Machines	OS	RAM	CPU
Controller	CentOS 7.9	8 GB	4 cores
Compute1	CentOS 7.9	8 GB	4 cores
Compute 2	CentOS 7.9	8 GB	4 cores
Block	CentOS 7.9	4 GB	4 cores
Object 1	CentOS 7.9	4 GB	4 cores
Object 2	CentOS 7.9	4 GB	4 cores

1/ Configurer les interfaces Rx de chaque hôte (Fixer les @ IP, passerelle, DNS etc…)

2/ Editer le ficher hosts en indiquant le nom de chaque machine suivie de sa @ IP.

3/ Tester la connectivité entre les machines.

4/ Installer NTP service (le package chrony)→ time server

- Serveur = Controller
- Clients = les autres machines

5/ Installer le package de la version « Train » de OpenStack sur toutes les machines→centos-release-openstack-train

6/ Installer le package python-openstackclient sur toutes les machines pour rassembler tous les commandes des différents services OpenStack (network, compute, storageetc.) dans un seul shellafin de garder une structure uniforme des commandes.

7/ Installer le package openstack-selinux sur toutes les machines pour integrer OpenStack dans SElinux

8/ Installer le package de la base de données MariaDB dans la machine Controller, puis on ajoute la section [mysql] au niveau de fichier `openstack.conf` en configurant l'@ du Controller comme étant le bind-address

9/ Installer le package RabbitMQ dans la machine Controller pour coordonner les opérations et les états entre les services OpenStack→`rabbitmq-server`

10/ Installer le package Memcached dans la machine Controller pour mettre en cache les jetons utilisés durant Le mécanisme d'authentification des différents services OpenStack→`memcached python-memcached`

11/ Installer le package Etcd dans la machine Controller qui est nécessaire pour le suivi des différents services OpenStack→`etcd`

Keystone (le service d'authentification de Openstack) :

Controller node

1/ Créer une base de données sql intitulé « keystone » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Installer le package Keystone →`openstack-keystone httpd mod_wsgi`

3/ configurer le controller comme étant le serveur HTTP Apache→`httpd.conf`→`ServerNamecontroller`

4/ Créer un projet Openstack→`openstack project create --domain <domain_name> <project_name>`

5/ Créer un user OpenStack→`openstack user create --domain <domain_name> --password-prompt <user_name>` NB : --password-prompt pour définir un mot de passe de user

6/ Créer un role Openstack→`openstack role create <role_name>`

7/ Ajouter le role au projet et au user →`openstack role add --project <project_name> --user <user_name> <role_name>`

8/ demander un jeton d'authentification →`openstack token issue`

Glance (représente un répo d'image nécessaire pour le lancement des instances)

Controller node

1/ Créer une base de données sql intitulé « glance » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Créer un utilisateur glance→`openstack user create --domain <domain_name> --password-prompt glance`

3/ Ajouter le role « admin » au projet et au user glance→`openstack role add --project<project_name> --user glance admin`

4/ créer un service glance→`openstack service create --name image`

5/ Créer des Glance 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/ Installer le package Glance →`openstack-glance`

et éditer le ficher "glance-api.conf" selon la documentation officielle de Openstack

7/ télécharger une image « Cirros » (une distribution Linux minimal)

→`wget http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img`

8/ Uploader l'image "Cirros" au service image OpenStack

→`glance image-create --name "cirros" --file cirros-0.4.0-x86_64-disk.img --disk-format qcow2 --container-format bare --visibility public`

9/ Vérifier l'upload de l'image →`glance image-list`

Placement (Placement est un service OpenStack qui fournit une API pour le suivi des inventaires et des utilisations des ressources cloud afin d'aider les autres services à gérer et à allouer efficacement leurs ressources):

Controller node

1/ Créer une base de données sql intitulé « placement » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Créer un utilisateur placement→`openstack user create --domain <domain_name> --password-prompt placement`

3/ Ajouter le role « admin » au projet et au user placement→`openstack role add --project<project_name> --user placement admin`

4/ Créer un service placement →`openstack service create --name placement placement`

5/ Créer des Placement API EndPoints

`openstack endpoint create --region RegionOneplacement public http://controller:8778`

```
openstack endpoint create --region RegionOneplacement internal http://controller:8778

openstack endpoint create --region RegionOneplacement admin http://controller:8778
```

6/ Installer le package placement →`openstack-placement-api`

et éditer le fichier "placement.conf" selon la documentation officielle de Openstack.

Horizon (tableau de board Openstack qui permet de gérer les instances)

Controller node

1/ installer le package openstack dashbord → `openstack-dashboard`

et éditer le fichier "/etc/openstack-dashboard/local_settings" selon la documentation officielle de Openstack.

Nova (gére les opérations de computing et processing) needs [keystone+Placement+Glance+Horizon]

Controller node

1/ Créer une base de données sql intitulé « nova » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Créer un utilisateur nova→`openstack user create --domain<domain_name> --password-prompt nova`

3/ Ajouter le role « admin » au projet et au user nova→`openstack role add –project <project_name> --user placement admin`

4/ Créer un service nova→`openstack service create --namenova(name)compute(type)`

5/ Créer des Compute API EndPoints

`openstack endpoint create --region RegionOne compute public http://controller:8774/v2.1`

`openstack endpoint create --region RegionOne compute internal http://controller:8774/v2.1`

`openstack endpoint create --region RegionOnecompute admin http://controller:8774/v2.1`

6/ Installer les packages nova→o`penstack-nova-api openstack-nova-conductor openstack-nova-novncproxy openstack-nova-scheduler`

et éditer le fichier "nova.conf" selon la documentation officielle de Openstack.

Compute node1 & Compute node2

1/ Installer le package nova →`openstack-nova-compute`

en éditant le fichier "nova.conf" selon la documentation officielle de Openstack.

2/ Editer la section [libvirt] dans le fichier « nova.conf » afin d'utiliser l'hyperviseur QEMU (Quick EMUlator) dans compute1 et l'hyperviseur KVM dans compute 2.

→`virt_type = qemu`

→`virt_type = kvm`

Neutron (gére les opérations de networking et responsable des communications entres les différents services d'Openstack)

Controller node

1/ Créer une base de données sql intitulé « neutron » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Créer un utilisateur neutron→`openstack user create --domain<domain_name> --password-prompt neutron`

3/ Ajouter le role « admin » au projet et au user nova →`openstack role add –project <project_name> --user neutron admin`

4/ Créer un service nova →`openstack service create --name neutron(name) network(type)`

5/ Créer des Network 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
```

PS: On va configurer l'option 2 networking (Self-service networks) Sans connaissance de l'infrastructure du réseau virtuel implémentée

6/ Installer les packages neutron→`openstack-neutron openstack-neutron-ml2 openstack-neutron-linuxbridge ebtables`

et éditer le fichier "neutron.conf" selon la documentation officielle de Openstack.

Compute node1 & Compute node2

1/ installer le package neutron →`openstack-neutron-linuxbridge ebtables ipset`

en éditant le fichier "neutron.conf" selon la documentation officielle de Openstack.

Cinder (gére le stockage en block "volumes" et l'attacher à une machine virtuelle)

Controller node

1/ Créer une base de données sql intitulé « cinder » dans la machine Controller et créer un user en lui accordant les privilèges sur la base de données.

2/ Créer un utilisateur cinder→`openstack user create --domain <domain_name> --password-prompt cinder`

3/ Ajouter le role « admin » au projet et au user cinder →`openstack role add --project<project_name> --user cinder admin`

4/ Créer un service cinder →`openstack service create --name cinder(name) volume(type)`

5/ Créer des volume API EndPoints

```
openstack endpoint create --region RegionOne volume public
http://controller:8776/v2/%\(project_id\)s
```

```
openstack endpoint create --region RegionOne volume internal
http://controller:8776/v2/%\(project_id\)s
```

```
openstack endpoint create --region RegionOne \ volume admin
http://controller:8776/v2/%\(project_id\)s
```

6/ installer openstack cinder →`openstack-cinder`

et éditer le fichier "cinder.conf" selon la documentation officielle de Openstack.

7/ Ajouter la section [cinder] dans le fichier « nova.conf » pour que le compute utilise le stockage en bloc

```
[cinder]
```

```
os_region_name = RegionOne
```

Block node

1/ ajouter un disk de 60 GB→`/dev/sdb`.

2/ installer le package LVM →`lvm2 device-mapper-persistent-data`

3/ Créer un volume logique à partir de disk physique →`pvcreate /dev/sdb`

4/ Créer un volume group intitulé "cinder-volumes" →`vgcreate cinder-volumes /dev/sdb`

5/ installer le package openstack cinder →`openstack-cinder targetcli python-keystone`

et éditer le fichier "cinder.conf" selon la documentation officielle de Openstack.

Swift (gére le stockage des données non structurées "blobs" tel que: images, texte etc.)

Controller node

1/ pour le Object Storage on s'intéresse à utiliser SQLite au lieu MariDB.

2/ Créer un utilisateur swift→`openstack user create --domain <domain_name> --password-prompt swift`

3/ Ajouter le role « admin » au projet et au user swift →`openstack role add --project<project_name> --user swift admin`

4/ Créer un service swift →`openstack service create --name swift(name) object-store(type)`

5/ Créer des object storage API EndPoints

```
openstack endpoint create --region RegionOne object-store public http://controller:8080/v1/AUTH_%\(project_id\)s
```

```
openstack endpoint create --region RegionOne object-store internal http://controller:8080/v1/AUTH_%\(project_id\)s
```

```
openstack endpoint create --region RegionOne object-store admin http://controller:8080/v1
```

6/ installer les packages openstack swift

```
openstack-swift-proxy python-swiftclient \ python-keystoneclient python-keystonemiddleware \ memcached
```

7/ Obtenir le fichier de configuration de service proxy à partir du repo de Object Storage (source https://opendev.org)

```
curl -o /etc/swift/proxy-server.conf https://opendev.org/openstack/swift/raw/branch/master/etc/proxy-server.conf-sample
```

et éditer le fichier "proxy-server.conf" selon la documentation officielle de Openstack.

Object node1 & Object node2

Ps : ajouter deux disk de 60 GB chacun→`/dev/sdb et /dev/sdc`

1/ installer les packages nécessaire pour supporter le système de fichier XFS en assurant la synchronisation distante des fichiers →`xfsprogs rsync`.

2/ Formater les deux disks en XFS.

```
mkfs.xfs /dev/sdb
```

```
mkfs.xfs /dev/sdc
```

3/ Créer deux point de mount.

```
mkdir -p /srv/node/sdb
```

```
mkdir -p /srv/node/sdc
```

4/ mounter les deux disks.

```
mount /srv/node/sdb
```

```
mount /srv/node/sdc
```

5/ éditer le fichier "rsyncd.conf" selon la documentation officielle de Openstack.

6/ Installer les packages openstack account, container et object

```
openstack-swift-account openstack-swift-container openstack-swift-object
```

7/ Obtenir le fichier de configuration de service acount, container et object à partir du repo de Object Storage (source https://opendev.org)

```
curl -o /etc/swift/account-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/account-server.conf-sample
```

```
curl -o /etc/swift/container-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/container-server.conf-sample
```

```
curl -o /etc/swift/object-server.conf
https://opendev.org/openstack/swift/raw/branch/master/etc/object-server.conf-sample
```

8/ éditer les fichier "account-server.conf", "container-server.conf" et "object-server.conf" selon la documentation officielle de Openstack.

Controller node

1/ Créer des rings pour account, container et object à traver the ring builder afin de créer des fichiers de configuration que chaque node utilise pour déterminer et déployer l'architecture de stockage.

2/ Distribuer les fichiers de configuration rings en format .gz aux object nodes en utilisant le service scp.

3/ Obtenir le fichier "swift.conf" à partir du repo de Object Storage (source https://opendev.org)

```
curl -o /etc/swift/swift.conf \
https://opendev.org/openstack/swift/raw/branch/master/etc/swift.conf-sample
```

en l'éditant selon la documentation officielle de Openstack.

4/ Distribuer le fichier "swift.conf" aux object nodes en utilisant le service scp

5/ Vérifier la bonne configuration de service swift en tapant la commande `swift stat`