

# OpenStack Lab Guide

Version 2.06g- Ubuntu – December 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.

<b>OpenStack Lab Overview .....</b>	<b>7</b>
Lab Topology.....	7
OpenStack Labs – Flow .....	8
Lab Access.....	8
<b>Lab 1: Identity Service Installation .....</b>	<b>13</b>
Objective:.....	13
Basic Configuration.....	13
<b>OpenStack Package Repository Configuration .....</b>	<b>14</b>
Prerequisites for OpenStack .....	15
<b>Configuring NTP .....</b>	<b>15</b>
<b>MySQL .....</b>	<b>16</b>
Messaging Queue Sever .....	17
Identity Service Keystone .....	18
Keystone Database .....	18
Define users, tenants and roles .....	20
Define services and service endpoints .....	22
Verify the Identity service installation .....	23
<b>Lab 2: Image Service Installation .....</b>	<b>24</b>
Image Service Installation.....	25
Create Database for Image Service.....	25
User Creation .....	26
Configure Image Service .....	26
Define services and service endpoints .....	28
Importing Images:.....	29
<b>Lab 3: Compute Service Installation .....</b>	<b>30</b>

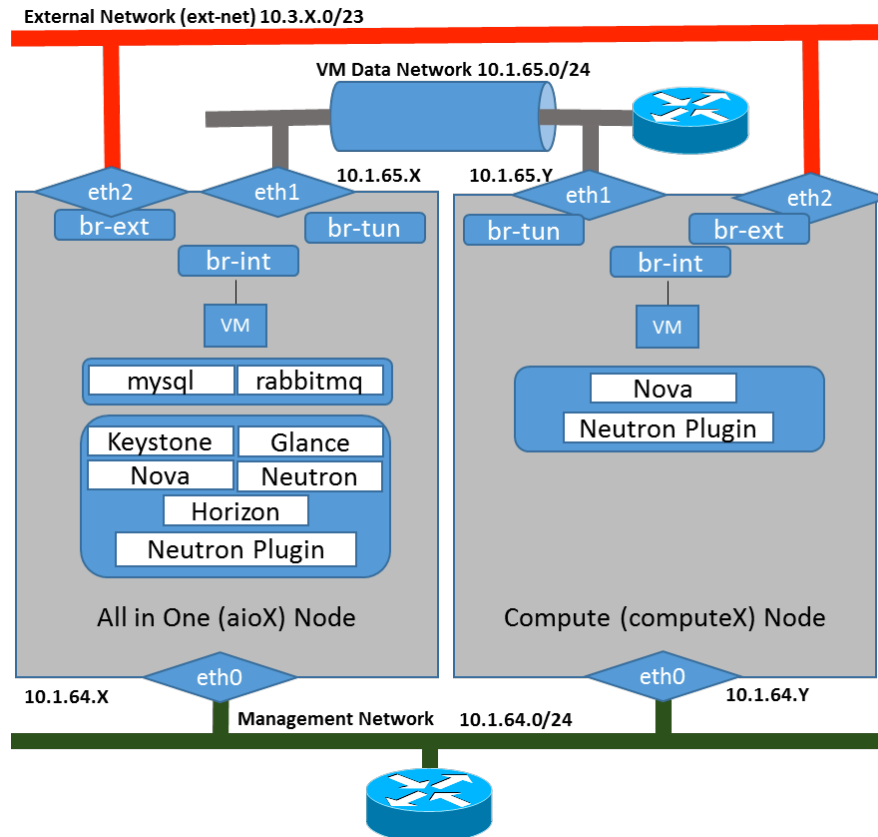
Compute Service Installation .....	31
Create Database for Compute Service.....	32
Configure Compute Service .....	33
Define services and service endpoints .....	34
<b>Lab 4: Network Service Neutron Installation .....</b>	<b>36</b>
Neutron Installation.....	37
Database Creation .....	38
Define services and service endpoints .....	39
Configure Neutron Service .....	40
<b>Lab 5: Adding Compute Node.....</b>	<b>45</b>
Basic Configuration.....	45
<b>Lab 6: Test Neutron/GRE tunnel.....</b>	<b>51</b>
Launch an Instance .....	51
<b>Lab 7: Block Storage Service Installation .....</b>	<b>54</b>
Cinder Installation on AIO Node .....	55
Create Database for Storage Service .....	55
Define services and service endpoints .....	56
Configure Cinder Service .....	58
Configure physical hard disks .....	59
Test Cinder Service .....	61
Attach Cinder Volume to an instance .....	62
<b>Lab 8: Horizon Dashboard Installation.....</b>	<b>65</b>
Install Horizon.....	65
<b>Lab 8a: Log in to the dashboard .....</b>	<b>67</b>
OpenStack dashboard—Project tab .....	67

OpenStack dashboard—Admin tab .....	69
<b>Lab 8b: Upload and manage image .....</b>	<b>72</b>
Upload an image.....	72
Update an image .....	73
Delete an image.....	74
<b>Lab 8c: Configure access and security for instances .....</b>	<b>74</b>
Add a rule to the default security group .....	75
Add a key pair .....	76
Import a key pair .....	76
Allocate a floating IP address to an instance.....	77
<b>Lab 8d: Launch and manage instances.....</b>	<b>78</b>
Launch an instance .....	78
Connect to your instance by using SSH .....	81
Track usage for instances .....	81
Create an instance snapshot .....	82
Manage an instance.....	82
<b>Lab 8e: Create and manage volumes .....</b>	<b>82</b>
Create a volume.....	83
Attach a volume to an instance .....	84
Detach a volume from an instance.....	84
Create a snapshot from a volume .....	84
Edit a volume .....	85
Delete a volume.....	85
<b>Lab 8f: Create and manage networks .....</b>	<b>86</b>
Create a network .....	86

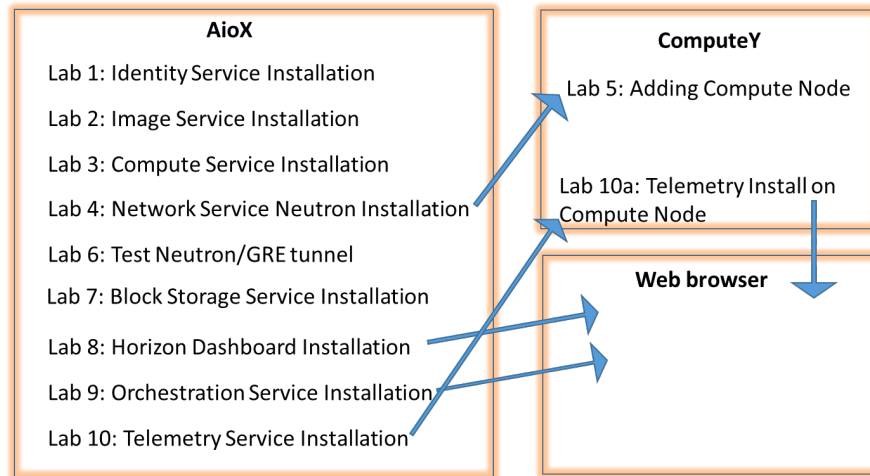
Create a router .....	87
<b>Lab 9: Orchestration Service Installation .....</b>	<b>89</b>
Heat Installation on AIO Node .....	89
Create Database for Orchestration Service .....	89
Define services and service endpoints .....	90
Configure Heat Service .....	91
Verify the Orchestration service installation .....	92
<b>Lab 10: Telemetry Service Installation .....</b>	<b>93</b>
Telemetry Module Installation on AIO Node.....	94
Create Database for Telemetry Service .....	94
Define services and service endpoints .....	95
Configure Ceilometer Service .....	96
Configure Compute agent for Telemetry.....	97
Configure the Image Service for Telemetry.....	97
Add Block Storage service agent for Telemetry.....	97
<b>Lab 10a: Telemetry Service Installation on Compute Node .....</b>	<b>98</b>
Install the Telemetry service on the compute node(ComputeY):.....	99
Configure Ceilometer Service .....	99
Verify the Telemetry service installation .....	100
<b>Lab 11: Install OpenStack with DevStack .....</b>	<b>102</b>

# OpenStack Lab Overview

## Lab Topology



## OpenStack Labs – Flow



## Lab Access

Download lab credentials in an Excel spreadsheet from the Github site provided in the lecture manual. Next, VNC to the POD machines with **VNC** credentials.

You will have an aioX node and ComputeY node installed with Ubuntu 14.04 as shown in the Topology diagram.

Initially you will connect to the aioX node. From **within the virtual desktop** you just connected to; use the local terminal program to SSH to the aioX machine using the provided aio credentials.

### aioX Node

eth0 IP Address: 10.1.64.X

Username: localadmin

Password: ubuntu

In lab 5, we will be accessing our ComputeY node in a similar way:

### computeY Node

eth0 IP Address: 10.1.64.Y

Username: localadmin

Password: ubuntu



**Once you are logged in as ‘localadmin,’ you are ready to start with Lab1.**

---

## How to debug issues in OpenStack

Part of the learning process- and the reality of working with OpenStack- is that problems do arise that require troubleshooting. Here we provide strategy and guidance about how to figure out where the problem might be.

NOTE: In guidance below, replace {service} with the name of the service of interest to you.

screen: screen is a useful tool as it allows you to have multiple “views” in a single terminal window, and lets you switch between them with a keystroke rather than opening multiple windows and point/clicking between them. in addition, screen can log the output of a terminal session to disk, making it useful for capturing the output of failures for later review.

Start screen and log the output:

```
screen -L
```

Once started, you can create a new screen with:

```
<ctrl>-a c
```

Which is hold the control key, and press a, release the control and a, and the press c

Move between the screens with

```
<ctrl>-a <space>
```

Screen does live within a single “view” of the terminal, so to look at the scrollback buffer, you have to

```
<ctrl>-a <esc>
```

***And then use the arrow keys on your keyboard to scroll up (<ctrl>-b scrolls back a page, <ctrl>-f scrolls forward)***

Get help for screen (many more capabilities):

```
<ctrl>-a ?
```

log files: There are numerous log files per service, and often the error will be exposed in the log file output. Each project will have a log directory in /var/log/{service} (for example /var/log/glance will have the log files for glance). Since services have different components that operate in similar but not always equivalent fashion, it’s best to use screen to open a

session per log file so you can watch the entire set of possible outputs to find the system that isn't responding appropriately. You can follow the logs with the "tail" command such as:

```
tail -f /var/log/{service}/{service}-api.log
```

When you are done following the file, you can exit out with:

```
<ctrl>-c
```

- Logging may be low level, look at .conf file (/etc/{service}/\*conf), turn on verbose and debug options. (e.g. etc/glance/glance.conf)
- Restart services if changing logging levels

Now that we know how to watch the logs and can see what's going on, we can try to spot the errors both in the .conf file (just by reviewing the actual lines typed in).

Usual issues are that AMQP isn't connected and eventually quits trying to connect (usually in the API log). The same is true for mysql mis-configurations. If an error is found it's best to restart all of the services in that project group (e.g. the entire nova processes) in case there's a non-obvious interaction with Rabbit or Mysql buried in the application components themselves.

You can check through the following configurations in /etc/{service}/{service}.conf

- the keystone API URI/URL components and port.
- the hostname provided for all the different services
- check target URI/URL components in .conf file
- validate endpoint in keystone. You can do this via the {service} cli tool.

```
glance --debug image-list
```

Should have an output something like:

```
$ glance --debug image-list
curl -i -X GET -H 'Accept-Encoding: gzip, deflate' -H 'Accept: */*'
-H 'User-Agent: python-glanceclient' -H 'Connection: keep-alive' -H
'X-Auth-Token: {SHA1}73459d07b6cf38f67503dc2ec9090c48c479c8b7' -H
'Content-Type: application/octet-stream' http://c240-1.onecloud:9292/v1/images/detail?sort\_key=name&sort\_dir=asc&limit=20
```

### REST Commands

The cli with --debug also prints out the correctly focused REST calls (with curl as the application already even) that you can generally pasted into your console to see what's happening.

### Rapidly restarting services

If you need to restart all of the nova or neutron services, you can do so quickly on Ubuntu as follows:

```
for name in `(cd /etc/init; ls *{service}' | cut -d. -f1)`; do
service ${name} restart; done
```

# Lab 1: Identity Service Installation

---

## Objective:

- Install prerequisites for OpenStack installation in AIO node
  - ntp
  - mysql
  - OpenStack Repository configuration
  - Messaging queue server - RabbitMQ Server
- Configuration and verification 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 aioX node as shown below. Only eth0 should have a gateway/dns configuration

```
eth0 10.1.64.X
```

```
eth1 10.1.65.X
```

The file should look like:

```
auto eth0
iface eth0 inet static
    address 10.1.64.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.X
    netmask 255.255.255.0
auto eth2
iface eth2 inet manual
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down
```

### 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.X	aioX.onecloud	aioX
10.1.64.Y	computeY.onecloud	computeY
10.1.64.1	gw.onecloud	gw

## OpenStack Package Repository Configuration

### Step 3:

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

Execute the below commands to prep your system for the OpenStack install.

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

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

### Step 4:

Type following command to restart network interfaces

```
ifdown eth1; ifup eth1; ifdown eth2; ifup eth2
```

```
ifdown: interface eth1 not configured
```

```
ifdown: interface eth2 not configured
```

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

## Prerequisites for OpenStack

### Configuring NTP

#### Step 5:

You must install NTP to properly synchronize services among OpenStack nodes (compute, controller and network).

In the lab we have a NTP server running on “gw.onecloud” which will be providing a reference clock for the nodes.

```
ntpdate gw.onecloud
```

Note: It might take 15-20 seconds for it to complete.

```
24 Aug 22:17:36 ntpdate[1163]: adjust time server 10.1.64.1  
offset 0.000462 sec
```

```
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
```

remote	refid	st	t	when	poll	reach	delay	offset	jitter
li506-17.member	209.51.161.238	2	u	60	64	1	74.016	15.343	0.000
sola-dal-09.ser	184.173.173.205	3	u	59	64	1	41.609	13.524	0.000
tock5.usshc.com	.GPS.	1	u	58	64	1	59.961	15.893	0.000
199.96.82.197.r	132.163.4.103	2	u	57	64	1	54.124	13.980	0.000
<b>golem.canonical</b>	<b>192.93.2.20</b>	<b>2</b>	<b>u</b>	<b>56</b>	<b>64</b>	<b>1</b>	<b>139.331</b>	<b>17.047</b>	<b>0.000</b>
gw.onecloud	91.189.94.4	3	u	55	64	1	1.159	14.705	0.000

## MySQL

### Step 6:

Most of the OpenStack services require a database to store information. In the lab we will be using MySQL.

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

Installation will prompt for password, **enter pass as root password**. 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 7:

Edit /etc/mysql/my.cnf and set the bind-address to the IP address of AIO node.

```
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.X from its default of 127.0.0.1 so that remote services can access they Mysql service.

```
bind-address = 10.1.64.X
```

Add the following lines to my.cnf below the bind-address

```
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```



### Step 8:

Restart the MySQL service to apply the changes.

```
service mysql restart
```

The below command initializes the MySQL data directory and creates the system table etc.

```
mysql_install_db
```

“mysql\_secure\_installation” is a script available to improve the security of your MySQL installation like removing anonymous accounts and control access to root accounts.

```
mysql_secure_installation
```

Select defaults (press Enter) for all prompts, yes to keep pass word the same, no to subsequent questions.

Note: The log files are mysql are placed in /var/log/mysql.

## Messaging Queue Sever

### Step 9:

OpenStack uses a message broker to coordinate operations and status information among services. The message broker service typically runs on the controller node which in our case is the AIO node. OpenStack supports several message brokers including RabbitMQ, Qpid, and ZeroMQ.

For this lab we will be using rabbitmq as the message broker.

Execute the below command to install the rabbitmq-server package.

```
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
```

Note: The log file for rabbitmq server is at “/var/log/rabbitmq”

## Identity Service Keystone

### Step 11:

Install the OpenStack identity service on the AIO node.

```
apt-get install keystone -y
```

### Step 12:

The Identity Service uses a database to store information. We will be using MySQL as the database as per Step 8. The username service keystone will use will be “keystone” and needs to be specified in the /etc/keystone.conf file.

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@aioX/keystone
```

Press **Esc** key

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

### Step 13:

By default, the Ubuntu packages creates 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
```

Note: If you execute the command “show databases;” at the mysql prompt you will see the newly created keystone database.

### Step 15:

At this time, the database for keystone has been created but not tables have been populated.

Create the database tables for the Identity Service:

```
su -s /bin/sh -c "keystone-manage db_sync" keystone
```

Note: The above command created the tables. To see the tables created, at “mysql” prompt execute “use keystone; show tables;”

### 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 uncomment to 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:

**Note:** Change **X** with AIO node Number

```
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. Other services (like nova, glance etc) created later will be a user in the “service” tenant.

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

Property	Value
description	Admin Tenant
enabled	True
id	6c7ecac71357496fabf959f70e0681b3
name	admin

Note: The “id” will differ as it’s a unique ID which OpenStack assigns to every object

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

Property	Value
description	Service Tenant
enabled	True
id	4e05f27ffc7a474db3821ada26b7a5fa
name	service

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
```

Property	Value
email	admin@onecloud.com
enabled	True
id	ffd8ac42900746919a1d9c3307c53445
name	admin
username	admin

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
```

Property	Value
id	ec81d33f35484a538d5cb050db8177eb
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 user-role-add --user=admin --role=_member_ --
tenant=admin
```

Check the user and tenant list

```
keystone user-list
```

id	name	enabled	email
ffd8ac42900746919a1d9c3307c53445	admin	True	admin@onecloud.com

```
keystone tenant-list
```

id	name	enabled
----	------	---------

6c7ecac71357496fabf959f70e0681b3	admin	True
4e05f27ffc7a474db3821ada26b7a5fa	service	True

## Define services and service endpoints

One 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, the following commands are used:

- `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"
```

Property	Value
description	Keystone Identity Service
enabled	True
id	aa421d6640fc4f6b8e697a800f6515ea
name	keystone
type	identity



When you specify an endpoint, you provide URLs for the public API, internal API, and admin API where each of these URLs expose different or subset of the APIs.

Note: 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.

**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.

**Note:** Change X with AIO node Number

```
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
```

Property	Value
adminurl	http://aio51:35357/v2.0
id	c56a01e470d940eda883328b45d0be19
internalurl	http://aio51:5000/v2.0
publicurl	http://aio51:5000/v2.0
region	regionOne
service_id	aa421d6640fc4f6b8e697a800f6515ea

## 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 **openrc.sh** file with the admin credentials and admin endpoint.

```
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://aioX: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
```

id	name	enabled	email
ffd8ac42900746919a1d9c3307c53445	admin	True	admin@onecloud.com

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 have been installed successfully.

### For the inquisitive mind:

- Execute the command “keystone --debug user-list” and walk through the output.

## Lab 2: Image Service Installation



OpenStack Image Service (Glance) provides discovery, registration, and delivery services for disk and server images.

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 which makes the images 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 -
```

Execute the following command to source the environmental variables.

```
source ~/openrc.sh
```

Execute the following command to install the glance package and also the python glance client.

```
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 glance database by logging 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 which might give you the below error

```
rm: cannot remove /var/lib/glance/glance.sqlite: No such file or directory
```

## 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
```

Property	Value
email	glance@onecloud.com
enabled	True
id	6e652e807f934e1e893419165a9a788e
name	glance
username	glance

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

## Configure Image Service

The configuration files for glance are kept in “/etc/glance”.

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

```
rpc_backend = rabbit
rabbit_host = aioX
rabbit_userid = guest
rabbit_password = pass
```

### Step 6:

Configure the location of the database that was created in Step 2.

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

```
#sqlite_db = /var/lib/glance/glance.sqlite
connection = mysql://glance:pass@aioX/glance
```

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

To configure the Image Service to use the Identity Service for authentication, you need to specify the keystone end-point details.

Press **Down Arrow** key to Scroll down up to **[keystone\_authtoken]** section.

```
auth_uri = http://aioX:5000
auth_host = aioX
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 update the **[database]** and **[keystone\_authtoken]** sections 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 8:

Register the service and create the endpoint:

Registering a service with keystone is done with the “keystone service-create” command.

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

Property	Value
description	Glance Image Service
enabled	True
id	96d1583e720f4104855f624c3a2af733
name	glance
type	image

Creating an end-point for the service is done with the “keystone endpoint-create” command which takes the service id (generated by the “keystone service-create” command) as an argument.

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

**Note:** Change X with AIO node Number

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

Property	Value
adminurl	http://aio51:9292
id	9fb1114401334a3fbf7c2bdb7193558c
internalurl	http://aio51:9292
publicurl	http://aio51:9292
region	regionOne
service_id	96d1583e720f4104855f624c3a2af733

Note: With the above, the endpoint is listening on TCP port 9292

### Step 9:

Create the database tables for the Image Service:

Using the “db\_sync” argument with glance-manage, one populates the tables in the “glance” database which was created earlier.

```
su -s /bin/sh -c "glance-manage db_sync" glance
```

**Note:** If you get an error like “**CRITICAL glance [-] ValueError: Tables "migrate\_version" have non utf8 collation, please make sure all tables are CHARSET=utf8**” you can change the CHARSET in mysql by

```
mysql -uroot -ppass
```

```
use glance;
```

```
alter table migrate_version convert to character set utf8 collate utf8_unicode_ci;
```

```
flush privileges;
```

```
quit
```

and then executing the “glance-manage db\_sync” command given above.

## Step 10:

Restart the glance service with its new settings.

```
service glance-registry restart
```

```
service glance-api restart
```

You can validate the service running by executing “ps -ef | grep glance” to see the service running.

## Importing Images:

### Step 11:

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 12:

Import 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:

Property	Value
checksum	64d7c1cd2b6f60c92c14662941cb7913
container_format	bare
created_at	2014-08-25T00:11:02
deleted	False
deleted_at	None
disk_format	qcow2
id	ddl193c0-e365-41e1-b879-89b3eedb4575
is_public	True
min_disk	0
min_ram	0
name	CirrOS 0.3.2
owner	6c7ecac71357496fabf959f70e0681b3
protected	False
size	13167616
status	active
updated_at	2014-08-25T00:11:02
virtual_size	None

Notice the image metadata which is set by the Image creator. This metadata information influences the scheduling of compute resources.

```
glance image-list
```

ID	Name	Disk Format	Container Format	Size	Status
ddl193c0-e365-41e1-b879-89b3eedb4575	CirrOS 0.3.2	qcow2	bare	13167616	active

```
cd ..
```

Image Service configured and image upload successfully.

**For the inquisitive mind:**

- Navigate to “/var/lib/glance/images” and execute “ls” and “file \*”. Why is that file there and what type is it?
- Execute “glance --debug image-list” and walkthrough the output.
- What all OpenStack services do we have running now? Execute “keystone service-list or endpoint-list” to find out.
- Explore the log files in the “/var/log/glance” directory.

## Lab 3: Compute Service Installation

The Compute service is a cloud computing fabric controller, which is the main part of an IaaS system.

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

### Step 1:

```
sudo su -  
  
source ~/openrc.sh
```

### Install Compute Controller Service packages

### Step 2:

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

```
apt-get install nova-api nova-cert nova-conductor nova-  
consoleauth nova-novncproxy nova-scheduler python-novaclient -y
```

nova-api: Accepts and responds to end user compute API calls.

nova-cert: Manages x509 certificates

nova-conductor: Enables OpenStack to function without compute nodes accessing the database

nova-consoleauth : Authorizes tokens for users that console proxies provide.

nova-novncproxy: Provides a proxy for accessing running instances through a VNC connection using a web browser.

nova-scheduler: determines how to dispatch compute and volume requests.

python-novaclient: Client library for OpenStack Compute API.

## Install Compute Node packages

### Step 3:

Install the appropriate packages for the Compute service as we will be using KVM as the hypervisor.

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

nova-compute-kvm : the compute package for KVM as the hypervisor

python-guestfs: Guest disk image management library.

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 4:

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 5:

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 6:

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
```

Property	Value
email	nova@onecloud.com
enabled	True
id	0b3c90d65d514fce813666320cb7b103
name	nova
username	nova

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

## Configure Compute Service

### Update nova.conf

#### Step 7:

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.

The message broker as you recall is running on the AIO node.

```
vif_plugging_is_fatal=false
vif_plugging_timeout=0
rpc_backend =rabbit
rabbit_host=aioX
rabbit_password=pass
```

Add the my\_ip, vncserver\_listen, and vncserver\_proxyclient\_address configuration options to the **[DEFAULT]** section: To get VNC access from the POD machine enter IP address instead of hostname.

```
my_ip=10.1.64.X
vnc_enabled = True
vncserver_listen=10.1.64.X
vncserver_proxyclient_address=10.1.64.X
novncproxy_base_url=http://10.1.64.X:6080/vnc_auto.html

auth_strategy=keystone
glance_host=aioX
```

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

Note: You will have to create these sections as they do not exist in the default nova.conf.

```
[database]
connection = mysql://nova:pass@aioX/nova

[keystone_authtoken]
auth_uri = http://aioX:5000
auth_host = aioX
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = pass
```

The database section specifies the database to use which in this case is MySQL running on the AIO node.

The keystone\_authtoken specifies the keystone service end-point and the credentials that nova will use to authenticate with.

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

### **Edit Nova-Compute.Conf**

#### **Step 8:**

In this lab, the controller (AIO node) will also be a compute node. A compute node is a node which hosts virtual machine instances and it runs the nova-compute daemon.

In this case we will be using QEMU as the hypervisor. Edit the [libvirt] section in the /etc/nova/nova-compute.conf

**vi /etc/nova/nova-compute.conf**

```
virt_type=qemu
```

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

## **Define services and service endpoints**

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

#### **Step 9:**

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

Property	Value
description	Nova Compute service
enabled	True
id	3c85f9ccd55d4485a4096ff4a014e593
name	nova
type	compute

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

**Note:** Change X with AIO node Number

```
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
```

Property	Value
adminurl	http://aio51:8774/v2/%(tenant_id)s
id	5c1c4cada2ea41299c53a31df57be562
internalurl	http://aio51:8774/v2/%(tenant_id)s
publicurl	http://aio51:8774/v2/%(tenant_id)s
region	regionOne
service_id	3c85f9ccd55d4485a4096ff4a014e593

## Step 10:

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
```

Compute Service is installed successfully.

For the inquisitive mind:

- To see the list of physical hosts used by nova execute “nova host-list”
- Execute “nova --debug service-list” and walkthrough the output.
- What all OpenStack services do we have running now? Execute “keystone service-list or endpoint-list” to find out.
- What does “nova hypervisor-list” and “nova image-list” tell you?
- How will you find out the nova flavors defined?
- Explore the log files in the “/var/log/nova” directory.

## Lab 4: 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 different technologies to create the virtual networks: VLANs, GRE or VXLAN. 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:

Install the following packages for neutron.

```
apt-get install neutron-server neutron-plugin-ml2 -y
```

neutron-server: exposes the Neutron API, and passes all webservice calls to the Neutron plugin for processing.

neutron-plugin-ml2: The Modular Layer 2 (ml2) plugin is a framework allowing OpenStack Networking to simultaneously utilize the variety of layer 2 networking technologies which currently works with the existing openvswitch, linuxbridge, and hyperv L2 agents, and is intended to replace and deprecate the monolithic plugins associated with those L2 agents.

Additional packages which are required -

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

neutron-dhcp-agent: Provides instances with IP addresses.

neutron-plugin-openvswitch-agent: Plugin agent for the Open vSwitch.

neutron-l3-agent: Allows tenants to create routers using the Linux IP stack and iptables to perform L3 forwarding and NAT. In order to support multiple routers and potentially overlapping address space, it utilizes the network namespace feature of the kernel.

(Optional) openvswitch-datapath-dkms: Ubuntu installations using Linux kernel version 3.11 or newer do not require the openvswitch-datapath-dkms package. This package provides the Open vSwitch datapath module source code that is needed by openvswitch-switch.

## 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 Neutron with password as pass

### Step 4:

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

Property	Value
email	neutron@onecloud.com
enabled	True
id	249487a4198d447389cd050b109bf446
name	neutron
username	neutron

```
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"
```

Property	Value
description	OpenStack Networking Service
enabled	True
id	f5fafa686097469bb31f8cc77fd9b51f
name	neutron
type	network

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

**Note:** Change X with AIO node Number

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

Property	Value
adminurl	http://aio51:9696
id	e79afffbbe7645478e444840b1af76e9
internalurl	http://aio51:9696
publicurl	http://aio51:9696
region	regionOne
service_id	f5fafa686097469bb31f8cc77fd9b51f

### Step 6:

Certain kernel level networking functions need to be enabled for coordination of traffic for the VMs.

Edit the `/etc/sysctl.conf` file, as follows:

```
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 which will return the above values saved for ipv4 to confirm the changes made -

```
sysctl -p
```

## Configure Neutron Service

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

**Step 7:**

```
vi /etc/neutron/neutron.conf
```

```
[DEFAULT]
service_plugins = router
auth_strategy = keystone
allow_overlapping_ips = True
```

Note: If you look for "core\_plugin" in the file you will find that the ml2 plugin is the default.

Also, the below need to be specified in the same file -

```
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = aioX
rabbit_password = pass

[keystone_authtoken]
auth_uri = http://aioX:5000
auth_host = aioX
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = pass

[database]
connection = mysql://neutron:pass@aioX/neutron
```

Kombu is a messaging library for Python. The aim of Kombu is to make messaging in Python easy by providing a high-level interface for the AMQ protocol.



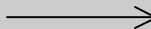
Save and exit the file.

### Step 8:

Obtain the service tenant identifier (id) with following command and fill it in neutron.conf

**keystone tenant-get service**

Property	Value
description	Service Tenant
enabled	True
id	d1f0515ee4174c9ca4e6f73f13f0da2f
name	service



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

```
[DEFAULT]
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://aioX:8774/v2
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID (Get it from the output of keystone tenant-get service)
nova_admin_password = pass
nova_admin_auth_url = http://aioX:35357/v2.0
```

### Step 9:

By default to provide DHCP services on the software-defined networks neutron uses dnsmasq which is a light weight DNS forwarder and a DHCP server.

In our case the DHCP agent is also on the AIO node so the configuration file for it needs to be modified to reflect the desired settings.

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

**vi /etc/neutron/dhcp\_agent.ini**

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

### Step 10:

The L3 agent configuration is stored in the “/etc/neutron/l3\_agent.ini” file.

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

```
vi /etc/neutron/l3_agent.ini
```

```
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True
```

### Step 11:

The ML2 plugin configuration is stored in the “/etc/neutron/plugins/ml2/ml2\_conf.ini” file and it specifies things like what network types (vlan, gre etc) or mechanism drivers (for specific hardware for example) need to be used.

We will be using GRE as the encapsulation.

```
vi /etc/neutron/plugins/ml2/ml2_conf.ini
```

Add the following keys to the [ml2], [ml2\_type\_gre] section:

```
[ml2]
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch

[ml2_type_gre]
tunnel_id_ranges = 1:1000

[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
enable_security_group = True

[ovs]
local_ip = eth1_INTERFACE_IP_ADDRESS #replace text with IP address
tunnel_type = gre
enable_tunneling = True
```

### Step 12:

Metadata service allows a VM instance to retrieve instance specific data like SSH keys, startup scripts etc. The request is received by the Metadata agent and is relayed to nova.

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

```
vi /etc/neutron/metadata_agent.ini
```

```
[DEFAULT]
auth_url = http://aioX:5000/v2.0
```

```
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = pass
nova_metadata_ip = aioX
metadata_proxy_shared_secret = pass
```

### Step 13:

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.

**vi /etc/nova/nova.conf**

Add following to the [DEFAULT] section:

```
[DEFAULT]
service_neutron_metadata_proxy = true
neutron_metadata_proxy_shared_secret = pass

network_api_class=nova.network.neutronv2.api.API
neutron_url=http://aioX:9696
neutron_auth_strategy=keystone
neutron_admin_tenant_name=service
neutron_admin_username=neutron
neutron_admin_password=pass
neutron_admin_auth_url=http://aioX:35357/v2.0
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSIInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
security_group_api = neutron
```

### Step 14:

**service nova-api restart**

**service nova-scheduler restart**

**service nova-conductor restart**

**service neutron-plugin-openvswitch-agent restart**

**service neutron-l3-agent restart**

**service neutron-dhcp-agent restart**

**service neutron-metadata-agent restart**

```
service openvswitch-switch restart
```

We need to add the integration bridge (this connects to the VMs) and the external bridge (this connects to the outside world), called br-int and br-ex, respectively. br-int already exists so we will create br-ex.

```
ovs-vsctl add-br br-ex
```

Now we add a port (eth2) to the external bridge br-ex

```
ovs-vsctl add-port br-ex eth2
```

### Step 15:

Restart Services

```
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
```

```
service nova-compute restart
```

Check all the Neutron agents are working

```
neutron agent-list
```

id	agent_type	host	alive	admin_state_up
05d590f8-68a6-4803-8f9c-2820e37454f8	Open vSwitch agent	aio51	:-)	True
2bf643e6-2114-4187-ae33-04c9ec435d05	Metadata agent	aio51	:-)	True
592f2bbc-4e9f-4517-adaa-33cc10a6f62d	L3 agent	aio51	:-)	True
5b332f0a-f9f7-4bc3-8d59-76258cd5a3a6	DHCP agent	aio51	:-)	True

Note: If the agents are not shown as “alive” then look at the log files in “/var/log/neutron” for errors.

### Step 16:

Create a Tenant network:

```
neutron net-create private-net
```

Create a new Subnet for Tenant network:

```
neutron subnet-create private-net --name private-subnet  
10.10.10.0/24
```

Neutron has been installed successfully in AIO node.

#### For the inquisitive mind:

- What all OpenStack services do we have running now? Execute “keystone service-list or endpoint-list” to find out.
- Execute “neutron --debug subnet-list” and walk through the output.
- Execute “ps -ef | grep neutron” to check the processes.
- Explore the log files in the “/var/log/neutron” directory.

## Lab 5: Adding Compute Node

---

We are now going add ComputeY as a compute node to the OpenStack install. Please see the lab topology diagram and the beginning of the guide to see the network adapters the ComputeY node will have.

### Basic Configuration

SSH to ComputeY node with the credentials provided earlier.

#### Step 1:

```
sudo su -
```

```
vi /etc/network/interfaces
```

Enter the network details for the ComputeY node as shown below.

```
auto eth0  
iface eth0 inet static  
    address 10.1.64.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.Y
    netmask 255.255.255.0
auto eth2
iface eth2 inet manual
    up ip link set dev $IFACE up
    down ip link set dev $IFACE down
```

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.X    aioX.onecloud    aioX
10.1.64.Y    computeY.onecloud    computeY
```

### 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
```

Enter the below command to confirm changes made.

**sysctl -p**

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.accept_ra = 0
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.all.accept_ra = 0
```

Type following command to restart network service

**ifdown eth1; ifup eth1; ifdown eth2; ifup eth2**

**ifdown: interface eth1 not configured**

**ifdown: interface eth2 not configured**

Note: it is ok if the system complains about interfaces not being configured in the previous step. You can execute “ifconfig -a” to check interface IP address assignments.

#### Step 4:

Install NTP Package in Compute Node

```
ntpdate gw.onecloud
```

```
25 Aug 19:01:19 ntpdate[1837]: adjust time server 10.1.64.1 offset 0.026366 sec
```

Note: It might take 15-20 seconds for the command to complete

```
apt-get install -y ntp
```

```
echo "server gw.onecloud iburst" > /etc/ntp.conf
```

```
service ntp restart
```

Execute the below command to update the OS distribution

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

Note, if doing this **outside the lab**, you'll also want to do the following to add the Ubuntu cloud archives:

```
# add-apt-repository cloud-archive:icehouse
```

#### Step 5:

Install Nova Hypervisor and Network plugins on the ComputeY node.

Note: The below is very similar to what we did on the AIO node as the AIO node is also a compute node.

```
apt-get install -y neutron-common neutron-plugin-ml2 neutron-  
plugin-openvswitch-agent
```

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

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

#### Step 6:

Create openrc.sh file

```
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://aioX: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 = rabbit  
rabbit_host = aioX  
rabbit_password = pass  
  
my_ip=10.1.64.Y  
vnc_enabled=True  
vncserver_listen=0.0.0.0  
vncserver_proxyclient_address=10.1.64.Y  
novncproxy_base_url=http://10.1.64.X:6080/vnc_auto.html  
  
glance_host=aioX  
  
#networking  
network_api_class = nova.network.neutronv2.api.API  
neutron_url = http://aioX:9696  
neutron_auth_strategy = keystone  
neutron_admin_tenant_name = service  
neutron_admin_username = neutron  
neutron_admin_password = pass  
neutron_admin_auth_url = http://aioX:35357/v2.0  
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSIfaceDriver  
firewall_driver = nova.virt.firewall.NoopFirewallDriver  
security_group_api = neutron  
  
[database]
```



```
connection = mysql://nova:pass@aioX/nova
```

### Step 8:

Edit the [libvirt] section in the /etc/nova/nova-compute.conf

```
vi /etc/nova/nova-compute.conf
```

```
virt_type=qemu
```

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

### Step 9:

Edit the file /etc/neutron/neutron.conf to specify the rabbit\_host, keystone and the database information.

```
vi /etc/neutron/neutron.conf
```

```
auth_strategy = keystone
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = aioX
rabbit_password = pass

[keystone_authtoken]
auth_uri = http://aioX:35357/v2.0
auth_host = aioX
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = pass

[database]
connection = mysql://neutron:pass@aioX/neutron
```

### Step 10:

The ML2 plug-in uses the Open vSwitch (OVS) mechanism (agent) to build the virtual networking framework for instances.

Edit the /etc/neutron/plugins/ml2/ml2\_conf.ini file:

Add the following keys to the [ml2] section:

```
[ml2]
type_drivers = gre
```

```

tenant_network_types = gre
mechanism_drivers = openvswitch

[ml2_type_gre]
tunnel_id_ranges = 1:1000

[ovs]
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS #replace with eth1 IP address
tunnel_type = gre
enable_tunneling = True

[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
enable_security_group = True

```

### Step 11:

Create a bridge for internal communication and restart the services

```
service nova-compute restart
```

```
service openvswitch-switch restart
```

```
service neutron-plugin-openvswitch-agent restart
```

### Step 12:

Source the openrc.sh file

```
source ~/openrc.sh
```

### Step 13:

Type following command and check the new Compute node is listed

```
nova service-list
```

Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
nova-cert	aio51	internal	enabled	up	2014-08-25T19:25:46.000000	-
nova-consoleauth	aio51	internal	enabled	up	2014-08-25T19:25:46.000000	-
nova-scheduler	aio51	internal	enabled	up	2014-08-25T19:25:44.000000	-
nova-conductor	aio51	internal	enabled	up	2014-08-25T19:25:48.000000	-
nova-compute	aio51	nova	enabled	up	2014-08-25T19:25:46.000000	-
<b>nova-compute</b>	<b>compute61</b>	<b>nova</b>	<b>enabled</b>	<b>up</b>	2014-08-25T19:25:48.000000	-

```
neutron agent-list
```

id	agent_type	host	alive	admin_state_up
431efefa-a00c-485a-b788-a560f6f73134	Open vSwitch agent	aio51	:-)	True
6e6a914e-f2cb-43a4-9014-57bd9222af17	DHCP agent	aio51	:-)	True
9865554f-bce0-4b5e-aed6-88626a24a9ec	L3 agent	aio51	:-)	True
a4850201-79ba-4e7a-aeb1-05cfc3bd8695	Metadata agent	aio51	:-)	True
ae466dc9-93b2-4b35-816d-7b97d60da398	Open vSwitch agent	compute61	:-)	True

## ovs-vsctl show

```
Bridge br-int
    fail_mode: secure
    Port br-int
        Interface br-int
            type: internal
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
Bridge br-tun
    Port br-tun
        Interface br-tun
            type: internal
    Port "gre-0a000205"
        Interface "gre-0a000205"
            type: gre
            options: {in_key=flow, local_ip="10.0.2.4", out_key=flow, remote_ip="10.0.2.5"}
    Port patch-int
        Interface patch-int
            type: patch
            options: {peer=patch-tun}
ovs_version: "2.0.1"
```

The output shows GRE, local\_ip and remote\_ip.

ComputeY is successfully added to AIO node as a Compute Node.

# Lab 6: Test Neutron/GRE tunnel

## Launch an Instance

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

When you launch a virtual machine, you can inject a key pair, which provides SSH access to your instance. These keys are injected into the instances to make password-less SSH access to the instance. If the key pair is generated with an external tool, you can import it into OpenStack.

If an image uses a static root password or a static key set—neither is recommended—you must not provide a key pair when you launch the instance.

### Step 1:

## Generate a keypair that consists of a private and public key

On the AIO node, execute `ssh-keygen` to generate the keypair and choose default values for all prompts

### `ssh-keygen`

```
root@aio51:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
b4:d7:7f:6c:25:4f:19:dc:9f:6f:7c:ed:90:a8:d1:fd root@aio51
The key's randomart image is:
+--[ RSA 2048 ]-----+
|                      |
|      . .             |
|      .   o.          |
|      . . .   =       |
|      S . . .o+       |
|      . . + B+        |
|      . o = @         |
|      o   B.          |
|      .    E          |
+-----+

```

Add the key to the system as mykey

```
nova keypair-add --pub_key ~/.ssh/id_rsa.pub mykey
```

You have just created a keypair named “mykey”. 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
```

```
+-----+-----+
| Name | Fingerprint |
+-----+-----+
| mykey | b4:d7:7f:6c:25:4f:19:dc:9f:6f:7c:ed:90:a8:d1:fd |
+-----+-----+

```

## Step 2:

Security groups and security group rules allows administrators and tenants the ability to specify the type of traffic and direction (ingress/egress) that is allowed to pass through a port.

To test connectivity to the VMs (using ping and ssh), modify the security group rule named “default”.

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

IP Protocol	From Port	To Port	IP Range	Source Group
tcp	22	22	0.0.0.0/0	

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

IP Protocol	From Port	To Port	IP Range	Source Group
icmp	-1	-1	0.0.0.0/0	

### Step 3:

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 the instance is allocated.

To see a list of the available profiles:

```
nova flavor-list
```

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
1	m1.tiny	512	1	0		1	1.0	True
2	m1.small	2048	20	0		1	1.0	True
3	m1.medium	4096	40	0		2	1.0	True
4	m1.large	8192	80	0		4	1.0	True
5	m1.xlarge	16384	160	0		8	1.0	True

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

```
nova image-list
```

ID	Name	Status	Server
dd1193c0-e365-41e1-b879-89b3eedb4575	Cirros 0.3.2	ACTIVE	

#### Step 4:

Create one VM in each host

**Note:** Change **X** and **Y** Values with your AIO and Compute nodes

```
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**

ID	Name	Status	Task State	Power State	Networks
8ca061ea-f1f1-4cd4-a1cf-7b0be9a56561	test-aio	ACTIVE	-	Running	private=172.16.0.2
b4b0cba6-bc3f-4266-b5be-1894683fc835	test-compute	ACTIVE	-	Running	private=172.16.0.7

Note the IP address of the VMs

Get the VM's VNC console URL by following command

**nova get-vnc-console test-aio novnc**

Type	Url
novnc	http://10.1.64.151:6080/vnc_auto.html?token=045600c3-297d-49f9-a9da-3d7b8b804639

Paste this URL in a web browser and login to the vm console with the user name and password shown in the console of Cirros VM.

Ping the other VM and confirm that GRE tunnel has been established.

## Lab 7: Block Storage Service Installation

The Block Storage Service (Cinder) enables management of volumes, volume snapshots, and volume types. The Block Storage Service interacts with Compute to provide volumes for instances.

### **Step 1:**

SSH to AIO node with the credentials in Lab access

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

```
sudo su -
```

```
source ~/openrc.sh
```

## **Cinder Installation on AIO Node**

In a multi node environment install following OpenStack Block Storage services on the Controller (AIO) node. The Storage node contains the disk that will serve volumes.

### **Step 2:**

You can configure OpenStack to use various back end storage systems. In this lab we will be using LVM (Logical Volume Manager) as the storage backend. The LVM backend implements block storage as LVM logical partitions.

Install the appropriate packages via apt-get.

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

cinder-api - Responsible for receiving and handling the request.

cinder-scheduler - Determines the volume server which will service the request.

```
apt-get install -y lvm2
```

lvm2 - Provides logical volume management facilities on linux.

```
apt-get install -y cinder-volume
```

cinder-volume - Runs on the storage node and manages the storage space.

## **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
```

Property	Value
email	cinder@onecloud.com
enabled	True
id	98cde01da3044f18a6f3cf3ccf7babaa
name	cinder
username	cinder

```
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"
```

Property	Value
description	OpenStack Block Storage
enabled	True
id	a934854782f6481ba5b5ffe479a0e9cc
name	cinder
type	volume

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

**Note:** Change X with AIO node Number

```
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
```

Property	Value
adminurl	http://aio51:8776/v1/%(tenant_id)s
id	055f81541eb5474ba46b75e3a7883a70
internalurl	http://aio51:8776/v1/%(tenant_id)s
publicurl	http://aio51:8776/v1/%(tenant_id)s
region	regionOne
service_id	a934854782f6481ba5b5ffe479a0e9cc

Similarly register a service and endpoint for version 2 of the Block Storage service API. Multiple end points can exist for a single service where some clients could use v1 and others v2.

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

Property	Value
description	OpenStack Block Storage v2
enabled	True
id	3d7606abdd134e68bc81964796f43807
name	cinderv2

type	volumev2

**Note:** Copy the Service id to use in endpoint-create command. Or we can use keystone service-list | awk '/ volumev2 / {print \$2}' to get the service id of type volumev2.

**Note:** Change X with AIO node Number

```
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
```

Property	Value
adminurl	http://aio51:8776/v2/%(tenant_id)s
id	b9d142c8346143189eb03455fdddf3bb
internalurl	http://aio51:8776/v2/%(tenant_id)s
publicurl	http://aio51:8776/v2/%(tenant_id)s
region	regionOne
service_id	3d7606abdd134e68bc81964796f43807

## 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 = aioX
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = pass
glance_host = aioX

[keystone_authtoken]
auth_uri = http://aioX:5000
auth_host = aioX
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
```

```
admin_user = cinder
admin_password = pass

[database]
connection = mysql://cinder:pass@aioX/cinder
```

## Configure physical hard disks

### Step 8:

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

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

```
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 1.20612 s, 86.9 MB/s
```

```
fdisk -l
```

```
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000cb3c6
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	2048	499711	248832	83	Linux
/dev/sda2		501758	41940991	20719617	5	Extended
/dev/sda5		501760	41940991	20719616	8e	Linux LVM

```
Disk /dev/sdb: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

```
Disk /dev/sdb doesn't contain a valid partition table
```

```
Disk /dev/mapper/aio51--vg-root: 16.9 GB, 16903045120 bytes
255 heads, 63 sectors/track, 2055 cylinders, total 33013760 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

```

Disk /dev/mapper/aio51--vg-root doesn't contain a valid partition
table

Disk /dev/mapper/aio51--vg-swap_1: 4290 MB, 4290772992 bytes
255 heads, 63 sectors/track, 521 cylinders, total 8380416 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/mapper/aio51--vg-swap_1 doesn't contain a valid partition
table

```

Select the second disk (/dev/sdb) to create physical volume and cinder volume group.

```
pvcreate /dev/sdb
```

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

```
Volume group "cinder-volumes" successfully created
```

```
pvdisplay /dev/sdb
```

```

--- Physical volume ---
PV Name                /dev/sdb
VG Name                cinder-volumes
PV Size                20.00 GiB / not usable 4.00 MiB
Allocatable            yes
PE Size                4.00 MiB
Total PE               5119
Free PE                5119
Allocated PE           0
PV UUID                33ZiCg-9fR8-YucF-dVZe-l477-Coba-S8JFGa

```

```
vgdisplay cinder-volumes
```

```

--- Volume group ---
VG Name                cinder-volumes
System ID
Format                 lvm2
Metadata Areas          1
Metadata Sequence No    1
VG Access               read/write
VG Status               resizable

```

```

MAX LV          0
Cur LV         0
Open LV         0
Max PV          0
Cur PV         1
Act PV          1
VG Size         20.00 GiB
PE Size         4.00 MiB
Total PE        5119
Alloc PE / Size 0 / 0
Free PE / Size  5119 / 20.00 GiB
VG UUID         peIcCZ-i2bl-ncC3-cvb2-M49k-bp7G-wiplsY

```

## 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 10:

Create 10GB of block storage as Vol1.

```
cinder create --display-name Vol1 10
```

Property	Value
attachments	[]
availability_zone	nova
bootable	false
created_at	2014-08-25T17:43:37.106041
display_description	None
display_name	Vol1
encrypted	False
id	2fecb1f2-e0d0-4283-bfad-15267eb0564c
metadata	{}
size	10
snapshot_id	None

source_volid	None
status	creating
volume_type	None

## cinder list

ID	Status	Display Name	Size	Volume Type	Bootable	Attached to
2fecb1f2-e0d0-4283-bfad-15267eb0564c	available	Voll	10	None	false	

If the status is available, then volume creation and Cinder installation has completed successfully.

## Attach Cinder Volume to an instance

### Step 11:

To attach the Cinder Volume to the aio-test instance that has been created, you need the instance id of the instance and also the cinder volume.

## nova list

```
root@aio53:/etc/cinder# nova list
```

ID	Name	Status	Task State	Power State	Networks
3ae56cb3-5b96-4186-9477-565b8568b4a3	test-aio	ACTIVE	-	Running	private-net=10.10.10.4
155b9159-57a1-4ac1-96e6-ala67ce312da	test-compute	ACTIVE	-	Running	private-net=10.10.10.6

## cinder list

ID	Status	Display Name	Size	Volume Type	Bootable	Attached to
55c03b5e-c7c8-432e-b9ef-bcfcc978c9c9	available	Voll	10	None	false	

Get the VM's VNC console URL by following command

## nova get-vnc-console test-aio novnc

Type	Url
novnc	http://10.1.64.151:6080/vnc_auto.html?token=045600c3-297d-49f9-a9da-3d7b8b804639

Paste this URL in a web browser and login to the vm console with the user name and password shown in the console of Cirros VM and execute “fdisk -l” to see the current disks.

```
Connected (unencrypted) to: QEMU (instance-00000002)
$ fdisk -l
Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1  *        16065      2088449     1036192+  83   Linux
$
```

On the aioX node,

```
nova volume-attach INSTANCE_ID VOLUME_ID auto
```

where INSTANCE\_ID is the instance id of the nova instance and VOLUME\_ID is the id of the cinder volume.

```
root@aio53:/etc/cinder# nova volume-attach 3ae56cb3-5b96-4186-9477-565b8568b4a3 55c03b5e-
c7c8-432e-b9ef-bcfcc978c9c9
```

```
+-----+-----+
| Property | Value          |
+-----+-----+
| device  | /dev/vdb       |
| id      | 55c03b5e-c7c8-432e-b9ef-bcfcc978c9c9 |
| serverId | 3ae56cb3-5b96-4186-9477-565b8568b4a3 |
| volumeId | 55c03b5e-c7c8-432e-b9ef-bcfcc978c9c9 |
+-----+-----+
```

At this time, the console window should show the volume attachment.

```
Connected (unencrypted) to: QEMU (instance-00000002)
$ fdisk -l

Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1 *        16065      2088449    1036192+   83   Linux
$ [324342.629913] pci 0000:00:00.0: no hotplug settings from platform
[324342.630426] pci 0000:00:01.0: no hotplug settings from platform
[324342.630773] ata_piix 0000:00:01.1: no hotplug settings from platform
[324342.631133] uhci_hcd 0000:00:01.2: no hotplug settings from platform
[324342.631554] pci 0000:00:01.3: no hotplug settings from platform
[324342.631900] pci 0000:00:02.0: no hotplug settings from platform
[324342.632515] virtio-pci 0000:00:03.0: no hotplug settings from platform
[324342.632883] virtio-pci 0000:00:04.0: no hotplug settings from platform
[324342.633330] virtio-pci 0000:00:05.0: no hotplug settings from platform
[324342.633899] pci 0000:00:07.0: no hotplug settings from platform
```

Execute “fdisk -l” to see a hard disk attached which matches the Cinder volume.

```
Connected (unencrypted) to: QEMU (instance-00000002)
[324342.632883] virtio-pci 0000:00:04.0: no hotplug settings from platform
[324342.633330] virtio-pci 0000:00:05.0: no hotplug settings from platform
[324342.633899] pci 0000:00:07.0: no hotplug settings from platform

$ fdisk -l

Disk /dev/vda: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1 *        16065      2088449    1036192+   83   Linux

Disk /dev/vdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/vdb doesn't contain a valid partition table
$
```

This completes the Cinder lab where we created a Cinder Volume using LVM as backend and successfully attached it to a VM.



# Lab 8: 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
```

apache2 - Apache HTTP Server and binaries

memcached - Memory caching system

libapache2-mod-wsgi - An Apache module that provides a WSGI (Web Server Gateway Interface) compliant interface for hosting Python based web applications within Apache.

OpenStack-dashboard - OpenStack Dashboard (also known as Horizon)

```
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 = "aioX"
```

#### Step 4:

Start the Apache web server and memcached:

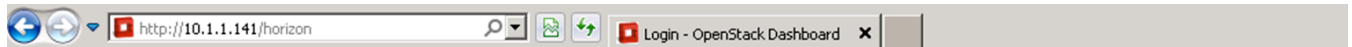
```
service apache2 restart
```

```
service memcached restart
```

#### Step 5:

Open the web browser and type <http://IP-address-of-AIO-Node/horizon>

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

The image shows the OpenStack Dashboard login form. At the top is the OpenStack logo (a red cube) and the text 'openstack DASHBOARD'. Below this is a 'Log In' section. It contains two input fields: 'User Name' with the value 'admin' and 'Password' with masked characters '....'. To the right of the password field is an eye icon. At the bottom right of the form is a blue 'Sign In' button.

# Lab 8a: 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. For a list of supported browsers, see Browser support.

3. 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.

4. 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.

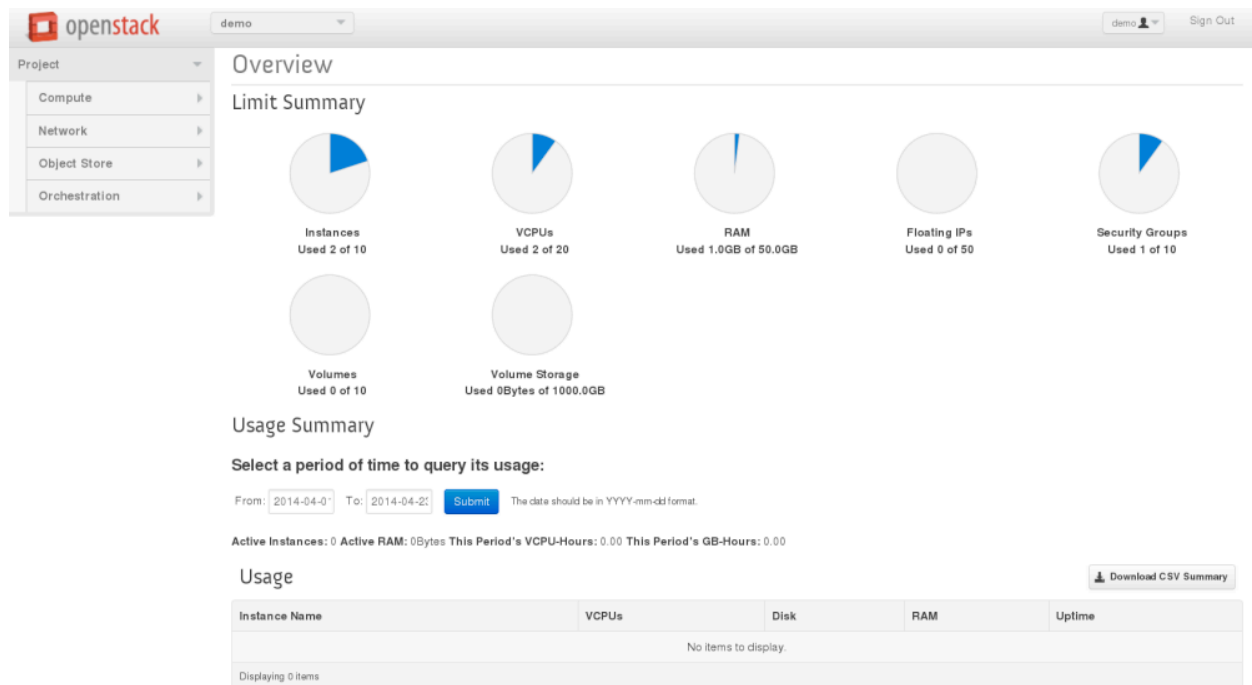
The visible tabs and functions in the dashboard depend on the access permissions, or roles, of the user you are logged in as.

- If you are logged in as an end user, the Project tab is displayed.
- If you are logged in as an administrator, the Project tab and Admin 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.



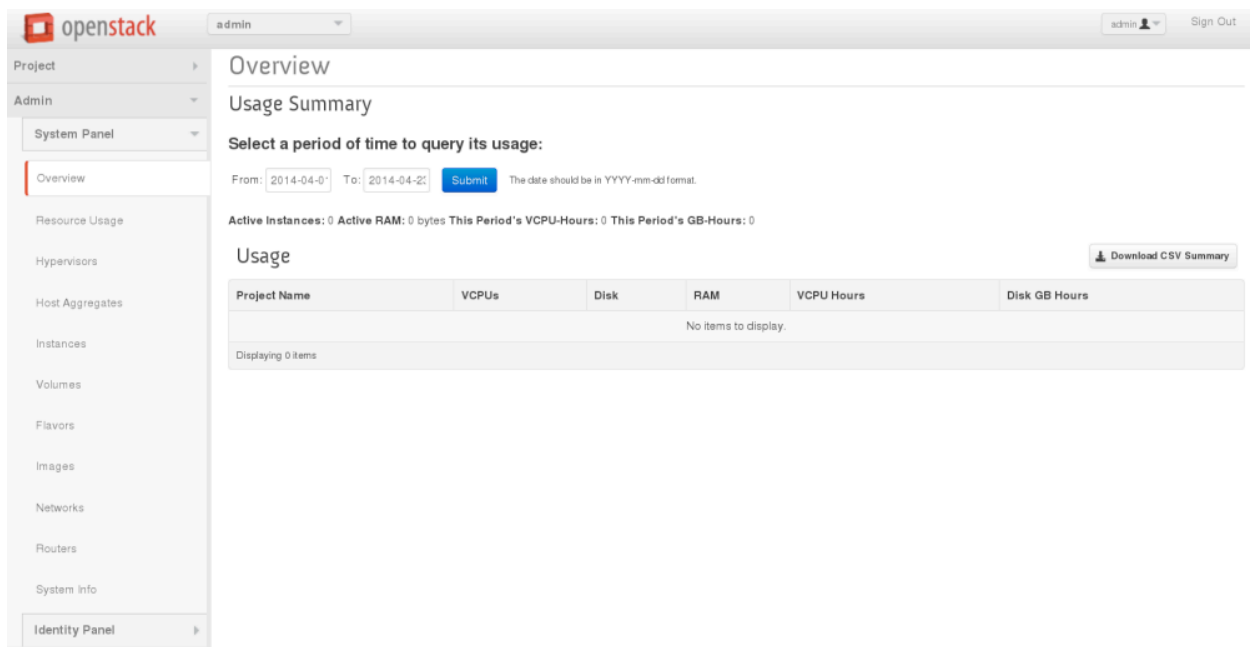
From the **Project** tab, you can access the following tabs:

Compute tab	
<b>Overview</b>	View reports for the project.
<b>Instances</b>	View, launch, create a snapshot from, stop, pause, or reboot instances, or connect to them through VNC.
<b>Volumes</b>	<p>Use the following tabs to complete these tasks:</p> <p><b>Volumes</b></p> <p>View, create, edit, and delete volumes.</p> <p><b>Volume Snapshots</b></p> <p>View, create, edit, and delete volume snapshots.</p>
<b>Images</b>	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.
<b>Access &amp; Security</b>	<p>Use the following tabs to complete these tasks:</p> <p><b>Security Groups</b></p>

Compute tab	
	<p>View, create, edit, and delete security groups and security group rules.</p> <p><b>Key Pairs</b></p> <p>View, create, edit, import, and delete key pairs.</p> <p><b>Floating IPs</b></p> <p>Allocate an IP address to or release it from a project.</p> <p><b>API Access</b></p> <p>View API endpoints.</p>
Network tab	
<b>Network Topology</b>	View the network topology.
<b>Networks</b>	Create and manage public and private networks.
<b>Routers</b>	Create and manage subnets.
Object Store tab	
<b>Containers</b>	Create and manage containers and objects.
Orchestration tab	
<b>Containers</b>	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	
<b>Overview</b>	View basic reports.
<b>Resource Usage</b>	<p>Use the following tabs to view the following usages:</p> <p><b>Daily Report</b></p> <p>View the daily report.</p> <p><b>Stats</b></p> <p>View the statistics of all resources.</p>
<b>Hypervisors</b>	View the hypervisor summary.
<b>Host Aggregates</b>	View, create, and edit host aggregates. View the list of availability zones.
<b>Instances</b>	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.
<b>Volumes</b>	View, create, edit, and delete volumes and volume types.

System Panel tab	
<b>Flavors</b>	View, create, edit, view extra specifications for, and delete flavors. A flavor is size of an instance.
<b>Images</b>	View, create, edit properties for, and delete custom images.
<b>Networks</b>	View, create, edit properties for, and delete networks.
<b>Routers</b>	View, create, edit properties for, and delete routers.
<b>System Info</b>	<p>Use the following tabs to view the service information:</p> <p><b>Services</b></p> <p>View a list of the services.</p> <p><b>Compute Services</b></p> <p>View a list of all Compute services.</p> <p><b>Network Agents</b></p> <p>View the network agents.</p> <p><b>Default Quotas</b></p> <p>View default quota values. Quotas are hard-coded in OpenStack Compute and define the maximum allowable size and number of resources.</p>
Identity Panel tab	
<b>Projects</b>	View, create, assign users to, remove users from, and delete projects.
<b>Users</b>	View, create, enable, disable, and delete users.

# Lab 8b: 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.

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”.

## 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.

5. 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	Based on your selection for Image Source, you either enter



Location	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 x86-64 for a 64-bit architecture.
Minimum Disk (GB) and Minimum 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.

## 6. 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.
7. 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 8c: 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 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”.

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”.

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:

<b>Rule</b>	SSH
<b>Remote</b>	CIDR
<b>CIDR</b>	0.0.0.0/0

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

5. Click **Add**.

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

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

<b>Rule</b>	All ICMP
-------------	----------

<b>Direction</b>	Ingress
<b>Remote</b>	CIDR
<b>CIDR</b>	0.0.0.0/0

8. 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.

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.
8. The list shows all the instances with their fixed IP addresses.
9. 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 8d: Launch and manage instances

---

Instances are virtual machines that run inside the cloud.

You can launch an instance from the following sources:

- Images uploaded to the OpenStack Image Service, as described in the section called “Upload and manage images”.
- 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 in the OpenStack Virtual Machine Image Guide*.

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

Details tab	
<b>Availability Zone</b>	By default, this value is set to the availability zone given by the cloud provider (for example, <code>us-west</code> or <code>apac-south</code> ). For some cases, it could be <code>nova</code> .
<b>Instance Name</b>	<p>Assign a name to the virtual machine.</p> <p>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.</p> <p>Server names are not guaranteed to be unique when created so you could have two instances with the same host name.</p>
<b>Flavor</b>	<p>Specify the size of the instance to launch.</p> <p>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 <b>Minimun RAM (MB)</b> field as 2048, then on selecting the image, the default flavor is <b>m1.small</b>.</p>
<b>Instance Count</b>	To launch multiple instances, enter a value greater than 1. The default is 1.
<b>Instance Boot Source</b>	<p>Your options are:</p> <p><b>Boot from image</b></p> <p>If you choose this option, a new field for <b>Image Name</b> displays. You can select the image from the list.</p> <p><b>Boot from snapshot</b></p> <p>If you choose this option, a new field for <b>Instance Snapshot</b> displays. You can select the snapshot from the list.</p> <p><b>Boot from volume</b></p> <p>If you choose this option, a new field for <b>Volume</b> displays. You can select the volume from the list.</p> <p><b>Boot from image (creates a new volume)</b></p> <p>With this option, you can boot from an image and create a volume by entering the <b>Device Size</b> and <b>Device Name</b> for your volume.</p>

Details tab	
	<p>Click the <b>Delete on Terminate</b> option to delete the volume on terminating the instance.</p> <p><b>Boot from volume snapshot (creates a new volume)</b></p> <p>Using this option, you can boot from a volume snapshot and create a new volume by choosing <b>Volume Snapshot</b> from a list and adding a <b>Device Name</b> for your volume. Click the <b>Delete on Terminate</b> option to delete the volume on terminating the instance.</p> <p>Since you are launching an instance from an image, <b>Boot from image</b> is chosen by default.</p>
<b>Image Name</b>	<p>This field changes based on your previous selection. Since you have chosen to launch an instance using an image, the <b>Image Name</b> field displays. Select the image name from the dropdown list.</p>
Access & Security tab	
<b>Keypair</b>	<p>Specify a key pair.</p> <p>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.</p>
<b>Security Groups</b>	<p>Activate the security groups that you want to assign to the instance.</p> <p>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”.</p> <p>If you have not created any security groups, you can assign only the default security group to the instance.</p>
Networking tab	
<b>Selected Networks</b>	<p>To add a network to the instance, click the <b>+</b> in the <b>Available Networks</b> field.</p>
Post-Creation tab	
<b>Customization Script</b>	<p>Specify a customization script that runs after your instance launches.</p>
Advanced Options tab	



Details tab	
Disk Partition	Select the type of disk partition from the dropdown list.
	<b>Automatic</b> Entire disk is single partition and automatically resizes.
	<b>Manual</b> Faster build times but requires manual partitioning.

#### 4. 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”.

## 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
```

3. 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 8e: 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.

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

3. 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.

5. 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 8f: 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	
<b>Network Name</b>	Specify a name to identify the network.
Subnet tab	
<b>Create Subnet</b>	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.
<b>Subnet Name</b>	Specify a name for the subnet.
<b>Network Address</b>	Specify the IP address for the subnet.
<b>IP Version</b>	Select IPv4 or IPv6.
<b>Gateway IP</b>	Specify an IP address for a specific gateway. This parameter is optional.
<b>Disable Gateway</b>	Select this check box to disable a gateway IP address.
Subnet Detail tab	

Network tab	
<b>Enable DHCP</b>	Select this check box to enable DHCP.
<b>Allocation Pools</b>	Specify IP address pools.
<b>DNS Name Servers</b>	Specify a name for the DNS server.
<b>Host Routes</b>	Specify the IP address of host routes.

4. 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.

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

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

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

7. Click **Add Interface**.

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



# Lab 9: 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 node with the credentials in Lab access  
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 -y
```

## Create Database for Orchestration Service

### Step 3:

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

```
mysql -u root -ppass  
  
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
```

```
keystone role-create --name heat_stack_user
```

## 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. Change X with AIO node Number

```
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:8004/v1/%(tenant_id)s \  
  
--internalurl=http://aioX:8004/v1/%(tenant_id)s \  
  
--adminurl=http://aioX:8004/v1/%(tenant_id)s
```

```
--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 = aioX  
rabbit_password = pass  
...  
heat_metadata_server_url = http://10.1.64.X:8000  
heat_waitcondition_server_url = http:// 10.1.64.X:8000/v1/waitcondition  
  
[database]  
connection = mysql://heat:pass@aioX/heat  
  
[keystone_authtoken]  
auth_host = aioX  
auth_port = 35357  
auth_protocol = http  
auth_uri = http://aioX:5000/v2.0  
admin_tenant_name = service  
admin_user = heat  
admin_password = pass  
  
[ec2authtoken]  
auth_uri = http://aioX:5000/v2.0
```

### Step 8:

```
su -s /bin/sh -c "heat-manage db_sync" heat
```

```
service heat-api restart

service heat-api-cfn restart

service heat-engine restart
```

## Verify the Orchestration service installation

### Step 9:

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

```
vi test-stack.yml
```

Copy following content to the test-stack.yml:

```
heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      name: Stack-VM
      key_name: mykey
      image: CirrOS 0.3.2
      flavor: m1.tiny
```

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

```
heat stack-create -f test-stack.yml Stack1
```

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

```
heat stack-list
```

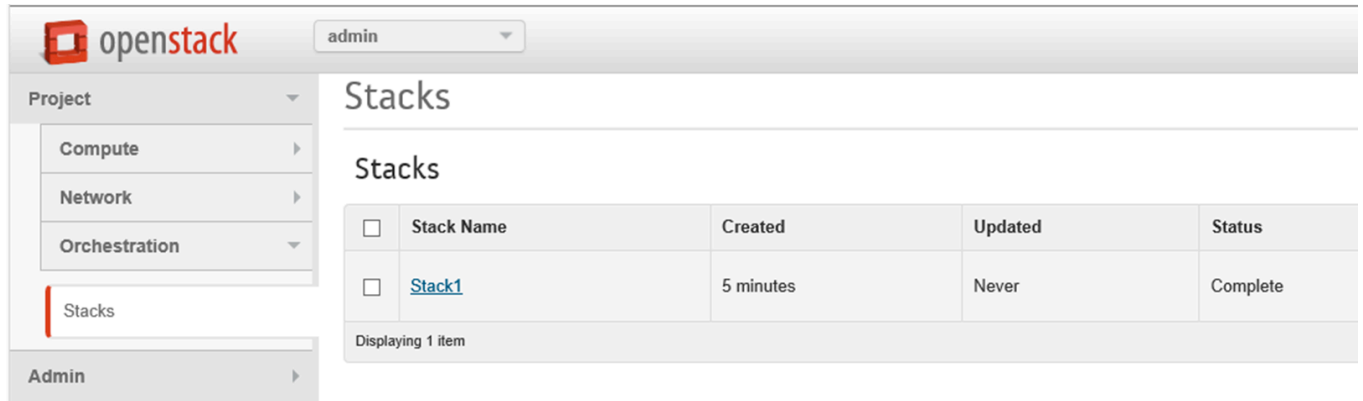
```
+-----+-----+-----+-----+
| id                | stack_name | stack_status | creation_time |
+-----+-----+-----+-----+
| 847ee6a4-61ff-4bbe-953a-7d080cbac2f8 | Stack1     | CREATE_COMPLETE | 2014-08-30T15:08:15Z |
+-----+-----+-----+-----+
```

### Step 10:

Log on to OpenStack Dashboard by Opening the webbrowser and type <http://aio node IP address/horizon>. Type user name as **admin** and password as **pass**

Goto Project → Orchestration → Stacks

Click on **Stack1**



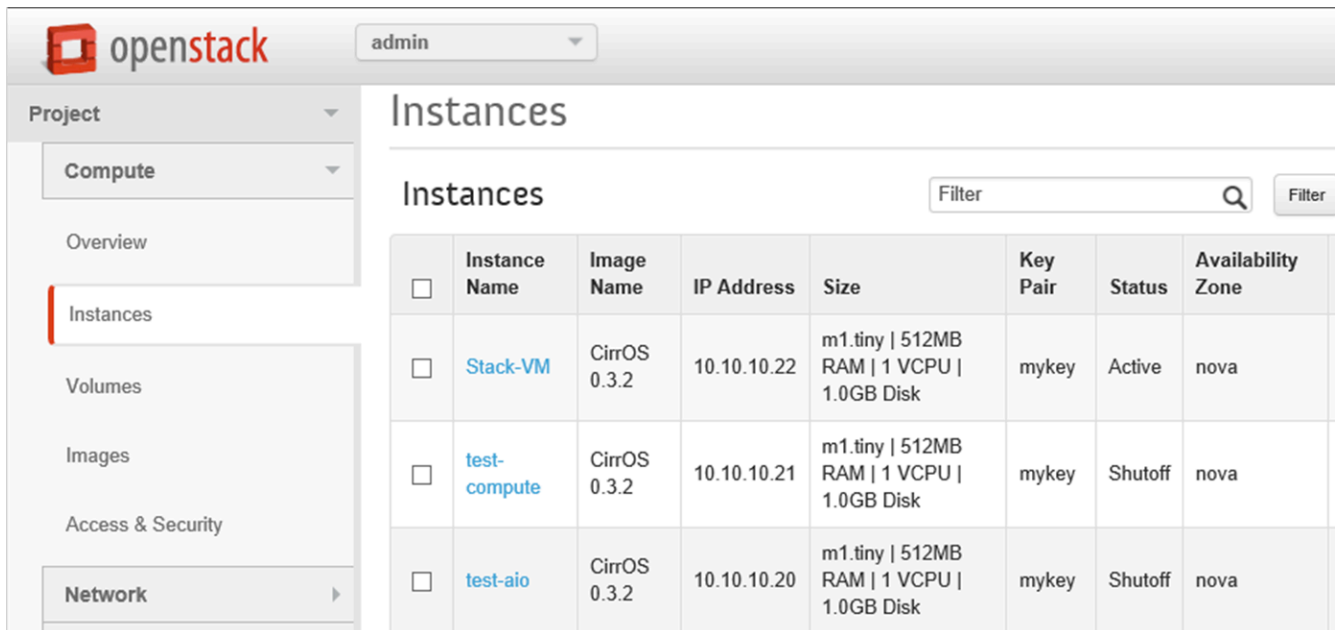
The screenshot shows the OpenStack dashboard interface. The top header includes the OpenStack logo and a user dropdown menu set to 'admin'. The left sidebar is under the 'Project' section, with 'Orchestration' selected, and 'Stacks' highlighted. The main content area is titled 'Stacks' and displays a table with the following data:

<input type="checkbox"/>	Stack Name	Created	Updated	Status
<input type="checkbox"/>	<a href="#">Stack1</a>	5 minutes	Never	Complete

Below the table, it states 'Displaying 1 item'.

This will create a new instance Stack-VM as described in the Stack1 template.

Goto Project → Compute → Instance



The screenshot shows the OpenStack dashboard interface. The top header includes the OpenStack logo and a user dropdown menu set to 'admin'. The left sidebar is under the 'Project' section, with 'Compute' selected, and 'Instances' highlighted. The main content area is titled 'Instances' and displays a table with the following data:

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone
<input type="checkbox"/>	<a href="#">Stack-VM</a>	Cirros 0.3.2	10.10.10.22	m1.tiny   512MB RAM   1 VCPU   1.0GB Disk	mykey	Active	nova
<input type="checkbox"/>	<a href="#">test-compute</a>	Cirros 0.3.2	10.10.10.21	m1.tiny   512MB RAM   1 VCPU   1.0GB Disk	mykey	Shutoff	nova
<input type="checkbox"/>	<a href="#">test-ai0</a>	Cirros 0.3.2	10.10.10.20	m1.tiny   512MB RAM   1 VCPU   1.0GB Disk	mykey	Shutoff	nova

Now Orchestration service is installed and working successfully.

## Lab 10: Telemetry Service Installation

The Telemetry module:

- Efficiently collects 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-agent-notification ceilometer-alarm-evaluator ceilometer-alarm-notifier python-ceilometerclient
```

Since we have installed Compute in AIOX, we need install ceilometer agent for compute

```
apt-get install ceilometer-agent-compute
```

## 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"
```

**Note:** Create the service end point for the service. Change X with AIO node Number

```
keystone endpoint-create \

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

--publicurl=http://aioX:8777 \

--internalurl=http://aioX:8777 \
```

```
--adminurl=http://aioX:8777
```

## Configure Ceilometer Service

### Step 7:

Edit /etc/ceilometer/ceilometer.conf

```
vi /etc/ceilometer/ceilometer.conf
```

Uncomment and Update the following in [Default] Section

```
[DEFAULT]
...
auth_strategy = keystone
...
log_dir = /var/log/ceilometer
...
rabbit_host = aioX
...
rabbit_password = pass
```

Update the following in [database] Section

```
[database]
connection = mysql://ceilometer:pass@aioX/ceilometer
```

Uncomment and Update the following in [keystone\_authtoken] Section

```
[keystone_authtoken]
auth_host = aioX
auth_port = 35357
auth_protocol = http
auth_uri = http://aioX:5000
admin_user = ceilometer
admin_password = pass
admin_tenant_name = service
```

Uncomment and Update the following in [publisher] Section

```
[publisher]
metering_secret = pass
```

Uncomment and Update the following in [service\_credentials] Section

```
[service_credentials]
os_username = ceilometer
os_password = pass
```



```
os_tenant_name = service
os_auth_url = http://aioX:5000/v2.0
```

## Configure 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 8:

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

**vi /etc/nova/nova.conf**

```
[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
```

## Configure the Image Service for Telemetry

### Step 9:

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

Edit `/etc/glance/glance-api.conf`

Uncomment and modify the `[DEFAULT]` section:

```
notification_driver = messaging
rpc_backend = rabbit
rabbit_host = aioX
rabbit_password = pass
```

## Add Block Storage service agent for Telemetry

### Step 10:

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:

```
vi /etc/cinder/cinder.conf
```

```
control_exchange = cinder  
notification_driver = cinder.openstack.common.notifier.rpc_notifier
```

### Step 11:

Run the following command to populate database

```
ceilometer-dbsync
```

Restart the following service

```
service ceilometer-agent-central restart
```

```
service ceilometer-agent-notification restart
```

```
service ceilometer-api restart
```

```
service ceilometer-collector restart
```

```
service ceilometer-alarm-evaluator restart
```

```
service ceilometer-alarm-notifier restart
```

```
service ceilometer-agent-compute restart
```

```
service glance-registry restart
```

```
service glance-api restart
```

```
service cinder-api restart
```

```
service cinder-scheduler restart
```

```
service cinder-volume restart
```

```
service nova-compute restart
```

### Step 12:

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

```
ceilometer meter-list
```

## Lab 10a: Telemetry Service Installation on Compute Node

## Install the Telemetry service on the compute node(ComputeY):

### Step 1:

Log On to ComputeY and type following commands

```
sudo su -
```

Install telemetry agent for compute to collect information from Compute Node.

```
apt-get install ceilometer-agent-compute
```

### Step 2:

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
```

## Configure Ceilometer Service

### Step 3:

Edit /etc/ceilometer/ceilometer.conf

```
vi /etc/ceilometer/ceilometer.conf
```

Uncomment and Update the following in [Default] Section

```
[DEFAULT]
...
log_dir = /var/log/ceilometer
...
rabbit_host = aioX
...
rabbit_password = pass
```

Update the following in [database] Section

```
[database]
connection = mysql://ceilometer:pass@aioX/ceilometer
```

Uncomment and Update the following in [keystone\_authtoken] Section

```
[keystone_authtoken]
auth_host = aioX
auth_port = 35357
auth_protocol = http
auth_uri = http://aioX:5000
admin_user = ceilometer
admin_password = pass
admin_tenant_name = service
```

Uncomment and Update the following in [publisher] Section

```
[publisher]
metering_secret = pass
```

Uncomment and Update the following in [service\_credentials] Section

```
[service_credentials]
os_username = ceilometer
os_password = pass
os_tenant_name = service
os_auth_url = http://aioX:5000/v2.0
```

#### Step 4:

Restart the following service

```
service nova-compute restart
```

```
service ceilometer-agent-compute restart
```

## Verify the Telemetry service installation

#### Step 5:

Download an image from the Image Service:

```
glance image-download "CirrOS 0.3.2" > cirros.img
```

You can now get usage statistics for the various meters:

```
ceilometer statistics -m image.download -p 60
```

This command will give a statistics of image download meter.

#### Step 6:

Log on to OpenStack Dashboard by Open the webbrowser and type `http://aio node IP address/horizon`. Type user name as **admin** and password as **pass**

Goto Admin → System Panel → Resource Usage

Click **Stats** and check for different Metric

Telemetry Service is now installed successfully.

# Lab 11: Install OpenStack with DevStack

---

Instructions for installing OpenStack using DevStack and VirtualBox are documented in a separate document available at:

[https://github.com/onecloud/OpenStack\\_bootcamp](https://github.com/onecloud/OpenStack_bootcamp)

Files needed for this exercise are available for free online or may be available from your instructor.

# Thank you

