Mansoura University
Faculty Of Engineering
Dept. of Electronics and
Communications Engineering

Open Source Integrated Infra-Structure Using Ansible Configuration Management

2016

Supervised By

Dr. Ahmed Elnakib

Mansoura University
Faculty Of Engineering
Dept. of Electronics and Communications Engineering

Open Source Integrated Infra-structure Using Ansible Configuration Management

A Graduation Project

2016

Supervised By

Dr. Ahmed Elnakib

Presented By

Dyaa Eldin Ahmed Mohamed Khalil

Shimaa Mohamed Abd-Elghany Elboghdady

Shimaa Shehata Ahmed Elmeligy

2016

Table of contents

Appendix Table

Appendix Table

Preface

What you need to apply this project ?

To be able apply this project , you will need root access to computers or servers that have hardware virtualization capabilities. To set up the labs environments you will install and use RHEL , or Centos ISO images .

Who this project is for ?

This project is aimed at system administrators who are familiar with open source systems . Knowledge and experience of managing Linux environments is required .

Feedback :

Feedback from our readers is always welcome. Let us know what you think about our project idea , what you liked or may have disliked .
To send us general feedback, simply send an e-mail to smart.tuxproj@gmail.com . You can also check our LinkedIn account : https://eg.linkedin.com/in/the-smart-tux-93a904122

Chapter 1
Introduction

Introduction

Data centers are **physical** or **virtual** infrastructure used by enterprises to house computer, server and networking systems and components for the company's information technology (IT) needs, which typically involve **storing**, **processing** and **serving** large amounts of mission-critical data to clients in a client/server architecture. But data centers have some difficulties such as :

- Needs large effort to deploy .
- Admins need More experience to start .
- Servers maybe down .
- Data may be lost .
- Low security .
- Large hardware and cost .

1.1 Objectives :

Our project aims to build integrated infrastructure design 'data center' with some improvements to achieve :

- More stability .
- Faster deployment .
- Higher security .
- More data reliable.
- Less errors .
- More availability .
- Less hardware .

1.2 Introduction to project component :

Project is specifically designed to give the confidence and understanding to create infrastructure design using several products such as :

1.2.1. OpenStack cloud computing :

Which is open source cloud computing platform , Provides an Infrastructure-as-a-Service "IaaS" solution using some services , aims for simple implementation, massive scalability, and a rich set of features . OpenStack consist of 3 nodes :

1. Controller node : manage all OpenStack services and nodes using command line or dashboard .

2. Compute nodes : runs the hypervisor portion of Compute that operates instances .

3. Storage node : contains the disks that the Block Storage and Shared File System services provision for instances .

1.2.2. Ceph storage :

Ceph is open source project , distribute object storage designed to provide high performance, reliability and scalability . Ceph consist of 3 nodes :

1. **Admin nodes :** used to manage and deploy all clusters .
2. **Monitor nodes :** used to monitor all clusters and it is have a map to all of them .
3. **OSD nodes :** used to store data .

1.2.3. Pacemaker cluster :

A group of servers and other resources that act like a single system to provide high-availability services and resources by redundant multiple machines .

1.2.4. Software firewall :

which is a network security that provides secure connectivity between networks (internal/external) . Prevent hackers, spam, spyware and viruses to reach a computer or (part of a) network . In our project we used Iptables . Iptables is a type of software firewall implemented in Linux kernel which able to allow, block or forward traffic .

1.2.5. Ansible configuration management :

Ansible is an open source automation tool for configuring , managing and deploying all servers at the same time instead of managing each server individually . Ansible has several features such as :

- Easy to read .
- Easy (to learn and setup) .
- Large number of modules .
- Run on OpenSSH .
- Agent less .

1.3 Motivation :

When we tried to create data center we face many problems, and found it is difficult to deploy specially because no resources and references available can help us . So data center deployment became difficult operation . We are sure that many people met the same problems, so we decided to try solving it by helping them to create their data center using easier and faster way .

1.4 Products history :

Year	Product
2010	**OpenStack began as a joint project of Rackspace Hosting and NASA .**
2003	**Ceph was developed at University of California, Santa Cruz by Sage Weil as a part of his PhD project .**
2006	**Ceph made open source under a Lesser GNU Public License (LGPL) .**
2014	**Red Hat acquire Ceph storage .**
2008	**Red Hat Clustering initial release .**
2012	**Ansible initial the first release written in python .**

1.5 How this project is different ?

When administrators need to make deployment it was necessary to study each product used in the data center, it is difficult operation and waste of time, so we start to provide them a tool to be able to automatically deploy the data center with fast way, less time and less efforts .

1.6 Project contents :

Now, we will gives you a brief about the next chapters content in this documentation as shown :

Chapter 3, Starting OpenStack cloud computing environment, the chapter describe its components , installation and configuration steps .

Chapter 4, Installing Ceph storage, how to configure and use it as centralized Object or Block storage environment .

Chapter 5, OpenStack Integration with Ceph storage and clusters , how to configure Ceph to be OpenStack backend storage .

Chapter 6, How to install , manage and configure cluster which is a group of servers and other resources that act like a single system to provide high-availability services and resources by redundant multiple machines , this redundancy is used guard against failure of many types .

Chapter 7, Setting firewall rules and using the clusters instances to install and configure services such as IPA which is a centralized authentication, DNS to resolve from IP to name and name to IP , web developers servers .

Chapter 8, How to automate each product to automatically run using Ansible configuration management .

1.7 What this documentation cover ?

This documentation covers a wide range of topics that helps to install, configure, integrate and automate each product . Open Source Integrated Infra-structure using Ansible configuration management project helps to know :

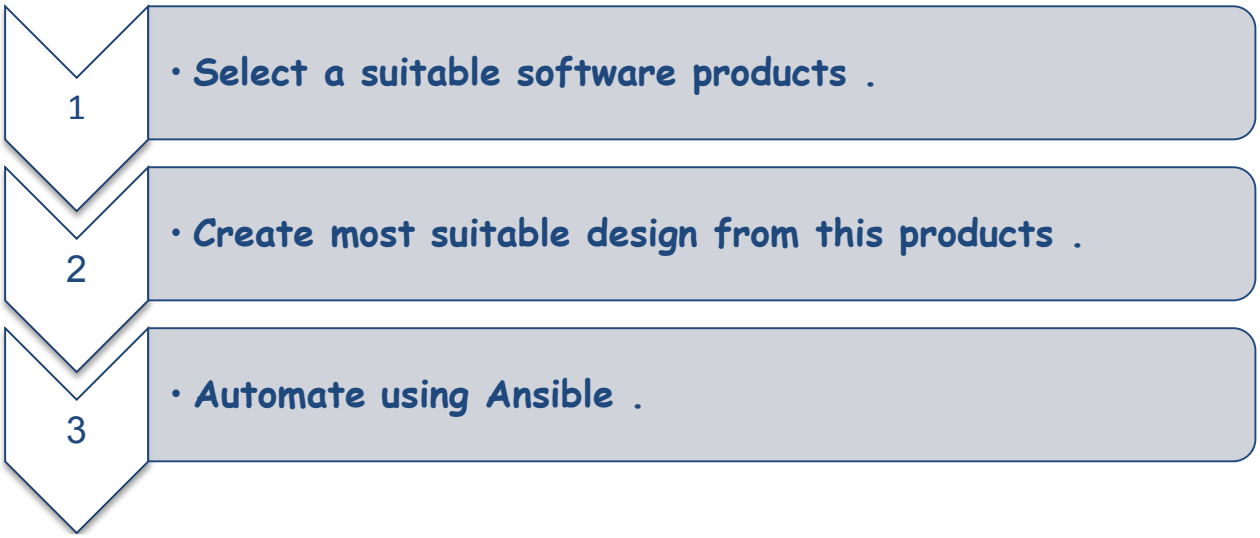
- How to install and configure all the core components of each product to run an environment that can be managed and operated easily way .
- Practical, real design contain several products and services in each chapter, allowing you to progress with the confidence that they will work in your own environments.
- How to manage and automate the complete design with storage backend to improve highly redundant and highly available storage .

It gives a clear, step-by-step instructions to design , deploy and automate your data center successfully. It is full of practical and applicable recipes that enable you to use the latest capabilities of used products and implement them .

Chapter 2
Project design

Project design

The Main steps we followed to reach our project goal is divided into 3 stages :



2.1 Selective products of the project :

The products we have been chosen are :

- OpenStack cloud computing .
- Ceph storage .
- Pacemaker cluster .
- Software firewall .
- Ansible .

Note : we list a brief about each product in the last chapter , and the details explanation will be in the next chapters .

2.2 Project Design :

Figure 2.1 Open Source Integrated Infra-structure Design using Ansible management show that :

1. Remote user connect only to web cluster .
2. Web developer connect only to internet and web cluster .
3. Admin nodes connect to all infrastructure .
4. Only admins can connect directly to OpenStack or Ceph .

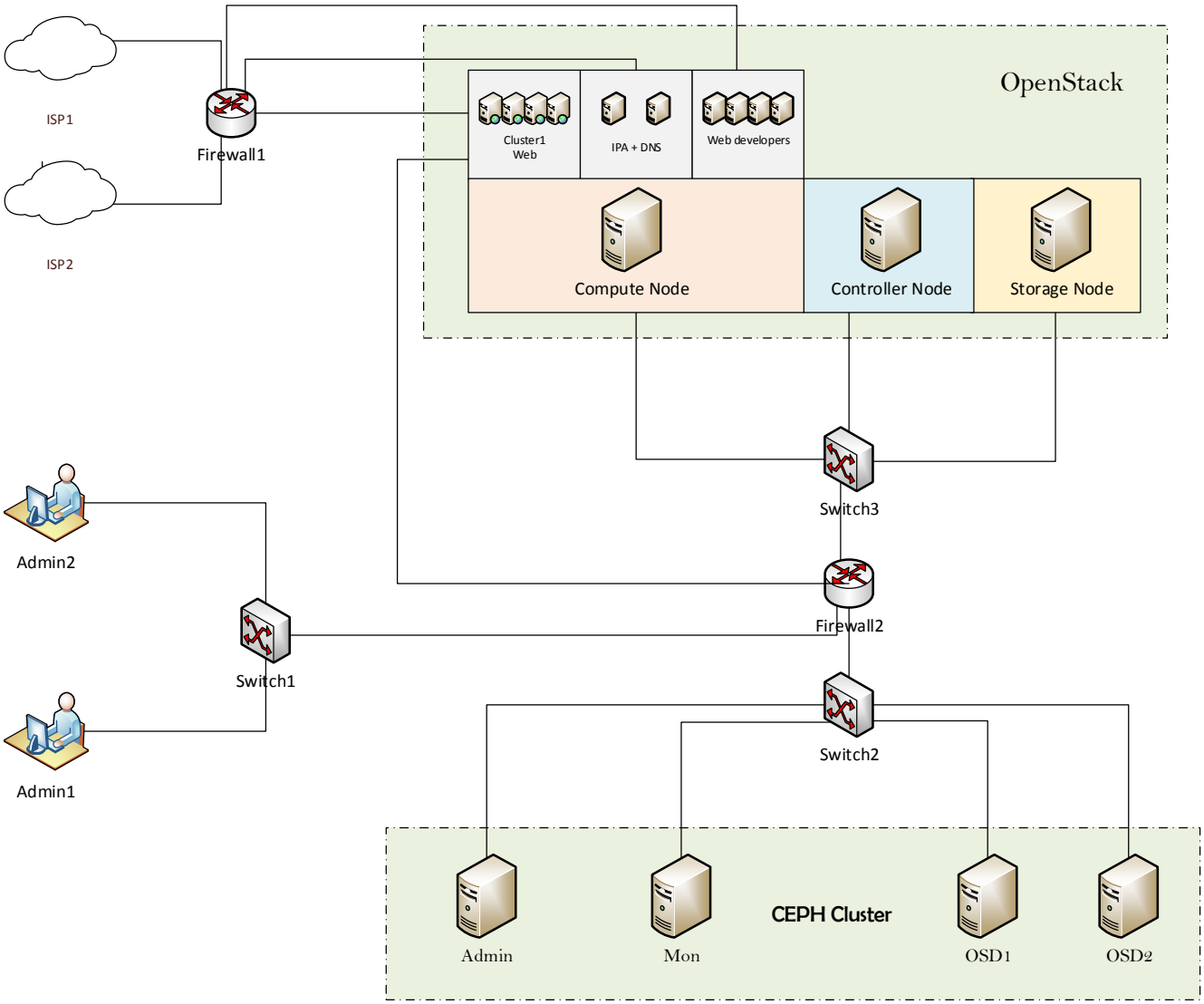


Figure 2.1 Open Source Integrated Infra-Structure Design Using Ansible Configuration Management .

2.3 Steps for Applying project design :

The steps we followed to apply the design is shown in figure 2.2 :

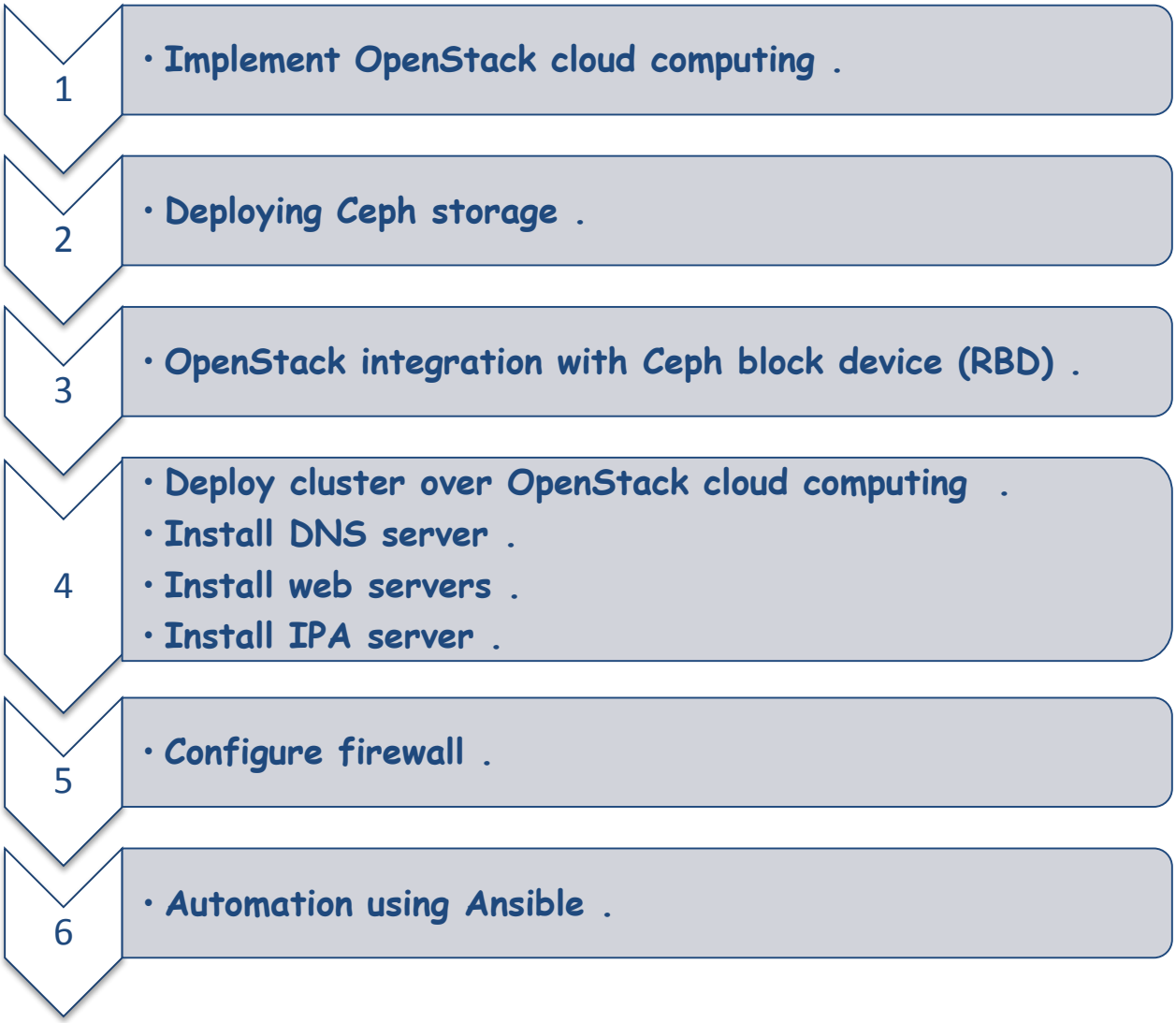


Figure 2.2 Project diagram .

1. First we start to implement OpenStack cloud computing as in chapter 3 .

2. Deploying Ceph storage as chapter 4 .

3. OpenStack integration with Ceph block device (RBD) to use Ceph as OpenStack backend as explained in chapter 5 .

4. Deploy cluster over OpenStack cloud computing as shown in chapter 6 then install the following servers in your clusters :

- DNS 'Domain Name Server ' .
- web servers .
- Identity management with IPA server .

5. As explained in chapter 7 , you should configure firewall .

6. Automate Ceph , Cluster and OpenStack using Ansible configuration management to run them automatically as in chapter 8 .

Chapter 3
OpenStack cloud computing

OpenStack Cloud Computing

In this chapter we will cover :

- Introduction to OpenStack .
- Preparing Basic Environment to start with OpenStack.
- Adding Identity Service .
- Adding Image Service .
- Adding Compute Service .
- Adding Network Service .
- Adding Dashboard Service .
- Adding Block storage Service .
- Launch Instances .

3.1 Introduction :

OpenStack is an open source cloud computing platform , it provides an **Infrastructure-as-a-Service "IaaS"** solution using some services , each service offer an **application programming interface "API"** . The reference of this chapter is [1] . Fig **3.1** provides the conceptual architecture of a typical OpenStack environment :

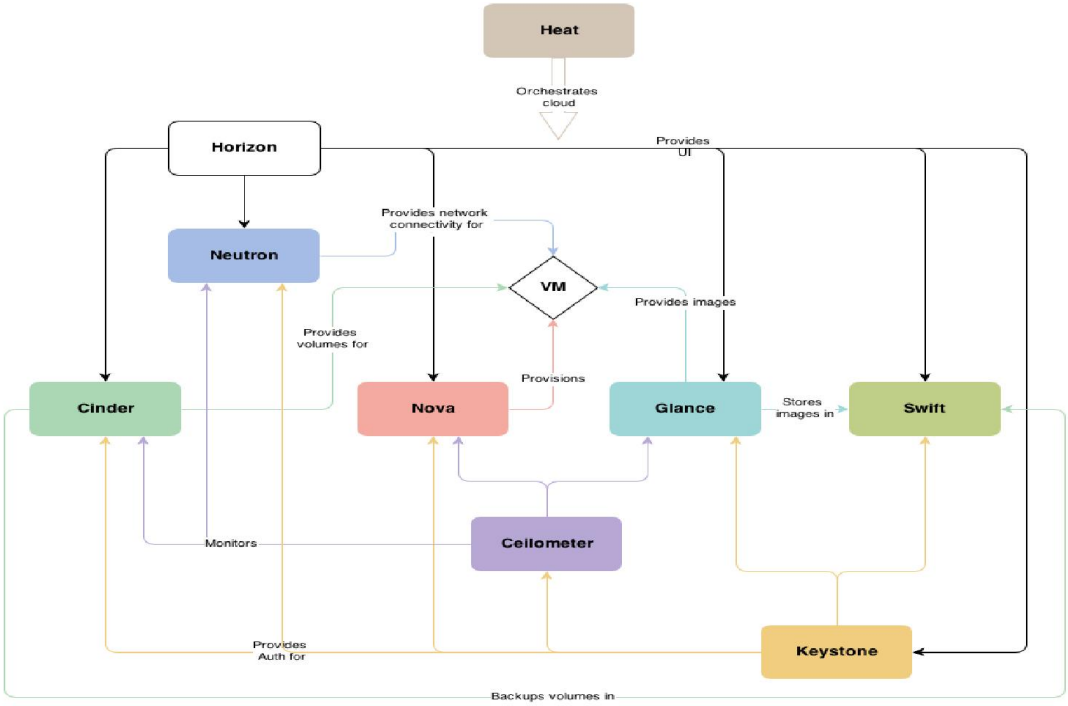


Figure 3.1 : The conceptual architecture of a typical OpenStack environment

As shown in Fig3.1 we have several services , let us list a simple definition for each one :

- **Identity service :**. Provides authentication services and a catalog of endpoints for all OpenStack services .
- **Image service :** Stores and retrieves virtual machine disk images.
- **Block storage service :** Provides block storage to running instances .
- **Dashboard service :** Provides a web-based self-service portal to interact with underlying OpenStack services .
- **Compute service :** Manages the lifecycle of compute instances in an OpenStack environment .
- **Networking service :** Enables Network-Connectivity-as-a-Service for other OpenStack services .

For more about this topic, it can be found in [2] :

We will use 3 nodes on this product :

- **controller node :** Runs the Identity service, Image service, management portions of Compute, management portion of Networking, various Networking agents, and the dashboard. It also includes supporting services such as an SQL database, message queue, and NTP. Requires a minimum of two network interfaces , 1 processor, 4 GB memory, and 5 GB storage .
- **compute node :** Runs the hypervisor portion of Compute that operates instances. Requires a minimum of two network interfaces , 1 processor, 2 GB memory, and 10 GB storage .
- **Storage node :** Contains the disks that the Block Storage and Shared File System services provision for instances . Requires a minimum of one network interface , 1 processor, 1 GB memory, and 10 GB storage .

3.2 Basic environment :

In all nodes :

1. Install your operating system and enable OpenStack repository then update the operating system .

2. Set hostname and IP's for each node to be 'for example' 'controller.project.local , 192.168.1.11/24' for controller node ,

'compute1.project.local , 192.168.1.31/24' for compute node and 'block1.project.local , 192.168.1.41/24' for storage node .

3. Install and configure Chrony, to properly synchronize services among nodes **"Appendix A.1"**.

4. Install the OpenStack client .

```
# yum -y install python-openstackclient
```

5. Install openstack-selinux package to automatically manage security policies for OpenStack services .

```
# yum -y install openstack-selinux
```

In Controller node :

6. Install the database packages :

```
# yum install mariadb mariadb-server  python2-PyMySQL
```

7. Create the /etc/my.cnf/openstack.cnf file and complete the following actions:

```
# vim /etc/my.cnf/openstack.cnf

      [mysqld]

            bind-address = 192.168.1.11

            default-storage-engine = innodb

            innodb_file_per_table

            collation-server = utf8_general_ci

            character-set-server = utf8
```

8. Start and enable the database service :

```
# systemctl enable mariadb.service
# systemctl start mariadb.service
```

9. Secure the database service including choosing a suitable password for the root account **"Appendix A.2"** :

```
# mysql_secure_installation
```

10. Install the MongoDB packages:

```
# yum install mongodb-server mongodb
```

11. Edit the /etc/mongod.conf file and complete the following actions:

```
# vim /etc/mongod.conf

      bind_ip = 192.168.1.11
```

12. Start the MongoDB service and configure it to start when the system boots:

```
# systemctl enable mongod.service
# systemctl start mongod.service
```

13. OpenStack uses a message broker to coordinate operations and status information among services install , start and enable the service , the add messager broker user to be openstack and a suitable password :

```
# yum install rabbitmq-server
# systemctl enable rabbitmq-server
# systemctl start rabbitmq-server
# rabbitmqctl add_user openstack openstack
# rabbitmqctl set_permissions openstack ".*"".*"".*"
```

14. Install the packages:

```
# yum install memcached python-memcached
```

15. Start the Memcached service and configure it to start when the system boots:

```
# systemctl enable memcached.service
# systemctl start memcached.service
```

3.3 Add identity service :

All incoming commands will be in the controller node :

1. Create and grant proper access to the keystone database :

```
# mysql -u root -p
password : ****
      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';
      mysql > quit ;
```

2. Install the identity service packages :

```
# yum -y install  openstack-keystone  httpd  mod_wsgi
```

3. Generate a random value to use as the administration token during initial configuration:

```
# openssl rand -hex 10
```

4. Edit the identity service configuration file /etc/keystone/keystone.conf **"Appendix A.3"** .

5. Populate the Identity service database:

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

6. Initialize Fernet keys:

```
# keystone-manage fernet_setup --keystone-user keystone --
keystone-group keystone
```

7. Configure Apache HTTP server "**Appendix A.4**" then enable and start http service :

```
# systemctl enable httpd.service
# systemctl start httpd.service
```

8. Configure the authentication token which equal to the random value we set in step 3 , then configure endpoint URL and Identity API version as shown :

```
# export OS_TOKEN=294a4c8a8a475f9b9836
# export OS_URL=http://controller:35357/v3
# export OS_IDENTITY_API_VERSION=3
```

9. Create the service entity :

```
# openstack service create --name keystone --description
"OpenStack Identity" identity
```

10. Create the Identity service API endpoints :

```
# openstack endpoint create --region RegionOne identity public
http://controller:5000/v3

# openstack endpoint create --region RegionOne identity internal
http://controller:5000/v3

# openstack endpoint create --region RegionOne identity admin
http://controller:35357/v3
```

11. Create the default domain :

```
# openstack domain create --description "Default Domain" default
```

12. Create the admin project :

```
# openstack project create --domain default --description "Admin
Project" admin
```

13. Create the admin user :

```
# openstack user create --domain default --password-prompt admin
```

14. Create the admin role :

```
# openstack role create admin
```

15. Add the admin role to the admin project and user :

```
# openstack role add --project admin --user admin admin
```

16. Create the service project:

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

17. Create the demo project :

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

18. Create the demo user :

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

19. Create the user role :

```
# openstack role create user
```

20. Add the user role to the demo project and user :

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

21. Unset the temporary OS_TOKEN and OS_URL environment variable:

```
# unset OS_TOKEN OS_URL
```

22. Create client environment scripts to the admin and demo projects and users :

```
# vim /root/admin-openrc
      export OS_PROJECT_DOMAIN_NAME=default
      export OS_USER_DOMAIN_NAME=default
      export OS_PROJECT_NAME=admin
      export OS_USERNAME=admin
      export OS_PASSWORD=admin
      export OS_AUTH_URL=http://controller:35357/v3
      export OS_IDENTITY_API_VERSION=3
      export OS_IMAGE_API_VERSION=2
# vim /root/demo-openrc
      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
      export OS_AUTH_URL=http://controller:5000/v3
      export OS_IDENTITY_API_VERSION=3
```

```
export OS_IMAGE_API_VERSION=2
```

23. For security reasons, 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.

3.4 Add image service :

All incoming commands will be in the controller node :

1. Create and grant proper access to the glance database :

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

2. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

3. Create the glance user , then add the admin role to the glance user and service project :

```
# openstack user create --domain default --password-prompt
glance

# openstack role add --project service --user glance admin
```

4. Create the glance service entity :

```
# openstack service create --name glance --description
"OpenStack Image" image
```

5. Create the Image 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
```

6. Install the image service packages :

```
# yum install openstack-glance
```

7. Edit the /etc/glance/glance-api.conf configuration file **"Appendix A.5" .**

8. Edit the /etc/glance/glance-registry.conf configuration file **"Appendix A.6"**

9. Populate the Image Service database :

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

10. Start and enable the Image Service services:

```
# systemctl enable openstack-glance-api.service openstack-
glance-registry.service
# systemctl start openstack-glance-api.service openstack-glance-
registry.service
```

11. 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
```

12. Confirm upload of the image and validate attributes :

```
# openstack image list
```

3.5 Add compute service :

All incoming commands will be in the controller node :

1. Create and grant proper access to the nova and nova_api database :

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

2. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

3. Create the nova , then add the admin role to the nova user:

```
# openstack user create --domain default --password-prompt nova

# openstack role add --project service --user nova admin
```

4. Create the nova service entity :

```
# openstack service create --name nova --description "OpenStack
Compute" compute
```

5. Create the compute 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
```

6. Install the compute service packages :

```
# yum -y install openstack-nova-api openstack-nova-conductor
openstack-nova-console openstack-nova-novncproxy openstack-nova-
scheduler
```

7. Edit the /etc/nova/nova.conf configuration file **"Appendix A.7"** .

8. Populate the Compute Service database :

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

9. Start and enable the Compute Service services:

```
# systemctl enable openstack-nova-api.service openstack-nova-
consoleauth.service openstack-nova-scheduler.service openstack-
nova-conductor.service openstack-nova-novncproxy.service

# systemctl start openstack-nova-api.service openstack-nova-
consoleauth.service openstack-nova-scheduler.service openstack-
nova-conductor.service openstack-nova-novncproxy.service
```

All incoming commands will be in the compute node :

10. Install the compute service packages :

```
# yum -y install openstack-nova-compute
```

11. Edit the /etc/nova/nova.conf configuration file **"Appendix A.8" .**

12. Start and enable the Compute Service services:

```
# systemctl enable libvirtd.service openstack-nova-
compute.service
# systemctl start libvirtd.service openstack-nova-
compute.service
```

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

```
# openstack compute service list
```

3.6 Add network service :

All incoming commands will be in the controller node .

1. Create and grant proper access to the neutron database :

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

2. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

3. Create the neutron user , then add the admin role to the neutron user :

```
# openstack user create --domain default --password-prompt
neutron
# openstack role add --project service --user neutron admin
```

4. Create neutron service entity

```
# openstack service create --name neutron --description
"OpenStack Networking" network
```

5. Create the Networking Service API endpoints :

```
# 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
```

6. Install the network service packages :

```
# yum install openstack-neutron openstack-neutron-ml2 openstack-
neutron-linuxbridge ebtables
```

7. Edit the /etc/neutron/neutron.conf configuration file **"Appendix A.9" .**

8. Edit the /etc/neutron/plugins/ml2/ml2_conf.ini Modular Layer 2 (ML2) plug-in configuration file **"Appendix A.10" .**

9. Edit the /etc/neutron/plugins/ml2/linuxbridge_agent.ini Linux bridge agent configuration file **"Appendix A.11" .**

10. Edit the /etc/neutron/l3_agent.ini Layer3 agent configuration file **"Appendix A.12" .**

11. Edit the /etc/neutron/dhcp_agent.ini dhcp agent configuration file **"Appendix A.13" .**

12. Edit the /etc/neutron/metadata_agent.ini metadata agent configuration file **"Appendix A.14" .**

13. Edit the /etc/nova/nova.conf to configure compute node to use Networking **"Appendix A.15" .**

14. Create a symbolic link /etc/neutron/plugin.inipointing to the ML2 plug-in configuration file, /etc/neutron/plugins/ml2/ml2_conf.ini

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini
/etc/neutron/plugin.ini
```

15. Populate the Networking Service 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
```

16. Restart the compute API services , start and enable the Network Service services:

```
# systemctl restart openstack-nova-api.service

# systemctl enable neutron-server.service neutron-linuxbridge-
agent.service neutron-dhcp-agent.service neutron-metadata-
agent.service neutron-l3-agent.service

# systemctl start neutron-server.service neutron-linuxbridge-
agent.service neutron-dhcp-agent.service neutron-metadata-
agent.service neutron-l3-agent.service
```

All incoming commands will be in the comoute node .

17. Install the network service packages :

```
# yum install openstack-neutron-linuxbridge ebtables ipset
```

18. Edit the /etc/neutron/neutron.conf configuration file **"Appendix A.16" .**

19. Edit the /etc/neutron/plugins/ml2/linuxbridge_agent.ini Linux bridge agent configuration file **"Appendix A.17" .**

20. Edit the /etc/nova/nova.conf compute configuration file to enable networking services **"Appendix A.18" .**

21. Restart the compute service , then enable and restart Linux bridge agent :

```
# systemctl restart openstack-nova-compute.service

# systemctl enable neutron-linuxbridge-agent.service

# systemctl start neutron-linuxbridge-agent.service
```

22. List loaded extensions to verify successful launch of the neutron-server process :

```
# neutron ext-list

# neutron agent-list
```

3.7 Add dashboard service :

All incoming commands will be in the controller node .

1. Install the dashboard service packages :

```
# yum install openstack-dashboard
```

2. Edit the /etc/openstack-dashboard/local_settings configuration file **"Appendix A.19" .**

3. Restart the web server and session storage service: :

```
# systemctl restart httpd.service memcached.service
```

4. Access the dashboard using a web browser at http://controller/dashboard .

3.8 Add block storage service :

All incoming commands will be in the controller node :

1. Create and grant proper access to the cinder database :

```
# mysql -u root -p
password : ****
      mysql > CREATE DATABASE cinder;
      mysql > GRANT ALL PRIVILEGES ON cinder.* TO
```

```
'cinder'@'localhost' IDENTIFIED BY 'cinder';
      mysql > GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%'
IDENTIFIED BY 'cinder';
      mysql > quit ;
```

2. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

3. Create the cinder user , then add the admin role to the cinder user :

```
# openstack user create --domain default --password-prompt
cinder

# openstack role add --project service --user cinder admin
```

4. Create the cinder and cinderv2 service entities :

```
# openstack service create --name cinder --description
"OpenStack Block Storage" volume

# openstack service create --name cinderv2 --description
"OpenStack Block Storage" volumev2
```

5. Create the Block storage Service API endpoints :

```
# openstack endpoint create --region RegionOne volume public
http://controller:8776/v1/%\(tenant_id\)s

# openstack endpoint create --region RegionOne volume internal
http://controller:8776/v1/%\(tenant_id\)s

# openstack endpoint create --region RegionOne volume admin
http://controller:8776/v1/%\(tenant_id\)s

# openstack endpoint create --region RegionOne volumev2 public
http://controller:8776/v2/%\(tenant_id\)s
```

```
# openstack endpoint create --region RegionOne volumev2 internal
http://controller:8776/v2/%\(tenant_id\)s

# openstack endpoint create --region RegionOne volumev2 admin
http://controller:8776/v2/%\(tenant_id\)s
```

6. Install the Block storage service packages :

```
# yum -y install openstack-cinder
```

7. Edit the /etc/cinder/cinder.conf configuration file **"Appendix A.20" .**

8. Populate the Block Storage database :

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

9. Configure compute to use block storage by editing /etc/nova/nova/conf configuration file **"Appendix A.21" .**

10. Restart the Compute API service , then start the Block Storage services and configure them to start when the system boots :

```
# systemctl restart openstack-nova-api.service

# systemctl enable openstack-cinder-api.service openstack-
cinder-scheduler.service

# systemctl start openstack-cinder-api.service openstack-cinder-
scheduler.service
```

All incoming commands will be in the storage node .

11. Install LVM packages , then start and enable it :

```
# yum install lvm2

# systemctl enable lvm2-lvmetad.service

# systemctl start lvm2-lvmetad.service
```

12. Create the LVM physical volume and volume group /dev/sdb :

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

13. Edit the /etc/lvm/lvm.conf file with the following actions :

```
# vim /etc/lvm/lvm.conf
      devices {
            filter = [ "a/sdb/", "r/.*/"]
```

14. Install block storage packages :

```
# yum -y install openstack-cinder targetcli
```

15. Edit /etc/cinder/cinder.conf configuration file **"Appendix A.22"** .

16. Start and enable the Block storage Service services:

```
# systemctl enable openstack-cinder-volume.service
target.service
# systemctl start openstack-cinder-volume.service target.service
```

17. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

18. List service components to verify successful launch of each process :

```
# cinder service-list
```

3.9 Launch instance :

All incoming commands will be in the controller node .

1. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

2. Create provider network :

```
# neutron net-create --shared --provider:physical_network
provider --provider:network_type flat provider
```

3. Create subnet on the provider network :

```
# neutron subnet-create --name provider  --allocation-pool
start=203.0.113.101,end=203.0.113.250  --dns-nameserver 8.8.4.4
--gateway 203.0.113.1  provider 203.0.113.0/24
```

4. Source the demo credentials to gain access to user-only CLI commands :

```
# . ~/demo-openrc
```

5. Create the self service network :

```
# neutron net-create selfservice
```

6. Create subnet on the self service network :

```
neutron subnet-create --name selfservice  --dns-nameserver
8.8.4.4 --gateway 172.16.1.1  selfservice 172.16.1.0/24
```

7. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

8. Add the router: external option to the provider network :

```
# neutron net-update provider --router:external
```

9. Source the demo credentials to gain access to user-only CLI commands :

```
# . ~/demo-openrc
```

10. Create the router :

```
# neutron router-create router
```

11. Add the self-service network subnet as an interface on the router:

```
# neutron router-interface-add router selfservice
```

12. Set a gateway on the provider network on the router:

```
# neutron router-gateway-set router provider
```

13. Source the admin credentials to gain access to admin-only CLI commands :

```
# . ~/admin-openrc
```

14. List network namespaces :

```
# ip netns
```

15. List ports on the router to determine the gateway IP address on the provider network :

```
# neutron router-port-list router
```

16. Ping this IP address from the controller node or any host on the physical provider network :

```
# ping -c 4 203.0.113.102
```

17. Create m1.nano flavor :

```
# openstack flavor create --id 0 --vcpus 1 --ram 64 --disk 1
m1.nano
```

18. Source the demo credentials to gain access to user-only CLI commands :

```
# . ~/demo-openrc
```

19. Generate and add a key pair :

```
# ssh-keygen -q -N ""

# openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
```

20. Verify addition of the key pair :

```
# openstack keypair list
```

21. Add rules to the default security group : (Permit *ICMP* (ping) and Permit secure shell (SSH) access) :

```
# openstack security group rule create --proto icmp default

# openstack security group rule create --proto tcp --dst-port 22
default
```

22. List available flavors :

```
# openstack flavor list
```

23. List available images :

```
# openstack image list
```

24. List available networks :

```
# openstack network list
```

25. List available securitygroups :

```
# openstack security group list
```

26. Launch the instance :

```
openstack server create --flavor m1.tiny --image cirros  --nic
net-id=SELFSERVICE_NET_ID --security-group default --key-name
mykey selfservice-instance
```

27. Check the status of your instance :

```
# openstack server list
```

28 . Create a *floating IP address* on the provider virtual network :

```
# openstack ip floating create provider
```

29 . Associate the floating IP address with the instance :

```
# openstack ip floating add 203.0.113.104 selfservice-instance
```

30 . Check the status of your floating IP address :

```
# openstack server list
```

31 . Verify connectivity to the instance via floating IP address from the controller node or any host on the provider physical network :

```
# ping -c 4 203.0.113.104
```

32 . Access your instance using SSH from the controller node or any host on the provider physical network :

```
# ssh cirros@203.0.113.104
```

33. Create a 10 GB volume :

```
$ openstack volume create --size 10 volume1
```

34. Attach Block Storage volume to your instance

```
openstack server add volume provider-instance volume1
```

Note : If you have unclear points or errors in this chapter you can check [3] .

Chapter 4
Ceph Storage

Ceph Storage

In this chapter we will cover :

- Introduction .
- Ceph components .
- Ceph deployment .
- Deploy Ceph cluster .
- Verifying cluster health .

4.1 Introduction :

Ceph is open source project , distribute object storage designed to provide high performance, reliability and scalability .

Ceph can be used as block storage, file storage or object storage as my system require, the reference of this chapter is [4] and [5] .

4.2 Ceph components :

1. **Admin nodes :** used to manage and deploy all clusters .
2. **Monitor nodes :** used to monitor all clusters and it is have a map to all of them .
3. **OSD nodes :** used to store data .

when client wants to store or access data, first he must connect to Mon node to get cluster map . use this map to identify where he put data or access data . **Figure 4.1 Ceph simple over view of Ceph** . For more explanation you can also check [6] and [7] .

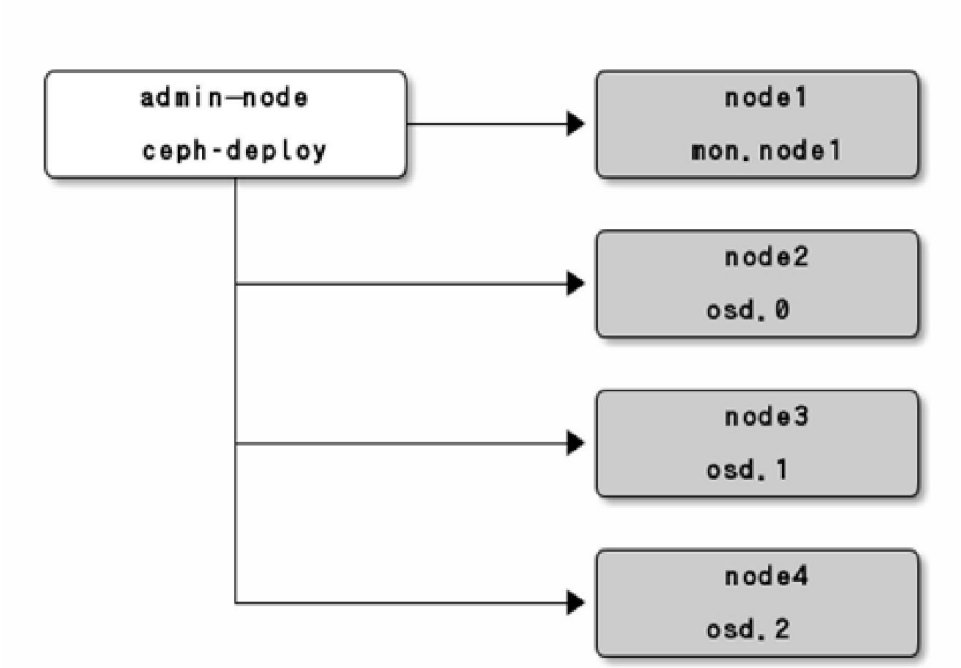


Figure 4.1 Simple overview of Ceph sorage

4.3 Ceph deployment :

4.3.1 Preparing nodes :

a. Admin node :

1. Disable or permissive selinux , then configure your network as shown :

```
# setenforce 0

# nmcli con mod eno3355046 ipv4.method manual ipv4.address
"192.168.1.5/24"

# nmcli con reload eno3355046
```

2. Install Ceph-deploy packages which is a tool used to deploy and manage cluster service in all cluster nodes .

```
# yum install Ceph-deploy
```

b. Monitor node :

1. Disable or permissive selinux , then configure your network .

```
# setenforce 0

# nmcli con mod eno3355046 ipv4.method manual ipv4.address
"192.168.1.6 /24"

# nmcli con reload eno3355046
```

c. OSD node :

1. Disable or permissive selinux , then configure your network .

```
# setenforce 0

# nmcli con mod eno3355046 ipv4.method manual ipv4.address
"192.168.1.7 /24"

# nmcli con reload eno3355046
```

4.3.2 Deploy Ceph cluster :

There are 2 ways to deploy CEPH : first way is installing packages through red hat subscription 'see installation documentation' and second way is downloading red hat CEPH ISO image which contains all packages you need . In our project will use the second way to deploy CEPH as incoming :

All incoming steps will be in the admin node .

1. Download RHCEPH ISO image from redhat.com .

2. Mount CEPH ISO to /mnt :

```
# mount /rhCeph.iso   /mnt
```

3. Install ice_setup package , which is used to install Ceph-deploy package that is used to deploy services to other nodes .

```
# yum install /mnt/installer/ice_setup
```

4. Use ice_setup to install Ceph-deploy .

```
# ice_setup   -d    /mnt
```

5. Make all other nodes as a client to admin repos .

```
# Ceph-deploy install --repo --release=osd osd1

# Ceph-deploy install --repo --release=osd osd2

# Ceph-deploy install --repo --release=mon mon
```

6. Create directory for configuration files .

```
# mkdir ~/Ceph-config

# cd ~/Ceph-config
```

7. Create cluster :

```
# Ceph-deploy new mon
```

8. Edit configuration file .

```
# vim /etc/Ceph/Ceph.conf
      cluster-network = 192.168.1.0/24
      public-network = 192.168.2.0/24
```

9. Deploy mon node .

```
# Ceph-deploy install --mon mon-node
```

10. Deploy osd node .

```
# Ceph-deploy install --osd osd0 osd1 osd2
```

11. Configure osd disks .

```
# Ceph-deploy osd prepare osd0:sdb
# Ceph-deploy osd prepare osd1:sdb
# Ceph-deploy osd prepare osd2:sdb
```

12. Verify cluster health at mon node .

```
# Ceph -s
             health ok
```

Chapter 5 OpenStack Integration With Ceph Block Device (RBD)

OpenStack integration With Ceph Block Device (RBD)

In this chapter we will cover :

- Introduction .
- The parts of OpenStack that integrates with Ceph .
- Configure OpenStack Ceph clients .
- Install Ceph client packages .
- Setup Ceph client authentication .
- Configure image node to use Ceph .
- Configure cinder node to use Ceph .
- Configure compute node to use Ceph.

5.1 Introduction :

This chapter aims at implementing Rados Block Device (RBD) as the storage backend for OpenStack block storage service, more about this topic can be found in [8] and [9] .

To use Ceph Block Devices with OpenStack , You must have access to a running Ceph storage cluster as we explain before in **chapter 4** . So make sure you have configured a Ceph storage Cluster and is in 'active + clean' state as in **figure 5.1** shown below .

```
cephuser@ceph-deploy:~$ ceph -s
    cluster a102c642-563f-4f37-80ec-88ed7bc6cffa
     health HEALTH_WARN
            too many PGs per OSD (320 > max 300)
     monmap e1: 1 mons at {cephnode1=192.168.1.195:6789/0}
            election epoch 1, quorum 0 cephnode1
     osdmap e83: 2 osds: 2 up, 2 in
      pgmap v1966: 320 pgs, 4 pools, 0 bytes data, 3 objects
            12923 MB used, 850 GB / 908 GB avail
                 320 active+clean
```

Figure 5.1 Ceph state

The parts of OpenStack that integrates with CEPH are :

- **Images :**

In OpenStack , the Image service manages images for VM instances. The VM uses the bootable images from glance to boot for the first time. Integrating this image into CEPH means, the VM images that Glance Service creates is stored in the RADOS cluster rather than storing it in the same physical node where the glance service runs.

- **Volumes :**

The Block Storage Service of OpenStack(cinder) uses cinder-volume service to manage Block devices. All the required storage space for a VM instance is provided as block storage. OpenStack uses block devices to either provide bootable volumes to VM's or to attach volumes to running VMs. The storage space is taken from the same node where the cinder-volume service runs. In my OpenStack configuration, a separate node is used for serving Block devices. Integrating volumes with CEPH means that, the volumes created with OpenStack is actually provided by the nodes participating in the RADOS cluster. The entire Block Device is stripped and stored as objects in CEPH, but virtually we get a block device as such to use with VM's.

- **VM's :**

By default when you boot a VM in OpenStack, the disk appears as a disk on the node that runs the hypervisor portion of compute service . In our case its the compute node that holds the files for the operating system of the VM's. Integrating this with CEPH makes the files for the operating system of a VM to be stored on the RADOS cluster in a separate pool. This allows us to boot every VM inside CEPH directly without using OpenStack Cinder Service, which is advantageous because it allows us to perform maintenance operations easily with the live-migration process.

Note : Ceph doesn't support QCOW2 for hosting a virtual machine disk. Thus if you want to boot virtual machines in CEPH, the Glance image format must be RAW.

5.2 Configure OpenStack ceph clients :

All incoming commands will be in admin node .

1. Create three pools named VM, images and volumes to store VM data, glance images and cinder volumes respectively in CEPH cluster using the following commands :

```
# Ceph osd pool create volumes 128

# Ceph osd pool create images 128

# Ceph osd pool create vms 128
```

2. Make sure that the pools are created using the following command :

```
# Ceph osd lspools
```

3. Copy /etc/Ceph/Ceph.conf file to the nodes running glance-api , cinder-volume and nova-compute as shown below :

```
# ssh root@controller sudo tee /etc/Ceph/Ceph.conf
</etc/Ceph/Ceph.conf

# ssh root@compute sudo tee /etc/Ceph/Ceph.conf
</etc/Ceph/Ceph.conf

# ssh root@cinder sudo tee /etc/Ceph/Ceph.conf
</etc/Ceph/Ceph.conf
```

5.3 Install ceph client packages :

1. On the node that runs glance-api service **'controller node'** install Python-rbd package :

```
# yum -y  install python-rbd
```

2. On the nova-compute **'compute node'** and cinder-volume node **'cinder node'** , install Python-rbd and Ceph-common package as shown :

```
# yum -y install Ceph-common python-rbd
```

5.4 Setup Ceph client Authentication :

All incoming commands will be in admin node .

1. Create a new user for Cinder and Glance :

```
# Ceph auth get-or-create client.cinder mon 'allow r' osd 'allow
class-read object_prefix rbd_children, allow rwx pool=volumes,
allow rwx pool=vms, allow rx pool=images'

# Ceph auth get-or-create client.glance mon 'allow r' osd 'allow
class-read object_prefix rbd_children, allow rwx pool=images'
```

2. Add the keyrings for client.cinder and client.glance to the respective nodes and change their ownership as shown below :

```
# Ceph auth get-or-create client.glance | ssh root@controller
sudo tee /etc/Ceph/Ceph.client.glance.keyring

# ssh root@controller sudo chown glance:glance
/etc/Ceph/Ceph.client.glance.keyring

# Ceph auth get-or-create client.cinder | ssh root@cinder sudo
tee /etc/Ceph/Ceph.client.cinder.keyring

# ssh root@cinder sudo chown cinder:cinder
/etc/Ceph/Ceph.client.cinder.keyring
```

3. Add the key ring file as shown :

```
# Ceph auth get-or-create client.cinder | ssh root@compute sudo
tee /etc/Ceph/Ceph.client.cinder.keyring
```

4. Add the secret key to libvirt and remove the temporary copy of the key on the **'compute node'** :

Note : The output will be a key similar to **457eb676-33da-42ec-9a8c-9293d545c337**

```
# uuidgen

# cat > secret.xml <<EOF

            <secret ephemeral='no' private='no'>

            <uuid>457eb676-33da-42ec-9a8c-9293d545c337</uuid>

            <usage type='Ceph'>

            <name>client.cinder secret</name>

            </usage>

            </secret>

            EOF

#  virsh secret-define --file secret.xml

Secret 457eb676-33da-42ec-9a8c-9293d545c337 created

# virsh secret-set-value --secret 457eb676-33da-42ec-9a8c-
9293d545c337 --base64 $(cat client.cinder.key) && rm
client.cinder.key secret.xml
```

5.5 Configure OpenStack nodes to use ceph :

5.5.1 Configuring Glance :

1. Edit /etc/glance/glance-api.conf glance configuration file on the '**controller node**' **"Appendix B.1"**.

2. Restart glance-api service .

```
# service glance-api restart
```

5.5.2 Configuring Cinder :

1. On the cinder node, edit /etc/cinder/cinder.conf **"Appendix B.2"**.

2. Restart cinder-volume service .

```
# sudo service cinder-volume restart
```

5.5.3 Configuring Nova :

1. On the compute node, edit the Ceph configuration file /etc/Ceph/Ceph.conf **"Appendix B.3"**.

2. Create the necessary directories for the socket file and the log files:

```
# mkdir -p /var/run/Ceph/guests/ /var/log/qemu/
```

3. Edit the /etc/nova/nova.conf file **"Appendix B.4"** .

4. Restart nova-compute service .

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

Now the OpenStack is ready with Storage Backend as CEPH. All the volumes created are stored as objects in the Volumes pool, glance images created are stored as objects in the images pool and the VM data is stored as objects in the vms pool.

Chapter 6
Red Hat Clustering

Red Hat Clustering

In this chapter we will cover :

- Introduction .
- Preparing nodes to deploy cluster .
- Installing Cluster Software .
- Adding cluster resources .
- Add cluster IP resources .
- Add Apache resources .

6.1 Introduction :

Computer cluster can be used to provide high-availability services and resources by redundant multiple machines , this redundancy is used to guard against failure of many types, Figure 6.1 shows Overview of Cluster structure . The reference of this chapter is [10] .

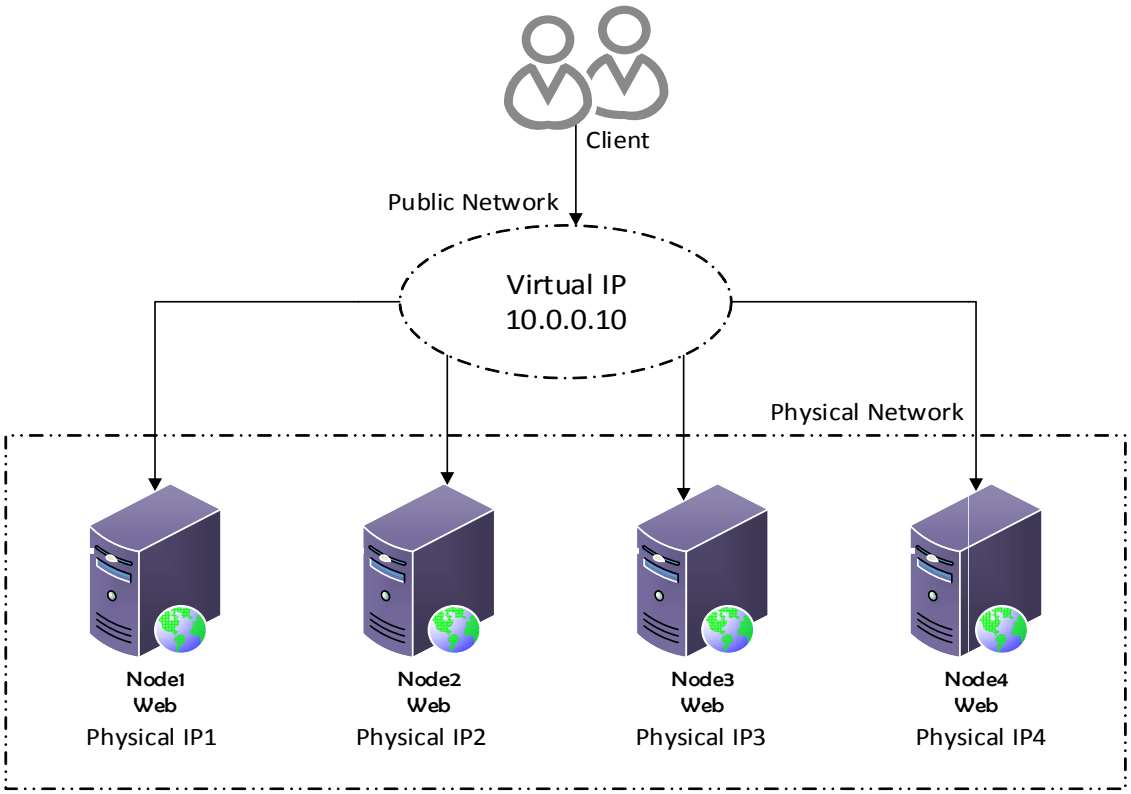


Figure 6.1 : Overview of Cluster structure

Pacemaker uses **CIB** " cluster information base " which is **xml** file contains information and configuration of cluster synced to entire cluster , used by **PE** engine "policy engine" to compute and achieve ideal state , this instructions fed to **DC** "designated controller" .

Decision making done by master node that chosen by election between cluster nodes . DC carry out PE's instruction and passing them to **LRMD** "local resource management daemon" and **CRMD** "cluster resource management daemon" .

The next step is using instances we launched before in chapter 3 OpenStack cloud computing to make a cluster of them as shown in the next sections, you can check more in [11] .

6.2 Prepare nodes to deploy cluster :

In this product we will use centos7 distribution , to be able to start we need to prepare operating system and repository to deploy cluster .

1. Configure NIC with a suitable static IP as shown :

```
# nmcli con mod con-name eno16777736 ipv4.address
"192.168.122.x/24" ipv4.dns "8.8.8.8" ipv4.method manual
# nmcli con down eno16777736
# nmcli con up eno16777736
```

2. configure nodes hostname :

```
# hostnamectl set-hostname nodex.project.local
```

3. configure firewall :

```
# firewall-cmd --add-service=high-availability --permanent
```

6.3 Installing Cluster Software :

1. Install packages ,then start and enable pcs daemon on all nodes :

```
# yum -y install pcs pacemaker psmisc policycore utils-python
```

```
# systemctl enable pcsd

# systemctl start pcsd
```

2. Configure cluster user , as cluster services need user that used to sync corosync configuration :

```
# echo "project" | passwd --stdin hacluster
```

3. Test if hacluster user can authenticate to all nodes :

```
# pcs cluster setup auth node1 node2
```

4. Now setup cluster , generate and sync corosync configuration then start cluster and verify performance :

```
# pcs cluster setup --name mtcluster nodea nodeb

# pcs cluster start --all

# pcs status
```

Note : Repeat all previous commands to all nodes in cluster .

6.4 Adding cluster resources :

6.4.1 Add cluster IP resources :

1. Add virtual IP that will be shared across cluster and all clients request will be forward to it .

```
# pcs resource create ClusterIP ocf:heartbeat:IPaddr2
ip=192.168.122.x cidr_netmask=32 op monitor interval=30s
```

2. Verify the last command :

```
# pcs status | grep -i '^ClusterIP'

ClusterIP (ocf:heartbear:IPaddr2)      started:nodea
```

Quorum is used to prevent resources from starting on more nodes than desired .

Condition : total nodes <2 * active nodes .

6.4.2 Add Apache resources :

1. Install apache on all nodes , then configure firewall :

```
# yum install httpd
# firewall-cmd --add-service=httpd --permanent
```

2. Configure cluster to work with apache , then verify that resource successfully added .

```
# pcs resource create WebSite ocf:heartbeat:apache
# vim /etc/httpd/conf/httpd.conf
            op monitor interval = 1min
# pcs status
```

3. Ensuring that all resources will run on the same host , pacemaker by default spread resources across cluster nodes to reduce load , but we want all resources to run on the same host by telling cluster that resources are related .

```
# pcs constraint location add Website with ClusterIP
# pcs constraint
```

4. Make resources start and stop in specific order , we need cluster to start IP and apache service , then verify operation.

```
# pcs constraint order ClusterIP then WebSite
# pcs status
```

Chapter 7
Software firewall

Software firewall

In this chapter we will cover :

- Introduction .
- Software vs. Hardware Firewalls.
- Netfilter subsystem .
- Iptables .
- Difference between source NAT and masquerading .
- Firewall configuration .

7.1 Introduction :

Firewall is a network security provides secure connectivity between networks (internal/external). Firewalls is one of the core components of a network security implementation. It is may be a **hardware**, **software**, or a **combination** of both that is used to prevent hackers, spam, spyware and viruses to reach a computer or (part of a) network, Figure 7.1 show simple over view for software firewall . The reference of this chapter is [12] and [13] .

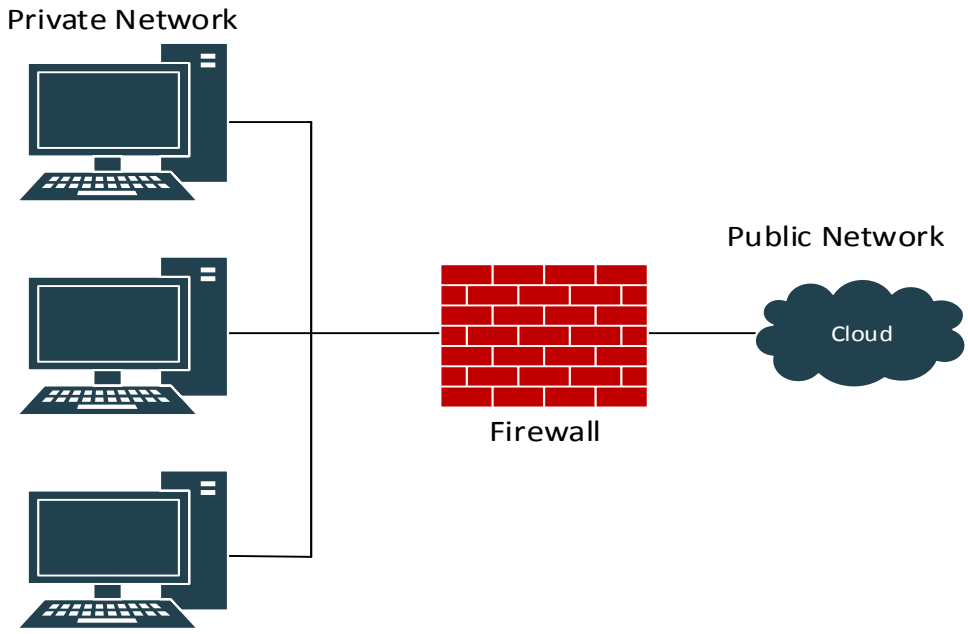


Figure 7.1 : Overview of software firewall .

7.2 Software vs. Hardware Firewalls :

- **Hardware firewall solutions :**

  - Protect an entire network.

  - Usually more expensive, harder to configure

  such as : firewall appliances by Cisco, Nokia, and Sonicwall.

- **software firewall solutions :**

  - Protect a single computer .

  - Usually less expensive, easier to configure .

  such as : home and business markets.

7.3 Netfilters :

The Linux kernel features a powerful networking subsystem called **Netfilter**. The **Netfilter** subsystem provides stateful or stateless packet filtering as well as NAT and IP masquerading services. **Netfilter** has the ability to mangle IP header information for advanced routing and connection state management. Netfilter is controlled using the **Iptables** tool.

7.4 Iptables :

Iptables is a command-line utility for configuring Linux kernel firewall to allow or block traffic . Firewall is called Iptables as it is set of tables . Tables consist of chains, which are lists of rules . Let us define table chains as shown :

- **Input chain** : This chain is used to control the behavior for incoming connections , the packet coming from internet to local server , shown in figure 7.1 Iptables chains explain .

- **output chain** : This chain is used to control the behavior for out coming connections , the packet coming from local server to internet , shown in figure 7.1 Iptables chains explain .

- **Forward chain** : Packet for another NIC on the local server. For packets routed through the local server , packet through internet

and locally machine , shown in figure 7.2 Iptables Input, Output and forward chains overview .

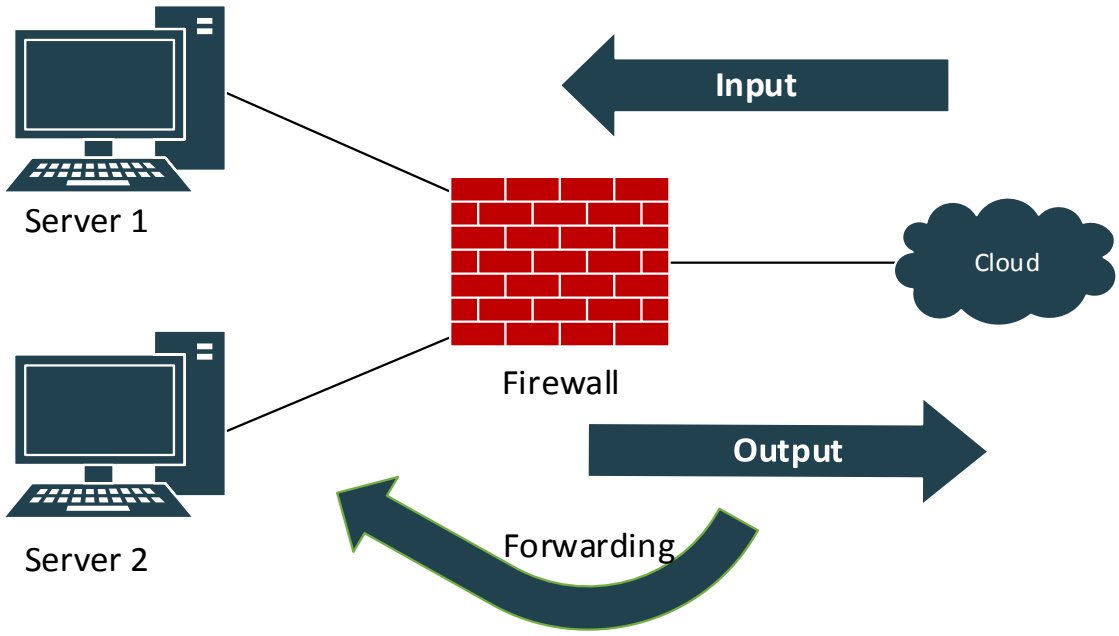


Figure 7.2 : Iptables Input, output and forward chains overview .

- **PRE-ROUTING chain:** Alters packets before routing. This helps to translate the destination IP address of the packets to something that matches the routing on the local server. This is used for DNAT (destination NAT) , shown in figure 7.3 Iptables Pre-routing and Post-routing chains overview .

- **POST-ROUTING chain:** Alters packets after routing. This helps to translate the source IP address of the packets to something that might match the routing on the destination server. This is used for SNAT (source NAT) , shown in figure 7.3 Iptables Pre-routing and Post-routing chains overview .

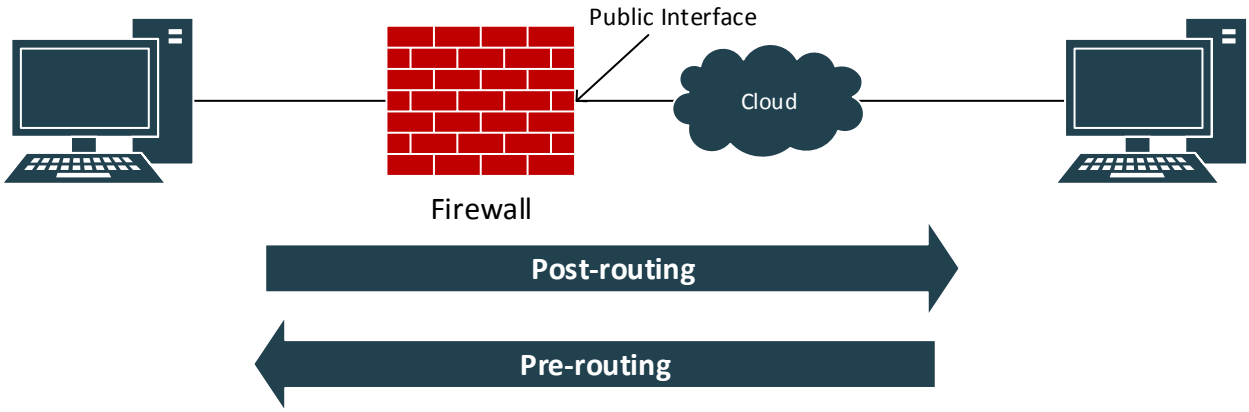


Figure 7.3 : Iptables Pre-routing and Post-routing chains overview .

7.5 Difference between source NAT and masquerading:

Masquerading is a special form of source NAT as it allows to dynamic IP range(more than one IP) to access unlocal devices , but source NAT allows one IP only. Following are the possible special values that you can specify in the target.

- **ACCEPT** : Firewall will accept the packet .
- **DROP** : Firewall will drop the packet without ICMP .
- **REJECT** : Firewall will reject the packet with ICMP .

7.6 Firewall configuration :

1. Edit /etc/sysctl.conf to enable forward packets as shown :

```
# vim /etc/sysctl.conf

            net.ipv4.ip-forward = 1
```

2. Create and edit iptablesscript.sh in a suitable place as shown:

```
# vim iptablesscript.sh

#!/bin/bash
```

```
#iptables script

#Flushing firewall rules to prevent dupplicatting rules

  iptables -F

  iptables -t nat -F

#Default policy reject

  iptables -p INPUT DROP

  iptables -p OUTPUT DROP

  iptables -p FORWARD DROP

#Accept packets if connection was related or established

  iptables -A OUTPUT -m state --state ESTABLISHED RELATED -j
ACCEPT

#Drop invalid packets to Increase processor effeciency

  iptables -A INPUT -m state --state INVALID -j DROP

#Allow ssh connection

  iptables -A INPUT --p tcp --dport 22 -j ACCEPT

#Rule that makes unlocal users see web cluster & DNS

#For web cluster

  iptables -t nat -A PREROUTING -p tcp -d 192.168.0.1 -m
multiport --dport 80.443 -j DNAT \ --to-destination
192.168.2.0/24

#Allow access to web server as default policy of forward chain
is drop

  iptables -A FORWARD -p tcp -d 192.168.2.0/24 -m multiport --
dport 80,443 -j ACCEPT

#For DNS
```

```
  iptables -t nat -A PREROTING -d 192.168.0.1 DNAT \ --to-
destination 192.168.3.2

#Web cluster see ceph

  iptables -A FORWARD -s 192.168.2.0/24 -d 192.168.1.51 -j
ACCEPT

  iptables -A FORWARD -s 192.168.2.0/24 -d 192.168.1.52 -j
ACCEPT

  iptables -A FORWARD -s 192.168.2.0/24 -d 192.168.1.53 -j
ACCEPT

#192.168.4.0/24 see ceph

  iptables -A FORWARD -s 192.168.4.0/24 -d 192.168.1.0/24 -j
ACCEPT

#Admin 1,2 see web cluster & DNS cluster & 192.168.4.0/24

#for web cluster

  iptables -A FORWARD -s 192.168.1.3 -p tcp --dport 22 -d
192.168.2.0/24 -j ACCEPT

  iptables -A FORWARD -s 192.168.1.4 -p tcp --dport 22 -d
192.168.2.0/24 -j ACCEPT

#for DNS cluster

  iptables -A FORWARD -s 192.168.1.3 -p tcp --dport 22 -d
192.168.3.0/24 -j ACCEPT

  iptables -A FORWARD -s 192.168.1.4 -p tcp --dport 22 -d
192.168.3.0/24 -j ACCEPT

#for 192.168.4.0/24

  iptables -A FORWARD -s 192.168.1.3 -p tcp --dport 22 -d
192.168.4.0/24 -j ACCEPT

  iptables -A FORWARD -s 192.168.1.4 -p tcp --dport 22 -d
192.168.4.0/24 -j ACCEPT
```

```
#iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o eth0 -j
MASQUERADING

#END
```

3. Verify Iptables using the next command :

```
# iptables -l
```

Chapter 8 Configuration Management Using Ansible

Configuration management using Ansible

In this chapter we will cover :

- Introduction .
- How Ansible work ?
- Ansible features .
- What do you need to use Ansible ?
- Example for playbook .
- Ceph automation .
- Red Hat clustering automation .
- OpenStack automation .
- Automate OpenStack Integration With Ceph Block Device (RBD).

8.1 Introduction :

Ansible is an open source , powerful automation tool for configuring, managing and deploying all servers at the same time instead of managing each server individually . Ansible describe state of servers and force server to do this, **Figure 8.1** show difference between Ansible and other CM platform . The reference of this chapter is [14] , [15] and [16] .

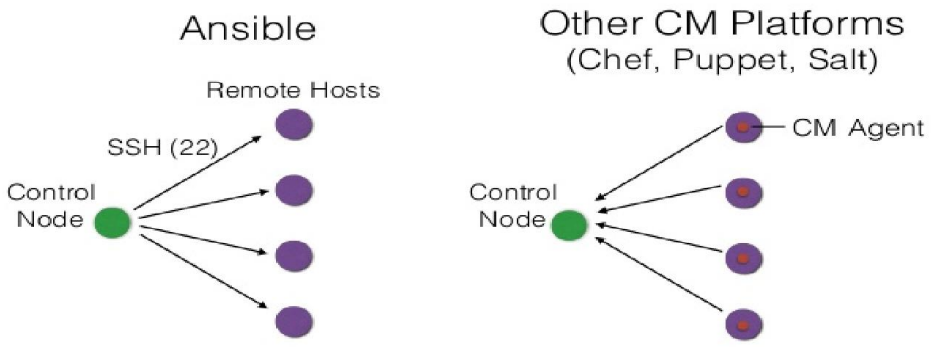


Figure 8.1 : Difference between Ansible and other CM platform

Ansible has 2 modes :

1. Ad-Hoc .

2. playbook .

8.2 How Ansible work ?

1. Ansible playbook describe :

- Hosts to configure .

- Tasks to be run on this hosts .

2. When you run Yaml playbook , Ansible will open parallel ssh connections to all remote hosts , and start to run tasks on it .

3. At playbook running , Ansible generate python script for tasks , copy and run them at the remote hosts through SSH connection .

4. Ansible repeat all the steps for each task .

8.3 Ansible features :

1. Easy to read :

- Script called playbook .

- Playbook built on top of Yaml which is data format for human read and write .

Note : Playbooks are bunch of commands which can perform multiple tasks and each playbooks are in YAML file format .

2. Agent less : there is no agent required at remote hosts , agent only for performance improvement .

3. Ansible is push-based as there is no agent to pull configuration unlike other configuration management 'puppet , chef' which needs agent to pull .

Note : pull-based is suitable for servers come any time .

4. Ansible has large numbers of modules such as 'setup, file, yum, service, copy, ... etc' .

5. Ansible module is idempotent .

6. Choosing modules and writing scripts depending on the operation system we use .

8.4 What do you need to use Ansible ?

To be able to use Ansible you need to know :

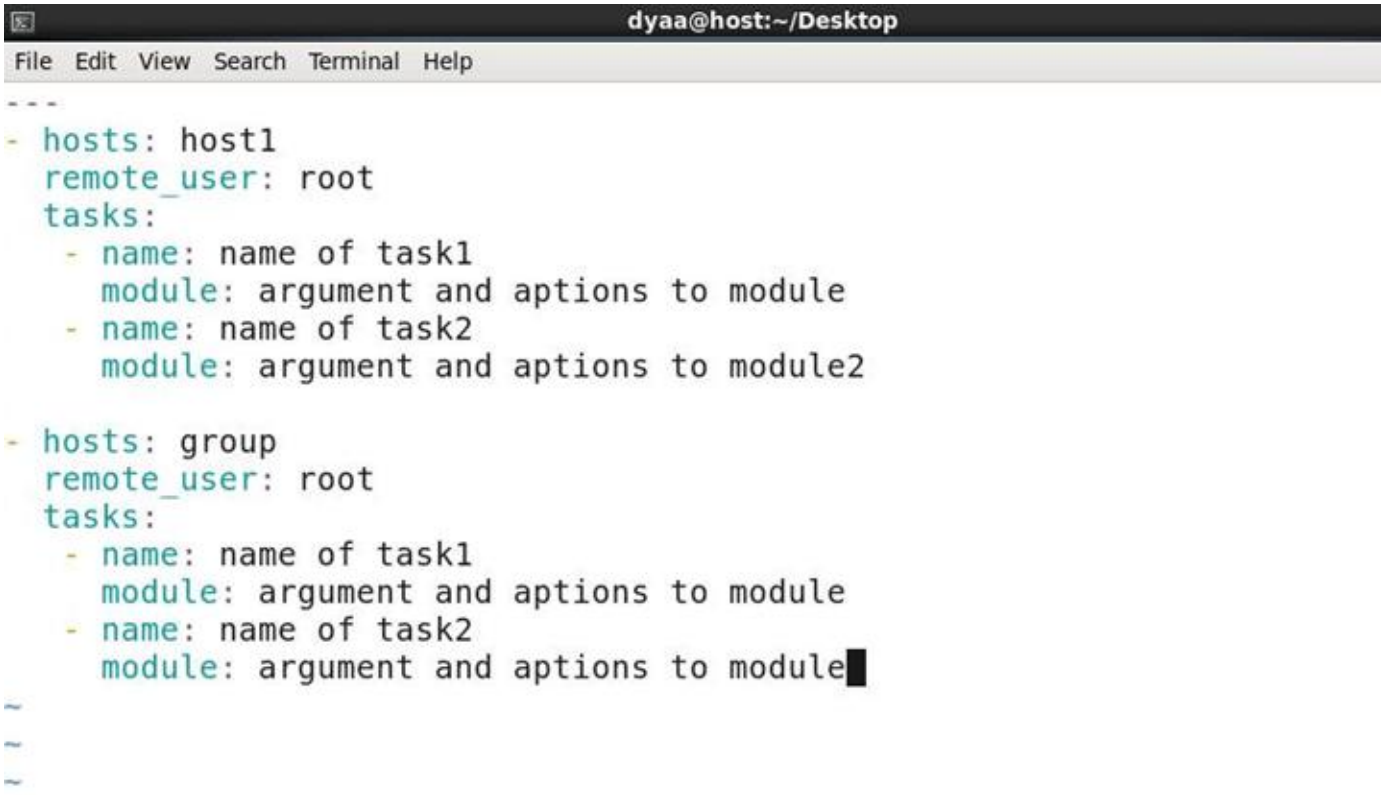
- SSH (Secure Shell) concept .
- Pipe and redirection .
- Package installation .
- Sudo command .
- Start and stop services .
- Set environmental variables .
- You don't need to know python unless you need to write specific module .

8.5 Playbook example :

1. To be able to run any playbook you must install Ansible first and prepare hosts file as shown:

```
# yum -y  install epel-release

# yum -y install ansible

# vim /etc/ansible/hosts

             # the file syntax such as comming

             host1

             host2

             [group]

             host3

             host4
```

2. Write the script as shown :



3. Run yml script :

```
# ansible-playbook    playbook1.yml
```

Note : If you configure hosts at different file use -i option to specify the path of the file .

Note : You need root access to run the next command :

```
# ssh-copy-id  remote host1

# ssh-copy-id  remote host2
```

8.6 Ceph storage automation :

1. Edit /etc/ansible/templates/hosts.j2 :

```
# vim /etc/ansible/templates/hosts.j2

127.0.0.1   localhost localhost.localdomain {{ ansible_fqdn }}
{{ ansible_hostname }}

::1         localhost localhost.localdomain {{ ansible_fqdn }}
{{ ansible_hostname }}

192.168.1.11       controller.project.local        controller

192.168.1.31       compute.project.local           compute

192.168.1.41       cinder.project.local            cinder

192.168.1.50       admin.project.local             admin

192.168.1.51       mon.project.local               mon

192.168.1.52       osd1.project.local              osd1

192.168.1.53       osd2.project.local               osd2

192.168.2.11       nodea.project.local              nodea

192.168.2.21       nodeb.project.local              nodeb

192.168.2.31       nodec.project.local              nodec
```

2. Edit /etc/ansible/hosts :

```
# vim /etc/ansible/hosts

      repo.project.local

      [ceph]

      admin.project.local

      mon.project.local

      osd.project.local
```

```
osd2.project.local

[cluster]

nodea.project.local

nodeb.project.local

nodec.project.local

[openstack]

controller.project.local
```

3. Edit /etc/ansible/ice-answers :

```
# vim /etc/ansible/ice-answers.j2

y

/mnt/

admin.project.local

http
```

4. Edit /etc/ansible/templates/newmon-answers.j2 :

```
# vim /etc/ansible/newmon-answers

yes
```

5. Edit /etc/ansible/repo :

```
# vim /etc/ansible/repo

[localrepo]

name=local

baseurl=http://repo.project.local/rh/

enabled=1
```

```
gpgcheck=0
```

6. Edit ceph.yml and write the script as shown :

```
# vim /etc/ansible/ceph.yml

---

- hosts: repo
  remote_user: root
  tasks:

              - name: hosts file

                template: src=/etc/ansible/templates/hosts.j2
dest=/etc/hosts mode=0644

              - name: make selinux permissive

                selinux: policy=targeted state=permissive

              - name: disable firewall

                service: name=firewalld state=stopped enabled=no

              - name: start apache server

                service: name=httpd state=started enabled=yes

              - name: mkdir /var/www/html/rh

                file: path=/var/www/html/rh state=directory
mode=0777



              - name: mount rh iso to /var/www/html/rh

                shell: mount /osi/rh.iso /var/www/html/rh -o loop
```

```
  - hosts: ceph
    remote_user: root
    tasks:

              - name: make selinux permissive

                selinux: policy=targeted state=permissive

              - name: disable firewall

                service: name=firewalld state=stopped enabled=no

              - name: hosts file

                template: src=/etc/ansible/templates/hosts.j2
dest=/etc/hosts mode=0644

              - name: yum repos client to admin

                copy: src=/etc/ansible/repo
dest=/etc/yum.repos.d/local.repo mode=0777

- hosts: mon
  remote_user: root
  tasks:

              - name: set hostname

                shell: hostnamectl set-hostname
"mon.project.local"

 - hosts: osd1
   remote_user: root
   tasks:

              - name: set hostname

                shell: hostnamectl set-hostname
"osd1.project.local"

- hosts: osd2
  remote_user: root
  tasks:
```

```
            - name: set hostname osd2

              shell: hostnamectl set-hostname
"osd2.project.local"

- hosts: admin
  remote_user: root
  tasks:

            - name: set hostname

              shell: hostnamectl set-hostname
"admin.project.local"

            - name: ensure ssh to mon

              shell: ssh mon ls /

            - name: copy file

              copy: src=/etc/ansible/ice-answers dest=/ice-
answers

            - name: copy /ice-answers to current location

              shell: cp /ice-answers .

            - name: copy newmon-answers

              copy: src=/etc/ansible/newmon-answers
dest=/newmon-answers

            - name: copy /newmon-answers to current location

              shell: cp /newmon-answers .

            - name: create dir to copy iso

              file: path=/isos state=directory mode=0777

            - name: copy iso

              copy: src=/isos/ceph.iso dest=/isos

            - name: mount iso to /mnt
```

```
              shell: mount /iso/ceph.iso /mnt

            - name: install ice_setup and ceph-deploy

              shell: rpm -ivh /mnt/Installer/ice_*.rpm

            - name: install ceph use ice_setup

              shell: ice_setup -d /mnt < ice-answers

            - name: install ceph

              shell: ceph-deploy new mon <newmon-answers

            - name: modify ceph.conf1

              shell: echo "cluster_network =
192.168.1.0/24">>ceph.conf

            - name: modify ceph.conf2

              shell: echo "public_network =
192.168.1.0/24">>ceph.conf

            - name: install mon repo service

              shell: ceph-deploy install --repo --release=ceph-
mon mon

            - name: modify osd repo

              shell: ceph-deploy install --repo --release=ceph-
osd mon osd

            - name: install ceph mon service

              shell: ceph-deploy install --mon mon

            - name: install ceph mon service

              shell: ceph-deploy install --osd1 osd2 mon
```

7. Run ceph.yml script :

```
# ansible-playbook    ceph.yml
```

8.7 Red Hat clustering automation :

1. hosts file already configured at Ceph sections 8.5

2. Edit /etc/ansible/hapassword :

```
# vim /etc/ansible/hapassword

      hacluster

       hapassword
```

3. Edit /etc/ansible/template/web.j2 :

```
# vim /etc/ansible/template/web.j2

            <html>
            graduation project
            </html>
```

4. Edit cluster.yml and write the script as shown :

```
# vim /etc/ansible/cluster.yml

---

- hosts: cluster

  remote_user: root

  tasks:

            - name: configure hosts file

              template: src=/etc/ansible/templates/hosts.j2
dest=/etc/hosts

            - name: make selinux permissive
              selinux: policy=targeted state=permissive
```

```
            - name: disable firewall
              service: name=firewalld state=stopped enabled=no

            - name: install epel-release
              yum: name=epel-release state=installed

            - name : install apache
              yum: name=httpd state=installed

            - name: start httpd
              service: name=httpd state=started

            - name: create web page
              template: src=/etc/ansible/templates/web.j2
dest=/var/www/html/index.html

            - name: install cluster software
              shell : yum install -y pacemaker pcs psmisc
policycoreutils-python

            - name: start and enable pcs service
              service: name=pcsd state=started enabled=yes

            - name: set user hacluster password
              shell: echo "hapassword" | passwd hacluster --
stdin

- hosts: nodea

  remote_user: root

  tasks:

            - name: copy hapassword
              copy: src=/etc/ansible/hapassword dest=/hapassword

            - name: copy hapassword to current dir
              shell: cp /hapassword .

            - name: ensure hacluster
              shell: pcs cluster auth nodea nodeb nodec
<hapassword
```

```
            - name: create cluster
              shell: pcs cluster setup --name mycluster nodea
nodeb nodec

            - name: start cluster
              shell: pcs cluster start --all

            - name: stop stonith
              shell: pcs property set stonith-enabled=false

            - name: add ip resource
              shell: pcs resource create ClusterIP
ocf:heartbeat:IPaddr2 ip=192.168.1.102 cidr_netmask=32 op
monitor interval=30s

            - name: add apache resource
              shell: pcs resource create WebSite
ocf:heartbeat:apache configfile=/etc/httpd/conf/httpd.conf op
monitor interval=1min

            - name: ensure that all resource run on same host
              shell: pcs constraint colocation add WebSite with
ClusterIP INFINITY
            - name: ensure that resources run on order
             shell: pcs constraint order ClusterIP then WebSite
```

5. Run cluster.yml script :

```
# ansible-playbook    cluster.yml
```

8.8 OpenStack cloud computing automation :

1. Edit /etc/ansible/openstack.yml :

```
# vim /etc/ansible/openstack.yml

---

- hosts: openstack

  remote_user: root
```

```
  roles:

      - openstack
```

2. Create roles dir as shown :

```
# mkdir /etc/ansible/roles/openstack/{tasks,files}
```

3. Create and edit tasks file as shown :

```
# vim /etc/ansible/roles/openstack/tasks/main.yml

---

- name: install packstack

  yum: name=openstack-packstack state=latest

- name: copy answer file to node

  copy: src=/etc/ansible/roles/openstack/files/answers.txt
dest=/answers.txt

- name: run packstack using answer file

  shell: packstack --answer-file /answers.txt

- name: conf network1

  copy: src=/etc/ansible/roles/openstack/files/ifcfg-eth0
dest=/etc/sysconfig/network-scripts/ifcfg-eth0

- name: conf network2

  copy: src=/etc/ansible/roles/openstack/files/ifcfg-br-ex
dest=/etc/sysconfig/network-scripts/ifcfg-br-ex

- name: restart network service

  service: name=NetworkManager state=restarted enabled=yes
```

4. Create and edit needed files as shown :

```
# vim /etc/ansible/roles/openstack/files/answers.txt

CONFIG_CONTROLLER_HOSTS=192.168.1.11

CONFIG_COMPUTE_HOSTS=192.168.1.21

CONFIG_NETWORK_HOSTS=192.168.1.31

CONFIG_STORAGE_HOST=192.168.1.11

CONFIG_HORIZON_SSL=y

CONFIG_PROVISION_DEMO=n
```

5. Edit /etc/ansible/roles/openstack/files/ifcfg-br-ex as shown :

```
# vim /etc/ansible/roles/openstack/files/ifcfg-br-ex

            DEVICE=br-ex

            BOOTPROTO=static

            ONBOOT=yes

            TYPE=OVSBridge

            DEVICETYPE=ovs

            USERCTL=yes

            PEERDNS=yes

            IPV6INIT=no

            IPADDR=192.168.1.31

            NETMASK=255.255.255.0

            GATEWAY=192.168.1.1

            DNS1=192.168.1.1
```

6.Edit /etc/ansible/roles/openstack/files/ifcfg-eno16777736 as shown :

```
# vim /etc/ansible/roles/openstack/files/ifcfg-eno16777736

             DEVICE=eno16777736

             ONBOOT=yes

             TYPE=OVSPort

             DEVICETYPE=ovs

             OVS_BRIDGE=br-ex
```

7. Run openstack.yml script :

```
# ansible-playbook    openstack.yml
```

8.9 Automate OpenStack Integration With Ceph Block Device (RBD):

1. Create and edit /etc/ansible/integration.yml as shown :

```
# vim /etc/ansible/integration.yml

---

- hosts: ceph

  remote_user: root

  tasks:

   - name: install ceph-common python rbd

     shell: yum install -y python-rbd ceph-common


- hosts: mon

  remote_user: root
```

```
  tasks:

   - name: create pool

     shell: ceph osd pool create volumes 128

   - name: create cinder user and keyring

     shell: ceph auth get-or-create client.cinder mon 'allow r'
osd 'allow class-read object_prefix rbd_children, allow rwx
pool=volumes' > /home/ceph.client.cinder.keyring

   - name: key ring for client

     shell: ceph auth get-key client.cinder >
/home/client.cinder.key

   - name: fetch ceph.client.keyring

     fetch: src=/home/ceph.client.cinder.keyring
dest=/etc/ansible/roles/integration/files/ceph.client.cinder.key
ring

   - name: fetch ceph.conf

     fetch: src=/etc/ceph/ceph.conf
dest=/etc/ansible/roles/integration/files/ceph.conf

   - name: fetch client cinder key

     fetch: src=/home/client.cinder.key
dest=/etc/ansible/files/client.cinder.key



- hosts: controller

  remote_user: root

  tasks:

   - name: copy conf file to controller

     copy: src=/etc/ansible/files/ceph.conf dest=/etc/ceph.conf
```

```
   - name: copy cinder keyring to controller

     copy: src=/etc/ansible/files/cceph.client.cinder.keyring
dest=/etc/ceph/ceph.client.cinder.keyring

   - name: chmod cinder file

     shell: chown cinder:cinder
/etc/ceph/ceph.client.cinder.keyring

   - name: copy client cinder key

     copy: src=/etc/ansible/files/client.cinder.key
dest=/root/client.cinder.key

   - name: uuid

     shell: uuidgen > /home/uuid

   - name: copy secret.xml

     copy: src=/etc/ansible/files/secret.xml
dest=/root/secret.xml

   - name: modify secret.xml

     shell: sed -i -e "s/uuidno/$(cat /home/uuid)/g"
/root/secret.xml

   - name: virsh secret

     shell: virsh secret-define --file /root/secret.xml

   - name: virsh secret2

     shell: virsh secret-set-value --secret $(cat /home/uuid) --
base64 $(cat /root/client.cinder.key)

   - name: delete files

     shell: rm /root/client.cinder.key /root/secret.xml -rf

   - name: copy modify cinder.conf
```

```
     copy: src=/etc/ansible/files/cinder.conf.modify
dest=/home/cinder.conf.modify

   - name: edit cinder.conf

     shell: cat /home/cinder.conf.modify >>
/etc/cinder/cinder.conf

   - name: edit conf2

     shell: sed -i -e "s/uuid2/$(cat /home/uuid)/g"
/etc/cinder/cinder.conf

   - name: edit conf3

     shell: sed -i -e
"s/enabled_backends=lvm/enabled_backends=rbd/g"
/etc/cinder/cinder.conf

- hosts: controller:compute

  remote_user: root

  tasks:

   - name: copy uuid to all nodes

     copy: src=/etc/ansible/files/uuid dest=/home/uuid

   - name: modify nova.conf

     shell: echo "rbd_user = cinder" >> /etc/nova/nova.conf

   - name: modify nova.conf2

     shell: echo "rbd_secret_uuid = $(cat /home/uuid)" >>
/etc/nova/nova.conf

   - name: restart services

     shell: openstack-service restart nova

   - name: restart services

     shell: openstack-service restart cinder
```

```
- hosts: controller:controller
  remote_user: root
  tasks:
   - name: keystonerc
     shell: source /root/keystonerc_admin
   - name: create volumes
     shell: cinder create --display-name cephvol1 20
   - name: create volumes
     shell: cinder create --display-name cephvol2 20
   - name: create volumes
     shell: cinder create --display-name cephvol3 20
```

2. Edit /etc/ansible/files/cinder.conf.modify as shown :

```
# vim /etc/ansible/files/cinder.conf.modify
[rbd]
volume_driver = cinder.volume.drivers.rbd.RBDDriver
rbd_pool = volumes
rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
glance_api_version = 2
```

```
rbd_user = cinder

rbd_secret_uuid = uuid2
```

3. Run integration.yml script :

```
# ansible-playbook    integration.yml
```

8.10 Automation selective product shell script:

1. Create and edit grad-proj.sh as shown :

```
#!/bin/bash

#integreated infrastructure

echo -e "\033[33m  Integrated infrastructure using Ansible conf
management"

echo "================================================

This script deploy ceph , openstack , integrate openstack with
ceph , deploy pacemaker cluster over openstack
If you want to deploy product press number beside product

                1- deploy all infrastructure

               2- deploy ceph storage

             3- deploy openstack cloud
             4- integrate ceph with openstack
             5- deploy redhat clustering

          ============================================

                          Note:

                        ----------

look at our book to know how to prepare nodes before deployment
```

```
                        Contact info

                        =-=-=-=-=-=-=

          ============================================

          ||  smart.tuxproj@gmail.com                ||

          || linkedin.com/in/the-smart-tux-93a904122 ||

          ============================================

     "  echo -n "choose no of product you want to deploy: "

read x
if [ "$x" -eq 2 ]
then
echo "installing ceph"
ansible-playbook /etc/ansible/ceph.yml
elif [ "$x" -eq 3 ]
then
echo "installing openstack"
ansible-playbook /etc/ansible/openstack.yml
elif [ "$x" -eq 4 ]
then
echo "integrating ceph with openstack"
ansible-playbook /etc/ansible/integration.ymlelif [ "$x"  -eq 5
]
then
echo "installing pacemaker cluster"
ansible-playbook /etc/ansible/cluster.yml

elif [ "$x" -eq 1 ]
then
echo "installing all"
ansible-playbook /etc/ansible/ceph.yml
ansible-playbook /etc/ansible/openstack.yml
ansible-playbook /etc/ansible/integration.yml
ansible-playbook /etc/ansible/cluster.yml
else
echo "try again"
fi
```

2. Run grad-proj.sh script as shown :

```
# sh grad-proj.sh
```

Chapter 9
Results and discussions

Results and discussions

Now we will list some screenshots for our project results to ensure that it is working :

9.1 OpenStack cloud computing results :

The following screenshot show the OpenStack dashboard which manage all OpenStack services and nodes using Graphical User Interface (GUI) as shown :

9.2 Ceph storage results :

The following screenshot show Ceph verification result :

```
[root@mon ~]# ceph -s
    cluster bf7b18e7-8e58-410f-b674-b81755127fd6
     health HEALTH_OK
     monmap e1: 1 mons at {mon=192.168.1.51:6789/0}
            election epoch 2, quorum 0 mon
     osdmap e20: 3 osds: 3 up, 3 in
      pgmap v52: 64 pgs, 1 pools, 0 bytes data, 0 objects
            102 MB used, 149 GB / 149 GB avail
                  64 active+clean
[root@mon ~]# ceph health
HEALTH_OK
[root@mon ~]#
```

9.3 OpenStack Integration With Ceph Block Device (RBD) results :

The following screenshot show result of OpenStack Integration With Ceph Block Device (RBD) :

```
[root@controller ~(keystone_admin)]# cinder list
+--------------------------------------+-----------+--------------+------+------
-------+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volum
e Type | Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+------
-------+----------+-------------+
| 1885283a-f586-4411-b304-4d392b5bbb64 | available |    shosho    |  1   |     N
one    |  false   |             |
| 1a7523f1-c8c2-4d55-95e3-449063043f11 | available |   cephvol1   |  20  |     N
one    |  false   |             |
| 34733f36-1f9a-44f5-854f-1d361a9c6f04 | available |   cephvol2   |  20  |     N
one    |  false   |             |
| ec08ebee-df41-4c0e-ac7f-3f3d3050680f | available |   cephvol3   |  20  |     N
one    |  false   |             |
+--------------------------------------+-----------+--------------+------+------
-------+----------+-------------+
[root@controller ~(keystone_admin)]#
```

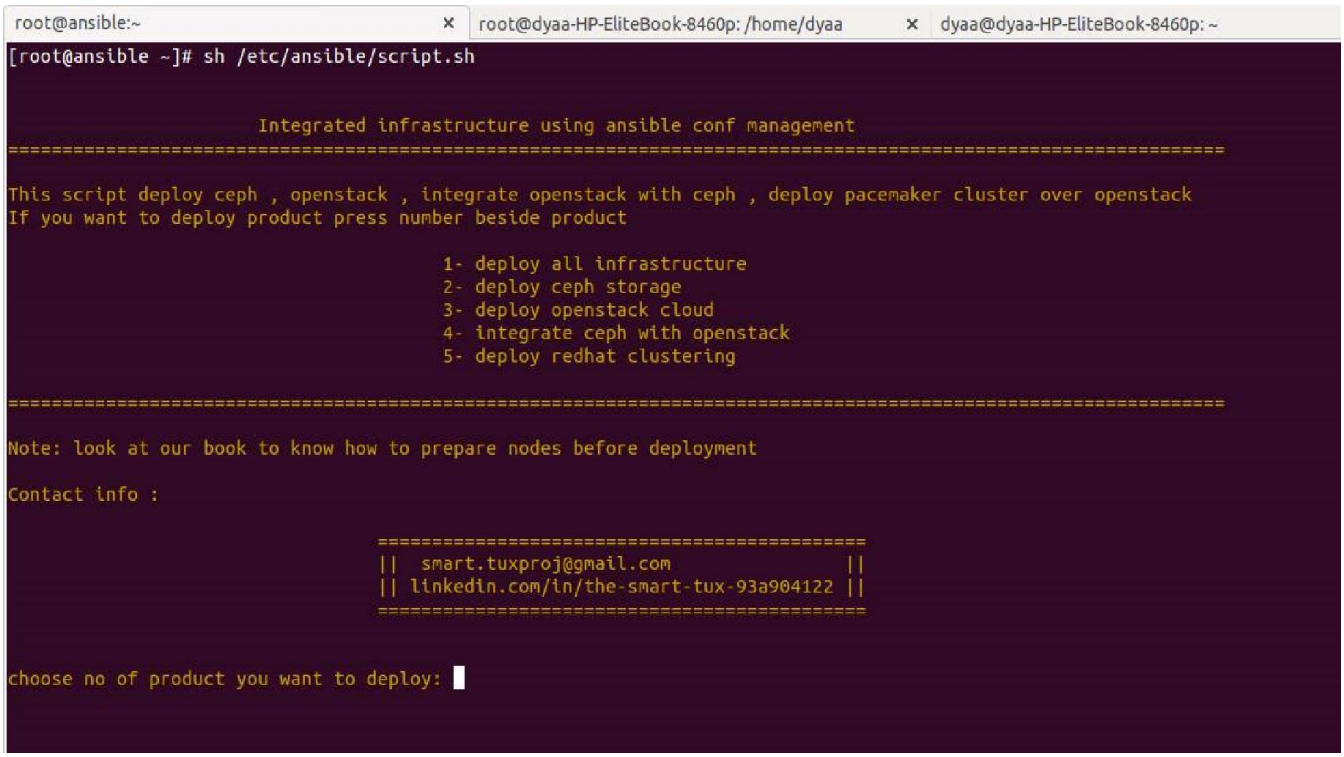
9.4 Red Hat clustering results :

The following screenshot show Red Hat clustering verification result :

```
[root@localhost ~]# pcs cluster status
Cluster Status:
 Last updated: Wed Jun 22 11:22:45 2016          Last change: Sat Jun 18 14:02:29 2016 by hacluster via crmd on nodea
 Stack: corosync
 Current DC: nodea (version 1.1.13-10.el7_2.2-44eb2dd) - partition with quorum
 3 nodes and 2 resources configured
 Online: [ nodea nodeb nodec ]

PCSD Status:
  nodea: Online
  nodeb: Online
  nodec: Online
[root@localhost ~]#
```

9.5 Automation using Ansible Configuration management results :

The following screenshot show that; before you start automation you will be asked which product you want to deploy . For example: if you want to deploy (Ceph product) press 2 and the Ceph automation file will be directly start working .



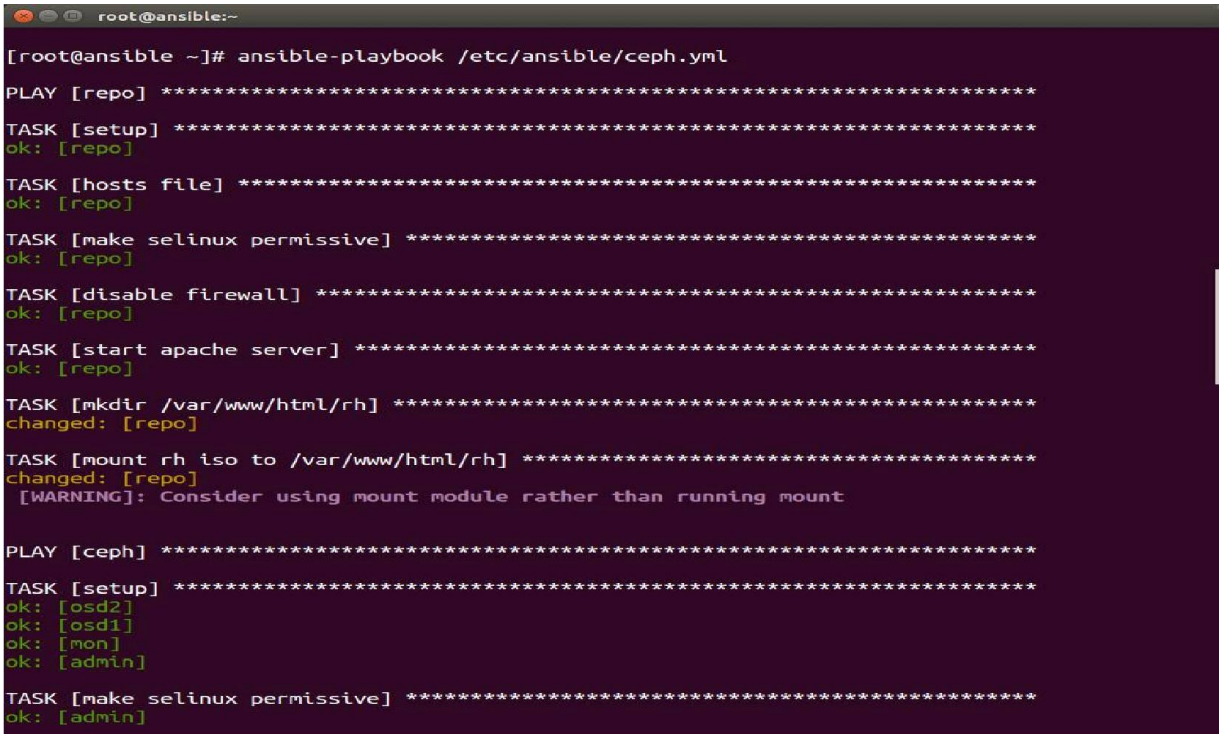
9.5.1 OpenStack automation results :

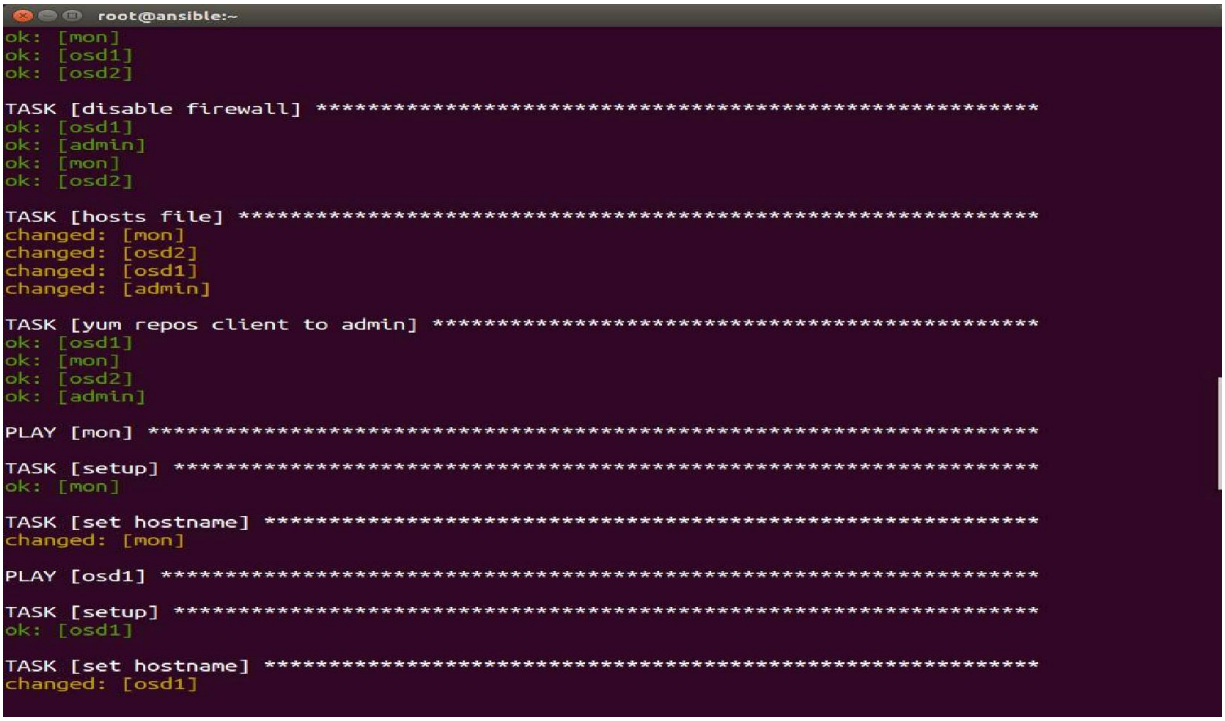
The incoming screenshot show that OpenStack cloud computing is successfully installed and gives you some information to helps you such as how to access OpenStack environment using dashboard as shown :

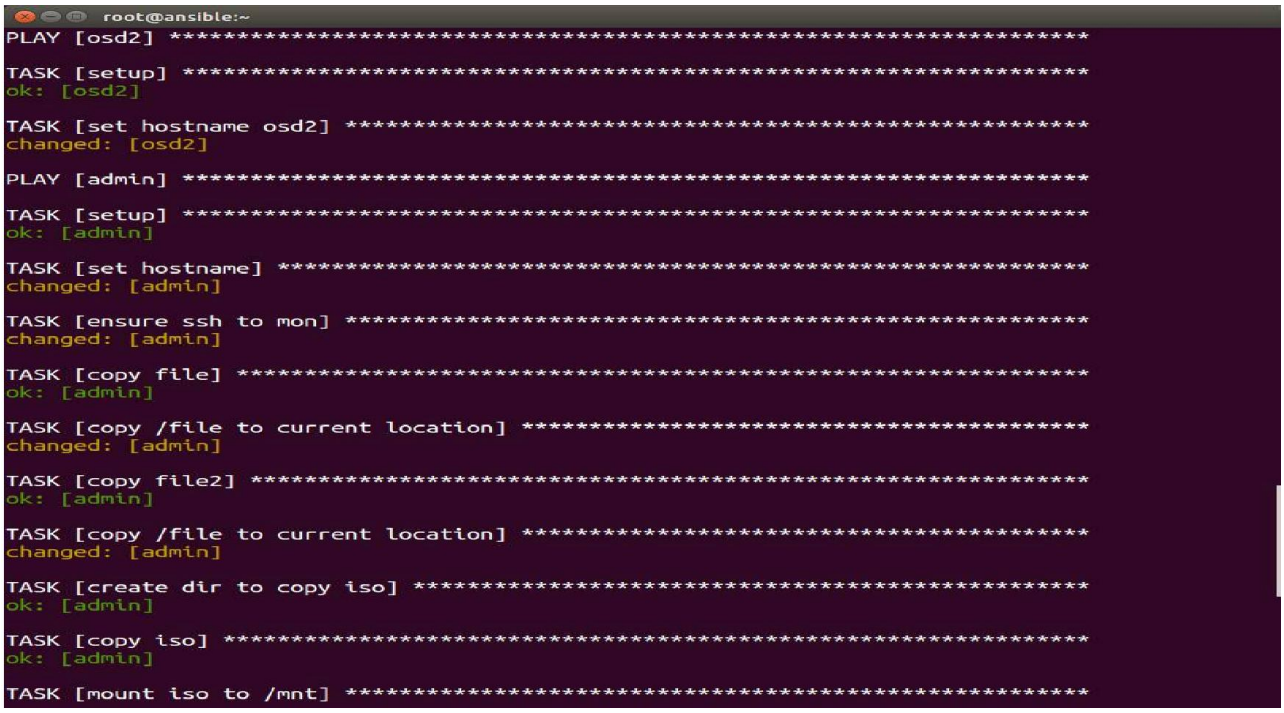
```
 **** Installation completed successfully ******

Additional information:
 * Time synchronization installation was skipped. Please note that unsynchronized time on server instances might be
 problem for some OpenStack components.
 * File /root/keystonerc_admin has been created on OpenStack client host 192.168.1.11. To use the command line tool
s you need to source the file.
 * NOTE : A certificate was generated to be used for ssl, You should change the ssl certificate configured in /etc/
httpd/conf.d/ssl.conf on 192.168.1.11 to use a CA signed cert.
 * To access the OpenStack Dashboard browse to https://192.168.1.11/dashboard .
Please, find your login credentials stored in the keystonerc_admin in your home directory.
 * To use Nagios, browse to http://192.168.1.11/nagios username: nagiosadmin, password: 632ed0e2a46f4071
 * The installation log file is available at: /var/tmp/packstack/20160701-003459-5yQ0go/openstack-setup.log
 * The generated manifests are available at: /var/tmp/packstack/20160701-003459-5yQ0go/manifests
[root@controller Desktop]#
```

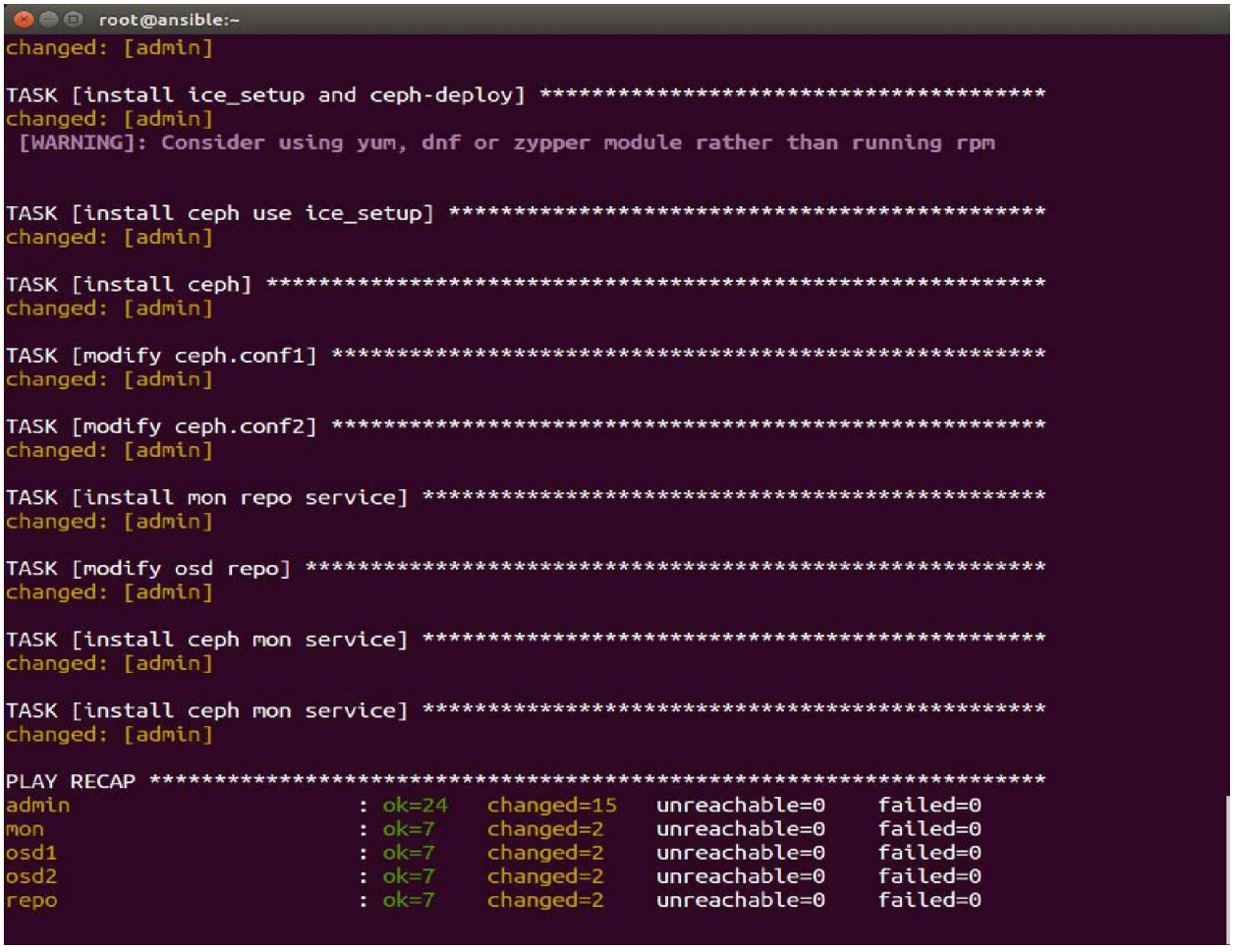
9.5.2 Ceph storage automation results :

The incoming screenshots show Ceph while it is running automation file ceph.yml and it is successfully installed as shown :



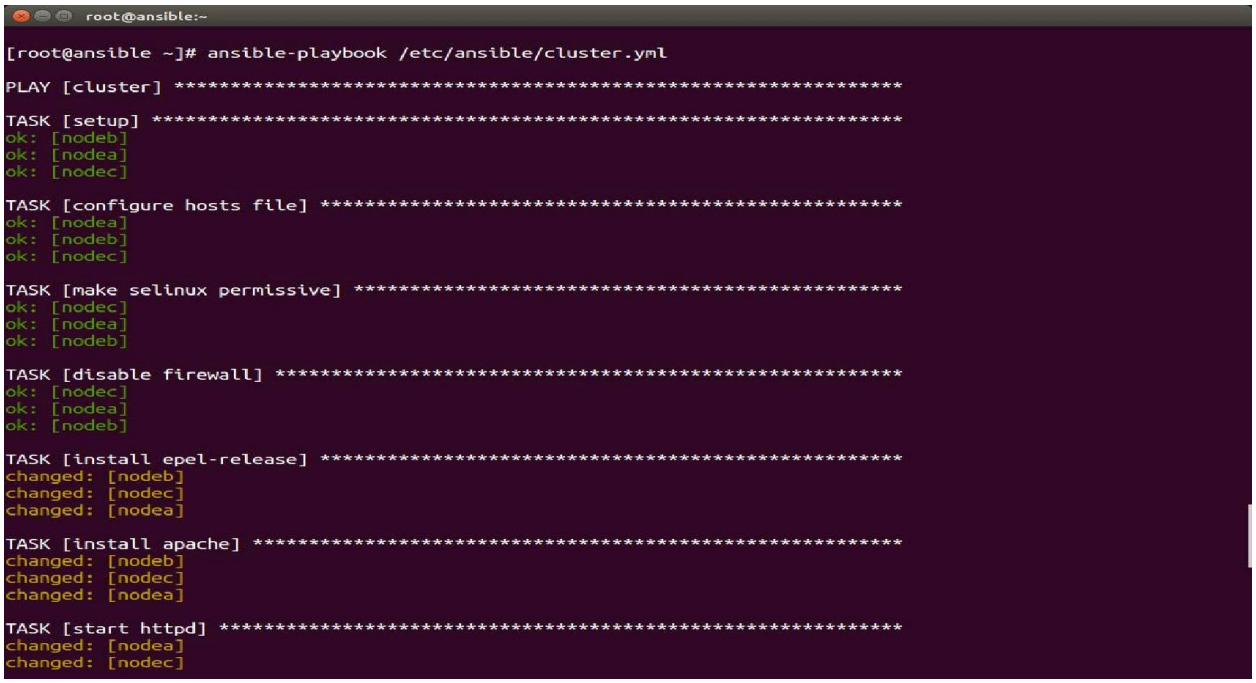


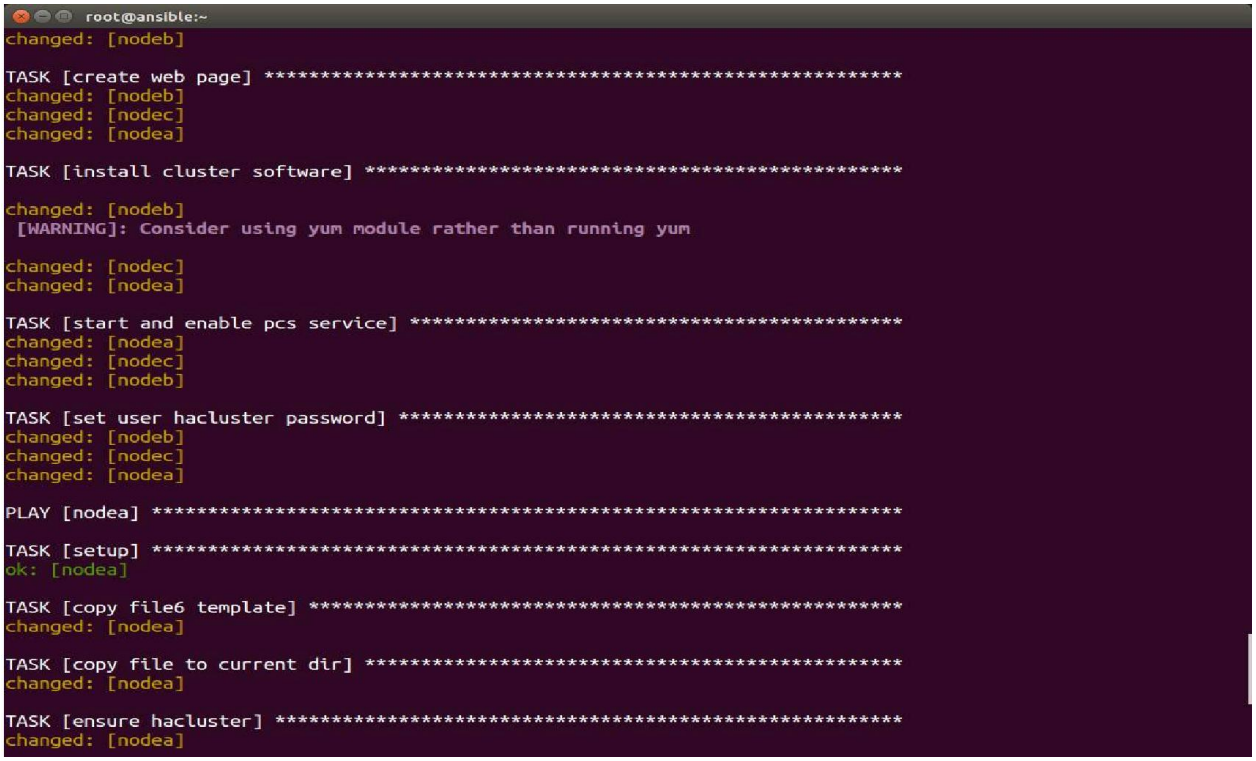

root@ansible:~
PLAY [osd2] ***
TASK [setup] ***
ok: [osd2]
TASK [set hostname osd2] ***
changed: [osd2]
PLAY [admin] ***
TASK [setup] ***
ok: [admin]
TASK [set hostname] **
changed: [admin]
TASK [ensure ssh to mon] ***
changed: [admin]
TASK [copy file] ***
ok: [admin]
TASK [copy /file to current location] **
changed: [admin]
TASK [copy file2] **
ok: [admin]
TASK [copy /file to current location] **
changed: [admin]
TASK [create dir to copy iso] **
ok: [admin]
TASK [copy iso] **
ok: [admin]
TASK [mount iso to /mnt] ***

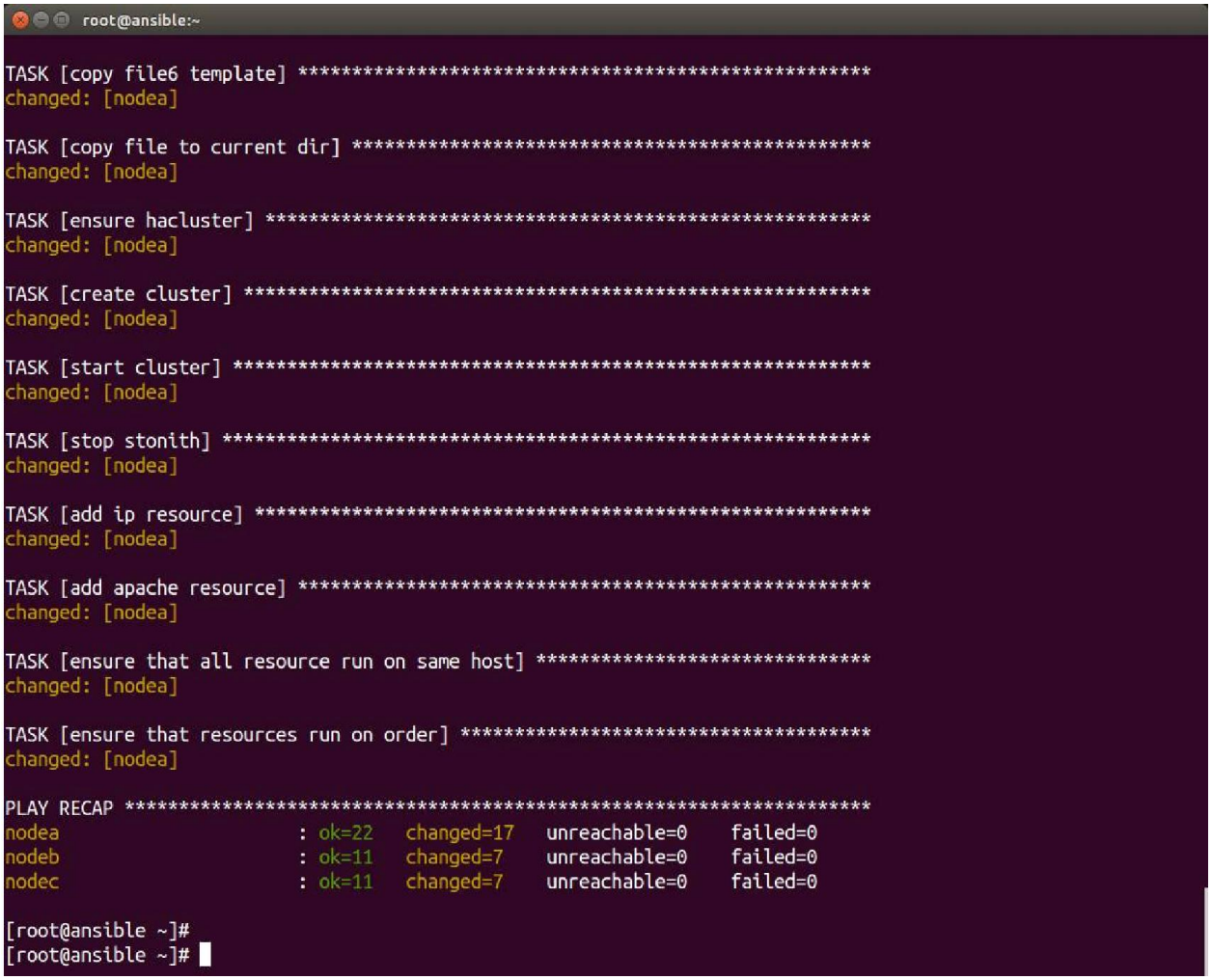

root@ansible:~
changed: [admin]
TASK [install ice_setup and ceph-deploy] ***************************************
changed: [admin]
[WARNING]: Consider using yum, dnf or zypper module rather than running rpm
TASK [install ceph use ice_setup] **
changed: [admin]
TASK [install ceph] **
changed: [admin]
TASK [modify ceph.conf1] ***
changed: [admin]
TASK [modify ceph.conf2] ***
changed: [admin]
TASK [install mon repo service] **
changed: [admin]
TASK [modify osd repo] ***
changed: [admin]
TASK [install ceph mon service] **
changed: [admin]
TASK [install ceph mon service] **
changed: [admin]
PLAY RECAP ***
admin : ok=24 changed=15 unreachable=0 failed=0
mon : ok=7 changed=2 unreachable=0 failed=0
osd1 : ok=7 changed=2 unreachable=0 failed=0
osd2 : ok=7 changed=2 unreachable=0 failed=0
repo : ok=7 changed=2 unreachable=0 failed=0

9.5.3 Red Hat clustering automation results :

The incoming screenshots show Red Hat clustering while it is running automation file cluster.yml and it is successfully installed as shown :






root@ansible:~
TASK [copy file6 template] **
changed: [nodea]
TASK [copy file to current dir] ***
changed: [nodea]
TASK [ensure hacluster] ***
changed: [nodea]
TASK [create cluster] ***
changed: [nodea]
TASK [start cluster] **
changed: [nodea]
TASK [stop stonith] ***
changed: [nodea]
TASK [add ip resource] **
changed: [nodea]
TASK [add apache resource] **
changed: [nodea]
TASK [ensure that all resource run on same host] ******************************
changed: [nodea]
TASK [ensure that resources run on order] *************************************
changed: [nodea]
PLAY RECAP **
nodea : ok=22 changed=17 unreachable=0 failed=0
nodeb : ok=11 changed=7 unreachable=0 failed=0
nodec : ok=11 changed=7 unreachable=0 failed=0
[root@ansible ~]#
[root@ansible ~]#

Chapter 10 Conclusion and future Work

Conclusion and future work

10.1 Conclusion :

Our project is a data center used **OpenStack cloud computing,** which is a single computer to be portioned into multiple virtual computers each run its own operating system with **Ceph storage product** as backend storage to provide high availability, performance, reliability and scalability .

We build a pacemaker cluster over OpenStack instances acting like a single system to provide high-availability services and resources by redundant multiple machines .

Then we configured firewall rules to provide secure connectivity between internal and external networks .This rules achieve :

- Remote user connect only to web cluster .
- Web developer cluster connect only to internet and web cluster .
- Admin nodes connect to all infrastructure .
- No one except admins can connect directly to OpenStack or Ceph .

Last step is automating all products using **Ansible** configuration management to be run automatically and make deployment faster and easier .

10.2 Future work :

1. Automate more products such as :

- **Red Hat Network (RHN)** which is a family of systems-management services, makes updates, patches, and bug fixes of packages .

- **Red Hat Satellite** because From a performance side, it reduces hits to your network bandwidth because local systems can download everything they need locally; from a security side, it can limit the risks of malicious content or access, even enabling entirely disconnected environments.

- **OpenShift** which is a platform-as-a-Service (PaaS) cloud for open source developers . It provides developers with a choice in languages, frameworks, and clouds to build, test, run, and manage Java, Ruby, PHP, Perl and Python applications. Developers can also choose the cloud provider the applications will run on. .

2. Adding new features and modifying the script to be more organized and achieving higher user experience, such as adding Graphical user interface (GUI) .

3. Modify the design to achieve more availability , security, more data reliable and more stability .

4. Marketing the project over our website .

5. Make the project code open source for any user to be able to modify or add new features .

Appendix A OpenStack cloud computing

APPENDIX A
OpenStack cloud computing

Appendix A.1 : Chrony Configuration .

Controller Node :

```
# vim /etc/chrony.conf
       server        compute1.project.local             iburst
       server block1.project.local        iburst
(OR)  allow 192.168.1.0/24
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

Compute Node :

```
# vim /etc/chrony.conf
       server        controller.project.local    iburst
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

Storage Node :

```
# vim /etc/chrony.conf
       server        controller.project.local    iburst
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

Appendix A.2 : mysql_secure_installation configuration .

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and you
haven't set the root password yet, the password will be blank, so you
should just press enter here.
Enter current password for root (enter for none):
Change the root password? [Y/n] Y
Enter password :
By default, a MariaDB installation has an anonymous user, allowing
anyone to log into MariaDB without having to have a user account
created for them. This is intended only for testing, and to make the
installation go a bit smoother. You should remove them before moving
into a production environment.
Remove anonymous users? [Y/n] y
... Success!
Normally, root should only be allowed to connect from 'localhost'.
This ensures that someone cannot guess at the root password from the
network.
Disallow root login remotely? [Y/n] Y
... Success!
By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
Reloading the privilege tables will ensure that all changes made so
far will take effect immediately.
Reload privilege tables now? [Y/n] y
... Success!
Cleaning up...
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Appendix A.3 : Identity service configuration file .

```
[DEFAULT]
admin_token = 294a4c8a8a475f9b9836

[database]
connection =
mysql+pymysql://keystone:keystone@controller/keystone

[token]
provider = fernet
```

Appendix A.4 : Apache server configuration .

```
# vim /etc/httpd/conf/httpd.conf

      ServerName   controller

# vim /etc/httpd/conf.d/wsgi-keystone.conf

Listen 5000

Listen 35357

<VirtualHost *:5000>

WSGIDaemonProcess keystone-public processes=5 threads=1
user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-public

WSGIScriptAlias / /usr/bin/keystone-wsgi-public

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/httpd/keystone-error.log

CustomLog /var/log/httpd/keystone-access.log combined
```

```
<Directory /usr/bin>

Require all granted

</Directory>

</VirtualHost>

<VirtualHost *:35357>

WSGIDaemonProcess keystone-admin processes=5 threads=1
user=keystone group=keystone display-name=%{GROUP}

WSGIProcessGroup keystone-admin

WSGIScriptAlias / /usr/bin/keystone-wsgi-admin

WSGIApplicationGroup %{GLOBAL}

WSGIPassAuthorization On

ErrorLogFormat "%{cu}t %M"

ErrorLog /var/log/httpd/keystone-error.log

CustomLog /var/log/httpd/keystone-access.log combined

<Directory /usr/bin>

Require all granted

</Directory>

</VirtualHost>
```

Appendix A.5 : Image service configuration file /etc/glance/glance-api.conf .

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


[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 = glance
password = glance

[paste_deploy]
flavor = keystone

[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Appendix A.6 : Image service configuration file /etc/glance/glance-registry.conf .

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

[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 = glance
password = glance

[paste_deploy]
flavor = keystone
```

Appendix A.7 : Compute service configuration file /etc/nova/nova.conf on controller node .

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

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

[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.1.11
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[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

[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

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

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

Appendix A.8 : Compute service configuration file /etc/nova/nova.conf on compute node .

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.1.31
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[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

[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

[libvirt]
virt_type = qemu
```

Appendix A.9 : Network service configuration file /etc/neutron/neutron.conf on controller node .

```
[database]
connection =
mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron

[DEFAULT]
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
rpc_backend = rabbit
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[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 = neutron
password = neutron

[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
```

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

Appendix A.10 : Modular Layer 2 (ML2) plug-in configuration file /etc/neutron/plugins/ml2/ml2_conf.ini on controller node .

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

[ml2_type_flat]
flat_networks = provider

[ml2_type_vxlan]
vni_ranges = 1:1000

[securitygroup]
enable_ipset = True
```

Appendix A.11 : Linux bridge agent configuration file /etc/neutron/plugins/ml2/linuxbridge_agent.ini on controller node .

```
[linux_bridge]
physical_interface_mappings = provider:eno3355046

[vxlan]
enable_vxlan = True
local_ip = 192.168.1.11
l2_population = True

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

Appendix A.12 : Layer3 agent configuration file /etc/neutron/l3_agent.ini on controller node .

```
[DEFAULT]
interface_driver =
neutron.agent.linux.interface.BridgeInterfaceDriver
external_network_bridge = br-ex
```

Appendix A.13 : DHCP agent configuration file /etc/neutron/dhcp_agent.ini on controller node .

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

Appendix A.14 : metadata agent configuration file /etc/neutron/metadata_agent.ini on controller node .

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

Appendix A.15 : Edit Compute service configuration file to enable networking service /etc/nova/nova.conf on controller node .

```
[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
```

```
service_metadata_proxy = True
metadata_proxy_shared_secret = METADATA_SECRET
```

Appendix A.16 : Network service configuration file /etc/neutron/neutron.conf on compute node .

```
[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
roject_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = neutron

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

Appendix A.17 : Linux bridge agent configuration file /etc/neutron/plugins/ml2/linuxbridge_agent.ini on compute node .

```
[linux_bridge]
physical_interface_mappings = provider: eno3355046

[vxlan]
enable_vxlan = True
local_ip = 192.168.1.31
l2_population = True
```

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

Appendix A.18 : Compute service configuration file /etc/nova/nova.conf on compute node .

```
[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
```

Appendix A.19 : Configure dashboard configuration file /etc/openstack-dashboard/local_settings on controller node .

```
OPENSTACK_HOST = "controller"
ALLOWED_HOSTS = ['*']
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

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

OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
```

```
    "volume": 2,
}

OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"

TIME_ZONE = "TIME_ZONE"
```

Appendix A.20 : Block storage configuration file /etc/cinder/cinder.conf on controller node .

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

[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.1.11

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[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 = cinder
password = cinder
```

Appendix A.21 : Compute service configuration file /etc/nova/nova.conf on controller node .

```
[cinder]
os_region_name = RegionOne
```

Appendix A.22 : Block storage configuration file /etc/cinder/cinder.conf on storage node .

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

[DEFAULT]
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 192.168.1.41

enabled_backends = lvm
glance_api_servers = http://controller:9292

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = openstack

[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 = cinder
password = cinder

[lvm]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = lioadm

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

APPENDIX B OpenStack integration With Ceph Block Device (RBD)

APPENDIX B
OpenStack integration With Ceph Block Device (RBD)

Appendix B.1 : Editing /etc/glance/glance-api.conf glance configuration file to use ceph .

Controller Node :

```
# vim /etc/glance/glance-api.conf

[DEFAULT]

        default_store = rbd
        rbd_store_user = glance
        rbd_store_pool = images
        rbd_store_chunk_size = 8
        show_image_direct_url = True #enables copy-on-write cloning of
images
        rbd_store_ceph_conf = /etc/ceph/ceph.conf
# To avoid images getting cached under /var/lib/glance/image-cache/,
add the following entries under the [paste_deploy] section in
/etc/glance/glance-api.conf.

[paste_deploy]

flavor = keystone
```

Appendix B.2 : Editing /etc/cinder/cinder.conf configuration file to use ceph .

Cinder Node :

```
# vim /etc/cinder/cinder.conf

[DEFAULT]
```

```
volume_driver = cinder.volume.drivers.rbd.RBDDriver
rbd_pool = volumes
rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
glance_api_version = 2
rbd_user = cinder
rbd_secret_uuid = 457eb676-33da-42ec-9a8c-9293d545c337
```

Appendix B.3 : Editing /etc/ceph/ceph.conf configuration file .

Compute Node :

```
# vim /etc/ceph/ceph.conf

[client]
        rbd cache = true
        rbd cache writethrough until flush = true
        admin socket = /var/run/ceph/guests/$cluster-
        $type.$id.$pid.$cctid.asok
        log file = /var/log/qemu/qemu-guest-$pid.log
        rbd concurrent management ops = 20
```

Appendix B.4 : Editing /etc/nova/nova.conf configuration file .

Compute Node :

```
# vim /etc/nova/nova.conf

[DEFAULT]

        libvirt_images_type = rbd
        libvirt_images_rbd_pool = vms
        libvirt_images_rbd_ceph_conf = /etc/ceph/ceph.conf
        libvirt_disk_cachemodes=”network=writeback”
```

```
        rbd_user = cinder
        rbd_secret_uuid = 457eb676-33da-42ec-9a8c-9293d545c337

        #disable file injection

        libvirt_inject_password = false
        libvirt_inject_key = false
        libvirt_inject_partition = -2

        #ensure proper live migration

        libvirt_live_migration_flag="VIR_MIGRATE_UNDEFINE_SOURCE,VIR_MI
GRATE_PEER2PEER,VIR_MIGRATE_LIVE,VIR_MIGRATE_PERSIST_DEST,VIR_MIGRATE_
TUNNELLED"
```

References

References

[1] http://docs.openstack.org/
[2] http://docs.openstack.org/mitaka/install-guide-rdo/overview.html
[3] http://ask.openstack.org
[4] http://docs.ceph.com
[5] http://access.redhat.com/documentation/en/red-hat-ceph-storage
[6] http://amazon.com/Learning-Ceph-Karan-Singh/dp/1783985623
[7] http://www.virtualtothecore.com/en/adventures-ceph-storage-part-1-introduction/
[8] http://supportsages.com/blog/2016/01/openstack-integration-with-ceph-block-device/
[9] http://docs.ceph.com/docs/master/rbd/rbd-openstack/
[10] http://clusterlabs.org/doc/en-US/Pacemaker/1.1-plugin/html/Clusters_from_Scratch/
[11] https://www.youtube.com/playlist?list=PLy1Fx2HfcmWB43uqzexU9WjYfrP2vdCxw
[12] http://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/index.html
[13] https://wiki.archlinux.org/index.php/iptables
[14] www.ansible.com/ansible-book
[15] www.linuxcbt.com/demos
[16] http://docs.ansible.com/