for basic functioning.

◘ Networking (Neutron)

OpenStack Networking service is a SDN networking service that delivers networking-as-a-service (NaaS) in virtual compute environments.

Neutron is the OpenStack project that provides this functionality. It implements the Neutron API and provides the API for users to define networks and the attachments into them. It has a pluggable architecture that supports many popular networking vendors and technologies.

◘ Dashboard (Horizon)

OpenStack Dashboard provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls. Horizon is the canonical implementation of OpenStack's Dashboard.

◘ Storage

Block Storage (Cinder)

OpenStack Block Storage service provides persistent block storage to running instances. It's pluggable driver architecture facilitates the creation and management of block storage devices.

Cinder is the Block Storage service for OpenStack. It's designed to present storage resources to end users that can be consumed by the OpenStack Compute Project (Nova). This is done through the use of either a reference implementation (LVM) or plugin drivers for other storage. The short description of Cinder is that it virtualizes the management of block storage devices and provides end users with a self service API to request and consume those resources without requiring any knowledge of where their storage is actually deployed or on what type of device.

Object Storage (Swift)

OpenStack Object Storage service stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API.

Swift is the OpenStack Object storage project. It's built for scale and optimized for durability, availability, and concurrency across the entire data set. Swift is ideal for storing unstructured data that can grow without bound. It is highly fault tolerant with its data replication and scale-out architecture. It's implementation is not like a file server with mountable directories. It writes objects and files to multiple drives, ensuring the data is replicated across a server cluster.

OS Installation

We will be using Ubuntu 14.04.5 LTS (Trusty Tahr) 64-bit Server Edition for our nodes.
This edition of Ubuntu was released on ***2016-Aug-04*** and is supported until ***2019-Apr***.

Download the corresponding ISO from http://releases.ubuntu.com/trusty/.
We have downloaded the ISO file ***ubuntu-14.04.5-server-amd64.iso***.

We are using a 64-bit version of the distribution, because it is recommended to have a 64-bit OS on at least the compute node. Otherwise if we install a 32-bit version of a distribution on the compute node, attempting to start an instance using a 64-bit image will fail.

Prepare a bootable media using the ISO.
If optical media is being used, prepare the bootable media using appropriate application.
But if we are using HDD or SSD, bootable media may be created using following method.
Open a terminal in the directory containing ISO file. Execute the command with following syntax.

SYNTAX:

```
# dd if= ubuntu-14.04.5-server-amd64.iso of=/dev/sdaX
```

Replace **X** with the drive letter of HDD or SSD device

Our OpenStack setup consists of one Control node, Compute node and Cinder based Storage node.

OpenStack Control node	⟹	10.3.0.1	[***horizon-network.office.brookwin.com***]
OpenStack Compute node	⟹	10.3.0.2	[***nova.office.brookwin.com***]
OpenStack Storage node	⟹	10.3.0.3	[***cinder.office.brookwin.com***]

Let us start with installing Ubuntu on each nodes. Boot the nodes with the bootable media we just created.

1) Upon initial booting we will presented a choice to select the Language, with ***English*** highlighted as default.

Make sure ***English*** is selected and press Enter.

2) Next we will see the installation splash screen with Boot menu. We can initiate the installation of OS from here.

Select ***Install Ubuntu Server*** and press Enter.

3) Next will be the Language selection screen with ***English*** highlighted as the default. This language will be used for the installation process as well as the default language for installed system.

Make sure ***English*** is selected and press Enter.

4) Next will be the Location selection screen. Here we can select the location for our server
Choose ***India*** and press Enter.

5) On next screen, the installer will ask us whether we want it to detect our keyboard layout or we want to choose from a list of available options. Let us manually configure the keyboard.
Select ***No*** and press Enter.

6) On the next screen, the installer will list the country of origin of our keyboard.
Select ***English(US)*** and press Enter.

7) This will be followed by a screen that lists countries that follows different keyboard layouts. Our keyboard layout language is English and is followed by US.
Select ***English (US)*** and press Enter.

This will be followed by detection of hardware, after which installer will load installation components from media.

The installer will further detect network hardware, after which we will be guided through the Network Configuration setup.

8) If there is more than one network interface(*which we surely has for all nodes*), we will be given an option to select the primary network interface.
We may choose an option of our choice and press Enter.

9) If our network has DHCP configured, it will detect and automatically configure the network.
Our network has DHCP server available. So the network will be automatically configured.

10) On the next screen, we will be able to specify our server's hostname. Specify the hostname appropriately for each node.

NOTE:

OpenStack Control node	⟹	10.3.0.1 ***[horizon-network.office.brookwin.com]***
OpenStack Compute node	⟹	10.3.0.2 ***[nova.office.brookwin.com]***
OpenStack Storage node	⟹	10.3.0.3 ***[cinder.office.brookwin.com]***

After specifying the hostname, select ***Continue*** and press Enter.

After the Network is configured, there will be steps to setup Users and Passwords.

11) We will be presented with a screen where we can specify the user's real name.

 Specify our user's real name as **user1**

Select ***Continue*** and press Enter.

12) On next screen, we can specify a username for our user. The username should start with a lower case letter, which can be followed by any combination of numbers and more lower case letters.

 Specify the username as **user1**

Select ***Continue*** and press Enter.

13) On next screen, we can specify a password for our user.

 Set the password as **101010**

Select ***Continue*** and press Enter.

14) This will be followed by a password verification screen.

 Set the password as **101010**

Select ***Continue*** and press Enter.

15) Since we have used a weak password, the installer will prompt us if we should proceed with this password. We are doing this for testing purposes and is fine to do so.

Select ***Yes*** and press Enter.

16) Next, the installer will ask us if we want to encrypt the home directory of our user. We will not be needing this here.

Select ***No*** and press Enter.

After Users and Passwords are setup, the installer will proceed with setting up Time, Date and Timezone.

17) First installer will go for retrieving Time and Date. If there is no internet connection, time will be retrieved from the CMOS in motherboard. But if we are connected to internet, installer will get time from the Network Time Server.

Then based on our present physical location, installer will prompt us to confirm a specific Timezone.

This has to be be ***Asia/Calcutta*** for us.

Select ***Yes*** and press Enter. The installer will proceed with setting up the retrieved Time and date for our server.

After this comes the important step ➠ Disk Partitioning.

18) We will be presented with a list of choices for Partitioning.

Select the Partitioning method as ***Manual*** and press Enter.

19) Let us setup a new partition for installing the operating system.

Create a new partition.

➜ Select Create a new partition

➜ Select new partition size as ***20 GB***

➜ Select new partition type as ***Primary***

➜ Select Location for the new partition as ***Beginning***

We will be presented with Partition settings for new partition. Configure the options as following.

➜ Set *Use as* to ***XFS journaling file system***

➜ Set *Mount point* to ***/***

➜ If there is such an option *Format the partition,* set it to ***yes, format it***

➜ Set *Bootable flag* to ***on***

Select ***Done setting up the partition***.

20) Next setup a partition for use as swap space.

Create a new partition.

➜ Select Create a new partition

➜ Select new partition size for swap space. Use the Redhat official documentation for determining swap space.

➜ Select new partition type as ***Primary***

➜ Select Location for the new partition as ***Beginning***

We will be presented with Partition settings for new partition. Configure the options as following.

➜ Set *Use as* to ***swap area***

Select ***Done setting up the partition***

21) After we are done, take an overview of the partitioning setup.

If everything is done accordingly, select ***Finish partitioning and write changes to disk***.

22) The installer will prompt us one more time if it should write the changes to disk.

Select ***Yes*** and press Enter.

The installer will now proceed with installing the base system. This process may take a while.
After the base system is installed, we will be guided to the Software selection and installation.

23) The first screen will be where the installer asks us whether we want to configure a HTTP proxy. We do not plan to do this.
Leave the field blank. Select ***Continue*** and and press Enter.

24) On next screen, we can select how to manage upgrades on the installed system.
Select ***No automatic updates***.

25) The next screen is for Software Selection. Depending on the purpose for which this server will be used we can select different software groups. We must select the following softwares to install.

- ***Virtual Machine Host***
- ***OpenSSH server***

Select ***Continue*** and press Enter.

Depending on the Software groups we have chosen, this may take some time.

After the installation of software is complete, we will be taken to the Bootloader installation screen.

26) The installer will prompt us whether to install GRUB boot loader to the MBR or not.
Select ***Yes*** and press Enter.

27) On the next screen, installer will allow us to select the device for boot loader installation
Select the appropriate device and press Enter.

Finally the installer will show us the Installation complete screen.

Remove the installation media and press ***Continue***.

Our Ubuntu Server operating system should boot into the login prompt.

Basic OS Setup

Let us configure our installed systems.

1) Setup root user

The root user is disabled in a typical Ubuntu installation. Let us enable the root user.

Perform the following step in all nodes.

Set a password for root user. This will also enable the root user.

```
#  passwd root
```

IMPORTANT: Set the password as 101010

2) Configure Text Editor

We need a versatile and comfortable text editor to work in a command line environment.

Let us use ***vi*** or it's clone ***vim*** for this purpose.

Perform the following step in all nodes.

Enable line number in ***vi*** instances for root user.

```
#  echo 'set number' > /root/.vimrc
```

3) Configure networking

Now let us configure networking in all our nodes.
Our OpenStack setup consists of following nodes as shown below.
OpenStack Control node ⟹ 10.3.0.1 [***horizon-network.office.brookwin.com***]

OpenStack Compute node ⟹ 10.3.0.2 [***nova.office.brookwin.com***]
OpenStack Storage node ⟹ 10.3.0.3 [***cinder.office.brookwin.com***]

Perform the following changes in OpenStack Control node.

In interface configuration file ***/etc/network/interfaces***, make the following changes.

```
#auto enp3s0
#iface enp3s0 inet dhcp

auto enp3s0
iface enp3s0 inet static
      address 10.3.0.1
      netmask 255.0.0.0
      network 10.0.0.0
      broadcast 10.255.255.255
      gateway 10.1.0.2
      dns-nameservers 10.1.0.10
      dns-search office.brookwin.com
```

In host configuration file ***/etc/hosts***, make the following changes.

```
#127.0.1.1          horizon-network.office.brookwin.com

10.3.0.1            horizon-network.office.brookwin.com      horizon-network

10.3.0.2            nova.office.brookwin.com                 nova
10.3.0.3            cinder.office.brookwin.com               cinder
```

Reboot the system.

Perform the following changes in OpenStack Compute node.

In interface configuration file ***/etc/network/interfaces***, make the following changes.

```
#auto enp2s0
#iface enp2s0 inet dhcp

auto enp2s0
iface enp2s0 inet static
      address 10.3.0.2
      netmask 255.0.0.0
      network 10.0.0.0
      broadcast 10.255.255.255
      gateway 10.1.0.2
      dns-nameservers 10.1.0.10
      dns-search office.brookwin.com
```

In host configuration file ***/etc/hosts***, make the following changes.

```
#127.0.1.1      nova.office.brookwin.com

10.3.0.2                nova.office.brookwin.com                nova

10.3.0.1                horizon-network.office.brookwin.com     horizon-network
10.3.0.3                cinder.office.brookwin.com              cinder
```

Reboot the system.

Perform the following changes in OpenStack Storage node.

In interface configuration file ***/etc/network/interfaces***, make the following changes.

```
#auto enp3s0
#iface enp3so inet dhcp

auto enp3s0
iface enp3s0 inet static
      address 10.3.0.3
      netmask 255.0.0.0
      network 10.0.0.0
      broadcast 10.255.255.255
      gateway 10.1.0.2
      dns-nameservers 10.1.0.10
      dns-search office.brookwin.com
```

In host configuration file ***/etc/hosts***, make the following changes.

```
#127.0.1.1              cinder.office.brookwin.com

10.3.0.3                cinder.office.brookwin.com              cinder

10.3.0.1                horizon-network.office.brookwin.com     horizon-network
10.3.0.2                nova.office.brookwin.com                nova
```

Reboot the system.

4) Configure SSH server

All the nodes in our cluster requires to be able to perform passwordless SSH login to each other as root user. To enable this, perform the following changes in all nodes.

In OpenSSH Server configuration file ***/etc/ssh/sshd_config***, make the following changes.

Change the logging level from ***INFO*** to ***VERBOSE***.

```
#LogLevel INFO
LogLevel VERBOSE
```

Allow root login using password.
```
#PermitRootLogin prohibit-password
PermitRootLogin yes
```

Restart ***ssh*** service.
```
# service ssh restart
```

5) Configure SSH keys and passwordless SSH login

Now generate the SSH keys. Execute the following command in all nodes.
```
#  ssh-keygen -t rsa
```
This will ask for a passphrase. Leave the passphrase blank and press enter two times.

This will generate an SSH private key ***id_rsa*** and SSH public key ***id_rsa.pub*** under directory ***/root/.ssh/***. The keys will follow PEM standard and use RSA public key algorithm using ***2048-bit keysize***. It will be signed using the hash algorithm SHA-256. This will also be accompanied by the key fingerprint and ***randomart image*** being displayed in the standard output.

Finally enable passwordless SSH login on each nodes separately. The below given ssh-copy-id commands does the following in each nodes.

ssh-copy-id will display the ECDSA key fingerprint of the remote host, followed by a ***yes*** or ***no*** prompt; Type ***yes*** and press Enter. Further the root password of remote host will be requested. Type the password and press Enter.

Upon entering the password, ***ssh-copy-id*** copies the identity file which is the local host's public key ***/root/.ssh/id_rsa.pub*** to the remote-host's authorized keys file ***/root/.ssh/authorized_keys***. ssh-copy-id also assigns proper permission to the remote host's home and ***authorized_keys*** file.

Execute the following commands in OpenStack Control node ***10.3.0.1***
```
root@horizon-network:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.4
root@horizon-network:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.5
```
This will enable passwordless root SSH login ***from 10.3.0.1*** to nodes ***10.3.0.2 and 10.3.0.3***

Execute the following commands in OpenStack Compute node ***10.3.0.2***
```
root@nova:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.3
root@nova:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.5
```
This will enable passwordless root SSH login ***from 10.3.0.2*** to nodes ***10.3.0.1 and 10.3.0.3***

Execute the following commands in OpenStack Storage node ***10.3.0.3***
```
root@ceph-osd2:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.3
root@ceph-osd2:~#  ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.3.0.4
```
This will enable passwordless root SSH login ***from 10.3.0.3*** to nodes ***10.3.0.1 and 10.3.0.2***

6) Configure iptables

Setting up a good firewall is an essential step to secure any modern operating system. Ubuntu ships with iptables as the standard firewall. But in the default setup, the rules that we add to iptables are ephemeral. This means when we restart the server, the iptables rules will be gone. This maybe a feature for some as it gives them an avenue to get back in if they have accidentally locked themselves out of the server. However for most practical applications this is not the desired behavior.

There are a few ways to make the iptables rules persistent across boot. But the following must be noted.

For security, the iptables configuration should be applied at an early stage of the bootstrap process: preferably before any network interfaces are brought up, and certainly before any network services are started or routing is enabled. If this is not done then there will be a window of vulnerability during which the machine is remotely accessible but not firewalled.

So the best way to make iptables rules persistent across boot is with the ***iptables-persistent*** package.

Perform the following changes in all nodes.

Install ***iptables-persistent*** package.

```
#  apt install iptables-persistent
```

This will also install the ***netfilter-persistent*** package.
During installation, there will be prompts to save current IPv4 and IPv6 rules to ***/etc/iptables/rules.v4*** and ***/etc/iptables/rules.v6*** respectively. Select ***No*** for both. This is because libvirtd and associated services always activate a certain set of iptables rules when they detect a NAT network(*For us, we are in one*). So restoring the iptables from the *rules* files will create duplicate iptables entries. This is a scenario we do not want. To tackle this we will manually create the file ***/etc/iptables/rules.v4*** and also manually add rules whenever required.

Create file ***/etc/iptables/rules.v4*** with following content.

```
*filter
:INPUT ACCEPT
:FORWARD ACCEPT
:OUTPUT ACCEPT
COMMIT
```

7) Install and Configure NTP

While time passes by, computers internal clock tend to drift which can lead to inconsistent time. This is of concern, especially on servers and clusters where we want to cross check clients logs across nodes or we want to replicate server resources or databases.

NTP or Network Time Protocol comes to the rescue here. Servers can be synced using an NTP daemon that slowly shift the server clock to match a reference time that servers around the world agree upon.

Install NTP Server in all nodes.
Install the NTP package.

```
#  apt-get install ntp
```

Perform following steps in OpenStack Controller node
In configuration file ***/etc/ntp.conf***, make the following changes.

```
#restrict -4 default kod notrap nomodify nopeer noquery
#restrict -6 default kod notrap nomodify nopeer noquery
restrict -4 default kod notrap nomodify
restrict -6 default kod notrap nomodify
```

Perform the following steps in OpenStack Compute and Storage nodes
In configuration file ***/etc/ntp.conf***, make the following changes.

```
#server 0.ubuntu.pool.ntp.org
#server 1.ubuntu.pool.ntp.org
#server 2.ubuntu.pool.ntp.org
#server 3.ubuntu.pool.ntp.org
#server ntp.ubuntu.com
server horizon-network iburst
```

Restart the NTP service in all nodes.

```
# service ntp restart
```

Now let us verify the results in all nodes.

Perform the following steps in OpenStack Controller node.

Obtain and print a current list peers of the server, along with a summary of each peer's state.

```
root@horizon-network:~# ntpq -c peers
```

EXPECTED OUTPUT:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
host39.0dd0.net 199.101.100.221  3 u   15   64    3  270.195   64.244   1.149
yurizoku.tk     64.246.132.14    2 u   14   64    3  275.301   37.462  14.052
ec2-52-0-56-137 198.82.247.71    3 u   15   64    3  306.763   43.885   3.286
108.61.194.85.v 132.163.4.101    2 u   16   64    3  327.404    1.401   8.559
juniperberry.ca 193.79.237.14    2 u   14   64    3  223.545   17.464  32.015
```

Obtain and print a list of association identifiers and peer statuses for in-spec peers of the server being queried.

```
root@horizon-network:~# ntpq -c assoc
```

EXPECTED OUTPUT:

```
ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1  7170  9024   yes   yes  none    reject   reachable  2
  2  7171  9024   yes   yes  none    reject   reachable  2
  3  7172  9024   yes   yes  none    reject   reachable  2
  4  7173  9024   yes   yes  none    reject   reachable  2
  5  7174  9024   yes   yes  none    reject   reachable  2
```

Perform the following steps in OpenStack Compute node.

Obtain and print a current list peers of the server, along with a summary of each peer's state.

```
root@nova:~# ntpq -c peers
```

EXPECTED OUTPUT:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
horizon-network .INIT.          16 u   26   64    0    0.000    0.000   0.000
```

Obtain and print a list of association identifiers and peer statuses for in-spec peers of the server being queried.

```
root@nova:~# ntpq -c assoc
```

EXPECTED OUTPUT:

```
ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1 40402  8011   yes   no  none    reject    mobilize  1
```

Perform the following steps in OpenStack Storage node.

Obtain and print a current list peers of the server, along with a summary of each peer's state.

```
root@cinder1:~# ntpq -c peers
```

EXPECTED OUTPUT:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 horizon-network 216.152.240.220  3 u  21h 1024    0    0.216    1.273   0.000
```

Obtain and print a list of association identifiers and peer statuses for in-spec peers of the server being queried.

```
root@cinder1:~# ntpq -c assoc
```

EXPECTED OUTPUT:

```
ind assid status  conf reach auth condition  last_event cnt
===========================================================
  1 27489  8043   yes   no  none    reject unreachable  4
```

8) Enable OpenStack repository in all nodes

Now let us enable the OpenStack repository in all nodes.

Perform the following steps in all nodes.

Install the Ubuntu Cloud archive keyring
```
# apt-get install ubuntu-cloud-keyring
```

Install the Ubuntu Cloud archive repository
```
# echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu" "trusty-updates/juno main" >
/etc/apt/sources.list.d/cloudarchive-juno.list
```

Update the repository cache and upgrade the packages in your system.
```
# apt-get update && apt-get dist-upgrade
```

9) Install and Configure Database Server

Let us install and configure Database server in the Controller node

Install the following packages.
```
root@horizon-network:~# apt-get install mariadb-server python-mysqldb
```
NOTE: Use MariaDB root user password as 101010

In configuration file ***/etc/mysql/my.cnf***, make the following changes in file
```
#bind-address           = 127.0.0.1
bind-address = 10.3.0.1
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```

Restart the Database service
```
root@horizon-network:~# service mysql restart
```

Secure the database service
```
root@horizon-network:~# mysql_secure_installation
```

During the securing process, select the following options
Change the root password? [Y/n] n
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y

10) Install and Configure RabbitMQ message broker

Now let us install and configure RabbitMQ message broker in Controller node.

Install the following package.
root@horizon-network:~# apt-get install rabbitmq-server

Change the password for user guest.
root@horizon-network:~# rabbitmqctl change_password guest 101010
EXPECTED OUTPUT:
Changing password for user "guest" ...
...done.
NOTE:

➠ Let us use 101010 as guest user's password

➠ We must configure the rabbit_password key in the configuration file for each OpenStack service that uses the message broker.

For RabbitMQ version 3.3.0 or newer, we must enable remote access for the guest account
root@horizon:~# rabbitmqctl status | grep rabbit
Status of node rabbit@horizon ...
{running_applications,[{rabbit,"RabbitMQ","3.2.4"},

As per documentation http://docs.openstack.org/juno/install-guide/install/apt/content/ch_basic_environment.html#basics-messaging-server if RabbitMQ version is 3.3.0 or newer, we must enable remote access for the guest account.
IMPORTANT: But there were access issues even when RabbitMQ version was less than 3.3.0, and remote access for the guest account was not enabled.

To enable remote access for the guest account, add the following contents to configuration file ***/etc/rabbitmq/rabbitmq.config***
[{rabbit, [{loopback_users, []}]}].

Finally restart the RabbitMQ message broker service
root@horizon-network:~# service rabbitmq-server restart

11) Checking for errors

Finally, let us check for any errors in our basic OS setup.

Perform the following operations in all the nodes.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
# grep crit -irl /var/log
# grep err -irl /var/log
# grep warn -irl /var/log
```

12) Installed Services

Controller

/etc/init.d/ntp
/etc/init.d/mysql
/etc/init.d/rabbitmq-server

Compute

/etc/init.d/ntp

Storage

/etc/init.d/ntp

Identity Service

The OpenStack Identity Service performs the following functions:

- Tracking users and their permissions
- Providing a catalog of available services with their API endpoints

This section describes how to deploy OpenStack Identity service on the Controller node.

1) Prerequisites
    1a) Create Database
    1b) Create Administration token
2) Install required packages
3) Configuring Identity service
4) Finalize installation
5) Create Tenants, Users, Roles
6) Create Service entity and API endpoints
7) Verifying Operation
8) Creating OpenStack client environment scripts
9) Checking for errors
10) Installed Services

1) Prerequisites

Before we install and configure the OpenStack Identity service, we must do the following in Controller node.

1a) Create Database

1b) Create Administration token

1a) Create Database

Perform the following steps to create the keystone database.

Use the mysql client to connect to the database server as the root user.
root@horizon-network:~# mysql -u root -p

Create the keystone database.
MariaDB [(none)]> CREATE DATABASE keystone;

Grant proper access to the keystone database.
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY '101010';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '101010';

SYNTAX:
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'KEYSTONE_DBPASS';

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_DBPASS';

1b) Create Administration token

Generate a random value to use as the administration token during initial configuration
root@horizon-network:~# openssl rand -hex 10
d56664e8cf0cbfcac7c1

2) Install required packages

Install the packages required for OpenStack Identity Service.
root@horizon-network:~# apt-get install keystone python-keystoneclient

3) Configuring Identity service

Perform the following operations to configure Identity server.

In configuration file ***/etc/keystone/keystone.conf***, make the following changes.
[DEFAULT]
admin_token = d56664e8cf0cbfcac7c1
verbose = True

[database]
connection = mysql://keystone:101010@horizon-network/keystone

[token]
provider = keystone.token.providers.uuid.Provider
driver = keystone.token.persistence.backends.sql.Token

[revoke]
driver = keystone.contrib.revoke.backends.sql.Revoke
SYNTAX:
[DEFAULT]
admin_token = ADMIN_TOKEN
verbose = True

[database]
connection = mysql://keystone:KEYSTONE_DBPASS@CONTROLLER_HOSTNAME/keystone

[token]
provider = keystone.token.providers.uuid.Provider
driver = keystone.token.persistence.backends.sql.Token

[revoke]
driver = keystone.contrib.revoke.backends.sql.Revoke

Populate the Identity service database.
root@horizon-network:~# su -s /bin/sh -c "keystone-manage db_sync" keystone

4) Finalize installation

To finalize the installation, perform the following.

Restart the Identity service.
```
root@horizon-network:~# service keystone restart
```

By default, the Ubuntu packages create an SQLite database.

Because this configuration uses a SQL database server, we can remove the SQLite database file.
```
root@horizon-network:~# rm -f /var/lib/keystone/keystone.db
```

By default, the Identity service stores expired tokens in the database indefinitely. The accumulation of expired tokens considerably increases the database size and might degrade service performance, particularly in environments with limited resources.

So let us create a cron job to purge expired tokens hourly.
```
root@horizon-network:~# (crontab -l -u keystone 2>&1 | grep -q token_flush) || echo '@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/keystone-tokenflush.log 2>&1' >> /var/spool/cron/crontabs/keystone
```

The entry in keystone's crontab ***/var/spool/cron/crontabs/keystone***, should be as given below.
```
@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/keystone-tokenflush.log 2>&1
```

5) Create Tenants, Users, Roles

For creating Tenants, Users, and Roles, we must run the keystone commands. Before the keystone commands can be run, we must manually configure the endpoint(location) of the Identity service using the temporary administration token that we created in the previous section.

While executing keystone command, we can pass the value of

➠ administration token using either ***--os-token*** option or ***OS_SERVICE_TOKEN*** environment variable

➠ endpoint of the Identity service using either ***--os-endpoint*** option or ***OS_SERVICE_ENDPOINT*** environment variable

Here we will be using Environment Variables to reduce the command length.

Let us set,
the value of the administration token as environment variable OS_SERVICE_TOKEN, and
the location of the Identity service as environment variable OS_SERVICE_ENDPOINT

Create a file ***/etc/profile.d/keystone.sh***, with the following contents.

```
if ! env | /bin/grep -q OS_SERVICE_TOKEN ; then
export OS_SERVICE_TOKEN=d56664e8cf0cbfcac7c1
fi

if ! env | /bin/grep -q OS_SERVICE_ENDPOINT ; then
export OS_SERVICE_ENDPOINT=http://horizon-network:35357/v2.0
fi
```

SYNTAX:

```
if ! env | /bin/grep -q OS_SERVICE_TOKEN ; then
export OS_SERVICE_TOKEN=ADMIN_TOKEN
fi

if ! env | /bin/grep -q OS_SERVICE_ENDPOINT ; then
export OS_SERVICE_ENDPOINT=http://CONTROLLER_HOSTNAME:35357/v2.0
fi
```

Finally, source the ***keystone.sh*** file.

```
root@horizon-network:~# source /etc/profile.d/keystone.sh
```

Now let us start creating Tenants, Users, and Roles for our environment.

Create the ***admin*** tenant.

```
root@horizon-network:~# keystone tenant-create --name admin --description "Admin Tenant"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Admin Tenant           |
|   enabled   |               True               |
|      id     | 4c103890ca11491aa74bcd4c48194cf6 |
|     name    |              admin               |
+-------------+----------------------------------+
```

Create the ***admin*** user.

```
root@horizon-network:~# keystone user-create --name admin --pass 101010 --email root@localhost
```

SYNTAX:

```
root@horizon-network:~# keystone user-create --name admin --pass ADMIN_PASS --email EMAIL_ADDRESS
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         root@localhost           |
| enabled  |              True                |
|    id    | 7a5f2bef969e4a84979f75cbebcd8930 |
|   name   |              admin               |
| username |              admin               |
+----------+----------------------------------+
```

Create the ***admin*** role.

NOTE: Any roles created must map to roles specified in the ***policy.json*** file included with each OpenStack service.

```
root@horizon-network:~# keystone role-create --name admin
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 4bcee42818254e6fb0856bd7c9427cbb |
|   name   |              admin               |
+----------+----------------------------------+
```

Add admin role to the admin tenant and user.

```
root@horizon-network:~# keystone user-role-add --user admin --tenant admin --role admin
```

Also, we must create a demo tenant and user for typical operations in our environment.
Create the demo tenant.

```
root@horizon-network:~# keystone tenant-create --name demo --description "Demo Tenant"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Demo Tenant            |
|   enabled   |               True               |
|      id     | a2aef343e2c54ef8bb8cd0fe0e4f3946 |
|     name    |               demo               |
+-------------+----------------------------------+
```

Create the demo user under the demo tenant.

```
root@horizon-network:~# keystone user-create --name demo --tenant demo --pass 101010 --email root@localhost
```

SYNTAX:

```
root@horizon-network:~# keystone user-create --name demo --tenant demo --pass DEMO_PASS --email EMAIL_ADDRESS
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |          root@localhost          |
| enabled  |               True               |
|    id    | 3e9faa9ea1be47ea8b079e483b95e64f |
|   name   |               demo               |
| tenantId | a2aef343e2c54ef8bb8cd0fe0e4f3946 |
| username |               demo               |
+----------+----------------------------------+
```

OpenStack services also require a tenant, user, and role to interact with other services. Each service typically requires creating one or more unique users with the admin role under the service tenant.

Create the service tenant.

```
root@horizon-network:~# keystone tenant-create --name service --description "Service Tenant"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |               True               |
|      id     | 24dba7072f56456b8d98e2c4304058a4 |
|     name    |             service              |
+-------------+----------------------------------+
```

6) Create Service entity and API endpoints

The Identity service manages a catalog of services in our OpenStack environment. Services use this catalog to locate other services in our environment.

Let us create the service entity for the Identity service.

```
root@horizon-network:~# keystone service-create --name keystone --type identity --description "OpenStack Identity"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |         OpenStack Identity       |
|   enabled   |               True               |
|      id     | 392c5c2da3a34af6a9f28886228d3676 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

The Identity service manages a catalog of API endpoints associated with the services in our OpenStack environment. Services use this catalog to determine how to communicate with other services in our environment. OpenStack provides three API endpoint variations for each service: admin, internal, and public. In a production environment, the variants might reside on separate networks that service different types of users for security reasons. Also, OpenStack supports multiple regions for scalability. For simplicity, this configuration uses the management network for all endpoint variations and the regionOne region.

Let us create the Identity service API endpoints.

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ identity / {print $2}') --publicurl http://horizon-network:5000/v2.0 --internalurl http://horizon-network:5000/v2.0 --adminurl http://horizon-network:35357/v2.0 --region regionOne
```

SYNTAX:

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ identity / {print $2}') --publicurl http://CONTROLLER_HOSTNAME:5000/v2.0 --internalurl http://CONTROLLER_HOSTNAME:5000/v2.0 --adminurl http://CONTROLLER_HOSTNAME:35357/v2.0 --region regionOne
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  | http://horizon-network:35357/v2.0 |
|      id     | 81338e2d42a44aa4bd5cd9a8e1aebc8e |
| internalurl |  http://horizon-network:5000/v2.0 |
|  publicurl  |  http://horizon-network:5000/v2.0 |
|    region   |            regionOne             |
| service_id  |  392c5c2da3a34af6a9f28886228d3676 |
+-------------+----------------------------------+
```

7) Verifying Operation

Now let us verify the operation of OpenStack Identity service.

Unset the temporary environment variables OS_SERVICE_TOKEN and OS_SERVICE_ENDPOINT.

```
root@horizon-network:~# unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

As the admin tenant and user, request an authentication token.

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password 101010 --os-auth-url http://horizon-network:35357/v2.0 token-get
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 token-get
```

EXPECTED OUTPUT:

```
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2016-03-29T21:37:03Z       |
|     id    | ce4f6d4150954e3f87f25c6330538a86 |
| tenant_id | 4c103890ca11491aa74bcd4c48194cf6 |
|  user_id  | 7a5f2bef969e4a84979f75cbebcd8930 |
+-----------+----------------------------------+
```

As the admin tenant and user, list tenants to verify the following.

→ admin tenant and user can execute admin-only CLI commands

→ Identity service contains the tenants that we had created

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password
101010 --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 tenant-list
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password
ADMIN_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 tenant-list
```

EXPECTED OUTPUT:

```
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 4c103890ca11491aa74bcd4c48194cf6 |  admin  |   True  |
| a2aef343e2c54ef8bb8cd0fe0e4f3946 |   demo  |   True  |
| 24dba7072f56456b8d98e2c4304058a4 | service |   True  |
+----------------------------------+---------+---------+
```

As the admin tenant and user, list users to verify that the Identity service contains the users that we had created.

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password
101010 --os-auth-url http://horizon-network:35357/v2.0 user-list
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password
ADMIN_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 user-list
```

EXPECTED OUTPUT:

```
+----------------------------------+-------+---------+----------------+
|                id                | name | enabled |     email      |
+----------------------------------+-------+---------+----------------+
| 7a5f2bef969e4a84979f75cbebcd8930 | admin |   True  | root@localhost |
| 3e9faa9ea1be47ea8b079e483b95e64f |  demo |   True  | root@localhost |
+----------------------------------+-------+---------+----------------+
```

As the admin tenant and user, list roles to verify that the Identity service contains the role that we had created.

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password 101010 --os-auth-url http://horizon-network:35357/v2.0 role-list
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name admin --os-username admin --os-password ADMIN_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 role-list
```

EXPECTED OUTPUT:

```
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| 4bcee42818254e6fb0856bd7c9427cbb |  admin   |
+----------------------------------+----------+
```

As the demo tenant and user, request an authentication token.

```
root@horizon-network:~# keystone --os-tenant-name demo --os-username demo --os-password 101010 --os-auth-url http://horizon-network:35357/v2.0 token-get
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 token-get
```

EXPECTED OUTPUT:

```
+-----------+----------------------------------+
|  Property |              Value               |
+-----------+----------------------------------+
|  expires  |       2016-03-29T21:38:05Z       |
|     id    | 1fa725709dc845aba10b77867285a849 |
| tenant_id | a2aef343e2c54ef8bb8cd0fe0e4f3946 |
|  user_id  | 3e9faa9ea1be47ea8b079e483b95e64f |
+-----------+----------------------------------+
```

As the demo tenant and user, attempt to list users to verify that we cannot execute admin-only CLI commands.

```
root@horizon-network:~# keystone --os-tenant-name demo --os-username demo --os-password 101010 --os-auth-url http://horizon-network:35357/v2.0 user-list
```

SYNTAX:

```
root@horizon-network:~# keystone --os-tenant-name demo --os-username demo --os-password DEMO_PASS --os-auth-url http://CONTROLLER_HOSTNAME:35357/v2.0 user-list
```

EXPECTED OUTPUT:

```
You are not authorized to perform the requested action: admin_required (HTTP 403)
```

8) Creating OpenStack client environment scripts

The previous section used a combination of environment variables and command options to interact with the Identity service via the keystone client. To increase efficiency of client operations, OpenStack supports simple client environment scripts also known as OpenRC files. These scripts typically contain common options for all clients, but also support unique options. For more information, see the OpenStack User Guide at http://docs.openstack.org/user-guide/content/cli_openrc.html

Let us create a Client environment script for admin user.
Create a file ***/root/admin-openrc.sh*** with following contents.

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=101010
export OS_AUTH_URL=http://horizon-network:35357/v2.0
```

SYNTAX:

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_AUTH_URL=http://CONTROLLER_HOSTNAME:35357/v2.0
```

Also let us a Client environment script for demo user.
Create a file ***/root/demo-openrc.sh*** with following contents.

```
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=101010
export OS_AUTH_URL=http://horizon-network:5000/v2.0
```

SYNTAX:

export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://CONTROLLER_HOSTNAME:5000/v2.0

9) Checking for errors

Finally, let us check for any errors in our deployment of OpenStack Identity service.

Perform the following operations in Controller node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

10) Installed Services

Controller

/etc/init/keystone.conf

Image Service

The OpenStack Image Service performs the following functions:

➠ Enables users to discover, register, and retrieve virtual machine images

➠ Offers a REST API that enables us to query virtual machine image metadata and retrieve an actual image

We can store virtual machine images made available through the Image Service in a variety of locations, from simple file systems to object-storage systems like OpenStack Object Storage. For simplicity, this configuration stores images on the local file system.

This section describes how to deploy OpenStack Image Service(code-named glance), on the Controller node.

1) Prerequisites

    1a) Create Database

2) Create Users and Roles

3) Create Service entity and API endpoints

4) Installing required packages

5) Configuring Image service

    5a) Configure Image service registry

    5b) Configure Image service API

6) Finalize installation

7) Verifying Operation

8) Checking for errors

9) Installed Services

1) Prerequisites

Before we install and configure the OpenStack Image service, we must do the following in Controller node.

1a) Create Database

1a) Create Database

Perform the following steps to create the glance database.

Use the mysql access client to connect to the database server as root user.
root@horizon-network:~# mysql -u root -p

Create the glance database.
MariaDB [(none)]> CREATE DATABASE glance;

Grant proper access to the glance database.
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY '101010';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY '101010';
SYNTAX:
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'GLANCE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'GLANCE_DBPASS';

2) Create Users and Roles

Let us create Users and Roles for our environment.

Source the client environment script for admin user.
root@horizon-network:~# source admin-openrc.sh

Create the glance user.
root@horizon-network:~# keystone user-create --name glance --pass 101010

SYNTAX:

root@horizon-network:~# keystone user-create --name glance --pass GLANCE_PASS

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 405adda8faba40c1bee466e25a07d4ad |
|   name   |              glance              |
| username |              glance              |
+----------+----------------------------------+
```

Add admin role to the glance user.

root@horizon-network:~# keystone user-role-add --user glance --tenant service --role admin

3) Create Service entity and API endpoints

Let us create the service entity for glance service.

root@horizon-network:~# keystone service-create --name glance --type image --description "OpenStack Image Service"

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Image Service      |
|   enabled   |               True               |
|      id     | d0091016419e4070996321d175681444 |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+
```

Now let us create the glance API endpoints.

root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ image / {print $2}') --publicurl http://horizon-network:9292 --internalurl http://horizon-network:9292 --adminurl http://horizon-network:9292 --region regionOne

SYNTAX:

root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ image / {print $2}') --publicurl http://CONTROLLER_HOSTNAME:9292 --internalurl http://CONTROLLER_HOSTNAME:9292 --adminurl http://CONTROLLER_HOSTNAME:9292 --region regionOne

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |   http://horizon-network:9292    |
|      id     | 2905fcbce6054e9bb93f28c545312f71 |
| internalurl |   http://horizon-network:9292    |
|  publicurl  |   http://horizon-network:9292    |
|    region   |           regionOne              |
|  service_id | d0091016419e4070996321d175681444 |
+-------------+----------------------------------+
```

4) Installing required packages

Install the packages required for OpenStack Image Service.

```
root@horizon-network:~# apt-get install glance python-glanceclient
```

5) Configuring Image service

Configuring the Image service involves the following two steps.

- **5a) Configure Image service registry**
- **5b) Configure Image service API**

5a) Configure Image service registry

In configuration file ***/etc/glance/glance-api.conf***, perform the following.

Configure database access.

In ***[database]*** section, make the following change.

```
connection = mysql://glance:101010@horizon-network/glance
```

SYNTAX:
connection = mysql://glance:GLANCE_DBPASS@CONTROLLER_HOSTNAME/glance

Configure Identity service access.
In ***[keystone_authtoken]*** section, make the following changes.
auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = glance
admin_password = 101010
SYNTAX:
auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

In ***[paste_deploy]*** section, make the following changes.
flavor = keystone

Configure the local file system store and location of image files
In ***[glance_store]*** section, make the following changes.
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

Configure the noop notification driver to disable notifications. This is because they only pertain to the optional Telemetry service. The Telemetry chapter provides an Image Service configuration that enables notifications.

In ***[DEFAULT]*** section, make the following changes.
notification_driver = noop

Enable verbose logging. This is to assist with troubleshooting.
In ***[DEFAULT]*** section, make the following changes.
verbose = True

5b) Configure Image service API

In configuration file ***/etc/glance/glance-registry.conf***, perform the following.

Configure database access.

In ***[database]*** section, make the following changes.
connection = mysql://glance:101010@horizon-network/glance
SYNTAX:
connection = mysql://glance:GLANCE_DBPASS@CONTROLLER_HOSTNAME/glance

Configure Identity service access.
In ***[keystone_authtoken]*** section, make the following changes.
auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = glance
admin_password = 101010
SYNTAX:
auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

In ***[paste_deploy]*** section, make the following changes.
flavor = keystone

Configure the noop notification driver to disable notifications. This is because they only pertain to the optional Telemetry service. The Telemetry chapter provides an Image Service configuration that enables notifications.
In ***[DEFAULT]*** section, make the following changes.
notification_driver = noop

Enable verbose logging. This is to assist with troubleshooting.
In ***[DEFAULT]*** section, make the following changes.
verbose = True

Now as the changes in configuration file are made, populate the Image Service database.
root@horizon-network:~# su -s /bin/sh -c "glance-manage db_sync" glance

6) Finalize installation

To finalize the installation, perform the following.

Restart the Image services.
```
root@horizon-network:~# service glance-registry restart
root@horizon-network:~# service glance-api restart
```

By default, the Ubuntu packages create an SQLite database.

Because this configuration uses a SQL database server, we can remove the SQLite database file.
```
root@horizon-network:~# rm -f /var/lib/glance/glance.sqlite
```

7) Verifying Operation

Now let us verify the operation of the Openstack Image service.

We will be using CirrOS for this purpose. CirrOS is a tiny Linux OS that specializes in running on a cloud. We can download CirrOS from https://launchpad.net/cirros

Create a directory named ***images*** in ***/tmp/***
```
root@horizon-network:~# mkdir /tmp/images
```

Change to the directory ***/tmp/images/***
```
root@horizon-network:~# cd /tmp/images
```

Download the CirrOS disk image to the this directory.
```
root@horizon-network:~# wget -P /tmp/images http://download.cirros-cloud.net/0.3.3/cirros-0.3.3-x86_64-disk.img
```

Now source the client environment script for admin user.
```
root@horizon-network:~# source admin-openrc.sh
```

Upload the CirrOS image to the Image Service.
```
root@horizon-network:~# glance image-create --name "cirros-0.3.3-x86_64" --file /tmp/images/cirros-0.3.3-x86_64-disk.img --disk-format qcow2 --container-format bare --is-public True –progress
```

EXPECTED OUTPUT:

```
[=============================>] 100%
+-----------------+----------------------------------+
| Property        | Value                            |
+-----------------+----------------------------------+
| checksum        | 133eae9fb1c98f45894a4e60d8736619 |
| container_format | bare                            |
| created_at      | 2016-03-29T21:19:48              |
| deleted         | False                            |
| deleted_at      | None                             |
| disk_format     | qcow2                            |
| id              | 55dda035-43fe-49a3-9cde-12c2b04b29d0 |
| is_public       | True                             |
| min_disk        | 0                                |
| min_ram         | 0                                |
| name            | cirros-0.3.3-x86_64              |
| owner           | 4c103890ca11491aa74bcd4c48194cf6 |
| protected       | False                            |
| size            | 13200896                         |
| status          | active                           |
| updated_at      | 2016-03-29T21:19:49              |
| virtual_size    | None                             |
+-----------------+----------------------------------+
```

Confirm upload of the CirrOS image and validate the image attributes.

```
root@horizon-network:~# glance image-list
```

EXPECTED OUTPUT:

```
+--------------------------------------+---------------------+-------------+-----------------+----------+--------+
| ID                                   | Name                | Disk Format | Container Format | Size     | Status |
+--------------------------------------+---------------------+-------------+-----------------+----------+--------+
| 55dda035-43fe-49a3-9cde-12c2b04b29d0 | cirros-0.3.3-x86_64 | qcow2       | bare             | 13200896 | active |
+--------------------------------------+---------------------+-------------+-----------------+----------+--------+
```

Now we may remove the temporary directory named ***images*** in ***/tmp/***.

```
root@horizon-network:~# rm -r /tmp/images
```

8) Checking for errors

Finally, let us check for any errors in our deployment of OpenStack Image service.

Perform the following operations in Controller node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

9) Installed Services

Controller

/etc/init/glance-api.conf
/etc/init/glance-registry.conf

Compute Service

OpenStack Compute is a major part of an IaaS system. It is used to host and manage Cloud computing systems.

OpenStack Compute interacts with

- OpenStack Identity for authentication
- OpenStack Image Service for disk and server images
- OpenStack dashboard for the user and administrative interface

It consists of the following areas along with their components

- ➟ API
- ➟ Compute core
- ➟ Networking for VMs
- ➟ Console interface
- ➟ Image management (EC2 scenario)
- ➟ Command-line clients and other interfaces
- ➟ Queue
- ➟ database

This section describes how to install and configure the Compute service, code-named nova.

1) Configuring Controller Node

1a) Prerequisites

1a1) Create Database

1a2) Create Compute service credentials

1a3) Create Compute service API endpoints

1b) Install required packages

1c) Configure Compute components

1c1) Configure Database access

1c2) Configure Message broker access

1c3) Configure Authentication mechanism

1c4) Configure Management interface

1c5) Configure VNC proxy

1c6) Configure Image service location

1d) Finalize installation

2) Configuring Compute Node

- 2a) Install required packages
- 2b) Configure Compute components
  - 2b1) Configure Message broker access
  - 2b2) Configure Authentication mechanism
  - 2b3) Configure Management interface
  - 2b4) Configure Remote console access
  - 2b5) Configure Image service location
  - 2b6) Configure Hardware acceleration
- 2c) Finalize installation

3) Verify operation

4) Check for errors

5) Installed services

1) Configuring Controller Node

Perform the following operations in Controller Node.

1a) Prerequisites

Before we install and configure the Compute service, we must do the following in Controller node.

1a1) Create Database

1a2) Create Compute service credentials

1a3) Create Compute service API endpoints

1a1) Create Database

Performing the following steps to create nova database.

Use the mysql client to connect to the database server as root user.

```
root@horizon-network:~# mysql -u root -p
```

Create the nova database.

```
MariaDB [(none)]> CREATE DATABASE nova;
```

Grant proper access to the nova database.

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY '101010';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY '101010';
```

SYNTAX:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'NOVA_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'NOVA_DBPASS';
```

1a2) Create Compute service credentials

Source the client environment script for admin user.

```
root@horizon-network:~# source admin-openrc.sh
```

Create the nova user.

```
root@horizon-network:~# keystone user-create --name nova --pass 101010
```

SYNTAX:

```
root@horizon-network:~# keystone user-create --name nova --pass NOVA_PASS
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 90b9c99e2fbb4c81bcd61caab2748795 |
|   name   |               nova               |
| username |               nova               |
+----------+----------------------------------+
```

Add admin role to the nova user.

```
root@horizon-network:~# keystone user-role-add --user nova --tenant service --role admin
```

Create the nova service entity.

```
root@horizon-network:~# keystone service-create --name nova --type compute --description "OpenStack Compute"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        OpenStack Compute         |
|   enabled   |               True               |
|      id     | 8946fe8725014f658a7235d5d7228803 |
|     name    |               nova               |
|     type    |             compute              |
+-------------+----------------------------------+
```

1a3) Create Compute service API endpoints

Perform the following steps to create Compute Service API endpoints.

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ compute / {print $2}') --publicurl http://horizon-network:8774/v2/%\(tenant_id\)s --internalurl http://horizon-network:8774/v2/%\(tenant_id\)s --adminurl http://horizon-network:8774/v2/%\(tenant_id\)s --region regionOne
```

SYNTAX:

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ compute / {print $2}') --publicurl http://CONTROLLER_HOSTNAME:8774/v2/%\(tenant_id\)s --internalurl http://CONTROLLER_HOSTNAME:8774/v2/%\(tenant_id\)s --adminurl http://CONTROLLER_HOSTNAME:8774/v2/%\(tenant_id\)s --region regionOne
```

EXPECTED OUTPUT:

```
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://horizon-network:8774/v2/%(tenant_id)s |
|      id     |      b18f040a58834be9981fce86183f9330       |
| internalurl | http://horizon-network:8774/v2/%(tenant_id)s |
|  publicurl  | http://horizon-network:8774/v2/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |      8946fe8725014f658a7235d5d7228803       |
+-------------+-----------------------------------------+
```

1b) Install required packages

Install the necessary required for OpenStack Compute service.

```
root@horizon-network:~# apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-scheduler python-novaclient
```

Now, the packages nova-cert and nova-scheduler installs the services nova-cert and nova-scheduler respectively. They are installed as Upstart jobs which start asynchronously. On startup these services try to connect to an available MySQL Server on Controller. Only problem is that the mysql service is installed as a SysV service which does not start asynchronously.

So when nova-cert and nova-scheduler services start, MySQL server most probably will not be ready. As a result these two services will fail to start.

To remedy this, In the configuration files, ***/etc/init/nova-cert.conf*** and ***/etc/init/nova-scheduler.conf***, make the following changes.

```
#start on runlevel
start on mysql_started
```

1c) Configure Compute components

Now let us configure Compute service components. This involves the following six steps.

1c1) Configure Database access

1c2) Configure Message broker access

1c3) Configure Authentication mechanism

1c4) Configure Management interface

1c5) Configure VNC proxy

1c6) Configure Image service location

In configuration file ***/etc/nova/nova.conf***, make the following changes.

1c1) Configure Database access

Let us specify the MySQL connection string.

In ***[database]*** section, make the following changes.

```
connection = mysql://nova:101010@horizon-network/nova
```

SYNTAX:

```
connection = mysql://nova:NOVA_DBPASS@CONTROLLER_HOSTNAME/nova
```

1c2) Configure Message broker access

Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.

```
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
```

SYNTAX:

```
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS
```

1c3) Configure Authentication mechanism

Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.

auth_strategy = keystone

In ***[keystone_authtoken]*** section, make the following changes.

auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = nova
admin_password = 101010

SYNTAX:

auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS

IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

1c4) Configure Management interface

Configure Compute service to use the management interface IP address of controller node.

In ***[DEFAULT]*** section, make the following changes.

my_ip = 10.3.0.1

SYNTAX:

my_ip = CONTROLLER_IP

1c5) Configure VNC proxy

Configure VNC proxy to use management interface IP address of controller node.

In ***[DEFAULT]*** section, make the following changes.

vncserver_listen = 10.3.0.1\
vncserver_proxyclient_address = 10.3.0.1

SYNTAX:

vncserver_listen = CONTROLLER_IP
vncserver_proxyclient_address = CONTROLLER_IP

1c6) Configure Image service location

Configure the location of image service.

In ***[glance]*** section, make the following changes.

host = horizon-network

SYNTAX:

host = CONTROLLER_HOSTNAME

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.

verbose = True

1d) Finalize installation

Let us finalize the installation.

Populate the Compute database.

```
root@horizon-network:~# su -s /bin/sh -c "nova-manage db sync" nova
```

Restart the Compute services.

```
root@horizon-network:~# service nova-api restart
root@horizon-network:~# service nova-cert restart
root@horizon-network:~# service nova-consoleauth restart
root@horizon-network:~# service nova-scheduler restart
root@horizon-network:~# service nova-conductor restart
root@horizon-network:~# service nova-novncproxy restart
```

By default, the Ubuntu packages create an SQLite database.

Because this configuration uses a SQL database server, we can remove the SQLite database file.

```
root@horizon-network:~# rm -f /var/lib/nova/nova.sqlite
```

2) Configuring Compute Node

Compute service supports several hypervisors to deploy instances or VMs. For simplicity, this configuration uses the QEMU hypervisor with the KVM extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor.

2a) Install required packages

Install the required packages for OpenStack Compute service in compute node.
root@nova:~# apt-get install nova-compute sysfsutils

2b) Configure Compute components

Now let us configure Compute service components. This involves the following six steps.

2b1) Configure Message broker access

2b2) Configure Authentication mechanism

2b3) Configure Management interface

2b4) Configure Remote console access

2b5) Configure Image service location

2b6) Configure Hardware acceleration

In configuration file ***/etc/nova/nova.conf***, make the following changes.

2b1) Configure Message broker access

Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
SYNTAX:
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS

2b2) Configure Authentication mechanism

Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.

auth_strategy = keystone

In ***[keystone_authtoken]*** section, make the following changes.

auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = nova
admin_password = 101010

SYNTAX:

auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS

IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

2b3) Configure Management interface

Configure Compute service to use the management interface IP address of compute node.

In ***[DEFAULT]*** section, make the following changes.

my_ip = 10.3.0.2

SYNTAX:

my_ip = NOVA_IP

2b4) Configure Remote console access

Configure remote console access.

In ***[DEFAULT]*** section, make the following changes.

vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = 10.3.0.2
novncproxy_base_url = http://10.3.0.1:6080/vnc_auto.html

SYNTAX:
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = NOVA_IP
novncproxy_base_url = http://CONTROLLER_IP:6080/vnc_auto.html
NOTE:

- Server component listens on all IP addresses
- Proxy component only listens on the management interface IP address of the compute node.
- Base URL indicates the location where you can use a web browser to access remote consoles of instances on this compute node.
- If the web browser to access remote consoles resides on a host that cannot resolve the controller hostname, we must replace controller with the management interface IP address of the controller node.

2b5) Configure Image service location
Configure the location of image service.

In ***[glance]*** section, make the following changes.
host = horizon-network
SYNTAX:
host = CONTROLLER_HOSTNAME

Enable verbose logging.
In ***[DEFAULT]*** section, make the following changes.
verbose = True

2b6) Configure Hardware acceleration
Determine whether our compute node supports hardware acceleration for virtual machines
root@nova:~# egrep -c '(vmx|svm)' /proc/cpuinfo

If the above command returns a value of 1 or greater, our compute node supports hardware acceleration which typically requires no additional configuration.

If the above command returns a value of 0, our compute node does not support hardware acceleration and we must configure libvirt to use QEMU instead of KVM.

In configuration file ***/etc/nova/nova-compute.conf***, make the following changes.
In ***[libvirt]*** section, make the following changes.
virt_type = qemu

2c) Finalize installation

Let us finalize the installation.

Restart the Compute service.
```
root@nova:~# service nova-compute restart
```

By default, the Ubuntu packages create an SQLite database.
Because this configuration uses a SQL database server, we can remove the SQLite database file.
```
root@nova:~# rm -f /var/lib/nova/nova.sqlite
```

3) Verify operation

Now let us verify the operation of the Openstack Compute service

Perform the following operations in controller node.

Source the client environment script for admin user
root@horizon-network:~# source admin-openrc.sh

List service components to verify successful launch of each process
root@horizon-network:~# nova service-list
EXPECTED OUTPUT:

```
+----+------------------+-----------------+----------+---------+-------+----------------------------+-----------------+
| Id | Binary           | Host            | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+----+------------------+-----------------+----------+---------+-------+----------------------------+-----------------+
| 1  | nova-cert        | horizon-network | internal | enabled | up    | 2016-03-29T22:05:50.000000 | -
|
| 2  | nova-consoleauth | horizon-network | internal | enabled | up    | 2016-03-29T22:05:50.000000 | -
|
| 3  | nova-scheduler   | horizon-network | internal | enabled | up    | 2016-03-29T22:05:50.000000 | -
|
| 4  | nova-conductor   | horizon-network | internal | enabled | up    | 2016-03-29T22:05:50.000000 | -
|
| 5  | nova-compute     | nova            | nova     | enabled | up    | 2016-03-29T22:05:54.000000 | -               |
+----+------------------+-----------------+----------+---------+-------+----------------------------+-----------------+
```

NOTE: This output should indicate 4 components enabled on the controller node 1 component enabled on the compute node.

List images in the Image Service catalog to verify connectivity with the Identity service and Image Service.
root@horizon-network:~# nova image-list
EXPECTED OUTPUT:

```
+--------------------------------------+---------------------+--------+--------+
| ID                                   | Name                | Status | Server |
+--------------------------------------+---------------------+--------+--------+
| 55dda035-43fe-49a3-9cde-12c2b04b29d0 | cirros-0.3.3-x86_64 | ACTIVE |        |
+--------------------------------------+---------------------+--------+--------+
```

4) Check for errors

Finally, let us check for any errors in our deployment of OpenStack Compute service

Perform the following operations in controller node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

Perform the following operations in compute node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
root@nova:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@nova:~# grep crit -irl /var/log
root@nova:~# grep err -irl /var/log
root@nova:~# grep warn -irl /var/log
```

5) Installed Services

Controller

/etc/init/nova-api.conf
/etc/init/nova-cert.conf
/etc/init/nova-conductor.conf
/etc/init/nova-consoleauth.conf
/etc/init/nova-novncproxy.conf
/etc/init/nova-scheduler.conf

Nova

/etc/init.d/open-iscsi
/etc/init.d/sysfsutils
/etc/init.d/umountiscsi.sh [Not a service, and not added as a service. This script is invoked by /etc/init.d/open-iscsi]

/etc/init/iscsi-network-interface.conf
/etc/init/nova-compute.conf

Networking Service

OpenStack Networking service enables operators to leverage different networking technologies to power their cloud networking. It allows two types of Networking

➠ Legacy networking (nova-network)

This enables us to deploy one network type per instance and is suitable for basic network functionality.

➠ OpenStack Networking (neutron)

This enables us to deploy multiple network types per instance. It includes plug-ins accommodate different networking equipment and software, providing flexibility to OpenStack architecture and deployment.

Here we will be using OpenStack Networking (neutron).

This section describes how to install and configure OpenStack's Neutron networking component.

1) Configuring Controller Node
    1a) Prerequisites
        1a1) Create Database
        1a2) Create Neutron service entity
        1a3) Create Neutron service API endpoints
    1b) Install required packages
    1c) Configure Networking server components
        1c1) Configure Database access
        1c2) Configure Message broker access
        1c3) Configure Authentication mechanism
        1c4) Configure Network topology change notifications
        1c5) Configure Plugins
        1c6) Reconfigure Compute
    1d) Finalize installation
    1e) Verify operation
    1f) Check for errors

2) Configuring Network Node
    2a) Change Network kernel settings
    2b) Install required packages
    2c) Configure Networking server component
        2c1) Configure Database access
        2c2) Configure Message broker access
        2c3) Configure Authentication mechanism
        2c4) Configure Plugins
        2c5) Configure L3 agent
        2c6) Configure DHCP agent
        2c7) Configure Metadata agent
        2c8) Reconfigure Compute

- 2c9) Configure Network interfaces
- 2c10) Configure Bridges
- 2d) Finalize installation
- 2e) Verify operation
- 2f) Check for errors

3) Configuring Compute Node

- 3a) Change Network kernel settings
- 3b) Install required packages
- 3c) Configure Networking server component
  - 3c1) Configure Database access
  - 3c2) Configure Message broker access
  - 3c3) Configure Authentication mechanism
  - 3c4) Configure Plugins
  - 3c5) Reconfigure Compute
  - 3c6) Configure Network interfaces
  - 3c7) Configure Bridges
- 3d) Finalize installation
- 3e) Verify operation
- 3f) Check for errors

4) Installed Services

1) Configuring Controller Node

Perform the following operations in Controller Node.

1a) Prerequisites

Before we install and configure the Neutron service, we must do the following in Controller node.

1a1) Create Database
1a2) Create Neutron service entity
1a3) Create Neutron service API endpoints

1a1) Create Database

Performing the following steps to create neutron database.

Use the mysql client to connect to the database server as root user.

```
root@horizon-network:~# mysql -u root -p
```

Create the neutron database.

```
MariaDB [(none)]> CREATE DATABASE neutron;
```

Grant proper access to the neutron database.

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY '101010';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY '101010';
```

SYNTAX:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' IDENTIFIED BY 'NEUTRON_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'NEUTRON_DBPASS';
```

1a2) Create Neutron service entity

Source the client environment script for admin user.

```
root@horizon-network:~# source admin-openrc.sh
```

Create the neutron user.

```
root@horizon-network:~# keystone user-create --name neutron --pass 101010
```

SYNTAX:

```
root@horizon-network:~# keystone user-create --name neutron --pass NEUTRON_PASS
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 54d8b64fbba146738c35450b850a6d17 |
|   name   |              neutron             |
| username |              neutron             |
+----------+----------------------------------+
```

Add admin role to the neutron user.

```
root@horizon-network:~# keystone user-role-add --user neutron --tenant service --role admin
```

Create the neutron service entity.

```
root@horizon-network:~# keystone service-create --name neutron --type network --description
"OpenStack Networking"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       OpenStack Networking       |
|   enabled   |               True               |
|      id     | 08e520167765499f98e9073845001078 |
|     name    |              neutron             |
|     type    |              network             |
+-------------+----------------------------------+
```

1a3) Create the Networking service API endpoints

Perform the following steps to create the Networking service API endpoints.

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ network /
{print $2}') --publicurl http://horizon-network:9696 --adminurl http://horizon-network:9696 --internalurl
http://horizon-network:9696 --region regionOne
```

SYNTAX:

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ network /
{print $2}') --publicurl http://CONTROLLER_HOSTNAME:9696 --adminurl
http://CONTROLLER_HOSTNAME:9696 --internalurl http://CONTROLLER_HOSTNAME:9696 --
region regionOne
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|  Property   |              Value               |
+-------------+----------------------------------+
|  adminurl   |   http://horizon-network:9696    |
|      id     | 34ed928fb2014f779def3a11161f4224 |
| internalurl |   http://horizon-network:9696    |
|  publicurl  |   http://horizon-network:9696    |
|    region   |            regionOne             |
| service_id  | 08e520167765499f98e9073845001078 |
+-------------+----------------------------------+
```

1b) Install required packages

Install the required packages for OpenStack Networking.

```
root@horizon-network:~# apt-get install neutron-server neutron-plugin-ml2 python-neutronclient
```

1c) Configure Networking server components

Now let us configure Networking server components. This involves the following six steps.

1c1) Configure Database access
1c2) Configure Message broker access
1c3) Configure Authentication mechanism
1c4) Configure Network topology change notifications
1c5) Configure Plugins
1c6) Reconfigure Compute

In configuration file /etc/neutron/neutron.conf, perform the following.

1c1) Configure Database access

Let us specify the MySQL connection string.

In ***[database]*** section, make the following changes.

```
connection = mysql://neutron:101010@horizon-network/neutron
```

SYNTAX:

```
connection = mysql://neutron:NEUTRON_DBPASS@CONTROLLER_HOSTNAME/neutron
```

1c2) Configure Message broker access

Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.

rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010

SYNTAX:

rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS

1c3) Configure Authentication mechanism

Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.

auth_strategy = keystone

In ***[keystone_authtoken]*** section, make the following changes.

auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = neutron
admin_password = 101010

SYNTAX:

auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = neutron
admin_password = 101010

IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

1c4) Configure Network topology change notifications

Let us setup Networking to notify Compute of network topology changes.

In ***[DEFAULT]*** section, make the following changes.

notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://horizon-network:8774/v2
nova_admin_auth_url = http://horizon-network:35357/v2.0
nova_region_name = regionOne
nova_admin_username = nova
nova_admin_tenant_id = 24dba7072f56456b8d98e2c4304058a4
nova_admin_password = 101010

SYNTAX:

notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://CONTROLLER_HOSTNAME:8774/v2
nova_admin_auth_url = http://CONTROLLER_HOSTNAME:35357/v2.0
nova_region_name = regionOne
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID
nova_admin_password = NOVA_PASS

NOTE: ***nova_admin_tenant_id*** is the id of the ***Service Tenant*** we created while configuring Identity service.

To get this id, we may execute the following command in Controller node.

root@horizon-network:~# source admin-openrc.sh
root@horizon-network:~# keystone tenant-get service

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.

verbose = True

1c5) Configure Plugins

Let us enable the ML2 or Modular Layer 2 plugin, Router service plugin, and Overlapping IP addresses.

In ***[DEFAULT]*** section, make the following changes.

core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

NOTE: Option 'core_plugin = ml2' may be already enabled.

Now let us configure the ML2 plugin.
The ML2 plugin uses the OVS or Open vSwitch mechanism/agent to build the virtual networking framework for instances.
NOTE: Controller node does not need the OVS components because it does not handle instance network traffic.

In configuration file ***/etc/neutron/plugins/ml2/ml2_conf.ini***, make the following changes.

In ***[ml2]*** section, make the following changes.
type_drivers = local,flat,vlan,gre,vxlan
tenant_network_types = vlan
mechanism_drivers = openvswitch,linuxbridge
WARNING: Once we configure the ML2 plugin, disabling a network type driver and re-enabling it later can lead to database inconsistency.
NOTE:
local - This is the simplest networking mode. It can only be realized on a single host. It is only used in proof-of-concept or development environments, because a typical OpenStack environment will have multiple hosts. Here each instance receives a fixed IP from the pool. All instances are attached to the same bridge(br100) by default. The bridge must be configured manually. The networking configuration is injected into the instance before it is booted. Also there is no floating IP feature in this mode.

flat - An example of flat network is traditional L2 ethernet network. Any servers attached to this network are able to see the same broadcast traffic and can contact each other without requiring a router. These are often used to attach Nova servers to an existing L2 network called provider network. This networking mode does not provide any segmentation options.

vlan - This is a network that uses VLANs for segmentation. It is the default networking mode for Nova. In case of Neutron, when we create a new network it will assign a VLAN ID from the range we have configured in our Neutron configuration. Using vlan networks requires that any switches in our environment are configured to trunk the corresponding VLANs.

gre - GRE stands for Generic Routing Encapsulation. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 network traffic in IP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using GRE as tenant networks in Neutron avoids the need for a network interface connected to a switch configured to trunk a range of VLANs.

vxlan - vxlan stands for Virtual eXtensible LAN. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 Ethernet frames over layer 4 UDP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using vxlan as tenant networks in Neutron avoids the need for a network interface connected to a switch configured to trunk a range of VLANs.

Enable security groups, IPset, and OVS iptables firewall driver.

In ***[securitygroup]*** section, make the following changes.
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

1c6) Reconfigure Compute

The distribution packages by default configure Compute to use legacy networking.

Let us reconfigure Compute to use Neutron Networking.
Perform the following operations in Controller Node.

In configuration file ***/etc/nova/nova.conf***, make the following changes.

Configure the APIs and drivers.

In ***[DEFAULT]*** section, make the following changes.

```
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

NOTE: By default, Compute uses an internal firewall service. Since Networking includes a firewall service, we disable the Compute firewall service by using the nova.virt.firewall.NoopFirewallDriver firewall driver.

Configure access parameters for Neutron.

In ***[neutron]*** section, make the following changes.

```
url = http://horizon-network:9696
auth_strategy = keystone
admin_auth_url = http://horizon-network:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = 101010
```

SYNTAX:

```
url = http://CONTROLLER_HOSTNAME:9696
auth_strategy = keystone
admin_auth_url = http://CONTROLLER_HOSTNAME:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = NEUTRON_PASS
```

1d) Finalize installation

Let us finalize the installation.

Populate the Neutron database.
root@horizon-network:~# su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade juno" neutron
NOTE: Database population occurs later for Networking because the script requires complete server and plugin configuration files

Restart the Compute services.
root@horizon-network:~# service nova-api restart
root@horizon-network:~# service nova-scheduler restart
root@horizon-network:~# service nova-conductor restart

Restart the Networking service
root@horizon-network:~# service neutron-server restart

By default, the Ubuntu packages create an SQLite database.
Because this configuration uses a SQL database server, we can remove the SQLite database file.
root@horizon-network:~# rm -f /var/lib/neutron/neutron.sqlite

1e) Verify Operation

Perform the following operations in Controller Node.

Source the admin credentials to gain access to admin-only CLI commands.
root@horizon-network:~# source admin-openrc.sh

List the loaded extensions to verify successful launch of the Neutron server process.
root@horizon-network:~# neutron ext-list

EXPECTED OUTPUT:

```
+---------------------+-------------------------------------------+
| alias               | name                                      |
+---------------------+-------------------------------------------+
| security-group      | security-group                            |
| l3_agent_scheduler  | L3 Agent Scheduler                        |
| ext-gw-mode         | Neutron L3 Configurable external gateway mode |
| binding             | Port Binding                              |
| provider            | Provider Network                          |
| agent               | agent                                     |
| quotas              | Quota management support                  |
| dhcp_agent_scheduler | DHCP Agent Scheduler                     |
| l3-ha               | HA Router extension                       |
| multi-provider      | Multi Provider Network                    |
| external-net        | Neutron external network                  |
| router              | Neutron L3 Router                         |
| allowed-address-pairs | Allowed Address Pairs                   |
| extraroute          | Neutron Extra Route                       |
| extra_dhcp_opt      | Neutron Extra DHCP opts                   |
| dvr                 | Distributed Virtual Router                |
+---------------------+-------------------------------------------+
```

1f) Check for errors

Finally, let us check for any errors in our deployment of OpenStack Networking service.

Perform the following operations in Controller node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.

```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

2) Configuring Network Node

The Network node primarily handles internal and external routing and DHCP services for virtual networks.

Perform the following operations in Controller(Network) Node.

2a) Change Network kernel settings

Before we install and configure OpenStack Networking, we must configure certain networking kernel parameters.

In configuration file ***/etc/sysctl.conf***, make the following changes.

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

Load the sysctl settings from ***/etc/sysctl.conf***

```
root@horizon-network:~# sysctl -p
```

2b) Install required packages

Install the necesary packages for configuring Controller's(Network) Networking components.

```
root@horizon-network:~# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent neutron-l3-agent neutron-dhcp-agent
```

NOTE: Of this, the package neutron-plugin-ml2 would be already installed in Controller(Network) Node.

2c) Configure Networking server components

Now let us configure Networking server components. This involves the following ten steps

- **2c1) Configure Database access**
- **2c2) Configure Message broker access**
- **2c3) Configure Authentication mechanism**
- **2c4) Configure Plugins**
- **2c5) Configure L3 agent**
- **2c6) Configure DHCP agent**
- **2c7) Configure Metadata agent**
- **2c8) Reconfigure Compute**
- **2c9) Configure Network interfaces**
- **2c10) Configure Bridges**

In configuration file ***/etc/neutron/neutron.conf***, make the following changes.

2c1) Configure Database access
Let us disable direct access to the database.

In ***[database]*** section, comment out any connection options. This is because Network node does not directly access the database.

2c2) Configure Message broker access
Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
SYNTAX:
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS

2c3) Configure Authentication mechanism
Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.
auth_strategy = keystone

In ***[keystone_authtoken]*** section, make the following changes.
auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = neutron
admin_password = 101010
SYNTAX:
auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.
verbose = True

2c4) Configure Plugins
Let us enable the ML2 or Modular Layer 2 plugin, Router service plugin, and Overlapping IP addresses.

In ***[DEFAULT]*** section, make the following changes.
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True

Now let us configure the ML2 plugin.
The ML2 plugin uses the OVS or Open vSwitch mechanism/agent to build the virtual networking framework for instances.

In configuration file ***/etc/neutron/plugins/ml2/ml2_conf.ini***, make the following changes.

In ***[ml2]*** section, make the following changes.
type_drivers = local,flat,vlan,gre,vxlan
tenant_network_types = vlan
mechanism_drivers = openvswitch,linuxbridge
WARNING: Once we configure the ML2 plugin, disabling a network type driver and re-enabling it later can lead to database inconsistency.
NOTE:
local - This is the simplest networking mode. It can only be realized on a single host. It is only used in proof-of-concept or development environments, because a typical OpenStack environment will have multiple hosts. Here each instance receives a fixed IP from the pool. All instances are attached to the same bridge(br100) by default. The bridge must be configured manually. The networking configuration is injected into the instance before it is booted. Also there is no floating IP feature in this mode.
flat - An example of flat network is traditional L2 ethernet network. Any servers attached to this network are able to see the same broadcast traffic and can contact each other without requiring a router. These are often used to attach Nova servers to an existing L2 network called provider network. This networking mode does not provide any segmentation options.
vlan - This is a network that uses VLANs for segmentation. It is the default networking mode for Nova. In case of Neutron, when we create a new network it will assign a VLAN ID from the range we have configured in our Neutron configuration. Using vlan networks requires that any switches in our environment are configured to trunk the corresponding VLANs.
gre - GRE stands for Generic Routing Encapsulation. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 network traffic in IP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using GRE as tenant networks in Neutron avoids the need for a network interface connected to a switch configured to trunk a range of VLANs.

vxlan - vxlan stands for Virtual eXtensible LAN. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 Ethernet frames over layer 4 UDP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using vxlan as tenant networks in Neutron avoids the need for a network interface connected to a switch configured to trunk a range of VLANs.

Configure range of VLAN IDs available for allocation to tenant networks.
In ***[ml2_type_vlan]*** section, make the following changes.
network_vlan_ranges = physnet2:1000:2999

Enable security groups, IPset, and OVS iptables firewall driver.
In ***[securitygroup]*** section, make the following changes.
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

Configure the IP address of the instance tunnels network interface of our compute node(which is currently BRIDGE1_IP or 192.168.32.1), network type of tenant, ovs integration bridge, range of VLAN network, mapping of external flat provider network to bridge, and tunnel types.

In ***[ovs]*** section, make the following changes.
local_ip = 192.168.32.1
tenant_network_type = vlan
integration_bridge = br-int
network_vlan_ranges = physnet2:1000:2999
bridge_mappings = physnet2:br-eth2
tunnel_types = vlan
SYNTAX:
local_ip = INSTANCE_TUNNELS_INTERFACE_IP
tenant_network_type = vlan
integration_bridge = br-int
network_vlan_ranges = physnet2:1000:2999
bridge_mappings = physnet2:br-eth2
tunnel_types = vlan

2c5) Configure L3 agent

The L3 or Layer 3 agent provides routing services for virtual networks.

Let us configure the L3 agent now.

In configuration file ***/etc/neutron/l3_agent.ini***, make the following changes.

Configure the driver, enable network namespaces, configure the external network bridge, and enable deletion of defunct router namespaces.
In ***[DEFAULT]*** section, make the following changes.

```
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True
external_network_bridge = br-ex
router_delete_namespaces = True
```

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.

```
verbose = True
```

2c6) Configure DHCP agent

The DHCP agent provides DHCP services for virtual networks.

Lets us configure the DHCP agent.
In configuration file /etc/neutron/dhcp_agent.ini, make the following changes.

Configure the drivers, enable namespaces and enable deletion of defunct DHCP namespaces.
In ***[DEFAULT]*** section, make the following changes.

```
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
use_namespaces = True
dhcp_delete_namespaces = True
```

Enable verbose logging.
In ***[DEFAULT]*** section, make the following changes.

```
verbose = True
```

2c7) Configure Metadata agent

The metadata agent provides configuration information such as credentials to instances.

Let us configure the metadata agent.

In configuration file ***/etc/neutron/metadata_agent.ini***, make the following changes.

Configure access parameters for metadata agent
In ***[DEFAULT]*** section, make the following changes.

```
auth_url = http://horizon-network:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = 101010
```

SYNTAX:

```
auth_url = http://CONTROLLER_HOSTNAME:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

Configure the metadata host.
In ***[DEFAULT]*** section, make the following changes.

```
nova_metadata_ip = 10.3.0.1
```

SYNTAX:

```
nova_metadata_ip = CONTROLLER_IP
```

Configure the metadata proxy shared secret.
In ***[DEFAULT]*** section, make the following changes.

```
metadata_proxy_shared_secret = d56664e8cf0cbfcac7c1
```

SYNTAX:

```
metadata_proxy_shared_secret = METADATA_SECRET
```

Enable verbose logging.
In ***[DEFAULT]*** section, make the following changes.

```
verbose = True
```

2c8) Reconfigure Compute

Let us reconfigure Compute to use Neutron Networking. Perform the following operations in Controller node.

In configuration file ***/etc/nova/nova.conf***, make the following changes.

Enable the metadata proxy and configure the secret.
In ***[neutron]*** section, make the following changes.

```
service_metadata_proxy = True
metadata_proxy_shared_secret = d56664e8cf0cbfcac7c1
```

SYNTAX:

```
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Restart the Compute API service in Controller node.

```
root@horizon-network:~# service nova-api restart
```

2c9) Configure Network interfaces

In the Controller Node, make the following changes in configuration file ***/etc/network/interfaces***

```
auto p2p1
iface p2p1 inet manual
        up ip address add 0/0 dev $IFACE
        up ip link set $IFACE up
        down ip link set $IFACE down

auto br-ex
iface br-ex inet static
        address 10.3.0.1
        netmask 255.0.0.0
        network 10.0.0.0
        broadcast 10.255.255.255
        gateway 10.1.0.2
        dns-nameservers 10.1.0.10
        dns-search office.brookwin.com

auto p3p1
iface p3p1 inet manual
        up ip address add 0/0 dev $IFACE
        up ip link set $IFACE up
        down ip link set $IFACE down
```

```
auto br-eth2
iface br-eth2 inet static
        address 192.168.32.1
        netmask 255.255.255.0
        network 192.168.32.0
        broadcast 192.168.32.255
```

SYNTAX:

```
auto p2p1
iface p2p1 inet manual
        up ip address add 0/0 dev $IFACE
        up ip link set $IFACE up
        down ip link set $IFACE down

auto br-ex
iface br-ex inet static
        address CONTROLLER_IP
        netmask LOCAL_NETMASK_IP
        network LOCAL_NETWORK_IP
        broadcast LOCAL_BROADCAST_IP
        gateway LOCAL_GATEWAY_IP
        dns-nameservers LOCAL_DNS_NAMESERVER_IP
        dns-search LOCAL_DNS_DOMAIN

auto p3p1
iface p3p1 inet manual
        up ip address add 0/0 dev $IFACE
        up ip link set $IFACE up
        down ip link set $IFACE down

auto br-eth2
iface br-eth2 inet static
        address BRIDGE1_IP
        netmask BRIDGE_NETMASK_IP
        network BRIDGE_NETWORK_IP
        broadcast BRIDGE_BROADCAST_IP
```

Reboot the system.

If we are connected from a remote system, we are gonna lose network connectivity after this step.

When the system is up and running, perform the following operations from a local terminal in the Controller node.

2c10) Configure Bridges

The OVS service provides the underlying virtual networking framework for instances.

The integration bridge br-int handles internal instance network traffic within OVS. The external bridge br-ex handles external instance network traffic within OVS. The external bridge requires a port on the physical external network interface to provide instances with external network access. In essence, this port connects the virtual and physical external networks in our environment.

Restart the OVS service.

```
root@horizon-network:~# service openvswitch-switch restart
```

Add the external bridge.

```
root@horizon-network:~# ovs-vsctl add-br br-ex
```

Add a port to the external bridge that connects to the physical external network interface.

```
root@horizon-network:~# ovs-vsctl add-port br-ex p2p1
```

Add the second external bridge.

```
root@horizon-network:~# ovs-vsctl add-br br-eth2
```

Add a port to the second external bridge that connects to the second physical external network interface.

```
root@horizon-network:~# ovs-vsctl add-port br-eth2 p3p1
```

NOTE: (Optional)

Depending on our network interface driver, we may need to disable GRO(Generic Receive Offload) to achieve suitable throughput between our instances and the external network. To temporarily disable GRO on the external network interface while testing our environment, execute the following command.

```
# ethtool -K p2p1 gro off
```

2d) Finalize installation

Finally let us restart the Networking services.

```
root@horizon-network:~# service neutron-plugin-openvswitch-agent restart
root@horizon-network:~# service neutron-l3-agent restart
root@horizon-network:~# service neutron-dhcp-agent restart
root@horizon-network:~# service neutron-metadata-agent restart
```

Now we should get network connectivity from outside.

But reboot the system again, so that Neutron agents start working properly.

2e) Verify operation

Perform the following operations on Controller node.

Source the admin credentials to gain access to admin-only CLI commands.
```
root@horizon-network:~# source admin-openrc.sh
```

List agents to verify successful launch of the neutron agents.
```
root@horizon-network:~# neutron agent-list
```
EXPECTED OUTPUT:
```
+--------------------------------------+--------------------+----------------+-------+----------------+---------------------------+
| id                                   | agent_type         | host           | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+----------------+-------+----------------+---------------------------+
| 73eda377-ada1-46b6-9312-9868bdd78683 | Open vSwitch agent | horizon-network | :-)   | True           | neutron-openvswitch-agent |
| c7f8964f-ac00-492f-a9a5-de10671f04ba | Metadata agent     | horizon-network | :-)   | True           | neutron-metadata-agent    |
| d235830c-b2e7-4da3-8e5a-4cda071b2c9d | L3 agent           | horizon-network | :-)   | True           | neutron-l3-agent          |
| d3c19cb1-7e7b-4247-b07b-7e27cb8915ca | DHCP agent         | horizon-network | :-)   | True           | neutron-dhcp-agent        |
+--------------------------------------+--------------------+----------------+-------+----------------+---------------------------+
```

2f) Check for errors

Finally, let us check for any errors in our deployment of OpenStack Networking service.

Perform the following operations in Controller node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.
```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.
```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

3) Configuring Compute Node

Compute node handles connectivity and security groups for instances.

Perform the following operations in Compute node.

3a) Change Network kernel settings

Before we install and configure OpenStack Networking, we must configure certain networking kernel parameters.

In configuration file ***/etc/sysctl.conf***, make the following changes.

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

Load the sysctl settings from ***/etc/sysctl.conf***

```
root@horizon-network:~# sysctl -p
```

3b) Install required packages

Install the required packages for OpenStack Networking service in compute node.

```
root@nova:~# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent
```

3c) Configure Networking server component

Now let us configure Networking service components. This involves the following seven steps.

3c1) Configure Database access
3c2) Configure Message broker access
3c3) Configure Authentication mechanism
3c4) Configure Plugins
3c5) Reconfigure Compute
3c6) Configure Network interfaces
3c7) Configure Bridges

In configuration file ***/etc/neutron/neutron.conf***, make the following changes.

3c1) Configure Database access
Let us disable direct access to the database.

In ***[database]*** section, comment out any connection options.
This is because Network node does not directly access the database.

3c2) Configure Message broker access
Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
SYNTAX:
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS

3c3) Configure Authentication mechanism
Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.
auth_strategy = keystone

In ***[keystone_authtoken]*** section, make the following changes.
auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = neutron
admin_password = 101010
SYNTAX:
auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

Enable verbose logging.
In ***[DEFAULT]*** section, make the following changes.
verbose = True

3c4) Configure Plugins
Let us enable the ML2 or Modular Layer 2 plugin, Router service plugin, and Overlapping IP addresses.

In ***[DEFAULT]*** section, make the following changes.
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
NOTE: Option ***core_plugin = ml2*** would already be enabled.

Now let us configure the ML2 plugin.
The ML2 plugin uses the OVS or Open vSwitch mechanism/agent to build the virtual networking framework for instances.

In configuration file ***/etc/neutron/plugins/ml2/ml2_conf.ini***, make the following changes.

In ***[ml2]*** section, make the following changes.
type_drivers = local,flat,vlan,gre,vxlan
tenant_network_types = vlan
mechanism_drivers = openvswitch,linuxbridge
WARNING: Once we configure the ML2 plugin, disabling a network type driver and re-enabling it later can lead to database inconsistency.
NOTE:
local - This is the simplest networking mode. It can only be realized on a single host. It is only used in proof-of-concept or development environments, because a typical OpenStack environment will have multiple hosts. Here each instance receives a fixed IP from the pool. All instances are attached to the same bridge(br100) by default. The bridge must be configured manually. The networking configuration is injected into the instance before it is booted. Also there is no floating IP feature in this mode.
flat - An example of flat network is traditional L2 ethernet network. Any servers attached to this network are able to see the same broadcast traffic and can contact each other without requiring a router. These are often used to attach Nova servers to an existing L2 network called provider network. This networking mode does not provide any segmentation options.
vlan - This is a network that uses VLANs for segmentation. It is the default networking mode for Nova. In case of Neutron, when we create a new network it will assign a VLAN ID from the range we have configured in our Neutron configuration. Using vlan networks requires that any switches in our environment are configured to trunk the corresponding VLANs.
gre - GRE stands for Generic Routing Encapsulation. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 network traffic in IP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using GRE as tenant networks in Neutron avoids the need for a network interface

connected to a switch configured to trunk a range of VLANs.
vxlan - vxlan stands for Virtual eXtensible LAN. This is an overlay network that works by tunneling. Each network we create receives a unique tunnel id. This encapsulates layer 2 Ethernet frames over layer 4 UDP packets that are routed between Compute and Networking nodes using the hosts' network connectivity and routing tables. Using vxlan as tenant networks in Neutron avoids the need for a network interface connected to a switch configured to trunk a range of VLANs.

Configure range of VLAN IDs available for allocation to tenant networks.
In ***[ml2_type_vlan]*** section, make the following changes.
network_vlan_ranges = physnet2:1000:2999

Enable security groups, IPset, and OVS iptables firewall driver.
In ***[securitygroup]*** section, make the following changes.
enable_security_group = True
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

Configure the IP address of the instance tunnels network interface of our compute node(which is currently BRIDGE2_IP or 192.168.32.2), network type of tenant, ovs integration bridge, range of VLAN network, and mapping of external flat provider network to bridge.

In ***[ovs]*** section, make the following changes.
local_ip = 192.168.32.2
tenant_network_type = vlan
integration_bridge = br-int
network_vlan_ranges = physnet2:1000:2999
bridge_mappings = physnet2:br-eth1
SYNTAX:
local_ip = INSTANCE_TUNNELS_INTERFACE_IP
tenant_network_type = vlan
integration_bridge = br-int
network_vlan_ranges = physnet2:1000:2999
bridge_mappings = physnet2:br-eth1

Restart the OVS service.
root@nova:~# service openvswitch-switch restart

3c5) Reconfigure Compute

Let us reconfigure Compute to use Neutron Networking. Perform the following operations in Compute node.

In configuration file ***/etc/nova/nova.conf***, make the following changes.

Configure the APIs and drivers.
In ***[DEFAULT]*** section, make the following changes.

```
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

NOTE: By default, Compute uses an internal firewall service. Since Networking includes a firewall service, we must disable the Compute firewall service by using the nova.virt.firewall.NoopFirewallDriver firewall driver.

Configure access parameters.
In ***[neutron]*** section, make the following changes.

```
url = http://horizon-network:9696
auth_strategy = keystone
admin_auth_url = http://horizon-network:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = 101010
```

SYNTAX:

```
url = http://CONTROLLER_HOSTNAME:9696
auth_strategy = keystone
admin_auth_url = http://CONTROLLER_HOSTNAME:35357/v2.0
admin_tenant_name = service
admin_username = neutron
admin_password = NEUTRON_PASS
```

Restart the Compute service.

```
root@nova:~# service nova-compute restart
```

Restart the OVS agent service.

```
root@nova:~# service neutron-plugin-openvswitch-agent restart
```

3c6) Configure Network interfaces

In the Compute Node, make following changes in configuration file ***/etc/network/interfaces***

```
auto eth0
iface eth0 inet static
        address 10.3.0.2
        netmask 255.0.0.0
        network 10.0.0.0
        broadcast 10.255.255.255
        gateway 10.1.0.2
        dns-nameservers 10.1.0.10
        dns-search office.brookwin.com

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

auto br-eth1
iface br-eth1 inet static
        address 192.168.32.2
        netmask 255.255.255.0
        network 192.168.32.0
        broadcast 192.168.32.255
```

SYNTAX:

```
auto eth0
iface eth0 inet static
        address NOVA_IP
        netmask LOCAL_NETMASK_IP
        network LOCAL_NETWORK_IP
        broadcast LOCAL_BROADCAST_IP
        gateway LOCAL_GATEWAY_IP
        dns-nameservers LOCAL_DNS_NAMESERVER_IP
        dns-search LOCAL_DNS_DOMAIN

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

auto br-eth1
iface br-eth1 inet static
        address BRIDGE2_IP
        netmask BRIDGE_NETMASK_IP
        network BRIDGE_NETWORK_IP
        broadcast BRIDGE_BROADCAST_IP
```

Reboot the system.
If we are connected from a remote system, we are gonna lose network connectivity after this step.

When the system is up and running, perform the following operations from a local terminal in the Controller node.

3c7) Configure Bridges

The OVS service provides the underlying virtual networking framework for instances.

The integration bridge br-int handles internal instance network traffic within OVS. The external bridge br-eth1 handles external instance network traffic within OVS. The external bridge requires a port on the physical external network interface to provide instances with external network access. In essence, this port connects the virtual and physical external networks in our environment.

Add the external bridge.

```
root@nova:~# ovs-vsctl add-br br-eth1
```

Add a port to the external bridge that connects to the physical external network interface.

```
root@nova:~# ovs-vsctl add-port br-eth1 eth1
```

3d) Finalize installation

Finally let us restart the necessary services.

Restart the Compute service.

```
root@nova:~# service nova-compute restart
```

Restart the OVS agent service.

```
root@nova:~# service neutron-plugin-openvswitch-agent restart
```

3e) Verify operation

Perform the following operations on Controller node.

Source the admin credentials to gain access to admin-only CLI commands.
root@horizon-network:~# source admin-openrc.sh

List agents to verify successful launch of the neutron agents.
root@horizon-network:~# neutron agent-list
EXPECTED OUTPUT:

```
+--------------------------------------+--------------------+-----------------+-------+----------------+---------------------------+
| id                                   | agent_type         | host            | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+-----------------+-------+----------------+---------------------------+
| 038aa747-a9ea-4e08-9ccb-368bccca794d | Open vSwitch agent | horizon-network | :-)   | True           | neutron-openvswitch-agent |
| 18b04cde-af21-4f2d-8943-1abd8e28f072 | Open vSwitch agent | nova            | :-)   | True           | neutron-openvswitch-agent |
| 815ab143-075f-414e-8a7f-a8e17bec283b | DHCP agent         | horizon-network | :-)   | True           | neutron-dhcp-agent        |
| b154255c-d9c6-41ec-a4f4-8471c2e65c47 | Metadata agent     | horizon-network | :-)   | True           | neutron-metadata-agent    |
| dbd6d7cd-37d0-4eca-86bc-f27ed28b3c10 | L3 agent           | horizon-network | :-)   | True           | neutron-l3-agent          |
+--------------------------------------+--------------------+-----------------+-------+----------------+---------------------------+
```

3f) Check for errors

Finally, let us check for any errors in our deployment of OpenStack Networking service.

Perform the following operations in Compute node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log/***, and reboot the server.
root@nova:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot

After the server comes up, check for any critical messages or errors or warnings.

```
root@nova:~# grep crit -irl /var/log
root@nova:~# grep err -irl /var/log
root@nova:~# grep warn -irl /var/log
```

4) Installed Services

Controller

/etc/init.d/ipvsadm
/etc/init.d/keepalived
/etc/init.d/openvswitch-switch [Not set to start on any Runelevel; but it is already enabled on startup as an Upstart job]
/etc/init.d/radvd

/etc/init/neutron-dhcp-agent.conf
/etc/init/neutron-l3-agent.conf
/etc/init/neutron-metadata-agent.conf
/etc/init/neutron-ovs-cleanup.conf
/etc/init/neutron-plugin-openvswitch-agent.conf
/etc/init/neutron-server.conf
/etc/init/openvswitch-force-reload-kmod.conf [Not set to start on any Runelevel]
/etc/init/openvswitch-switch.conf

Nova

/etc/init.d/openvswitch-switch [Not set to start on any Runelevel; but it is already enabled on startup as an Upstart job]

/etc/init/neutron-ovs-cleanup.conf
/etc/init/neutron-plugin-openvswitch-agent.conf
/etc/init/openvswitch-force-reload-kmod.conf [Not set to start on any Runelevel]
/etc/init/openvswitch-switch.conf

Dashboard

This section describes how to install and configure OpenStack dashboard.

This deployment uses an Apache HTTP Server.

The OpenStack dashboard is also known as the Horizon.
It is a Web interface that enables cloud administrators and users to manage various OpenStack resources and services. It also provides the following.

- Web-based interactions with the OpenStack Compute cloud controller through the OpenStack APIs.
- Set of core classes and reusable templates and tools.
- Ability to Customize the brand of the dashboard.

NOTE: Dashboard relies on functional core services including Identity, Image Service, Compute, and either Networking (neutron) or legacy networking (nova-network). Environments with stand-alone services such as Object Storage cannot use the dashboard.

Perform the following operations on the controller node.

Install the necessary packages.

```
root@horizon-network:~# apt-get install openstack-dashboard apache2 libapache2-mod-wsgi memcached python-memcache
```

IMPORTANT: Ubuntu installs the ***openstack-dashboard-ubuntu-theme*** package as a dependency. Some users reported issues with this theme in previous releases. If you encounter issues, remove this package to restore the original OpenStack theme.

Let us configure the Dashboard.

In the configuration file ***/etc/openstack-dashboard/local_settings.py***, make the following changes.

Configure the Dashboard to use OpenStack services. Make the following changes.

```
OPENSTACK_HOST = "horizon-network"
```

Allow all hosts to access the dashboard. Make the following changes.

```
ALLOWED_HOSTS = ['*']
```

Configure the memcached session storage service. Make the following changes.

```
CACHES = {
  'default': {
     'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
     'LOCATION': '127.0.0.1:11211',
  }
}
```

NOTE:

- ➠ This entry is already in the configuration file.
- ➠ Comment out any other session storage configuration.

Configure the time zone.

A list of time zones is available at http://en.wikipedia.org/wiki/List_of_tz_database_time_zones

```
TIME_ZONE = "Asia/Calcutta"
```

To finalize installation, let us restart the web server and session storage services.

```
# service apache2 restart
# service memcached restart
```

Now we can verify the operation of Dashboard by accessing the URL http://10.3.0.1/horizon
Authentication can be done using admin or demo user credentials.

Done.

Block Storage Service

The OpenStack Block Storage service (cinder) provides block storage devices to instances using various backends.

This service provides an infrastructure for managing volumes, and interacts with OpenStack Compute to provide volumes for instances. The service also enables management of volume snapshots, and volume types.

The Block Storage service consists of the following components.

➠ cinder-api
This runs on the Controller Node.
It accepts API requests, and routes them to the cinder-volume for action.

➠ cinder-volume
This runs on the Storage Node(s).
It interacts directly with the Block Storage service, and processes such as the cinder-scheduler. It also interacts with these processes through a message queue. The cinder-volume service responds to read and write requests sent to the Block Storage service to maintain state. It can interact with a variety of storage providers through a driver architecture.

➠ cinder-scheduler daemon
This runs on the Controller Node.
It selects the optimal storage provider node on which to create the volume. A similar component to the nova-scheduler.

➠ Messaging queue
This runs on the Controller Node.
It routes information between the Block Storage processes.

This section describes how to install and configure the Block Storage service.

1) Configuring Controller Node

- 1a) Prerequisites
  - 1a1) Create Database
  - 1a2) Create Block storage service entity
  - 1a3) Create Block storage service API endpoints
- 1b) Install required packages
- 1c) Configure Block storage components
  - 1c1) Configure Database access
  - 1c2) Configure Message broker access
  - 1c3) Configure Authentication mechanism
  - 1c4) Configure Management interface
- 1d) Finalize installation

2) Configuring Storage Node(s)

- 2a) Deploy LVM
- 2b) Install required packages
- 2c) Configure Block storage components
  - 2c1) Configure Database access
  - 2c2) Configure Message broker access
  - 2c3) Configure Authentication mechanism
  - 2c4) Configure Management interface
  - 2c5) Configure Image service location
- 2d) Finalize installation

3) Verify operation

4) Check for errors

5) Installed services

1) Configuring Controller Node

Perform the following operations in Controller Node.

1a) Prerequisites

Before we install and configure the Block Storage service, we must do the following.

1a1) Create Database
1a2) Create Block storage service entity
1a3) Create Block storage service API endpoints

1a1) Create Database

Performing the following steps to create cinder database.

Use the mysql client to connect to the database server as root user.

```
root@horizon-network:~# mysql -u root -p
```

Create the cinder database.

```
MariaDB [(none)]> CREATE DATABASE cinder;
```

Grant proper access to the cinder database.

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY '101010';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY '101010';
```

SYNTAX:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'CINDER_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'CINDER_DBPASS';
```

1a2) Create Block storage service entity

Source the client environment script for admin user.

```
root@horizon-network:~# source admin-openrc.sh
```

Create the cinder user.

```
root@horizon-network:~# keystone user-create --name cinder --pass 101010
```

SYNTAX:

```
root@horizon-network:~# keystone user-create --name cinder --pass CINDER_PASS
```

EXPECTED OUTPUT:

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 8ec7778b0ba44f81a233259cfa2cd9ce |
|   name   |              cinder              |
| username |              cinder              |
+----------+----------------------------------+
```

Add admin role to the cinder user.

```
root@horizon-network:~# keystone user-role-add --user cinder --tenant service --role admin
```

The Block Storage service requires two different service entities for supporting API versions 1 and 2. Let us create both those cinder service entities.

```
root@horizon-network:~# keystone service-create --name cinder --type volume --description "OpenStack Block Storage"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Block Storage      |
|   enabled   |               True               |
|      id     | a44f94a070044d03859439b1e637e786 |
|     name    |              cinder              |
|     type    |              volume              |
+-------------+----------------------------------+
```

```
root@horizon-network:~# keystone service-create --name cinderv2 --type volumev2 --description "OpenStack Block Storage"
```

EXPECTED OUTPUT:

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Block Storage      |
|   enabled   |               True               |
|      id     | 270846d0061e4dd58b037466390059f0 |
|     name    |             cinderv2             |
|     type    |             volumev2             |
+-------------+----------------------------------+
```

1a3) Create Block storage service API endpoints

Create Block storage service API endpoints for API version 1.

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ volume / {print $2}') --publicurl http://horizon-network:8776/v1/%\(tenant_id\)s --internalurl http://horizon-network:8776/v1/%\(tenant_id\)s --adminurl http://horizon-network:8776/v1/%\(tenant_id\)s --region regionOne
```

SYNTAX:

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ volume / {print $2}') --publicurl http://CONTROLLER_HOSTNAME:8776/v1/%\(tenant_id\)s --internalurl http://CONTROLLER_HOSTNAME:8776/v1/%\(tenant_id\)s --adminurl http://CONTROLLER_HOSTNAME:8776/v1/%\(tenant_id\)s --region regionOne
```

EXPECTED OUTPUT:

```
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://horizon-network:8776/v1/%(tenant_id)s |
|      id     |      66867a3e015b4ad2a18c49739d829538       |
| internalurl | http://horizon-network:8776/v1/%(tenant_id)s |
|  publicurl  | http://horizon-network:8776/v1/%(tenant_id)s |
|    region   |              regionOne                  |
|  service_id |      a44f94a070044d03859439b1e637e786       |
+-------------+-----------------------------------------+
```

Create Block storage service API endpoints for API version 2.

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ volumev2 / {print $2}') --publicurl http://horizon-network:8776/v2/%\(tenant_id\)s --internalurl http://horizon-network:8776/v2/%\(tenant_id\)s --adminurl http://horizon-network:8776/v2/%\(tenant_id\)s --region regionOne
```

SYNTAX:

```
root@horizon-network:~# keystone endpoint-create --service-id $(keystone service-list | awk '/ volumev2 / {print $2}') --publicurl http://CONTROLLER_HOSTNAME:8776/v2/%\(tenant_id\)s --internalurl http://CONTROLLER_HOSTNAME:8776/v2/%\(tenant_id\)s --adminurl http://CONTROLLER_HOSTNAME:8776/v2/%\(tenant_id\)s --region regionOne
```

EXPECTED OUTPUT:

```
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://horizon-network:8776/v2/%(tenant_id)s |
|      id     |      42b48ef791b8483d8916ded3209d6c0f       |
| internalurl | http://horizon-network:8776/v2/%(tenant_id)s |
|  publicurl  | http://horizon-network:8776/v2/%(tenant_id)s |
|    region   |              regionOne                  |
|  service_id |      270846d0061e4dd58b037466390059f0       |
+-------------+-----------------------------------------+
```

1b) Install required packages

Install the required packages for OpenStack Block storage.

```
root@horizon-network:~# apt-get install cinder-api cinder-scheduler python-cinderclient
```

1c) Configure Block storage components

Now let us configure Block storage components. This involves the following four steps.

1c1) Configure Database access
1c2) Configure Message broker access
1c3) Configure Authentication mechanism
1c4) Configure Management interface

In configuration file ***/etc/cinder/cinder.conf***, make the following changes.

1c1) Configure Database access

Let us specify the MySQL connection string.

In ***[database]*** section, make the following changes.

```
connection = mysql://cinder:101010@horizon-network/cinder
```

SYNTAX:

```
connection = mysql://cinder:CINDER_DBPASS@CONTROLLER_HOSTNAME/cinder
```

1c2) Configure Message broker access

Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.

```
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
```

SYNTAX:

```
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS
```

1c3) Configure Authentication mechanism

Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.
auth_strategy = keystone
NOTE: This value may already exist.

In ***[keystone_authtoken]*** section, make the following changes.
auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = cinder
admin_password = 101010
SYNTAX:
auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS

IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

1c4) Configure Management interface

Configure Block storage service to use the management interface IP address of controller node.

In ***[DEFAULT]*** section, make the following changes.
my_ip = 10.3.0.1
SYNTAX:
my_ip = CONTROLLER_IP

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.
verbose = True
NOTE: This value may already exist.

1d) Finalize installation

Let us finalize the installation.

Populate the Block Storage database.
```
root@horizon-network:~# su -s /bin/sh -c "cinder-manage db sync" cinder
```

Restart the Block Storage services.
```
root@horizon-network:~# service cinder-scheduler restart
root@horizon-network:~# service cinder-api restart
```

By default, the Ubuntu packages create an SQLite database.
Because this configuration uses a SQL database server, we can remove the SQLite database file.
```
root@horizon-network:~# rm -f /var/lib/cinder/cinder.sqlite
```

2) Configuring Storage Node(s)

This configuration references two storage nodes cinder1 and cinder2, each with a local block storage device /dev/sda that contains a suitable partition table with a partition /dev/sda3 occupying the entire device. Follow the below instructions in both cinder1 and cinder2.

2a) Deploy LVM

The Block Storage service will provision logical volumes on this device using the LVM driver and provides them to instances via iSCSI transport.

Let us install the LVM packages.
```
root@cinder1:~# apt-get install lvm2
```

Create the LVM physical volume /dev/sda3
```
root@cinder1:~# pvcreate /dev/sda3
```

Create the LVM volume group cinder-volumes
```
root@cinder1:~# vgcreate cinder-volumes /dev/sda3
```

Only instances can access the Block Storage volumes. However, the underlying OS manages the devices associated with the volumes. By default, the LVM volume scanning tool scans the /dev directory for block storage devices that contain volumes. If tenants use LVM on their volumes, the scanning tool will detect these volumes and attempts to cache them which can cause problems with both the underlying OS and tenant volumes.

So we must reconfigure LVM to scan only the devices that contain the cinder-volume volume group.

In configuration file ***/etc/lvm/lvm.conf***, make the following changes.

Configure LVM to accept the /dev/sda device and rejects all other device.

In the devices section, add a filter that accepts the /dev/sda device and rejects all other devices.
```
filter = [ "a/sda/", "r/.*/"]
```
NOTE:

- ➠ We can use the 'vgs -vvvv' command to test filters.
- ➠ If our storage nodes use LVM on the OS disk, we must also add the associated device to the filter.
- ➠ For example, if the /dev/sdb device contains the OS, the following entry must be added to the filter.

```
filter = [ "a/sdb/", "a/sda/", "r/.*/"]
```

- ➠ If other nodes such as compute nodes use LVM on the OS disk, then we must modify the filter to use only the OS disk.
- ➠ For example, if the /dev/sdb device contains the OS, the following entry must be added to the filter.

```
filter = [ "a/sdb/", "r/.*/"]
```

2b) Install required packages

Install the necessary packages for Block Storage service.
root@cinder1:~# apt-get install cinder-volume python-mysqldb

2c) Configure Block storage components

Now let us configure Block storage components. This involves the following five steps.

2c1) Configure Database access
2c2) Configure Message broker access
2c3) Configure Authentication mechanism
2c4) Configure Management interface
2c5) Configure Image service location

In configuration file ***/etc/cinder/cinder.conf***, make the following changes.

2c1) Configure Database access

Let us specify the MySQL connection string.

In ***[database]*** section, make the following changes.
connection = mysql://cinder:101010@horizon-network/cinder
SYNTAX:
connection = mysql://cinder:CINDER_DBPASS@CONTROLLER_HOSTNAME/cinder

2c2) Configure Message broker access

Let us configure access for RabbitMQ message broker.

In ***[DEFAULT]*** section, make the following changes.
rpc_backend = rabbit
rabbit_host = horizon-network
rabbit_password = 101010
SYNTAX:
rpc_backend = rabbit
rabbit_host = CONTROLLER_HOSTNAME
rabbit_password = RABBIT_PASS

2c3) Configure Authentication mechanism

Let us configure access for Identity service.

In ***[DEFAULT]*** section, make the following changes.

auth_strategy = keystone

NOTE: This value may already exist.

In ***[keystone_authtoken]*** section, make the following changes.

auth_uri = http://horizon-network:5000/v2.0
identity_uri = http://horizon-network:35357
admin_tenant_name = service
admin_user = cinder
admin_password = 101010

SYNTAX:

auth_uri = http://CONTROLLER_HOSTNAME:5000/v2.0
identity_uri = http://CONTROLLER_HOSTNAME:35357
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS

IMPORTANT: Comment out any ***auth_host***, ***auth_port***, and ***auth_protocol*** options because the ***identity_uri*** option replaces them.

2c4) Configure Management interface

Configure Block storage service to use the management interface IP address of controller node.

In ***[DEFAULT]*** section, make the following changes.

my_ip = 10.3.0.3

SYNTAX:

my_ip = CINDER1_IP

2c5) Configure Image service location

Configure the location of image service.

In ***[DEFAULT]*** section, make the following changes.

glance_host = horizon-network

SYNTAX:

glance_host = CONTROLLER_HOSTNAME

Enable verbose logging.

In ***[DEFAULT]*** section, make the following changes.
verbose = True
NOTE: This value may already exist.

2d) Finalize installation

Let us finalize the installation.

Restart the Block Storage volume service and SCSI Target Daemon service.
root@cinder1:~# service tgt restart
root@cinder1:~# service cinder-volume restart

By default, the Ubuntu packages create an SQLite database.
Because this configuration uses a SQL database server, we can remove the SQLite database file.
root@cinder1:~# rm -f /var/lib/cinder/cinder.sqlite

3) Verify operation

Now let us verify the operation of Block Storage service by creating a volume.

Perform the following operations in Controller node.

Source the admin credentials.
root@horizon-network:~# source admin-openrc.sh

List service components to verify successful launch of each process.
root@horizon-network:~# cinder service-list

EXPECTED OUTPUT:

```
+------------------+----------------+------+---------+-------+----------------------------+----------------+
|      Binary      |      Host      | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+----------------+------+---------+-------+----------------------------+----------------+
| cinder-scheduler | horizon-network | nova | enabled |   up  | 2016-01-13T07:53:06.000000 |      None      |
|  cinder-volume   |     cinder1    | nova | enabled |   up  | 2016-01-13T07:53:07.000000 |      None      |
|  cinder-volume   |     cinder2    | nova | enabled |   up  | 2016-01-13T07:53:11.000000 |      None      |
+------------------+----------------+------+---------+-------+----------------------------+----------------+
```

Source the demo tenant credentials.

```
root@horizon-network:~# source demo-openrc.sh
```

Create a 1 GB volume.

```
root@horizon-network:~# cinder create --display-name demo-volume1 1
```

EXPECTED OUTPUT:

```
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2016-01-13T07:39:02.387775      |
| display_description |                 None                 |
|     display_name    |             demo-volume1             |
|      encrypted      |                False                 |
|          id         | 6e66713a-112b-42e7-8cf5-8df59ab3f789 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

Finally, verify the creation and availability of the volume.

```
root@horizon-network:~# cinder list
```

EXPECTED OUTPUT:

```
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
| 6e66713a-112b-42e7-8cf5-8df59ab3f789 | available | demo-volume1 |  1   |     None    |  false   |             |
+--------------------------------------+-----------+--------------+------+-------------+----------+-------------+
```

4) Check for errors

Finally, let us check for any errors in our deployment of OpenStack Block storage service.

Perform the following operations in controller node.

Copy the directory /var/log/ as a backup renamed with current date and time, delete any existing log files from directory /var/log, and reboot the server.

```
root@horizon-network:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@horizon-network:~# grep crit -irl /var/log
root@horizon-network:~# grep err -irl /var/log
root@horizon-network:~# grep warn -irl /var/log
```

Perform the following operations in cinder node.

Copy the directory ***/var/log/*** as a backup renamed with current date and time, delete any existing log files from directory ***/var/log***, and reboot the server.

```
root@cinder1:~# cp -avr /var/log /var/log_bak_$(date +"%Y%m%d-%H%M%S") && find /var/log -type f | xargs rm -f && reboot
```

After the server comes up, check for any critical messages or errors or warnings.

```
root@cinder1:~# grep crit -irl /var/log
root@cinder1:~# grep err -irl /var/log
root@cinder1:~# grep warn -irl /var/log
```

Done.