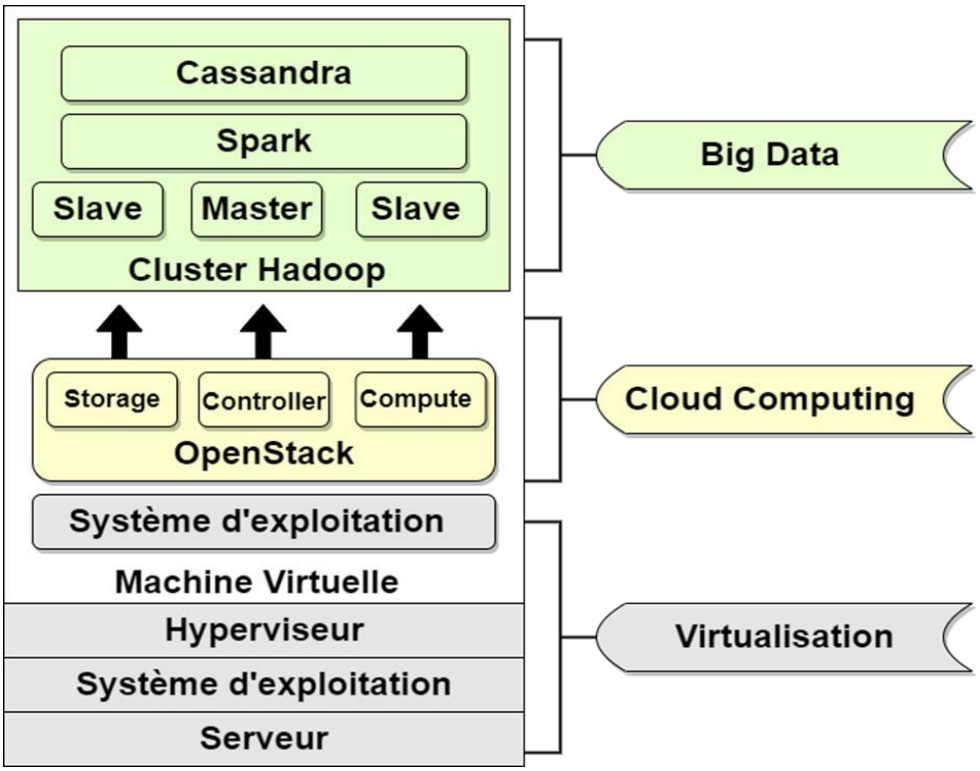
Guide d'implémentation d'une solution Big-data dans une plate-forme Iaas basée sur Openstack

Ce guide décrit la méthode d'implémentation des différents composants d'Openstack ainsi de l'implémentation d'une plate-forme Big Data basée sur Hadoop et Spark.



Hlel Firas

Dehech Med Firas

Sommaire

I. OpenStack

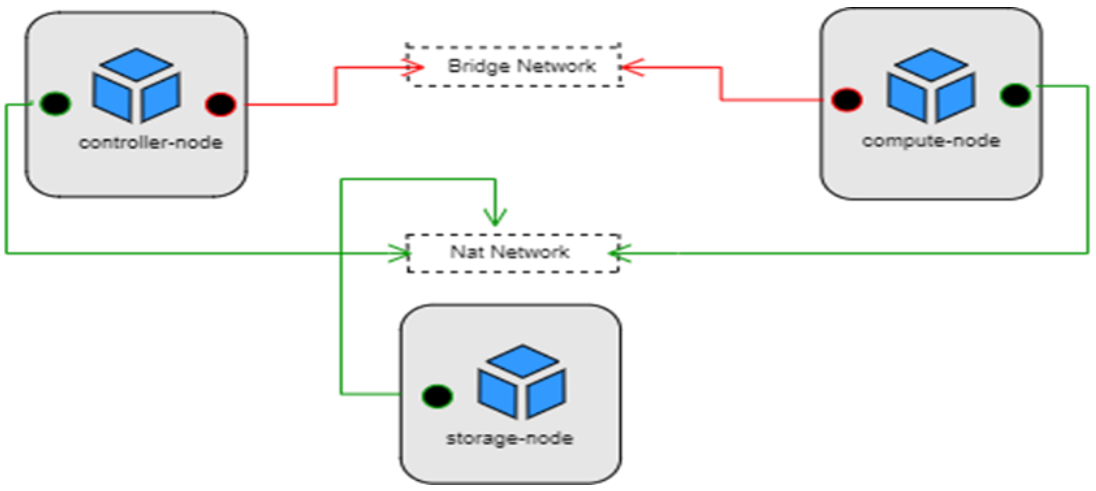
1. Architecture OpenStack :

Openstack est une plate-forme open source composée par plusieurs services, chacun d'eux offre un service particulier.

Les services qu'on a installés :

- Keystone : Fournit un service d'authentification et d'autorisation pour les autres services.
- Glance : Permet de stocker et récupérer les images des SE.
- Placement : Permet la communication entre le serveur web (Apache, Nginx) et des autres services.
- Nova : Gère le cycle de vie des machines virtuel dans l'environnement Openstack (la création, la suppression, la mise hors service).
- Neutron : Fournit le réseau dans l'environnement Openstack.
- Cinder : Service de stockage sur Openstack, gère les volumes des instances créer.
- Horizon : Fournit une interface web pour interagir avec les services.
- Heat : Permet l'orchestration entre les applications dans le Cloud.

On a utilisé une architecture de trois nœuds comme le montre la figure ci-dessous.



Nœud de contrôleur (controller-node) : le nœud de contrôleur est une Vm exécute les services suivants : Horizon, une partie de service de gestion des instances(nova), Neutron, glance, keystone et Heat.

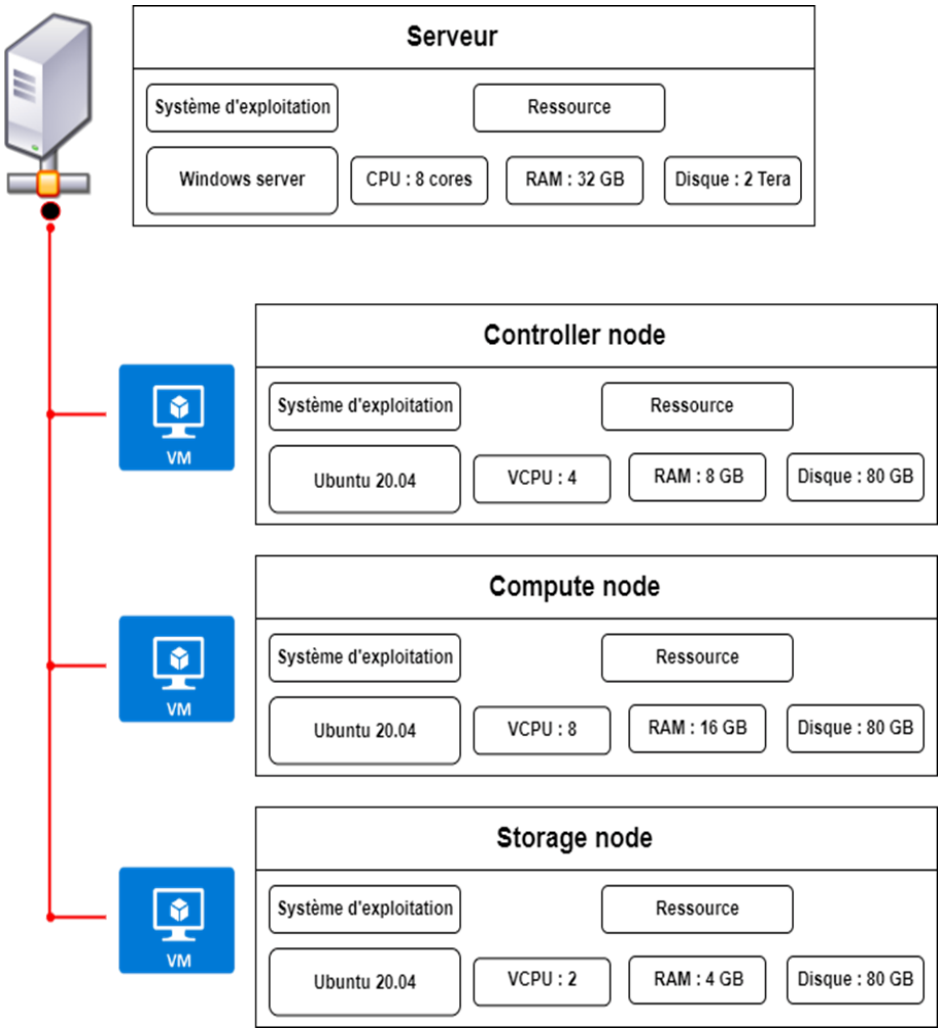
Nœud de calcul(compute-node) : ce nœud permet la gestion des instances.

Nœud de stockage(storage-node) : ce nœud inclut le service de stockage.

2. Mise en place de l'environnement :

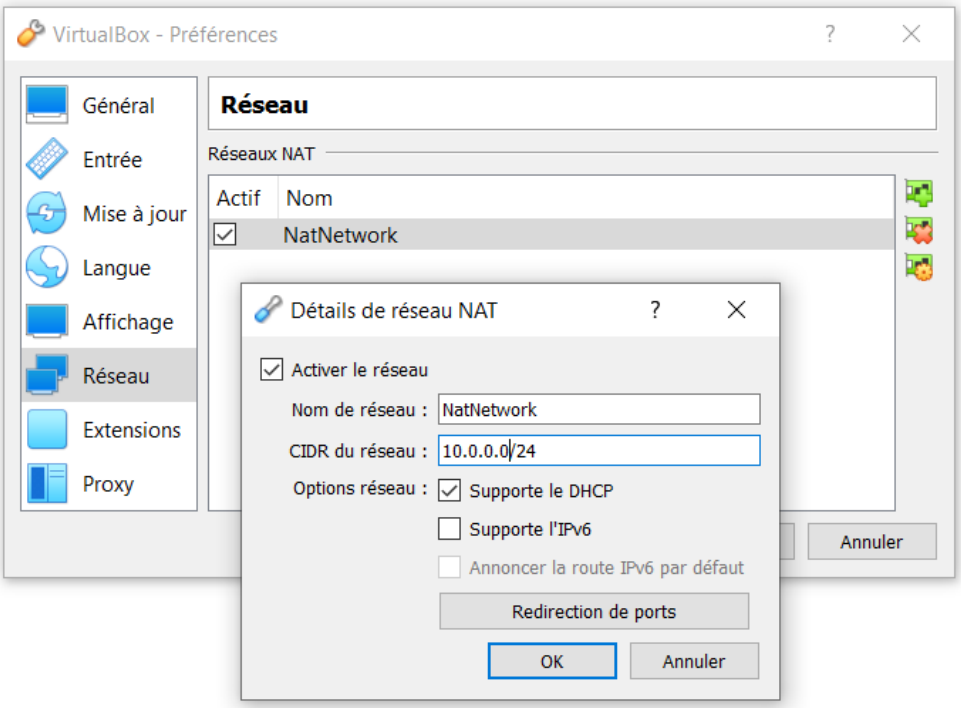
Exigence matérielle :

- Serveur
- Hyperviseur Oracle VirtualBox
- SE Ubuntu 20.04



Configuration Réseau :

On a créé un réseau Nat Network sur VirtualBox pour faire connecter les VMs entre eux et pour fournir un accès internet.

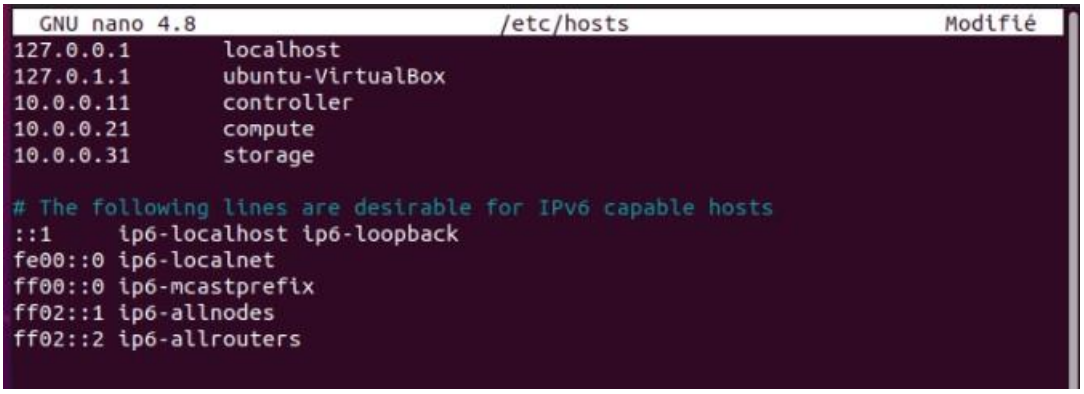


Adressage :

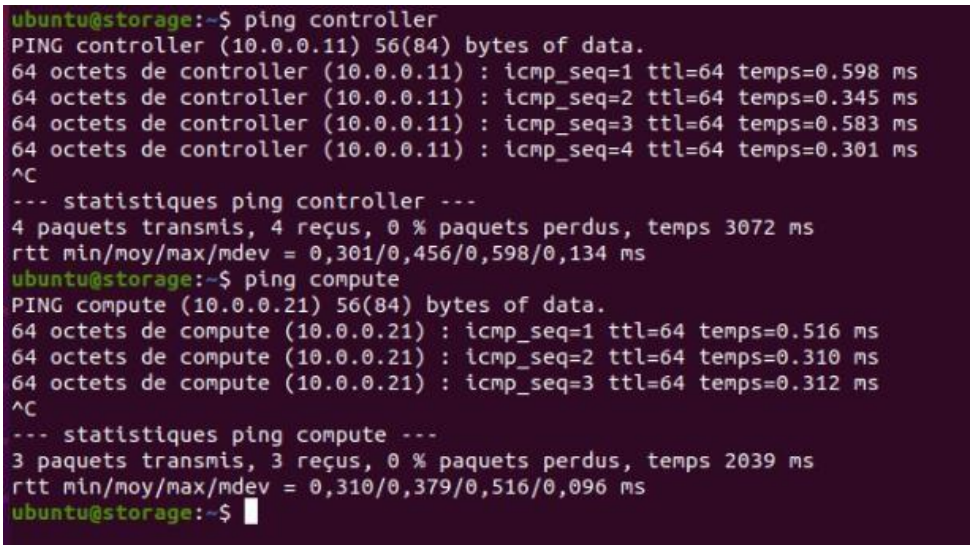
VM	Interface	Adresse
Controller-node	Bridge NatNetwork	Automatique 10.0.0.11/24
Compute-node	Bridge NatNetwork	Automatique 10.0.0.21/24
Storage-node	NatNetwork	10.0.0.31/24

Sur chacun des trois nœuds, il est recommandé de :

- Effectuer l'adressage approprié.
- Configurer la résolution de noms en éditant les deux fichiers /etc/hosts et /etc/hostname.
- Redémarrer les machines.
- Tester la connectivité.



Vérification de connectivité



Synchronisation d'horloge à l'aide de Network Time Protocol :

Nous utilisons le protocole NTP pour synchroniser l'horloge entre les nœuds. Le nœud de contrôleur synchronisé avec les serveurs NTP de ubuntu.

Les autres nœuds seront synchronisés avec le nœud contrôleur.

Sur tous les nœuds, nous installons le service chrony a l'aide de la commande :

 # apt install chrony

Sur le nœud de controleur, nous editons le fichiers **/etc/chrony/chrony.conf** et nous ajoutons les lignes suivantes :

Puis nous redemarrons le service NTP avec la commande :

 # service chrony restart

Pour les autres nœuds :

Installation des packages d'openstack :

La version openstack victoria est la plus stable et récente.

#add-apt-repository cloud-archive : victoria

- ⇨ **Sur tous les nœuds.**

Installation de client Openstack :

#apt-get install python-openstackclient

- ⇨ **Sur tous les nœuds.**

Installation de Base de données SQL :

La plupart des services OpenStack utilisent une base de données SQL pour stocker des informations. La base de données s'exécute généralement sur le nœud du contrôleur.

 # apt install mariadb-server python3-pymysql

Configuration de la base de données :

Ensuite, nous créons le fichier /etc/mysql/mariadb.conf.d/99-openstack.cnf et en spécifiant l'adresses IP du nœud contrôleur pour autoriser l'accès aux autres nœuds.

Pour sécuriser la base de données en exécutant la commande suivante et choisissons un mot de passe :

#mysql_secure_installation

Redémarrage de la base de données :

#service mysql restart

Installation de la file d'attente RabbitMQ :

OpenStack utilise une file d'attente de messages pour coordonner les opérations et les informations d'état entre les services.

#apt install rabbitmq-server

- ⇨ **Sur le nœud contrôleur**

Ensuite, nous créons un utilisateur d'OpenStack nommé « openstack ». Puis, nous autorisons l'utilisateur à configurer, écrire et lire avec ces deux instructions sans oublier le redémarrage du service.

```
root@controller:/home/ubuntu# rabbitmqctl add_user openstack 123456
root@controller:/home/ubuntu# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
root@controller:/home/ubuntu#
```

Installation de Memcached :

Le mécanisme d'authentification du service d'identité pour les services utilise Memcached pour mettre en cache les jetons.

#apt install memcached python3-memcache

Ensuite, nous éditons le fichier **/etc/memcached.conf** en y ajoutant l'adresse IP du noeud contrôleur. Puis nous redémarrons le service.

```
 GNU nano 4.8                                  /etc/memcached.conf
# This parameter is one of the only security measures that memcached has, so make sure
# it's listening on a firewalled interface.
-l 10.0.0.11
```

#service memcached restart

- ⇨ **Sur le nœud contrôleur.**

Installation de l'ETCD :

Les services OpenStack utilisent **Etcd**. C'est un fournisseur de clé-valeur distribuée permettant le verrouillage de clé distribuée, le stockage de la configuration, le suivi de la qualité de service.

#apt install etcd

Ensuite, nous modifions le fichier **/etc/default/etcd** en ajoutons l'adresse IP du nœud contrôleur.

```
 GNU nano 4.8                                  /etc/default/etcd

ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://10.0.0.11:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.0.11:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.0.11:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://10.0.0.11:2379"
```

Redémarrage de service ETCD :

#systemctl enable etcd

#systemctl restart etcd

3. Services Openstack

Service Keystone

Avant d'installer et de configurer le service d'identité, nous créons une base de données nommées keystone avec les privilèges nécessaires.

```
root@controller:/home/ubuntu# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 44
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE keystone;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)
```

D'abord, nous installons le package de service keystone :

#sudo apt install keystone

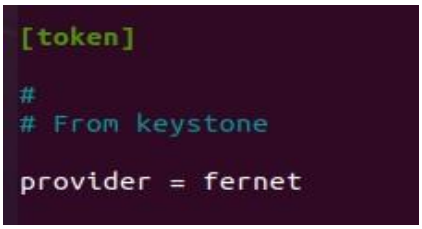
Puis, nous éditons le fichiers **/etc/keystone/keystone.conf**

- A la section **[database],** on ajoute la ligne suivante pour configurer l'accès a la base de données.

```
[database]
#connection = sqlite:////var/lib/keystone/keystone.db
connection = mysql+pymysql://keystone:123456@controller/keystone
```

- A la section **[token],** on ajoute le fournisseur de jetons **Fernet**

Les jetons Fernet sont des jetons contenant des messages et des données d'authentification et d'autorisation. Nous choisissons ce type de jeton car ils sont signés et cryptés avant d'être remis aux utilisateurs.



Après les modifications, nous remplissons la base de données du service d'identité et nous initialisons les référentiels de clés Fernet avec ces instructions :

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

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

 # keystone-manage credential_setup --keystone-user keystone --keystone-group keystone

Une fois que Keystone est déployé et configuré, il doit être rempli avec certaines données initiales avant de pouvoir être utilisé. Ce processus est appelé l'amorçage. Pour ceci nous initialisons l'URL de l'admin, l'URL interne, l'URL public et la région de Keystone :

**# keystone-manage bootstrap --bootstrap-password 123456 **

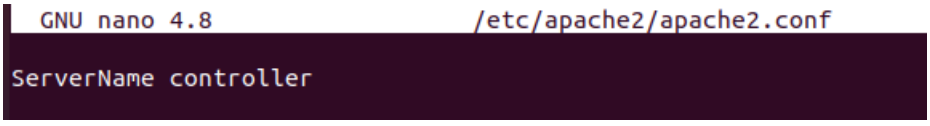
**--bootstrap-admin-url http://controller:5000/v3/ **

**--bootstrap-internal-url http://controller:5000/v3/ **

**--bootstrap-public-url http://controller:5000/v3/ **

--bootstrap-region-id RegionOne

Après, nous éditons le fichiers /etc/apache2/apache2.conf du serveur HTTP Apache.



Puis, nous configurons le compte de l'administrateur en exécutant les commandes suivantes :

```
openstack@openstack-VirtualBox:~$
openstack@openstack-VirtualBox:~$ export OS_USERNAME=admin
openstack@openstack-VirtualBox:~$ export OS_PASSWORD=123456
openstack@openstack-VirtualBox:~$ export OS_PROJECT_NAME=admin
openstack@openstack-VirtualBox:~$ export OS_USER_DOMAIN_NAME=Default
openstack@openstack-VirtualBox:~$ export OS_PROJECT_DOMAIN_NAME=Default
openstack@openstack-VirtualBox:~$ export OS_AUTH_URL=http://controller:5000/v3
openstack@openstack-VirtualBox:~$ export OS_IDENTITY_API_VERSION=3
openstack@openstack-VirtualBox:~$
openstack@openstack-VirtualBox:~$
```

Le service d'authentification utilise une combinaison de domaines, de projets, des utilisateurs et des rôles. Le domaine **default** existe grâce à l'amorçage de Keystone, une manière formelle de créer un nouveau domaine serait :

$ openstack domain create --description "An Example Domain" example

Dans ce domaine, nous créons le projet nommé **Service Project** (ce projet contenant un utilisateur unique pour chaque service dans l'environnement OpenStack) avec la commande suivante :

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

Les tâches régulières (non-admin) doivent utiliser un projet et un utilisateur non privilégiés. À titre d'exemple, ce guide crée le projet myproject et l'utilisateur myuser.

Nous créons le projet myproject :

$ openstack project create --domain default --description "Demo Project" myproject

Nous créons l'utilisateur myuser :

$ openstack user create --domain default --password-prompt myuser

Nous créons le rôle myrole :

$ openstack role create myrole

Nous ajoutons le rôle myrole au projet myproject et à l'utilisateur myuser :

$ openstack role add --project myproject --user myuser myrole

Pour faciliter les autres tâches des autres services, nous préparons des scripts d'environnement client pour les projets admin et demo ainsi que les utilisateurs.

Nous créons les deux fichiers admin-openrc et demo-openrc . Nous y configurons le nom de projet, le nom d'utilisateur, le nom de domaine, le mot de passe et l'URL d'authentification.

```
  GNU nano 4.8                                         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=123456
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

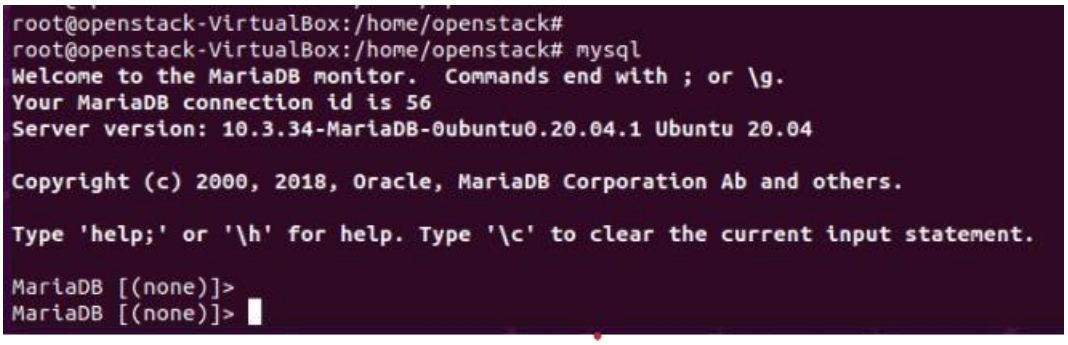
```
  GNU nano 4.8                                         demo-openrc

export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=myproject
export OS_USERNAME=myuser
export OS_PASSWORD=123456
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

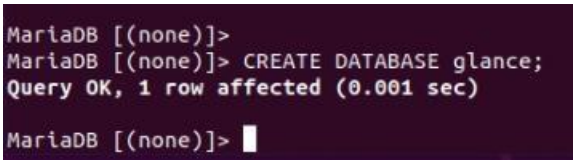
Enfin, nous vérifions le fonctionnement du service d'identité avant d'installer les autres services en tant qu'utilisateur **admin ou demo**, nous demandons un jeton d'authentification avec la commande suivante :

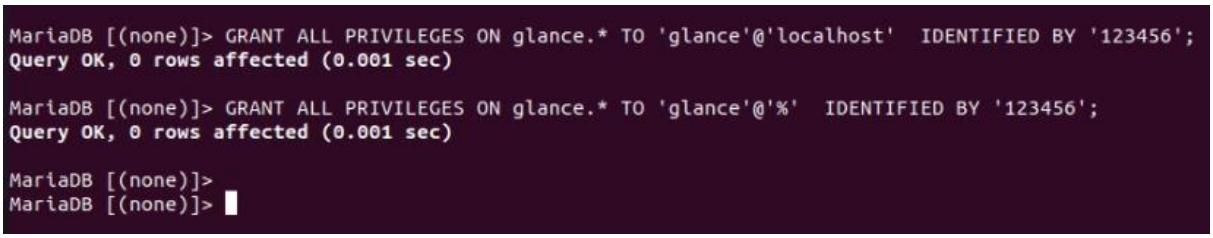
```
root@controller:/home/openstack#
root@controller:/home/openstack# . admin-openrc
root@controller:/home/openstack# openstack token issue
+------------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| Field      | Value
                                                                          |
+------------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| expires    | 2022-05-16T22:47:41+0000
                                                                          |
| id         | gAAAAABigsZ9Lp8seypkzteUbzq8mjPGeWq1m72GvLM1SX686cwSzKujCDgddxmlhQcYfR7uOGY2UICXoo_B6WwTNnRs
oYuRqKrdq5m0s1rpu225IF2Rz8cmXTlr3cfgQKSfYbv3RQ-mPOUEZgWnPvBR9Aa5vQJxNHe0M0GtbhgMGjyOAcAG_nk |
| project_id | 6df4cfe6ebd44fa5b567185b3a7e9278
                                                                          |
| user_id    | 073b5535f3b84230a92bd2ab471d8c08
                                                                          |
+------------+--------------------------------------------------------------------------------
--------------------------------------------------------------------------+
root@controller:/home/openstack#
root@controller:/home/openstack#
root@controller:/home/openstack#
```

Service Glance

Sur le noeud contrôleur, nous devons créer en premier lieu une base de données des informations d'identification du service, un utilisateur nommé **glance** et les points de terminaison d'API :







D’abord, nous créons un utilisateur nommé **glance** avec cette commande :

$ openstack user create --domain default --password-prompt glance

Puis, nous ajoutons l'utilisateur **glance** avec le rôle **admin** au projet **service**

$ openstack role add --project service --user glance admin

Nous créons par la suite une entité de service **glance.**

$ openstack service create --name glance --description "OpenStack Image" image

Enfin, nous concevons les points de terminaison de l'API du service **glance** (public, interne, admin).

$ 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
Nous engageons, par la suite, l’installation du package **Glance** avec la commande :

 # apt install glance

Puis, nous éditons le fichier **/etc/glance/glance-api.conf** pour effectuer les modifications suivantes :

- Au la section **[database]**, nous autorisons l'accès à la base de données.

```
[database]
#connection = sqlite:////var/lib/glance/glance.sqlite
connection = mysql+pymysql://glance:123456@controller/glance
backend = sqlalchemy

#
# From oslo.db
#
```

- Aux sections **[keystone_authtoken]** et **[paste_deploy]**, nous configurons l'accès au service d'identité.

```
[paste_deploy]


# Provide a string value representing the appropriate deployment
# flavor used in the server application pipleline. This is typically
# the partial name of a pipeline in the paste configuration file with
# the service name removed.
# For example, if your paste section name in the paste configuration
# file is [pipeline:glance-api-keystone], set ``flavor`` to
# ``keystone``.
# Possible values:
#     * String value representing a partial pipeline name.
# Related Options:
#     * config_file
#  (string value)
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
flavor = keystone
```

```
[keystone_authtoken]

www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = 123456
```

- Au niveau de la section **[glance_store]**, nous indiquons le fournisseur du système du fichier local et l'emplacement des fichiers image.

```
[glance_store]

stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/

#
# From glance.multi_store
#
```

Par la suite, nous éditons le fichier **/etc/glance/glance-registry.conf** et nous y modifions les sections suivantes :

Après ces modifications, nous remplissons la base de données du glance et nous redémarrons les différents services de glance avec la commande :

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

 # service glance-api restart

Pour vérifier le fonctionnement du service Image on a besoin d'installer CirrOS , une petite image Linux qui vous aide à tester votre déploiement OpenStack.

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

Chargez l'image dans le service Image à l'aide du format de disque QCOW2, du format de conteneur nu et de la visibilité publique afin que tous les projets puissent y accéder :

**$ glance image-create --name "cirros" **

**--file cirros-0.4.0-x86_64-disk.img **

**--disk-format qcow2 --container-format bare **

--visibility=public

Vérifier si l'image a été bien charger sous glance :

Service Placement

Sur le nœud contrôleur, nous devons créer en premier lieu une base de données des informations d'identification du service, un utilisateur nommé **placement** et les points de terminaison d'API :

```
root@openstack-VirtualBox:/home/openstack# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 61
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

```
MariaDB [(none)]> CREATE DATABASE placement;
Query OK, 1 row affected (0.021 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```

Nous créons un utilisateur du service de placement :

$ openstack user create --domain default --password-prompt placement

Nous ajoutons l'utilisateur Placement au projet de service avec le rôle d'administrateur :

$ openstack role add --project service --user placement admin

Nous créons l'entrée de l'API d'emplacement dans le catalogue de services :

$ openstack service create --name placement --description "Placement API" placement

Nous créons les points de terminaison du service de l'API Placement :

**$ openstack endpoint create --region RegionOne **

placement public http://controller:8778

**$ openstack endpoint create --region RegionOne **

placement internal http://controller:8778

**$ openstack endpoint create --region RegionOne **

placement admin http://controller:8778

Nous engageons, par la suite, l'installation du package **Placement** avec la commande :

 # apt install placement

Nous modifions le fichier **/etc/placement/placement.conf** et effectuez les actions suivantes :

- Dans la section **[placement_database]**, configurons l'accès à la base de données :

```
[placement_database]
#connection = sqlite:////var/lib/placement/placement.sqlite
connection = mysql+pymysql://placement:123456@controller/placement
```

- Aux sections **[keystone_authtoken]** et **[paste_deploy]**, nous configurons l'accès au service d'identité.

```
[api]
#
# Options under this group are used to define Placement API.#
# From placement.conf
# This determines the strategy to use for authentication: keystone or noauth2.
# 'noauth2' is designed for testing only, as it does no actual credential
# checking. 'noauth2' provides administrative credentials only if 'admin' is
# specified as the username.
#  (string value)
# Possible values:
# keystone - <No description provided>
# noauth2 - <No description provided>
auth_strategy = keystone
```

```
[keystone_authtoken]

auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = 123456
```

Nous remplirons la base de données placement avec la commande :

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

Finaliser l'installation par redémarrage d'apache

 # service apache2 restart

Service Nova

Sur le nœud contrôleur, nous créons des bases de données pour **nova**, **nova-api** et **nova-cell0** en y accordant les privilèges nécessaires :

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 74
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

```
MariaDB [(none)]> CREATE DATABASE nova_api;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE DATABASE nova;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE DATABASE nova_cell0;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)
```

Nova utilise trois types de bases de données :

- Une **« base de données API »** utilisée principalement par nova-api pour suivre les informations d'emplacement relatives aux instances et l'emplacement temporaire pour les instances en cours de création mais non encore planifiées.
- Une **« base de données de cellules »** utilisée par les services de l'API et du calcul contenant la majorité des informations sur les instances.

- Une **« base de données cell0 »** qui ressemble à la base de données de cellules, mais ne contient que les instances dont la planification a échoué.

Ensuite, nous créons un utilisateur nommé **nova** avec ces commandes :

$ openstack user create --domain default --password-prompt nova

Puis, nous ajoutons le rôle admin à l'utilisateur nova :

$ openstack role add --project service --user nova admin

Nous créons par la suite une entité de service nova :

**$ openstack service create --name nova **
--description "OpenStack Compute" compute

Enfin, nous créons les points de terminaison de l'API (public, interne, admin)

**$ 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 RegionOne **
compute admin http://controller:8774/v2.1

Nous passons maintenant à l'installation du service nova avec cette commande

 # apt install nova-api nova-conductor nova-novncproxy nova-scheduler

Ensuite, nous éditons le fichier **/etc/nova/nova.conf** pour effectuer les changements suivants :

- Aux sections **[api_database]** et **[database]**, nous autorisons l'accès à la base de données.

```
[api_database]
#connection = sqlite:////var/lib/nova/nova_api.sqlite
connection = mysql+pymysql://nova:123456@controller/nova_api
```

```
[database]
#connection = sqlite:////var/lib/nova/nova.sqlite
connection = mysql+pymysql://nova:123456@controller/nova
```

- Au niveau de la section **[DEFAULT]**, nous autorisons l'accès à la file d'attente de messages RabbitMQ, nous déclarons l'adresse IP de l'interface de gestion du noeud contrôleur.

```
[DEFAULT]
#log_dir = /var/log/nova
my_ip = 10.0.0.11
transport_url = rabbit://openstack:123456@controller:5672/
lock_path = /var/lock/nova
state_path = /var/lib/nova
```

- Aux sections **[api]** et **[keystone_authtoken]**, nous autorisons l'accès au service d'identité.

```
[api]

auth_strategy = keystone
```

```
[keystone_authtoken]

www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = 123456
```

- Au niveau de la section **[vnc]**, nous configurons le proxy **vnc** pour activer l'accès console des instances en déclarant l'adresse IP de l'interface de gestion du noeud contrôleur.

```
[vnc]

enabled = true
server_listen = $my_ip
server_proxyclient_address = $my_ip
```

- A la section **[glance]**, nous ajoutons l'emplacement de l'API du service d'image.

```
[glance]

api_servers = http://controller:9292
```

- Au niveau de la section **[oslo_concurrency]**, nous déclarons le chemin de verrouillage.

```
[oslo_concurrency]

lock_path = /var/lib/nova/tmp
```

- A la section **[placement]**, nous indiquons l'API de placement.

```
[placement]

region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = 123456
```

En effet, nous remplissons la base de données du service **nova-api** avec la commande

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

Puis, nous enregistrons la base de données

 # su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova

Par la suite nous créons la **cellule cell1** avec ces commandes :

 # su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova

Enfin, nous remplissons la base de données **nova**

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

Enfin nous redémarrons les services

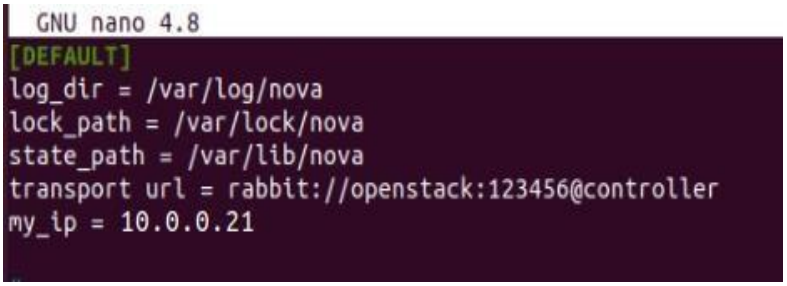
```
root@openstack-VirtualBox:/home/openstack# service nova-api restart
root@openstack-VirtualBox:/home/openstack# service nova-scheduler restart
root@openstack-VirtualBox:/home/openstack# service nova-conductor restart
root@openstack-VirtualBox:/home/openstack# service nova-novncproxy restart
root@openstack-VirtualBox:/home/openstack#
```

Sur le noeud de calcul, nous effectuons les étapes suivantes tout en commençant par l'installation du package avec la commande :

 # apt install nova-compute

Ensuite, nous éditons le fichier **/etc/nova/nova.conf** en effectuant les actions ci-dessous :

- A la section **[DEFAULT]**, nous autorisons l'accès à la file d'attente de messages RabbitMQ, l'option my_ip sert à déclarer l'adresse IP de l'interface de gestion du noeud compute.



- Aux sections **[api]** et **[keystone_authtoken],** nous autorisons l'accès au service d'identité. (Même étape sur le nœud contrôleur)
- A la section **[vnc],** nous autorisons l'accès à la console distante

- Au niveau de la section **[glance**], nous déclarons l'emplacement de l'API du service d'image. (Même étape sur le nœud contrôleur)
- A la section **[oslo_concurrency],** nous indiquons le chemin de verrouillage. (Même étape sur le nœud contrôleur)
- Au niveau de la section [placement], nous ajoutons l'API de placement (Même étape sur le nœud contrôleur)

Enfin, nous devons déterminer si notre noeud de calcul prend en charge l'accélération matérielle pour les machines virtuelles avec la commande suivant :

 # egrep -c '(vmx| svm)' /proc/cpuinfo

- Si cette commande renvoie la valeur « 0 » ; c'est-à-dire notre noeud de calcul ne prend pas en charge l'accélération matérielle et nous devons configurer libvirt pour utiliser QEMU au lieu de KVM. Il faut donc éditer la section **[libvirt]** dans le fichier **/etc/nova/nova-compute.conf** comme suit :

```
  GNU nano 4.8
[DEFAULT]
compute_driver=libvirt.LibvirtDriver
[libvirt]
virt_type=qemu
```

- Si elle renvoie la valeur « 1 », notre noeud de calcul prend en charge l'accélération matérielle.

Après avoir terminé l'installation et la configuration du service Nova on doit maintenant vérifier son fonctionnement :

```
openstack@openstack-VirtualBox:~$
openstack@openstack-VirtualBox:~$ openstack compute service list
+----+----------------+---------------------+----------+---------+-------+----------------------------+
| ID | Binary         | Host                | Zone     | Status  | State | Updated At                 |
+----+----------------+---------------------+----------+---------+-------+----------------------------+
|  1 | nova-scheduler | openstack-VirtualBox | internal | enabled | up    | 2022-05-12T16:08:16.000000 |
|  2 | nova-conductor | openstack-VirtualBox | internal | enabled | up    | 2022-05-12T16:08:17.000000 |
|  3 | nova-compute   | openstack-VirtualBox | nova     | enabled | up    | 2022-05-12T16:08:15.000000 |
|  7 | nova-compute   | compute             | nova     | enabled | up    | 2022-05-12T16:08:11.000000 |
+----+----------------+---------------------+----------+---------+-------+----------------------------+
openstack@openstack-VirtualBox:~$
```

Service Neutron

La configuration du service neutron s'effectue sur le noeud contrôleur et le noeud de calcul.

Sur le noeud contrôleur, nous procédons comme suit :

Nous devons créer en premier lieu une base de données des informations d'identification de service et des points de terminaison d'API

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1353
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE neutron;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.017 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]>
```

Ensuite, nous créons un utilisateur nommé neutron avec ces commandes :

$ openstack user create --domain default --password-prompt neutron

Puis, nous ajoutons le rôle admin à l'utilisateur neutron avec cette commande :

$ openstack role add --project service --user neutron admin

Nous créons par la suite une entité du service neutron

**$ openstack service create --name neutron **

--description "OpenStack Networking" network

Enfin, nous créons les points de terminaison de l'API du service neutron (public, interne, admin)

**$ 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

NB : pour déployer le service de mise en réseau neutron, nous devons choisir l'une des deux ces options :

L'option 1 : le réseau **provider** qui déploie une architecture qui attache les instances aux réseaux externes uniquement. Nous ne pouvons pas créer des réseaux privés.
Seul l'utilisateur admin ou tout autre utilisateur à privilège peut gérer les réseaux provider.
L'option 2 : le réseau **self-service** inclut les avantages de l'option 1 avec des services de couche-3 qui supporte l'attachement des instances à des réseaux privés.
L'utilisateur sans privilège peut gérer des réseaux self-service (réseaux privés) qui possèdent des routeurs. Ces derniers fournissent la connectivité entre les réseaux self-service et les réseaux provider.

Nous choisissons l'option 2.
Pour configurer l'option 2, nous suivons les instructions suivantes :

D'abord, nous installons le paquet avec la commande :

**# apt install neutron-server neutron-plugin-ml2 **

**neutron-linuxbridge-agent neutron-l3-agent neutron-dhcp-agent **

neutron-metadata-agent

Puis, nous éditons le fichier /etc/neutron/neutron.conf en effectuant les actions ci-dessous :

- A la section **[database]**, nous autorisons l'accès à la base de données.

```
[database]
#connection = sqlite:////var/lib/neutron/neutron.sqlite
connection = mysql+pymysql://neutron:123456@controller/neutron
```

- Au niveau de la section **[DEFAULT],** nous activons le plug-in du service de routeur Modular Layer 2 (ML2). En outre, nous autorisons l'accès à la file d'attente de messages RabbitMQ.

```
[DEFAULT]
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = true
transport_url = rabbit://openstack:123456@controller
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
```

- A la section **[keystone_authtoken]**, nous autorisons l'accès au service d'identité.

```
[keystone_authtoken]

www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = 123456
```

- A la section **[nova]**, nous déclarons le réseau pour informer le service nova des modifications de la topologie du réseau.

```
[nova]

auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = 123456
```

- Au niveau de la section **[oslo_concurrency]**, nous indiquons le chemin de verrouillage.

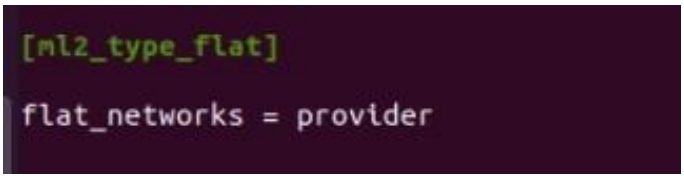
```
[oslo_concurrency]

lock_path = /var/lib/neutron/tmp
```

Pour configurer le plug-in ML2, nous éditons le fichier **/etc/neutron/plugins /ml2/ml2_conf.ini** :

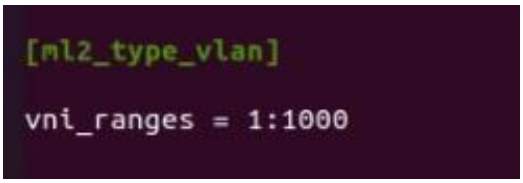
- A la section **[ml2]**, nous activons les réseaux de self-service flat1, VLAN2, VXLAN3, le pont Linux, les mécanismes de population de couche 2 et le pilote d'extension de sécurité du port.

```
[ml2]

type_drivers = flat,vlan,vxlan
tenant_network_types = vxlan
mechanism_drivers = linuxbridge,l2population
extension_drivers = port_security
flat_networks = provider
```

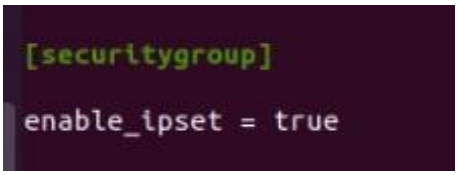
- Au niveau de la section **[ml2_type_flat]**, nous configurons notre réseau virtuel en tant que réseau provider



- A la section **[ml2_type_vxlan]**, nous identifions la plage d'identificateurs de réseau VXLAN pour les réseaux self-service.

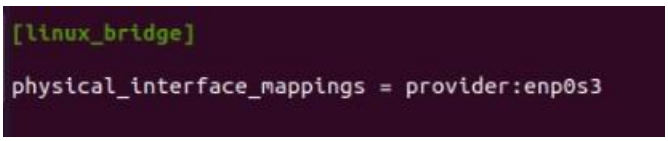


- Au niveau de la section **[securitygroup]**, nous activons ipset pour améliorer l'efficacité des règles du groupe de sécurité.

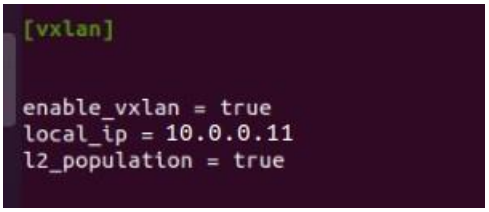


L'agent pont Linux crée une infrastructure de réseau virtuel pour les instances de couche 2 (routage et commutation) et il gère aussi les groupes de sécurité. Afin de configurer cet agent, nous éditons le fichier **/etc/neutron/plugins/ml2/linuxbridge_agent.ini** en effectuant les modifications suivantes :

- A la section **[linux_bridge]**, nous mappons notre réseau virtuel sur l'interface réseau physique de notre noeud.

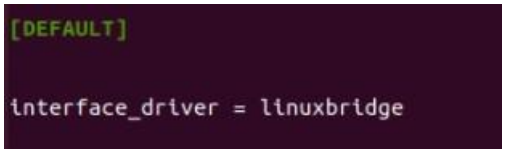


- Au niveau de la section **[vxlan]**, nous activons les réseaux superposés VXLAN. De plus, nous déclarons l'adresse IP de l'interface réseau physique qui gère les réseaux superposés.



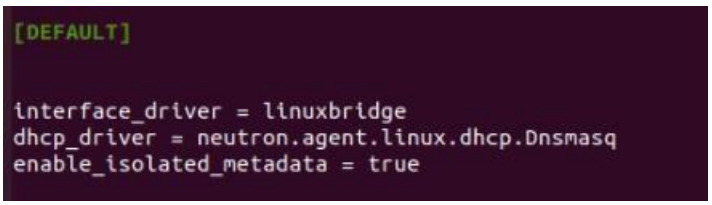
- A la section **[securitygroup]**, nous activons les groupes de sécurité et nous spécifions le pilote de pare-feu iptables du pont Linux.

L'agent de couche 3 (L3) fournit des services de routage pour les réseaux virtuels self-service. Pour configurer cet agent, nous éditons le fichier **/etc/neutron/l3_agent.ini** et nous ajoutons à la section **[DEFAULT]** le pilote d'interface de pont Linux et le pont de réseau externe.



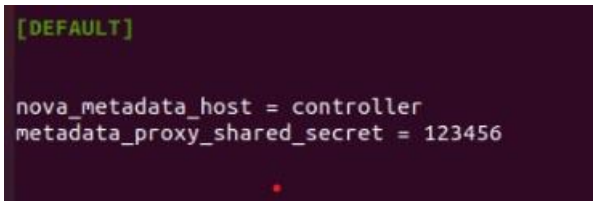
L'agent DHCP fournit les services DHCP pour les réseaux virtuels. Pour configurer cet agent DHCP, nous éditons le fichier **/etc/neutron/dhcp_agent.ini** :

- A la section **[DEFAULT]**, nous configurons le pilote d'interface de pont Linux, le pilote DHCP Dnsmasq (qui fournit les services DNS, DHCP, Bootstrap Protocol et TFTP pour un petit réseau) sans oublier d'activer les métadonnées isolées afin que les instances puissent accéder aux métadonnées sur le réseau.



Nous configurons l'agent de métadonnées par la suite. En effet, cet agent fournit des informations de configuration, telles que les informations d'identification aux instances. Pour configurer cet agent, nous éditons le fichier **/etc/neutron/metadata_agent.ini :**

- Au niveau la section **[DEFAULT]**, nous configurons l'hôte de métadonnées et le secret partagé.



Nous configurons ensuite le service nova afin d'activer le service de mise en réseau neutron en éditant le fichier **/etc/nova/nova.conf** :

- A la section **[neutron]**, nous activons les paramètres d'accès, le proxy de métadonnées

Enfin, nous remplissons la base de données et nous redémarrons le service nova API et les services de mise en réseau (neutron-server, neutron-linuxbridge-agent, neutron-dhcp-agent, metadata-agent, neutron-l3-agent)

**# 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

Sur le noeud de calcul, nous accomplissons les étapes suivantes tout en commençant par l'installation du package :

 # apt install neutron-linuxbridge-agent

Nous éditons, par la suite le fichier **/etc/neutron/neutron.conf** :

- A la section **[DEFAULT]**, nous autorisons l'accès à la file d'attente de messages RabbitMQ et l'accès au service d'identité.

```
[DEFAULT]
core_plugin = ml2
transport_url = rabbit://openstack:123456@controller
auth_strategy = keystone
```

- Au niveau de la section **[keystone_authtoken]**, nous autorisons l'accès au service d'identité. (même étape dans le nœud contrôleur)

- A la section **[oslo_concurrency]**, nous indiquons le chemin de verrouillage. (même étape dans le nœud contrôleur)

Nous configurons, par la suite, l'option de mise en réseau en éditant **/etc/neutron/plugins/ml2/linuxbridge_agent.ini**

- A la section **[linux_bridge]**, nous mappons notre réseau virtuel à l'interface physique de notre noeud (ens33).

```
[linux_bridge]

physical_interface_mappings = provider:enp0s3
```

- Au niveau de la section **[vxlan]**, nous activons les réseaux superposés VXLAN et le remplissage de la couche 2. De plus, nous déclarons l'adresse IP de l'interface réseau physique qui gère les réseaux superposés.

```
[vxlan]

enable_vxlan = true
local_ip = 10.0.0.21
l2_population = true
```

- A la section **[securitygroup]**, nous activons les groupes de sécurité et nous spécifions le pilote du pare-feu iptables du pont Linux.

```
[securitygroup]

enable_security_group = true
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
```

Puis, nous devons configurer le service nova pour utiliser le service de mise en réseau neutron. Pour cela, nous éditons le fichier **/etc/nova/nova.conf** et dans la section **[neutron]**, nous configurons les paramètres d'accès.

```
[neutron]

auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = 123456
```

Sur le nœud contrôleur redémarrer les services neutron

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# service nova-api restart
root@openstack-VirtualBox:/home/openstack# service neutron-server restart
root@openstack-VirtualBox:/home/openstack# service neutron-linuxbridge-agent restart
root@openstack-VirtualBox:/home/openstack# service neutron-dhcp-agent restart
root@openstack-VirtualBox:/home/openstack# service neutron-metadata-agent restart
root@openstack-VirtualBox:/home/openstack# service neutron-l3-agent restart
root@openstack-VirtualBox:/home/openstack#
```

Et sur le nœud compute redémarrer les services suivants

```
root@compute:/home/openstack#
root@compute:/home/openstack# service nova-compute restart
root@compute:/home/openstack# service neutron-linuxbridge-agent restart
root@compute:/home/openstack#
root@compute:/home/openstack#
```

Après avoir terminé l'installation et la configuration du service Neutron on doit maintenant vérifier son fonctionnement :

```
openstack@openstack-VirtualBox:~$
openstack@openstack-VirtualBox:~$ openstack network agent list
+--------------------------------------+--------------------+---------------------+-------------------+-------+-------+---------------------------
---------------+
| ID                                   | Agent Type         | Host                | Availability Zone | Alive | State | Binary
              |
+--------------------------------------+--------------------+---------------------+-------------------+-------+-------+---------------------------
---------------+
| 6c2284b0-54b9-41cd-b022-9e33e6d2e23e | Metadata agent     | openstack-VirtualBox | None              | :-)   | UP    | neutron-met
adata-agent     |
| a34d23ca-6035-45bc-97fa-2b28861e4a32 | Linux bridge agent | openstack-VirtualBox | None              | :-)   | UP    | neutron-lin
uxbridge-agent |
| f823228b-4a59-4b25-85d7-828888917b90 | DHCP agent         | openstack-VirtualBox | nova              | :-)   | UP    | neutron-dhc
p-agent         |
+--------------------------------------+--------------------+---------------------+-------------------+-------+-------+---------------------------
---------------+
openstack@openstack-VirtualBox:~$
```

Service Horizon

Sur le noeud contrôleur, nous installons le package du service Horizon avec la commande

 # apt install openstack-dashboard

Ensuite, nous modifions le fichier **/etc/openstack-dashboard/local_settings.py**. Nous configurons le tableau de bord pour utiliser les services d'OpenStack

- Configurez le tableau de bord

```
OPENSTACK_HOST = "controller"
```

- Configurez le service de stockage de session **memcached**

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

CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '10.55.92.60:11211',
    },
}
```

- Activez la version 3 de l'API Identity :

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

- Activer la prise en charge des domaines :

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
```

- Configurez les versions d'API :

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

- Configurez **Default** comme domaine par défaut pour les utilisateurs que vous créez via le tableau de bord :

```
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
```

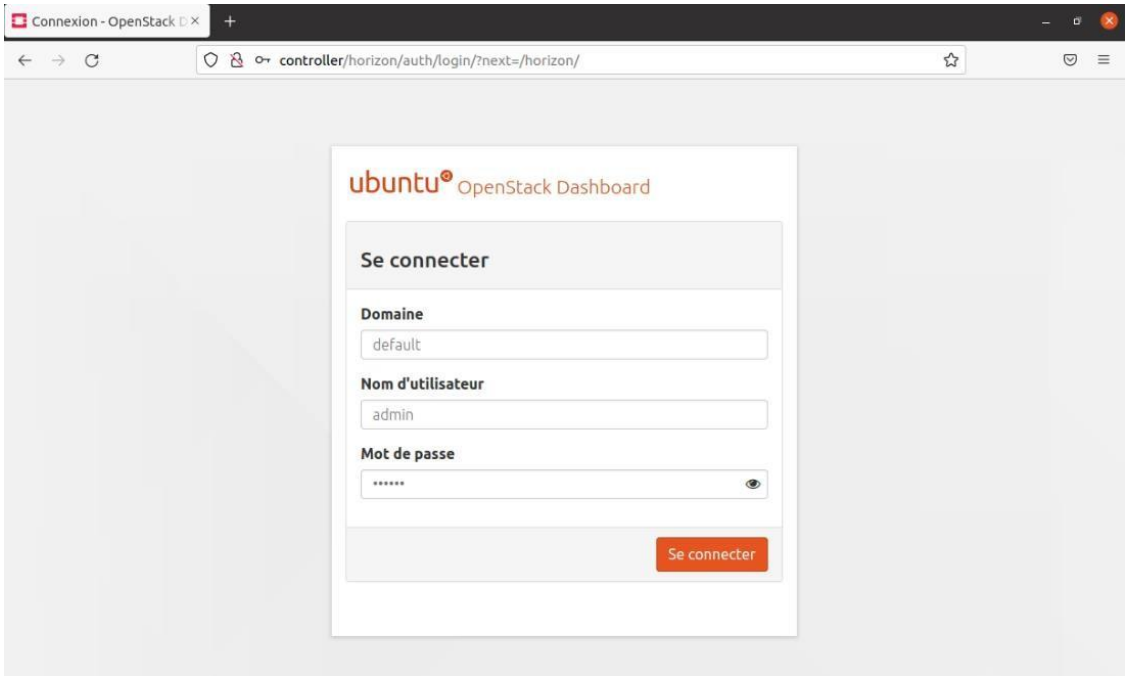
- Configurez **user** comme rôle par défaut pour les utilisateurs que vous créez via le tableau de bord :

```
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
```

On doit redémarrer le service apache

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# systemctl reload apache2.service
root@openstack-VirtualBox:/home/openstack# systemctl status apache2.service
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2022-05-14 00:00:23 CET; 46min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1154 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Process: 8551 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
   Main PID: 1385 (apache2)
      Tasks: 134 (limit: 9452)
     Memory: 78.5M
     CGroup: /system.slice/apache2.service
```

Pour tester le fonctionnement du service Horizon on doit accédez au tableau de bord à l'aide d'un navigateur Web à l'adresse **`http://controller/horizon`**



Service Cinder

Sur le nœud contrôleur, nous effectuons les étapes suivantes :

Nous créons en premier temps, une base de données des informations d'identification de service

```
root@openstack-VirtualBox:/home/openstack# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 454
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE cinder;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
    ->   IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
    ->   IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
```

Ensuite, nous créons un utilisateur nommé **cinder** avec ces commandes :

$ openstack user create --domain default --password-prompt cinder

Puis, nous ajoutons le rôle **admin** à l'utilisateur **cinder**

$ openstack role add --project service --user cinder admin

Nous créons par la suite les entités de service **cinderv2** et **cinderv3** car les services de stockage en mode bloc nécessitent deux entités de service

**$ openstack service create --name cinderv2 **

--description "OpenStack Block Storage" volumev2

**$ openstack service create --name cinderv3 **

--description "OpenStack Block Storage" volumev3

Enfin, nous produisons les points de terminaison de l'API du service **cinderv2** et **cinderv3** (public, interne, admin)

les points de terminaison de l'API du volumev2 :

**$ openstack endpoint create --region RegionOne **

volumev2 public http://controller:8776/v2/%\(project_id\)s

**$ openstack endpoint create --region RegionOne **

volumev2 internal http://controller:8776/v2/%\(project_id\)s

**$ openstack endpoint create --region RegionOne **

volumev2 admin http://controller:8776/v2/%\(project_id\)s

les points de terminaison de l'API du volumev3 :

**$ openstack endpoint create --region RegionOne **

Volumev3 public http://controller:8776/v3/%\(project_id\)s

**$ openstack endpoint create --region RegionOne **

Volumev3 internal http://controller:8776/v3/%\(project_id\)s

**$ openstack endpoint create --region RegionOne **

Volumev3 admin http://controller:8776/v3/%\(project_id\)s

Pour débuter la configuration du service cinder, nous installons le package avec la commande

 # apt install cinder-api cinder-scheduler

Au fichier **/etc/cinder/cinder.conf**, nous ajoutons les modifications suivantes :

- A la section **[database]**, nous configurons l'accès à la base de données

```
[database]
#connection = sqlite:////var/lib/cinder/cinder.sqlite
connection = mysql+pymysql://cinder:123456@controller/cinder
```

- Au niveau de la section **[DEFAULT]**, nous autorisons l'accès à la file d'attente de messages RabbitMQ, l'accès au service d'identité et nous déclarons l'adresse IP de l'interface de gestion du noeud contrôleur

```
  GNU nano 4.8
[DEFAULT]
my_ip = 10.0.0.11
auth_strategy = keystone
transport_url = rabbit://openstack:123456@controller
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes
enabled_backends = lvm
```

- A la section **[keystone_authtoken]**, nous configurons l'accès au service d'identité

```
[keystone_authtoken]

www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = 123456
```

- Au niveau de la section **[oslo_concurrency]**, nous indiquons le chemin de verrouillage

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

En plus, nous éditons le fichier **/etc/nova/nova.conf** et nous ajoutons la ligne suivante dans la section **[cinder]**

```
[cinder]

os_region_name = RegionOne
```

Après ces modifications, nous remplissons la base de données du service cinder et nous redémarrons le service API du service nova et les services de stockage en bloc

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

Enfin nous redémarrons les services du stockage

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# service nova-api restart
root@openstack-VirtualBox:/home/openstack# service cinder-scheduler restart
root@openstack-VirtualBox:/home/openstack# service apache2 restart
```

Sur le noeud de stockage par bloc, nous installons le paquet des utilitaires associés avec la commande :

 # apt install lvm2 thin-provisioning-tools

- Nous créons le groupe des volumes physiques /**dev/sda et** /**dev/sdb**

 # pvcreate /dev/sdb

 # vgcreate cinder-volumes /dev/sdb

- Les périphériques **/dev/sda et /dev/sdb** et rejette tous les autres périphériques.

```
devices {

        filter = [ "a/sda/", "a/sdb/", "r/.*/"]
```

Par la suite nous installons le package cinder avec la commande

 # apt install cinder-volume

Nous éditons le fichier **/etc/cinder/cinder.conf** et nous ajoutons les informations suivantes :

- A la section **[database]**, nous autorisons l'accès à la base de données

```
[database]
connection = mysql+pymysql://cinder:123456@controller/cinder
```

- Au niveau de la section [DEFAULT], nous autorisons l'accès à la file d'attente de messages RabbitMQ, l'accès au service d'identité. Nous déclarons aussi l'adresse IP de l'interface de gestion du noeud contrôleur et nous activons le backend LVM sans oublier de configurer l'emplacement de l'API du service d'image.

```
  GNU nano 4.8
[DEFAULT]
transport_url = rabbit://openstack:123456@controller
auth_strategy = keystone
my_ip = 10.55.92.62
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes
enabled_backends = lvm
glance_api_servers = http://controller:9292
```

- A la section **[keystone_authtoken]**, nous configurons l'accès au service d'identité

```
[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = 123456
```

- Au niveau de la section **[lvm]**, nous configurons le pilote LVM, le groupe de volumes **cinder-volumes**, le protocole de stockage en réseau iSCSI

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

- A la section **[oslo_concurrency]**, nous spécifions le chemin de verrouillage

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

Après ces modifications, nous redémarrons les services de volume Block Storage

```
root@stockage:/home/openstack#
root@stockage:/home/openstack# service tgt restart
root@stockage:/home/openstack# service cinder-volume restart
root@stockage:/home/openstack#
```

Et maintenant nous vérifions sur **le nœud contrôleur** le lancement de chaque processus

```
openstack@controller:~$ openstack volume service list
+------------------+-------------+------+---------+-------+----------------------------+
| Binary           | Host        | Zone | Status  | State | Updated At                 |
+------------------+-------------+------+---------+-------+----------------------------+
| cinder-scheduler | controller  | nova | enabled | up    | 2022-05-19T10:14:44.000000 |
| cinder-volume    | stockage@lvm | nova | enabled | up    | 2022-05-19T10:15:14.000000 |
+------------------+-------------+------+---------+-------+----------------------------+
openstack@controller:~$
```

Service Heat

Pour déployer ce service, nous réalisons les étapes suivantes **sur le noeud contrôleur** :

Nous créons une base de données des informations d'identification du service **Heat** et des points de terminaison d'API.

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 815
Server version: 10.3.34-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE heat;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.007 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%'  IDENTIFIED BY '123456';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```

Ensuite, nous créons un utilisateur nommé **heat** avec ces commandes :

$ openstack user create --domain default --password-prompt heat

Nous ajoutons le rôle admin à l'utilisateur heat avec cette commande :

$ openstack role add --project service --user heat admin

Nous créons par la suite les entités de services **heat** et **heat-cfn**

**$ openstack service create --name heat **

--description "Orchestration" orchestration

**$ openstack service create --name heat-cfn **

--description "Orchestration" cloudformation

Enfin, nous créons les points de terminaison de l'API du service **heat** et **heat-cfn** (public, interne, admin)

Les API de l'entité heat :

***$ openstack endpoint create --region RegionOne ***

orchestration public http://controller:8004/v1/%\(tenant_id\)s

***$ openstack endpoint create --region RegionOne ***

orchestration internal http://controller:8004/v1/%\(tenant_id\)s

***$ openstack endpoint create --region RegionOne ***

orchestration admin http://controller:8004/v1/%\(tenant_id\)s

Les API de l'entité heat-cfn :

**$ openstack endpoint create --region RegionOne **

cloudformation public http://controller:8000/v1

**$ openstack endpoint create --region RegionOne **

cloudformation internal http://controller:8000/v1

**$ openstack endpoint create --region RegionOne **

cloudformation admin http://controller:8000/v1

Le service d'orchestration nécessite des informations supplémentaires au niveau du service d'identité. Nous définissons le domaine **heat** qui contient les projets et les utilisateurs

$ openstack domain create --description "Stack projects and users" heat

Puis, nous créons l'utilisateur nommé heat_domain_admin pour gérer les projets et les utilisateurs du domaine heat

$ openstack user create --domain heat --password-prompt heat_domain_admin

Nous ajoutons le rôle admin à l'utilisateur heat_domain_admin du domaine heat afin de lui activer les privilèges d'administration

$ openstack role add --domain heat --user-domain heat --user heat_domain_admin admin

- Aussi, nous créons le rôle heat_stack_owner

$ openstack role create heat_stack_owner

- Nous ajoutons aussi le rôle heat_stack_owner au projet demo et à l'utilisateur demo afin d'activer la gestion de heat pour ce dernier

$ openstack role add --project demo --user demo heat_stack_owner

- Nous créons le rôle heat_stack_user

$ openstack role create heat_stack_user

Une fois cette étape achevée, nous installons le package avec la commande

 # apt-get install heat-api heat-api-cfn heat-engine

Nous éditons le fichier **/etc/heat/heat.conf** pour en ajouter les modifications :

- A la section **[database]**, nous autorisons l'accès à la base de données

```
[database]

connection = mysql+pymysql://heat:123456@controller/heat
```

- Au niveau de la section **[DEFAULT]**, nous autorisons l'accès à la file d'attente de messages RabbitMQ, les URL de métadonnées, le domaine de heat et les informations d'identification administratives

```
[DEFAULT]

transport_url = rabbit://openstack:123456@controller
heat_metadata_server_url = http://controller:8000
heat_waitcondition_server_url = http://controller:8000/v1/waitcondition
stack_domain_admin = heat_domain_admin
stack_domain_admin_password = 123456
stack_user_domain_name = heat
```

- Aux sections **[keystone_authtoken]**, **[trustee]** et **[clients_keystone]**, nous configurons l'accès au service d'identité.

```
[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 = heat
password = 123456
```

```
[trustee]

auth_type = password
auth_url = http://controller:35357
username = heat
password = 123456
user_domain_name = default
```

```
[clients_keystone]

auth_uri = http://controller:35357
```

Enfin, nous remplissons la base de données de service d'Orchestration

 # su -s /bin/sh -c "heat-manage db_sync" heat

Enfin, nous redémarrons les services d'orchestration :

```
root@openstack-VirtualBox:/home/openstack#
root@openstack-VirtualBox:/home/openstack# service heat-api restart
root@openstack-VirtualBox:/home/openstack# service heat-api-cfn restart
root@openstack-VirtualBox:/home/openstack# service heat-engine restart
root@openstack-VirtualBox:/home/openstack#
```

Une fois on a terminé la configuration du service d'orchestration on doit tester son fonctionnement.

```
openstack@openstack-VirtualBox:~$ . admin-openrc
openstack@openstack-VirtualBox:~$  openstack orchestration service list
+------------+------------+--------------------------------------+------------+--------+----------------------------+--------+
| Hostname   | Binary     | Engine ID                            | Host       | Topic  | Updated At                 | Status |
+------------+------------+--------------------------------------+------------+--------+----------------------------+--------+
| controller | heat-engine | 349766bc-75be-4654-bbbf-9937dc03c286 | controller | engine | 2022-05-17T10:58:01.000000 | up     |
| controller | heat-engine | 2405685a-b244-4f06-b874-6565596a0d61 | controller | engine | 2022-05-17T10:58:01.000000 | up     |
| controller | heat-engine | 5deffd07-80c3-4af8-8eb0-c99242308bf1 | controller | engine | 2022-05-17T10:58:01.000000 | up     |
| controller | heat-engine | 12f93e98-1b84-4d31-87f8-7dfd7311b3ac | controller | engine | 2022-05-17T10:58:01.000000 | up     |
+------------+------------+--------------------------------------+------------+--------+----------------------------+--------+
openstack@openstack-VirtualBox:~$
```

II. Hadoop

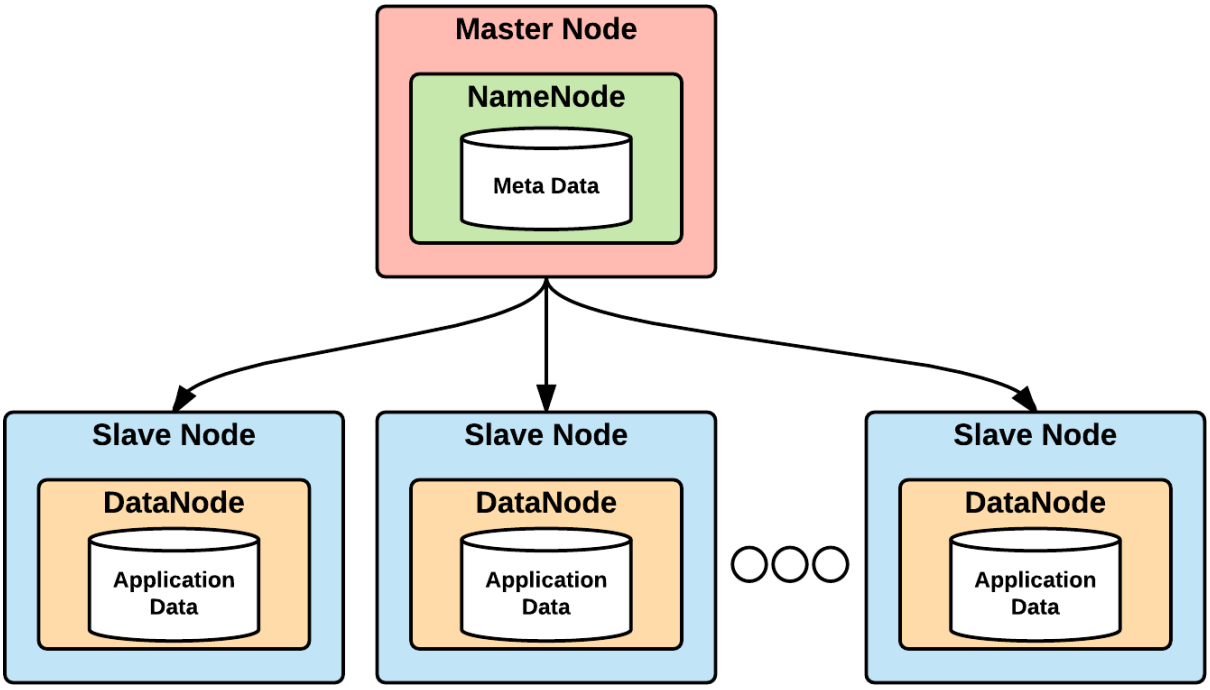
1. Architecture

Apache Hadoop est une collection d'utilitaires logiciels open source qui facilitent l'utilisation d'un réseau de plusieurs ordinateurs pour résoudre des problèmes impliquant des quantités massives de données et de calculs. Il fournit un cadre logiciel pour les systèmes distribués (un système de fichiers en cluster) et le traitement des mégadonnées à l'aide de **MapReduce**.

Il stocke les données sur des machines standard, fournissant une bande passante agrégée très élevée sur le cluster a l'aide du HDFS (Hadoop Distributed File System). La gestion des ressources informatiques la planification des applications des utilisateur est faite par **YARN**.

Hadoop fonctionne en mode **cluster**. Un **cluster** est un ensemble d'ordinateurs connectés qui fonctionnent ensemble de sorte qu'ils peuvent être considérés comme un système unique.

Maître/Esclave (ou Name Node et Data Node) : Un nœud maitre n'est rien de plus que l'ordinateur principal ou la machine qui vous permettra d'envoyer des commandes à tous les autres nœuds, appelés esclaves/secondaires/Data Node, dans un cluster qui effectuera la même tâche.



2. Installation

Pour installer et configurer un cluster Hadoop Il faut suivre les étapes suivantes :

- Installer SSH :

$ sudo apt installer ssh

⇨ Le protocole SSH permet la connexion à distance via une liaison sécurisée dans le but de transférer des fichiers ou exécuter des commandes.

- Installation de PDSH

sudo apt installer pdsh

⇨ C'est permis d'exécuter des commandes sur plusieurs noeuds en utilisant uniquement SSH.

- Configuration de l'environnement PDSH et de l'accès pour utiliser SSH :

Accéder au fichier .bashrc et écrivez la commande suivante :

exporter PDSH_RCMD_TYPE=ssh

- Générer la clé SSH :

ssh-keygen -t rsa -P ""

- Dupliquer la clé SSH dans un autre dossier :

cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

- Installer Java :

sudo apt installer openjdk-8-jdk

- Télécharger Hadoop :

Wget https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz

- Extrairer Hadoop :

tar -xzvf Hadoop-3.3.1.tar.gz

- Renommer le dossier :

mv Hadoop-3.3.1 hadoop

- Configurer le fichier de l'environnement du Hadoop :

Accéder au fichier de configuration et ajouter la ligne suivante :

nano ~/hadoop/etc/hadoop/hadoop-env.sh

exporter JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

- Déplacer Hadoop vers une répertoire d'utilisateurs :

sudo mv Hadoop /usr/local/Hadoop

- Configuration du chemin Hadoop et du chemin java home :

Accéder au fichier d'environnement et ajouter les lignes suivantes

sudo nano /etc/environnement

PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/ hadoop/bin:/usr/local/hadoop/sbin"

JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre"

- Création d'un utilisateur Hadoop :

sudo adduser Hadoop

- Donner toutes les autorisations nécessaires :

sudo usermod -aG Hadoop Hadoop

sudo chown Hadoop:root -R /usr/local/Hadoop/

sudo chmod g+rwx -R /usr/local/Hadoop/

sudo adduser Hadoop sudo

- Répéter toutes les étapes précédentes pour les nœuds que vous avez besoin :
- Accéder au fichier suivant et ajouter l'adresse IP de la machine ainsi du son nom d'hôte
- ⇨ Pour tous les nœuds

sudo nano /etc/hosts

- Redémarrer les machines :

sudo reboot

- Connecter au nouvel utilisateur :

su – hadoopuser

- Création de nouvelle clé SSH :
- ⇨ Nœud master uniquement

hadoop ssh-keygen -t rsa

- Partage du clé avec les autres machines :
- ⇨ Nœud master uniquement

ssh-copy-id hadoop@master

ssh-copy-id hadoop@slave1

ssh-copy-id hadoop@slave2

- Configuration des ports Hadoop :

⇨ Nœud master uniquement

Accéder au fichier de configuration suivant

sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
```

- Configuration du HDFS :

⇨ Nœud master uniquement

Accéder au fichier de configuration suivant

sudo nano /usr/local/Hadoop/etc/Hadoop/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

- Etiqueter les noeuds secondaires :

⇨ Nœud master uniquement

Accéder au fichier de configuration suivant

sudo nano /usr/local/Hadoop/etc/Hadoop/workers

Ajouter tous les noms d'hôtes des machines esclaves

- Copier les fichiers du maitre vers les esclaves :

⇨ Nœud master uniquement

scp /usr/local/hadoop/etc/Hadoop/* hadoop-slave1: /usr/local/hadoop/etc/Hadoop/

scp /usr/local/hadoop/etc/Hadoop/* hadoop-slave2: /usr/local/hadoop/etc/Hadoop/

- Enregistrer les configurations :

⇨ Nœud master uniquement

source /etc/environnement

- Formater de système HDFS :

⇨ Nœud master uniquement

hdfs namenode -format

- Démarrer HDFS :

⇨ Nœud master uniquement

start-fds.sh

Exportation des chemins et des fichiers :

⇨ Nœud master uniquement

export HADOOP_HOME="/usr/local/hadoop"

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_HDFS_HOME=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

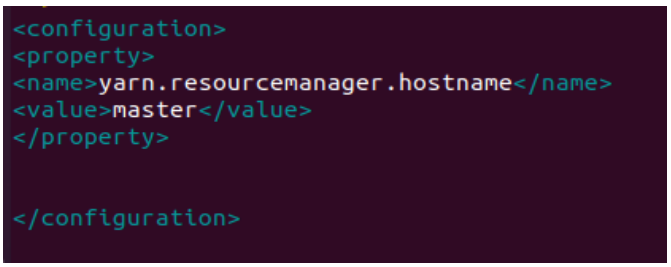
export HADOOP_YARN_HOME=$HADOOP_HOME

Configuration de l'outil Yarn :

⇨ Les nœuds slaves uniquement

Accéder au fichier de configuration suivant

sudo nano /usr/local/hadoop/etc/Hadoop/yarn-site.xml

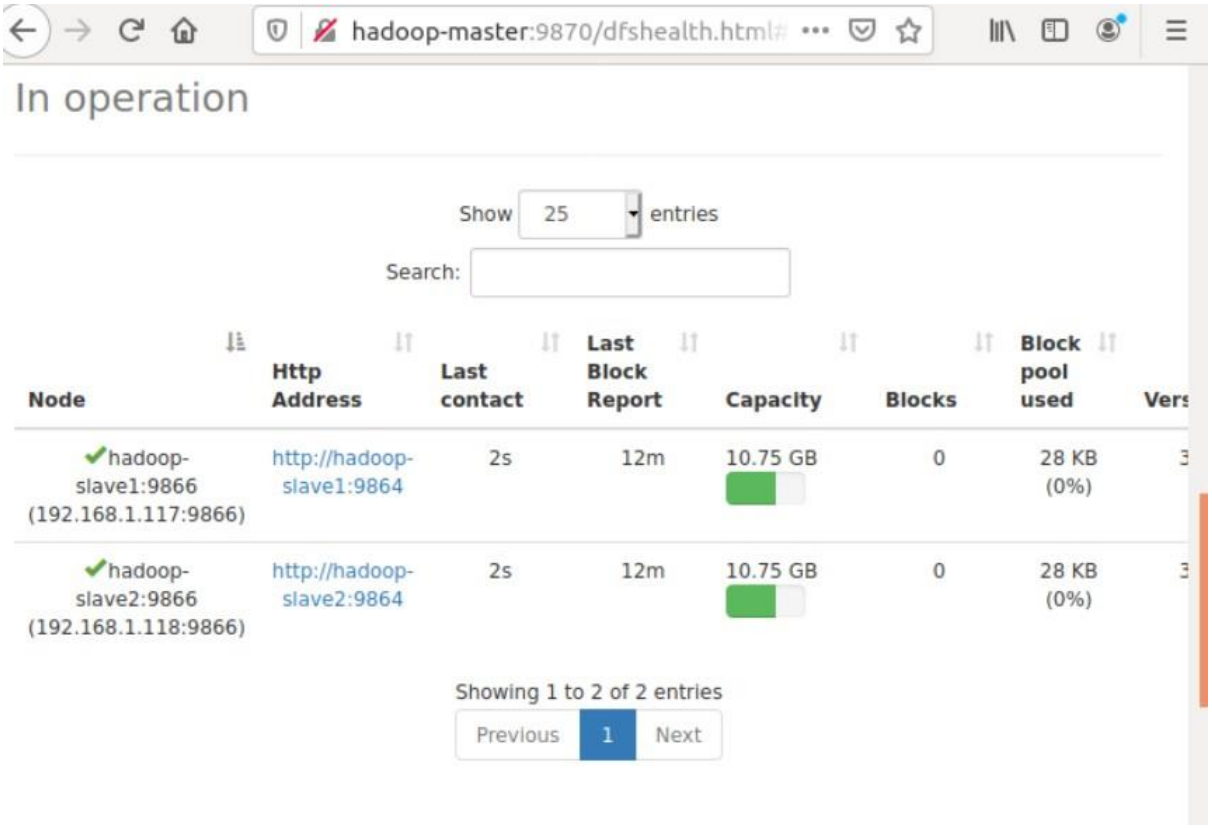


- Démarrer Yarn :

⇨ Nœud master uniquement

start-yarn.sh

La configuration du cluster Hadoop est terminée et maintenant nous accédons via le navigateur à l'aide de l'adresse : **http://hadoop-master:9870** au dashboard de la plateforme Hadoop.



3. Installation Spark

Dans cette partie, nous allons montré comment installer Apache Spark sur Ubuntu 20.04

- Créer un répertoire pour Apache Spark :

mkdir -p spark

- installer le kit de développement Java (JDK) :

sudo apt-get install default-jdk -y

- Télécharger Apache Spark :

Accéder a la repertoire crée « spark »

cd spark

Wget https://dlcdn.apache.org/spark/spark-3.3.1/spark-3.3.0-bin-hadoop3.3.tgz

- Installer Apache Spark sur Ubuntu

Tar -xvf spark-3.3.1-bin-hadoop3.3.tgz

Accéder a la répertoire suivante

Sudo nano ~/.bashrc

```
SPARK_HOME=/home/hadoopuser/spark/spark-3.2.1-bin-hadoop3.2
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

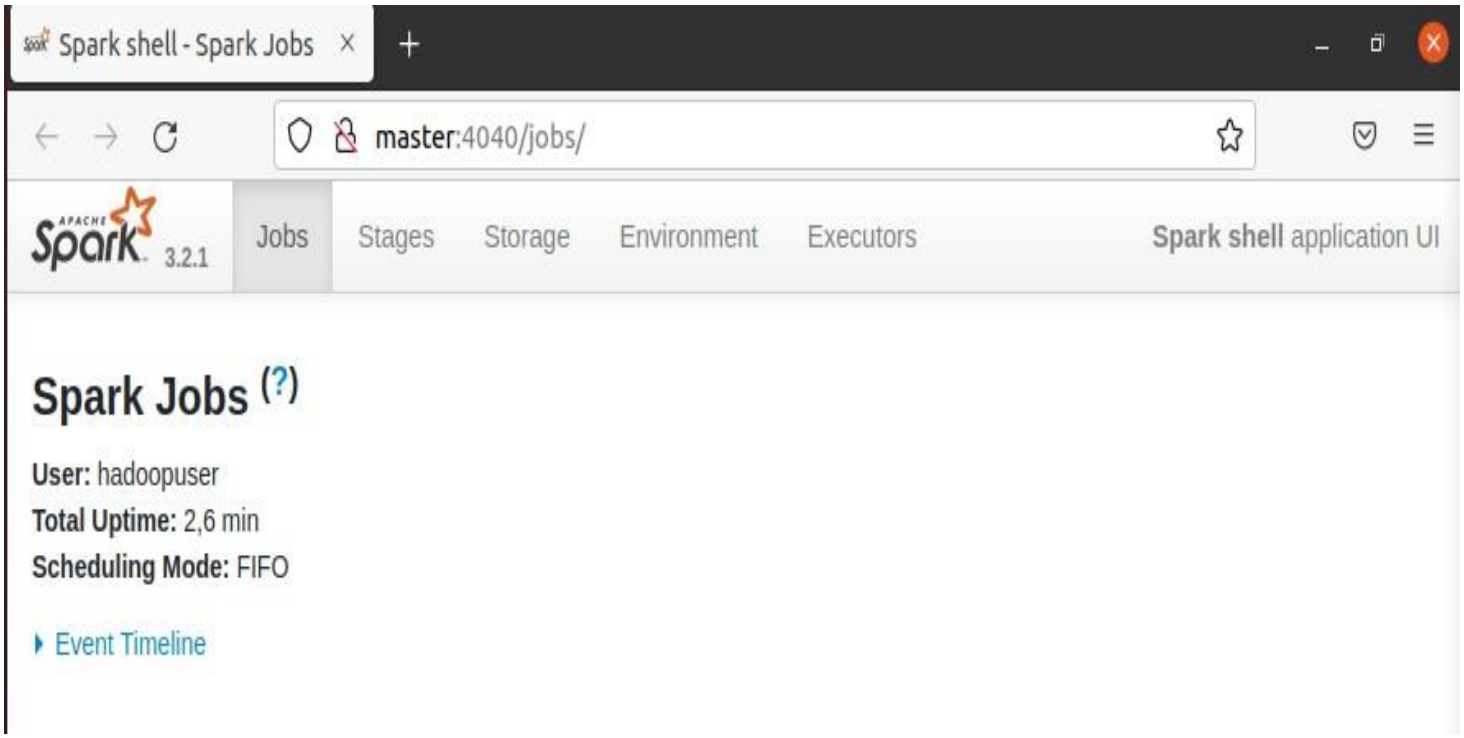
Terminer l'installation en utilisant la commande suivante

Source ~/.bashrc

- **Vérifier l'installation :**

```
hadoopuser@master:~$ spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/06/02 01:14:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = local[*], app id = local-1654128853253).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 1.8.0_312)
Type in expressions to have them evaluated.
Type :help for more information.
```

- **Accéder à l'interface utilisateur Web de l'Apache Spark :**



4. Apache Cassandra

Dans cette partie, nous allons montré comment installer Apache Cassandra sur Ubuntu 20.04

- installer le kit de développement Java (JDK) :

sudo apt-get install default-jdk -y

- vérifier la version du Java

java -version

- Ajoutez le référentiel Apache de Cassandra au fichier cassandra.sources.list.

echo "deb http://www.apache.org/dist/cassandra/debian 40x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

deb http://www.apache.org/dist/cassandra/debian 40x main

- Ajoutez les clés du référentiel Apache Cassandra à la liste des clés de confiance sur le serveur :

curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -

- Installez Cassandra :

sudo apt-get install cassandra

- Démarrer Cassandra

tail -f logs/system.log

- Vérifiez l'état de Cassandra :

```
hadoopuser@master:~$ nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load        Tokens  Owns (effective)  Host ID                               Rack
UN  127.0.0.1  178,29 KiB  16      100,0%            10d3d716-5856-4613-b985-c71fbdcd0843  rack1

hadoopuser@master:~$
```

- Connecter à la base de données Cassandra :

```
hadoopuser@master:~$  cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.4 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh>
```

Conclusion

Ce guide décrit d'une manière détaillée le déploiement des composants d'OpenStack afin de créer une plateforme de cloud privé ainsi que l'installation du Hadoop, Spark et Cassandra dans le but de déployer une solution Big Data complète.