

Projet du cloud computing :

Mise en œuvre d'une infrastructure Cloud Openstack
Victoria

RÉALISÉ PAR :

- LAAZIZ Fatima zahrae
- OUSSAKEL Khadija
- TAMLALTI Maryam

ENCADRÉ PAR :

- Pr. E. KANDOUSSI

INTRODUCTION

OpenStack est une collection de modules logiciels et d'outils **open source** qui fournit un cadre pour créer et gérer à la fois une infrastructure de **cloud public** et de **cloud privé**. Il fournit des fonctionnalités d'infrastructure en tant que service (**IaaS**) : il regroupe, provisionne et gère de grandes concentrations de ressources de calcul, de stockage et de réseau. Ces ressources, qui comprennent du matériel bare metal, des machines virtuelles (VM) et des conteneurs, sont gérées via des interfaces de programmation d'applications (API) ainsi qu'un tableau de bord OpenStack. D'autres composants OpenStack fournissent l'orchestration, la gestion des pannes et des services destinés à prendre en charge des opérations fiables et à haute disponibilité.

Les entreprises et les fournisseurs de services peuvent **déployer** OpenStack sur site (dans le centre de données pour créer un cloud privé), dans le cloud pour activer ou piloter des plates-formes de cloud public, et à la périphérie du réseau pour les systèmes informatiques distribués.

OpenStack utilise des **scripts** regroupés dans des projets pour créer et gérer des ressources cloud à grande échelle en s'appuyant sur deux types de logiciels pour créer et gérer ces ressources cloud :

- Un **hyperviseur** (ou logiciel de virtualisation) qui crée une couche de ressources virtuelles à partir du matériel physique, permettant ainsi de créer et de gérer des machines virtuelles (VM). OpenStack prend en charge plusieurs hyperviseurs tels que KVM, Xen, VMware, Hyper-V, etc.
- Un **système d'exploitation de base** (tel que Linux ou CentOS) qui exécute les commandes transmises par les scripts OpenStack et fournit les services de base tels que la gestion des utilisateurs, la gestion des fichiers, etc.

Le Cloud OpenStack se compose de plusieurs éléments, dont les principaux sont les suivants:

- **Nova**

Nova est le composant de **calcul du Cloud** OpenStack et le premier composant à avoir été intégré à OpenStack en 2010. C'est en quelque sorte la colonne vertébrale du Cloud et il se charge d'une tâche essentielle : la gestion des ordinateurs virtuels. Sur la durée, Nova s'est amélioré et permet désormais de gérer des groupes entiers d'ordinateurs. Les ordinateurs virtuels sont reliés entre eux par une connexion synaptique. Le nombre de noeuds (aussi appelés synapses) est variable. Nova utilise principalement les hyperviseurs sans licence KVM intégrés dans le noyau Linux et XEN développé par l'Université de Cambridge comme base des machines virtuelles utilisées.



openstack®

- **Keystone**

Keystone est responsable de **l'autorisation et de l'authentification des utilisateurs** (Identity). Le Cloud computing ne se contente pas de gérer des ordinateurs virtuels, il met en place des réseaux entiers. Dans ce contexte, l'identification des utilisateurs et la compartimentation des activités sont indispensables. Keystone accorde à chaque utilisateur du Cloud (appelé « mandant ») un accès individualisé. Cet accès recense également les droits du mandant.

- **Glance**

Glance met à disposition les « **images** », c'est-à-dire les images des supports de données des machines virtuelles. En outre, Glance peut également sauvegarder et restaurer les images. Il est possible de créer une sorte de bibliothèque avec les modèles des systèmes requis. Ceux-ci peuvent être recréés ultérieurement dans le réseau aussi souvent que nécessaire. Par ailleurs, Glance garantit la disponibilité, car les machines nécessaires peuvent être recréées à tout moment.

- **Neutron**

Neutron (anciennement Quantum) constitue **l'infrastructure réseau virtuelle** d'OpenStack. Il est ainsi possible de répartir des sous-réseaux, de gérer les adresses IP et de générer des réseaux virtuels (VLAN). Les VPN (Virtual Private Networking) sont également pris en charge par Neutron. Ce composant permet avant tout l'échange de données entre les éléments d'OpenStack, par exemple entre les machines virtuelles individuelles. Le pare-feu du réseau est également mis en place par Neutron.

- **Cinder**

Cinder est responsable de la mise à disposition d'un **stockage permanent** sous forme de stockage **en bloc**, un disque dur par exemple. Celui-ci est créé via Cinder grâce à la virtualisation. Il est ainsi possible d'adapter les volumes aux besoins (évolutivité). Le stockage en bloc Cinder se comporte comme un disque dur physique dans un ordinateur. La sécurisation des données est simple, car l'utilisateur peut accéder au disque dur via une interface centrale qui possède également une fonction snapshot (instantané).

- **Swift**

Swift est le **stockage objet**. Il peut connecter les stockages dans différents endroits afin de pouvoir utiliser des objets de données répartis aléatoirement sur des stockages adjacents. Cela crée, si nécessaire, de manière transparente une redondance puisque les objets peuvent être stockés physiquement plusieurs fois. En outre, les stockages mis à disposition par Swift peuvent être utilisés par Cinder ou Glance. Il est également possible, grâce à Ceph ou GlusterFS, d'utiliser la mémoire d'objet distribuée comme sous-structure.

- **Horizon**

Horizon est un **tableau de bord**. Autrement dit, Horizon est l'interface utilisateur graphique qui permet de gérer les composants regroupés dans OpenStack. Il est également utilisé pour gérer les utilisateurs. Le design et les fonctionnalités d'Horizon sont adaptables.



openstack®

Table des matières :

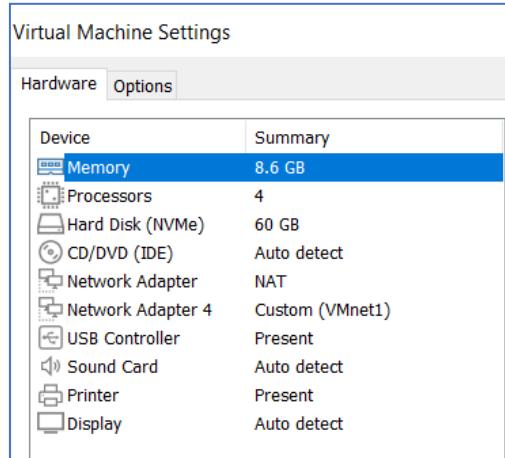
I.	INSTALLATION ET PREPARATION DE LA MACHINE VIRTUELLE	5
II.	CONFIGURATION DES PRE-REQUIREMENTS	9
III.	CONFIGURATION DE KEYSTONE #1	19
IV.	CONFIGURATION DE KEYSTONE #2	25
V.	CONFIGURATION DE GLANCE	28
VI.	AJOUT DES IMAGES DE LA MACHINE VIRTUELLE	34
VII.	CONFIGURATION DE NOVA #1	40
VIII.	CONFIGURATION DE NOVA #2	46
IX.	CONFIGURATION DE NOVA #3	54
X.	CONFIGURATION DE NEUTRON #1	57
XI.	CONFIGURATION DE NEUTRON #2	61
XII.	CONFIGURATION DE LA MISE EN RESEAU DE NEUTRON	68
XIII.	AJOUT DES UTILISATEURS OPENSTACK	72
XIV.	CREATION ET EXECUTION DES INSTANCES	75
XV.	CONFIGURATION D'HORIZON	80



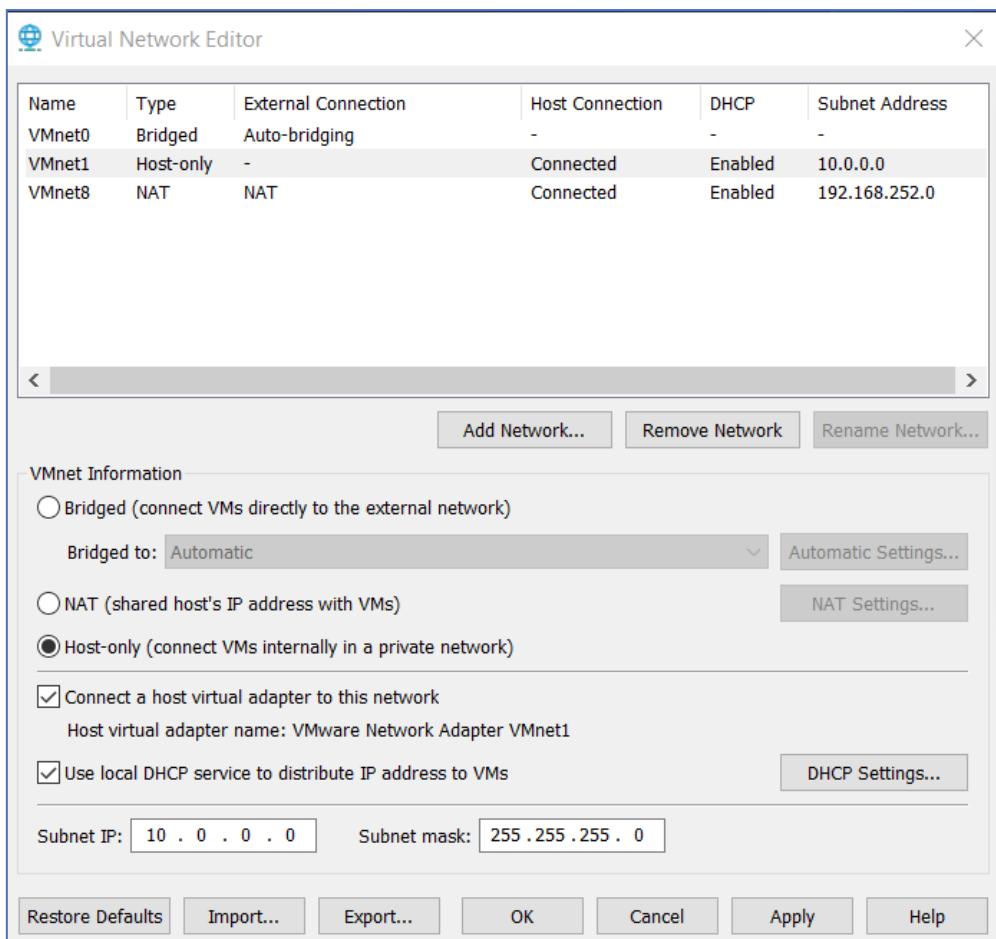
I. Installation et préparation de la machine virtuelle



Tout d'abord on installe une machine virtuelle CentOS 8 sur VMware dont les caractéristiques sont les suivantes :



Puis on ajoute l'adaptateur de réseau NAT qui sert à communiquer avec Internet, et un autre adaptateur *Host Only* pour communiquer avec la machine hôte :



Ainsi, on change le hostname de la machine vers “controller”, et on désactive également le firewall par la commande suivante :

```
$systemctl stop firewalld  
$systemctl disable firewalld
```

```
khadijaoussakel@localhost:/home/khadijaoussakel  
Fichier Édition Affichage Rechercher Terminal Aide  
[khadijaoussakel@localhost ~]$ su root  
Mot de passe :  
[root@localhost khadijaoussakel]# systemctl stop firewalld  
[root@localhost khadijaoussakel]# systemctl disable firewalld  
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.  
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.  
[root@localhost khadijaoussakel]# systemctl status firewalld  
● firewalld.service - firewalld - dynamic firewall daemon  
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor >  
  Active: inactive (dead)  
    Docs: man:firewalld(1)  
  
janv. 15 11:43:08 localhost.localdomain systemd[1]: Starting firewalld - dynamic>  
janv. 15 11:43:09 localhost.localdomain systemd[1]: Started firewalld - dynamic>  
janv. 15 13:02:09 localhost.localdomain systemd[1]: Stopping firewalld - dynamic>  
janv. 15 13:02:13 localhost.localdomain systemd[1]: Stopped firewalld - dynamic>  
lines 1-9/9 (END)
```

On édite le fichier /etc/hostname :

```
File Edit View Search Terminal Help  
localhost.localdomain  
~  
~
```

```
File Edit View Search Terminal Help  
controller  
~
```

Puis on ajoute dans le fichier /etc/hosts la ligne suivante :

```
File Edit View Search Terminal Help  
GNU nano 2.9.8 /etc/hosts Modified  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
10.0.0.30 controller
```

Ensuite, on configure les interfaces réseau de la machine hôte pour qu'elle puisse communiquer avec les autres machines de notre réseau local. Cela implique de configurer les fichiers ifcfg-ens160 et ifcfg-ens192 avec les bonnes informations de réseau :

```
[root@localhost network-scripts]# ls  
ifcfg-ens160 ifcfg-ens192  
[root@localhost network-scripts]#
```

Pour le fichier ifcfg-ens160 :

```
khadijaoussakel@controller:/etc/sysconfig/network-scripts
```

```
Fichier Édition Affichage Rechercher Terminal Aide
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens160
UUID=85bb2482-87f6-4f44-95e1-970852262d86
DEVICE=ens160
ONBOOT=yes
~
```

Pour le fichier ifcfg-ens192 :

```
Fichier Édition Affichage Rechercher Terminal Aide
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=10.0.0.30
NETMASK=255.255.255.0
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens192
UUID=f642c224-14ef-4402-8452-3bd5d64f015b
DEVICE=ens192
ONBOOT=yes
~
```

Puis on pingue le host pour vérifier que la machine hôte est accessible sur le réseau :

```
[khadijaoussakel@controller ~]$ ping controller
PING controller (10.0.0.30) 56(84) bytes of data.
64 bytes from controller (10.0.0.30): icmp_seq=1 ttl=64 time=0.138 ms
64 bytes from controller (10.0.0.30): icmp_seq=2 ttl=64 time=0.124 ms
64 bytes from controller (10.0.0.30): icmp_seq=3 ttl=64 time=0.110 ms
64 bytes from controller (10.0.0.30): icmp_seq=4 ttl=64 time=0.090 ms
64 bytes from controller (10.0.0.30): icmp_seq=5 ttl=64 time=0.119 ms
64 bytes from controller (10.0.0.30): icmp_seq=6 ttl=64 time=0.080 ms
64 bytes from controller (10.0.0.30): icmp_seq=7 ttl=64 time=0.115 ms
^C
--- controller ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 130ms
rtt min/avg/max/mdev = 0.080/0.110/0.138/0.023 ms
```

II. Configuration des pré-requirements



On commence tout d'abord par configurer le serveur NTP pour l'ajustement des dates.

Le serveur NTP (Network Time Protocol) est un type de serveur qui fournit des informations de synchronisation de temps précises à des dispositifs informatiques connectés à un réseau.

Pour ce faire on installe Chrony :

```
khadijaoussakel@controller:/home/khadijaoussakel
Fichier Édition Affichage Rechercher Terminal Aide
[root@controller khadijaoussakel]# dnf -y install chrony
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:28
le lun. 16 janv. 2023 09:34:32 EST.
Package chrony-4.1-1.el8.x86_64 is already installed.
Dépendances résolues.
Rien à faire.
Terminé !
[root@controller khadijaoussakel]#
```

On modifie le fichier de configuration de Chrony en ajoutant un serveur NTP et en spécifiant notre réseau local :

```
[root@controller khadijaoussakel]# vi /etc/chrony.conf
khadijaoussakel@controller:/home/khadijaoussakel
Fichier Édition Affichage Rechercher Terminal Aide
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#pool 2.centos.pool.ntp.org iburst
pool ntp.nict.jp iburst

# Record the rate at which the system clock gains/losses time.
driftfile /var/lib/chrony/drift

# Allow NTP client access from local network.
#allow 192.168.0.0/16
allow 10.0.0.0/24
```

On active le service Chrony et on vérifie par la commande « chronyc sources » que tout marche bien :

```
[root@controller khadijaoussakel]# systemctl enable --now chronyd
[root@controller khadijaoussakel]# chronyc sources
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^? ntp-b3.nict.go.jp        0    8    0    -    +0ns[  +0ns] +/-   0ns
^? ntp-b2.nict.go.jp        0    8    0    -    +0ns[  +0ns] +/-   0ns
^? ntp-k1.nict.jp           0    8    0    -    +0ns[  +0ns] +/-   0ns
^? ntp-a2.nict.go.jp        0    8    0    -    +0ns[  +0ns] +/-   0ns
^? ntp-a3.nict.go.jp        0    8    0    -    +0ns[  +0ns] +/-   0ns
[root@controller khadijaoussakel]#
```

On vérifie le status de chrony par la commande suivante :

```
[root@controller khadijaoussakel]# systemctl status chronyd.service
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2023-01-16 09:34:37 EST; 11min ago
    Docs: man:chronyd(8)
           man:chrony.conf(5)
   Main PID: 11451 (chronyd)
      Tasks: 1 (limit: 23859)
     Memory: 2.7M
        CPU: 0.000 CPU(s) used
       CGroup: /system.slice/chronyd.service
                 └─11451 /usr/sbin/chronyd

janv. 16 09:34:37 controller systemd[1]: Starting NTP client/server...
janv. 16 09:34:37 controller chronyd[11451]: chronyd version 4.1 starting (+CMDMON +NTP +DHT +GLOBE +ARVOR)
janv. 16 09:34:37 controller chronyd[11451]: Frequency 0.293 +/- 544.295 ppm read from file /etc/chrony.conf
janv. 16 09:34:37 controller chronyd[11451]: Using right/UTC timezone to obtain leap second information
janv. 16 09:34:37 controller systemd[1]: Started NTP client/server.
lines 1-16/16 (END)
```

On installe ensuite le serveur MariaDB :

MariaDB est un système de gestion de base de données relationnelles qui est une alternative à MySQL.

```
[root@controller khadijaoussakel]# dnf module -y install mariadb:10.3
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:08:42 le lundi 16 janv. 2023 09:34:32 EST.
Dépendances résolues.
=====
Paquet          Architecture      Version      Dépôt      Taille
=====
Installing group/module packages:
mariadb-server      x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream  16 M
Installation des dépendances:
mariadb            x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream 6.0 M
mariadb-common      x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream 64 k
mariadb-connector-c x86_64 3.1.11-2.el8_3                           AppStream 200 k
mariadb-connector-c-config
                        noarch 3.1.11-2.el8_3                         AppStream 15 k
mariadb-errmsg       x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream 234 k
perl-DBD-MySQL       x86_64 4.046-3.module_el8.3.0+419+c2dec72b   AppStream 156 k
Installation des dépendances faibles:
mariadb-backup       x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream 6.1 M
mariadb-gssapi-server
                        x86_64 3:10.3.28-1.module_el8.3.0+757+d382997d AppStream 51 k
mariadb-server-utils
```

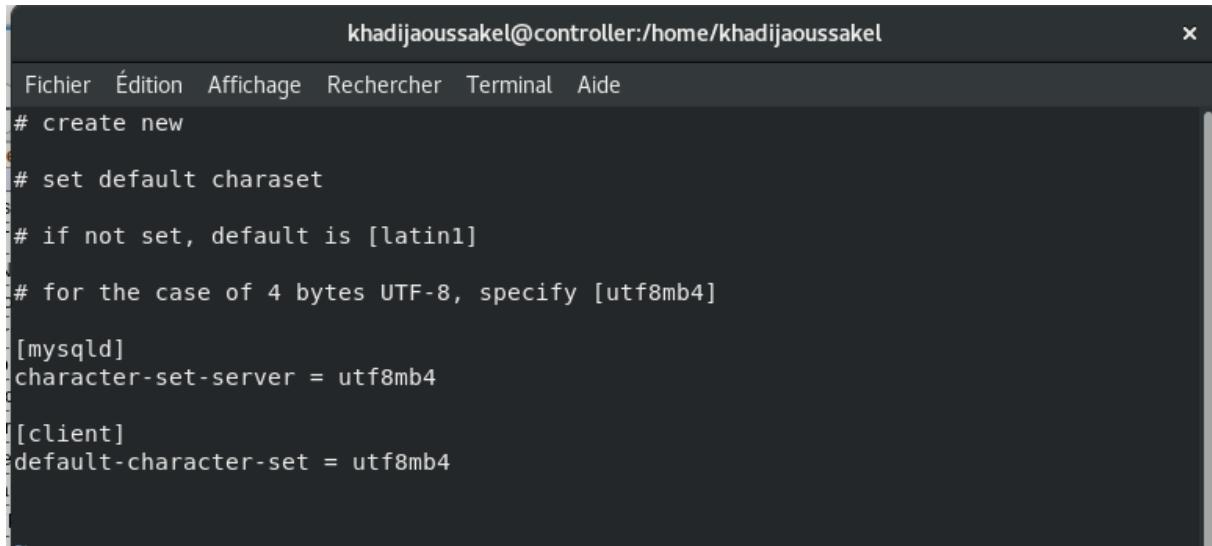
On crée un nouveau fichier de configuration « charset.cnf » pour spécifier le jeu de caractères par défaut utilisé par le serveur MySQL et les clients MySQL :

```
[root@controller khadijaoussakel]# vi /etc/my.cnf.d/charset.cnf
```

La ligne [mysqld] character-set-server = utf8mb4 définit le jeu de caractères par défaut du serveur MariaDB en utf8mb4. Cela signifie que toutes les données stockées dans la base de données MariaDB utiliseront ce jeu de caractères.

La ligne [client] default-character-set = utf8mb4 spécifie le jeu de caractères par défaut utilisé par le client MariaDB. Cela signifie que toutes les requêtes envoyées au serveur MariaDB par le client utiliseront le jeu de caractères utf8mb4.

En utilisant le jeu de caractères utf8mb4, on peut stocker des caractères de tous les langues et symboles dans la base de données MariaDB.



The screenshot shows a terminal window titled "khadijaoussakel@controller:/home/khadijaoussakel". The window contains the following configuration file content:

```
Fichier Édition Affichage Rechercher Terminal Aide
# create new

# set default charset

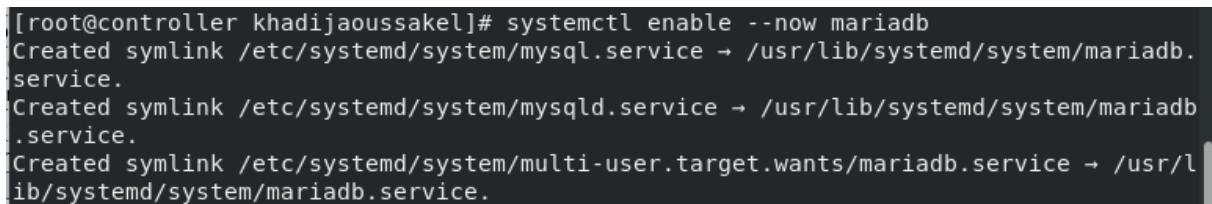
# if not set, default is [latin1]

# for the case of 4 bytes UTF-8, specify [utf8mb4]

[mysqld]
character-set-server = utf8mb4

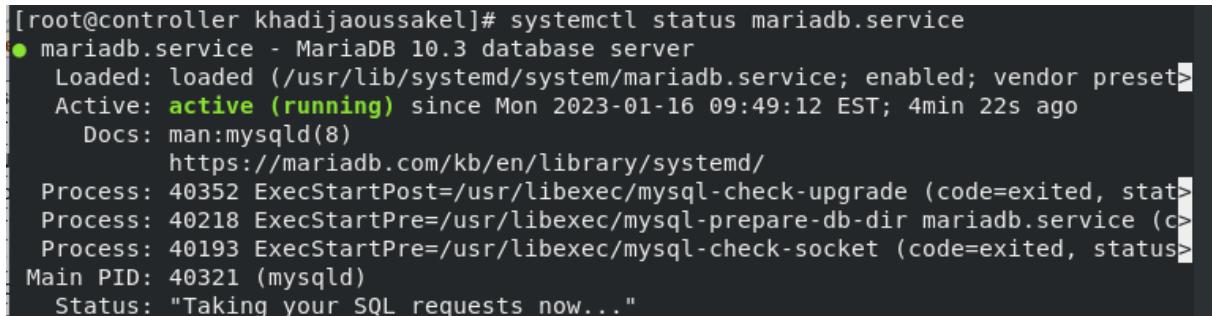
[client]
default-character-set = utf8mb4
```

On utilise la commande "systemctl enable --now mariadb" qui permet d'activer et de démarrer le service MariaDB :



```
[root@controller khadijaoussakel]# systemctl enable --now mariadb
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/mariadb.service.
```

Pour vérifier que le service MariaDB est actif et fonctionne normalement, on utilise la commande "systemctl status mariadb.service" qui permet d'afficher l'état du service MariaDB :



```
[root@controller khadijaoussakel]# systemctl status mariadb.service
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-01-16 09:49:12 EST; 4min 22s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 40352 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0)
   Process: 40218 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.service (code=exited, status=0)
   Process: 40193 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0)
 Main PID: 40321 (mysqld)
    Status: "Taking your SQL requests now..."
```

On tape "mysql_secure_installation" pour effectuer des tâches de sécurité de base sur le système MariaDB nouvellement installé. Cela va nous garantir que le serveur de base de données est correctement sécurisé :

```
[root@controller khadijaoussakel]# mysql_secure_installation

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.
```

En activant et en démarrant le service MariaDB, on peut accéder à la base de données et effectuer des opérations sur les données stockées dans celle-ci :

```
[root@controller khadijaoussakel]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 20
Server version: 10.3.28-MariaDB MariaDB Server

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)]> 
```

On peut afficher la liste des utilisateurs :

```
MariaDB [(none)]> select user,host,password from mysql.user;
+-----+-----+
| user | host      | password                         |
+-----+-----+
| root | localhost | *0A22508A21ECD305B83BF95241BBAA40BE49102A |
| root | 127.0.0.1  | *0A22508A21ECD305B83BF95241BBAA40BE49102A |
| root | ::1        | *0A22508A21ECD305B83BF95241BBAA40BE49102A |
+-----+-----+
3 rows in set (0.000 sec)

MariaDB [(none)]>
```

On peut afficher aussi la liste de base de données existantes :

```
MariaDB [(none)]> show databases;
+-----+
| Database          |
+-----+
| information_schema |
| mysql              |
| performance_schema |
+-----+
3 rows in set (0.001 sec)
```

On exécute une séquence de commandes pour tester MariaDB :

```
MariaDB [(none)]> create database test_database;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> create table test_database.test_table (id int, name varchar(50),
address varchar(50), primary key (id));
Query OK, 0 rows affected (0.007 sec)

MariaDB [(none)]> insert into test_database.test_table(id, name, address) values("0
01", "CentOS", "Hiroshima");
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]>

MariaDB [(none)]> select * from test_database.test_table;
+---+-----+-----+
| id | name   | address |
+---+-----+-----+
| 1  | CentOS | Hiroshima |
+---+-----+-----+
1 row in set (0.001 sec)

MariaDB [(none)]> drop database test_database;
Query OK, 1 row affected (0.006 sec)

MariaDB [(none)]> exit
Bye
```

On ajoute le référentiel d'Openstack Victoria et on met également à niveau le système CentOS :

Le référentiel d'OpenStack Victoria se compose de tous les composants logiciels inclus dans la version Victoria, tels que Nova (le composant de calcul), Neutron (le composant de réseau), Cinder (le composant de stockage de blocs), Glance (le composant d'image), Keystone (le composant d'identité), et bien d'autres encore.

```
[root@controller khadijaoussakel]# dnf -y install centos-release-openstack-victoria

Dernière vérification de l'expiration des métadonnées effectuée il y a 0:25:54 le l
un. 16 janv. 2023 09:34:32 EST.
Dépendances résolues.
=====
 Paquet          Architecture      Version       Dépôt     Taille
=====
Installing:
 centos-release-openstack-victoria      noarch    1-2.el8      extras    10 k
Installation des dépendances:
 centos-release-advanced-virtualization  noarch    1.0-4.el8    extras    16 k
 centos-release-ceph-nautilus           noarch    1.2-2.el8    extras    8.8 k
 centos-release-messaging              noarch    1-2.el8      extras    9.4 k
 centos-release-nfv-common             noarch    1-3.el8      extras    9.3 k
 centos-release-nfv-openvswitch        noarch    1-3.el8      extras    8.6 k
 centos-release-rabbitmq-38            noarch    1-2.el8      extras    8.3 k
 centos-release-storage-common         noarch    2-2.el8      extras    9.4 k
 centos-release-virt-common            noarch    1-2.el8      extras    8.9 k

[root@controller khadijaoussakel]# sed -i -e "s/enabled=1/enabled=0/g" /etc/yum.
repos.d/CentOS-OpenStack-victoria.repo
```

En particulier, on doit mettre à niveau certains packages Python3 à partir du référentiel Openstack Victoria :

```
[root@controller khadijaoussakel]# sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
[root@controller khadijaoussakel]# sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-*
[root@controller khadijaoussakel]# sudo dnf -y install centos-release-ceph-nautilus
CentOS-8 - Advanced Virtualization           273 kB/s | 255 kB     00:00
CentOS-8 - Ceph Nautilus                     735 kB/s | 549 kB     00:00
CentOS Linux 8 - AppStream                   3.9 MB/s | 8.4 MB    00:02
CentOS Linux 8 - BaseOS                      3.7 MB/s | 4.6 MB    00:01
CentOS Linux 8 - Extras                      13 kB/s | 10 kB     00:00
CentOS-8 - RabbitMQ 38                       227 kB/s | 137 kB     00:00
CentOS-8 - NFV OpenvSwitch                  79 kB/s | 67 kB     00:00
Le paquet centos-release-ceph-nautilus-1.2-2.el8.noarch est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
```

```
[root@controller khadijaoussakel]# dnf --enablerepo=centos-openstack-victoria -y upgrade
CentOS-8 - OpenStack victoria          2.7 MB/s | 3.7 MB     00:01
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:02 le lun. 16 janv. 2023 10:29:03 EST.
Dépendances résolues.
=====
Paquet          Architecture      Version       Dépôt        Taille
=====
Mise à jour:
librados2        x86_64 2:14.2.22-2.el8  centos-ceph-nautilus   3.5 M
librbd1          x86_64 2:14.2.22-2.el8  centos-ceph-nautilus   1.7 M
libzstd           x86_64 1.4.5-6.el8    centos-openstack-victoria 335 k
python3-beautifulsoup4 noarch 4.9.1-2.el8  centos-openstack-victoria 214 k
python3-cffi       x86_64 1.13.2-1.el8  centos-openstack-victoria 245 k
python3-cssselect noarch 0.9.2-13.el8   centos-openstack-victoria 40 k
python3-decorator noarch 4.4.0-5.el8    centos-openstack-victoria 32 k
python3-pexpect    noarch 4.7.0-4.el8    centos-openstack-victoria 144 k
python3-nvutil     x86_64 5.7.2-1.el8    centos-openstack-victoria 419 k
```

On passe maintenant à l'installation des paquets "rabbitmq-server" et "memcached" sur le système, ces paquets fournissent respectivement un serveur de messagerie ainsi qu'un système de mise en cache de données en mémoire vive :

- **RabbitMQ est un logiciel de messagerie open source qui permet aux applications de communiquer entre elles de manière asynchrone. Il utilise un protocole de messagerie standard appelé Advanced Message Queuing Protocol (AMQP).**
- **Memcached est un système de cache en mémoire distribué open source qui permet d'accélérer les applications Web en stockant temporairement des données fréquemment utilisées dans la mémoire principale du serveur. Cela permet de réduire les temps de réponse en évitant les accès fréquents à la base de données, qui sont souvent plus lents.**

```
[root@controller khadijaoussakel]# dnf --enablerepo=powertools -y install rabbitmq-server memcached
CentOS Linux 8 - PowerTools                               458 kB/s | 2.3 MB   00:05
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:04 le dim. 26 févr. 2023 03:16:32 +01.
Dépendances résolues.
=====
Paquet          Architecture      Version       Dépôt     Taille
=====
Installation:
memcached      x86_64        1.5.22-2.el8    appstream   162 k
rabbitmq-server x86_64        3.8.3-1.el8    centos-rabbitmq-38 11 M
Installation des dépendances:
```

On ouvre le fichier de configuration de MariaDB Server "mariadb-server.cnf" :

```
[root@controller khadijaoussakel]# vi /etc/my.cnf.d/mariadb-server.cnf
```

Dans la directive "max_connections" dans la section "[mysqld]", on définit le nombre maximal de connexions simultanées autorisées pour les clients MySQL à 500 :

```
# instructions in http://fedoraproject.org/wiki/Systemd
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mariadb/mariadb.log
pid-file=/run/mariadb/mariadb.pid
max_connections=500
```

On ouvre le fichier /etc/sysconfig/memcached qui est un fichier de configuration pour le service Memcached :

```
[root@controller khadijaoussakel]# vi /etc/sysconfig/memcached
```

Ce fichier est utilisé pour définir les variables d'environnement qui seront utilisées par le service Memcached :

```
Fichier Édition Affichage Rechercher Terminal Aide
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS="-l 0.0.0.0,::"
```

On utilise la commande "systemctl restart" qui redémarre les services de MariaDB, RabbitMQ Server et Memcached. Cela permet de forcer le redémarrage des services et de s'assurer qu'ils sont opérationnels :

```
[khadijaoussakel@controller ~]$ sudo systemctl restart mariadb rabbitmq-server memcached
[sudo] Mot de passe de khadijaoussakel :
```

On vérifie que le status de tous ces services :

```
[khadijaoussakel@controller ~]$ systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ broker
  Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; disabled; vendor>
  Active: active (running) since Wed 2023-01-25 08:29:08 EST; 3s ago
    Process: 11627 ExecStop=/bin/sh -c while ps -p $MAINPID >/dev/null 2>&1; do s>
    Process: 11357 ExecStop=/usr/lib/rabbitmq/bin/rabbitmqctl stop (code=exited, >
```

```
[khadijaoussakel@controller ~]$ systemctl status memcached.service
● memcached.service - memcached daemon
  Loaded: loaded (/usr/lib/systemd/system/memcached.service; disabled; vendor >
  Active: active (running) since Wed 2023-01-25 08:28:08 EST; 31min ago
    Main PID: 11549 (memcached)
      Tasks: 10 (limit: 23520)
        Memory: 3.9M
       CGroup: /system.slice/memcached.service
               └─11549 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l 0.>
```

```
[khadijaoussakel@controller ~]$ systemctl status mariadb.service
● mariadb.service - MariaDB 10.3 database server
  Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor pre>
  Active: active (running) since Wed 2023-01-25 08:28:09 EST; 31min ago
    Docs: man:mysqld(8)
          https://mariadb.com/kb/en/library/systemd/
   Process: 11591 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, s>
  Process: 11508 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mariadb.servic>
```

Maintenant on tape la commande "systemctl enable" qui active les services de MariaDB, RabbitMQ Server et Memcached au démarrage du système. Cela permet de s'assurer que les services seront automatiquement démarrés au démarrage du système et que les utilisateurs pourront y accéder dès que le système sera en ligne :

```
[khadijaoussakel@controller ~]$ systemctl enable mariadb rabbitmq-server memcached
Created symlink /etc/systemd/system/multi-user.target.wants/rabbitmq-server.service → /usr/lib/systemd/system/rabbitmq-server.service.
Created symlink /etc/systemd/system/multi-user.target.wants/memcached.service → /usr/lib/systemd/system/memcached.service.
[khadijaoussakel@controller ~]$
```

La première commande, "rabbitmqctl add_user openstack password", crée un nouvel utilisateur dans RabbitMQ appelé "openstack" avec un mot de passe "password".

La deuxième commande, "rabbitmqctl set_permissions openstack ". " ." ." "", définit les autorisations pour l'utilisateur "openstack" dans le vhost "/" (qui est le vhost par défaut de RabbitMQ). Les autorisations sont définies pour "." pour les trois paramètres - config, write et read, ce qui signifie que l'utilisateur "openstack" est autorisé à configurer, écrire et lire dans tous les canaux et files d'attente de RabbitMQ dans le vhost "/" :

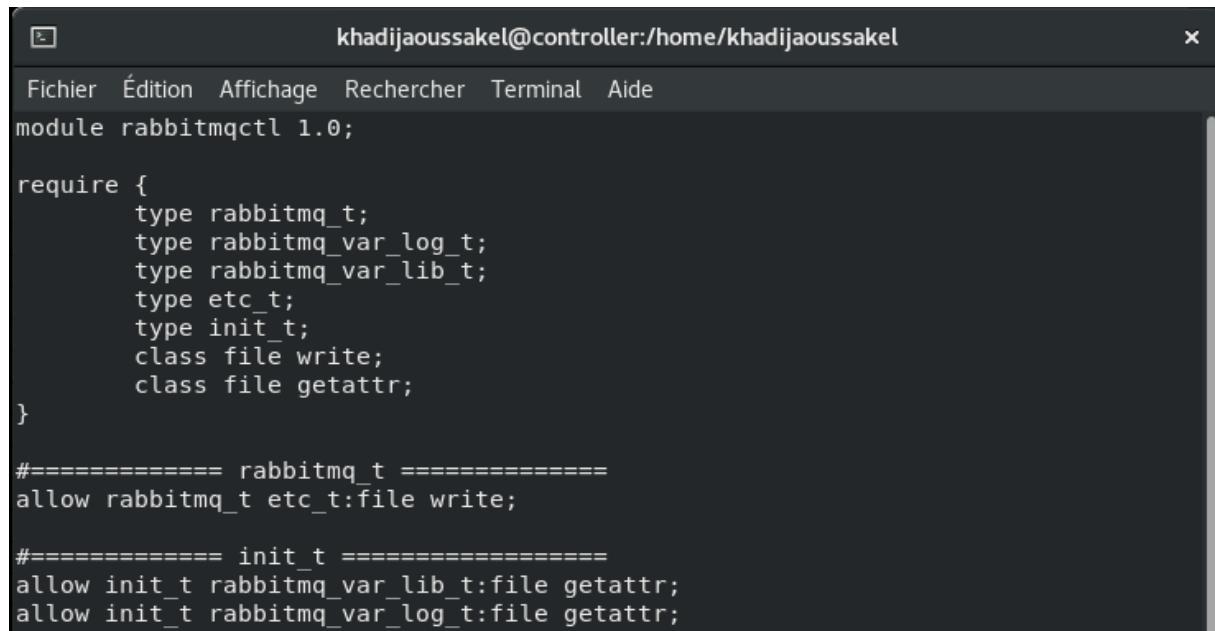
```
[root@controller khadijaoussakel]# rabbitmqctl add_user openstack password
Adding user "openstack" ...
[root@controller khadijaoussakel]# rabbitmqctl set_permissions openstack ".*" "."
Setting permissions for user "openstack" in vhost "/" ...
```

SELinux est « enabled », on doit donc changer les politiques de sécurité pour autoriser le service RabbitMQ à accéder à certaines ressources système nécessaires à son fonctionnement.

SELinux est un système de sécurité pour Linux qui permet de restreindre l'accès aux ressources système en fonction de politiques de sécurité.

```
[root@controller ~]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                  enforcing
Mode from config file:         enforcing
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Memory protection checking:   actual (secure)
Max kernel policy version:    33
```

Sur le fichier « rabbitmqctl.te » on ajoute le script suivant qui commence par définir un module SELinux nommé "rabbitmqctl" avec une version de 1.0. Il définit ensuite les autorisations requises pour le module. Plus précisément, il autorise le type "rabbitmq_t" à écrire dans des fichiers de type "etc_t", ainsi qu'à accéder aux fichiers de type "rabbitmq_var_lib_t" et "rabbitmq_var_log_t" à partir du type "init_t" :



```
khadijaoussakel@controller:/home/khadijaoussakel
Fichier Édition Affichage Rechercher Terminal Aide
module rabbitmqctl 1.0;

require {
    type rabbitmq_t;
    type rabbitmq_var_log_t;
    type rabbitmq_var_lib_t;
    type etc_t;
    type init_t;
    class file write;
    class file getattr;
}

===== rabbitmq_t =====
allow rabbitmq_t etc_t:file write;

===== init_t =====
allow init_t rabbitmq_var_lib_t:file getattr;
allow init_t rabbitmq_var_log_t:file getattr;
```

Ensuite on utilise les commandes "checkmodule", "semodule_package" et "semodule" pour créer un module SELinux compilé et l'installer sur le système, en utilisant le fichier de politique précédemment défini :

```
[root@controller khadijaoussakel]# vi rabbitmqctl.te
[root@controller khadijaoussakel]# checkmodule -m -M -o rabbitmqctl.mod rabbitmq
ctl.te
[root@controller khadijaoussakel]# semodule_package --outfile rabbitmqctl.pp --m
odule rabbitmqctl.mod
[root@controller khadijaoussakel]# semodule -i rabbitmqctl.pp
[root@controller khadijaoussakel]#
```

III.Configuration de keystone

#1

Keystone est un service d'identification et d'authentification centralisé pour OpenStack. Il est utilisé pour gérer les identités des utilisateurs, les informations d'authentification et les autorisations d'accès pour les services OpenStack. Keystone permet également de gérer les projets et les domaines, qui sont des unités organisationnelles clés dans OpenStack.



On exécute cette série de commandes qui créent une nouvelle base de données MariaDB appelée "keystone" et accordent tous les privilèges à l'utilisateur "keystone" pour accéder à cette base de données à partir de n'importe quelle adresse IP ou localement depuis "localhost" :

- "create database keystone;" crée une nouvelle base de données appelée "keystone".
- "grant all privileges on keystone.* to keystone@'localhost' identified by 'password';" accorde tous les privilèges sur la base de données "keystone" à l'utilisateur "keystone" avec le mot de passe "password" pour les connexions locales à partir de "localhost".
- "grant all privileges on keystone.* to keystone@'%' identified by 'password';" accorde tous les privilèges sur la base de données "keystone" à l'utilisateur "keystone" avec le mot de passe "password" pour les connexions à partir de n'importe quelle adresse IP.
- "flush privileges;" met à jour les privilèges pour que les modifications effectuées prennent effet immédiatement.

```
[root@controller khadijaoussakel]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.3.28-MariaDB MariaDB Server

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 (0.018 sec)

MariaDB [(none)]> grant all privileges on keystone.* to keystone@'localhost' identified by 'password';
Query OK, 0 rows affected (0.028 sec)

MariaDB [(none)]> grant all privileges on keystone.* to keystone@'%' identified by 'password';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.028 sec)

MariaDB [(none)]> exit
Bye
```

La commande "dnf --enablerepo=centos-openstack-victoria,epel,powertools -y install openstack-keystone python3-openstackclient httpd mod_ssl python3-mod_wsgi python3-oauth2client" installe plusieurs paquets logiciels :

- openstack-keystone : il s'agit d'un service d'identification et d'authentification centralisé pour les services OpenStack.
- python3-openstackclient : c'est un outil de ligne de commande pour interagir avec les différents services OpenStack.
- httpd : c'est un serveur web Apache HTTP Server.
- mod_ssl : c'est un module Apache pour prendre en charge les connexions SSL / TLS.

- `python3-mod_wsgi` : c'est un module Python pour Apache qui permet d'exécuter des applications web Python.
- `python3-oauth2client` : c'est une bibliothèque Python pour interagir avec des services qui utilisent le protocole OAuth 2.0.

```
[root@controller khadijaoussakel]# dnf --enablerepo=centos-openstack-victoria,epel,powertools -y install openstack-keystone python3-openstackclient httpd mod_ssl python3-mod_wsgi python3-oauth2client
CentOS Linux 8 - PowerTools           7.1 kB/s | 4.3 kB     00:00
CentOS-8 - OpenStack victoria       3.5 kB/s | 3.0 kB     00:00
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:00:01 le mer. 25 janv. 2023 09:17:51 EST.
Dépendances résolues.
=====
Paquet          Architecture      Version      Dépôt      Taille
=====
Installation:
httpd           x86_64 2.4.37-43.module_el8.5.0+1022+b541f3b1
                  appstream      1.4 M
mod_ssl         x86_64 1:2.4.37-43.module_el8.5.0+1022+b541f3b1
                  appstream      136 k
openstack-keystone noarch 1:18.0.0-1.el8 centos-openstack-victoria 83 k
python3-mod_wsgi   x86_64 4.6.4-4.el8    appstream      2.5 M
python3-oauth2client noarch 4.1.3-9.el8   epel        149 k
python3-openstackclient noarch 5.4.0-2.el8 centos-openstack-victoria 1.1 M
Installation des dépendances:
apr             x86_64 1.6.3-12.el8    appstream      129 k
=====
python3-yappi-1.2.5-1.el8.x86_64
python3-zipp-0.5.1-3.el8.noarch
qpidd-proton-c-0.37.0-1.el8.x86_64

Terminé !
[root@controller khadijaoussakel]#
```

On ouvre le fichier de configuration "keystone.conf" à l'aide de l'éditeur de texte "gedit" :

```
[root@controller khadijaoussakel]# gedit /etc/keystone/keystone.conf
```

La ligne 440 est modifiée pour spécifier le serveur Memcache utilisé par Keystone. Le serveur spécifié est "10.0.0.30:11211" :

```
# IPv4 (list value)
#memcache_servers = localhost:11211
memcache_servers = 10.0.0.30:11211
# Number of seconds memcached server is considered dead before it is tried
```

La ligne 615 est modifiée pour spécifier les informations de connexion à la base de données MariaDB utilisée par Keystone. Les informations de connexion spécifiées sont "mysql+pymysql://keystone:password@10.0.0.30/keystone" :

```
#connection = <None>
connection = mysql+pymysql://keystone:password@controller/keystone
```

La ligne 2499 est décommentée pour spécifier le fournisseur de jetons utilisé par Keystone. Le fournisseur de jetons spécifié est "fernet" :

```
# IDs. (string value)
provider = fernet
```

Pour configurer le service Keystone, on utilise les commandes suivantes :

- La commande « su -s /bin/bash keystone -c "keystone-manage db_sync" » pour synchroniser la base de données de Keystone avec les derniers schémas de base de données.
- La commande « keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone » configure Keystone pour utiliser le système de chiffrement Fernet pour la gestion des jetons d'authentification. Cela implique la création de clés de chiffrement et leur stockage dans le système.
- La commande « keystone-manage credential_setup --keystone-user keystone --keystone-group keystone » configure Keystone pour stocker les informations d'identification des utilisateurs dans la base de données de Keystone. Cela implique la création des tables de base de données nécessaires pour stocker les informations d'identification.

```
[root@controller khadijaoussakel]# su -s /bin/bash keystone -c "keystone-manage db_sync"
[root@controller khadijaoussakel]# keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
[root@controller khadijaoussakel]# keystone-manage credential_setup --keystone-user keystone --keystone-group keystone
[root@controller khadijaoussakel]# export controller=10.0.0.30
[root@controller khadijaoussakel]#
```

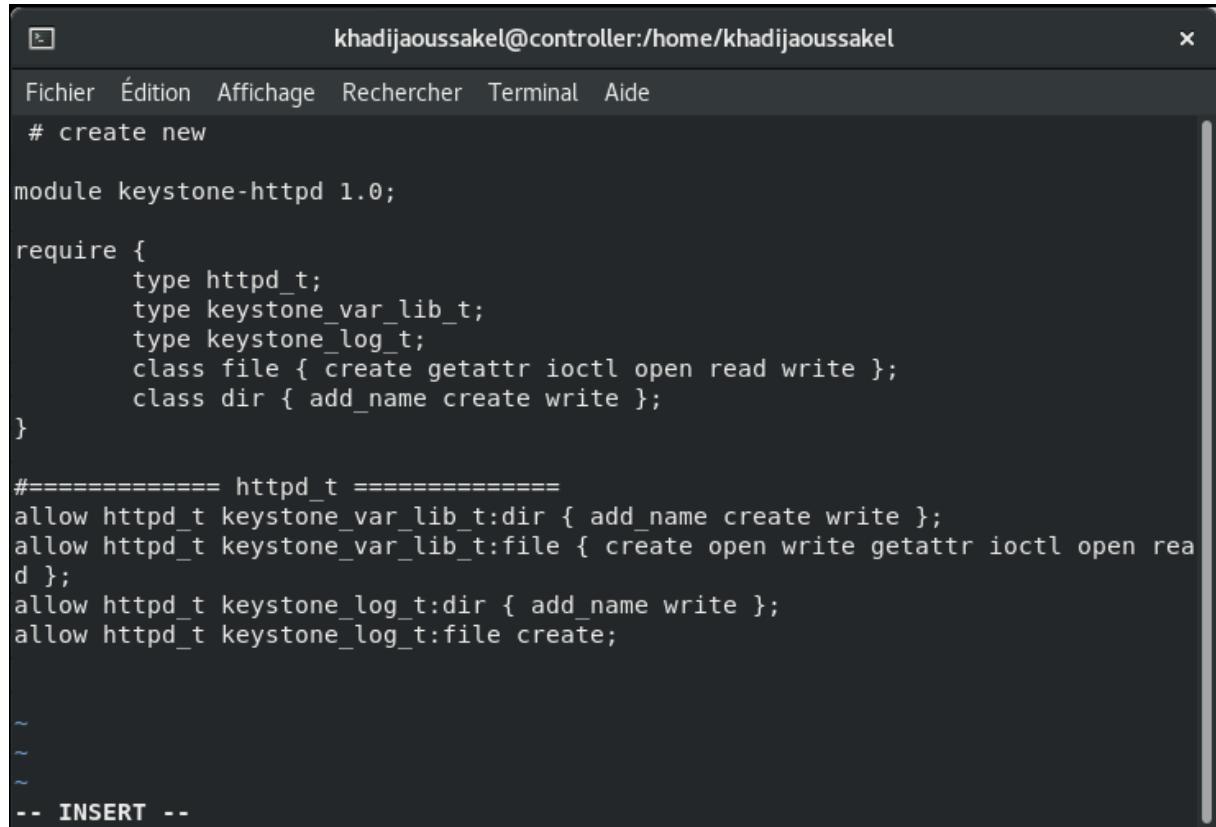
Puis on initialise Keystone en créant un compte d'administration Keystone et en créant les services et les points de terminaison Keystone.

```
[root@controller khadijaoussakel]# keystone-manage bootstrap --bootstrap-password adminpassword \
> --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
[root@controller khadijaoussakel]#
```

Et puisque SELinux est activé, on doit configurer les paramètres de sécurité SELinux pour le service web Apache HTTPD, lorsqu'il est utilisé pour se connecter à des ressources OpenStack telles que la base de données Keystone. Les commandes setsebool définissent les paramètres booléens qui permettent à Apache HTTPD de se connecter à OpenStack. Le paramètre -P spécifie que les changements doivent être persistants même après un redémarrage :

```
[root@controller khadijaoussakel]# setsebool -P httpd_use_openstack on
[root@controller khadijaoussakel]# setsebool -P httpd_can_network_connect on
[root@controller khadijaoussakel]# setsebool -P httpd_can_network_connect_db on
[root@controller khadijaoussakel]#
```

On édite le fichier nommé keystone-httpd.te. Il s'agit d'un fichier de module de sécurité SELinux qui définit les autorisations nécessaires pour que le service web Apache HTTPD puisse accéder aux ressources de Keystone :



```
khadijaoussakel@controller:/home/khadijaoussakel
Fichier Édition Affichage Rechercher Terminal Aide
# create new

module keystone-httpd 1.0;

require {
    type httpd_t;
    type keystone_var_lib_t;
    type keystone_log_t;
    class file { create getattr ioctl open read write };
    class dir { add_name create write };
}

===== httpd_t =====
allow httpd_t keystone_var_lib_t:dir { add_name create write };
allow httpd_t keystone_var_lib_t:file { create open write getattr ioctl open read };
allow httpd_t keystone_log_t:dir { add_name write };
allow httpd_t keystone_log_t:file create;

~
~
~
-- INSERT --
```

Ce module est créé en utilisant la commande checkmodule pour compiler le fichier keystone-httpd.te, puis en utilisant la commande semodule_package pour créer un package de module SELinux nommé keystone-httpd.pp. Enfin, la commande semodule est utilisée pour installer le module de sécurité SELinux dans le système :

```
[root@controller khadijaoussakel]# checkmodule -m -M -o keystone-httpd.mod keystone-httpd.te
[root@controller khadijaoussakel]# semodule_package --outfile keystone-httpd.pp --module keystone-httpd.mod
[root@controller khadijaoussakel]# semodule -i keystone-httpd.pp
```

On ouvre le fichier de configuration principal d'Apache HTTP Server (httpd.conf) :

```
[root@controller khadijaoussakel]# gedit /etc/httpd/conf/httpd.conf
(gedit:4831): GLib-GIO-CRITICAL **: 11:02:35.709: g_dbus_proxy_new_sync: assertion failed: (error == NULL)
```

On modifie la valeur de la directive "ServerName" dans le fichier httpd.conf :

```
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName controller
```

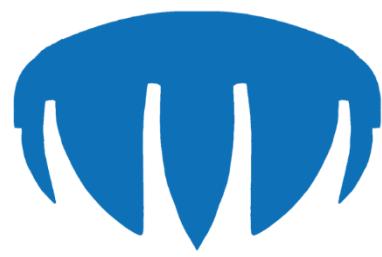
« `ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/` »: Cette commande crée un lien symbolique vers le fichier de configuration de l'interface WSGI de Keystone (`wsgi-keystone.conf`) dans le répertoire de configuration d'Apache (`/etc/httpd/conf.d/`). Cela permet à Apache de charger la configuration de Keystone lorsqu'il est démarré.

On active et démarre le service Apache HTTP Server. L'option `enable` configure le service pour démarrer automatiquement au démarrage du système, tandis que l'option `--now` démarre immédiatement le service :

```
[root@controller khadijaoussakel]# ln -s /usr/share/keystone/wsgi-keystone.conf  
/etc/httpd/conf.d/  
[root@controller khadijaoussakel]# systemctl enable --now httpd  
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr  
/lib/systemd/system/httpd.service.  
[root@controller khadijaoussakel]# systemctl status httpd.service  
● httpd.service - The Apache HTTP Server  
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor prese  
   Active: active (running) since Sat 2023-01-28 11:04:53 EST; 16s ago  
     Docs: man:httpd.service(8)  
 Main PID: 5009 (httpd)  
    Status: "Running, listening on: port 5000, port 443, port 80"  
       Tasks: 233 (limit: 23520)
```

IV. Configuration de keystone #2

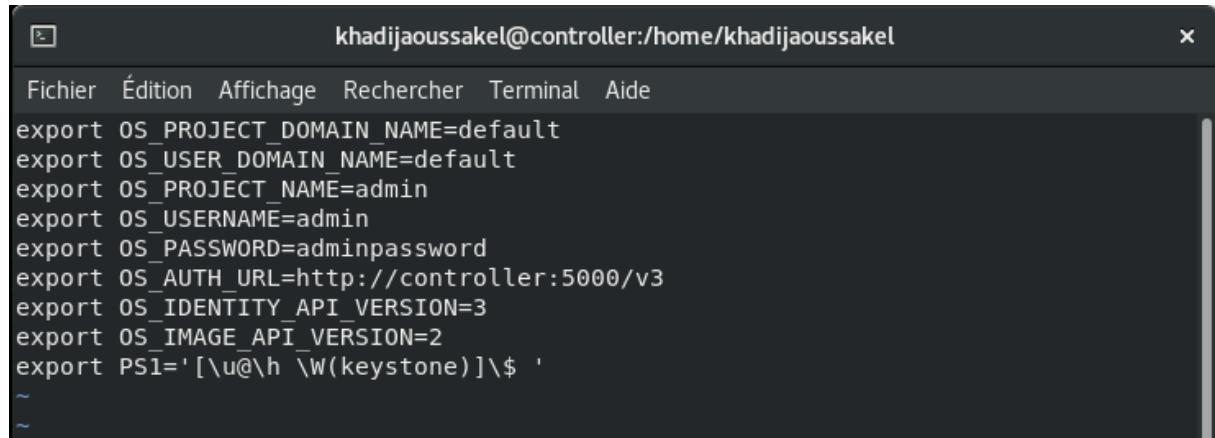
Keystone est un service d'identification et d'authentification centralisé pour OpenStack. Il est utilisé pour gérer les identités des utilisateurs, les informations d'authentification et les autorisations d'accès pour les services OpenStack. Keystone permet également de gérer les projets et les domaines, qui sont des unités organisationnelles clés dans OpenStack.



On va configurer l'environnement d'authentification pour accéder à une installation OpenStack en éditant le fichier suivant :

```
[root@controller khadijaoussakel]# vi ~/keystonerc
```

Ces lignes exportent des variables d'environnement qui spécifient les informations d'identification pour accéder à l'API OpenStack, telles que le nom de projet, le nom d'utilisateur, le mot de passe et l'URL d'authentification. Ces variables sont utilisées par les outils en ligne de commande OpenStack tels que nova et glance :



The screenshot shows a terminal window titled "khadijaoussakel@controller:/home/khadijaoussakel". The window contains the following text:

```
Fichier Édition Affichage Rechercher Terminal Aide
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=adminpassword
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export PS1='[\u@\h \W(keystone)]\$ '
```

Ensuite on utilise ces commandes :

- La commande chmod 600 ~/keystonerc est utilisée pour définir les autorisations de lecture et d'écriture uniquement pour l'utilisateur actuel sur le fichier ~/keystonerc qui contient les informations d'identification d'OpenStack.
- La commande source ~/keystonerc charge les variables d'environnement définies dans le fichier ~/keystonerc dans l'environnement actuel de la ligne de commande.
- Enfin, la commande echo "source ~/keystonerc " >> ~/.bash_profile ajoute la commande source ~/keystonerc au fichier de profil bash de l'utilisateur actuel pour que les variables d'environnement soient automatiquement chargées à chaque ouverture d'une nouvelle session de terminal.

```
[root@controller khadijaoussakel]# vi ~/keystonerc
[root@controller khadijaoussakel]# chmod 600 ~/keystonerc
[root@controller khadijaoussakel]# source ~/keystonerc
[root@controller khadijaoussakel(keystone)]# echo "source ~/keystonerc " >> ~/.bash_profile
[root@controller khadijaoussakel(keystone)]#
```

On crée un projet appelé "service" dans le domaine par défaut avec une description "Service Project" :

```
[root@controller khadijaoussakel(keystone)]# openstack project create --domain default --description "Service Project" service
+-----+-----+
| Field      | Value   |
+-----+-----+
| description | Service Project |
| domain_id  | default  |
| enabled     | True    |
| id          | 6fd19e7348054e229e9788ce6446924f |
| is_domain   | False   |
| name        | service  |
| options     | {}      |
| parent_id   | default  |
| tags        | []      |
+-----+
[root@controller khadijaoussakel(keystone)]#
```

Ici on affiche la liste des projets :

```
[root@controller khadijaoussakel(keystone)]# openstack project list
+-----+-----+
| ID            | Name   |
+-----+-----+
| 6fd19e7348054e229e9788ce6446924f | service |
| 8c267e475f444a73a2715e71ea5c3f1b | admin   |
+-----+-----+
```

V. Configuration de Glance

Dans OpenStack Victoria, Glance est le service d'image. Il permet de stocker, de découvrir et de récupérer des images de machines virtuelles (VM) et de disques durs virtuels (VDI). La configuration de Glance dans OpenStack Victoria est importante car elle permet de définir les paramètres d'authentification et de stockage pour les images.



On crée un utilisateur nommé "glance" avec un mot de passe "servicepassword" dans le domaine "default" et le projet "service" du système OpenStack :

```
[root@controller khadijaoussakel(keystone)]# openstack user create --domain default --project service --password servicepassword glance
+-----+-----+
| Field      | Value
+-----+-----+
| default_project_id | 6fd19e7348054e229e9788ce6446924f
| domain_id     | default
| enabled        | True
| id             | b4b5a2ee12aa458b93671255c3868586
| name           | glance
| options         | {}
| password_expires_at | None
+-----+-----+
```

On ajoute l'utilisateur "glance" au rôle "admin" dans le projet "service". Cela signifie que l'utilisateur "glance" aura des priviléges de niveau administrateur dans le projet "service" :

```
[root@controller khadijaoussakel(keystone)]# openstack role add --project service --user glance admin
```

On crée une entrée de service nommée "glance" avec une description "OpenStack Image service" et un type de service "image". Le type de service "image" est utilisé pour stocker et gérer les images de machine virtuelle utilisées dans OpenStack. Cette entrée de service sera utilisée pour fournir des fonctionnalités de stockage et de gestion d'images dans OpenStack :

```
[root@controller khadijaoussakel(keystone)]# openstack service create --name glance --description "OpenStack Image service" image
+-----+-----+
| Field      | Value
+-----+-----+
| description | OpenStack Image service
| enabled     | True
| id          | 92c2d7a4a3f64124974849e6986d2f7d
| name        | glance
| type        | image
+-----+-----+
```

La commande "export controller=10.0.0.30" définit une variable d'environnement appelée "controller" avec l'adresse IP "10.0.0.30". Cette variable sera utilisée dans la commande "openstack endpoint create" pour créer un endpoint public (point d'accès) pour l'API Glance :

```
[root@controller khadijaoussakel(keystone)]# export controller=10.0.0.30
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne image public http://$controller:9292
+-----+-----+
| Field      | Value
+-----+-----+
| enabled    | True
| id          | 1b26ec96faf54a47969f3014371cd46f
| interface   | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | 92c2d7a4a3f64124974849e6986d2f7d
| service_name| glance
| service_type| image
| url         | http://10.0.0.30:9292
+-----+-----+
```

Ensuite on crée un point d'accès interne pour le service d'image dans la région "RegionOne" :

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne image internal http://$controller:9292
+-----+-----+
| Field      | Value
+-----+-----+
| enabled     | True
| id          | b8843c634e214513b0bc748469ab9ef9
| interface   | internal
| region      | RegionOne
| region_id   | RegionOne
| service_id  | 92c2d7a4a3f64124974849e6986d2f7d
| service_name| glance
| service_type| image
| url         | http://10.0.0.30:9292
```

On crée un autre point d'accès (endpoint) dans OpenStack pour le service "image" (ou "glance") avec le rôle administrateur ("admin") :

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne image admin http://$controller:9292
+-----+-----+
| Field      | Value
+-----+-----+
| enabled     | True
| id          | eb7606e004024f70a61901527b9a1adc
| interface   | admin
| region      | RegionOne
| region_id   | RegionOne
| service_id  | 92c2d7a4a3f64124974849e6986d2f7d
| service_name| glance
| service_type| image
| url         | http://10.0.0.30:9292
```

On crée une base de données appelée "glance", et on accorde tous les privilèges sur la base de données "glance" à l'utilisateur "glance" avec le mot de passe "password", à la fois pour la connexion locale et distante, et de vider les privilèges en cours :

```
[root@controller khadijaoussakel(keystone)]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.3.28-MariaDB MariaDB Server

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 glance;
Query OK, 1 row affected (0.015 sec)

MariaDB [(none)]> grant all privileges on glance.* to glance@'localhost' identified by 'password';
Query OK, 0 rows affected (0.066 sec)

MariaDB [(none)]> grant all privileges on glance.* to glance@'%' identified by 'password';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> flush privileges;
```

On installe le paquet "openstack-glance" :

"openstack-glance" est un service de stockage d'images dans l'écosystème OpenStack qui permet de stocker et de partager des images de machines virtuelles pour une utilisation avec des instances de machines virtuelles.

```
[root@controller khadijaoussakel(keystone)]# dnf --enablerepo=centos-openstack-victoria,powertools,epel -y install openstack-glance
CentOS Linux 8 - PowerTools                               5.1 kB/s | 4.3 kB     00:00
CentOS-8 - OpenStack victoria                           4.5 kB/s | 3.0 kB     00:00
Dépendances résolues.

=====
 Paquet          Architecture      Version       Dépôt        Taille
=====
 Installation:
  openstack-glance    noarch 1:21.1.0-1.el8  centos-openstack-victoria  78 k
 Installation des dépendances:
  blosc            x86_64 1.17.0-2.el8    epel           229 k
```

On va modifier le fichier de configuration "glance-api.conf" du service Glance :

```
[root@controller khadijaoussakel(keystone)]# mv /etc/glance/glance-api.conf /etc/glance/glance-api.conf.org
[root@controller khadijaoussakel(keystone)]# vi /etc/glance/glance-api.conf
```

Les modifications apportées au fichier de configuration concernent principalement la configuration de la base de données MariaDB et de l'authentification Keystone pour Glance.

La configuration de la base de données MariaDB est définie dans la section "database", où la connexion est établie avec l'utilisateur "glance" et le mot de passe "password" pour la base de données "glance" hébergée sur le serveur "10.0.0.30".

La configuration de l'authentification Keystone est définie dans la section "keystone_auth_token", où les informations d'authentification sont fournies pour l'authentification des demandes reçues par Glance, telles que l'URL d'authentification, les informations d'identification de l'utilisateur, le nom de domaine et le projet associé. La section "paste_deploy" définit la méthode d'intégration de Keystone avec le service Glance.

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[DEFAULT]
bind_host = 0.0.0.0

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

[database]
# MariaDB connection info
connection = mysql+pymysql://glance:password@controller/glance

# keystone auth info
[keystone_auth_token]
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 = servicepassword
-- INSERT --
```

Les commandes effectuent les opérations suivantes :

- "chmod 640 /etc/glance/glance-api.conf" définit les permissions pour le fichier "glance-api.conf" en autorisant l'utilisateur root à lire et à écrire le fichier, tandis que le groupe "glance" peut uniquement le lire.
- "chown root:glance /etc/glance/glance-api.conf" modifie le propriétaire et le groupe du fichier "glance-api.conf" pour root et glance respectivement.
- "su -s /bin/bash glance -c "glance-manage db_sync"" exécute la commande "glance-manage db_sync" en tant qu'utilisateur "glance" en utilisant le shell bash. Cette commande synchronise la base de données Glance avec le schéma de la base de données.

```
[root@controller khadijaoussakel(keystone)]# chmod 640 /etc/glance/glance-api.co
nf
[root@controller khadijaoussakel(keystone)]# chown root:glance /etc/glance/glanc
e-api.conf
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash glance -c "glance-m
anage db_sync"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade  -> liberty, liberty initial
INFO [alembic.runtime.migration] Running upgrade liberty -> mitaka01, add index
on created_at and updated_at columns of 'images' table
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
Database is synced successfully.
```

On active et démarre le service "openstack-glance-api". L'option "--now" est utilisée pour démarrer le service immédiatement après son activation :

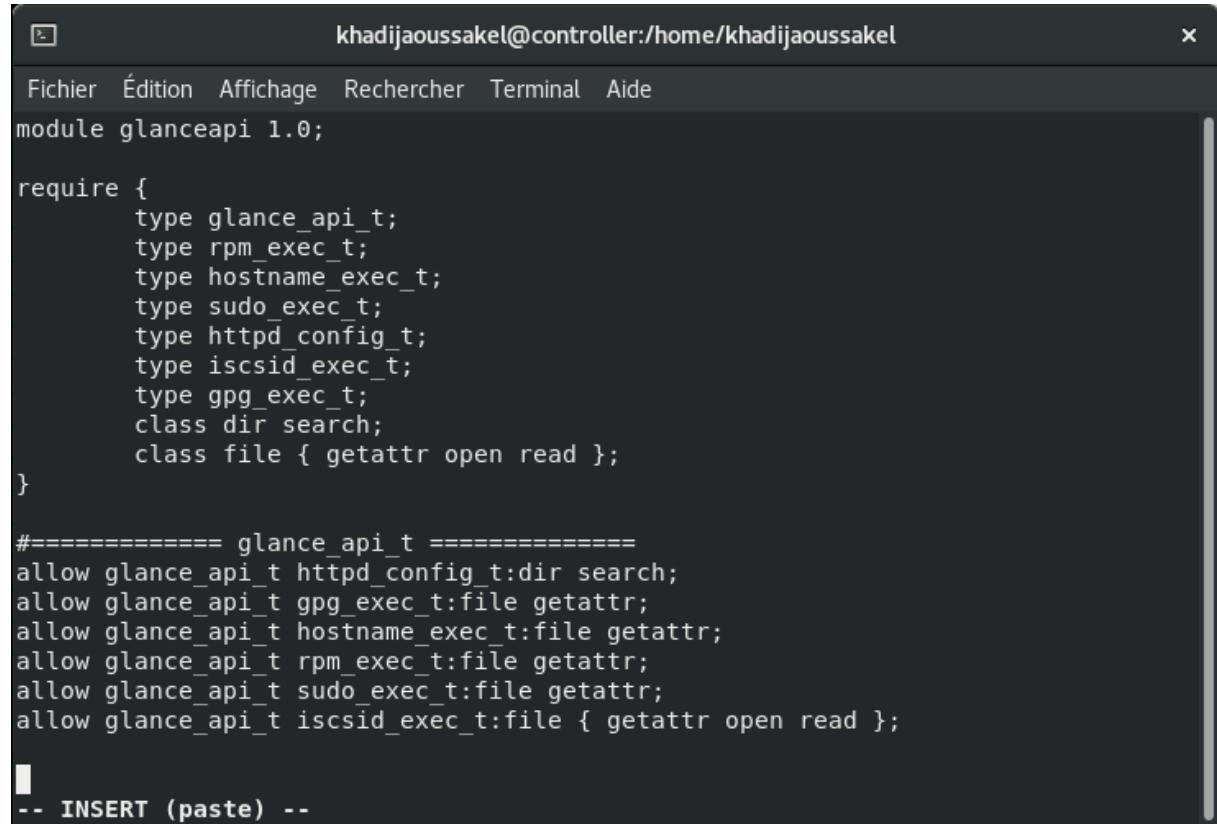
```
[root@controller khadijaoussakel(keystone)]# systemctl enable --now openstack-gl
ance-api
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-glance-api
.service → /usr/lib/systemd/system/openstack-glance-api.service.
[root@controller khadijaoussakel(keystone)]# systemctl status openstack-glance-a
pi.service
● openstack-glance-api.service - OpenStack Image Service (code-named Glance) AP>
   Loaded: loaded (/usr/lib/systemd/system/openstack-glance-api.service; enable>
   Active: active (running) since Sat 2023-01-28 15:46:48 EST; 27s ago
     Main PID: 8459 (glance-api)
        Tasks: 5 (limit: 23520)
       Memory: 116.8M
```

Puisque SELinux est activé, on active le paramètre de booléen "glance_api_can_network" pour l'application "glance_api" :

```
[root@controller khadijaoussakel(keystone)]# setsebool -P glance_api_can_network
on
[root@controller khadijaoussakel(keystone)]# vi glanceapi.te
```

On ajoute cette politique de sécurité qui autorise l'application "glance_api" à effectuer des opérations spécifiques sur des fichiers et des répertoires de types "httpd_config_t", "gpg_exec_t", "hostname_exec_t", "rpm_exec_t", "sudo_exec_t" et "iscsid_exec_t". Plus précisément, l'application "glance_api" est autorisée à rechercher des répertoires de type "httpd_config_t", à accéder en lecture aux fichiers de type "gpg_exec_t", à obtenir les

attributs de fichiers de type "hostname_exec_t", "rpm_exec_t" et "sudo_exec_t" et à obtenir les attributs, ouvrir et lire les fichiers de type "iscsid_exec_t" :



The screenshot shows a terminal window titled "khadijaoussakel@controller:/home/khadijaoussakel". The window contains SELinux policy code for the "glanceapi" module. The code defines types for glance_api_t, rpm_exec_t, hostname_exec_t, sudo_exec_t, httpd_config_t, and iscsid_exec_t. It also defines classes for dir search and file operations. A section for "glance_api_t" permissions is shown, allowing various operations like getattr and open/read on these types. At the bottom of the terminal, there is a prompt "-- INSERT (paste) --" indicating where to paste the policy code.

```
module glanceapi 1.0;

require {
    type glance_api_t;
    type rpm_exec_t;
    type hostname_exec_t;
    type sudo_exec_t;
    type httpd_config_t;
    type iscsid_exec_t;
    type gpg_exec_t;
    class dir search;
    class file { getattr open read };
}

===== glance_api_t =====
allow glance_api_t httpd_config_t:dir search;
allow glance_api_t gpg_exec_t:file getattr;
allow glance_api_t hostname_exec_t:file getattr;
allow glance_api_t rpm_exec_t:file getattr;
allow glance_api_t sudo_exec_t:file getattr;
allow glance_api_t iscsid_exec_t:file { getattr open read };

-- INSERT (paste) --
```

Enfin on tape ces commandes qui permettent de compiler, d'empaqueter et d'installer une politique de sécurité SELinux personnalisée pour l'application "glance_api" :

```
[root@controller khadijaoussakel(keystone)]# checkmodule -m -M -o glanceapi.mod glanceapi.te
[root@controller khadijaoussakel(keystone)]# semodule_package --outfile glanceapi.pp --module glanceapi.mod
[root@controller khadijaoussakel(keystone)]# semodule -i glanceapi.pp
```

VI. Ajout des images de la machine virtuelle

En ajoutant des images de machines virtuelles, les utilisateurs peuvent créer des instances de machines virtuelles à partir de ces images, leur permettant de déployer rapidement des environnements de serveur pour des applications spécifiques. Cela peut être utile pour les tests de développement, la mise à l'échelle de l'infrastructure ou pour déployer des applications dans le cloud.



On installe les outils de virtualisation suivants :

- qemu-kvm : est un hyperviseur qui permet de créer et de gérer des machines virtuelles. Il fournit également des outils pour la migration en direct, la sauvegarde et la restauration des machines virtuelles.
- libvirt : est une bibliothèque d'API pour la gestion de la virtualisation. Il permet aux applications de contrôler et de gérer les hyperviseurs tels que KVM, Xen et LXC.
- virt-install : est un outil en ligne de commande qui permet de créer et d'installer des machines virtuelles. Il peut être utilisé pour configurer des machines virtuelles à partir de fichiers de configuration XML ou en utilisant des options de ligne de commande.

```
[root@controller khadijaoussakel]# dnf -y install qemu-kvm libvirt virt-install
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:03:57 le lun. 06 févr. 2023 08:08:33 EST.
Le paquet qemu-kvm-15:6.0.0-33.el8.x86_64 est déjà installé.
Dépendances résolues.
=====
Paquet          Architecture      Version       Dépôt           Taille
=====
Installation:
libvirt          x86_64 7.6.0-6.el8 centos-advanced-virtualization 74 k
virt-install      noarch 2.2.1-4.el8 appstream            100 k
Installation des dépendances:
libvirt-client   x86_64 7.6.0-6.el8 centos-advanced-virtualization 384 k
libvirt-daemon-config-nwfilter
```



```
Installé:
libvirt-7.6.0-6.el8.x86_64
libvirt-client-7.6.0-6.el8.x86_64
libvirt-daemon-config-nwfilter-7.6.0-6.el8.x86_64
python3-argcomplete-1.9.3-6.el8.noarch
python3-libvirt-7.6.0-1.el8.x86_64
virt-install-2.2.1-4.el8.noarch
virt-manager-common-2.2.1-4.el8.noarch

Terminé !
```

Avec la commande "lsmod | grep kvm", on confirme que les modules nécessaires pour la virtualisation avec KVM sont chargés dans le noyau Linux :

```
[root@controller khadijaoussakel]# lsmod | grep kvm
kvm_intel          323584  0
kvm                 880640  1 kvm_intel
irqbypass          16384  1 kvm
[root@controller khadijaoussakel]#
```

On active et démarre le service "libvirtd", qui est le démon de gestion de la virtualisation pour libvirt :

```
[root@controller khadijaoussakel]# systemctl enable libvirtd && systemctl start libvirtd
Created symlink /etc/systemd/system/sockets.target.wants/libvirtd.socket → /usr/lib/systemd/system/libvirtd.socket.
Created symlink /etc/systemd/system/sockets.target.wants/libvirtd-ro.socket → /usr/lib/systemd/system/libvirtd-ro.socket.
```

On vérifie que le service libvirtd a été bien activé :

```
[root@controller khadijaoussakel]# systemctl status libvirtd.service
● libvirtd.service - Virtualization daemon
  Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor pr>
  Active: active (running) since Mon 2023-02-06 09:05:00 EST; 1min 46s ago
    Docs: man:libvirtd(8)
          https://libvirt.org
   Main PID: 6468 (libvirtd)
     Tasks: 21 (limit: 32768)
    Memory: 46.3M
   CGroup: /system.slice/libvirtd.service
           ├─2411 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf
           ├─2412 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf
           └─6468 /usr/sbin/libvirtd --timeout 120
```

On installe deux outils de gestion de la virtualisation :

- libguestfs-tools : est un ensemble d'outils en ligne de commande pour la gestion des images de disque et des systèmes de fichiers dans les environnements virtualisés. Il permet de manipuler les images de disque virtuelles (par exemple, pour ajouter ou supprimer des fichiers) sans avoir besoin de démarrer la machine virtuelle à laquelle l'image appartient.
- virt-top : est un outil en ligne de commande pour surveiller et afficher les performances des machines virtuelles. Il fournit des informations en temps réel sur l'utilisation des ressources système, telles que le CPU, la mémoire et les E/S de disque.

```
[root@controller ~]# dnf -y install libguestfs-tools virt-top
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:57:23 le dim. 26 févr. 2023 11:27:03 +01.
Dépendances résolues.
=====
Paquet          Architecture      Version       Dépôt          Taille
=====
Installation:
libguestfs-tools      noarch 1:1.44.0-3.el8 centos-advanced-virtualization 28 k
virt-top            x86_64 1.0.8-32.el8  appstream        729 k
Installation des dépendances:
hexedit             x86_64 1.2.13-12.el8  appstream          46 k
hivex               x86_64 1.3.18-22.el8 centos-advanced-virtualization 114 k
libguestfs          x86_64 1:1.44.0-3.el8 centos-advanced-virtualization 2.8 M
libguestfs-tools-c  x86_64 1:1.44.0-3.el8 centos-advanced-virtualization 5.3 M
```

On crée le répertoire "/var/kvm/images", puis on utilise la commande "virt-builder centos-8.2 --format qcow2 --size 10G -o /var/kvm/images/centos8.qcow2 --root-password password" qui télécharge une image officielle de CentOS 8.2, puis la transforme au format qcow2 et la stocke dans le répertoire "/var/kvm/images" avec le nom "centos8.qcow2". Cette image sera utilisée pour créer une machine virtuelle CentOS 8.2. L'option "--size" définit la taille du disque virtuel à 10G, et l'option "--root-password" définit le mot de passe root de la machine virtuelle à "password" :

```
[root@controller ~]# mkdir -p /var/kvm/images
[root@controller ~]# virt-builder centos-8.2 --format qcow2 --size 10G
-o /var/kvm/images/centos8.qcow2 --root-password password
[ 4.8] Downloading: http://builder.libguestfs.org/centos-8.2.xz
#####
[ 674.6] Planning how to build this image
[ 674.6] Uncompressing
[ 697.1] Resizing (using virt-resize) to expand the disk to 10.0G
[ 858.1] Opening the new disk
[ 874.6] Setting a random seed
[ 874.7] Setting passwords
[ 877.7] Finishing off
          Output file: /var/kvm/images/centos8.qcow2
          Output size: 10.0G
          Output format: qcow2
          Total usable space: 9.3G
          Free space: 8.0G (85%)
```

On crée et installe une nouvelle machine virtuelle avec les options suivantes :

```
[root@controller khadijaoussakel]# virt-install \
> --name centos-8.2 \
> --ram 2048 \
> --disk path=/var/kvm/images/centos8.qcow2 \
> --vcpus 2 \
> --os-type linux \
> --os-variant rhel8.2 \
> --network network=default \
> --graphics none \
> --serial pty \
> --console pty \
> --boot hd \
[    0.000000] Linux version 4.18.0-193.6.3.el8_2.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)) #1 SMP Wed Jun 10 11:09:32 UTC 2020
[    0.000000] Command line: BOOT_IMAGE=(hd0,gpt2)/vmlinuz-4.18.0-193.6.3.el8_2.x86_64 root=UUID=4fd120e4-1f6d-46b3-a404-5569ef6af1f9 ro console=tty0 rd_NO_PLYMOUTH crashkernel=auto resume=UUID=40f14688-2619-4046-a9eb-b7333fff1b84 console=ttyS0,115200
```

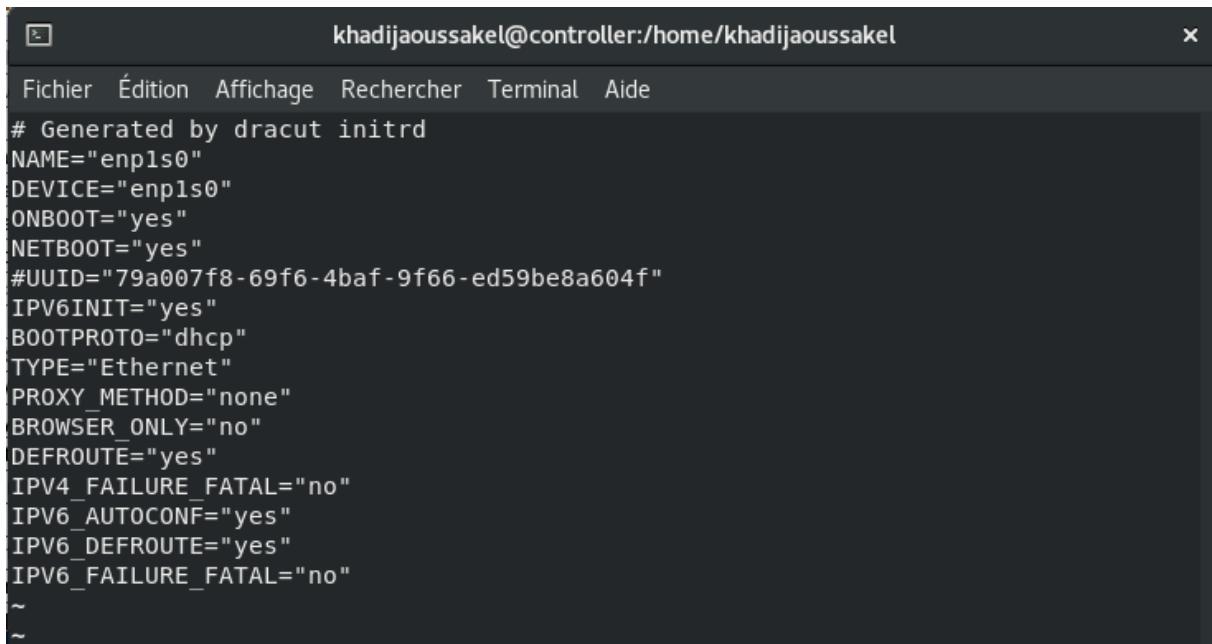
On ajoute un nouvel utilisateur nommé "centos" au système et un mot de passe pour ce nouvel utilisateur "centos" :

```
localhost login: root
[root@localhost ~]# passwd root
Changing password for user root.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# useradd centos
[root@localhost ~]# passwd centos
Changing password for user centos.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

On ouvre le fichier de configuration pour l'interface réseau "enp1s0" pour permettre la modification des paramètres réseau :

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-enp1s0
```

On met ensuite la ligne UUID en commentaire et on change le BOOTPROTO à dhcp :



The screenshot shows a terminal window with the title "khadijaoussakel@controller:/home/khadijaoussakel". The window contains the contents of the /etc/sysconfig/network-scripts/ifcfg-enp1s0 file. The file is a configuration script for the enp1s0 interface. It includes parameters like NAME, DEVICE, ONBOOT, NETBOOT, IPV6INIT, BOOTPROTO, TYPE, PROXY_METHOD, BROWSER_ONLY, DEFROUTE, IPV4_FAILURE_FATAL, IPV6_AUTOCONF, IPV6_DEFROUTE, and IPV6_FAILURE_FATAL. The UUID line is preceded by a hash symbol (#), indicating it is a comment. The BOOTPROTO line is set to "dhcp". There are two tilde characters (~) at the bottom of the file.

```
# Generated by dracut initrd
NAME="enp1s0"
DEVICE="enp1s0"
ONBOOT="yes"
NETBOOT="yes"
#UUID="79a007f8-69f6-4baf-9f66-ed59be8a604f"
IPV6INIT="yes"
BOOTPROTO="dhcp"
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
~
~
```

On crée une image publique dans OpenStack à partir d'un fichier de disque CentOS 8 et on la rend disponible pour une utilisation ultérieure par tous les utilisateurs autorisés :

```
[root@controller ~]# openstack image create "CentOS8" --file /var/kvm/images/centos8.qcow2 --disk-format qcow2 --container-format bare --public
+-----+
| Field          | Value
|               |
+-----+
| container_format | bare
| created_at      | 2023-02-25T20:18:15Z
| disk_format     | qcow2
| file            | /v2/images/41a478e1-b1db-4b60-8c83-9721d56c5084/file
```

A la fin on affiche la liste d'images de machine virtuelle qui ont été créées dans l'infrastructure OpenStack pour vérifier que l'image est prête à être utilisée pour créer une instance de machine virtuelle dans OpenStack :

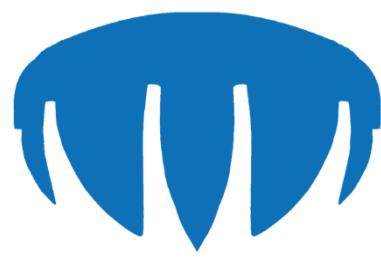
```
[root@controller ~]# openstack image list
+-----+-----+-----+
| ID      | Name    | Status |
+-----+-----+-----+
| 41a478e1-b1db-4b60-8c83-9721d56c5084 | CentOS8 | active |
+-----+-----+-----+
```

VII. Configuration de Nova

#1

Nova est le composant central de la plateforme OpenStack qui permet de transformer des instances offertes par “Glance” à des machines virtuelles provisionnées sur les différents nœuds de calcul disponibles dans le cloud. Pour ce faire, Nova s'appuie sur plusieurs services OpenStack tels que Keystone, Glance, Neutron et Placement.

Placement, que nous allons aussi configurer dans cette partie, est le service qui suit l'état des ressources disponibles dans le cloud, notamment les ressources de calcul et de stockage, et qui aide à choisir le fournisseur de ces ressources qui sera utilisé lors de la création d'une machine virtuelle.



On crée tout d'abord un utilisateur [nova] dans le projet [service] et on lui attribue le rôle [admin]

```
[root@controller khadijaoussakel(keystone)]# openstack user create --domain default --project service --password servicepassword nova
+-----+-----+
| Field | Value |
+-----+-----+
| default_project_id | 6fd19e7348054e229e9788ce6446924f |
| domain_id | default |
| enabled | True |
| id | 3edcdd18aff1410085a3be8392408057 |
| name | nova |
| options | {} |
| password_expires_at | None |
+-----+-----+
```

```
[root@controller khadijaoussakel(keystone)]# openstack role add --project service --user nova admin
```

On crée également un utilisateur [placement] dans le projet [service] et on lui attribue le rôle [admin]

```
[root@controller khadijaoussakel(keystone)]# openstack user create --domain default --project service --password servicepassword placement
+-----+-----+
| Field | Value |
+-----+-----+
| default_project_id | 6fd19e7348054e229e9788ce6446924f |
| domain_id | default |
| enabled | True |
| id | 73773ce42d4146d2905268c54be6a590 |
| name | placement |
| options | {} |
| password_expires_at | None |
+-----+-----+
```

```
[root@controller khadijaoussakel(keystone)]# openstack role add --project service --user placement admin
```

Ensuite, on crée une entrée de service pour [nova] de même que pour [emplacement]

Dans le contexte d'OpenStack Nova, une entrée de service identifie l'emplacement et les détails du service Nova qui est utilisé pour gérer les instances de machines virtuelles dans l'environnement OpenStack.

```
[root@controller khadijaoussakel(keystone)]# openstack service create --name nova --description "OpenStack Compute service" compute
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Compute service |
| enabled | True |
| id | a2780fd3653e462eac2c3009900f920e |
| name | nova |
| type | compute |
+-----+-----+
```

```
[root@controller khadijaoussakel(keystone)]# openstack service create --name placement --description "OpenStack Compute Placement service" placement
+-----+-----+
| Field      | Value
+-----+-----+
| description | OpenStack Compute Placement service
| enabled     | True
| id          | b0c2f2cd9e4641e59f865f85d03b6118
| name        | placement
| type        | placement
+-----+-----+
```

Puis, on définit l'hôte de l'API Nova et on crée un endpoint pour l'API [Nova] publique, qui est accessible depuis l'extérieur du réseau OpenStack dans la région "RegionOne"

```
[root@controller khadijaoussakel(keystone)]# export controller=10.0.0.30
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne compute public http://$controller:8774/v2.1/%(tenant_id)s
+-----+-----+
| Field      | Value
+-----+-----+
| enabled    | True
| id         | 121c008fd3e34e9884890a0c5b9aa7a0
| interface   | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | a2780fd3653e462eac2c3009900f920e
| service_name| nova
| service_type| compute
| url        | http://10.0.0.30:8774/v2.1/%(tenant_id)s
+-----+-----+
```

On crée aussi un endpoint pour l'API interne de [Nova] qui est utilisé pour gérer les instances de machines virtuelles dans l'environnement OpenStack.

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne compute internal http://$controller:8774/v2.1/%(tenant_id)s
+-----+-----+
| Field      | Value
+-----+-----+
| enabled    | True
| id         | 6900de1db7cb4a64ad3d41c032299109
| interface   | internal
| region     | RegionOne
| region_id  | RegionOne
| service_id | a2780fd3653e462eac2c3009900f920e
| service_name| nova
| service_type| compute
| url        | http://10.0.0.30:8774/v2.1/%(tenant_id)s
+-----+-----+
```

Et on crée un endpoint pour l'API administrative de [Nova] qui est utilisé pour effectuer des tâches d'administration sur l'environnement OpenStack, telles que la création de projets ou la gestion des utilisateurs.

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region
RegionOne compute admin http://$controller:8774/v2.1/%\$(tenant_id)\$s
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | 4fa5c997dc3d4e508ac77c80d91174d8
| interface | admin
| region | RegionOne
| region_id | RegionOne
| service_id | a2780fd3653e462eac2c3009900f920e
| service_name | nova
| service_type | compute
| url | http://10.0.0.30:8774/v2.1/%(tenant_id)s
+-----+-----+
```

De même pour [placement], on crée un endpoint pour l'API publique, interne et administrative.

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region
RegionOne placement public http://$controller:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | 019467ac38b24f1cad2ac60f295eaf08
| interface | public
| region | RegionOne
| region_id | RegionOne
| service_id | b0c2f2cd9e4641e59f865f85d03b6118
| service_name | placement
| service_type | placement
| url | http://10.0.0.30:8778
+-----+-----+
```

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region
RegionOne placement internal http://$controller:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | a3b6d79d64d1431eb8851b5b2b04a3ec
| interface | internal
| region | RegionOne
| region_id | RegionOne
| service_id | b0c2f2cd9e4641e59f865f85d03b6118
| service_name | placement
| service_type | placement
| url | http://10.0.0.30:8778
+-----+-----+
```

```
[root@controller khadijaoussakel(keystone)]# openstack endpoint create --region RegionOne placement admin http://$controller:8778
+-----+-----+
| Field      | Value          |
+-----+-----+
| enabled     | True           |
| id          | 7047802e6ca34c748880b1161fa06541 |
| interface   | admin          |
| region      | RegionOne      |
| region_id   | RegionOne      |
| service_id  | b0c2f2cd9e4641e59f865f85d03b6118 |
| service_name| placement      |
| service_type| placement      |
| url         | http://10.0.0.30:8778 |
+-----+-----+
```

On exécute la commande ***openstack user list*** afin de visualiser les utilisateurs [nova] et [placement] qu'on vient d'ajouter dans le catalogue d'identité Keystone.

```
[root@controller ~](keystone)]# openstack user list
+-----+-----+
| ID            | Name          |
+-----+-----+
| 1afb8abe33a84b79917d8842290add48 | admin         |
| b4b5a2ee12aa458b93671255c3868586 | glance        |
| 3edcdd18aff1410085a3be8392408057 | nova          |
| 73773ce42d4146d2905268c54be6a590 | placement     |
+-----+-----+
```

Maintenant on exécute les commandes SQL suivantes en tant que root, pour créer plusieurs bases de données et accorder des priviléges d'accès à ces bases de données à des utilisateurs spécifiques. Plus exactement, les bases de données créées sont les suivantes : nova, nova_api, nova_cell0 et placement. Les priviléges d'accès à ces bases de données sont accordés à des utilisateurs spécifiques : "nova" pour les bases de données "nova", "nova_api" et "nova_cell0", et "placement" pour la base de données "placement".

```
[root@controller khadijaoussakel(keystone)]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 15
Server version: 10.3.28-MariaDB MariaDB Server

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 nova;
Query OK, 1 row affected (0.014 sec)

MariaDB [(none)]> grant all privileges on nova.* to nova@'localhost' identified
by 'password';
Query OK, 0 rows affected (0.027 sec)

MariaDB [(none)]> grant all privileges on nova.* to nova@'%' identified by 'pass
word';
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> create database nova_api;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> grant all privileges on nova_api.* to nova@'localhost' identified by 'password';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> grant all privileges on nova_api.* to nova@'%' identified by 'password';
Query OK, 0 rows affected (0.001 sec)
```

```
MariaDB [(none)]> create database nova_cell0;
Query OK, 1 row affected (0.013 sec)

MariaDB [(none)]> grant all privileges on nova_cell0.* to nova@'localhost' identified by 'password';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> grant all privileges on nova_cell0.* to nova@'%' identified by 'password';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> create database placement;
Query OK, 1 row affected (0.015 sec)

MariaDB [(none)]> grant all privileges on placement.* to placement@'localhost' identified by 'password';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> grant all privileges on placement.* to placement@'%' identified by 'password';
Query OK, 0 rows affected (0.000 sec)

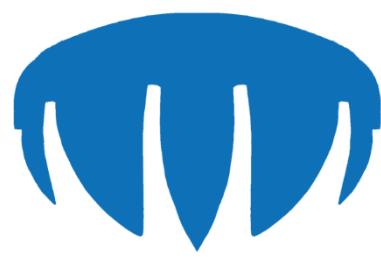
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.026 sec)
```

VIII. Configuration de Nova

#2

Nova est le composant central de la plateforme OpenStack qui permet de transformer des instances offertes par “Glance” à des machines virtuelles provisionnées sur les différents nœuds de calculs disponibles dans le cloud. Pour ce faire, Nova s'appuie sur plusieurs services OpenStack tels que Keystone, Glance, Neutron et Placement.

Placement, que nous allons aussi configurer dans cette partie, est le service qui suit l'état des ressources disponibles dans le cloud, notamment les ressources de calcul et de stockage, et qui aide à choisir le fournisseur de ces ressources qui sera utilisé lors de la création d'une machine virtuelle.



On commence par l'installation des services Nova, en effet : L'installation de ces paquets est utile pour les déploiements OpenStack, car "openstack-nova" est le composant de calcul d'OpenStack et "openstack-placement-api" est un service d'allocation de ressources pour OpenStack.

```
[root@controller khadijaoussakel(keystone)]# dnf --enablerepo=centos-openstack-victoria,powertools,epel -y install openstack-nova openstack-placement-api
CentOS Linux 8 - PowerTools           8.4 kB/s | 4.3 kB   00:00
CentOS-8 - OpenStack victoria       5.5 kB/s | 3.0 kB   00:00
Dépendances résolues.
=====
Paquet          Architecture      Version      Dépôt      Taille
=====
Installation:
openstack-nova      noarch 1:22.2.2-1.el8 centos-openstack-victoria 7.3 k
openstack-placement-api noarch 4.0.0-1.el8    centos-openstack-victoria 12 k
Installation des dépendances:
ipmitool           x86_64 1.8.18-18.el8 appstream      395 k
iptables-services  x86_64 1.8.4-20.el8  baseos        63 k
libsodium           x86_64 1.0.18-2.el8  centos-ceph-nautilus 163 k
network-scripts     x86_64 10.00.15-1.el8 baseos        196 k
network-scripts-openvswitch2.13
=====
python3-websockify-0.10.0-3.el8.noarch
python3-zake-0.2.2-18.el8.noarch
rdo-openvswitch-1:2.13-3.el8.noarch
=====
Terminé !
```

Premièrement, on conserve une copie de sauvegarde "nova.conf.org" du fichier original "nova.conf" dans le répertoire "/etc/nova" au cas où quelque chose se passerait mal lors de la modification du fichier.

```
[root@controller khadijaoussakel(keystone)]# mv /etc/nova/nova.conf /etc/nova/nova.conf.org
[root@controller khadijaoussakel(keystone)]# vi /etc/nova/nova.conf
```

Après avoir ouvert le fichier nova.conf, on effectue des modifications de sorte que :

- La ligne "my_ip" est modifiée pour définir l'adresse IP de l'hôte Nova à 10.0.0.30.
- Les informations de connexion RabbitMQ et de Glance sont définies dans "transport_url" et "api_servers" respectivement
- Les informations de connexion MariaDB sont définies dans "connection" sous les sections [api_database] et [database]
- Les informations d'authentification Keystone sont définies dans [keystone_auth_token], y compris le nom du projet, le nom d'utilisateur, le mot de passe et l'URL d'authentification...

```

Fichier Édition Affichage Rechercher Terminal Aide
[DEFAULT]
# define own IP address
my_ip = 10.0.0.30
state_path = /var/lib/nova
enabled_apis = osapi_compute,metadata
log_dir = /var/log/nova
# RabbitMQ connection info
transport_url = rabbit://openstack:password@10.0.0.30

[api]
auth_strategy = keystone

# Glance connection info
[glance]
api_servers = http://10.0.0.30:9292

[oslo_concurrency]
lock_path = $state_path/tmp

# MariaDB connection info
[api_database]
connection = mysql+pymysql://nova:password@10.0.0.30/nova_api

```

Afin de changer les autorisations et la propriété de groupe du fichier **/etc/nova/nova.conf**, on utilise les 2 commandes suivantes :

- ***chmod 640 /etc/nova/nova.conf*** : Cela définit les autorisations de fichier sur 640, ce qui signifie que le propriétaire a des autorisations de lecture et d'écriture, et que le groupe a des autorisations de lecture. Les autres utilisateurs n'ont aucune autorisation.
- ***chgrp nova /etc/nova/nova.conf*** : Cela définit le groupe de propriété du fichier sur "nova". Cela signifie que les membres du groupe "nova" auront des autorisations spécifiques sur ce fichier, comme défini par les autorisations de fichier définies précédemment avec la commande chmod.

```
[root@controller khadijaoussakel(keystone)]# chmod 640 /etc/nova/nova.conf
[root@controller khadijaoussakel(keystone)]# chgrp nova /etc/nova/nova.conf
```

Pour ce qui est de la configuration du fichier **placement.conf**, on suit les mêmes étapes qu'on a fait précédemment au niveau du fichier **nova.conf**: on conserve une copie de sauvegarde "**placement.conf.org**" du fichier original "**placement.conf**" dans le répertoire "/etc/placement". Ensuite, on change les autorisations et la propriété de groupe du fichier **/etc/placement/placement.conf**

```
[root@controller khadijaoussakel(keystone)]# mv /etc/placement/placement.conf /etc/placement/placement.conf.org
[root@controller khadijaoussakel(keystone)]# vi /etc/placement/placement.conf
```

```

Fichier Édition Affichage Rechercher Terminal Aide
[DEFAULT]
debug = false

[api]
auth_strategy = keystone

[keystone_auth_token]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = placement
password = servicepassword

[placement_database]
connection = mysql+pymysql://placement:password@10.0.0.30/placement

```

-- INSERT --

```

[root@controller khadijaoussakel(keystone)]# chmod 640 /etc/placement/placement.conf
[root@controller khadijaoussakel(keystone)]# chgrp placement /etc/placement/placement.conf

```

Ainsi, on modifie le fichier "/etc/httpd/conf.d/00-placement-api.conf" en ajoutant une nouvelle section "Directory" qui spécifie que l'accès au répertoire "/usr/bin" est autorisé pour tous les utilisateurs (Require all granted).

```

[root@controller khadijaoussakel(keystone)]# gedit /etc/httpd/conf.d/00-placement-api.conf

```

```

Listen 8778

<VirtualHost *:8778>
    WSGIProcessGroup placement-api
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    WSGIDaemonProcess placement-api processes=3 threads=1 user=placement group=placement
    WSGIScriptAlias / /usr/bin/placement-api
    <IfVersion >= 2.4>
        ErrorLogFormat "%M"
    </IfVersion>
    ErrorLog /var/log/placement/placement-api.log
    #SSLEngine On
    #SSLCertificateFile ...
    #SSLCertificateKeyFile ...
    <Directory /usr/bin>
        Require all granted
    </Directory>
</VirtualHost>

```

A ce niveau on installe et on configure les politiques SELinux OpenStack.

On installe le package openstack-selinux depuis le dépôt centos-openstack-victoria.

```
[root@controller khadijaoussakel(keystone)]# dnf --enablerepo=centos-openstack-victoria -y install openstack-selinux
Extra Packages for Enterprise Linux 8 - x86_64 2.0 kB/s | 45 kB     00:22
Dépendances résolues.

=====
Paquet          Architecture      Version       Dépôt           Taille
=====
Installation:
  openstack-selinux    noarch    0.8.27-1.el8   centos-openstack-victoria 224 k

Résumé de la transaction
=====
Installer 1 Paquet

Installé:
  openstack-selinux-0.8.27-1.el8.noarch

Terminé !
```

On ajoute une règle pour permettre au port TCP 8778 d'être utilisé avec le type http_port_t.

```
[root@controller khadijaoussakel(keystone)]# semanage port -a -t http_port_t -p
tcp 8778
[root@controller khadijaoussakel(keystone)]# vi novaapi.te
```

Après avoir ouvert le fichier novaapi.te, on le modifie pour permettre la création d'un nouveau module SELinux.

```
Fichier Édition Affichage Rechercher Terminal Aide
module novaapi 1.0;

require {
    type rpm_exec_t;
    type hostname_exec_t;
    type nova_t;
    type gpg_exec_t;
    class file setattr;
}

===== nova_t =====
allow nova_t gpg_exec_t:file setattr;
allow nova_t hostname_exec_t:file setattr;
allow nova_t rpm_exec_t:file setattr;
~
```

A l'aide de la commande **checkmodule -m -M -o novaapi.mod novaapi.te** on compile le fichier novaapi.te en un module SELinux compilé. Puis on crée un package.pp pour le module SELinux compilé et on installe le module SELinux compilé dans le système.

```
[root@controller khadijaoussakel(keystone)]# checkmodule -m -M -o novaapi.mod novaapi.te
[root@controller khadijaoussakel(keystone)]# semodule_package --outfile novaapi.pp --module novaapi.mod
[root@controller khadijaoussakel(keystone)]# semodule -i novaapi.pp
```

Tout d'abord, on synchronise les bases de données et on configure les cellules du service Nova

```
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash placement -c "placement-manage db sync"
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash nova -c "nova-manage api_db sync"
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash nova -c "nova-manage cell_v2 map cell0"
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash nova -c "nova-manage db sync"
[root@controller khadijaoussakel(keystone)]# su -s /bin/bash nova -c "nova-manage cell_v2 create_cell --name cell1"

[root@controller nova(keystone)]# su -s /bin/bash nova -c "nova-manage cell_v2 create_cell --name cell1"
--transport-url not provided in the command line, using the value [DEFAULT]/transport_url from the configuration file
--database connection not provided in the command line, using the value [database]/connection from the configuration file
```

Ensuite, on redémarre le service httpd, et on vérifie qu'il s'est activé avec succès.

```
[root@controller ~(keystone)]# systemctl restart httpd
[root@controller ~(keystone)]# systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-02-13 10:18:41 EST; 4s ago
     Docs: man:httpd.service(8)
 Main PID: 5388 (httpd)
   Status: "Started, listening on: port 5000, port 443, port 8778, port 80"
    Tasks: 245 (limit: 23520)
   Memory: 85.9M
      CGroup: /system.slice/httpd.service
              ├─5388 /usr/sbin/httpd -DFOREGROUND
              ├─5396 /usr/sbin/httpd -DFOREGROUND
              ├─5397 /usr/sbin/httpd -DFOREGROUND
```

On change le propriétaire d'un fichier journal à "placement", et on active les services OpenStack Nova.

Les services activés sont "api", "conductor", "scheduler", et "novncproxy".

```
[root@controller ~(keystone)]# chown placement. /var/log/placement/placement-api.log

[root@controller ~(keystone)]# for service in api conductor scheduler novncproxy ; do
> systemctl enable --now openstack-nova-$service
> done
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-nova-api.service → /usr/lib/systemd/system/openstack-nova-api.service.
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-nova-conductor.service → /usr/lib/systemd/system/openstack-nova-conductor.service.
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-nova-scheduler.service → /usr/lib/systemd/system/openstack-nova-scheduler.service.
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-nova-novncproxy.service → /usr/lib/systemd/system/openstack-nova-novncproxy.service.
```

On remarque qu'ils sont en cours d'exécution.

```
[root@controller ~]# systemctl status openstack-nova-$service.service
● openstack-nova-novncproxy.service - OpenStack Nova NoVNC Proxy Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-nova-novncproxy.service; enabled; v>
   Active: active (running) since Mon 2023-02-13 10:20:59 EST; 45s ago
     Main PID: 6416 (nova-novncproxy)
        Tasks: 1 (limit: 23520)
       Memory: 114.0M
      CGroup: /system.slice/openstack-nova-novncproxy.service
              └─6416 /usr/bin/python3 /usr/bin/nova-novncproxy --web /usr/share/no>

févr. 13 10:20:59 controller systemd[1]: Started OpenStack Nova NoVNC Proxy Ser>
lines 1-10/10 (END)
```

```
● openstack-nova-scheduler.service - OpenStack Nova Scheduler Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-nova-scheduler.service; enabled; v>
   Active: active (running) since Sat 2023-02-25 09:51:03 EST; 34s ago
     Main PID: 54366 (nova-scheduler)
        Tasks: 4 (limit: 23520)
       Memory: 95.6M
      CGroup: /system.slice/openstack-nova-scheduler.service
              └─54366 /usr/bin/python3 /usr/bin/nova-scheduler

févr. 25 09:50:58 controller systemd[1]: Starting OpenStack Nova Scheduler Server...
févr. 25 09:51:03 controller systemd[1]: Started OpenStack Nova Scheduler Server.

● openstack-nova-conductor.service - OpenStack Nova Conductor Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-nova-conductor.service; enabled; v>
   Active: active (running) since Sat 2023-02-25 09:49:20 EST; 2min 17s ago
     Main PID: 53571 (nova-conductor)
        Tasks: 4 (limit: 23520)
       Memory: 122.2M
      CGroup: /system.slice/openstack-nova-conductor.service
              ├─53571 /usr/bin/python3 /usr/bin/nova-conductor
              ├─55060 /usr/bin/python3 /usr/bin/nova-conductor
```

```
[root@controller khadijaoussakel]# systemctl enable --now libvirtd
[root@controller khadijaoussakel]# systemctl status libvirtd.service
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-02-20 10:08:09 EST; 15s ago
     Docs: man:libvirtd(8)
           https://libvirt.org
   Main PID: 44616 (libvirtd)
      Tasks: 21 (limit: 32768)
     Memory: 55.5M
    CGroup: /system.slice/libvirtd.service
              ├─2356 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --lea>
              ├─2357 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --lea>
              └─44616 /usr/sbin/libvirtd --timeout 120
```

La commande `openstack compute service list` nous montre que les services de calcul OpenStack pour **nova-conductor** et **nova-scheduler** sont actuellement activés et opérationnels.

ID	Binary	Host	Zone	Status	State	Updated At
4	nova-conductor	controller	internal	enabled	up	2023-02-26T13:14:38.000000
5	nova-scheduler	controller	internal	enabled	up	2023-02-26T13:14:34.000000

- Le service **nova-conductor** est responsable de la coordination entre les différents services de calcul Nova, tels que les agents de calcul et les gestionnaires de ressources. Il est également en charge de la planification des tâches et de la communication avec la base de données de Nova.
- Le service **nova-scheduler** est responsable de la planification des instances de calcul pour qu'elles soient exécutées sur les hôtes de calcul appropriés en fonction de leur disponibilité et de leurs capacités. Le planificateur prend également en compte les politiques de placement définies par l'utilisateur.

IX. Configuration de Nova

#3

Nova est le composant central de la plateforme OpenStack qui permet de transformer des instances offertes par “Glance” à des machines virtuelles provisionnées sur les différents nœuds de calcul disponibles dans le cloud. Pour ce faire, Nova s'appuie sur plusieurs services OpenStack tels que Keystone, Glance, Neutron et Placement.

Placement, que nous allons aussi configurer dans cette partie, est le service qui suit l'état des ressources disponibles dans le cloud, notamment les ressources de calcul et de stockage, et qui aide à choisir le fournisseur de ces ressources qui sera utilisé lors de la création d'une machine virtuelle.



On installe les packages nécessaires pour la virtualisation avec KVM, y compris le logiciel QEMU pour émuler des processeurs et des périphériques, la bibliothèque libvirt pour gérer les machines virtuelles, et virt-install pour créer des machines virtuelles.

```
[root@controller ~]# dnf -y install qemu-kvm libvirt virt-install
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:1
5:31 le dim. 26 févr. 2023 13:01:15 +01.
Le paquet qemu-kvm-15:6.0.0-33.el8.x86_64 est déjà installé.
Le paquet libvirt-7.6.0-6.el8.x86_64 est déjà installé.
Le paquet virt-install-2.2.1-4.el8.noarch est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
```

Après, on vérifie que les modules du noyau nécessaires pour la virtualisation avec KVM sont chargés et on active le service libvirtd.

```
[root@controller ~]# lsmod | grep kvm
kvm_intel           323584  4
kvm                 880640  1 kvm_intel
irqbypass          16384   5 kvm
[root@controller ~]# systemctl enable --now libvirtd
[root@controller ~]#
```

On installe le paquet "openstack-nova-compute" à partir des dépôts "Victoria", "EPEL" et "powertools"

```
[root@controller ~]# dnf --enablerepo=centos-openstack-victoria,
epel,powertools -y install openstack-nova-compute
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:1
6:34 le dim. 26 févr. 2023 13:01:15 +01.
Le paquet openstack-nova-compute-1:22.2.2-1.el8.noarch est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
```

En plus des paramètres de base de Nova, on ajoute des paramètres permettant de configurer VNC pour écouter sur toutes les adresses IP ("0.0.0.0"), spécifier l'adresse IP du proxy client ("10.0.0.30"), et définir l'URL de base du serveur proxy NoVNC.

```
[root@controller ~]# vi /etc/nova/nova.conf
```

```
[vnc]
enabled = True
server_listen = 0.0.0.0
server_proxyclient_address = 10.0.0.30
novncproxy_base_url = http://10.0.0.30:6080/vnc_auto.html
```

Puisque SELinux est activé, on change la politique :

```
[root@controller ~]# dnf --enablerepo=centos-openstack-victoria -y install openstack-selinux
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:18:09 le dim. 26 févr. 2023 13:01:15 +01.
Le paquet openstack-selinux-0.8.27-1.el8.noarch est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
```

On démarre le service "openstack-nova-compute" à l'aide de la commande *systemctl* :

```
[root@controller ~]# systemctl enable --now openstack-nova-compute
Created symlink /etc/systemd/system/multi-user.target.wants/openstack-nova-compute.service → /usr/lib/systemd/system/openstack-nova-compute.service.
[root@controller ~]# systemctl status openstack-nova-compute.service
● openstack-nova-compute.service - OpenStack Nova Compute Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-nova-compute.service)
   Active: active (running) since Sun 2023-02-26 14:19:40 +01; 21s ago
     Main PID: 63608 (nova-compute)
        Tasks: 22 (limit: 23520)
       Memory: 158.0M
      CGroup: /system.slice/openstack-nova-compute.service
              └─63608 /usr/bin/python3 /usr/bin/nova-compute

févr. 26 14:19:35 controller systemd[1]: Starting OpenStack Nova Compute >
févr. 26 14:19:40 controller systemd[1]: Started OpenStack Nova Compute S>
```

Puis, on exécute la commande "*nova-manage cell_v2 discover_hosts*" qui permet de détecter les hôtes de calcul disponibles dans la cellule OpenStack et de les ajouter à la base de données de Nova. Et on affiche l'état des services de calcul OpenStack à l'aide de la commande "*openstack compute service list*". Les trois services, "**nova-conductor**", "**nova-scheduler**" et "**nova-compute**", sont activés et en cours d'exécution :

```
[root@controller ~]# su -s /bin/bash nova -c "nova-manage cell_v2 discover_hosts"
[root@controller ~]# openstack compute service list
+-----+-----+-----+-----+-----+
| ID | Binary          | Host           | Zone    | Status | State | Updated At
+-----+-----+-----+-----+-----+
| 4 | nova-conductor | controller | internal | enabled | up    | 2023-02-26T13:20:48
.000000 |
| 5 | nova-scheduler | controller | internal | enabled | up    | 2023-02-26T13:20:46
.000000 |
| 8 | nova-compute   | controller | nova     | enabled | up    | 2023-02-26T13:20:48
.000000 |
+-----+-----+-----+-----+-----+-----+
```

X. Configuration de neutron

#1

Neutron est un projet du système d'exploitation OpenStack qui fournit des services de réseau en tant que service dans un environnement cloud. Il permet de gérer et de configurer des réseaux virtuels pour les machines virtuelles (VM) et les conteneurs.

Plus spécifiquement, Neutron fournit une API unifiée pour la gestion des réseaux virtuels, qui peut être utilisée pour créer et configurer des réseaux, des sous-réseaux, des ports et des groupes de sécurité pour les VM et les conteneurs. Il prend également en charge des fonctionnalités avancées telles que la qualité de service (QoS), les réseaux privés virtuels (VPN) et les pare-feux virtuels.



Ce qu'on va faire maintenant est de créer un utilisateur dans Keystone pour permettre à Neutron de s'authentifier et d'accéder aux ressources de l'infrastructure cloud OpenStack, et cela en utilisant la commande suivante **openstack user create --domain default --project service --password servicepassword neutron**. Ensuite on va ajouter le rôle "admin" à l'utilisateur "neutron" avec la commande **openstack role add --project service --user neutron admin** :

```
[root@controller ~]# openstack user create --domain default --project service --password servicepassword neutron
+-----+-----+
| Field      | Value   |
+-----+-----+
| default_project_id | 35b85efdfebc45b7bdbc6c52177d9597 |
| domain_id    | default |
| enabled       | True    |
| id            | 9a75dfe10ff0460f82065b0da0f6b197 |
| name          | neutron |
| options        | {}      |
| password_expires_at | None   |
+-----+-----+
[root@controller ~]# openstack role add --project service --user neutron admin
```

Après on va créer un service nommé neutron par la commande suivante :

```
[root@controller ~]# openstack service create --name neutron --description "OpenStack Networking service" network
+-----+-----+
| Field      | Value   |
+-----+-----+
| description | OpenStack Networking service |
| enabled     | True    |
| id          | aeff1bf88504461cab08730492c0ca93 |
| name        | neutron |
| type        | network |
+-----+-----+
```

On va définir par la suite une variable d'environnement appelée "controller" avec la valeur "10.0.0.30" afin de l'utiliser au cours du ce travail :

```
[root@controller ~]# export controller=10.0.0.30
```

- **Un endpoint est une URL qui identifie un service spécifique dans OpenStack.**
C'est un élément important de l'architecture d'OpenStack car il permet aux utilisateurs et aux services d'accéder aux fonctionnalités et aux ressources d'un service spécifique.
- **Un endpoint est associé à un service OpenStack particulier, tel que le service de réseau, le service d'identité ou le service de stockage objet. Il peut être de plusieurs types, notamment public, interne et admin, qui sont utilisés pour accéder à un service à partir de différents endroits ou avec différents niveaux de privilèges.**

Alors ce qu'on va faire maintenant est de créer trois endpoints avec la même URL mais avec des priviléges différents pour le service « neutron ».

```
[root@controller ~]# openstack endpoint create --region RegionOne network public http://$controller:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 69581c385f4f43ee8f6e88174c0efd23 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | aeef1bf88504461cab08730492c0ca93 |
| service_name | neutron |
| service_type | network |
| url | http://10.0.0.30:9696 |
```



```
[root@controller ~]# openstack endpoint create --region RegionOne network internal http://$controller:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 5c7276e52d4f494080c8cf67df36d764 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | aeef1bf88504461cab08730492c0ca93 |
| service_name | neutron |
| service_type | network |
| url | http://10.0.0.30:9696 |
```



```
[root@controller ~]# openstack endpoint create --region RegionOne network admin http://$controller:9696
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | e07bbf2492bc450da27c086a20373508 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| service_id | aeef1bf88504461cab08730492c0ca93 |
| service_name | neutron |
| service_type | network |
| url | http://10.0.0.30:9696 |
```

On va créer une base de données pour neutron sous le nom de « neutron_ml2 », ensuite on va créer un utilisateur neutron avec le mot de passe password et on va lui accorder tous les privilèges sur la base de données neutron_ml2.

```
[root@controller ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 84
Server version: 10.3.28-MariaDB MariaDB Server

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_ml2;
Query OK, 1 row affected (0.005 sec)

MariaDB [(none)]> grant all privileges on neutron_ml2.* to neutron@'localhost'
identified by 'password';
Query OK, 0 rows affected (0.013 sec)

MariaDB [(none)]> grant all privileges on neutron_ml2.* to neutron@'%' identifi
ed by 'password';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> flush privileges;
```

XI. Configuration de neutron #2

Neutron est un projet du système d'exploitation OpenStack qui fournit des services de réseau en tant que service dans un environnement cloud. Il permet de gérer et de configurer des réseaux virtuels pour les machines virtuelles (VM) et les conteneurs.

Plus spécifiquement, Neutron fournit une API unifiée pour la gestion des réseaux virtuels, qui peut être utilisée pour créer et configurer des réseaux, des sous-réseaux, des ports et des groupes de sécurité pour les VM et les conteneurs. Il prend également en charge des fonctionnalités avancées telles que la qualité de service (QoS), les réseaux privés virtuels (VPN) et les pare-feux virtuels.



Maintenant on va installer les paquets openstack-neutron, openstack-neutron-ml2 et openstack-neutron-openvswitch.

```
[root@controller ~]# dnf --enablerepo=centos-openstack-victoria,power tools,epel -y install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:27:19
le dim. 26 févr. 2023 13:01:15 +01.
Dépendances résolues.
=====
 Paquet          Architecture      Version       Dépôt        Taille
=====
Installation:
 openstack-neutron    noarch 1:17.2.1-1.el8 centos-openstack-victoria 26 k
 openstack-neutron-ml2   noarch 1:17.2.1-1.el8 centos-openstack-victoria 16 k
 openstack-neutron-openvswitch
                           noarch 1:17.2.1-1.el8 centos-openstack-victoria 18 k
Mise à jour:
```

Une fois fait, on va configurer les services de neutron en accédant au fichier de configuration **/etc/neutron/neutron.conf** sans oublier de créer avant une copie de ce dernier afin de ne pas perdre la configuration précédente par la commande suivante **mv /etc/neutron/neutron.conf /etc/neutron/neutron.conf.org**

```
[root@controller ~]# mv /etc/neutron/neutron.conf /etc/neutron/neutron.conf.org
[root@controller ~]# vi /etc/neutron/neutron.conf
```

```
Fichier Édition Affichage Rechercher Terminal Aide
[DEFAULT]
core_plugin = ml2
service_plugins = router
auth_strategy = keystone
state_path = /var/lib/neutron
dhcp_agent_notification = True
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
# RabbitMQ connection info
transport_url = rabbit://openstack:password@10.0.0.30

# Keystone auth info
[keystone_authtoken]
www_authenticate_uri = http://10.0.0.30:5000
auth_url = http://10.0.0.30:5000
memcached_servers = 10.0.0.30:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = servicepassword
```

Après on va définir les permissions d'accès au appropriées sur le fichier de configuration par les deux commandes suivantes :

```
[root@controller ~]# chmod 640 /etc/neutron/neutron.conf  
[root@controller ~]# chgrp neutron /etc/neutron/neutron.conf  
[root@controller ~]# vi /etc/neutron/l3_agent.ini
```

Puis on configure le fichier /etc/neutron/l3_agent.ini en spécifiant le pilote d'interface réseau utilisé :

```
Fichier Édition Affichage Rechercher Terminal Aide  
[DEFAULT]  
interface_driver = openvswitch  
#  
# From oslo.log  
#
```

Ensuite on configure le fichier /etc/neutron/dhcp_agent.ini en spécifiant les pilotes d'interface réseau et de serveur DHCP, et en activant la communication entre les machines virtuelles et le service de métadonnées OpenStack.

```
[root@controller ~]# vi /etc/neutron/dhcp_agent.ini
```

```
Fichier Édition Affichage Rechercher Terminal Aide  
[DEFAULT]  
interface_driver = openvswitch  
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq  
enable_isolated_metadata = true  
#
```

Dans ce qui suit on configure le fichier /etc/neutron/metadata_agent.ini pour configurer le service de métadonnées OpenStack, qui fournit des informations supplémentaires sur les instances de machines virtuelles.

```
[root@controller ~]# vi /etc/neutron/metadata_agent.ini
```

```
[DEFAULT]  
# specify Nova API server  
  
nova_metadata_host = 10.0.0.30  
# specify any secret key you like  
  
metadata_proxy_shared_secret = metadata_secret  
#  
  
#memcache_servers = localhost:11211  
memcache_servers = 10.0.0.30:11211
```

Ensuite on modifie le fichier /etc/nova/nova.conf pour configurer le module de groupe de sécurité Neutron

Pour le default block on va ajouter les lignes suivantes :

```
[root@controller ~]# vi /etc/nova/nova.conf

[DEFAULT]
# define own IP address
my_ip = 10.0.0.30
state_path = /var/lib/nova
enabled_apis = osapi_compute,metadata
log_dir = /var/log/nova
# RabbitMQ connection info
transport_url = rabbit://openstack:password@10.0.0.30
use_neutron = True
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
vif_plugging_is_fatal = True
vif_plugging_timeout = 300

[api]
```

Et pour le bloc neutron on va ajouter les lignes suivantes :

```
[neutron]
auth_url = http://10.0.0.30:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = servicepassword
service_metadata_proxy = True
metadata_proxy_shared_secret = metadata_secret
-- INSERT --
```

On va installer le paquet openstack-selinux qui fournit les règles de sécurité SELinux pour OpenStack

```
[root@controller ~]# dnf --enablerepo=centos-openstack-victoria -y install openstack-selinux
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:42:40
le dim. 26 févr. 2023 13:01:15 +01.
Le paquet openstack-selinux-0.8.27-1.el8.noarch est déjà installé.
Dépendances résolues.
Rien à faire.
Terminé !
```

Ensuite on va définir des paramètres SELinux pour les différents services du système (permettre à Neutron de configurer les réseaux, permet à HAProxy de se connecter à n'importe quelle adresse IP, et autoriser les démons à fonctionner en mode cluster)

```
[root@controller ~]# setsebool -P neutron_can_network on
[root@controller ~]# setsebool -P haproxy_connect_any on
[root@controller ~]# setsebool -P daemons_enable_cluster_mode on
[root@controller ~]# vi ovsofctl.te
```

Puis on va créer le module ovsofctl. Ce module définit les autorisations SELinux pour deux types de processus : neutron_t et dnsmasq_t. Il permet au processus neutron_t d'avoir les capacités dac_override et sys_rawio, et d'exécuter des fichiers sans être affecté par la transition de contexte SELinux. De même, il permet au processus dnsmasq_t d'avoir la capacité dac_override.

Pour ce faire on crée le fichier ovsofctl.te et on le configure comme suit :

```
Fichier Édition Affichage Rechercher Terminal Aide
module ovsofctl 1.0;

require {
    type neutron_t;
    type neutron_exec_t;
    type neutron_t;
    type dnsmasq_t;
    class file execute_no_trans;
    class capability { dac_override sys_rawio };
}

===== neutron_t =====
allow neutron_t self:capability { dac_override sys_rawio };
allow neutron_t neutron_exec_t:file execute_no_trans;

===== dnsmasq_t =====
allow dnsmasq_t self:capability dac_override;
~
~
```

Après pour compiler la politique de sécurité SELinux décrite dans le fichier qu'on vient de créer, la compilation va donner naissance au module qui sera stocké dans le fichier ovsofctl.mod

```
[root@controller ~]# checkmodule -m -M -o ovsofctl.mod ovsofctl.te
```

Puis on va créer un package SELinux à partir du module de sécurité ovsofctl.mod déjà généré

Le fichier de sortie ovsofctl.pp contiendra le module et les informations nécessaires pour l'ajouter au système SELinux.

```
[root@controller ~]# semodule_package --outfile ovsofctl.pp --module ovsofctl.mod
```

Enfin on installe le module SELinux qui a été généré précédemment.

```
[root@controller ~]# semodule -i ovsofctl.pp
```

D'abord il va falloir activer OpenvSwitch .

OpenvSwitch est un commutateur réseau virtuel utilisé dans les déploiements OpenStack pour gérer les réseaux virtuels, on va l'activer comme suit :

```
[root@controller ~]# systemctl enable --now openvswitch
Created symlink /etc/systemd/system/multi-user.target.wants/openvswitch.service
→ /usr/lib/systemd/system/openvswitch.service.
[root@controller ~]# systemctl status openvswitch.service
● openvswitch.service - Open vSwitch
   Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; enabled; vendor>
   Active: active (exited) since Sun 2023-02-26 14:46:40 +01; 14s ago
     Process: 67417 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 67417 (code=exited, status=0/SUCCESS)

févr. 26 14:46:40 controller systemd[1]: Starting Open vSwitch...
févr. 26 14:46:40 controller systemd[1]: Started Open vSwitch.
lines 1-8/8 (END)
```

Ensuite on va créer un nouveau bridge OVS nommé "br-int".

"br-int" est un bridge intégré de façon transparente par Neutron pour connecter les instances entre elles.

```
[root@controller ~]# ovs-vsctl add-br br-int
```

Maintenant on va créer un lien symbolique entre le fichier de configuration /etc/neutron/plugins/ml2/ml2_conf.ini et le fichier /etc/neutron/plugin.ini. Cela va nous permettre d'utiliser le fichier ml2_conf.ini comme configuration pour le plugin ML2 de Neutron

```
[root@controller ~]# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

Ensuite on va mettre à jour la base de données Neutron en exécutant les scripts SQL nécessaires pour la version actuelle de Neutron.

```
[root@controller ~]# su -s /bin/bash neutron -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugin.ini upgrade head"
INFO [alembic.runtime.migration] Context impl MySQLImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
Exécution de upgrade pour neutron...
```

Maintenant on va démarrer et activer les services Neutron qu'on a configuré précédemment, tels que le serveur, l'agent DHCP, l'agent L3, l'agent de métadonnées et l'agent Open vSwitch.

```
[root@controller ~]# for service in server dhcp-agent l3-agent metadata-agent openvswitch-agent; do
> systemctl enable --now neutron-$service
> done
Created symlink /etc/systemd/system/multi-user.target.wants/neutron-server.service → /usr/lib/systemd/system/neutron-server.service.
Created symlink /etc/systemd/system/multi-user.target.wants/neutron-dhcp-agent.service → /usr/lib/systemd/system/neutron-dhcp-agent.service.
Created symlink /etc/systemd/system/multi-user.target.wants/neutron-l3-agent.service → /usr/lib/systemd/system/neutron-l3-agent.service.
Created symlink /etc/systemd/system/multi-user.target.wants/neutron-metadata-agent.service → /usr/lib/systemd/system/neutron-metadata-agent.service.
```

Enfin, on redémarre les services OpenStack Nova API et Nova Compute, et on affiche la liste des agents du réseau et leurs états.

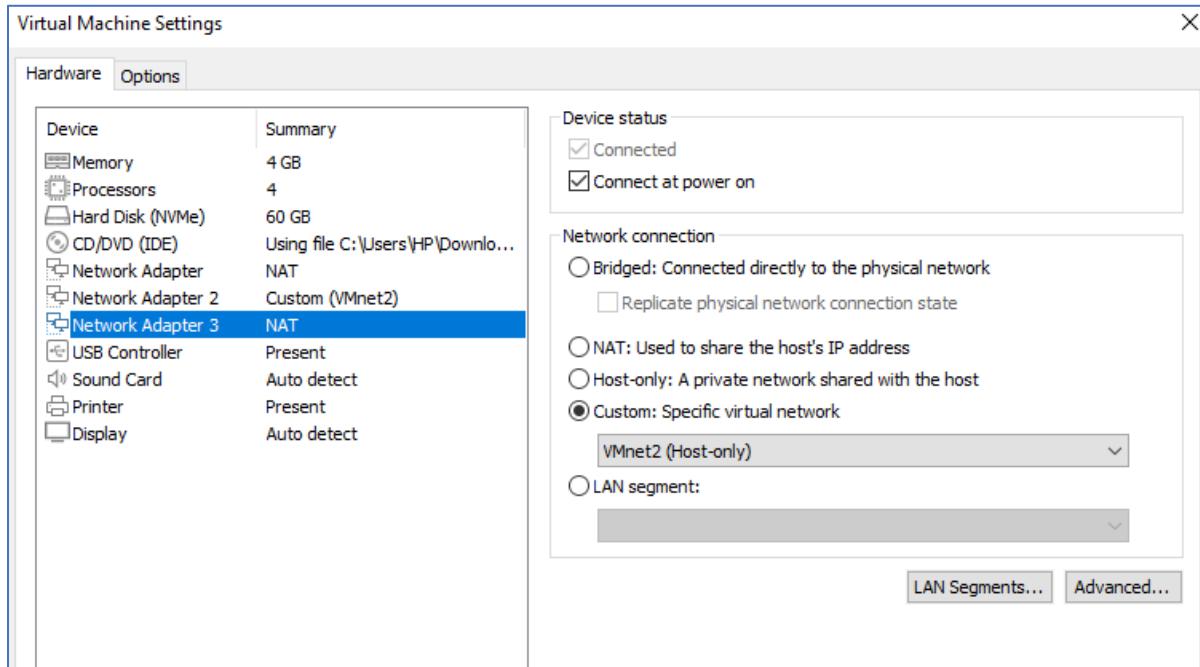
```
[root@controller ~]# systemctl restart openstack-nova-api openstack-nova-compute
[root@controller ~]# openstack network agent list
+-----+-----+-----+-----+-----+
| ID           | Agent Type   | Host     | Availability Zone | Alive | State | Binary
+-----+-----+-----+-----+-----+
| 2c5023b6-349a-4e41-a202-6a7948ef6168 | Open vSwitch agent | controller | None          | : -) | UP    | neutron-openvswitch
| 36c766a4-a6a9-48aa-973b-7b72dee7ac03 | L3 agent       | controller | nova          | : -) | UP    | neutron-l3-agent
| 4df3ec54-da19-4ac8-bf17-a8339a1edca8 | Metadata agent | controller | None          | : -) | UP    | neutron-metadata
| bee497e5-7887-49ef-ba69-0f5f8eafdd28 | DHCP agent     | controller | nova          | : -) | UP    | neutron-dhcp-agent
+-----+-----+-----+-----+-----+
```

XII. Configuration de la mise en réseau de neutron

Dans cette section, on va voir comment configurer la mise en réseau pour OpenStack Victoria en utilisant Neutron avec un commutateur externe (OVS). Dans ce qui suit il y a les étapes pour installer les paquets nécessaires, configurer les interfaces réseau et les tables de flux OVS, et configurer les fichiers de configuration Neutron.



Tout d'abord, on ajoute une interface réseau ens256 sans address IP :



```
9: ens256: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default  
qdisc pfifo_fast root  
link/ether 00:0c:29:48:7b:d8 brd ff:ff:ff:ff:ff:ff  
[root@controller ~]#
```

On crée alors un nouveau pont Open vSwitch appelé "br-eth1", ensuite on va ajouter le port Ethernet physique nommé "ens256" au pont Open vSwitch appelé "br-eth1".

```
[root@controller ~]# ovs-vsctl add-br br-eth1  
[root@controller ~]# ovs-vsctl add-port br-eth1 ens256
```

Ensuite on va modifier le fichier /etc/neutron/plugins/ml2/ml2_conf.ini afin de configurer le plugin de réseau Neutron ML2 pour prendre en charge les réseaux plats.

```
[root@controller ~]# vi /etc/neutron/plugins/ml2/ml2_conf.ini
```

```
[ml2_type_flat]  
flat_networks = physnet1  
-- INSERT --
```

De même pour le fichier /etc/neutron/plugins/ml2/openvswitch_agent.ini ,on le configure pour spécifier une correspondance entre l'interface réseau physique **physnet1** et le pont **br-eth1**.

```
[root@controller ~]# vi /etc/neutron/plugins/ml2/openvswitch_agent.ini
```

```
[ovs]  
bridge_mappings = physnet1:br-eth1  
-- INSERT --
```

```
[root@controller ~]# systemctl restart neutron-openvswitch-agent
```

Après avoir défini le projectId, on va créer un réseau nommé "sharednet1" dans le projet spécifié par ce dernier.

```
[root@controller ~]# projectID=$(openstack project list | grep service | awk '{print $2}')
[root@controller ~]# openstack network create --project $projectID \
> --share --provider-network-type flat --provider-physical-network physnet1 sharednet1

+-----+-----+
| Field          | Value        |
+-----+-----+
| admin_state_up | UP           |
| availability_zone_hints | None         |
| availability_zones | None         |
| created_at     | 2023-02-26T15:47:18Z |
| description    | None         |
| dns_domain     | None         |
| id             | f681a6e5-33c8-497d-8476-c9cd10b816e1 |
| ipv4_address_scope | None         |
| ipv6_address_scope | None         |
| is_default     | False        |
| is_vlan_transparent | None         |
| mtu            | 1500         |
| name           | sharednet1  |
| port_security_enabled | True         |
| project_id     | 35b85efdfebc45b7bdbcb6c52177d9597 |
| provider:network_type | flat         |
| provider:physical_network | physnet1   |
| provider:segmentation_id | None         |
| qos_policy_id  | None         |
+-----+-----+
```

Puis on va créer un sous-réseau nommé "subnet1" dans le réseau "sharednet1" avec une plage d'adresses IP de 10.0.0.0/24

```
[root@controller ~]# openstack subnet create subnet1 --network sharednet1 \
> --project $projectID --subnet-range 10.0.0.0/24 \
> --allocation-pool start=10.0.0.200,end=10.0.0.254 \
> --gateway 10.0.0.1 --dns-nameserver 10.0.0.10
+-----+-----+
| Field          | Value        |
+-----+-----+
| allocation_pools | 10.0.0.200-10.0.0.254 |
| cidr           | 10.0.0.0/24      |
| created_at     | 2023-02-26T15:48:03Z |
| description    | None           |
| dns_nameservers | 10.0.0.10       |
| dns_publish_fixed_ip | None           |
| enable_dhcp    | True            |
| gateway_ip     | 10.0.0.1        |
| host_routes    | None           |
| id             | 42575646-87b0-4a0b-8453-4ea18d30fd70 |
| ip_version     | 4               |
| ipv6_address_mode | None           |
| ipv6_ra_mode   | None           |
| name           | subnet1        |
| network_id     | f681a6e5-33c8-497d-8476-c9cd10b816e1 |
| prefix_length  | None           |
| project_id     | 35b85efdfebc45b7bdbcb6c52177d9597 |
| revision_number | 0              |
| segment_id     | None           |
| service_types  | None           |
| subnetpool_id  | None           |
+-----+-----+
```

Enfin on va afficher la liste des réseaux et des sous-réseaux disponibles

```
[root@controller ~]# openstack network list
+-----+-----+-----+
| ID          | Name      | Subnets           |
+-----+-----+-----+
| f681a6e5-33c8-497d-8476-c9cd10b816e1 | sharednet1 | 42575646-87b0-4a0b-8453-4ea18d30fd70 |
+-----+-----+-----+
```

```
[root@controller ~]# openstack subnet list
+-----+-----+-----+-----+
| ID          | Name      | Network           | Subnet           |
+-----+-----+-----+-----+
| 42575646-87b0-4a0b-8453-4ea18d30fd70 | subnet1   | f681a6e5-33c8-497d-8476-c9cd10b816e1 | 10.0.0.0/24    |
+-----+-----+-----+-----+
```

XIII. Ajout des utilisateurs Openstack

Cette partie décrit comment ajouter des comptes d'utilisateurs dans Keystone, le service d'identification d'OpenStack. En effet, on va découvrir comment créer des projets et des utilisateurs, attribuer des rôles aux utilisateurs, et configurer les fichiers de configuration Keystone pour permettre aux utilisateurs de se connecter à OpenStack.



Premièrement, on va créer un nouveau projet nommé "hiroshima" avec une description spécifiée comme "Hiroshima Project"

```
[root@controller ~]# openstack project create --domain default --description "Hiroshima Project" hiroshima
+-----+
| Field      | Value
+-----+
| description | Hiroshima Project
| domain_id   | default
| enabled     | True
| id          | 48d740b7fd0c4bd1888f6381fc448be4
| is_domain   | False
| name        | hiroshima
| options     | {}
| parent_id   | default
| tags        | []
+-----+
```

Le projet a été bien ajouté :

```
[root@controller ~]# openstack project list
+-----+-----+
| ID            | Name
+-----+-----+
| 35b85efdfefc45b7bdb6c52177d9597 | service
| 48d740b7fd0c4bd1888f6381fc448be4 | hiroshima
| 9bb7fb3fc20487db08b11724d3a3fb2 | admin
+-----+-----+
[root@controller ~]#
```

Une fois fait on va créer un nouvel utilisateur nommé "serverworld" avec le mot de passe "userpassword"

```
[root@controller ~]# openstack user create --domain default --project hiroshima --password userpassword serverworld
+-----+
| Field      | Value
+-----+
| default_project_id | 48d740b7fd0c4bd1888f6381fc448be4
| domain_id   | default
| enabled     | True
| id          | 38b89e731a68447fbc7468368afe52c9
| name        | serverworld
| options     | {}
| password_expires_at | None
+-----+
```

L'utilisateur a été bien ajouté :

```
[root@controller ~]# openstack user list
+-----+-----+
| ID            | Name
+-----+-----+
| b3981c21a830415883090d432b1caa6d | admin
| 27fd0dc277bc4934abdf977397f79f7e | glance
| 399c83ec676b48d883137670e5abbbff | nova
| 39e2c1286d884d368026b50c733b0ae1 | placement
| 9a75dfe10ff0460f82065b0da0f6b197 | neutron
| 38b89e731a68447fbc7468368afe52c9 | serverworld
+-----+-----+
```

Ensuite on va créer un nouveau rôle "CloudUser" dans Keystone pour l'attribuer par la suite à des utilisateurs ou des projets pour leur donner des privilèges spécifiques.

```
[root@controller ~ (keystone)]# openstack role create CloudUser
+-----+-----+
| Field | Value |
+-----+-----+
| description | None |
| domain_id | None |
| id | 5b31025e63d3467b8d06f3974f55a60d |
| name | CloudUser |
| options | {} |
+-----+-----+
```

Le rôle « CloudUser » a été bien créé :

```
[root@controller ~ (keystone)]# openstack role list
+-----+-----+
| ID | Name |
+-----+-----+
| 0be9104faeb742148a82a7215de3bbdd | member |
| 5b31025e63d3467b8d06f3974f55a60d | CloudUser |
| 5c07f7bcab3545a89898ca656d7857a8 | reader |
| 6b9834ca26bd4b58bf9c8ffb6741ab8 | admin |
+-----+-----+
```

Puis on va ajouter le rôle "CloudUser" à l'utilisateur "serverworld" pour le projet "hiroshima".

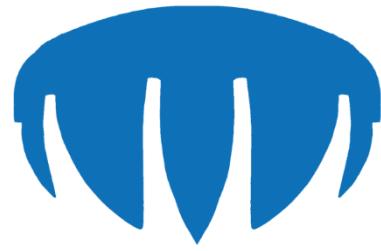
```
[root@controller ~ (keystone)]# openstack role add --project hiroshima --user serverworld CloudUser
```

Enfin on va créer une nouvelle instance nommée "m1.small" dont la taille est de 12 go.

```
[root@controller ~ (keystone)]# openstack flavor create --id 0 --vcpus 1 --ram 3048 -disk 12 m1.small
+-----+-----+
| Field | Value |
+-----+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 0 |
| disk | 12 |
| id | 0 |
| name | m1.small |
| os-flavor-access:is_public | True |
| properties |
| ram | 3048 |
| rxtx_factor | 1.0 |
| swap |
| vcpus | 1 |
+-----+-----+
```

XIV. Cration et execution des instances

Cette partie est consacre  la cration et au demarrage dune instance de machine virtuelle sur OpenStack Victoria. On va voir en details les tapes pour crer une image dinstance, crer une instance  partir de cette image, et effectuer des taches de gestion courantes, telles que la cration de volumes et de groupes de surite pour linstance. Cette section dcrit galement comment se connecter  linstance via une console Web et comment acceder  linstance via une connexion SSH.



Au niveau de cette étape, on va essayer de créer et exécuter l'instance centos-8 .

La première chose à faire est d'ouvrir le fichier /keystonerc afin de configurer les variables d'environnement OpenStack.

```
[khadijaoussakel@controller ~]$ vi ~/keystonerc
```

```
Fichier Édition Affichage Rechercher Terminal Aide
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=hiroshima
export OS_USERNAME=serverworld
export OS_PASSWORD=userpassword
export OS_AUTH_URL=http://10.0.0.30:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export PS1='[\u@\h \w(keystone)]$ '
```

On définit ensuite les permissions de fichier du fichier "keystonerc" à 600, puis on va charger les informations d'identification d'OpenStack à partir du fichier keystonerc.

```
[khadijaoussakel@controller ~]$ chmod 600 ~/keystonerc
[khadijaoussakel@controller ~]$ source ~/keystonerc
```

Ensuite on va ajouter la commande "source ~/keystonerc" au fichier de configuration de démarrage ~/.bash_profile. Cela signifie que chaque fois qu'on ouvre un nouveau terminal, les informations d'identification d'OpenStack seront automatiquement chargées

```
[khadijaoussakel@controller ~](keystone)]$ su root
Mot de passe :
[root@controller khadijaoussakel(keystone)]# cd
[root@controller ~](keystone)]# echo "source ~/keystonerc" >> ~/.bash_profile
```

Afin de s'assurer que tout est prêt pour créer l'instance centos-8, il va falloir afficher le nécessaire ainsi de s'assurer que tout fonctionne

En effet on va afficher la liste des flavors, la liste des images, et la liste et des réseaux disponibles.

```
[root@controller ~](keystone)]# openstack flavor list
+---+-----+-----+-----+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+---+-----+-----+-----+-----+-----+
| 0 | m1.small | 3048 | 12 | 0 | 1 | True |
+---+-----+-----+-----+-----+-----+
[root@controller ~](keystone)]# openstack image list
+-----+-----+-----+
| ID | Name | Status |
+-----+-----+-----+
| 83865c6c-316b-422e-bb53-344e58940bc5 | CentOS8 | active |
+-----+-----+-----+
[root@controller ~](keystone)]# openstack network list
+-----+-----+-----+
| ID | Name | Subnets |
+-----+-----+-----+
| f681a6e5-33c8-497d-8476-c9cd10b816e1 | sharednet1 | 42575646-87b0-4a0b-8453-4ea18d30fd70 |
+-----+-----+-----+
```

Ensuite on va créer le securitygroup qu'on va nommer secgroup01 et auquel on va attribuer notre instance :

```
[root@controller ~ (keystone)]# openstack security group create secgroup01
+-----+
| Field      | Value
+-----+
| created_at | 2023-02-26T17:04:11Z
| description | secgroup01
| id          | 76ea2b98-7e25-4507-b4e9-d7d461747d78
| name        | secgroup01
| project_id  | 48d740b7fd0c4bd1888f6381fc448be4
```

Ici on affiche les security group existants :

```
[root@controller ~ (keystone)]# openstack security group list
+-----+-----+-----+-----+
| ID           | Name     | Description | Project
| Tags         |          |             |          |
+-----+-----+-----+-----+
| 76ea2b98-7e25-4507-b4e9-d7d461747d78 | secgroup01 | secgroup01 | 48d740b7fd0c4bd
1888f6381fc448be4 | []       |             |          |
| c1f3c9cc-7b9a-4ebd-9a16-5e0a9ae976e7 | default   | Groupe de sécurité par défaut | 48d740b7fd0c4bd
1888f6381fc448be4 | []       |             |          |
+-----+-----+-----+-----+
```

Ensuite on va générer une paire de clés SSH en utilisant le type de clé RSA par défaut et créer une paire de clés OpenStack nommée "mykey" à partir de la clé publique générée.

```
[root@controller ~ (keystone)]# ssh-keygen -q -N ""
Enter file in which to save the key (/root/.ssh/id_rsa):
```

Et comme toujours pour afficher les clées disponibles on tape la commande openstack keypair list.

```
[root@controller ~ (keystone)]# openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
+-----+
| Field      | Value
+-----+
| fingerprint | 2d:47:fb:3e:f0:a8:a9:f0:65:1d:48:46:9f:42:0b:d8
| name       | mykey
| user_id    | 38b89e731a68447fbc7468368afe52c9
+-----+
[root@controller ~ (keystone)]# openstack keypair list
+-----+
| Name | Fingerprint
+-----+
| mykey | 2d:47:fb:3e:f0:a8:a9:f0:65:1d:48:46:9f:42:0b:d8
+-----+
```

Enfin et après avoir définir l'ID du réseau à utiliser on va arriver à l'étape de création de l'instance :

Field	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	NOSTATE
OS-EXT-STS:power_state	scheduling
OS-EXT-STS:task_state	building
OS-EXT-STS:vm_state	None
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	
accessIPv4	
accessIPv6	
addresses	
adminPass	KTGz4K5U7Lxa
config_drive	
created	2023-02-27T15:25:11Z
flavor	m1.small (0)
hostId	
id	1a4fea0d-4882-4f69-8644-3c5596a46eee
image	CentOS8 (83865c6c-316b-422e-bb53-344e58940bc5)

On utilise la commande "openstack server list" qui permet d'afficher la liste des instances de serveur créées dans OpenStack. Dans notre cas, l'instance CentOS-8 est en cours de construction :

ID	Name	Status	Networks	Image	Flavor
1a4fea0d-4882-4f69-8644-3c5596a46eee	CentOS-8	BUILD		CentOS8 (83865c6c-316b-422e-bb53-344e58940bc5)	m1.small

Malheureusement on a rencontré un problème lors de la création de l'instance :

ID	Name	Status	Networks	Image	Flavor
1a4fea0d-4882-4f69-8644-3c5596a46eee	CentOS-8	ERROR		CentOS8 (83865c6c-316b-422e-bb53-344e58940bc5)	m1.small

Maintenant on crée des règles de sécurité dans le groupe de sécurité nommé "secgroup01" pour permettre le trafic ICMP entrant (par exemple, les requêtes de ping). Cette commande autorise donc les instances qui appartiennent à ce groupe de sécurité à recevoir des requêtes ICMP entrantes :

[root@controller ~](keystone)]# openstack security group rule create --protocol icmp --ingress se
cgroup01
+-----+-----+
Field Value
+-----+-----+
created_at 2023-02-27T14:26:05Z
description ingress
direction IPv4
ether_type IPv4
id 528b7b3c-7f24-490b-9e50-f1f2a208b397
name None
port_range_max None
port_range_min None
project_id 9bb7fbb3fc20487db08b11724d3a3fb2
protocol icmp
remote_group_id None
remote_ip_prefix 0.0.0.0/0
revision_number 0
security_group_id 1590e6f7-4d49-459a-b98c-96124609014c
tags []
updated_at 2023-02-27T14:26:05Z

Ensuite on crée une règle pour permettre le trafic TCP entrant sur le port 22, qui est utilisé pour les connexions SSH. Cette commande autorise donc les instances qui appartiennent à ce groupe de sécurité à recevoir des connexions SSH entrantes :

[root@controller ~](keystone)]# openstack security group rule create --protocol tcp --dst-port 22
:22 secgroup01
+-----+-----+
Field Value
+-----+-----+
created_at 2023-02-27T14:26:55Z
description ingress
direction IPv4
ether_type IPv4
id 1db5b660-6ba0-4a68-b9a4-41f6982a281b
name None
port_range_max 22
port_range_min 22
project_id 9bb7fbb3fc20487db08b11724d3a3fb2
protocol tcp
remote_group_id None
remote_ip_prefix 0.0.0.0/0
revision_number 0
security_group_id 1590e6f7-4d49-459a-b98c-96124609014c
tags []
updated_at 2023-02-27T14:26:55Z

Enfin on liste toutes les règles de sécurité créées dans le groupe de sécurité :

ID	IP Protocol	Ethertype	IP Range	Port Range	Remote Security Group	Security Group
1db5b660-6ba0-4a68-b9a4-41f6982a281b	tcp	IPv4	0.0.0.0/0	22:22		None
			1590e6f7-4d49-459a-b98c-96124609014c			
25da63e2-9e45-4439-a61b-6c4fc23fb0c4	None	IPv4	0.0.0.0/0			None
			1590e6f7-4d49-459a-b98c-96124609014c			
3747fac1-acaa-41e7-a92f-0c6fd8199314	None	IPv4	0.0.0.0/0			None
			e498a630-b77e-4d1e-b989-77ecd0f9d784			
528b7b3c-7f24-490b-9e50-f1f2a208b397	icmp	IPv4	0.0.0.0/0			None
			1590e6f7-4d49-459a-b98c-96124609014c			
52b3536b-eb9b-48a2-af46-78fb34ac1fa8	None	IPv4	0.0.0.0/0			e498a630-b77e
-4d1e-b989-77ecd0f9d784			e498a630-b77e-4d1e-b989-77ecd0f9d784			
6eb61515-a669-4745-895a-8a04791d481e	None	IPv6	::/0			a562525c-de38
-4c07-b1ee-f025a123f8ae			a562525c-de38-4c07-b1ee-f025a123f8ae			
ac51841a-87fe-4a6f-b0da-c58ee2ec1576	None	IPv6	::/0			None
			e498a630-b77e-4d1e-b989-77ecd0f9d784			
b5c6d7d1-5b16-4349-8490-abe5f4a3a094	None	IPv4	0.0.0.0/0			a562525c-de38

XV. Configuration d'Horizon

Dans cette étape on va installer et exploiter le tableau de bord fournit par le service Horizon d'Openstack afin de gérer nos instances et nos clouds.

Horizon est l'interface web officielle d'OpenStack, permettant aux utilisateurs de gérer leurs ressources OpenStack via une interface graphique. Il permet de créer et de gérer des instances, des images, des réseaux, des volumes, des groupes de sécurité, des projets et bien plus encore.



On commence par installer le service Horizon à partir des dépôts "Victoria", "EPEL" et "powertools"

```
[root@controller ~]# dnf --enablerepo=centos-openstack-victoria,powertools,epel -y install openstack-dashboard
Dernière vérification de l'expiration des métadonnées effectuée il y a 1 day, 3:28:27 le dim. 26 févr. 2023 13:01:15 +01.
Dépendances résolues.
=====
Paquet          Architecture      Version       Dépôt        Taille
=====
Installation:
openstack-dashboard    noarch 1:18.6.2-1.el8   centos-openstack-victoria 12 M
Installation des dépendances:
xstatic-Angular-common  noarch 1:1.5.8.0-10.el8  centos-openstack-victoria 460 k
xstatic-Magic-Search-common  noarch 0.2.5.1-12.el8  centos-openstack-victoria 14 k
bootswatch-common      noarch 3.3.7.0-11.el8   centos-openstack-victoria 623 k
bootswatch-fonts        noarch 3.3.7.0-11.el8   centos-openstack-victoria 118 k
fontawesome-fonts       noarch 4.7.0-4.el8     appstream        203 k
```

On configure le fichier **local_settings** tel que :

```
[root@controller ~]# gedit /etc/openstack-dashboard/local_settings
```

- L'option **ALLOWED_HOSTS** permet de spécifier les hôtes qui sont autorisés à accéder au tableau de bord Horizon. Nous voulons permettre à n'importe quel hôte d'y accéder, c'est pourquoi on a mis une valeur de "*" dans cette option.

```
# https://docs.djangoproject.com/en/dev/ref/settings/#allowed-hosts
ALLOWED_HOSTS = ['*', ]
```

- L'option **CACHES** permet de spécifier les paramètres de cache utilisés par Horizon. Dans notre cas, il est configuré pour utiliser Memcached en tant que backend de cache.

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '10.0.0.30:11211',
    },
}
```

- L'option **SESSION_ENGINE** définit le backend de stockage de session utilisé par Horizon.

```
#SESSION_ENGINE = 'django.contrib.sessions.backends.signed_cookies'
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
```

- L'option **OPENSTACK_HOST** permet de spécifier l'hôte OpenStack à contacter pour les appels d'API. Dans notre cas, il est configuré pour utiliser l'adresse IP de l'hôte OpenStack.
- L'option **OPENSTACK_KEYSTONE_URL** spécifie l'URL de l'API Keystone à utiliser pour l'authentification OpenStack.

```
OPENSTACK_HOST = "10.0.0.30"
#OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
OPENSTACK_KEYSTONE_URL = "http://10.0.0.30:5000/v3"
```

- L'option **TIME_ZONE** permet de spécifier le fuseau horaire utilisé par Horizon.

```
# The timezone of the server. This should correspond with the timezone
# of your entire OpenStack installation, and hopefully be in UTC.
TIME_ZONE = "Africa/Casablanca"
```

- Les options **WEBROOT**, **LOGIN_URL**, **LOGOUT_URL**, **LOGIN_REDIRECT_URL** définissent les URL utilisées par Horizon pour les pages de connexion et de déconnexion.
- L'option **OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT** permet de spécifier si le domaine par défaut doit être utilisé pour les utilisateurs ou les projets s'ils ne sont pas spécifiés.
- L'option **OPENSTACK_KEYSTONE_DEFAULT_DOMAIN** spécifie le domaine par défaut à utiliser pour les utilisateurs ou les projets s'ils ne sont pas spécifiés.

```
WEBROOT = '/dashboard/'
LOGIN_URL = '/dashboard/auth/login/'
LOGOUT_URL = '/dashboard/auth/logout/'
LOGIN_REDIRECT_URL = '/dashboard/'
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
```

Dans ce qui suit, on configure le fichier **openstack-dashboard.conf** pour utiliser WSGI (Web Server Gateway Interface) pour les applications Python.

```
[root@controller ~]# gedit /etc/httpd/conf.d/openstack-dashboard.conf
```

Les commandes suivantes permettent de spécifier les processus et groupes de processus WSGI pour le tableau de bord, ainsi que le préfixe du socket pour la communication entre Apache et les processus WSGI.

```
WSGIProcessGroup dashboard
WSGIApplicationGroup %{GLOBAL}
```

On redémarre le serveur web Apache pour que les modifications apportées à la configuration du tableau de bord OpenStack (Horizon) soient prises en compte.

```
[root@controller ~]# systemctl restart httpd
[root@controller ~]# vi /etc/nova/policy.json
```

Puis, on modifier le fichier **policy.json** pour autoriser les utilisateurs disposant des priviléges d'administrateur ou de propriétaire à accéder aux attributs étendus du serveur, tels que les métadonnées ou les informations de disque dur supplémentaires.

```
{  
  "os_compute_api:os-extended-server-attributes": "rule:admin_or_owner",  
}
```

Et on redémarre le service Nova API pour appliquer les modifications apportées à la politique de sécurité des instances dans le fichier "policy.json" de Nova.

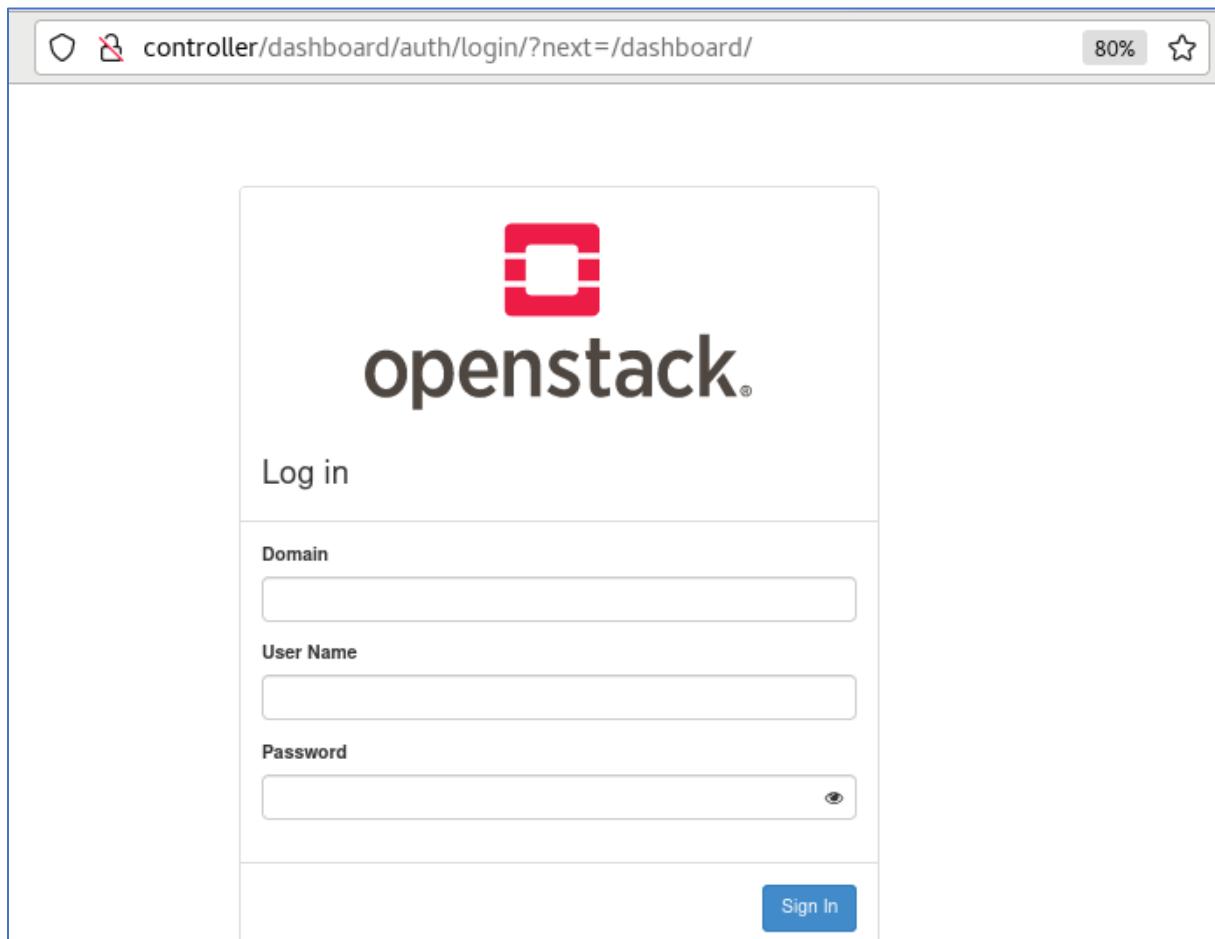
```
[root@controller ~]# systemctl restart openstack-nova-api
```

On modifie encore la politique SELinux puisqu'il est activé :

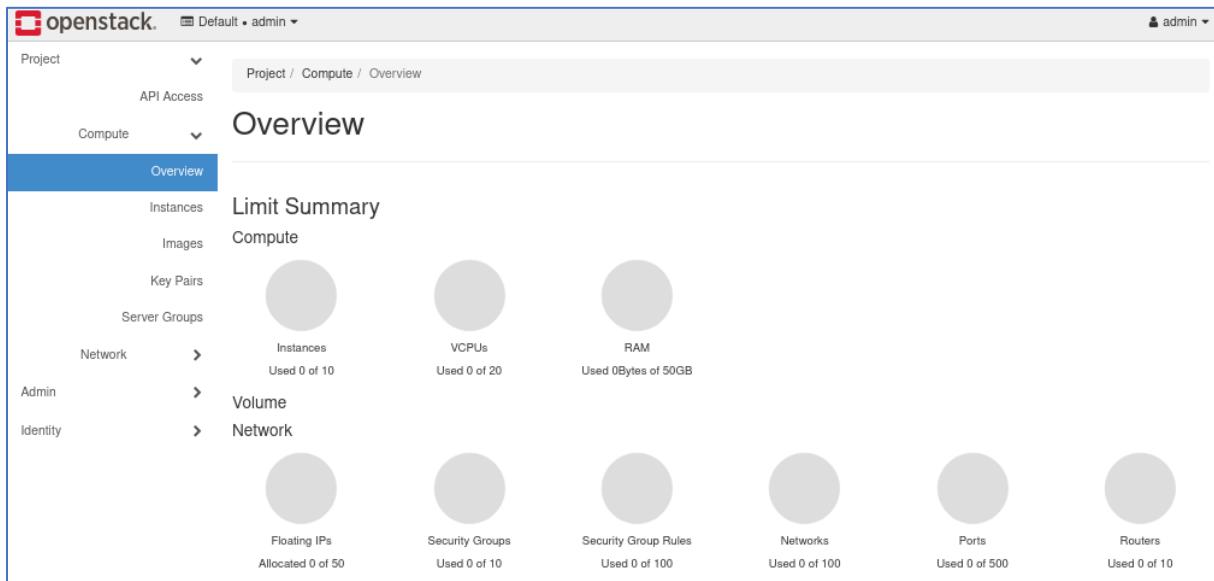
```
[root@controller ~]# setsebool -P httpd_can_network_connect on  
[root@controller ~]#
```

Après avoir accédé à l'URL <http://controller/dashboard> (\Rightarrow http://(nom d'hôte du serveur/dashboard/), l'écran suivant s'affiche, nous nous connectons alors avec un utilisateur dans Keystone.

Il est possible d'utiliser toutes les fonctionnalités si on se connecte avec l'utilisateur [admin] lorsque. Si on se connecte avec un utilisateur commun, il est possible d'utiliser ou de gérer ses propres instances.



Une fois la connexion réussie, l'écran suivant s'affiche. On peut contrôler Openstack sur ce tableau de bord.





Conclusion

En guise de conclusion, ce projet nous a permis de développer et d'approfondir nos connaissances en matière de cloud computing en le visualisant et en le mettant en pratique. Tout en explorant les différents services d'Openstack Vicoria, comprendre leur rôle et leur fonctionnement, et voir comment ils s'intègrent pour fournir une infrastructure de cloud évolutive et flexible.

Nous avons également découvert que OpenStack est une plateforme open source pour la gestion de l'infrastructure de cloud computing qui offre des fonctionnalités d'IaaS, de surveillance, de gestion et d'automatisation des opérations de cloud computing. L'IaaS et OpenStack sont caractérisées par l'évolutivité, la flexibilité, la réduction des coûts et l'automatisation, mais aussi par la complexité, la sécurité et les coûts à long terme. Les entreprises doivent donc peser ces avantages et ces limites avant de décider d'adopter une infrastructure de cloud computing.