**OpenStack Lab Guide**

Version 2.0 – August 2014

**OneCloud Consulting OpenStack Lab Guide**

**ATTENTION**

The information contained in this guide is for training purposes only.This guide contains information and activities that, while beneficial for purposes of training in close, non-production environment, can result in downtime or other severe consequences and therefore are not intended as a reference guide.

No part of this book may be reproduced in any form or by any means (graphic, electronic, mechanical, including photocopying, recording or taping, or storage in an electronic retrieval system) without permission of OneCloud Consulting.

OneCloud Consulting reserves the right to change the contents herein without any prior notice.

OpenStack Labs – Objectives 7

Lab1: 7

Lab2: 7

Lab3: 7

Lab4: 7

Lab5: 7

Lab6: 7

Lab7: 7

Lab8: 7

Lab9: 7

Lab10: 7

Lab11: 7

Lab Topology 8

Lab Access 8

RDP/VNC to the POD machines with credentials given by the Instructor 8

Each Student will have an AIOX node and ComputeY node installed with Ubuntu 14.04 8

Objective: 8

Basic Configuration 9

Prerequisites for OpenStack 10

**NTP Server** 10

**MySQL** 11

**Openstack Packages** 11

Messaging Queue Sever 12

Identity Service Keystone 12

Keystone Database 13

Define users, tenants and roles 13

Define services and service endpoints 15

Verify the Identity service installation 16

Image Service Installation 19

Create Database for Image Service 19

User Creation 20

Configure Image Service 20

**Update glance-api.conf and glance-registry.conf** 20

**Update glance-api-paste.ini and glance-registry-paste.ini** 21

Define services and service endpoints 22

Verify the Image Service Installation: 23

Compute Service Installation 23

Install Nova Network 24

Install Compute Controller Service packages 24

Install Compute Node packages 24

Create Database for Compute Service 24

Configure Compute Service 25

**Update nova.conf** 25

**Edit api-paste.ini** 26

Define services and service endpoints 27

Create Network 28

Launch an Instance 28

Cinder Installation on AIO Node 32

Create Database for Storage Service 33

Define services and service endpoints 33

Configure Cinder Service 34

Configure physical hard disks 35

Test Cinder Service 37

Neutron Installation 38

Database Creation 38

Define services and service endpoints 39

Configure Neutron Service 40

Basic Configuration 44

Test Neutron/GRE tunnel between aioX and computeY 49

Setting up the Nexus Switch 50

Installation of ncclient 51

Basic Configuration 51

VLAN Mode Configuration 52

VLAN Creation 55

Create a volume 56

Attach a volume to an instance 57

Detach a volume from an instance 57

Create a snapshot from a volume 58

Edit a volume 58

Delete a volume 58

Install Horizon 59

OpenStack dashboard—Project tab 61

OpenStack dashboard—Admin tab 63

Upload an image 65

Update an image 66

Delete an image 67

Add a rule to the default security group 68

Add a key pair 69

Import a key pair 69

Allocate a floating IP address to an instance 70

Launch an instance 71

Connect to your instance by using SSH 74

Track usage for instances 74

Create an instance snapshot 75

Manage an instance 75

Create a network 76

Create a router 77

Heat Installation on AIO Node 79

Create Database for Orchestration Service 79

Define services and service endpoints 80

Configure Heat Service 81

Verify the Orchestration service installation 82

Launch a stack 83

Manage a stack 85

Delete a stack 85

Telemetry Module Installation on AIO Node 86

Create Database for Telemetry Service 86

Define services and service endpoints 87

Configure Ceilometer Service 87

Install the Compute agent for Telemetry 89

Configure the Image Service for Telemetry 89

Add Block Storage service agent for Telemetry 90

Verify the Telemetry service installation 90

# OpenStack Labs – Objectives

#### Lab1:

Install prerequisites for OpenStack Havana

Install and Configure Identity Service Keystone

#### Lab2:

Install and Configure Image Service Glance

#### Lab3:

Install Compute Controller Service

Configure Compute Node

Launch Instance

#### Lab4:

Install Block Storage Service in aio node

Test and confirm the Service

#### Lab5:

Install Network Service Neutron

Configure Neutron

#### Lab6:

Install Second Compute Node Coupute01

Configure to Connect with aio node

Create Network and spin up vms to show the gre tunneling

Migrate to VLAN based networks

Add Nexus plugin

#### Lab7:

Install Horizon Dashboard

#### Lab8:

Orchestration Service Installation

#### Lab9:

Telemetry Service Installation

#### Lab10:

Install OpenStack with COI

#### Lab11:

Havate: OpenStack and UCS

OpenStack Lab Overview

# Lab Topology



# Lab Access

# RDP/VNC to the POD machines with credentials given by the Instructor

# Each Student will have an AIOX node and ComputeY node installed with Ubuntu 14.04

SSH to the Ubuntu machine with following credentials (**X** and **Y** values for each User will be provided by Instructor)

**aioX Node**

IP Address: 10.1.64.1**X** (eth0)

User Name / Password: localadmin / ubuntu

**ComputeY Node**

IP Address: 10.1.64.1**Y** (eth0)

User Name / Password: localadmin / ubuntu

Lab 1: Identity Service Installation

# Objective:

* Install prerequisites for OpenStack installation in AIO node
  + ntp
  + mysql
  + OpenStack packages
  + Messaging queue server - RabbitMQ Server
* Installation and Configuration of Identity Service Keystone

# Basic Configuration

Check the Network settings of AIO node

SSH to AIO node with the credentials in Lab access section (above)

**Step 1**:

Enter the following command and type **ubuntu** as the [sudo] password

sudo su -

vi /etc/network/interfaces

Enter the network details for aio node as shown below. Only eth0 should have a gateway/dns configuration

eth0 10.1.64.1**X**

eth1 10.1.**65**.1**X**

The file should look something like:

auto eth0

iface eth0 inet static

address 10.1.64.1**X**

netmask 255.255.255.0

network 10.1.64.0

broadcast 10.1.64.255

gateway 10.1.64.1

# dns-\* options are implemented by the resolvconf package, if installed

dns-nameservers 10.1.1.92

dns-search onecloud

auto eth1

iface eth1 inet static

address 10.1.**65**.1**X**

netmask 255.255.255.0

iface eth2 inet static

address 0.0.0.0

**Note:** vi Editor

Press key “**i**” for insert in vi editor

<esc> to get out of edit mode

**:wq** to save the file.

**:q!** to exit without saving

**Step 2**:

Check the hosts file configuration

vi /etc/hosts

Enter the IP address and host names of aio node and compute01 node. (**X** - Student POD Number)

10.1.64.1**X** aio**X**.onecloud aio**X**

10.1.64.1**Y** compute**Y**.onecloud compute**Y**  
10.1.64.1 gw.onecloud gw

**Step 3**:

Type following command to restart network service

ifdown eth1; ifup eth1; ifdown eth2; ifup eth2

Note: it is ok if the system complains about interfaces not being configured in the previous step

# Prerequisites for OpenStack

### **NTP Server**

**Step 4**:

To synchronize services across OpenStack nodes (compute, controller and network).

ntpdate gw.onecloud

apt-get install ntp -y

Edit the /etc/ntp.conf file to point to an accessible ntp server (the default may work as well):

vi /etc/ntp.conf

Remove or comment out the following the lines:

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

and add the following line:

server gw.onecloud

Then restart NTP and make sure it’s connected to the clock:

service ntp restart

ntpq -p

### **MySQL**

**Step 5:**

Most of the OpenStack services require a database to store information. Here we are using MySQL for that purpose.

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

Installation will prompt for password, Type **pass** as root password and press Enter. For this lab, it is important that you stick with the generic passwords and user-ids. Clearly you would use more secure random strings for a production system.

NOTE: Do not set a random password in the lab, just use the define passwords so that the configurations are consistent!

**Step 6:**

Edit /etc/mysql/my.cnf and set the bind-address to the IP address of AIO node, to enable access from outside.

vi /etc/mysql/my.cnf

Use **/bind-address** to find the **bind-address parameter** in **[mysqld]** section

Change the bind-address as 10.1.64.1**X** from its default of 127.0.0.1 so that remote services can access they Mysql service.

bind-address = 10.1.64.1**X**

default-storage-engine = innodb

innodb\_file\_per\_table

collation-server = utf8\_general\_ci

init-connect = 'SET NAMES utf8'

character-set-server = utf8

Then restart the MySQL service to apply the changes.

**Step 7:**

Restart the MySQL service to apply the changes.

service mysql restart

### **Openstack Packages**

**Step 8:**

This section describes the configuration you must complete after you configure machine to install the OpenStack Havana packages.

apt-get update && apt-get dist-upgrade -y

# Messaging Queue Sever

**Step 9:**

Since OpenStack maintaining asynchronous calls between its services, we need a messaging server RabbitMQ server.

apt-get install rabbitmq-server -y

**Step 10:**

Change the default guest password of RabbitMQ to **pass**

rabbitmqctl change\_password guest pass

service rabbitmq-server restart

# Identity Service Keystone

**Step 11:**

Install the OpenStack identity service on node.

apt-get install keystone -y

The Identity Service uses a database to store information. Specify the location of the database in the configuration file. We use a MySQL database with the username keystone.

**Step 12:**

Edit /etc/keystone/keystone.conf and change connection in the [database] section for MySQL

vi /etc/keystone/keystone.conf

/[database] to find the **[database]** section

Change the connection parameter:

connection = mysql://keystone:pass@aio**X**/keystone

Press **Esc** key

Type **:wq** for save the file.

**Step 13:**

By default, the Ubuntu packages create an SQLite database. Delete the keystone.db file created in the /var/lib/keystone/ directory so that it does not get used by mistake.

rm /var/lib/keystone/keystone.db

# Keystone Database

**Step 14:**

Create Keystone database by login to mysql with password as **pass**

mysql -uroot -ppass

CREATE DATABASE keystone;

GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'localhost' IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON keystone.\* TO 'keystone'@'%' IDENTIFIED BY 'pass';

exit

**Step 15:**

Create the database tables for the Identity Service:

su -s /bin/sh -c "keystone-manage db\_sync" keystone

**Step 16:**

Define an authorization token to use as a shared secret between the Identity Service and other OpenStack services. Edit /etc/keystone/keystone.conf and change admin\_token with ADMIN\_TOKEN in [DEFAULT] section:

vi /etc/keystone/keystone.conf

# Administrative Token

# admin\_token = ADMIN\_TOKEN

admin\_token = ADMIN\_TOKEN

^ NOTE: There cannot be a space at the beginning of configuration parameter lines

**Step 17:**

Restart the Identity Service:

service keystone restart

# Define users, tenants and roles

After you install the Identity Service, set up users, tenants, and roles. These are used to allow access to services and endpoints.

You would indicate a user and password to authenticate with the Identity Service. At this point, however, we have not created any users, so we have to use the authorization token created in an earlier step.

You can pass this with the --os-token option to the keystone command or set the OS\_SERVICE\_TOKEN environment variable. We'll set OS\_SERVICE\_TOKEN, as well as OS\_SERVICE\_ENDPOINT to specify where the Identity Service is running.

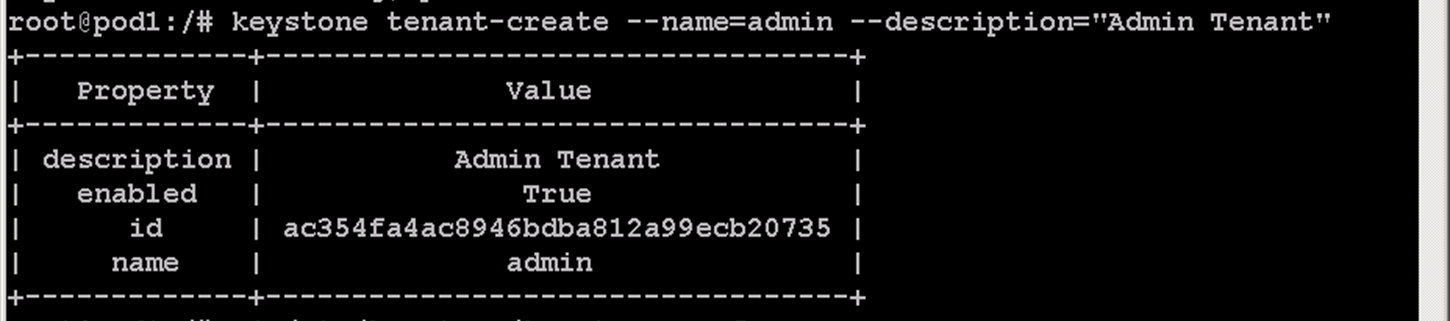
**Step 18:**

export OS\_SERVICE\_TOKEN=ADMIN\_TOKEN

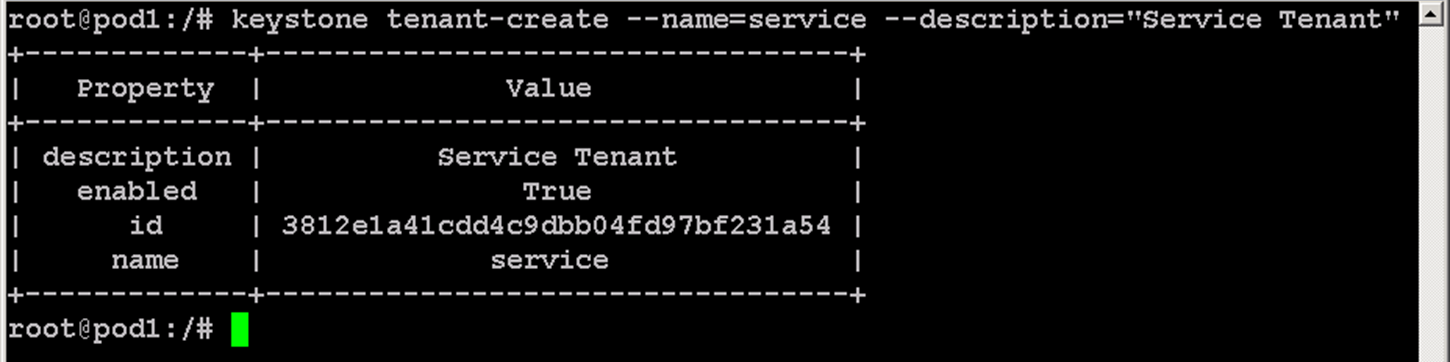
export OS\_SERVICE\_ENDPOINT=http://aioX:35357/v2.0

Create a tenant for an administrative user and a tenant for other OpenStack services

keystone tenant-create --name=admin --description="Admin Tenant"



keystone tenant-create --name=service --description="Service Tenant"



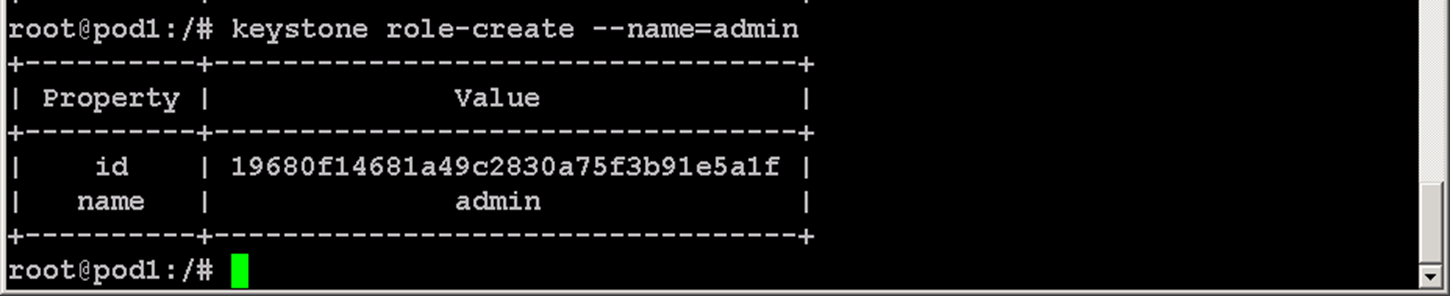
Create an administrative user called admin with password as **pass** and an email address for the account

keystone user-create --name=admin --pass=pass --email=admin@onecloud.com



Create a role for administrative tasks called admin. Any roles you create should map to roles specified in the policy.json files of the various OpenStack services. The default policy files use the admin role to allow access to most services.

keystone role-create --name=admin



Add roles to users. Users always log in with a tenant, and roles are assigned to users within tenants. Add the admin role to the admin user when logging in with the admin tenant.

keystone user-role-add --user=admin --tenant=admin --role=admin

We also add a member role and add admin as a member of the admin tenant, otherwise Horizon will not properly load.

keystone role-create --name=Member

keystone user-role-add --user=admin --tenant=admin --role=Member

Check the user and tenant list

keystone user-list

keystone tenant-list

# Define services and service endpoints

you must register each service in your OpenStack installation, so that the Identity Service can track which OpenStack services are installed and where they are located on the network. To register a service, run these commands:

• *keystone service-create* (Describes the service)

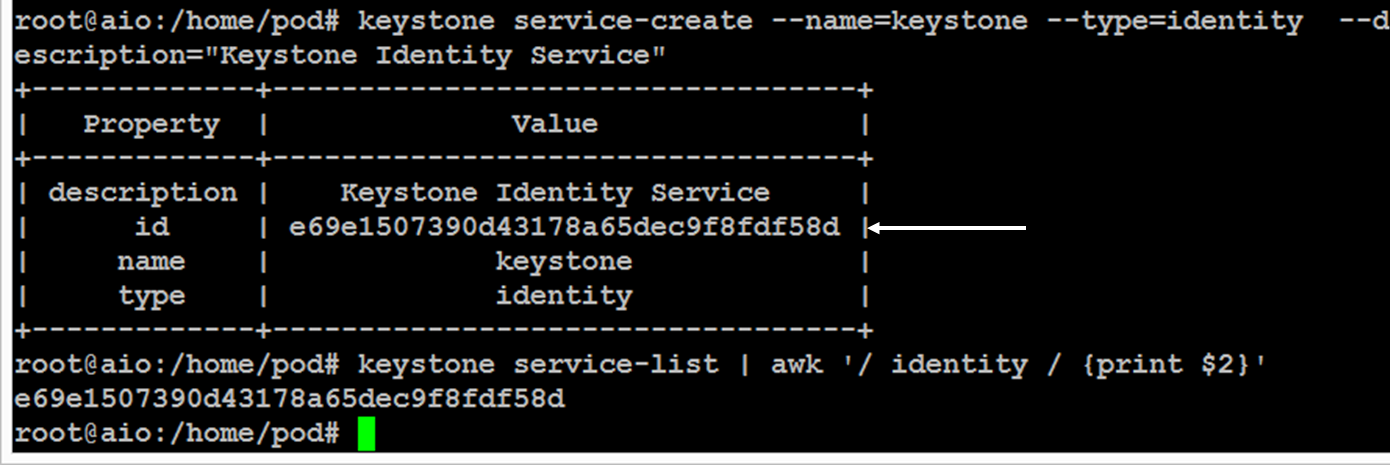
• *keystone endpoint-create* (Associates API endpoints with the service)

**Step 19:**

Create a service entry for the Identity Service:

keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"

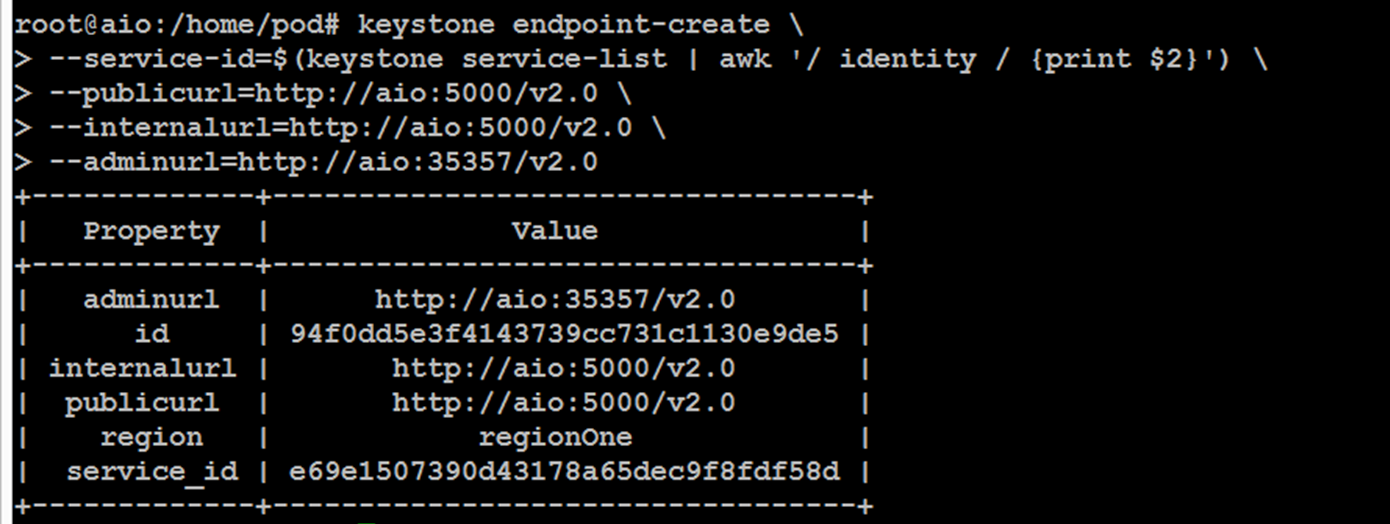
**Note:** Copy the Service id to use in endpoint-create command. Or we can use keystone service-list | awk '/ identity / {print $2}' to get the service id of type identity.



When you specify an endpoint, you provide URLs for the public API, internal API, and admin API. Note that the Identity Service uses a different port for the admin API.

The service ID is randomly generated and is different from the one shown here. The following command will create a keystone identity service endpoint with **service-id** value from the **keystone service-create** command.

keystone endpoint-create --service-id=$(keystone service-list | awk '/ identity / {print $2}') --publicurl=http://aioX:5000/v2.0 --internalurl=http://aioX:5000/v2.0 --adminurl=http://aioX:35357/v2.0



# Verify the Identity service installation

To verify the Identity Service is installed and configured correctly, first unset the OS\_SERVICE\_TOKEN and OS\_SERVICE\_ENDPOINT environment variables. These were only used to bootstrap the administrative user and register the Identity Service.

**Step 20:**

unset OS\_SERVICE\_TOKEN OS\_SERVICE\_ENDPOINT

You can now use regular username-based authentication. Request an authentication token using the admin user and the password you chose during the earlier administrative user-creation step.

keystone --os-username=admin --os-password=pass --os-auth-url=http://aioX:35357/v2.0 token-get

You should receive a token in response, paired with your user ID. This verifies that keystone is running on the expected endpoint, and that your user account is established with the expected credentials.(scroll up in ssh windows to see the user ID)

Next, verify that authorization is behaving as expected by requesting authorization on a tenant.

keystone --os-username=admin --os-password=pass --os-tenant-name=admin --os-auth-url=http://aioX:35357/v2.0 token-get

You should receive a new token in response, this time including the ID of the tenant you specified. This verifies that your user account has an explicitly defined role on the specified tenant, and that the tenant exists as expected.

**Step 21:**

You can also set your --os-\* variables in your environment to simplify command-line usage. Setup a **openrc.sh** file with the admin credentials and admin endpoint.

cd

vi ~/openrc.sh

Enter following line in the openrc.sh file by pressing key “**i**” to insert.

export OS\_USERNAME=admin

export OS\_PASSWORD=pass

export OS\_TENANT\_NAME=admin

export OS\_AUTH\_URL=http://aio**X**:35357/v2.0

Press **Esc** and Type **:wq** to save the file.

You can source this file to read in the environment variables.

**Step 22:**

source ~/openrc.sh

Verify that your openrc.sh file is configured correctly by performing the same command as above, but without the --os-\* arguments.

keystone token-get

The command returns a token and the ID of the specified tenant. This verifies that you have configured your environment variables correctly.

Finally, verify that your admin account has authorization to perform administrative commands.

keystone user-list



This verifies that your user account has the admin role, which matches the role used in the Identity Service policy.json file.

Identity service and prerequisites installed successfully.

Lab 2: Image Service Installation

The OpenStack Image Service enables users to storage and retrieve Operating Systems (virtual machine images), also known as the Glance project. The Image Service offers a REST API that enables you to query Virtual Machine Images and its metadata. Images can be stored in a variety of locations from simple file systems to object – storage systems like Openstack Object Storage.

This lab configures the Image Service to use the file backend. Images uploaded to the Image Service will be stored in a directory on the same system that hosts the service. By default this directory is/var/lib/glance/images/.

# Image Service Installation

**Step 1:**

SSH to AIO node with the credentials in Lab access

Enter following command and Type **pass** as the [sudo] password

sudo su -

**source ~/openrc.sh**

apt-get install glance python-glanceclient -y

# Create Database for Image Service

**Step 2:**

The Image Service stores information about images in a database. Create Keystone database by login to mysql with password as **pass**

mysql -uroot –ppass

CREATE DATABASE glance;

GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'localhost' IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON glance.\* TO 'glance'@'%' IDENTIFIED BY 'pass';

exit

**Step 3:**

By default, the Ubuntu packages create a SQLite database. Delete the glance.sqlite file created in the /var/lib/glance/ directory so that it does not get used by mistake.

rm /var/lib/glance/glance.sqlite

Note: This file may not exist in your system

## User Creation

Create a glance user that the Image Service can use to authenticate with the Identity Service.

**Step 4:**

Choose a password as **pass** and specify an email address for the glance user. Use service as tenant and admin as role.

keystone user-create --name=glance --pass=pass --email=glance@onecloud.com

keystone user-role-add --user=glance --tenant=service --role=admin

# Configure Image Service

The Image Service provides the glance-api and glance-registry services, each with its own configuration file glance-api.conf and glance-registry.conf

#### **Update glance-api.conf and glance-registry.conf**

**Step 5:**

Edit /etc/glance/glance-api.conf

vi /etc/glance/glance-api.conf

/rabbit\_host to find and change the rabbit host details as below

rabbit\_host = aio**X**

rabbit\_userid = guest

rabbit\_password = pass

**Step 6:**

Configure the location of the database.

Search for the [database] section: /[database] and add the sql connection information as follows:

#sqlite\_db = /var/lib/glance/glance.sqlite

connection = mysql://glance:pass@aio**X**/glance

**Note:** you need to comment out the sqlite path if it exists.



Press **Down Arrow** key to Scroll down up to **[keystone\_authtoken]** section. To configure the Image Service to use the Identity Service for authentication, edit the following keys under the **[keystone\_authtoken]** section

auth\_uri = http://aio**X**:5000

auth\_host = aio**X**

auth\_port=35357

auth\_protocol=http

admin\_tenant\_name = service

admin\_user = glance

admin\_password = pass

Press **Down Arrow** key to scroll down up to **[paste\_deploy]** section

Add the following key under the [paste\_deploy] section:

flavor = keystone

Press **Esc** key and Type **:wq** to save the file.

**Step 7:**

Similarly edit /etc/glance/glance-registry.conf

vi /etc/glance/glance-registry.conf

Follow the instructions in **Step 6** to finish this task.

#### **Update glance-api-paste.ini and glance-registry-paste.ini**

**Step 8:**

Add the credentials to the /etc/glance/glance-api-paste.ini

vi /etc/glance/glance-api-paste.ini

**Step 9:**

Press **Down Arrow** key to scroll down up to **[filter:authtoken]** section. Press “**i**” to insert

Add the following options under the [filter:authtoken] section and leave any other existing option as it is.

auth\_host=aio**X**

admin\_user=glance

admin\_tenant\_name=service

admin\_password=pass

Press **Esc** key and Type **:wq** to save the file.

**Step 10:**

Similarly edit /etc/glance/glance-registry-paste.ini file.

vi /etc/glance/glance-registry-paste.ini

Follow the instructions in **Step 9** to finish this task

# Define services and service endpoints

Register the Image Service with the Identity Service so that other OpenStack services can locate it.

**Step 11:**

Register the service and create the endpoint:

keystone service-create --name=glance --type=image --description="Glance Image Service"

**Note:** Service id to use in endpoint-create command

Replace the **service-id** value with value of id from the **keystone service-create** command and run the following command to create image service end point

keystone endpoint-create --service-id=$(keystone service-list | awk '/ image / {print $2}') --publicurl=http://aioX:9292 --internalurl=http://aioX:9292 --adminurl=http://aioX:9292

**Step 12:**

Create the database tables for the Image Service:

su -s /bin/sh -c "glance-manage db\_sync" glance

**If glance complains about UTF mismatch**

2014-07-16 23:23:50.875 4460 CRITICAL glance [-] ValueError: Tables "migrate\_version" have non utf8 collation, please make sure all tables are CHARSET=utf8

mysql -uroot -ppass

USE glance;

ALTER TABLE migrate\_version CONVERT TO character set utf8 collate utf8\_unicode\_ci;

exit

then run db\_sync again:

**su -s /bin/sh -c "glance-manage db\_sync" glance**

**Step 13:**

Restart the glance service with its new settings.

service glance-registry restart

service glance-api restart

# Verify the Image Service Installation:

**Step 14:**

Download the image into a dedicated directory using wget or curl:

mkdir images

cd images/

wget http://10.1.1.92/images/cirros-0.3.2-x86\_64-disk.img

**Step 15:**

Upload the image into Image service

glance image-create --name="CirrOS 0.3.2" --disk-format=qcow2 --container-format=bare --is-public=true < cirros-0.3.2-x86\_64-disk.img

Confirm that the image was uploaded and display its attributes:

glance image-list

cd ..

Image Service configured and image upload successfully.

Lab 3: Compute Service Installation

The Compute service is a cloud computing fabric controller, which is the main part of an IaaS system. Use it to host and manage cloud computing systems.

Compute interacts with the Identity Service for authentication, Image Service for images, and the Dashboard for the user and administrative interface. Access to images is limited by project and by user; quotas are limited per project (for example, the number of instances). The Compute service scales horizontally on standard hardware, and downloads images to launch instances as required.

# Compute Service Installation

SSH to AIO node with the credentials in Lab access

Enter following command and Type **pass** as the [sudo] password

**Step 1:**

sudo su -

**source ~/openrc.sh**

##### Install Nova Network

**Step 2:**

Install Nova Networking to test the vm provision in this lab. Later we replace this networking with Neutron

apt-get install nova-network nova-api-metadata -y

##### Install Compute Controller Service packages

**Step 3:**

Install these Compute packages, which provide the Compute services that run on the controller node.

apt-get install nova-novncproxy novnc nova-api nova-ajax-console-proxy nova-cert nova-conductor nova-consoleauth nova-doc nova-scheduler python-novaclient -y

##### Install Compute Node packages

**Step 4:**

Install the appropriate packages for the Compute service.

export DEBIAN\_FRONTEND=noninteractive

apt-get install nova-compute-kvm python-guestfs -y

To make the current kernel readable, run the following command

dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-$(uname -r)

# Create Database for Compute Service

**Step 5:**

Create Database nova for Compute service by login to mysql with password as **pass**

mysql -uroot -ppass

CREATE DATABASE nova;

GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'localhost' IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON nova.\* TO 'nova'@'%' IDENTIFIED BY 'pass';

exit

**Step 6:**

By default, the Ubuntu packages create a SQLite database. Delete the nova.sqlite file created in the /var/lib/nova/ directory so that it does not get used by mistake.

rm /var/lib/nova/nova.sqlite

**Step 7:**

Create a nova user that Compute uses to authenticate with the Identity Service. Use the service tenant and give the user the admin role: (**X** Student POD Number)

keystone user-create --name=nova --pass=pass --email=nova@onecloud.com

keystone user-role-add --user=nova --tenant=service --role=admin

# Configure Compute Service

#### **Update nova.conf**

**Step 8:**

Edit /etc/nova/nova.conf

vi /etc/nova/nova.conf

Configure the Compute Service to use the RabbitMQ message broker by adding these configuration keys at the end of **[DEFAULT]** section

vif\_plugging\_is\_fatal=false

vif\_plugging\_timeout=0

rpc\_backend =nova.rpc.impl\_kombu

rabbit\_host=aio**X**

rabbit\_password=pass

Add the my\_ip, vncserver\_listen, and vncserver\_proxyclient\_address configuration options to to the **[DEFAULT]** section: To get vnc acces from the pod machine give IP address instead of hostname.

my\_ip=10.1.64.1**X**

vncserver\_listen=10.1.64.1**X**

vncserver\_proxyclient\_address=10.1.64.1**X**

novncproxy\_base\_url=http://10.1.64.1**X**:6080/vnc\_auto.html

auth\_strategy=keystone

glance\_host=aio**X**

To configure network settings, add following lines to the end of **[DEFAULT]** section:

network\_manager=nova.network.manager.FlatDHCPManager

firewall\_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver

network\_size=254

allow\_same\_net\_traffic=False

multi\_host=True

send\_arp\_for\_ha=True

share\_dhcp\_address=True

force\_dhcp\_release=True

flat\_network\_bridge=br100

flat\_interface=eth1

public\_interface=eth1

To configure the location of the database, add following lines to the **[database]** and **[keystone\_authtoken]** section:

[database]

connection = mysql://nova:pass@aio**X**/nova

[keystone\_authtoken]

auth\_uri = http://aio**X**:5000

auth\_host = aio**X**

auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = nova

admin\_password = pass

Press **Esc** and Type **:wq** to Save the file.

#### **Edit api-paste.ini**

**Step 9:**

Add the credentials to the /etc/nova/api-paste.ini file. Add following options to the **[filter:authtoken]** section

vi /etc/nova/api-paste.ini

Add and update these setting under the [filter:authtoken] section

auth\_uri = http://aio**X**:5000/v2.0

auth\_host = aio**X**

auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = nova

admin\_password = pass

# Define services and service endpoints

Register the Compute service and specify the endpoint: (**X** Student POD Number)

**Step 10:**

keystone service-create --name=nova --type=compute --description="Nova Compute service"



**Note:** Create the Service endpoint for the compute service

keystone endpoint-create --service-id=$(keystone service-list | awk '/ compute / {print $2}') --publicurl=http://aioX:8774/v2/%\(tenant\_id\)s --internalurl=http://aioX:8774/v2/%\(tenant\_id\)s --adminurl=http://aioX:8774/v2/%\(tenant\_id\)s



**Step 11:**

Create the database tables for the nova database and Restart nova services

su -s /bin/sh -c "nova-manage db sync" nova

service nova-api restart

service nova-cert restart

service nova-consoleauth restart

service nova-scheduler restart

service nova-conductor restart

service nova-novncproxy restart

service nova-compute restart

service nova-network restart

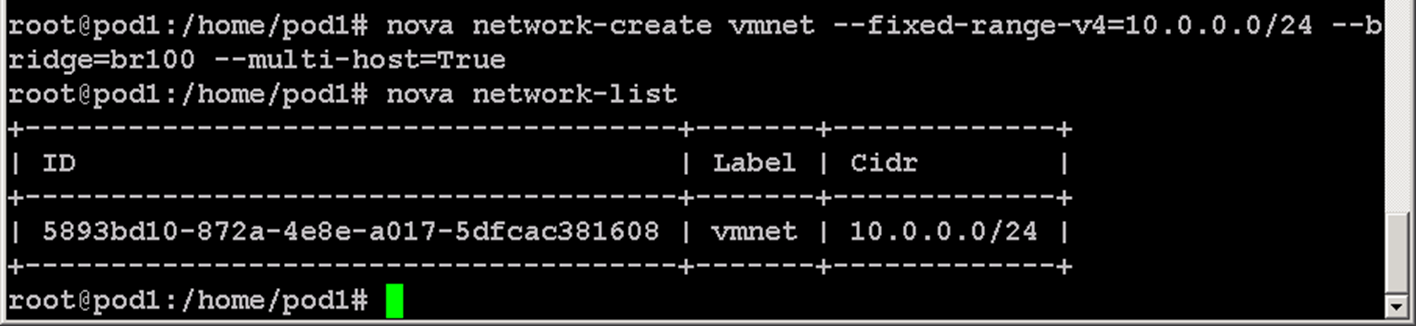
# Create Network

Create a nova network that virtual machines can use. Do this once for the entire installation and not on each compute node. This will be override later with neutron

**Step 12:**

nova network-create vmnet --fixed-range-v4=10.0.0.0/24 --bridge=br100 --multi-host=True

nova network-list



# Launch an Instance

An instance is a virtual machine that OpenStack provisions on a Compute servers.

Generate a keypair that consists of a private and public key to be able to launch instances on OpenStack. These keys are injected into the instances to make password-less SSH access to the instance.

**Step 13:**

Type ssh-keygen to generate the keypair and Press **Enter** for all prompts

ssh-keygen



Add the key to the system as mykey

nova keypair-add --pub\_key ~/.ssh/id\_rsa.pub mykey

You have just created the mykey keypair. The id\_rsa private key is saved locally in ~/.ssh, which you can use to connect to an instance launched by using mykey as the keypair. To view available keypairs:

nova keypair-list



**Step 14:**

To use SSH, ping and RDP to the VMs, configure security group rules.

nova secgroup-add-rule default tcp 22 22 0.0.0.0/0

nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0



To launch an instance, you must specify the flavor ID, keypair, image ID. A flavor is a resource allocation profile. It specifies how many virtual CPUs and how much RAM your instance gets.

**Step 15:**

To see a list of the available profiles:

nova flavor-list



Get the ID of the image to use for the instance:

nova image-list



We can copy the image ID of the CirrOS 0.3.1 or use the command (nova image-list | awk '/ CirrOS 0.3.1/ {print $2}'). Launch the instance as cirros with a flavor ID and image ID.

nova boot --flavor 1 --key\_name mykey --image $(nova image-list | awk '/ CirrOS 0.3.2 / {print $2}') --security\_group default cirros



After the instance launches, use the nova list to view its status. The status changes from BUILD to ACTIVE:

nova list



After the instance boots and initializes, you can ssh into the instance without a password by using the keypair you specified in the nova boot command. Use the nova list command to get the IP address for the instance.

You do not need to specify the private key because it was stored in the default location, ~/.ssh/.id\_rsa, for the ssh client.

**Step 16:**

SSH to the created instance

ssh cirros@10.0.0.2

hostname



Compute Service is installed and Launch an Instance successfully.

Exit from the vm instance.

Exit

**Step 17:**

Delete the instance and its network

nova delete $(nova list | awk '/ cirros / {print $2}')

nova net-delete $(nova net-list | awk '/ vmnet / {print $2}')

Lab 4: Block Storage Service Installation

In Storage service, we are installing and configuring block storage service Cinder. The Block Storage Service enables management of volumes, volume snapshots, and volume types. The Block Storage Service interacts with Compute to provide volumes for instances.

SSH to AIO node with the credentials in Lab access

Enter following command and Type **ubuntu** as the [sudo] password

**Step 1:**

sudo su -

source ~/openrc.sh

# Cinder Installation on AIO Node

In multinode environment install following OpenStack Block Storage services on the Controller node. The Storage node contains the disk that serves volumes.

**Step 2:**

You can configure OpenStack to use various storage systems. This Lab is using LVM.

apt-get install -y cinder-api cinder-scheduler

apt-get install -y lvm2

apt-get install -y cinder-volume

# Create Database for Storage Service

**Step 3:**

Create Database cinder for Storage service by login to mysql with password as **pass**

mysql -u root -ppass

CREATE DATABASE cinder;

GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'localhost' IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON cinder.\* TO 'cinder'@'%' IDENTIFIED BY 'pass';

exit

**Step 4:**

By default, the Ubuntu packages create a SQLite database. Delete the cinder.sqlite file created in the /var/lib/cinder/ directory so that it does not get used by mistake.

rm /var/lib/cinder/cinder.sqlite

**Step 5:**

Create a cinder user that Storage uses to authenticate with the Identity Service. Use the service tenant and give the user the admin role

keystone user-create --name=cinder --pass=pass --email=cinder@onecloud.com

keystone user-role-add --user=cinder --tenant=service --role=admin

# Define services and service endpoints

**Step 6:**

Register the Block Storage Service with the Identity Service so that other OpenStack services can locate it. Register the service and specify the endpoint.

keystone service-create --name=cinder --type=volume --description="OpenStack Block Storage"

Note: copy the service id and use it create the service end point

keystone endpoint-create --service-id=$(keystone service-list | awk '/ volume / {print $2}') --publicurl=http://aioX:8776/v1/%\(tenant\_id\)s --internalurl=http://aioX:8776/v1/%\(tenant\_id\)s --adminurl=http://aioX:8776/v1/%\(tenant\_id\)s

Similarly register a service and endpoint for version 2 of the Block Storage service API.

keystone service-create --name=cinderv2 --type=volumev2 --description="OpenStack Block Storage v2"

Note: copy the service id and use it create the service end point

keystone endpoint-create --service-id=$(keystone service-list | awk '/ volumev2 / {print $2}') --publicurl=http://aioX:8776/v2/%\(tenant\_id\)s --internalurl=http://aioX:8776/v2/%\(tenant\_id\)s --adminurl=http://aioX:8776/v2/%\(tenant\_id\)s

# Configure Cinder Service

**Step 7:**

Edit /etc/cinder/cinder.conf

vi /etc/cinder/cinder.conf

Change the following in [DEFAULT], [keystone\_authtoken] and [database] sections

[DEFAULT]

...

rpc\_backend = cinder.openstack.common.rpc.impl\_kombu

rabbit\_host = aio**X**

rabbit\_port = 5672

rabbit\_userid = guest

rabbit\_password = pass

glance\_host = aio**X**

[keystone\_authtoken]

auth\_uri = http://aio**X**:5000

auth\_host = aio**X**

auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = cinder

admin\_password = pass

[database]

connection = mysql://cinder:pass@aio**X**/cinder

**Step 8:**

Edit /etc/cinder/api-paste.ini

vi /etc/cinder/api-paste.ini

Change the following in [filter:authtoken] sections

[filter:authtoken]

paste.filter\_factory=keystoneclient.middleware.auth\_token:filter\_factory

auth\_host=aio**X**

auth\_port = 35357

auth\_protocol = http

auth\_uri = http://aio**X**:5000

admin\_tenant\_name=service

admin\_user=cinder

admin\_password=pass

# Configure physical hard disks

**Step 8:**

Type following commands to configure physical disks and create cinder-vloumes

dd if=/dev/zero of=/dev/sdb count=100 bs=1M

fdisk –l



Select the second disk and create physical volume and cinder volume group

pvcreate /dev/sdb

vgcreate cinder-volumes /dev/sdb



pvdisplay



vgdisplay



**Step 9:**

Populate Database and Restart Services

su -s /bin/sh -c "cinder-manage db sync" cinder

service cinder-scheduler restart

service cinder-api restart

service cinder-volume restart

service tgt restart

# Test Cinder Service

**Step 9:**

Create 10GB of block storage as Vol1.

cinder create --display-name Vol1 10

cinder list



vgdisplay cinder-volumes

Lab 5: Network Service Neutron Installation

Like Nova Networking, Neutron manages software-defined networking for your OpenStack installation. However, unlike Nova Networking, you can configure Neutron for advanced virtual network topologies, such as per-tenant private networks and more.

Any given Neutron set up has at least one external network. This network, unlike the other networks, is not merely a virtually defined network. Instead, it represents the view into a slice of the external network that is accessible outside the OpenStack installation.

The Open vSwitch plug-in is one of the most popular core plug-ins. Open vSwitch configurations consists of bridges and ports. With Open vSwitch, you can use two different technologies to create the virtual networks: GRE or VLANs. To use GRE with Open vSwitch, Neutron creates GRE tunnels. These tunnels are ports on a bridge and enable bridges on different systems to act as though they were one bridge, which allows the compute and network nodes to act as one for the purposes of routing.

SSH to AIO node with the credentials in Lab access

Enter following command and Type **pass** as the [sudo] password

**Step 1:**

sudo su -

source ~/openrc.sh

# Neutron Installation

**Step 2:**

ifconfig br100 down

brctl delbr br100

apt-get purge nova-network nova-api-metadata -y

apt-get install neutron-server -y

apt-get install neutron-dhcp-agent dnsmasq neutron-plugin-openvswitch-agent neutron-plugin-openvswitch neutron-l3-agent -y

apt-get install openvswitch-datapath-dkms -y

# Database Creation

Create neutron database by login to mysql with password as **pass**

**Step 3:**

mysql -u root -ppass

CREATE DATABASE neutron;

GRANT ALL PRIVILEGES ON neutron.\* TO 'neutron'@'localhost' \

IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON neutron.\* TO 'neutron'@'%' \

IDENTIFIED BY 'pass';

exit

Create User Nuetron with password as pass

**Step 4:**

keystone user-create --name=neutron --pass=pass --email=neutron@onecloud.com

keystone user-role-add --user=neutron --tenant=service --role=admin

# Define services and service endpoints

**Step 5:**

keystone service-create --name=neutron --type=network \

--description="OpenStack Networking Service"

Note the service ID for creating the end point

keystone endpoint-create --service-id $(keystone service-list | awk '/ network / {print $2}') --publicurl http://aioX:9696 --adminurl http://aioX:9696 --internalurl http://aioX:9696

Enable packet forwarding and disable packet destination filtering so that the network node can coordinate traffic for the VMs. Edit the /etc/sysctl.conf file, as follows:

**Step 6:**

vi /etc/sysctl.conf

Edit the following settings

net.ipv4.ip\_forward=1

net.ipv4.conf.all.rp\_filter=0

net.ipv4.conf.default.rp\_filter=0

Save the file by pressing ‘**Esc’** and then type **:wq**

Type the following command and confirm the changes are saved

sysctl -p

# Configure Neutron Service

Edit the /etc/neutron/neutron.conf file.

**Step 7:**

vi /etc/neutron/neutron.conf

[DEFAULT]

core\_plugin = neutron.plugins.openvswitch.ovs\_neutron\_plugin.OVSNeutronPluginV2

auth\_strategy = keystone

allow\_overlapping\_ips = True

rpc\_backend = neutron.openstack.common.rpc.impl\_kombu

rabbit\_host = aio**X**

rabbit\_password = pass

rabbit\_userid = guest

[keystone\_authtoken]

auth\_uri = http://aio**X**:5000

auth\_host = aio**X**

auth\_protocol = http

auth\_port = 35357

admin\_tenant\_name = service

admin\_user = neutron

admin\_password = pass

[database]

connection = mysql://neutron:pass@aio**X**/neutron

Edit the /etc/neutron/api-paste.ini file and add these lines to the [filter:authtoken] section

**Step 8:**

vi /etc/neutron/api-paste.ini

[filter:authtoken]

paste.filter\_factory = keystoneclient.middleware.auth\_token:filter\_factory

auth\_host = aio**X**

auth\_uri = http://aio**X**:5000

admin\_tenant\_name = service

admin\_user = neutron

admin\_password = pass

To perform DHCP on the software-defined networks, Networking supports several different plug-ins. However, in this lab we are using Dnsmasq plug-in.

Edit the /etc/neutron/dhcp\_agent.ini file:

**Step 9:**

vi /etc/neutron/dhcp\_agent.ini

interface\_driver = neutron.agent.linux.interface.OVSInterfaceDriver

dhcp\_driver = neutron.agent.linux.dhcp.Dnsmasq

use\_namespaces = True

enable\_isolated\_metadata = True

enable\_metadata\_network = True

Edit the /etc/neutron/l3\_agent.ini:

**Step 10:**

vi /etc/neutron/l3\_agent.ini

interface\_driver = neutron.agent.linux.interface.OVSInterfaceDriver

use\_namespaces = True

metadata\_port = 9697

external\_network\_bridge = br-ex

Edit the /etc/neutron/metadata\_agent.ini file and modify the [DEFAULT] section:

**Step 11:**

vi /etc/neutron/metadata\_agent.ini

[DEFAULT]

auth\_url = http://aio**X**:5000/v2.0

auth\_region = regionOne

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = neutron

admin\_password = pass

nova\_metadata\_ip = aio**X**

nova\_metadata\_port = 8775

metadata\_proxy\_shared\_secret = pass

neutron\_insecure = True

Configure the OVS plug-in to use GRE tunneling.

Edit the /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini file:

**Step 12:**

vi /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini

[ovs]

tenant\_network\_type = gre

enable\_tunneling = True

tunnel\_type = gre

tunnel\_id\_ranges = 1:1000

integration\_bridge = br-int

tunnel\_bridge = br-tun

local\_ip = 10.1.65.1**X**

[agent]

tunnel\_types = gre

[securitygroup]

firewall\_driver = neutron.agent.linux.iptables\_firewall.OVSHybridIptablesFirewallDriver

[database]

connection = mysql://neutron:pass@aio**X**/neutron

Edit the /etc/nova/nova.conf file to define a secret key that will be shared between the Compute Service and the Networking metadata agent.

**Step 13:**

vi /etc/nova/nova.conf

**Remove** following from the [DEFAULT] section:

……..

network\_manager=nova.network.manager.FlatDHCPManager

firewall\_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver

network\_size=254

allow\_same\_net\_traffic=False

multi\_host=True

send\_arp\_for\_ha=True

share\_dhcp\_address=True

force\_dhcp\_release=True

flat\_network\_bridge=br100

flat\_interface=eth1

public\_interface=eth1

Add following to the [DEFAULT] section:

[DEFAULT]

…

neutron\_metadata\_proxy\_shared\_secret = pass

service\_neutron\_metadata\_proxy = true

network\_api\_class=nova.network.neutronv2.api.API

neutron\_url=http://aio**X**:9696

neutron\_auth\_strategy=keystone

neutron\_admin\_tenant\_name=service

neutron\_admin\_username=neutron

neutron\_admin\_password=pass

neutron\_admin\_auth\_url=http://aio**X**:35357/v2.0

linuxnet\_interface\_driver = nova.network.linux\_net.LinuxOVSInterfaceDriver

No matter which networking technology you use, you must add the br-int integration bridge, which connects to the VMs, and the br-ex external bridge, which connects to the outside world.

**Step 14:**

service openvswitch-switch restart

ovs-vsctl add-br br-int

Add external management ip and br-ex to /etc/network/interfaces, as follows:

**Step 15:**

vi /etc/network/interfaces

auto eth2

iface eth2 inet manual

pre-down ovs-vsctl del-port br-ex eth2

pre-down ovs-vsctl del-br br-ex

post-up ovs-vsctl add-br br-ex

post-up ovs-vsctl add-port br-ex eth2

auto br-ex

iface br-ex inet manual

Save the file by press **Esc** key and then type **:wq**

ifup eth2

ifup br-ex

interface br-ex already configured

Restart Services

**Step 16:**

service nova-api restart

service nova-scheduler restart

service nova-conductor restart

service neutron-server restart

service neutron-dhcp-agent restart

service neutron-l3-agent restart

service neutron-metadata-agent restart

service neutron-plugin-openvswitch-agent restart

service openvswitch-switch restart

Check all the Neutron agents are working

neutron agent-list

Verify OVS configuration:

ovs-vsctl show

Lab 6: Adding Compute Node

We are going add Compute**Y** as a compute node to the AIO**X** node

# Basic Configuration

Check the Network settings of Compute**Y** node

SSH to Compute**Y** node with the credentials in Lab access

Enter the following command and Type pass as the [sudo] password

**Step 1**:

sudo su –

vi /etc/network/interfaces

Enter the network details for the Compute**Y** node as shown below.

auto eth0

iface eth0 inet static

address 10.1.64.1**Y**

netmask 255.255.255.0

gateway 10.1.64.1

dns-nameservers 10.1.1.92

dns-search onecloud

auto eth1

iface eth1 inet static

address 10.1.65.1**Y**

netmask 255.255.255.0

auto eth2

iface eth2 inet manual

pre-down ovs-vsctl del-port br-ex eth2

pre-down ovs-vsctl del-br br-ex

post-up ovs-vsctl add-br br-ex

post-up ovs-vsctl add-port br-ex eth2

auto br-ex

iface br-ex inet manual

Restart the network services if addresses have changed (eth1):

ifdown eth1; ifup eth1

ifdown eth2; ifup eth2

ifdown br-ex; ifup br-ex

Check the /etc/hosts file and add ip address and host name for aio node.

**Step 2**:

vi /etc/hosts

Edit the file as follows

10.1.64.1**X** aio**X**.onecloud aio**X**

10.1.64.1**Y** compute**Y**.onecloud compute**Y**

**Step 3**:

Edit /etc/sysctl.conf file and run the following command to activate changes:

vi /etc/sysctl.conf

net.ipv4.conf.all.rp\_filter=0

net.ipv4.conf.default.rp\_filter=0

sysctl -p

**Step 4**:

Install Openstack Packages in Compute Node

ntpdate gw.onecloud

apt-get install -y ntp

echo “server gw.onecloud iburst” > /etc/ntp.conf

service ntp restart

apt-get install -y python-mysqldb python-software-properties

apt-get update && apt-get dist-upgrade -y

**Step 5**:

Install Nova Hypervisor and Network plugins

apt-get install -y neutron-plugin-openvswitch neutron-plugin-openvswitch-agent openvswitch-datapath-dkms

export DEBIAN\_FRONTEND=noninteractive

apt-get install -y nova-compute-kvm python-novaclient python-guestfs

chmod 0644 /boot/vmlinuz\*

**Step 6**:

Create openrc.sh file

cd

vi ~/openrc.sh

Type the following lines in openrc.sh file

export OS\_USERNAME=admin

export OS\_PASSWORD=pass

export OS\_TENANT\_NAME=admin

export OS\_AUTH\_URL=http://aio**X**:35357/v2.0

Save the file by pressing **Esc** key and then type **:wq**

source ~/openrc.sh

**Step 7**:

Edit /etc/nova/nova.conf and add to the [DEFAULT] section

vi /etc/nova/nova.conf

[DEFAULT]

vif\_plugging\_is\_fatal=false

vif\_plugging\_timeout=0

auth\_strategy=keystone

rpc\_backend = nova.rpc.impl\_kombu

rabbit\_host = aio**X**

rabbit\_password = pass

my\_ip=10.1.64.1**Y**

vnc\_enabled=True

vncserver\_listen=0.0.0.0

vncserver\_proxyclient\_address=10.1.64.1**Y**

novncproxy\_base\_url=http://10.1.64.1**X**:6080/vnc\_auto.html

glance\_host=aio**X**

#networking

network\_api\_class=nova.network.neutronv2.api.API

neutron\_url=http://aio**X**:9696

neutron\_auth\_strategy=keystone

neutron\_admin\_tenant\_name=service

neutron\_admin\_username=neutron

neutron\_admin\_password=pass

neutron\_admin\_auth\_url=http://aio**X**:35357/v2.0

linuxnet\_interface\_driver = nova.network.linux\_net.LinuxOVSInterfaceDriver

[database]

connection = mysql://nova:pass@aio**X**/nova

**Step 8**:

Edit the file /etc/nova/api-paste.ini

vi /etc/nova/api-paste.ini

Type the following lines under [filter:authtoken] section

[filter:authtoken]

paste.filter\_factory=keystoneclient.middleware.auth\_token:filter\_factory

auth\_host=aio**X**

auth\_port = 35357

auth\_protocol = http

admin\_user=nova

admin\_tenant\_name=service

admin\_password=pass

**Step 9**:

Edit the file /etc/neutron/neutron.conf

vi /etc/neutron/neutron.conf

auth\_strategy = keystone

core\_plugin = neutron.plugins.openvswitch.ovs\_neutron\_plugin.OVSNeutronPluginV2

rpc\_backend = neutron.openstack.common.rpc.impl\_kombu

rabbit\_host = aio**X**

rabbit\_password = pass

rabbit\_port = 5672

rabbit\_userid = guest

[keystone\_authtoken]

auth\_url = http://aio**X**:35357/v2.0

auth\_host = aio**X**

admin\_tenant\_name = service

admin\_user = neutron

admin\_password = pass

[database]

connection = mysql://neutron:pass@aio**X**/neutron

**Step 10**:

Edit the file /etc/neutron/api-paste.ini

vi /etc/neutron/api-paste.ini

[filter:authtoken]

paste.filter\_factory = keystoneclient.middleware.auth\_token:filter\_factory

auth\_host = aio**X**

admin\_tenant\_name = service

admin\_user = neutron

admin\_password = pass

**Step 11**:

Edit the file /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini

vi /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini

[ovs]

tenant\_network\_type = gre

tunnel\_id\_ranges = 1:1000

enable\_tunneling = True

tunnel\_type = gre

integration\_bridge = br-int

tunnel\_bridge = br-tun

local\_ip = 10.1.65.1**Y**

[agent]

tunnel\_types = gre

[securitygroup]

firewall\_driver = neutron.agent.linux.iptables\_firewall.OVSHybridIptablesFirewallDriver

**Step 12**:

Create a bridge for internal communication and restart the services

ovs-vsctl add-br br-int

service nova-compute restart

service openvswitch-switch restart

service neutron-plugin-openvswitch-agent restart

**Step 13**:

Source the openrc.sh file

source ~/openrc.sh

**Step 14**:

Type following command and check the new Compute node is listed

nova service-list

neutron agent-list

ovs-vsctl show

Now Compute**Y** is successfully added to AIO node as a Compute Node.

# Test Neutron/GRE tunnel between aioX and computeY

On the controller node (aioX)

neutron net-create private

neutron subnet-create private 172.16.0.0/24 --dns-nameserver 10.1.1.92

nova boot --image CirrOS\ 0.3.2 --flavor 1 --availability\_zone nova:aioX --key\_name mykey test-aio

nova boot --image CirrOS\ 0.3.2 --flavor 1 --availability\_zone nova:computeY --key\_name mykey test-compute

# Check to see if the VMs were spun up correctly

nova list

# Log into the console on the VMs to see if they can ping each other

nova get-vnc-console test-aio novnc

nova get-vnc-console test-compute novnc

Lab 6a: Neutron Config with Nexus

We will now change from GRE forwarding to VLANS along with the ability to manage upstream Nexus switches

This plugin implementation provides the following capabilities:

* A reference implementation for a Neutron Plugin Framework (For details see: <http://wiki.openstack.org/quantum-multi-switch-plugin>)
* Supports multiple switches in the network
* Supports multiple models of switches concurrently
* Supports use of multiple L2 technologies
* Supports the Cisco Nexus family of switches (Verified with Nexus 3000, 5000, 7000, and 9000 series)

# Setting up the Nexus Switch

The following items need to be considered before proceeding with the Cisco Nexus Plugin deployment on the OpenStack Havana release.

* Your Nexus switch must be connected to a management network separate from the OpenStack data network. By default, we use the eth0 interface for this purpose. The plugin communicates with the switch over this network to set up your data flows.
* The switch must have SSH login and XML API enabled.
* Each compute host on the cloud should be connected to a port on the switch over a dedicated interface just for OpenStack data traffic. By default we use eth1 for this.
* The link from the node running the DHCP agent (by default, this is the OpenStack control node) should be a trunk link trunking all VLANs used for OpenStack traffic.
* Inter switch links should be trunk links trunking all VLANs.

# Installation of ncclient

We are facing “Unknown Host Key” issue with “ncclient-0.4.0” version. So install the latest version of ncclient which is “[ncclient-0.4.1](https://pypi.python.org/packages/source/n/ncclient/ncclient-0.4.1.tar.gz#md5=1c7cac36ad625b0a9d1ab8f94c31daa4)”.

**Download the package:**

wget https://pypi.python.org/packages/source/n/ncclient/ncclient-0.4.1.tar.gz#md5=1c7cac36ad625b0a9d1ab8f94c31daa4

or **Install using pip:**

pip install ncclient==0.4.1

This may ask several dependencies, install them using apt-get.

# Basic Configuration

1. Make a backup copy of /etc/neutron/neutron.conf

2. Edit /etc/neutron.conf and edit the "core\_plugin" for v2 API. Also verify/add keystone information.

core\_plugin = neutron.plugins.cisco.network\_plugin.PluginV2

[keystone\_authtoken]

auth\_host = aioX

auth\_port = 35357

auth\_protocol = http

admin\_tenant\_name = service

admin\_user = admin

admin\_password = pass

4. Configure the model layer to use Openvswitch as the vswitch plugin:

Download The Packages for cisco plugin manually

apt-get install neutron-plugin-cisco

* Now access the below directory and updates according to the instructions.
* Update the "vswitch\_plugin" value of the [cisco\_plugins] section of /etc/neutron/plugins/cisco/cisco\_plugins.ini:

vswitch\_plugin=neutron.plugins.openvswitch.ovs\_neutron\_plugin.OVSNeutronPluginV2

# VLAN Mode Configuration

* Configure the OVS plugin with the following settings in /etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini:

[ovs]

bridge\_mappings = physnet1:br-eth1

network\_vlan\_ranges = physnet1:1000:1100

tenant\_network\_type = vlan

* Configure the [cisco\_plugins] of /etc/neutron/plugins/cisco/cisco\_plugins.ini:

[cisco\_plugins]

#nexus\_plugin=neutron.plugins.cisco.nexus.cisco\_nexus\_plugin\_v2.NexusPlugin

vswitch\_plugin=neutron.plugins.openvswitch.ovs\_neutron\_plugin.OVSNeutronPluginV2

* Configure the Nexus switch information in /etc/neutron/plugins/cisco/cisco\_plugins.ini. The format should include the IP address of the switch, a host that's connected to the switch and the port on the switch that host is connected to. Also, add the Nexus switch credential username and password. You can configure multiple switches as well as multiple hosts per switch as shown in the example below:

[NEXUS\_SWITCH:1.1.1.1]

# Hostname and port used of the node

compute-1=1/1

# Hostname and port used of the node

compute-2=1/2

# Port number where the SSH will be running at the Nexus Switch, e.g.: 22 (Default)

ssh\_port=22

# Provide the Nexus credentials, if you are using Nexus switches. If not this will be ignored.

username=admin

password=mySecretPasswordForNexus

[NEXUS\_SWITCH:2.2.2.2]

# Hostname and port used of the node

compute-3=1/15

# Hostname and port used of the node

compute-4=1/16

# Port number where the SSH will be running at the Nexus Switch, e.g.: 22 (Default)

ssh\_port=22

# Provide the Nexus credentials, if you are using Nexus switches. If not this will be ignored.

username=admin

password=mySecretPasswordForNexus

* Make sure that SSH host key of all Nexus switches is known to the host on which you are running the Neutron service. You can do this simply by logging in to your Neutron host as the user that Neutron runs as and SSHing to the switches at least once. If the host key changes (e.g. due to replacement of the supervisor or clearing of the SSH config on the switch), you may need to repeat this step and remove the old hostkeys from ~/.ssh/known\_hosts.

sudo su – neutron

ssh admin@switch\_name\_or\_ip

* In general, make sure that every Nexus switch used in your system, has a credential entry in /home/neutron/.ssh/known\_hosts and that the file is owned by neutron. This is required for the system to be able to communicate with those switches.
* Start the Neutron service. If something doesn't work, verify the configuration of each of the above files. When you start neutron-server, make sure to include --config-file arguments for both files:

/usr/bin/python /usr/bin/neutron-server --config-file /etc/neutron/neutron.conf --log-file /var/log/neutron/server.log --config-file /etc/neutron/plugins/cisco/cisco\_plugins.ini --config-file/etc/neutron/plugins/openvswitch/ovs\_neutron\_plugin.ini

Alternately, you can include the [ovs] section directly top the top of /etc/neutron/plugins/cisco/cisco\_plugins.ini:

[ovs]

bridge\_mappings = physnet1:br-eth1

network\_vlan\_ranges = physnet1:1000:1100

tenant\_network\_type = vlan

[cisco\_plugins]

#nexus\_plugin=neutron.plugins.cisco.nexus.cisco\_nexus\_plugin\_v2.NexusPlugin

vswitch\_plugin=neutron.plugins.openvswitch.ovs\_neutron\_plugin.OVSNeutronPluginV2

[NEXUS\_SWITCH:1.1.1.1]

# Hostname and port used of the node

compute-1=1/1

# Hostname and port used of the node

compute-2=1/2

# Port number where the SSH will be running at the Nexus Switch, e.g.: 22 (Default)

ssh\_port=22

# Provide the Nexus credentials, if you are using Nexus switches. If not this will be ignored.

username=admin

password=mySecretPasswordForNexus

[NEXUS\_SWITCH:2.2.2.2]

# Hostname and port used of the node

compute-3=1/15

# Hostname and port used of the node

compute-4=1/16

# Port number where the SSH will be running at the Nexus Switch, e.g.: 22 (Default)

ssh\_port=22

# Provide the Nexus credentials, if you are using Nexus switches. If not this will be ignored.

username=admin

password=mySecretPasswordForNexus

Then start neutron-server with one less --config-file argument:

/usr/bin/python /usr/bin/neutron-server --config-file /etc/neutron/neutron.conf --log-file /var/log/neutron/server.log --config-file /etc/neutron/plugins/cisco/cisco\_plugins.ini

# VLAN Creation

**Creating Network:**

neutron net-create vlan3**X** --provider:network\_type vlan --provider:physical\_network physnet1 --provider:segmentation\_id 3**X** --shared --router:external=True

**Creating Subnet and attaching to created network:**

neutron subnet-create --name subnet3**X** --allocation-pool start=192.168.250.10,end=192.168.250.254 vlan3**X** 192.168.250.0/24 --dns\_nameservers list=true 10.121.12.10

**Verify the creation of VLAN network using:**

neutron net-list

Lab 6b: Create and manage volumes

Volumes are block storage devices that you attach to instances to enable persistent storage. You can attach a volume to a running instance or detach a volume and attach it to another instance at any time. You can also create a snapshot from or delete a volume. Only administrative users can create volume types.

# Create a volume

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Click **Create Volume**.

In the dialog box that opens, enter or select the following values.

|  |  |
| --- | --- |
| **Volume Name** | Specify a name for the volume. |
| **Description** | Optionally, provide a brief description for the volume. |
| **Type** | Leave this field blank. |
| **Size (GB)** | The size of the volume in gigabytes. |
| **Volume Source** | Select one of the following options:  **No source, empty volume**  Creates an empty volume.  An empty volume does not contain a file system or a partition table.  **Snapshot**  If you choose this option, a new field for **Use snapshot as a source** displays. You can select the snapshot from the list.  **Image**  If you choose this option, a new field for **Use image as a source**displays. You can select the image from the list.  Select the **Availability Zone** from the list. By default, this value is set to the availability zone given by the cloud provider (for example,us-west or apac-south). For some cases, it could be nova.  **Volume**  If you choose this option, a new field for **Use volume as a source**displays. You can select the volume from the list.  Options to use a snapshot or a volume as the source for a volume are displayed only if there are existing snapshots or volumes. |

1. Click **Create Volume**.

The dashboard shows the volume on the **Volumes** tab.

# Attach a volume to an instance

After you create one or more volumes, you can attach them to instances. You can attach a volume to one instance at a time.

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the volume to add to an instance and click **Edit Attachments**.
3. In the **Manage Volume Attachments** dialog box, select an instance.
4. Enter the name of the device from which the volume is accessible by the instance.

The actual device name might differ from the volume name because of hypervisor settings.

1. Click **Attach Volume**.

The dashboard shows the instance to which the volume is now attached and the device name.

You can view the status of a volume in the **Volumes** tab of the dashboard. The volume is either Available or In-Use.

Now you can log in to the instance and mount, format, and use the disk.

# Detach a volume from an instance

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the volume and click **Edit Attachments**.
3. Click **Detach Volume** and confirm your changes.

A message indicates whether the action was successful.

# Create a snapshot from a volume

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select a volume from which to create a snapshot.
3. From the **More** list, select **Create Snapshot**.
4. In the dialog box that opens, enter a snapshot name and a brief description.
5. Confirm your changes.

The dashboard shows the new volume snapshot in **Volume Snapshots** tab.

# Edit a volume

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Volumes**.
4. Select the image that you want to edit.
5. In the **Actions** column, click **Edit Volume**.
6. In the **Edit Volume** dialog box, update the name and description of the image.
7. Click **Edit Volume**.

You can extend a volume by using the **Extend Volume** option available in the **More** dropdown list and entering the new value for volume size.

# Delete a volume

When you delete an instance, the data in its attached volumes is not destroyed.

1. Log in to the dashboard, choose a project, and click **Volumes**.
2. Select the check boxes for the volumes that you want to delete.
3. Click **Delete Volumes** and confirm your choice.

A message indicates whether the action was successful.

Lab 7: Horizon Dashboard Installation

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

The dashboard enables web-based interactions with the OpenStack Compute cloud controller through the OpenStack APIs.

**Step 1:**

SSH to AIO node with the credentials in Lab access

Enter following command and Type **pass** as the [sudo] password

sudo su -

source ~/openrc.sh

# Install Horizon

**Step 2:**

Install the dashboard on the node that can contact the Identity Service as root. Remove the openstack-dashboard-ubuntu-theme package. This theme prevents translations, several menus as well as the network map from rendering correctly:

apt-get install apache2 memcached libapache2-mod-wsgi openstack-dashboard -y

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

**Step 3:**

Edit /etc/openstack-dashboard/local\_settings.py and change OPENSTACK\_HOST to the hostname of your Identity Service:

vi /etc/openstack-dashboard/local\_settings.py

OPENSTACK\_HOST = "aio**X**"

**Step 4:**

Start the Apache web server and memcached:

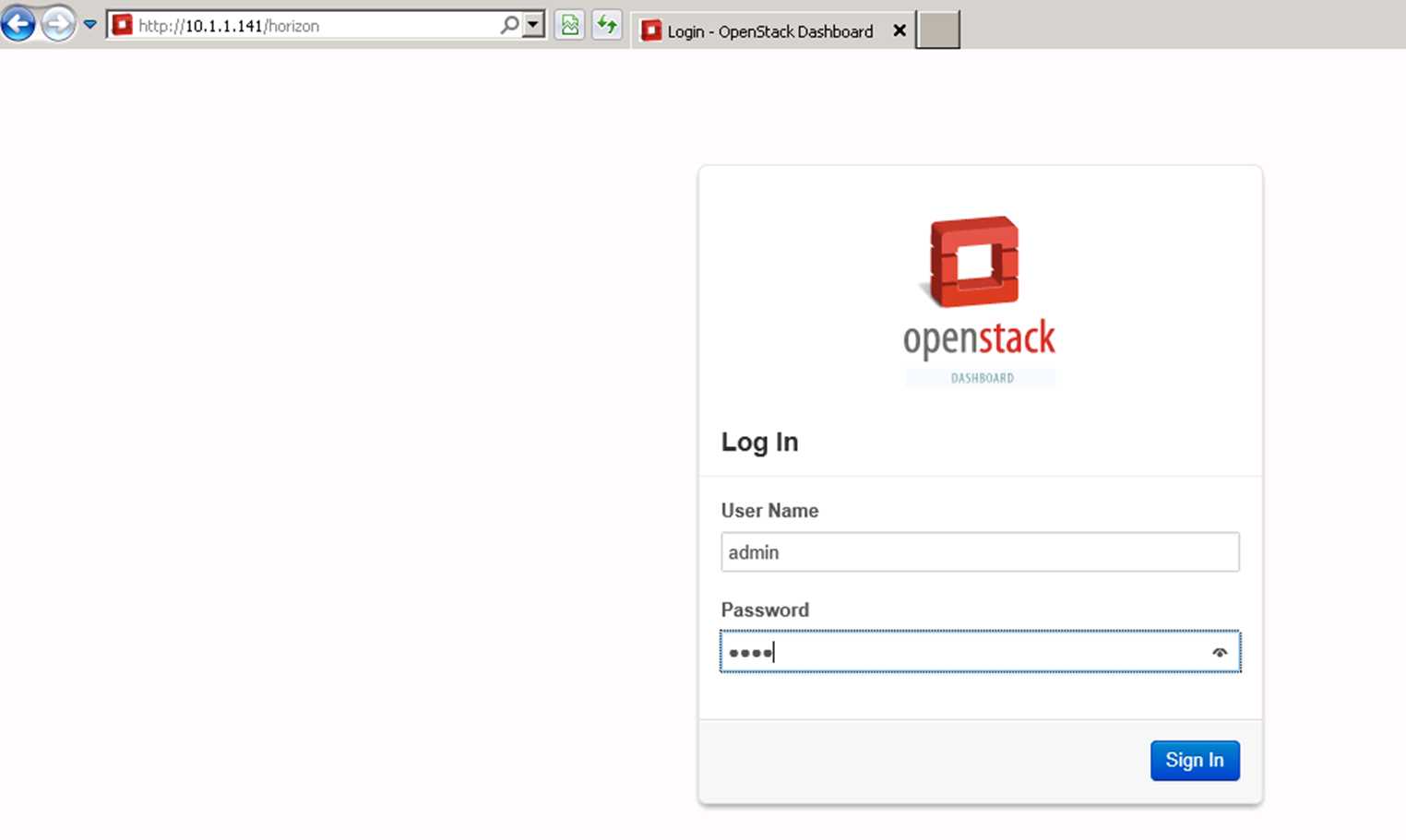
service apache2 restart

service memcached restart

**Step 5:**

Open the webbrowser and type http://***aio node IP address***/horizon

Type user name as admin and password as **pass**



Lab 7a: Log in to the dashboard

The dashboard is available on the node with the nova-dashboard server role.

1. Ask the cloud operator for the host name or public IP address from which you can access the dashboard, and for your user name and password.
2. Open a web browser that has JavaScript and cookies enabled.

To use the Virtual Network Computing (VNC) client for the dashboard, your browser must support HTML5 Canvas and HTML5 WebSockets. The VNC client is based on noVNC. For details, see [noVNC: HTML5 VNC Client](https://github.com/kanaka/noVNC/blob/master/README.md). For a list of supported browsers, see [Browser support](https://github.com/kanaka/noVNC/wiki/Browser-support).

1. In the address bar, enter the host name or IP address for the dashboard.

http://*ipAddressOrHostName*/horizon

If a certificate warning appears when you try to access the URL for the first time, a self-signed certificate is in use, which is not considered trustworthy by default. Verify the certificate or add an exception in the browser to bypass the warning.

1. On the **Log In** page, enter your user name and password, and click **Sign In**.

The top of the window displays your user name. You can also access **Settings** or sign out of the dashboard.

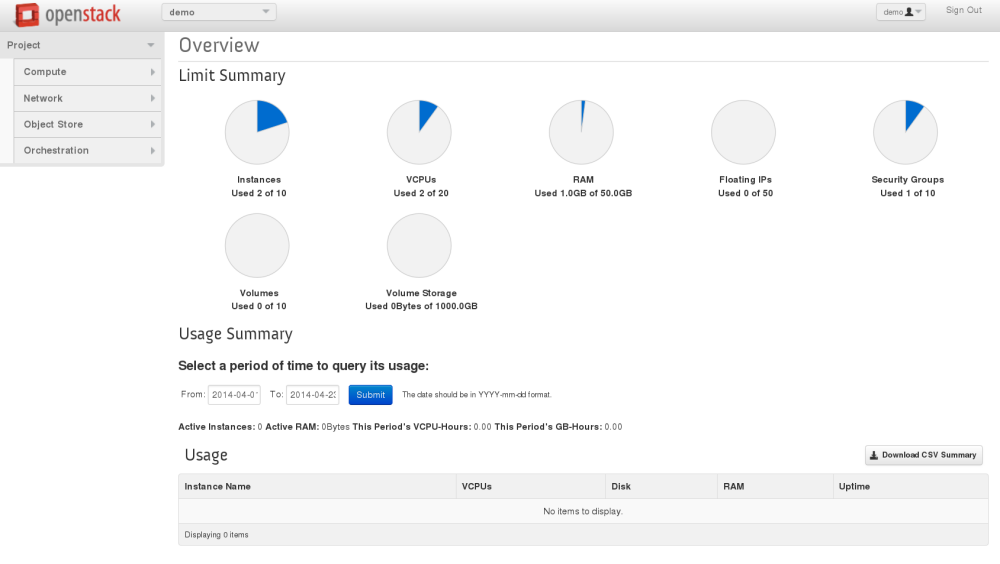
The visible tabs and functions in the dashboard depend on the access permissions, or roles, of the user you are logged in as.

* If you are logged in as an end user, the [Project](http://docs.openstack.org/user-guide/content/log_in_dashboard.html#dashboard_project_tab) tab is displayed.
* If you are logged in as an administrator, the [Project](http://docs.openstack.org/user-guide/content/log_in_dashboard.html#dashboard_project_tab) tab and [Admin](http://docs.openstack.org/user-guide/content/log_in_dashboard.html#dashboard_admin_tab) tab are displayed.

# OpenStack dashboard—Project tab

Projects are organizational units in the cloud, and are also known as tenants or accounts. Each user is a member of one or more projects. Within a project, a user creates and manages instances.

From the Project tab, you can view and manage the resources in a selected project, including instances and images. You select the project from the CURRENT PROJECT list at the top of the tab.

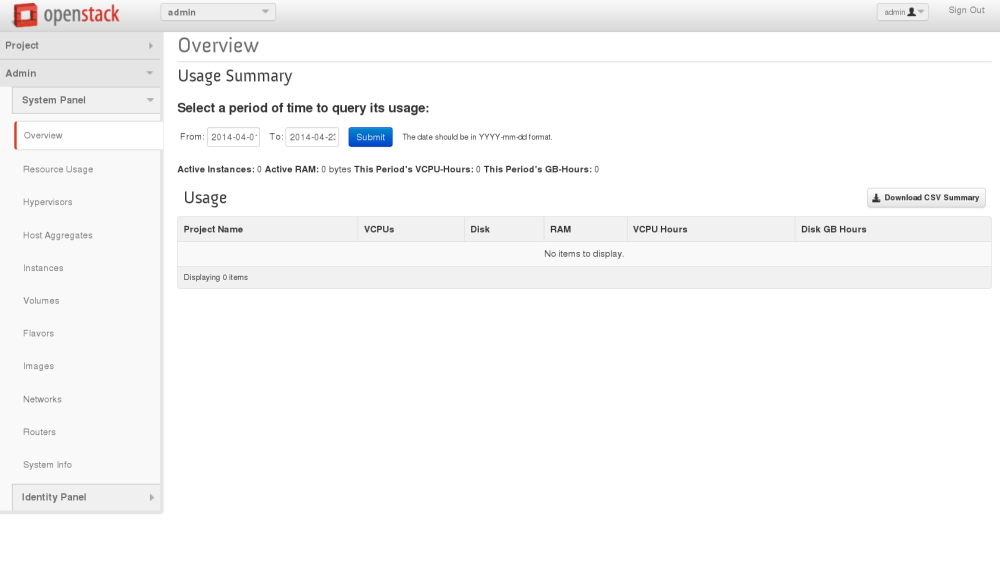


From the **Project** tab, you can access the following tabs:

| **Compute tab** | |
| --- | --- |
| **Overview** | View reports for the project. |
| **Instances** | View, launch, create a snapshot from, stop, pause, or reboot instances, or connect to them through VNC. |
| **Volumes** | Use the following tabs to complete these tasks:  **Volumes**  View, create, edit, and delete volumes.  **Volume Snapshots**  View, create, edit, and delete volume snapshots. |
| **Images** | View images and instance snapshots created by project users, plus any images that are publicly available. Create, edit, and delete images, and launch instances from images and snapshots. |
| **Access & Security** | Use the following tabs to complete these tasks:  **Security Groups**  View, create, edit, and delete security groups and security group rules.  **Key Pairs**  View, create, edit, import, and delete key pairs.  **Floating IPs**  Allocate an IP address to or release it from a project.  **API Access**  View API endpoints. |
| **Network tab** | |
| **Network Topology** | View the network topology. |
| **Networks** | Create and manage public and private networks. |
| **Routers** | Create and manage subnets. |
| **Object Store tab** | |
| **Containers** | Create and manage containers and objects. |
| **Orchestration tab** | |
| **Containers** | Use the REST API to orchestrate multiple composite cloud applications. |

# OpenStack dashboard—Admin tab

Administrative users can use the Admin tab to view usage and to manage instances, volumes, flavors, images, projects, users, services, and quotas.



Access the following categories to complete these tasks:

| **System Panel tab** | | |
| --- | --- | --- |
| **Overview** | | View basic reports. |
| **Resource Usage** | | Use the following tabs to view the following usages:  **Daily Report**  View the daily report.  **Stats**  View the statistics of all resources. |
| **Hypervisors** | | View the hypervisor summary. |
| **Host Aggregates** | | View, create, and edit host aggregates. View the list of availability zones. |
| **Instances** | | View, pause, resume, suspend, migrate, soft or hard reboot, and delete running instances that belong to users of some, but not all, projects. Also, view the log for an instance or access an instance through VNC. |
| **Volumes** | | View, create, edit, and delete volumes and volume types. |
| **Flavors** | | View, create, edit, view extra specifications for, and delete flavors. A flavor is size of an instance. |
| **Images** | | View, create, edit properties for, and delete custom images. |
| **Networks** | | View, create, edit properties for, and delete networks. |
| **Routers** | | View, create, edit properties for, and delete routers. |
| **System Info** | | Use the following tabs to view the service information:  **Services**  View a list of the services.  **Compute Services**  View a list of all Compute services.  **Network Agents**  View the network agents.  **Default Quotas**  View default quota values. Quotas are hard-coded in OpenStack Compute and define the maximum allowable size and number of resources. |
| **Identity Panel tab** | | |
| **Projects** | View, create, assign users to, remove users from, and delete projects. | |
| **Users** | View, create, enable, disable, and delete users. | |

Lab 7b: Upload and manage image

A virtual machine image, referred to in this document simply as an image, is a single file that contains a virtual disk that has a bootable operating system installed on it. Images are used to create virtual machine instances within the cloud. For information about creating image files, see the [OpenStack Virtual Machine Image Guide](http://docs.openstack.org/image-guide/content/).

Depending on your role, you may have permission to upload and manage virtual machine images. Operators might restrict the upload and management of images to cloud administrators or operators only. If you have the appropriate privileges, you can use the dashboard to upload and manage images in the admin project.

You can also use the **glance** and **nova** command-line clients or the Image Service and Compute APIs to manage images. See [the section called “Manage images”](http://docs.openstack.org/user-guide/content/cli_manage_images.html).

# Upload an image

Follow this procedure to upload an image to a project.

1. Log in to the dashboard.
2. From the CURRENT PROJECT on the Project tab, select the appropriate project.
3. On the Project tab, click Images.
4. Click Create Image.

The Create an Image dialog box appears.

1. Enter the following values:

|  |  |
| --- | --- |
| Name | Enter a name for the image. |
| Description | Optionally, enter a brief description of the image. |
| Image Source | Choose the image source from the list. Your choices are Image Location and Image File. |
| Image File or Image Location | Based on your selection for Image Source, you either enter the location URL of the image in the Image Location field. or browse to the image file on your system and add it.  e.g. http source and:  http://10.1.1.92/images/cirros-0.3.2-x86\_64-disk.img |
| Format | Select the correct format (for example, QCOW2) for the image. |
| Architecture | Specify the architecture. For example, i386 for a 32-bit architecture or x86-64for a 64-bit architecture. |
| Minimum Disk (GB) andMinimum RAM (MB) | Leave these optional fields empty. |
| Public | Select this check box to make the image public to all users with access to the current project. |
| Protected | Select this check box to ensure that only users with permissions can delete the image. |

1. Click Create Image.

The image is queued to be uploaded. It might take some time before the status changes from Queued to Active.

# Update an image

Follow this procedure to update an existing image.

1. Log in to the dashboard.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Images**.
4. Select the image that you want to edit.
5. In the **Actions** column, click **More** and then select **Edit** from the list.
6. In the Update Image dialog box, you can perform the following actions:

* Change the name of the image.
* Select the **Public** check box to make the image public.
* Clear the **Public** check box to make the image private.

1. Click **Update Image**.

# Delete an image

Deletion of images is permanent and **cannot** be reversed. Only users with the appropriate permissions can delete images.

1. Log in to the dashboard.
2. From the **CURRENT PROJECT** on the **Project** tab, select the appropriate project.
3. On the **Project** tab, click **Images**.
4. Select the images that you want to delete.
5. Click **Delete Images**.
6. In the **Confirm Delete Image** dialog box, click **Delete Images** to confirm the deletion.

Lab 7c: Configure access and security for instances

Before you launch an instance, you should add security group rules to enable users to ping and use SSH to connect to the instance. To do so, you either [add rules to the default security group](http://docs.openstack.org/user-guide/content/Launching_Instances_using_Dashboard.html#security_groups_add_rule) or add a security group with rules.

Key pairs are SSH credentials that are injected into an instance when it is launched. To use key pair injection, the image that the instance is based on must contain the cloud-init package. Each project should have at least one key pair. For more information, see [the section called “Add a key pair”](http://docs.openstack.org/user-guide/content/Launching_Instances_using_Dashboard.html#keypair_add).

If you have generated a key pair with an external tool, you can import it into OpenStack. The key pair can be used for multiple instances that belong to a project. For more information, see [the section called “Import a key pair”](http://docs.openstack.org/user-guide/content/Launching_Instances_using_Dashboard.html#dashboard_import_keypair).

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated. However, in addition to the fixed IP address, a floating IP address can also be attached to an instance. Unlike fixed IP addresses, floating IP addresses are able to have their associations modified at any time, regardless of the state of the instances involved.

# Add a rule to the default security group

This procedure enables SSH and ICMP (ping) access to instances. The rules apply to all instances within a given project, and should be set for every project unless there is a reason to prohibit SSH or ICMP access to the instances.

This procedure can be adjusted as necessary to add additional security group rules to a project, if your cloud requires them.

1. Log in to the dashboard, choose a project, and click **Access & Security**. The **Security Groups** tab shows the security groups that are available for this project.
2. Select the **default** security group and click **Edit Rules**.
3. To allow SSH access, click **Add Rule**.
4. In the Add Rule dialog box, enter the following values:

|  |  |
| --- | --- |
| **Rule** | SSH |
| **Remote** | CIDR |
| **CIDR** | 0.0.0.0/0 |

To accept requests from a particular range of IP addresses, specify the IP address block in the **CIDR** box.

1. Click **Add**.

Instances will now have SSH port 22 open for requests from any IP address.

1. To add an ICMP rule, click **Add Rule**.
2. In the Add Rule dialog box, enter the following values:

|  |  |
| --- | --- |
| **Rule** | All ICMP |
| **Direction** | Ingress |
| **Remote** | CIDR |
| **CIDR** | 0.0.0.0/0 |

1. Click **Add**.

Instances will now accept all incoming ICMP packets.

# Add a key pair

Create at least one key pair for each project.

1. Log in to the dashboard, choose a project, and click **Access & Security**.
2. Click the **Keypairs** tab, which shows the key pairs that are available for this project.
3. Click **Create Keypair**.
4. In the Create Keypair dialog box, enter a name for your key pair, and click **Create Keypair**.
5. Respond to the prompt to download the key pair.

# Import a key pair

If you don’t have a keypair on your local machine (aka OS-X or Linux or the AIO control host):

cat ~/.ssh/id\_rsa.pub

If that is blank, then:

ssh-keygen –t rsa –f ~/.ssh/id\_rsa –N ''

1. Log in to the dashboard, choose a project, and click **Access & Security**.
2. Click the **Keypairs** tab, which shows the key pairs that are available for this project.
3. Click **Import Keypair**.
4. In the Import Keypair dialog box, enter the name of your key pair, copy the public key into the **Public Key** box, and then click **Import Keypair**.

If you are using the dashboard from a Windows computer, use PuTTYgen to load the \*.pem file and convert and save it as \*.ppk. For more information see the [WinSCP web page for PuTTYgen](http://winscp.net/eng/docs/ui_puttygen).

1. To make the key pair known to SSH, run the **ssh-add** command.

$ ssh-add *yourPrivateKey*.pem

The Compute database registers the public key of the key pair.

The dashboard lists the key pair on the Access & Security tab, though you can not download the public keypair directly from horizon.

# Allocate a floating IP address to an instance

When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated.

However, in addition to the fixed IP address, a floating IP address can also be attached to an instance. Unlike fixed IP addresses, floating IP addresses can have their associations modified at any time, regardless of the state of the instances involved. This procedure details the reservation of a floating IP address from an existing pool of addresses and the association of that address with a specific instance.

1. Log in to the dashboard, choose a project, and click **Access & Security**.
2. Click the **Floating IPs** tab, which shows the floating IP addresses allocated to instances.
3. Click **Allocate IP to Project**.
4. Choose the pool from which to pick the IP address.
5. Click **Allocate IP**.
6. In the **Floating IPs** list, click **Associate**.
7. In the Manage Floating IP Associations dialog box, choose the following options:

* The **IP Address** field is filled automatically, but you can add a new IP address by clicking the **+** button.
* In the **Ports to be associated** field, select a port from the list.

1. The list shows all the instances with their fixed IP addresses.
2. Click **Associate**.

To disassociate an IP address from an instance, click the Disassociate button.

To release the floating IP address back into the pool of addresses, click the More button and select the Release Floating IP option.

Lab 7d: Launch and manage instances

Instances are virtual machines that run inside the cloud.

You can [launch an instance](http://docs.openstack.org/user-guide/content/dashboard_launch_instances_from_image.html) from the following sources:

* Images uploaded to the OpenStack Image Service, as described in [the section called “Upload and manage images”](http://docs.openstack.org/user-guide/content/dashboard_manage_images.html).
* Image that you have copied to a persistent volume. The instance launches from the volume, which is provided by the cinder-volume API through iSCSI.

# Launch an instance

When you launch an instance from an image, OpenStack creates a local copy of the image on the compute node where the instance starts.

When you launch an instance from a volume, note the following steps:

* To select the volume to from which to launch, launch an instance from an arbitrary image on the volume. The image that you select does not boot. Instead, it is replaced by the image on the volume that you choose in the next steps.

To boot a Xen image from a volume, the image you launch in must be the same type, fully virtualized or paravirtualized, as the one on the volume.

* Select the volume or volume snapshot from which to boot. Enter a device name. Enter vda for KVM images or xvda for Xen images.

1. Log in to the dashboard, choose a project, and click **Images**.

The dashboard shows the images that have been uploaded to OpenStack Image Service and are available for this project.

For details on creating images, see [Creating images manually](http://docs.openstack.org/image-guide/content/ch_creating_images_manually.html) in the *OpenStack Virtual Machine Image Guide*.

1. Select an image and click **Launch**.
2. In the Launch Instance dialog box, specify the following values:

| **Details tab** | |
| --- | --- |
| **Availability Zone** | By default, this value is set to the availability zone given by the cloud provider (for example, us-west or apac-south). For some cases, it could be nova. |
| **Instance Name** | Assign a name to the virtual machine.  The name you assign here becomes the initial host name of the server. After the server is built, if you change the server name in the API or change the host name directly, the names are not updated in the dashboard.  Server names are not guaranteed to be unique when created so you could have two instances with the same host name. |
| **Flavor** | Specify the size of the instance to launch.  The flavor is selected based on the size of the image selected for launching an instance. For example, while creating an image, if you have entered the value in the **Minimun RAM (MB)**field as 2048, then on selecting the image, the default flavor is **m1.small**. |
| **Instance Count** | To launch multiple instances, enter a value greater than 1. The default is 1. |
| **Instance Boot Source** | Your options are:  **Boot from image**  If you choose this option, a new field for **Image Name** displays. You can select the image from the list.  **Boot from snapshot**  If you choose this option, a new field for **Instance Snapshot**displays. You can select the snapshot from the list.  **Boot from volume**  If you choose this option, a new field for **Volume** displays. You can select the volume from the list.  **Boot from image (creates a new volume)**  With this option, you can boot from an image and create a volume by entering the **Device Size** and **Device Name** for your volume. Click the **Delete on Terminate** option to delete the volume on terminating the instance.  **Boot from volume snapshot (creates a new volume)**  Using this option, you can boot from a volume snapshot and create a new volume by choosing **Volume Snapshot** from a list and adding a **Device Name** for your volume. Click the **Delete on Terminate**option to delete the volume on terminating the instance.  Since you are launching an instance from an image, **Boot from image** is chosen by default. |
| **Image Name** | This field changes based on your previous selection. Since you have chosen to launch an instance using an image, the **Image Name** field displays. Select the image name from the dropdown list. |
| **Access & Security tab** | |
| **Keypair** | Specify a key pair.  If the image uses a static root password or a static key set (neither is recommended), you do not need to provide a key pair to launch the instance. |
| **Security Groups** | Activate the security groups that you want to assign to the instance.  Security groups are a kind of cloud firewall that defines which incoming network traffic is forwarded to instances. For details, see the section called “Add a rule to the default security group”.  If you have not created any security groups, you can assign only the default security group to the instance. |
| **Networking tab** | |
| **Selected Networks** | To add a network to the instance, click the **+** in the **Available Networks**field. |
| **Post-Creation tab** | |
| **Customization Script** | Specify a customization script that runs after your instance launches. |
| **Advanced Options tab** | |
| **Disk Partition** | Select the type of disk partition from the dropdown list.  **Automatic**  Entire disk is single partition and automatically resizes.  **Manual**  Faster build times but requires manual partitioning. |

1. Click **Launch**.

The instance starts on a compute node in the cloud.

The **Instances** tab shows the instance's name, its private and public IP addresses, size, status, task, and power state.

If you did not provide a key pair, security groups, or rules, users can access the instance only from inside the cloud through VNC. Even pinging the instance is not possible without an ICMP rule configured. To access the instance through a VNC console, see [the section called “Access an instance through a console”](http://docs.openstack.org/user-guide/content/instance_console.html).

# Connect to your instance by using SSH

To use SSH to connect to your instance, you use the downloaded keypair file.

The user name is ubuntu for the Ubuntu cloud images on TryStack.

1. Copy the IP address for your instance.
2. Use the **ssh** command to make a secure connection to the instance. For example:

$ ssh -i MyKey.pem ubuntu@10.0.0.2

1. At the prompt, type yes.

# Track usage for instances

You can track usage for instances for each project. You can track costs per month by showing metrics like number of vCPUs, disks, RAM, and uptime for all your instances.

1. Log in to the dashboard, choose a project, and click **Overview**.
2. To query the instance usage for a month, select a month and click **Submit**.
3. To download a summary, click **Download CSV Summary**.

# Create an instance snapshot

1. Log in to the dashboard, choose a project, and click **Instances**.
2. Select the instance from which to create a snapshot.
3. In the **Actions** column, click **Create Snapshot**.
4. In the Create Snapshot dialog box, enter a name for the snapshot, and click **Create Snapshot**.

The **Images** category shows the instance snapshot.

To launch an instance from the snapshot, select the snapshot and click **Launch**. Proceed with the section called “Launch an instance”.

# Manage an instance

1. Log in to the dashboard, choose a project, and click **Instances**.
2. Select an instance.
3. In the **More** list in the **Actions** column, select the state.

You can resize or rebuild an instance. You can also choose to view the instance console log, edit instance or the security groups. Depending on the current state of the instance, you can pause, resume, suspend, soft or hard reboot, or terminate it.

Lab 7e: Create and manage networks

The OpenStack Networking service provides a scalable system for managing the network connectivity within an OpenStack cloud deployment. It can easily and quickly react to changing network needs (for example, creating and assigning new IP addresses).

Networking in OpenStack is complex. This section provides the basic instructions for creating a network and a router. For detailed information about managing networks, refer to the OpenStack Cloud Administrator Guide.

# Create a network

1. Log in to the dashboard, choose a project, and click **Networks**.
2. Click **Create Network**.
3. In the Create Network dialog box, specify the following values.

| **Network tab** | |
| --- | --- |
| **Network Name** | Specify a name to identify the network. |
| **Subnet tab** | |
| **Create Subnet** | Select this check box to create a subnet  You do not have to specify a subnet when you create a network, but if you do not, any attached instance receives an Error status. |
| **Subnet Name** | Specify a name for the subnet. |
| **Network Address** | Specify the IP address for the subnet. |
| **IP Version** | Select IPv4 or IPv6. |
| **Gateway IP** | Specify an IP address for a specific gateway. This parameter is optional. |
| **Disable Gateway** | Select this check box to disable a gateway IP address. |
| **Subnet Detail tab** | |
| **Enable DHCP** | Select this check box to enable DHCP. |
| **Allocation Pools** | Specify IP address pools. |
| **DNS Name Servers** | Specify a name for the DNS server. |
| **Host Routes** | Specify the IP address of host routes. |

1. Click **Create**.

The dashboard shows the network on the **Networks** tab.

# Create a router

1. Log in to the dashboard, choose a project, and click **Routers**.
2. Click **Create Router**.
3. In the Create Router dialog box, specify a name for the router and click **Create Router**.

The new router is now displayed in the **Routers** tab.

1. Click the new router's **Set Gateway** button.
2. In the **External Network** field, specify the network to which the router will connect, and then click **Set Gateway**.
3. To connect a private network to the newly created router, perform the following steps:
4. On the **Routers** tab, click the name of the router.
5. On the Router Details page, click **Add Interface**.
6. In the Add Interface dialog box, specify the following information:

|  |  |
| --- | --- |
| **Subnet** | Select a subnet. |
| **IP Address (optional)** | Enter the router interface IP address for the selected subnet.  Note: If this value is not set, then by default, the first host IP address in the subnet is used by OpenStack Networking. |

The **Router Name** and **Router ID** fields are automatically updated.

1. Click **Add Interface**.

You have successfully created the router. You can view the new topology from the **Network Topology** tab.

Lab 8: Orchestration Service Installation

In Orchestration Service, we are installing and configuring orchestration service Heat.The Orchestration Service provides a template-based orchestration for describing a cloud application by running OpenStack API calls to generate running cloud applications. The software integrates other core components of OpenStack into a one-file template system. The templates enable you to create most OpenStack resource types, such as instances, floating IPs, volumes, security groups, users, and so on.

SSH to aio**X** node with the credentials in the lab access guide if you don’t still have a terminal connected.

Enter following command

**Step 1:**

sudo su -

source openrc.sh

# Heat Installation on AIO Node

**Step 2:**

On controller node install the orchestration module.

apt-get install heat-api heat-api-cfn heat-engine

# Create Database for Orchestration Service

**Step 3:**

Create Database heat for Orchestration service by login to mysql with password as **pass**

mysql -u root -p

mysql> CREATE DATABASE heat;

mysql> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'localhost' IDENTIFIED BY 'pass';

mysql> GRANT ALL PRIVILEGES ON heat.\* TO 'heat'@'%' IDENTIFIED BY 'pass';

mysql> exit

**Step 4:**

By default, the Ubuntu packages create a SQLite database. Delete the heat.sqlite file created in the /var/lib/heat/ directory so that it does not get used by mistake.

rm /var/lib/heat/heat.sqlite

**Step 5:**

Create a heat user that the Orchestration service uses to authenticate with the Identity Service. Use the service tenant and give the user the admin role

keystone user-create --name=heat --pass=pass --email=heat@onecloud.com

keystone user-role-add --user=heat --tenant=service --role=admin

# Define services and service endpoints

Register the Heat and CloudFormation APIs with the Identity Service so that other OpenStack services can locate these APIs. Register the services and specify the endpoints:

**Step 6:**

keystone service-create --name=heat --type=orchestration --description="Orchestration"

Note: Create service endpoint for the service

keystone endpoint-create \

--service-id=$(keystone service-list | awk '/ orchestration / {print $2}') \

--publicurl=http://aioX:8004/v1/%\(tenant\_id\)s \

--internalurl=http://aioX:8004/v1/%\(tenant\_id\)s \

--adminurl=http://aioX:8004/v1/%\(tenant\_id\)s

Similarly register a service and endpoint for heat-cfn.

keystone service-create --name=heat-cfn --type=cloudformation \

--description="Orchestration CloudFormation"

Note: copy the service id and use it create the service end point

keystone endpoint-create \

--service-id=$(keystone service-list | awk '/ cloudformation / {print $2}') \

--publicurl=http://aioX:8000/v1 \

--internalurl=http://aioX:8000/v1 \

--adminurl=http://aioX:8000/v1

# Configure Heat Service

Edit /etc/heat/heat.conf

**Step 7:**

vi /etc/heat/heat.conf

Change the following in [DEFAULT], [keystone\_authtoken], [ec2authtoken]

and [database] sections

[DEFAULT]

...

rabbit\_host = aio**X**

rabbit\_port = 5672

rabbit\_userid = guest

rabbit\_password = pass

verbose = True

log\_dir = /var/log/heat

[database]

connection = mysql://heat:pass@aio**X**/heat

[keystone\_authtoken]

auth\_host = aio**X**

auth\_port = 35357

auth\_protocol = http

auth\_uri = http://aio**X**:5000/v2.0

admin\_tenant\_name = service

admin\_user = heat

admin\_password = pass

[ec2authtoken]

auth\_uri = http://aio**X**:5000/v2.0

**Step 8:**

heat-manage db\_sync

service heat-api restart

service heat-api-cfn restart

service heat-engine restart

# Verify the Orchestration service installation

To verify that the Orchestration service is installed and configured correctly, make sure that your credentials are set up correctly in the demo-openrc.sh file. Source the file, as follows:

source openrc.sh

Create a test template in the test-stack.yml file with the following content:

heat\_template\_version: 2013-05-23

description: Test Template

parameters:

ImageID:

type: string

description: Image use to boot a server

NetID:

type: string

description: Network ID for the server

resources:

server1:

type: OS::Nova::Server

properties:

name: "Test server"

image: { get\_param: ImageID }

flavor: "m1.tiny"

networks:

- port: { get\_resource: server1\_port }

server1\_port:

type: OS::Neutron::Port

properties:

network\_id: { get\_param: NetID }

outputs:

server1\_private\_ip:

description: IP address of the server in the private network

value: { get\_attr: [ server1, first\_address ] }

Use the **heat stack-create** command to create a stack from this template:

NET\_ID=$(nova net-list | awk '/ private / { print $2 }')

heat stack-create -f test-stack.yml \

-P "ImageID= CirrOS 0.3.2;NetID=$NET\_ID" testStack

Verify that the stack was created successfully with the **heat stack-list**command:

heat stack-list

Lab 8a: Launch and manage stacks

OpenStack Orchestration is a service that you can use to orchestrate multiple composite cloud applications. This service supports use of both the Amazon Web Services (AWS) CloudFormation template format through both a Query API that is compatible with CloudFormation and the native OpenStack [Heat Orchestration Template (HOT)](http://docs.openstack.org/user-guide/content/dashboard_stacks.html) format through a REST API.

These flexible template languages enable application developers to describe and automate the deployment of infrastructure, services, and applications. The templates enable creation of most OpenStack resource types, such as instances, floating IP addresses, volumes, security groups, and users. The resources, once created, are referred to as stacks.

The template languages are described in [the Template Guide](http://docs.openstack.org/developer/heat/template_guide/index.html) in the [Heat developer documentation](http://docs.openstack.org/developer/heat/).

# Launch a stack

1. Log in to the dashboard, choose a project, and click **Stacks** in the **Orchestration** category on the **Projects** tab.
2. Click **Launch Stack**.
3. In the **Select Template** dialog box, specify the following values.

|  |  |
| --- | --- |
| **Template Source** | Choose the source of the template from the list. |
| **Template URL/File/Data** | Depending on the source that you selected, enter the URL, browse to the file location, or directly include the template. |
| **Environment Source** | Choose the source of the environment from the list. The environment files contain additional settings for the stack. |
| **Environment URL/File/Data** | Depending on the source that you selected, enter the URL, browse to the file location, or directly include the environment. |

1. Click **Next**.
2. In the **Launch Stack** dialog box, specify the following values.

|  |  |
| --- | --- |
| **Stack Name** | Enter a name to identify the stack. |
| **Creation Timeout (minutes)** | Specify the number of minutes that can elapse before the launch of the stack times out. |
| **Rollback On Failure** | Select this check box if you want if you want the service to roll back changes if the stack fails to launch. |
| **Password for user "demo"** | Specify the password that the default user uses when the stack is created. |
| **DBUsername** | Specify the name of the database user. |
| **LinuxDistribution** | Specify the Linux distribution that is used in the stack. |
| **DBRootPassword** | Specify the root password for the database. |
| **KeyName** | Specify the name of the key pair to use to log in to the stack. |
| **DBName** | Specify the name of the database. |
| **DBPassword** | Specify the password for the database. |
| **InstanceType** | Specify the flavor for the instance. |

1. Click **Launch** to create a stack.
2. The **Stacks** tab shows the stack.

After the stack is created, click on the stack name to see the following details:

**Topology**

The topology of the stack.

**Overview**

The parameters and details of the stack.

**Resources**

The resources used by the stack.

**Events**

The events related to the stack.

# Manage a stack

1. Log in to the dashboard, choose a project, and click **Stacks**.
2. Select the stack that you want to update.
3. Click **Change Stack Template**.
4. In the **Select Template** dialog box, select the new template source or environment source.
5. Click **Next**.

The **Update Stack Parameters** window appears.

1. Enter new values for any parameters that you want to update.
2. Click **Update**.

# Delete a stack

When you delete a stack, you cannot undo this action.

1. Log in to the dashboard, choose a project, and click **Stacks**.
2. Select the stack that you want to delete.
3. Click **Delete Stack**.
4. In the confirmation dialog box, click **Delete Stack** to confirm the deletion.

Lab 9: Telemetry Service Installation

The Telemetry module:

* Efficiently collets the metering data about the CPI and network costs.
* Collects data by monitoring notifications sent from servers or by polling the infrastructures
* Configures the type of collected data to meet various operating requirements. Accessing and inserting metering data through the REST API.
* Expands the framework to collect custom usage data by additional plug-ins.
* Produces signed metering messages that cannot be repudiated.

**Step 1:**

SSH to AIO node with the credentials in Lab access

Enter following command

sudo su -

source openrc.sh

# Telemetry Module Installation on AIO Node

Telemetry provides an API service that provides a collector and a range of disparate agents. Before you can install these agents on nodes such as the compute node, you must use this procedure to install the core components on the controller node.

**Step 2:**

apt-get install ceilometer-api ceilometer-collector ceilometer-agent-central ceilometer-alarm-evaluator ceilometer-alarm-notifier python-ceilometerclient

# Create Database for Telemetry Service

The Telemetry service uses a database to store information. Specify the location of the database in the configuration file. The examples use a MySQL database on the controller node:

**Step 3:**

mysql -u root -ppass

mysql> CREATE DATABASE ceilometer;

mysql> GRANT ALL PRIVILEGES ON ceilometer.\* TO 'ceilometer'@'localhost' IDENTIFIED BY 'pass';

mysql> GRANT ALL PRIVILEGES ON ceilometer.\* TO 'ceilometer'@'%' IDENTIFIED BY 'pass';

mysql> exit

**Step 4:**

By default, the Ubuntu packages create a SQLite database. Delete the ceilometer.sqlite file created in the /var/lib/ceilometer/ directory so that it does not get used by mistake.

rm /var/lib/ceilometer/ceilometer.sqlite

**Step 5:**

Create a ceilometer user that the Telemetry service uses to authenticate with the Identity Service. Use the service tenant and give the user the admin role:

keystone user-create --name=ceilometer --pass=pass --email=ceilometer@onecloud.com

keystone user-role-add --user=ceilometer --tenant=service --role=admin

# Define services and service endpoints

**Step 6:**

Register the Telemetry service with the Identity Service so that other OpenStack services can locate it. Use the keystone command to register the service and specify the endpoint:

keystone service-create --name=ceilometer --type=metering --description="Telemetry"

Create the service end point for the service

keystone endpoint-create \

--service-id=$(keystone service-list | awk '/ metering / {print $2}') \

--publicurl=http://aioX:8777/v1/%\(tenant\_id\)s \

--internalurl=http://aioX:8777/v1/%\(tenant\_id\)s \

--adminurl=http://aioX:8777/v1/%\(tenant\_id\)s

# Configure Ceilometer Service

**Step 7:**

Edit /etc/ceilometer/ceilometer.conf

vi /etc/ceilometer/ceilometer.conf

Change the following in [DEFAULT], [keystone\_authtoken], [publisher], [service\_credentials] and [database] sections

[DEFAULT]

...

rabbit\_host = aio**X**

rabbit\_password = pass

log\_dir = /var/log/ceilometer

[database]

connection = mysql://ceilometer:pass@aio**X**/ceilometer

[keystone\_authtoken]

auth\_host = aio**X**

auth\_port = 35357

auth\_protocol = http

auth\_uri = http://aio**X**:5000/v2.0

admin\_tenant\_name = service

admin\_user = ceilometer

admin\_password = pass

[publisher]

# Secret value for signing metering messages (string value)

metering\_secret = CEILOMETER\_TOKEN

[service\_credentials]

os\_auth\_url = http://aio**X**:5000/v2.0

os\_username = ceilometer

os\_tenant\_name = service

os\_password = CEILOMETER\_PASS

**Step 8:**

ceilometer-dbsync

service ceilometer-agent-central restart

service ceilometer-api restart

service ceilometer-collector restart

service ceilometer-alarm-evaluator restart

service ceilometer-alarm-notifier restart

# Install the Compute agent for Telemetry

Telemetry provides an API service that provides a collector and a range of disparate agents. This procedure details how to install the agent that runs on the compute node.

**Step 9:**

apt-get install ceilometer-agent-compute

Edit the /etc/nova/nova.conf file and add the following lines to the [DEFAULT] section:

[DEFAULT]

...

instance\_usage\_audit = True

instance\_usage\_audit\_period = hour

notify\_on\_state\_change = vm\_and\_task\_state

notification\_driver = nova.openstack.common.notifier.rpc\_notifier

notification\_driver = ceilometer.compute.nova\_notifier

Restart compute service

service nova-compute restart

Repeat the Step – 7 to configure in Compute node. Then

Restart the service with its new settings:

service ceilometer-agent-compute restart

# Configure the Image Service for Telemetry

**Step 10:**

To retrieve image samples, you must configure the Image Service to send notifications to the bus.

Edit /etc/glance/glance-api.conf and modify the [DEFAULT] section:

notification\_driver = messaging

rpc\_backend = rabbit

rabbit\_host = aioX

rabbit\_password = pass

Restart glance services

service glance-registry restart

service glance-api restart

# Add Block Storage service agent for Telemetry

**Step 11:**

To retrieve volume samples, you must configure the Block Storage service to send notifications to the bus.

Edit /etc/cinder/cinder.conf and add in the [DEFAULT] section on the controller and volume nodes:

control\_exchange = cinder

notification\_driver = cinder.openstack.common.notifier.rpc\_notifier

Restart the Block Storage services with their new settings.

On Controller:

service cinder-api restart

service cinder-scheduler restart

On Volume Node:

service cinder-volume restart

# Verify the Telemetry service installation

To test the Telemetry installation, download an image from the Image Service, and use the ceilometer command to display usage statistics.

source openrc.sh

Use the ceilometer meter-list command to test the access to Telemetry:

Lab 10: Install OpenStack with COI

http://docwiki.cisco.com/wiki/OpenStack:\_Icehouse\_Deployment\_Guide

Lab 11: Havate: OpenStack and UCS

https://github.com/Havate/havate-openstack

Thank you

