

# Tutorials: Openstack Based Hands-On Experimentation Infrastructure

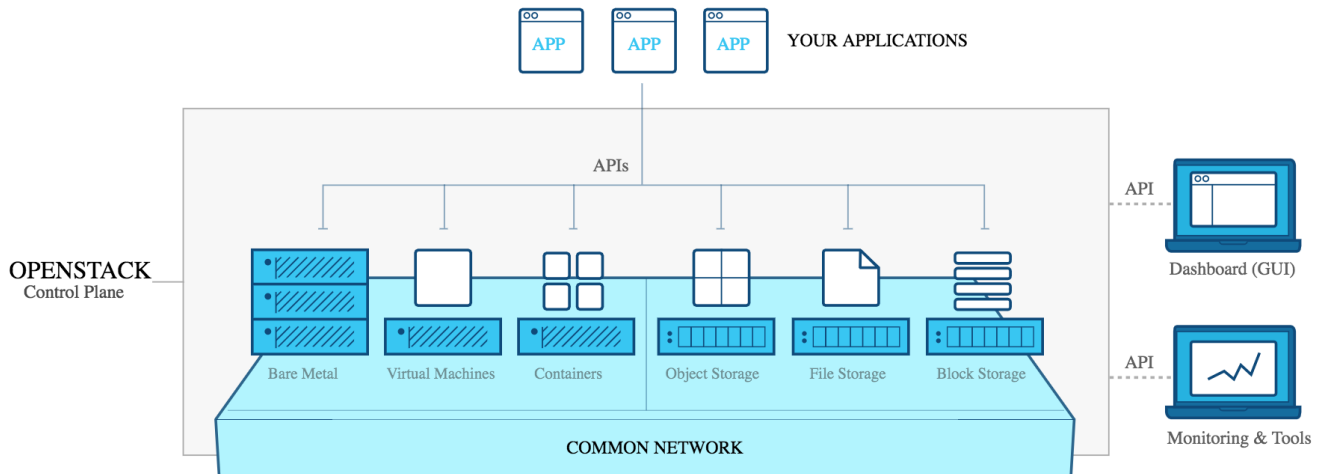
## Table of Contents

<b>PART I: INTRODUCTION TO OPENSTACK</b>	<b>1</b>
DEFINITION OF OPENSTACK	1
BACKGROUND / THE ORIGIN OF OPENSTACK	1
OVERVIEW:	2
HARDWARE REQUIREMENTS:	2
SOFTWARE REQUIREMENT	4
NETWORKING OPTIONS	4
<b>PART II: UNDERSTAND OPENSTACK PROJECTS</b>	<b>8</b>
OPENSTACK PROJECTS/ COMPONENTS	8
<b>PART III: OPENSTACK INSTALLATION AND CONFIGURATION</b>	<b>122</b>
CONFIGURE NETWORK INTERFACE	122
NETWORK TIME PROTOCOL	133
INSTALL OPENSTACK PACKAGES	144
SQL DATABASE	144
MESSAGE QUEUE	155
MEMCACHED	155
IDENTITY SERVICE – KEYSTONE	166
IMAGE SERVICE – GLANCE	19
COMPUTE SERVICE – NOVA	222
NETWORKING SERVICE – NEUTRON	255
<b>PART IV: OPENSTACK OPERATIONS</b>	<b>300</b>
DASHBOARD SERVICE – HORIZON	300
LAUNCH INSTANCE WITH DASHBOARD	311
LAUNCH AN INSTANCE WITH COMMAND LINE TOOL	344

## PART I: Introduction to OpenStack

### Definition of OpenStack:

OpenStack is a cloud operating system that controls large pools of *compute, storage, and networking* resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.



### Background / The origin of OpenStack:

OpenStack started in 2010, as a joint project of Rackspace Hosting and NASA (National Aeronautics and Space Administration):

NASA contributed their Nebula platform, which later developed into Nova.

Rackspace contributed their Cloud Files platform, which later became Swift.

In April of 2011, the OpenStack Bexar release was introduced in Ubuntu. Later that same year, Debian included OpenStack Cactus in their distribution.

In 2012, Red Hat announced a preview of their OpenStack distribution as well. Since then, many others followed, including Oracle, HP, and VMware.

In September 2012, the OpenStack Foundation was founded to provide an independent home for the OpenStack cloud operating system, which has since become one of the largest and most diverse open source projects in history.

The OpenStack Foundation promotes the global development, distribution, and adoption of the cloud operating system. It provides shared resources to grow the OpenStack cloud. It also enables technology vendors and developers to assist in the production of cloud software.

### Overview:

The OpenStack project is an open source cloud computing platform that support all types of cloud environments. The project aims for simple implementation, massive scalability, and a rich set of features.

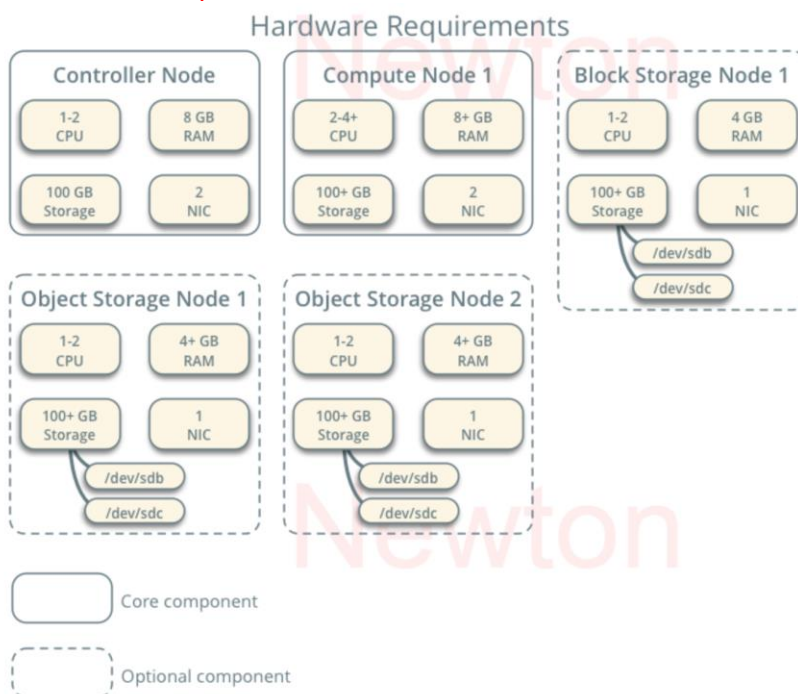
- Some application samples of OpenStack:
- Web Applications
- Big Data
- eCommerce
- Video Processing and Content Delivery
- High Throughput Computing
- Container Optimized
- Web Hosting
- Public Cloud
- DBaaS

In sum, OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complementary services. Each service offers an Application Programming Interface (API).

### Hardware Requirements:

Since OpenStack support the scalable extension of the computer platform, an example architecture is introduced below to show a minimal production configuration.

In order to satisfy the experiment requirements, we recommend the configuration with 1 controller node and 1 compute node.



### Controller node:

The services run on the controller node:

- Identity service
- Image service
- management of Compute nodes
- management of Networking
- various Networking agent
- Dashboard
- SQL database
- Message Queue
- NTP
- .....

Optionally, the Controller node could also run services:

- Block Storage
- Object Storage
- Orchestration
- Telemetry services
- .....

### Compute node:

The compute nodes runs the hypervisor portion of Compute that operates instances (VMs).  
(hypervisor: software that arbitrates and controls VM access to the actual underlying hardware)  
By default, Compute uses the KVM hypervisor.

The services on the Compute node:

- Instance hypervisor
- Networking service agent
- Firewalling service

### Minimum hardware requirement:

- Controller Node: 1 processor, 4 GB memory, and 5 GB storage
- Compute Node: 1 processor, 2 GB memory, and 10 GB storage
- Network Interface: each node should have at least two NICs that work compatibly
- Single disk partition

The above configuration can support a proof-of-concept environment with core services and several CirrOS instances.

(CirrOS: A minimal Linux distribution designed for use as a test image on clouds such as OpenStack.)

### Our experiment hardware:

- Controller Node: 4 processor, 16 GB memory, and 1 TB storage

- Compute Node: 4 processor, 16 GB memory, and 1 TB storage
- Network Interface: each node should have at least two NICs that work compatibly
- Single disk partition

Our configuration can support several Ubuntu 16 server version instances, combined with already development environment.

#### Another Option:

User can build each nodes as a virtual machine (VM) for the first-time installation and testing purpose. However, VMs will reduce performance of your instances.

### Software Requirement

When choosing an open source cloud operating system, a vibrant commercial ecosystem is key to the long-term viability of your platform choice. With OpenStack, there are over 180 participating companies and we encourage you to dig in and find the right path for you.

To start, there are many ways to install and deploy OpenStack through software distributions, each of which add their own value to the cloud operating system.

Some popular installation operating systems:

- Ubuntu 16.04 LTS server
- Red Hat Enterprise Linux 7
- CentOS 7
- openSUSE / SUSE Linux Enterprise Server

For our lecture, we choose Ubuntu 16.04 barely installed on the nodes. The machine has clean OS helps avoid some unexpected errors.

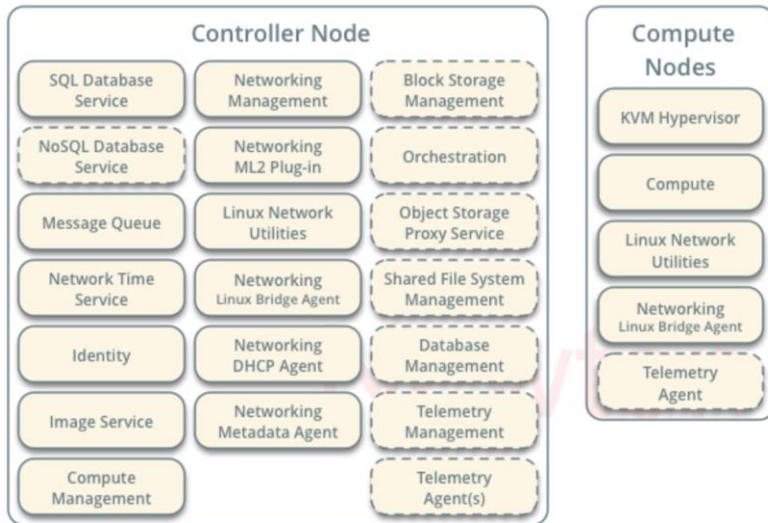
### Networking Options

#### Networking option 1: Provider Networks

- Simplest way
- Layer-2 services (bridging or switching)
- VLAN segmentation of networks

It bridges virtual network to physical network

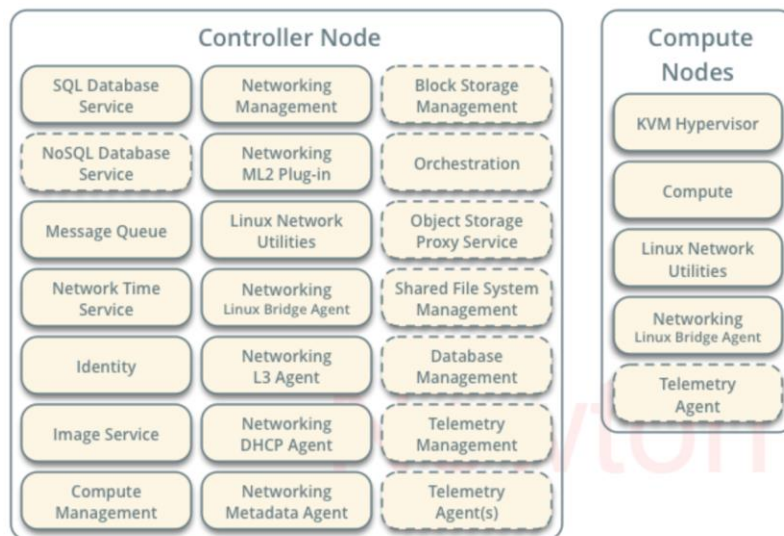
Physical network provides the layer-3 services (routing): e.g. DHCP



### Networking option2: Self-service networks

Besides the services in the provider networks, this option provides:

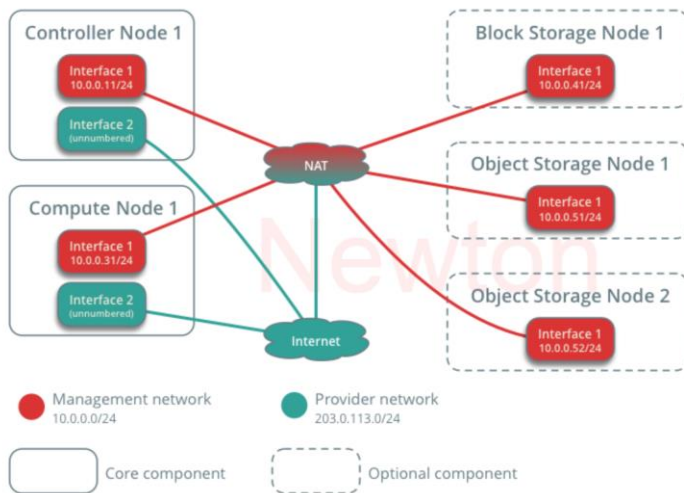
- Layer-3 routing service that support using overlay segmentation methods such as VXLAN.
- NAT (Network Address Translation) to route the virtual networks to physical networks.
- Some additional functions can be plugged in, such as LBaaS (Load Balance as a Service) and FWaaS (FireWall as a Service)



### Host Networking Example

After installing the operating system on each node for the architecture that you choose to deploy, you must configure the network interfaces. An example network layout is shown below:

Network Layout



- Management on 10.0.0.0/24 with gateway 10.0.0.1

This network requires a gateway to provide Internet access to all nodes for administrative purposes such as package installation, security updates, DNS, and NTP.

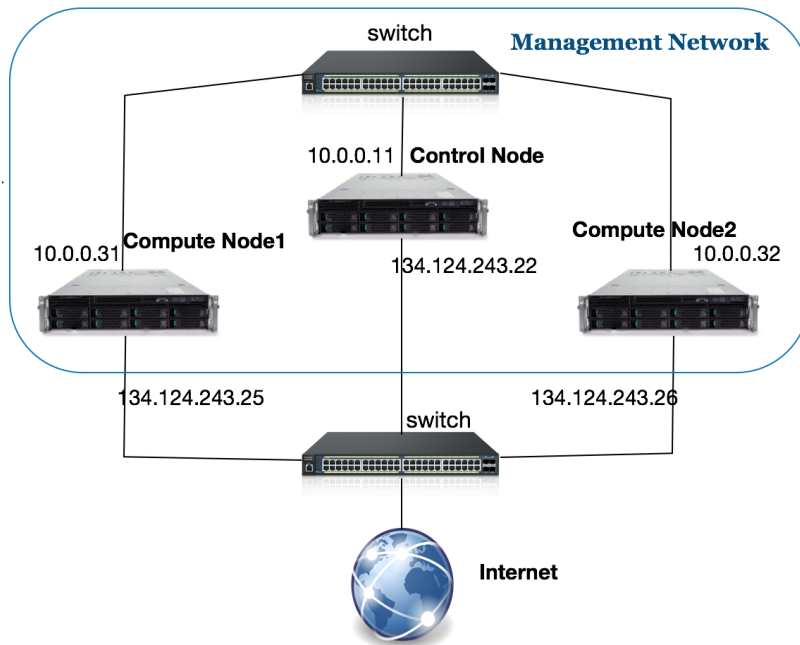
- Provider on 203.0.113.0/24 with gateway 203.0.113.1

This network requires a gateway to provide Internet access to instances in your OpenStack environment.

All nodes require Internet access for administrative purposes such as package installation, security updates, DNS, and NTP.

In most cases, nodes should obtain internet access through the management network interface.

The actual / practical for our installation is illustrated below:





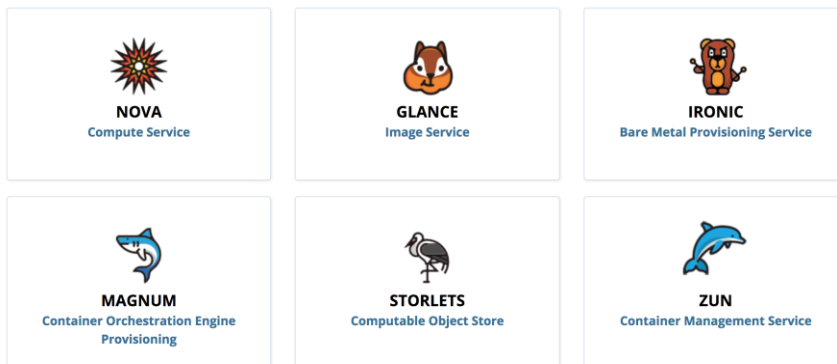
## PARTII: Understand OpenStack Projects

### OpenStack Projects/ Components

In OpenStack, there are many projects, each of them with a different adoption status. To learn Opensatck, it is important to become familiar with its different components and the corresponding functions.

Projects can be navigated from the following aspects:

- Compute : Nova, Glance ...
- Storage, Backup & Recovery : Swift, Cinder ...
- Networking & Content Delivery: Neutron ...
- Data & Analytics
- Security, identity & Compliance: Keystone
- Management Tools: Horizon
- Deployment Tools
- Application Services
- Monitoring & Metering



The OpenStack Foundation distinguishes these projects between core services and optional services. The core services are Nova, Neutron, Swift, Cinder, Keystone and Glance. Each of them has different data to indicate the development history and the level the maturity. For example, Nova is deployed in the 95% of all OpenStack clouds.



Some optional services are also performs very well, like Horizon, which has been used for five years and is adopted for 87% of all OpenStack deployments.

Let's introduce our core OpenStack projects.

#### Overview:

- Nova – Compute service: Nova is the interface to the hypervisor. It make sure that VMs can be run in the cloud nodes.
- Neutron – Networking service: Neutron provides Software Defined Networking to the cloud.
- Swift – Object Storage service: With help of Swift, storage is not bound to physical devices, but organized in binary objects that can be scattered all over the cloud in a distributed and replicated way.
- Cinder – Block Storage service: Cinder is used to support persistent storage to VMs deployed in the cloud.
- Keystone – Identity service: Keystone decides which user has access to which specific service and how specific services can communicate to one another.
- Glance – Image service: Glance stores and manages the images of VM instances.

#### Nova



Nova is responsible for managing the compute instance lifecycle:

- It interfaces to the hypervisor, but it is not a hypervisor.
- It is responsible for spawning, scheduling, and decommissioning of virtual machines on demand.
- It includes the Nova service processes running on the Cloud Controller, as well as the Nova agents running on the hypervisor.

Therefore, Nova is using a distributed architecture.

(Note: Hypervisor: Xen(ubuntu), KVM, VMware, vSphere)

#### Neutron



- Neutron enables Software Defined Networking (SDN). SDN allows users to define their own networking between the instances that are deployed, at the usage level, called the Overlay network.
- Neutron is using a pluggable architecture that supports numerous vendors and technologies.

- Neutron also provides an API, allowing its users to define the networks and the attachments into them, which is relatively easy for programmer or developer to create their own Software Defined Networking environment.

## Swift



**SWIFT**  
Object Store

Swift was designed to provide scalability at the storage level. Swift is ideal for storing unstructured data that can grow without bound.

Application <==> Swift proxy < == > hard disks

- It works with binary objects to store data in a distributed, replicated way.
- It can be accessed by applications via a RESTful API.
- It is designed to be very fault-tolerant.

## Glance



**GLANCE**  
Image Service

- Glance is used to store virtual machine disk images. VMs are not installed but launched from an image like a live usb/cd boot, which can be either downloaded or created. (<https://docs.openstack.org/image-guide/obtain-images.html>).
- VM images made available through Glance can be stored in a variety of locations from simple filesystems to object-storage systems like the OpenStack Swift project.
- So Glance is an image store, where the administrator could boot an instance with Glance service.
- Glance service includes discovering, registering, and retrieving VM images.
- It has a RESTful API that allows querying of VM images metadata.

## Cinder



**CINDER**  
Block Storage

Cinder provides persistent storage to instances.

Instance storage is ephemeral, but Cinder allows administrators to attach additional persistent block devices to the instances.

Cinder can use different backends, including Swift object storage.

## Keystone



**KEYSTONE**  
Identity service

Used for authentication and authorization

Keystone creates and assigns the users and roles for projects. By default, it uses a MariaDB database to store information, but other databases can be used as well.

Central repository for all services and endpoints (URL that provides access to the specific service)

It supports LDAP, OAuth, OpenID connect, SAML, and SQL.

## Horizon



**HORIZON**  
Dashboard

Horizon is not a core project but it exists beside them. It is frequently used, in 87% OpenStack installation.

Horizon is the Dashboard of OpenStack, providing web interface for easy management of instances and other OpenStack properties.

Horizon is the ideal platform for end-user self-management.

We can use command line interface (CLI) as an alternative.

## PART III: OpenStack Installation and Configuration

(This course is based on OpenStack Newton version)

There are different method to deploy OpenStack: manually, scripted and large scale deployment.

For our class, we introduce a manual installation process based on our experimental environment, where two physical machine with Ubuntu 16.04 system. This is a minimal production installation.

Besides, you can also try to install “devstack” in a scripted automatic way in a virtual machine or a single physical machine. In this case, all the functional components are existed in a single environment.

### Configure Network Interface

We have 2 nodes: controller + compute. Each is equipped with 2 NICs.

Our OpenStack has two networks: management network and provider network.

So one NIC maps to == > one network interface

We choose “Self-service Network” as the example.

For our installation, we use the a 3-level router to provide NAT function;

A cisco switch is utilized to provide internet.

Based on the IP pools of UMSL, the management network we use IP pool: 10.0.0.0/24 (private IP); the provider network we use IP pool: 134.124.xxx.0/24 (public IP)

#### (1) Controller Node

/etc/network/interfaces file

Manage interface:

- IP address: 10.0.0.11
- Network mask: 255.255.255.0 (or /24)
- Default gateway: 10.0.0.1

Provider interface:

Use special configuration without an IP address assigned.

```
auto INTERFACE_NAME
```

```
iface INTERFACE_NAME inet manual
```

```
up ip link set dev $IFACE up
```

```
down ip link set dev $IFACE down
```

```
(interface_name : eth0, eth1.....)
```

Name resolution:

```
/etc/hosts
```

```
# controller
```

```
10.0.0.11    controller
```

```
# compute1
10.0.0.31    compute1
```

## (2) Compute node

Manage interface:

- IP address: 10.0.0.31
- Network mask: 255.255.255.0 (or /24)
- Default gateway: 10.0.0.1

Provider interface:

```
auto INTERFACE_NAME
iface INTERFACE_NAME inet manual
up ip link set dev $IFACE up
down ip link set dev $IFACE down
```

Name resolution: same as controller

Note:

- Comment out the 127.0.1.1 entry if exists.
- Reboot the system to activate the changes.

## Network Time Protocol

We install Chrony, an implementation of NTP service, to synchronize services among nodes.

### (1) Controller node:

- a. Install the packages:

```
apt install chrony
```

- b. Edit the `/etc/chrony/chrony.conf` file and add, change, or remove these keys as necessary for your environment:

```
server 10.0.0.11 iburst
```

- c. To enable other nodes to connect to the chrony daemon on the controller node, add this key to the `/etc/chrony/chrony.conf` file:

```
allow 10.0.0.0/24
```

- d. Restart the NTP service:

```
service chrony restart
```

### (2) Compute node:

- a. Install the packages:

```
apt install chrony
```

- b. Edit the `/etc/chrony/chrony.conf` file and comment out or remove all but one server key.  
Change it to reference the controller node:  
`server controller iburst`
- c. Comment out the `pool 2.debian.pool.ntp.org offline iburst` line.
- d. Restart the NTP service:  
`service chrony restart`

## Install OpenStack Packages

Please install the following packages in both controller and compute nodes (or all the nodes you deployed). These packages are the fundament of the core service of OpenStack projects.

Note: try to use `sudo` if you meet the right problem to run the commands.

### (1) Enable the OpenStack repository

```
apt install software-properties-common
add-apt-repository cloud-archive:newton
```

### (2) Upgrade the package on your machines

```
apt update && apt dist-upgrade
```

### (3) Install the OpenStack client

```
apt install python-openstackclient
```

## SQL database

SQL database typically runs on the controller node, which store information for most OpenStack service. For our experiments, we use MySQL database.

### (1) Install MySQL

```
apt install mariadb-server python-pymysql
```

### (2) Create and edit the `/etc/mysql/mariadb.conf.d/99-openstack.cnf` file

Create a **[mysqld]** section, and set the **bind-address** key to the management IP address of the controller node to enable access by other nodes via the management network. Set additional keys to enable useful options and the UTF-8 character set:

```
[mysqld]
bind-address = 10.0.0.11
default-storage-engine = innodb
```

```
innodb_file_per_table
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

### (3) Restart the database service

Service mysql restart

Secure the database service by running the `mysql_secure_installation` script. You need to input the password for your database root account.

```
mysql_secure_installation
```

## Message Queue

Message queue coordinate operation and status information among services.

It runs on the Controller node.

OpenStack supports several message queue services including RabbitMQ, Qpid, and ZeroMQ.

Our guide implements the RabbitMQ message queue service because most distributions support it.

### (1) Install RabbitMQ package

```
apt install rabbitmq-server
```

### (2) Add the “openstack” user

```
rabbitmqctl add_user openstack Your_password
```

### (3) Permit configuration, write, and read access for the openstack user

```
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

## Memcached

Memcached is a free & open source, high-performance, distributed memory object caching system.

The identity service (Keystone) authentication mechanism for services use Memcaches to cache token.

It runs on the Controller node.

### (1) Install Memcached

```
apt install memcached python-memcache
```

(2) Edit the `/etc/memcached.conf` file and configure the service to use the management IP address of the controller node. This is to enable access by other nodes via the management network.

Change the line `-l 127.0.0.1` to `-l 10.0.0.11`



### (3) Restart the Memcached service

```
service memcached restart
```

## Identity Service – Keystone

The Identity service is typically the first service a user interacts with. Once authenticated, an end user can use their identity to access other OpenStack services. Likewise, other OpenStack services leverage the Identity service to ensure users are who they say they are and discover where other services are within the deployment.

OpenStack supports multiple regions for scalability. For simplicity, this guide uses the management network for all endpoint types and the default **RegionOne** region. Together, regions, services, and endpoints created within the Identity service comprise the service catalog for a deployment. Each OpenStack service in your deployment needs a service entry with corresponding endpoints stored in the Identity service.

The identify service is named “Keystone”, as one of the openstack projects, installed on **controller node**.

### (1) Create keystone database and grant proper access to the keystone database.

```
mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' /
IDENTIFIED BY 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' /
IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Replace KEYSTONE\_DBPASS with your own password

### (2) Install keystone package

```
apt install keystone
```

### (3) Configure Keystone components

Edit `/etc/keystone/keystone.conf` file and add the following lines in corresponding section

[database]

```
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

[token]

```
provider = fernet
```

Populate the Keystone database:

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

Initialize Fernet key repositories:

```
keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
```

#### Bootstrap the Identity service:

```
keystone-manage bootstrap --bootstrap-password ADMIN_PASS
--bootstrap-admin-url http://controller:35357/v3/
--bootstrap-internal-url http://controller:35357/v3/
--bootstrap-public-url http://controller:5000/v3/
--bootstrap-region-id RegionOne
```

#### (4) Finalize the installation

##### Configure the Apache HTTP server:

Edit the `/etc/apache2/apache2.conf` file and configure the **ServerName** option to reference the controller node

```
ServerName controller
```

##### Restart the Apache service and remove the default SQLite database

```
service apache2 restart
rm -f /var/lib/keystone/keystone.db
```

#### (5) Create a script to access the administrative account

##### Create a file named `admin` and add the following lines

```
$ export OS_USERNAME=admin
$ export OS_PASSWORD=ADMIN_PASS (same in (3))
$ export OS_PROJECT_NAME=admin
$ export OS_USER_DOMAIN_NAME=Default
$ export OS_PROJECT_DOMAIN_NAME=Default
$ export OS_AUTH_URL=http://controller:35357/v3
$ export OS_IDENTITY_API_VERSION=3
```

It is used to get the admin right of OpenStack quickly, use `source admin_filename` to execute this file.

#### (6) After installation of Keystone and create admin access, we need to initiate the authentication service.

The authentication service includes *domains, projects, users and roles* components.

Create a service project that contains a unique user for each service we will add later.

```
openstack project create --domain default
--description "Service Project" service
```

display:

Field	Value
-------	-------

description	Service Project
domain_id	default
enabled	True
id	24ac7f19cd944f4cba1d77469b2a73ed
is_domain	False
name	service
parent_id	default

Create a non-admin project for experiment purpose in the future. E.g., project name: demo

```
openstack project create --domain default
--description "Demo Project" demo
```

Field	Value
description	Demo Project
domain_id	default
enabled	True
id	231ad6e7ebba47d6a1e57e1cc07ae446
is_domain	False
name	demo
parent_id	default

Then create a user in the above project. We can also give the user name: demo

```
openstack user create --domain default
--password-prompt demo
```

User Password:

Repeat User Password:

Field	Value
domain_id	default
enabled	True
id	aeda23aa78f44e859900e22c24817832
name	demo
password_expires_at	None

Create a “user” role and assign the “user” role to demo project and user

```
openstack role create user
```

Field	Value
-------	-------

domain_id	None
id	997ce8d05fc143ac97d83fdb5998552
name	user

```
openstack role add --project demo --user demo user
```

### (7) For security, disable the temporary authentication token mechanism

Edit the `/etc/keystone/keystone-paste.ini` file and remove `admin_token_auth` from the `[pipeline:public_api]`, `[pipeline:admin_api]`, and `[pipeline:api_v3]` sections.

Unset the temporary `OS_AUTH_URL` and `OS_PASSWORD` environment variable:

```
unset OS_AUTH_URL OS_PASSWORD
```

**Note:** Now you can also create a script to quick obtain demo user right, like in (5)

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

## Image Service – Glance

The Image service (glance) enables users to discover, register, and retrieve virtual machine images.

It offers a REST API that enables you to query virtual machine image metadata and retrieve an actual image.

You can store virtual machine images made available through the Image service in a variety of locations, from simple file systems to object-storage systems like OpenStack Object Storage.

By default, this directory is `/var/lib/glance/images/`

Installation is performed on **controller node**.

### (1) Prerequisites

Create a database for Glance

```
mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost'
IDENTIFIED BY 'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
IDENTIFIED BY 'GLANCE_DBPASS';
```

Use admin credentials to access admin-only CLI commands and create glance user

```
. admin-openrc
openstack user create --domain default --password-prompt glance
openstack role add --project service --user glance admin
openstack service create --name glance --description "OpenStack Image" image
```

**Create glance service API endpoints:**

```
openstack endpoint create --region RegionOne \
    image public http://controller:9292
openstack endpoint create --region RegionOne \
    image internal http://controller:9292
openstack endpoint create --region RegionOne \
    image admin http://controller:9292
```

**(2) Install and configure Glance components**

**Install Glance**

```
apt install glance
```

**Edit the /etc/glance/glance-api.conf file and add following in corresponding section**

[database]

```
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
```

[keystone\_authtoken] (comment our other lines)

```
auth_uri = http://controller:5000
```

```
auth_url = http://controller:35357
```

```
memcached_servers = controller:11211
```

```
auth_type = password
```

```
project_domain_name = Default
```

```
user_domain_name = Default
```

```
project_name = service
```

```
username = glance
```

```
password = GLANCE_PASS
```

[paste\_deploy]

```
flavor = keystone
```

[glance\_store]

```
stores = file,http
```

```
default_store = file
```

```
filesystem_store_datadir = /var/lib/glance/images/
```

**Edit the /etc/glance/glance-registry.conf file**

```
[database]
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance

[keystone_authtoken] (comment our other lines)
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor = keystone
```

**Popular the Glance service database**

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

**Restart Glance service**

```
service glance-registry restart
service glance-api restart
```

**(3) Download images**

**Let's download testing image CirrOS.**

```
. admin-openrc
wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

**Upload the image to the Image service using the QCOW2 disk format, bare container format, and public visibility so all projects can access it**

```
openstack image create "cirros" \
  --file cirros-0.3.4-x86_64-disk.img \
  --disk-format qcow2 --container-format bare \
  --public
```

**Confirm the existence of image**

```
openstack image list
```

```
+-----+-----+-----+
| ID                                | Name    | Status |
+-----+-----+-----+
| 38047887-61a7-41ea-9b49-27987d5e8bb9 | cirros  | active |
+-----+-----+-----+
```

## Compute service – Nova

Use OpenStack Compute to host and manage cloud computing systems

OpenStack Compute interacts with OpenStack Identity for authentication; OpenStack Image service for disk and server images; and OpenStack dashboard for the user and administrative interface.

OpenStack Compute can scale horizontally on standard hardware, and download images to launch instances.

The computer service needs to be installed both on the **controller** and **computer** nodes.

### Controller node

#### (1) Prerequisites

Create and configure database for Compute service:

```
mysql -u root -p
mysql> CREATE DATABASE nova_api;
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' \
    IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' \
    IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
    IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
    IDENTIFIED BY 'NOVA_DBPASS';
```

Create Compute service credentials:

```
openstack user create --domain default \
    --password-prompt nova
openstack role add --project service --user nova admin
openstack service create --name nova \
    --description "OpenStack Compute" compute
```

Create Computer service API endpoints:

```
openstack endpoint create --region RegionOne \
    compute public http://controller:8774/v2.1/%(tenant_id)s
openstack endpoint create --region RegionOne \
    compute internal http://controller:8774/v2.1/%(tenant_id)s
openstack endpoint create --region RegionOne \
    compute admin http://controller:8774/v2.1/%(tenant_id)s
```

#### (2) Install Nova package

```
apt install nova-api nova-conductor nova-consoleauth \
    nova-novncproxy nova-scheduler
```

Edit the `/etc/nova/nova.conf` file and complete the following actions:

```

[api_database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api

[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 10.0.0.11
use_neutron = True (Enable Networking service)
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
api_servers = http://controller:9292

[oslo_concurrency]
lock_path = /var/lib/nova/tmp

```

Due to a packaging bug, remove the `log-dir` option from the `[DEFAULT]` section

### (3) Finish Installation

#### Populate the Compute database:

```

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

```

#### Start Computing services:

```

service nova-api restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart
service nova-novncproxy restart

```

### Compute Node



Our configuration uses the QEMU hypervisor with the KVM extension on compute nodes that support hardware acceleration for virtual machines. On legacy hardware, this configuration uses the generic QEMU hypervisor.

### (1) Install and Configure Nova service

```
apt install nova-compute
```

Edit the `/etc/nova/nova.conf` file and complete the following actions

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
my_ip = 10.0.0.31
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[DEFAULT]
my_ip = 10.0.0.31

[vnc]
enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://controller:6080/vnc_auto.html

[glance]
api_servers = http://controller:9292

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Due to a packaging bug, remove the `log-dir` option from the `[DEFAULT]` section

### (2) Finish installation

```
service nova-compute restart
```

### (3) Verify installation

```
. admin-openrc
```

List service components to verify successful launch and registration of each process

```
openstack compute service list
```

## Networking Service – Neutron

OpenStack Networking (neutron) allows you to create and attach interface devices managed by other OpenStack services to networks. Plug-ins can be implemented to accommodate different networking equipment and software, providing flexibility to OpenStack architecture and deployment.

OpenStack Networking (neutron) manages all networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking enables projects to create advanced virtual network topologies which may include services such as a firewall, a load balancer, and a virtual private network (VPN).

Networking provides networks, subnets, and routers as object abstractions. Each abstraction has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnets and networks.

Install both on **controller** and **compute** node.

You need to use the option: **self-service**

### Controller Node

#### (1) Prerequisites

Create Neutron database and grant access

```
mysql -u root -p
mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
IDENTIFIED BY 'NEUTRON_DBPASS';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
IDENTIFIED BY 'NEUTRON_DBPASS';
```

Create service user, API endpoints:

```
. admin-openrc
openstack user create --domain default --password-prompt neutron
openstack role add --project service --user neutron admin
openstack service create --name neutron \
--description "OpenStack Networking" network
openstack endpoint create --region RegionOne \
network public http://controller:9696
openstack endpoint create --region RegionOne \
```

```
network internal http://controller:9696
openstack endpoint create --region RegionOne \
network admin http://controller:9696
```

## (2) Configuring Networking Options:

### Install Neutron packages:

```
apt install neutron-server neutron-plugin-ml2 \
neutron-linuxbridge-agent neutron-l3-agent neutron-dhcp-agent \
neutron-metadata-agent
```

### Configure the server component:

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

```
[database] (comment out all other lines)
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
```

```
[DEFAULT]
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
```

```
[keystone_authtoken] (comment out all other lines)
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
```

```
[nova]
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

### Configure the Modular Layer2(ML2) plug-in:

The ML2 plug-in uses the Linux bridge mechanism to build layer-2 (bridging and switching) virtual networking infrastructure for instances.

Edit the `/etc/neutron/plugins/ml2/ml2_conf.ini`

```
[ml2] (enable networks)
type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = linuxbridge,l2population
extension_drivers = port_security
```

```
[ml2_type_flat]
flat_networks = provider
vni_ranges = 1:1000
```

```
[securitygroup]
enable_ipset = True
```

### Configure the Linux bridge agent:

The Linux bridge agent builds layer-2 (bridging and switching) virtual networking infrastructure for instances and handles security groups.

Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini` file

```
[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME
```

```
[vxlan]
enable_vxlan = True
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = True
```

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

### Configure the Layer-3 agent:

The Layer-3 (L3) agent provides routing and NAT services for self-service virtual networks.

Edit the `/etc/neutron/l3_agent.ini` file

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
```

Configure DHCP agent:

Edit the `/etc/neutron/dhcp_agent.ini` file

```
[DEFAULT]
interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
```

### (3) Configure the metadata agent

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

Edit the `/etc/neutron/metadata_agent.ini`

```
[DEFAULT]
nova_metadata_ip = controller
metadata_proxy_shared_secret = METADATA_SECRET
```

### (4) Configure the Computing Service to use the Networking service

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

```
[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

### (5) Final configuration

Populate the database

```
su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf \
  --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

Restart Services:

```
service nova-api restart
service neutron-server restart
service neutron-linuxbridge-agent restart
service neutron-dhcp-agent restart
service neutron-metadata-agent restart
service neutron-l3-agent restart
```

## **Compute Node**

### (1) Install and configure packages

```
apt install neutron-linuxbridge-agent
```

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

```
[database] comment out any connection options because compute nodes do not
directly access the database.
```

```
[DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@controller
```

```

auth_strategy = keystone

[keystone_authtoken] (Comment out any other options)
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS

```

## (2) Configure Networking Option

### Configure Linux bridge agent:

Edit the `/etc/neutron/plugins/ml2/linuxbridge_agent.ini`

```

[linux_bridge]
physical_interface_mappings = provider:PROVIDER_INTERFACE_NAME

[vxlan]
enable_vxlan = True
local_ip = OVERLAY_INTERFACE_IP_ADDRESS
l2_population = True

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

```

## (3) Configure Computing service to use Networking service

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

```

[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS

```

## (4) Final configuration

```

service nova-compute restart
service neutron-linuxbridge-agent restart

```

## PART IV: OpenStack Operations

In this part, we introduce some basic operations of OpenStack with Dashboard service or Command Line Tool. For the easy understanding and the logical introduction, we use Launching An Virtual Machine as our task.

### Dashboard service – Horizon

The Dashboard (horizon) is a web interface that enables cloud administrators and users to manage various OpenStack resources and services.

We use the Apache Web Server to deploy dashboard.

Dashboard is installed on the controller node.

#### Dashboard Installation and Configuration

Install package:

```
apt install openstack-dashboard
```

Edit the `/etc/openstack-dashboard/local_settings.py`

Configure the dashboard to use OpenStack services on the controller node

```
OPENSTACK_HOST = "controller"
```

Allow all hosts to access the dashboard:

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

Configure the memcached session storage service: (Comment out any other session storage configuration)

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
```

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

Enable the Identity API version 3:

```
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

Enable support for domains:

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

Configure API versions:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
```

```
    "volume": 2,  
}
```

Configure default as the default domain for users that you create via the dashboard:

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
```

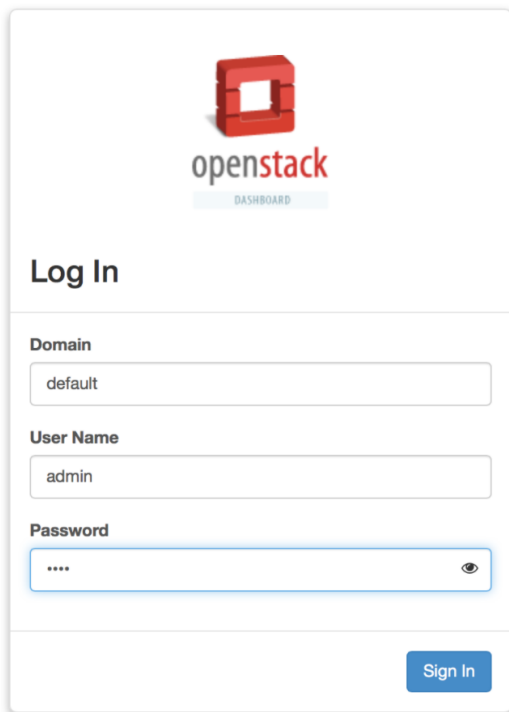
Start Dashboad service:

```
service apache2 reload
```

## Launch instance with Dashboard

### Login to Dashboard

Now you can access your Dashboard using a web browser at <http://controller/horizon>

The image shows the OpenStack Dashboard login interface. At the top, there is the OpenStack logo (a red cube) and the text "openstack DASHBOARD". Below this is a "Log In" section. It contains three input fields: "Domain" with the value "default", "User Name" with the value "admin", and "Password" with four asterisks. A "Sign In" button is located at the bottom right of the form.

The Apache web server is hosting the Horizon service. Google chrome, Safari and Firefox are good browser to access Dashboard.

### Upload and manage images

1. Select the appropriate project from the drop down menu at the top left.
2. On the Project tab, open the Compute tab and click Images category.
3. Click Create Image.

The Create An Image dialog box appears.



Create An Image

Name \*

Description

Image Source

Image Location

Image Location ?

Format \*

Select format

Architecture

Minimum Disk (GB) ?

Minimum RAM (MB) ?

☒ Copy Data ?

☐ Public

☐ Protected

Cancel

Create Image

Description:

Specify an image to upload to the Image Service.

Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)

**Please note:** The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.

### Configure access and security group

Before you launch an instance, you should add security group rules to enable users to ping and use SSH to connect to the instance. Security groups are sets of IP filter rules that define networking access and are applied to all instances within a project. To do so, you either add rules to the default security group. Add a rule to the default security group or add a new security group with rules.

Modify the rules in the security group:

Security Groups

Keypairs Floating IPs API Access

Security Groups

+ Create Security Group

Delete Security Groups

<input type="checkbox"/>	Name	Description	Actions
<input type="checkbox"/>	default	default	<div>Edit Rules</div>

Displaying 1 item

- On the Project tab, open the Network tab. The Security Groups tab shows the security groups that are available for this project.
- Select the default security group and click Manage Rules.
- To allow SSH access, click Add Rule.
- In the Add Rule dialog box, enter the following values  
Rule: SSH  
Remote: CIDR  
CIDR: 0.0.0.0/0
- Click Add.

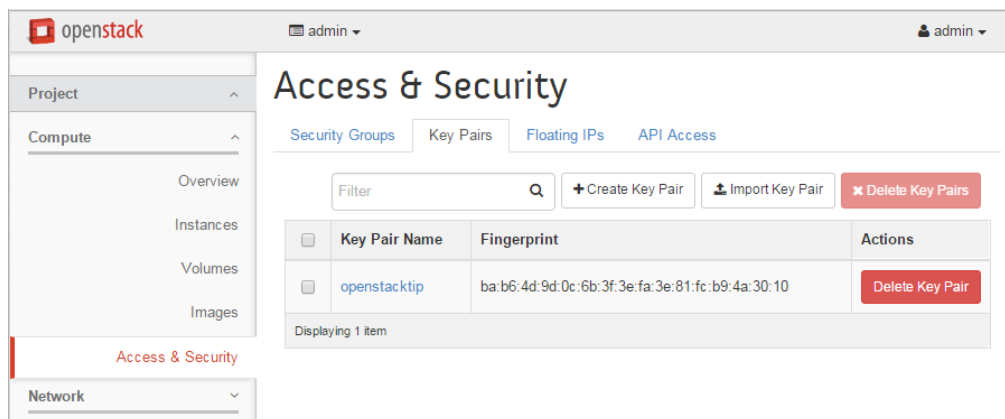
Instances will now have SSH port 22 open for requests from any IP address.

6. Add an ICMP rule, click Add Rule.
7. In the Add Rule dialog box, enter the following values:  
Rule: All ICMP  
Direction: Ingress  
Remote: CIDR  
CIDR: 0.0.0.0/0
8. Click Add.

### Add a key pair

Create a key pair and use it to launch the VM later.

1. Select the appropriate project from the drop down menu at the top left.
2. On the Project tab, open the Compute tab.
3. Click the Key Pairs tab, which shows the key pairs that are available for this project.
4. Click Create Key Pair.
5. In the Create Key Pair dialog box, enter a name for your key pair, and click Create Key Pair.
6. Respond to the prompt to download the key pair.



### Launch an instance

1. On the Project tab, open the Compute tab and click Instances category.
2. The dashboard shows the instances with its name, its private and floating IP addresses, size, status, task, power state, and so on.
3. Click Launch Instance.
4. In the Launch Instance dialog box, specify the following values:
  - Details: enter the name to the VM
  - Source: choose instance boot source and volume
  - Flavor: specify the size / type of the instance to launch the VM

- Network: select the network to add the VM. We choose self-service network
- Ports: Activate the ports that you want to assign to the instance.
- Security group: Activate the security groups that you want to assign to the instance. Security groups are a kind of cloud firewall that define which incoming network traffic is forwarded to instances. We choose the "default group" configured in the previous section.
- Key Pair: The ssh key to access the VM
- Configuration: Specify a customization script that runs after your instance launches.
- Metadata: Add Metadata items to your instance.

Launch Instance ✕

Details

**Source \***

Flavor \*

Networks \*

Network Ports

Security Groups

Key Pair

Configuration

Metadata

Instance source is the template used to create an instance. You can use a snapshot of an existing instance, an image, or a volume (if enabled). You can also choose to use persistent storage by creating a new volume. ?

Select Boot Source

Image ▼

Create New Volume

Yes No

Allocated

Name	Updated	Size	Type	Visibility
Select a source from those listed below.				

▼ Available 1 Select one

Q Click here for filters.

Name ^	Updated	Size	Type	Visibility	
> tecmint-test	4/25/16 7:11 PM	11.93 MB	QCOW2	Private	<span>+</span>

✕ Cancel
< Back
Next >
Launch Instance

## Launch An instance with command line tool

We want to launch an VM on the self-service network. The steps are following:

To launch an instance, you must at least specify the flavor, image name, network, security group, key, and instance name.



## **Launch An Instance**

1. On the controller node, source the demo credentials to gain access to user-only CLI commands:

```
. demo-openrc
```

2. Check the flavors on file, which includes processors, memory, and storage.

```
openstack flavor list
```

ID	Name	RAM	Disk	Ephemeral	VCPUs	Is Public
0	m1.nano	64	1	0	1	True

3. check the availble images

```
openstack image list
```

ID	Name	Status
390eb5f7-8d49-41ec-95b7-68c0d5d54b34	cirros	active

4. list available networks

```
openstack network list
```

ID	Name	Subnets
4716ddfe-6e60-40e7-b2a8-42e57bf3c31c	selfservice	2112d5eb-f9d6-45fd-906e-7cabd38b7c7c
b5b6993c-ddf9-40e7-91d0-86806a42edb8	provider	310911f6-acf0-4a47-824e-3032916582ff

5. list available security groups:

```
openstack security group list
```

ID	Name	Description
dd2b614c-3dad-48ed-958b-b155a3b38515	default	Default security group

The "default" group is edited by us before.

6. Launch a instance using Cirros image.

```
openstack server create --flavor m1.nano --image cirros \  
  --nic net-id=SELFSERVICE_NET_ID --security-group default \
```

--key-name mykey VM\_name

-- flavor: choose the instance size

-- image: pick a virtual os

-- nic: here we need to refer selfservice network ID

-- key-name: the SSH key we generated

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	7KTBYHSjEz7E
config_drive	
created	2016-02-26T14:52:37Z
flavor	m1.nano
hostId	
id	113c5892-e58e-4093-88c7-e33f502eaaa4
image	cirros (390eb5f7-8d49-41ec-95b7-68c0d5d54b34)
key_name	mykey
name	selfservice-instance

```

|           os-extended-volumes:volumes_attached           |           []
|
|   progress                                               |   0
|
|   project_id                                             |   ed0b60bf607743088218b0a533d5943f
|
|   properties                                             |
|
|   security_groups                                       |   [{u'name': u'default'}]
|
|   status                                                 |   BUILD
|
|   updated                                               |   2016-02-26T14:52:38Z
|
|   user_id                                               |   58126687cbcc4888bfa9ab73a2256f27
|
+-----+-----+-----+-----+
+

```

## 7. Then check the launching status of your instance

`openstack server list`

```

+-----+-----+-----+-----+
---+
| ID                | Name                | Status | Networks |
|
+-----+-----+-----+-----+
---+
| 113c5892-e58e-4093-88c7-e33f502eaaa4 | selfservice-instance | ACTIVE |          |
selfservice=172.16.1.3 |
| 181c52ba-aebc-4c32-a97d-2e8e82e4eaaaf | provider-instance   | ACTIVE |          |
provider=203.0.113.103 |
+-----+-----+-----+-----+
---+

```

Status changes from "BUILD" to "ACTIVE".

ACTIVE means the instance is running successfully.

### **Try to access the new instance.**

1. Access the instance using a virtual console.

Obtain a Virtual Network Computing (VNC) session URL for your instance and access it from a web browser:

(Virtual Network Computing (VNC)

Open source GUI and CLI tools used for remote console access to VMs. Supported by Compute.)

`openstack console url show selfservice-instance`

```

+-----+-----+-----+-----+
| Field | Value |
+-----+-----+-----+-----+
| type  | novnc |
| url   | http://controller:6080/vnc_auto.html?token=5eeccb47-525c-4918-ac2a-3ad1e9f1f493 |
+-----+-----+-----+-----+

```

Then use your web browser to access the above URL.

## 2. Access the instance remotely

First, we need to create floating IP address on the provider network (campus network / our local network)

(floating IP address

An IP address that a project can associate with a VM so that the instance has the same public IP address each time that it boots. You create a pool of floating IP addresses and assign them to instances as they are launched to maintain a consistent IP address for maintaining DNS assignment.)

```
openstack floating ip create provider
```

Field	Value
fixed_ip	None
id	3d05a9b1-b1af-4884-be1c-833a69744449
instance_id	None
ip	134.124.243.8
pool	provider

Then, associate the floating IP with the instance

```
openstack server add floating ip selfservice-instance 134.124.243.8
```

Now you can use your Key to ssh the instane:

```
ssh -i yourkey cirros@134.124.243.8
```

Beside, you can check the assignment status of floating IP.

```
openstack server list
```

ID	Name	Status	Networks
113c5892-e58e-4093-88c7-e33f502eaaa4	selfservice-instance	ACTIVE	selfservice=10.0.0.14, 134.124.243.8
181c52ba-aebc-4c32-a97d-2e8e82e4eaa4	provider-instance	ACTIVE	provider=134.124.243.12