OpenStack Installation (Icehouse)

OpenStack

OpenStack is a most prominent open-source middle ware software for cloud computing. OpenStack is driven and supported not only large open-source community but also by large number of big commercial players like Redhat, HP etc. OpenStack is primary used to deploy infrastructure-as-service and consist of many technologies like networking, storage, compute and all these technologies are integrated under the one umbrella, called as OpenStack as shown in figure 1.

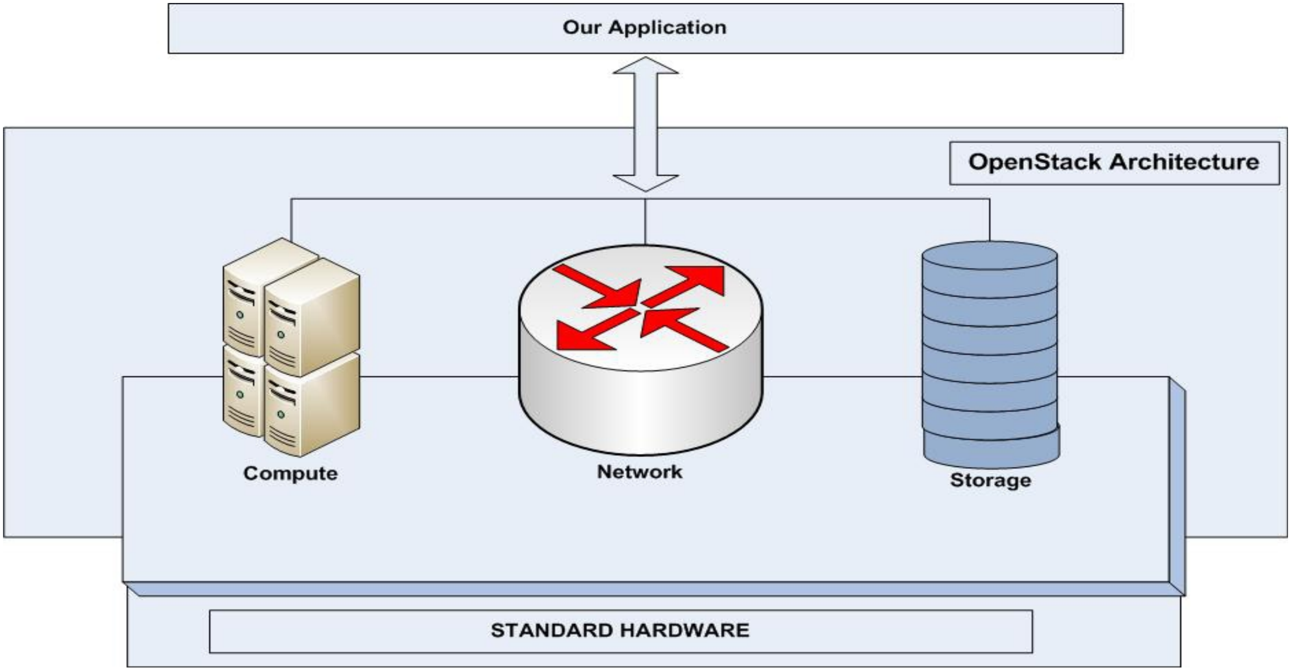

Fig. 1 OpenStack Architecture

Features

OpenStack is enriched collection of many features. OpenStack is developed with an idea modular approach and each module is a separate project. These modules work as separate and ease to plug and play with other software stack of OpenStack. OpenStack support all type of hardware and also support private, public and hybrid cloud. OpenStack software access and controls large number of compute, storage (object and block), and networking resources throughout a center and simply managed through a web interface or via the OpenStack access API. OpenStack also works with many popular enterprise and open source technologies making it ideal for heterogeneous infrastructure.

Component

They are many modules, which are provides with different version of openstack and we have also used many modules but we have discuss only few important modules only, which are used in to form our private cloud.

➢ Compute

Compute play an important role in formation of private cloud. On compute side OpenStack provides and enable the large support for virtualization. OpenStack support all type of hardware, software and even heterogeneous environment with no proprietary issue. OpenStack support multiples hypervisors in a virtualized environment. Both KVM and Xen are most popular choices for hypervisors. Now days, openstack extend its support Linux Container technology, LXC, where users wish to minimize virtualization overhead and achieve greater efficiency and performance.

- **Storage**

Storage is very important part of private cloud; OpenStack provides and support various type of storage from simple storage like for image to very high level Storage like object Storage. Majorly two type of storage are used into our private cloud.

- ✓ **Object Storage**

Object Storage [5] is very cost effective and having scale-out architecture. It provides a fully distributed and having API based accessible methods, that can be integrated or directly used into an applications and also used for various purpose like backup, archiving and data retention. Storage can be scale horizontally simply by adding new storage nodes. In case of a node or hard drive fail, openstack replicates its content to the other active nodes at different locations in the cluster. Because OpenStack having inbuilt capability and logic to ensure the data replication and distribution across different nodes, for this purpose inexpensive commodity hard drives and servers can be used.

- ✓ **Block Storage**

Block Storage is one of the best and ease of use among all the available storage methods provides by the openstack. Block Storage allow us to connect storage as an external hard disk, which we can use as a plug & play device, with a compute instance for better performance and integration with enterprise storage platforms. Block storage is best suit for where data will be used by various compute instances for different purpose such as database storage, expandable file systems, or providing a server with access to raw block level storage. If in a case, Block Storage is not in use than take a snapshot of that block storage for backup data. Block Storage having great facility of snapshot and restore back when needed again.

- **Network**

OpenStack Networking is a scalable, plug-gable and easily access through API for managing network devices and IP addresses. Like other component of the cloud computing, it will be controlled by the administrator with little user control or access. User will access the cloud resources within the access policy, define by the administrator. OpenStack provides full support with different variant and vendors. OpenStack Networking, software define networking (SDN), is supported by the world best known networking vendors such as Cisco, dell etc. OpenStack ensure the network will not be the bottleneck or limiting factor while access and deploy of the cloud.

Implementation

OpenStack offers highly modular architecture that offer great support and easy implementation. OpenStack can be implement in various form ranging from single level machine to multi nodes cluster(Three-node architecture). We have implement both type of architectures. Basically, Single level machine installation we had used for demo purpose and multi nodes cluster installation for the production cloud. In this paper we are discussed about the multi nodes architecture.

- **Pre-installation Requirement**

As discussed above, we had installed multi nodes architecture and minimum three nodes are required. Basically, for multi nodes installation there is not such hard hardware requirements.

- ✓ Controller Node: 1 processor, 2 GB memory, and 5 GB storage and 2 NIC.
- ✓ Network Node: 1 processor, 512 MB memory, and 5 GB storage and 3 NIC.
- ✓ Compute Node: 1 processor, 2 GB memory, and 10 GB storage and 2 NIC.

To synchronize the cluster's, we to setup NTP server and controller nodes will act as NTP server and in rest network and other compute nodes would be synchronize with the controller node and also all the nodes in the cluster except controller node having mysql client service and on controller mysql databases has been installed. Controller Node also contains the messaging server for passing message across the nodes and we have used the RabbitMQ server.

➢ **Installation**

As discussed previously, openstack provided more facility and option for installation, openstack as per user hardware availability. OpenStack will be installed to single system and also to form clusters. We have implement multi nodes installation architecture, in which minimum three nodes will be required. One node act as Controller Node, Second node act as Network Node and rest nodes work as Compute and other Server as shown in figure 2.

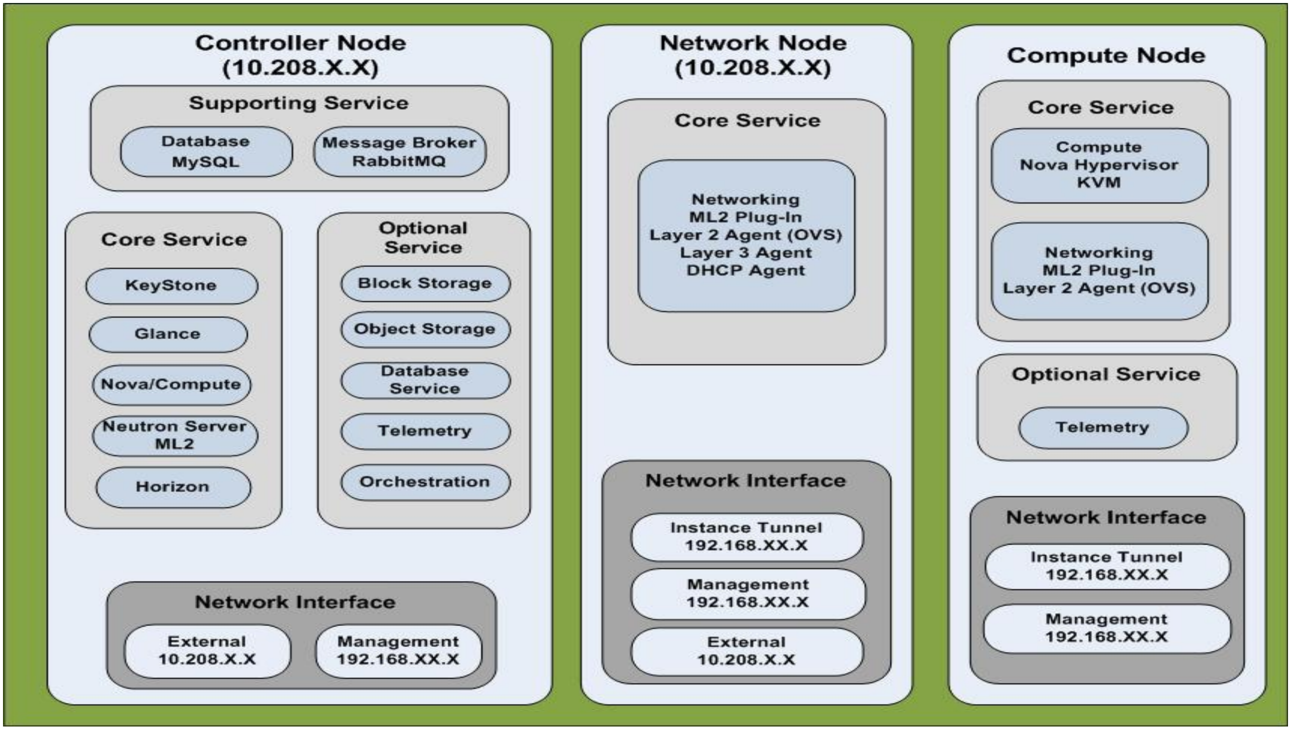

Fig 2. Shown the Service running at the nodes

Step 1: Configure the Keystone (Identity Service)

Keystone is a one of major project in the OpenStack software stack. Keystone provides Identify, Token, Credential, Catalog and Policy related to OpenStack. Keystone performs user management and service catalog. User management consists user's permission and tracking while service catalog consists availability of services with their API endpoints. All the installation done at Controller Nodes (10.208.X.X):

✓ Install the keystone

```
 # apt-get install keystone
```

✓ Open the Configuration file keystone.conf and add database connection in database section and other entries

```
 # vi /etc/keystone/keystone.conf
[database]
connection=mysql://keystone:keystone123@10.208.X.X/keystone
[DEFAULT]
# A "shared secret" between keystone and other openstack services
admin_token = admin123
```

```
log_dir = /var/log/keystone
```

✓ Create keystone database

```
$ mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'10.208.X.X' IDENTIFIED BY 'keystone123';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'keystone123';
mysql> exit
```

✓ Create the schema

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

✓ Restart the Keystone services

```
# service keystone restart
```

After restart add the admin and demo user and also add the various service and their endpoint URL.

Step 2: Configure the Glance (Image Service)

Glance service enables the openstack user to access, retrieve and store the images and snapshot. The default location storage of images and snapshot at controller nodes is /var/lib/glance/images/. Basically this service is installed and run at controller nodes. Glance run to services one is, glance-api to accept the image API requests for image discovery, retrieval, and storage and other one is, glance-registry to stores, processes, and retrieves meta-data about images.

✓ Installation of the glance at controller node(10.208.X.X)

```
# apt-get install glance python-glanceclient
```

✓ Open the Configuration file /etc/glance/glance-api.conf and /etc/glance/glance-registry.conf and edit the [database], [keystone] and [paste_deploy] section in each file

```
[database]
connection=mysql://glance:glance123@10.208.X.X/glance
…
[keystone_authtoken]
auth_uri = http://10.208.X.X:5000
auth_host = 10.208.X.X
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = glance123
...
[paste_deploy]
...
flavor = keystone
```

✓ Create glance database user

```
$ mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'10.208.X.X' IDENTIFIED BY 'glance123';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'glance123';
mysql> exit
```

✓ Create the glance schema into the database.

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

✓ Register the Glance Services with the Keystone service and create endpoint.

```
$ keystone service-create --name=glance --type=image --description="OpenStack Image Service"
$ keystone endpoint-create --service-id=$(keystone service-list | awk '/ image / {print $2}') –publicurl=http://10.208.X.X:9292 --internalurl=http://10.208.X.X:9292 --adminurl=http://10.208.X.X:9292
```

✓ Restart all the glance services

```
# cd /etc/init.d/; for i in $(ls glance-*); do sudo service $i restart; done
```

Step 3: Configure Nova services

Nova service is a core service or we can say its heart of the OpenStack. Nova is a main part of project from the starting of the OpenStack software stack. At starting of the OpenStack software stack, Nova service consist and do many task like networking, virtualization etc, but as OpenStack grows many services have been keep out as a separate project, but even today single node or two nodes installation nova service from many tasks.

Step 3.1: Configuration at Controller Node(10.208.X.X)

✓ Installation of the nova at the controller node(10.208.X.X)

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

✓ Open the Configuration file nova.conf, edit and add [database] and [DEFAULT] section

```
# vi /etc/nova/nova.conf

[database]
connection=mysql://nova:nova123@10.208.X.X/nova
[DEFAULT]
...
auth_strategy = keystone
…
rpc_backend = rabbit
rabbit_host = 10.208.X.X
rabbit_password = rabbit123
…
my_ip = 192.168.XX.X
vncserver_listen = 192.168.XX.X
```

```
vncserver_proxyclient_address = 192.168.XX.X
…
service_neutron_metadata_proxy = true
neutron_metadata_proxy_shared_secret = meta123
...
[keystone_authtoken]
...
auth_uri = http://10.208.X.X:5000
auth_host =  10.208.X.X
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova123
```

✓ Create a nova user into the keystone for authentication and authorization

```
$ keystone user-create --name=nova –pass=nova123 –email=nova@cdac.in
$ keystone user-role-add --user=nova --tenant=service –role=admin
```

✓ Register the Nova Services with the Keystone service and create endpoint.

```
  $ keystone service-create –name=nova --type=compute --description="OpenStack Compute"
  $ keystone endpoint-create --service-id=$(keystone service-list | awk '/ compute / {print $2}')
--publicurl=http://10.208.X.X:8774/v2/%\(tenant_id\)s   --internalurl=http://10.208.X.X:8774/v2/%
(tenant_id\)s --adminurl=http://10.208.X.X:8774/v2/%\(tenant_id\)s
```

✓ Restart the Nova Services

```
# cd /etc/init.d/; for i in $(ls nova-*); do sudo service $i restart; done
```

Step 3.2: Configuration at Compute Node(192.168.XX.X)

✓ Install the Nova Compute Service at Compute Nodes

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

✓ Edit the main configuration file /etc/nova/nova.conf and add in [DEFAULT] section

```
[DEFAULT]
...
auth_strategy = keystone
...
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://10.208.X.X:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = neutron123
neutron_admin_auth_url = http://10.208.X.X:35357/v2.0
linuxnet_interface_driver= nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
security_group_api = neutron
...
```

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:novadb@10.208.X.X/nova
...
[keystone_authtoken]
auth_uri = http://192.168.XX.X:5000
auth_host =  192.168.XX.X
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova123
…
rpc_backend = rabbit
rabbit_host = 192.168.XX.X
rabbit_password = rabbit123
...
my_ip = 192.168.XX.XX
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = 192.168.XX.XX
novncproxy_base_url=http://192.168.XX.X:6080/vnc_auto.html
...
glance_host =  192.168.XX.X
```

✓ Restart the Nova Compute Services

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

Step 4: Add a networking service(Neutron Service)

Networking is a major component and play important role in the success of any cloud. OpenStack provide many option and variance from various vendors and compatible with OpenStack software stacks. We have used the neutron with ml2 pluge-in.

Step 4.1: Configuration at Controller Node(10.208.X.X)

✓ Create neutron database with neutron user

```
$ mysql -u root -p
mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'10.208.X.X' IDENTIFIED BY 'neutron123';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' IDENTIFIED BY 'neutron123';
```

✓ Register the Neutron Services with the Keystone service and create endpoint.

```
$ keystone user-create --name neutron --pass NEUTRON_PASS --email neutron@cdac.in
$ keystone user-role-add --user neutron --tenant service --role admin
$ keystone service-create --name neutron --type network --description "OpenStack Networking"
$ keystone endpoint-create --service-id $(keystone service-list | awk '/ network / {print $2}') --publicurl http://10.208.X.X:9696 --adminurl http://10.208.X.X:9696 --internalurl http://10.208.X.X:9696
```

✓ Install the neutron server and ml2 plugin

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

✓ Edit the main configuration file /etc/neutron/neutron.conf and add in [database] and [DEFAULT] section

```
[database]
connection= mysql://neutron:neutron123@10.208.X.X/neutron

[DEFAULT]
….
auth_strategy = keystone
[keystone_authtoken]
...
auth_uri = http://10.208.X.X:5000
auth_host =  10.208.X.X
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = neutron123
...
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = 10.208.X.X
rabbit_password = rabbit123
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://10.208.X.X:8774/v2
nova_admin_username = nova
nova_admin_tenant_id= d0eae2882521477d9b556ea3f8064db2
nova_admin_password = nova123
nova_admin_auth_url = http://10.208.X.X:35357/v2.0
…
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
verbose = True
…
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://controller:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = neutron123
neutron_admin_auth_url = http://10.208.X.X:35357/v2.0
linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
security_group_api = neutron
```

✓ Edit and configure the ml2 files /etc/neutron/plugins/ml2/ml2_conf.ini and edit the section [ml2], [ml2_type_gre] and [securitygroup]

```
[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
```

✓ Restart the nova and neutron services

```
# cd /etc/init.d/; for i in $(ls nova-*); do sudo service $i restart; done
# cd /etc/init.d/; for i in $(ls neutron-*); do sudo service $i restart; done
```

Step 4.2: Configuration at Network Node(10.208.X.X)

Before start any installation process on network nodes one thing will be keep in mind, that network node has 3 NIC cards and one card act external, second act management and third one act as instances tunnel as shown in figure 3.

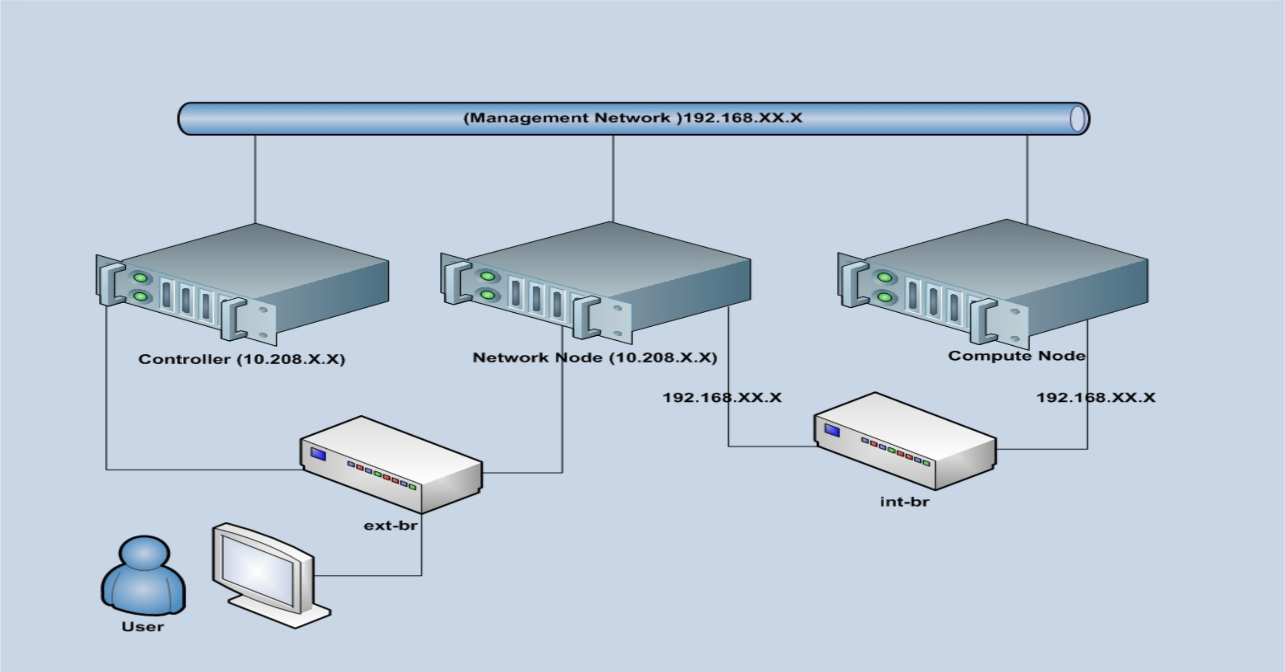

Fig 3. Network Connection between the nodes

✓ Edit the file /etc/sysctl.conf

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
net.bridge.bridge-nf-call-arptables=1
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
```

✓ Implement the change

```
# sysctl -p
```

✓ Download and Install neutron with dependent libraries

```
# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent neutron-l3-agent
neutron-dhcp-agent
```

✓ Edit the configuration file /etc/neutron/neutron.conf and add the following key to the [DEFAULT] and [keystone_authtoken] section

```
[DEFAULT]
...
auth_strategy = keystone
…
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = 10.208.X.X
rabbit_password = RABBIT_PASS
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
…
[keystone_authtoken]
...
auth_uri = http://10.208.X.X:5000
auth_host = 10.208.X.X
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = neutron123
```

✓ Now configure Layer-3(L3) agent, its provides the routing services for the instances. Edit the [DEFAULT] section of file /etc/neutron/l3_agent.ini

```
[DEFAULT]
...
interface_driver= neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True
```

✓ Now configure the DHCP agent provides DHCP services for the instances and edit the file /etc/neutron/dhcp_agent.ini and add modify the [DEFAULT] section

```
[DEFAULT]
...
interface_driver= neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
```

```
use_namespaces = True
dnsmasq_config_file = /etc/neutron/dnsmasq-neutron.conf
```

✓ Create file /etc/neutron/dnsmasq-neutron.conf and add line

```
dhcp-option-force=26,1454
```

✓ Kill all the dnsmasq processes

```
# killall dnsmasq
```

✓ Now configure the metadata agent, its provides the configuration information and the credential to access the instance remotely. Main configuration file is /etc/neutron/metadata_agent.ini

```
[DEFAULT]
...
auth_url = http://10.208.X.X:5000/v2.0
auth_region = cdacPune
admin_tenant_name = service
admin_user = neutron
admin_password = neutron123
nova_metadata_ip = 10.208.X.X
metadata_proxy_shared_secret = meta123
```

✓ Now configure the Modular Layer 2 (ml2) plug-in, provide the framework to build virtual network for the instances and the configuration file is /etc/neutron/plugins/ml2/ml2_conf.ini and add the section [ml2], [ml2_type_gre], [ovs] and [securitygroup]

```
[ml2]
...
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000

[ovs]
...
local_ip= 192.168.XX.X
tunnel_type = gre
enable_tunneling = True

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

✓ Now we need to configure Open vSwitch(OVS) service, provide support for the virtual network for the instances by direct and redirect the traffic. Restart the ovs services.

```
# service openvswitch-switch restart
```

✓ Add the integration bridge

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

- ✓ Restart all the network service

```
cd /etc/init.d/; for i in $(ls neutron-*); do sudo service $i restart; done
```

Step 4.3: Configuration at Compute Node

Before start any installation on network nodes one thing will be keep in mind, that network node has 3 NIC cards and one card act external, second one act management and third one act as instance tunnel.

✓ Edit the file /etc/sysctl.conf

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
net.bridge.bridge-nf-call-arptables=1
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1
```

✓ Implement the change

```
# sysctl -p
```

✓ Download and Install neutron service with depend libraries

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

✓ Edit the configuration file /etc/neutron/neutron.conf and add the following key to the [DEFAULT] and [keystone_authtoken] section

```
[DEFAULT]
...
auth_strategy = keystone
…
rpc_backend = neutron.openstack.common.rpc.impl_kombu rabbit_host = 10.208.X.X
rabbit_password = rabbit123
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
…
[keystone_authtoken]
...
auth_uri = http://10.208.X.X:5000
auth_host = 10.208.X.X
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = neutron123
```

✓ Now configure the Modular Layer 2 (ml2) plug-in, provide the framework to build virtual network for the instances and the main configuration file is /etc/neutron/plugins/ml2/ml2_conf.ini and add the section [ml2], [ml2_type_gre], [ovs] and [securitygroup]

```
[ml2]
...
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch
…
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000

[ovs]
local_ip= 192.168.XX.X
tunnel_type = gre
enable_tunneling = True
[securitygroup]
...
firewall_driver= neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
enable_security_group = True
```

✓ Now we need to configure Open vSwitch(OVS) service, provide support for the virtual network for the instances by direct and redirect the traffic. Restart the ovs services.

```
# service openvswitch-switch restart
```

✓ Add the integration bridge

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

✓ Restart all the network service

```
cd /etc/init.d/; for i in $(ls neutron-*); do sudo service $i restart; done
```

Step 5: Add the dashboard at Controller Node(10.208.X.X)

Although OpenStack based cloud is manage by command line but OpenStack also provides a beautiful gui dashboard and project name as Horizon. Horizon enable the user to deploy image, configure the virtual network and other thing.

✓ To horizon need at least Python 2.6
✓ Install the dashboard on the controller node(10.208..X.X)

```
# apt-get install apache2 memcached libapache2-mod-wsgi openstack-dashboard
```

✓ Update the Allow Host in local_setting of the Horizon and edit the file /etc/openstack-dashboard/local_settings.py

```
ALLOWED_HOSTS = ['localhost', 'my-desktop']
OPENSTACK_HOST = "10.208.X.X"
```

✓ Start the Apache server and memcached

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

Step 7: Launch an instance

All the major setup process has been over in the above steps, now its time to launch an instance. Before launch an instance we need upload image, create virtual network etc all these steps require time only for first time, later it a simple launch of instances.

✓ Setup the Environment variable for both admin and demo users.

```
$ vi admin.sh
export OS_USERNAME=admin
export OS_PASSWORD=admin123
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://10.208.X.X:35357/v2.0

$ source admin.sh
$ vi demo.sh
export OS_USERNAME=demo
export OS_PASSWORD=demo123
export OS_TENANT_NAME=demo
export OS_AUTH_URL=http://10.208.X.X:35357/v2.0
```

✓ Download the image from the net.

```
$ source admin.sh
$ mkdir /tmp/images
$ cd /tmp/images/
$ wget http://download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
```

✓ Upload the image into the OpenStack cloud

```
$ glance image-create --name "cirros-0.3.2-x86_64" --disk-format qcow2 --container-format
bare --is-public True --progress < cirros-0.3.2-x86_64-disk.img
$ glance image-list
+----------------------------------+------------------- +------------+-----------------+----------+--------+
| ID                               | Name              | Disk Format | Container Format | Size     | Status |
+----------------------------------+--------------------+------------+-----------------+----------+------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.2-x86_64 | qcow2 |bare | 13167616 | active |
+------------------------------------+--------------------+------------+-----------------+----------+--
```

✓ Create the external networked

```
$ source admin.sh
$ neutron net-create ext-net --shared --router:external=True
Created a new network:
+--------------------------+-------------------------------------+
| Field                    | Value                               |
+--------------------------+-------------------------------------+
| admin_state_up           | True                                |
| id                       | 893aebb9-1c1e-48be-8908-6b947f3237b3 |
| name                     | ext-net                             |
| provider:network_type    | gre                                 |
| provider:physical_network |                                    |
| provider:segmentation_id | 1                                   |
| router:external         | True                                  |
| shared                  | True                                  |
| status                  | ACTIVE                                |
| subnets                 |                                       |
| tenant_id               | 54cd044c64d5408b83f843d63624e0d8      |
+-------------------------+---------------------------------------+
```

✓ Create the external subnet

```
$ neutron subnet-create ext-net --name ext-subnet --allocation-pool start=10.208.X.X
,end=10.208.X.X --disable-dhcp --gateway 10.208.X.X 10.208.X.X/XX
```

✓ Create an internal network demo

```
$ source demo.sh
$ neutron net-create demo-net
Created a new network:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| admin_state_up | True                                 |
| id             | ac108952-6096-4243-adf4-bb6615b3de28 |
| name           | demo-net                             |
| shared         | False                                |
| status         | ACTIVE                               |
| subnets        |                                      |
| tenant_id      | cdef0071a0194d19ac6bb63802dc9bae     |
```

✓ Create an Subnet of demo network

```
$ neutron subnet-create demo-net --name demo-subnet --gateway 192.168.1.1 192.168.1.0/24
```

✓ Now create a router on the internal network and connect to an external network to access the instance from outside.

```
$ neutron router-create demo-router
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 635660ae-a254-4feb-8993-295aa9ec6418 |
| name                  | demo-router                          |
| status                | ACTIVE                               |
| tenant_id             | cdef0071a0194d19ac6bb63802dc9bae     |
+-----------------------+--------------------------------------+
$ neutron router-interface-add demo-router demo-subnet
```

✓ Attach the router to the external network and set the gateway

```
$ neutron router-gateway-set demo-router ext-net
```

✓ To verify and test by ping the router gateway ip addresses

```
$ ping -c 4 10.208.X.X
```

✓ Generate the public key

```
$ source demo.sh
$ ssh-keygen
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
$ nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

✓ Launch the instance

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=ac108952-6096-4243-
adf4-bb6615b3de28 --security-group default --key-name demo-key demo-instance1

$ nova list
+--------------------------------------+----------------+--------+------------+-------------+---------------------+
| ID                                   | Name           | Status | Task State |Power State | Networks     |
+--------------------------------------+----------------+--------+------------+-------------+---------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -          | Running     |
demo-net=192.168.X.X    |
```

✓ To access the instance remotely, add rule to default security list

```
$ 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 |              |
+-------------+-----------+---------+-----------+--------------+
$ 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 |              |
+-------------+-----------+---------+-----------+--------------+
```

✓ Create a floating IP from the external network ext-net

```
$ neutron floatingip-create ext-net
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 10.208.X.X                           |
| floating_network_id | 9bce64a3-a963-4c05-bfcd-161f708042d1 |
| id                 | 05e36754-e7f3-46bb-9eaa-3521623b3722 |
| port_id          |                                      |
| router_id       |                                      |
| status            | DOWN                                 |
| tenant_id       | 7cf50047f8df4824bc76c2fdf66d11ec     |
+---------------------+--------------------------------------+
```

✓ Assign the Floating IP to the demo-instance1 as shown in figure 4

```
$ nova floating-ip-associate demo-instance1 10.208.X.X
```

✓ Check the status of the Floating IP assign

```
$ nova list
+--------------------------------+----------------+--------+------------+---------+------------------------+
| ID                                    | Name            | Status | Task State | Power State | Networks
|+----------------------------+----------------+--------+------------+------------+--------------------------------
+| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | - | Running      | demo-net=192.168.X.X, 10.208.X.X    |
+----------------------------+----------------+--------+------------+-------------+--------------------------+
$ ping -c 4 10.208.X.X
```

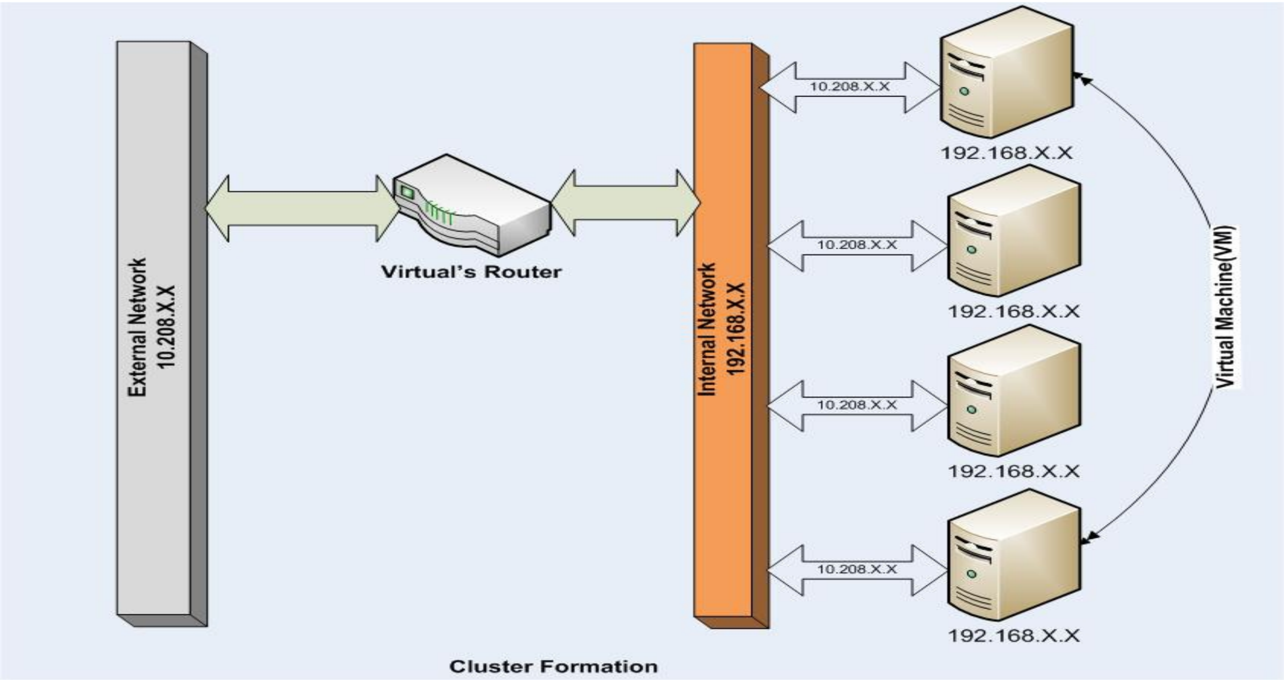

Fig 4. Instance Running and Virtual Network with Router