Assignment C 1: Open source Cloud Infrastructure

1 Title

Open source Cloud Infrastructure

2 Problem Definition

Installation of Open source Cloud Infrastructure

3 Learning Objective

To study Open source cloud deployment options. To implement and deploy cloud platform on your linux machine.

4 Learning Outcome

Succesfully deploying the cloud.

5 Software and Hardware Requirement

1. 64 bit open source LINUX
2. Openstack files
3. internet connectivity

6 Theory

Cloud computing
Cloud computing, also on-demand computing, is a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services),which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economy of scale, similar to a utility (like the electricity grid) over a network.
Advocates claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.Cloud providers typically use a "pay as you go" model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model.
The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing. Companies can scale up as computing needs increase and then scale down again as demands decrease.

Openstack
OpenStack is a free and open-source software platform for cloud computing, mostly deployed as an infrastructure-as-a-service (IaaS).The software platform consists of interrelated components that control hardware pools of processing, storage, and networking resources throughout a data center. Users either manage it through a web-based dashboard, through command-line tools, or through a RESTful API. OpenStack.org released it under the terms of the Apache

License.

Components of Openstack
There are basically eleven components of OpenStack (two of which were just included in the last Icehouse release), below is a quick breakdown of what they are called in OpenStack speak, and what they do.

OpenStack Compute (Nova)
OpenStack compute (codename: Nova) is the component which allows the user to create and manage virtual servers using the machine images. It is the brain of the Cloud. OpenStack compute provisions and manages large networks of virtual machines.

Block Storage (Cinder)
This component provides persistent block storage to running instances. The flexible architecture makes creating and managing block storage devices very easy.

Object Storage (Swift)
This component stores and retrieves unstructured data objects through the HTTP based APIs. Further, it is also fault tolerant due to its data replication and scale out architecture

OpenStack Networking (Neutron)
it is a pluggable, scalable and API-driven system for managing networks. OpenStack networking is useful for VLAN management, management of IP addresses to different VMs and management of firewalls using these components.

Identity Service (Keystone)
This provides a central directory of users mapped to the OpenStack services. It is used to provide an authentication and authorization service for other OpenStack services.

OpenStack Image Service (Glance)
This provides the discovery, registration and delivery services for the disk and server images. It stores and retrieves the virtual machine disk image.

Dashboard (Horizon)
This component provides a web-based portal to interact with all the underlying OpenStack services, such as NOVA, Neutron, etc.

Database as a Service (Trove)
Trove is Database as a Service for OpenStack. It's designed to run entirely on OpenStack, with the goal of allowing users to quickly and easily utilize the features of a relational database without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service will focus on providing resource isolation at high performance while automating complex administrative tasks including deployment, configuration, patching, backups, restores, and monitoring.

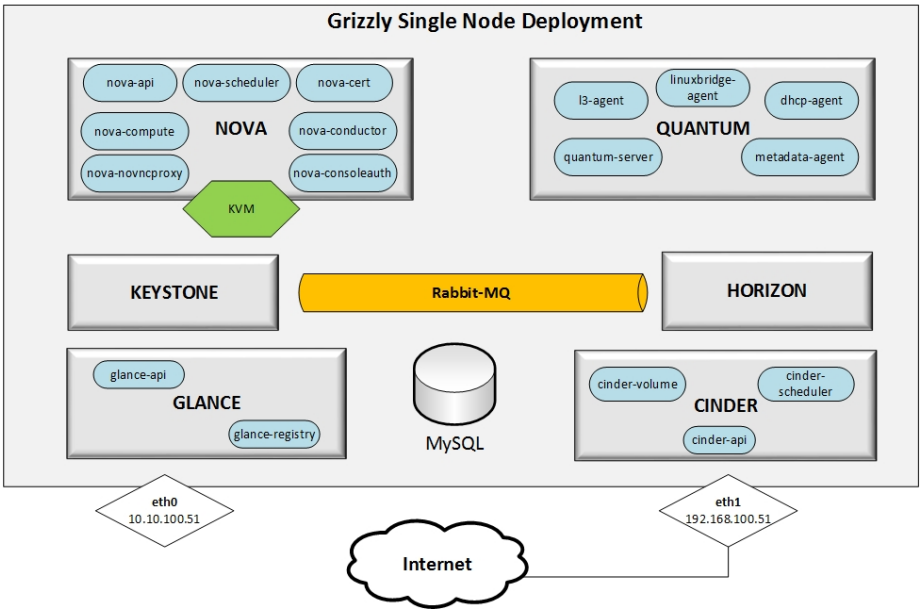


Figure 1: Openstack Architecture

7 Installation Steps :

1.Adding Repos

```
Add Grizzly repositories:
apt-get install ubuntu-cloud-keyring python-software-properties
    software-properties-common python-keyring
echo deb http://ubuntu-cloud.archive.canonical.com/ubuntu
    precise-updates/grizzly main >>/etc/apt/sources.list.d/grizzly.list

Update your system:
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

2.Configuring Network

```
Only one NIC should have an internet access (/etc/network/interfaces)
#For Exposing OpenStack API over the internet
auto eth1
iface eth1 inet static
address 192.168.100.51
netmask 255.255.255.0
gateway 192.168.100.1
dns-nameservers 8.8.8.8

#Not internet connected(used for OpenStack management)
auto eth0
iface eth0 inet static
address 10.10.100.51
netmask 255.255.255.0


Restart the networking service:
service networking restart
```

3.Installing MySQL and RabbitMQ

```
Install MySQL and specify a password for the root user:
apt-get install -y mysql-server python-mysqldb
```

```
Configure mysql to accept all incoming requests:
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart

Install RabbitMQ:
apt-get install -y rabbitmq-server
```

```
4.Other Services

Install NTP service:
apt-get install -y ntp

Install other services:
apt-get install -y vlan bridge-utils

Enable IP_Forwarding:
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/' /etc/sysctl.conf
# To save you from rebooting, perform the following
sysctl net.ipv4.ip_forward=1
```

```
5.Keystone
start by the keystone packages:
apt-get install -y keystone

Verify your keystone is running:
service keystone status

Create a new MySQL database for keystone:
mysql -u root -p
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystoneUser'@'%' IDENTIFIED BY
    'keystonePass';
quit;

Adapt the connection attribute in the /etc/keystone/keystone.conf to the
    new database:
connection = mysql://keystoneUser:keystonePass@10.10.100.51/keystone

Restart the identity service then synchronize the database:
service keystone restart
```

```
keystone-manage db_sync

To test Keystone, we use a simple CLI command:
keystone user-list
```

6.Glance

```
We Move now to Glance installation:
apt-get install -y glance

Verify your glance services are running:
service glance-api status
service glance-registry status

Create a new MySQL database for Glance:
mysql -u root -p
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glanceUser'@'%' IDENTIFIED BY 'glancePass';
quit;

Update /etc/glance/glance-api-paste.ini with:
[filter:authtoken]
paste.filter_factory =
    keystoneclient.middleware.auth_token:filter_factory
delay_auth_decision = true
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = service_pass

Update the /etc/glance/glance-registry-paste.ini with:
[filter:authtoken]
paste.filter_factory =
    keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = service_pass

Update /etc/glance/glance-api.conf with:
sql_connection = mysql://glanceUser:glancePass@10.10.100.51/glance
```

```
And:
[paste_deploy]
flavor = keystone
Update the /etc/glance/glance-registry.conf with:
sql_connection = mysql://glanceUser:glancePass@10.10.100.51/glance

And:
[paste_deploy]
flavor = keystone

Restart the glance-api and glance-registry services:
service glance-api restart; service glance-registry restart

Synchronize the glance database:
glance-manage db_sync

Restart the services again to take into account the new modifications:
service glance-registry restart; service glance-api restart

To test Glance, upload the cirros cloud image directly from the internet:
glance image-create --name myFirstImage --is-public true
    --container-format bare --disk-format qcow2 --location
    https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-disk.img

Now list the image to see what you have just uploaded:
glance image-list
```

```
7.Quantum/Neutron

Install the Quantum components:
apt-get install -y quantum-server quantum-plugin-linuxbridge
    quantum-plugin-linuxbridge-agent dnsmasq quantum-dhcp-agent
    quantum-l3-agent

Create a database:
mysql -u root -p
CREATE DATABASE quantum;
GRANT ALL ON quantum.* TO 'quantumUser'@'%' IDENTIFIED BY 'quantumPass';
quit;

Verify all Quantum components are running:
cd /etc/init.d/; for i in $( ls quantum-* ); do sudo service $i status;
    done
```

```
Edit the /etc/quantum/quantum.conf file:
core_plugin =
    quantum.plugins.linuxbridge.lb_quantum_plugin.LinuxBridgePluginV2

Edit /etc/quantum/api-paste.ini
[filter:authtoken]
paste.filter_factory =
    keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass

Edit the LinuxBridge plugin config file
    /etc/quantum/plugins/linuxbridge/linuxbridge_conf.ini with:
# under [DATABASE] section
sql_connection = mysql://quantumUser:quantumPass@10.10.100.51/quantum
# under [LINUX_BRIDGE] section
physical_interface_mappings = physnet1:eth1
# under [VLANS] section
tenant_network_type = vlan
network_vlan_ranges = physnet1:1000:2999

Edit the /etc/quantum/l3_agent.ini:
interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver

Update the /etc/quantum/quantum.conf:
[keystone_authtoken]
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass
signing_dir = /var/lib/quantum/keystone-signing

Edit the /etc/quantum/dhcp_agent.ini:
interface_driver = quantum.agent.linux.interface.BridgeInterfaceDriver

Update /etc/quantum/metadata_agent.ini:
# The Quantum user information for accessing the Quantum API.
auth_url = http://10.10.100.51:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass

# IP address used by Nova metadata server
```

```
nova_metadata_ip = 10.10.100.51

# TCP Port used by Nova metadata server
nova_metadata_port = 8775

metadata_proxy_shared_secret = helloOpenStack
Restart all quantum services:

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

8.1 KVM(NOVA)

```
make sure that your hardware enables virtualization:
apt-get install cpu-checker
kvm-ok

Normally you would get a good response. Now, move to install kvm and
    configure it:
apt-get install -y kvm libvirt-bin pm-utils

Edit the cgroup_device_acl array in the /etc/libvirt/qemu.conf file to:
cgroup_device_acl = [
"/dev/null", "/dev/full", "/dev/zero",
"/dev/random", "/dev/urandom",
"/dev/ptmx", "/dev/kvm", "/dev/kqemu",
"/dev/rtc", "/dev/hpet","/dev/net/tun"
]

Delete default virtual bridge
virsh net-destroy default
virsh net-undefine default

Enable live migration by updating /etc/libvirt/libvirtd.conf file:
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"

Edit libvirtd_opts variable in /etc/init/libvirt-bin.conf file:
env libvirtd_opts="-d -l"

Edit /etc/default/libvirt-bin file
libvirtd_opts="-d -l"

Restart the libvirt service to load the new values:
```

```
service libvirt-bin restart
```

8.2 NOVA

```
Start by installing nova components:
apt-get install -y nova-api nova-cert novnc nova-consoleauth
    nova-scheduler nova-novncproxy nova-doc nova-conductor
    nova-compute-kvm

Check the status of all nova-services:
cd /etc/init.d/; for i in $( ls nova-* ); do service $i status; cd; done

Prepare a Mysql database for Nova:
mysql -u root -p
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'novaUser'@'%' IDENTIFIED BY 'novaPass';
quit;

Now modify authtoken section in the /etc/nova/api-paste.ini file to this:
[filter:authtoken]
paste.filter_factory =
    keystoneclient.middleware.auth_token:filter_factory
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = service_pass
signing_dirname = /tmp/keystone-signing-nova
# Workaround for https://bugs.launchpad.net/nova/+bug/1154809
auth_version = v2.0

Synchronize your database:
nova-manage db sync

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

Check for the smiling faces on nova-* services to confirm your
    installation:
nova-manage service list
```

```
9.Cinder

Install the required packages:
apt-get install -y cinder-api cinder-scheduler cinder-volume iscsitarget
    open-iscsi iscsitarget-dkms

Configure the iscsi services:
sed -i 's/false/true/g' /etc/default/iscsitarget

Restart the services:
service iscsitarget start
service open-iscsi start

Prepare a Mysql database for Cinder:
mysql -u root -p
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinderUser'@'%' IDENTIFIED BY 'cinderPass';
quit;

Configure /etc/cinder/api-paste.ini like the following:
[filter:authtoken]
paste.filter_factory =
    keystoneclient.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.100.51
service_port = 5000
auth_host = 10.10.100.51
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = service_pass

Then, synchronize your database:
cinder-manage db sync

Restart the cinder services:
cd /etc/init.d/; for i in $( ls cinder-* ); do sudo service $i restart;
    done

Verify if cinder services are running:
cd /etc/init.d/; for i in $( ls cinder-* ); do sudo service $i status;
    done
```

10.Horizon

To install horizon, proceed like this
apt-get install openstack-dashboard memcached

Reload Apache and memcached:
service apache2 restart; service memcached restart

You can now access your OpenStack 192.168.100.51/horizon with
credentials admin:admin_pass.

11.Creating the first VM

To start your first VM, we first need to create a new tenant, user and
internal network.

Create a new tenant
keystone tenant-create --name project_one

Create a new user and assign the member role to it in the new tenant
(keystone role-list to get the appropriate id):
keystone user-create --name=user_one --pass=user_one --tenant-id
$put_id_of_project_one --email=user_one@domain.com
keystone user-role-add --tenant-id $put_id_of_project_one --user-id
$put_id_of_user_one --role-id $put_id_of_member_role

Create a new network for the tenant:
quantum net-create --tenant-id $put_id_of_project_one net_proj_one

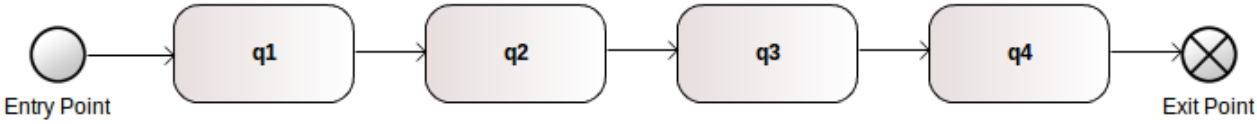
Create a new subnet inside the new tenant network:
quantum subnet-create --tenant-id $put_id_of_project_one net_proj_one
50.50.1.0/24

Create a router for the new tenant:
quantum router-create --tenant-id $put_id_of_project_one router_proj_one

Add the router to the subnet:
quantum router-interface-add $put_router_proj_one_id_here
$put_subnet_id_here

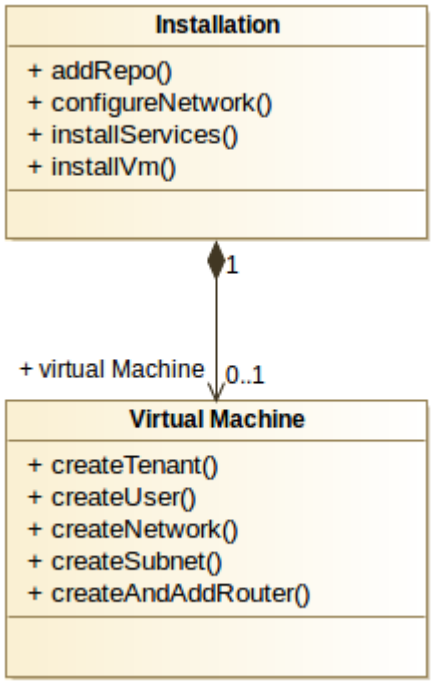
Restart all quantum services:
cd /etc/init.d/; for i in $(ls quantum-*); do sudo service $i restart;
done

State Diagram

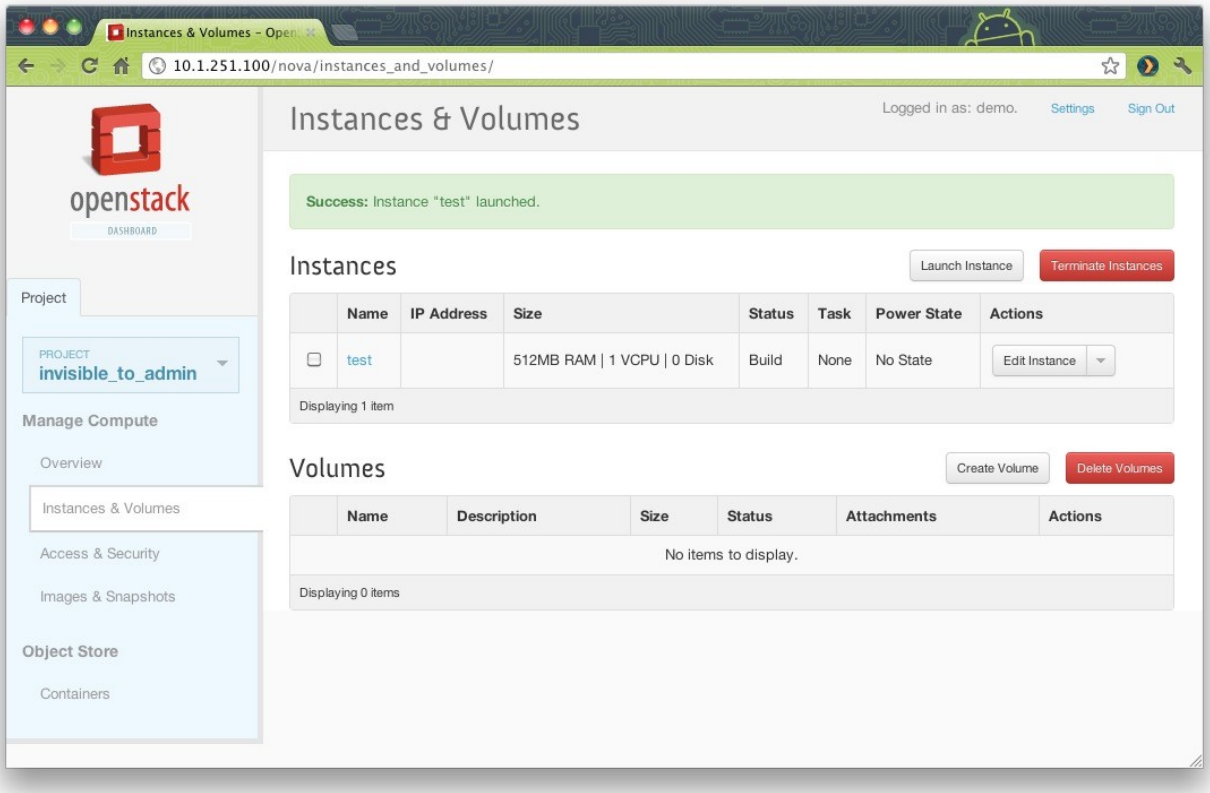


q1 = Add Repos
q2 = Configure Network
q3 = Install services
q4 = Create Virtual Machine

Class Diagram :



Screenshots:



Conclusion :

We successfully implemented and deployed Openstack cloud