Installation et prise en main

F. Diakhaté, R. Fihue

Date: 16/12/2018

Aperçu et Objectifs

Ce TP vous donnera un aperçu des principaux composants d'Openstack ainsi que des étapes nécessaires pour déployer une infrastructure minimale basée sur OpenStack. Pour chacun de ces composants, il y aura une étape d'installation puis un exemple d'utilisation de ces services. Ces services sont développés séparément mais ont souvent des dépendances les uns envers les autres, l'ordre d'installation est donc important. Vous étudierez aussi quelques aspects du fonctionnement interne de ces composants, notamment la partie réseau.

Ce TP est très largement inspiré de la documentation officielle d'OpenStack (Installation Guide) vous pourrez vous en inspirer et l'utiliser comme première source d'information en cas de problèmes :

- Installation Guide : https://docs.openstack.org/install-guide/
- Minimal Deployment: https://docs.openstack.org/install-guide/openstack-services.html#minimal-deployment-for-rocky
- Configuration Reference : https://docs.openstack.org/rocky/configuration/

Lancement des VMs

Créez un template pour le cluster openstack (dans `~/.pcocc/templates.yaml`):

```
#~/.pcocc/templates.yaml
openstack_tp:
  resource-set: ocluster
  inherits: centos75-cloud-pv
  user-data: "%{env:HOME}/openstack_tp"
  persistent-drives:
  - "/scratch/%{env:USER}/nova1.img":
      mmp: no
      cache: unsafe
  - "/scratch/%{env:USER}/nova2.img":
      mmp: no
      cache: unsafe
  - "/scratch/%{env:USER}/cinder.img":
      mmp: no
      cache: unsafe
```

Copiez le fichier cloud-init /scratch/tp/openstack/openstack_tp dans votre HOME et ajoutez y votre clef SSH. Créez à l'aide de *truncate* les 3 disques de 20G `/scratch/$USER/nova1.img` , `/scratch/$USER/nova2.img`, `/scratch/$USER/cinder.img`.

Lancez 2 VMs à l'aide de **pcocc** et du modèle **openstack** que vous venez de définir:

```
$ pcocc batch -t 600 -c 4 openstack:5
```

Puis connectez-vous sur la première VM et contrôlez que tout semble OK.

```
$ pcocc ssh builder@vm0
vm0 $ sudo ssh builder@compute1
vm1 $ sudo ssh builder@controller
```

Keystone : Identity Service

Pour rappel, le service **Keystone** est le service d'authentification et d'autorisation. Il implémente l'API **Identity** d'OpenStack et permet aussi la découverte des autres services. C'est donc le premier service à être installé car il est utilisé par les autres composants d'Openstack pour s'enregistrer et permettre d'initier les communications entre eux. De plus, il permettra par la suite l'authentification des différents utilisateurs et administrateurs du cluster OpenStack.

Installez Keystone à l'aide du script install_keystone, sauvegardez la sortie avec la commande tee

```
$ pcocc ssh builder@vm
$ sudo /usr/local/bin/install_keystone | sudo tee -a
/install_keystone.log
```

Vous pourrez ensuite lister les projets openstack, en créer de nouveaux, y ajouter des utilisateur avec différents rôles, etc …

Ajoutez le projet **service**:

```
$ sudo -i
# source /etc/openstack_admin_openrc
# openstack project create --domain default --description "Service
project, used by other openstack services" service
```

Puis un projet à votre nom (ou pseudo) **NOM_project**:

```
# openstack project create --domain default --description "TP Project"
NOM_project
```

Ajoutez un utilisateur (vous) et assignez le sur votre projet:

```
# openstack user create --domain default --password $(grep USER_PASS
/etc/openstack_pass | cut -f2 -d=) NOM
# openstack role create myrole
# openstack role add --project NOM_project --user NOM myrole
```

Adaptez le fichier **/etc/openstack_user_openrc** (fichier avec de quoi s'authentifier avec **OpenStack**) avec le nom du projet (**OS_PROJECT_NAME**) et le nom d'utilisateur (**OS_USERNAME**).

Vérifications :

```
# more /etc/openstack_{user,admin}_openrc | cat
# source /etc/openstack_admin_openrc
# openstack project list
# openstack user list
# openstack role assignment list -c Project -c User
# source /etc/openstack_user_openrc
# openstack project list
# openstack user list
# openstack role assignment list -c Project -c User
```

Répondez a la section 'Keystone' du Google forms.

Glance : Image service

Le service nommé **Glance** permets aux utilisateurs d'OpenStack d'uploader et de découvrir des données pouvant être utilisée par d'autres services. Notamment des images de VM avec les metadonnées associées.

Glance fournit une API REST permettant de récupérer ces metadonnées aussi bien que l'image elle-même.

Les données elles-même peuvent être stockées sur un filesystem local ou dans **Swift** le service de stockage object d'OpenStack.

Installez **Glance** à l'aide du script **install_glance**.

```
$ pcocc ssh builder@vm0
$ sudo /usr/local/bin/install_glance | sudo tee -a /install_glance.log
```

Vérifiez que tout est OK en vérifiant le catalogue de service OpenStack. Il doit contenir **Glance.**

```
$ sudo -i
# source /etc/openstack_admin_openrc
# openstack catalog list
```

Enfin, télécharger une image **CirrOS** et insérer l'image dans **Glance** :

```
# wget -O /tmp/image1 hpc01-ic/images/cirros.qcow2
# wget -O /tmp/image2 hpc01-ic/images/CentOS-7-x86_64-GenericCloud.qcow2
# openstack image create "cirros" --file /tmp/image1 --disk-format qcow2
--container-format bare --public
# openstack image create "centos7.5" --file /tmp/image2 --disk-format
qcow2 --container-format bare --public
```

Répondez a la section 'Glance' du Google forms.

Vérifications :

```
openstack image list
openstack image show cirros
```

Nova : Compute Service

Le service **Nova** est est l'un des principaux services mis en ouvre dans une solution l'**IaaS** basée sur OpenStack puisqu'il est en charge des ressources de calcul.

Le service **Nova** interagit avec **Keystone**, **Glance, Neutron** et **Cinder**. Tout cela se fait au travers des API REST.

Nova est composé de nombreux services (au sens système) qui gère chacun une partie du service :

- *nova-api*: API REST end-user
- *nova-api-metadata*: API REST pour les instance de l'IaaS
- *nova-compute:* service en charge du lancement des VMs
- *nova-placement-api*: API REST pour l'inventaire des instances
- *nova-scheduler*: Ordonnanceur des instances
- *nova-conductor*: Intermédiaire aux accès à la DB MySQL stockant les différents états des instances.
- *nova-novncproxy/nova-spicehtml5proxy/nova-xvpvncproxy*: Proxy permettant un accès aux consoles via leur protocoles respectifs.

L'installation de **Nova** se déroule sur 2 des VMs lancées précédemment:

- Installation des service de management **Nova** sur le controller (vm0)
- Installation du service **nova-compute** sur les hyperviseurs (vm1 et vm2)
- Enregistrement du service **nova-compute** sur le controller

Veillez à bien tourner les commandes sur les bons noeuds.

Installez la partie management de nova sur le noeud contrôleur (**vm0**)

```
$ pcocc ssh builder@vm0
vm0 $ sudo /usr/local/bin/install_nova_controller | sudo tee -a
/install_nova.log
```

Puis installez la partie **nova-compute** sur vm1:

```
$ pcocc ssh builder@vm1
vm1 $ sudo /usr/local/bin/install_nova_compute | sudo tee -a
/install_nova.log
```

Même chose sur vm2:

```
$ pcocc ssh builder@vm2
vm2 $ sudo /usr/local/bin/install_nova_compute | sudo tee -a
/install_nova.log
```

Et enfin enregistrer nos 2 noeuds hyperviseur sur vm0:

```
$ pcocc ssh builder@vm0
vm0 $ sudo /usr/local/bin/install_nova_controller_2 | sudo tee -a
/install_nova.log
```

Vérifications à fournir :

```
$ pcocc ssh builder@vm0
$ sudo -i
# source /etc/openstack_admin_openrc
# openstack compute service list
# openstack catalog list
# nova-status upgrade check
```

Répondez a la section ‘Nova’ du Google forms.

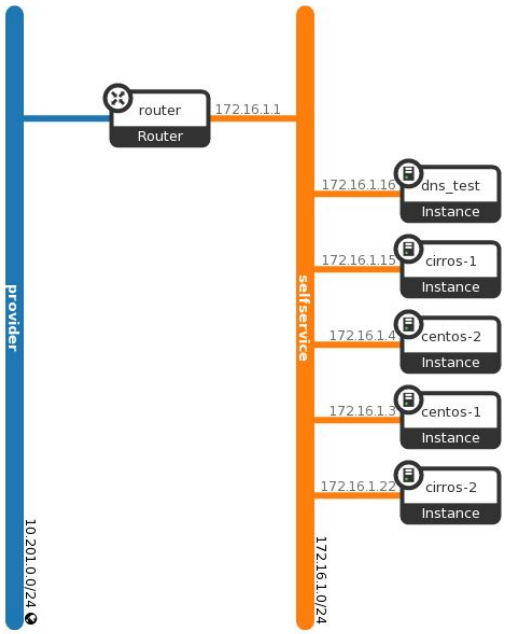
Neutron : Networking service

Le service **Neutron** est en charge de la gestion des réseaux virtuels d'OpenStack. Il permet aux utilisateurs de définir leurs propres réseaux virtuels en self-service.

Neutron est composé des composants suivants:

- neutron-server: API REST routant les requêtes vers le plug-in adapté
- plugins & agents: Fournissent un moyen de créer des ports réseau, des réseaux physiques ou virtuels et un adressage réseau. Ils s'adaptent aux différentes technologies en place :
    - Switchs CISCO physiques ou virtuels
    - OpenFlow
    - Open vSwitch
    - Bridge Linux
    - VMWare NSX
    - ...

Ici, nous utiliserons des *bridges linux* et ainsi que des *tunnels vxlan* pour les réseaux virtuels pour fournir une topologie réseau qui ressemblera à ceci :



L'installation de ce service s'effectue sur le noeud controlleur (vm0) et hyperviseur (vm1, vm2).

```
$ pcocc ssh builder@vm0
vm0 $ sudo /usr/local/bin/install_neutron_controller | sudo tee -a
/install_neutron.log
```

Puis sur les 2 noeuds hyperviseurs

```
$ pcocc ssh builder@vm1
vm1 $ sudo /usr/local/bin/install_neutron_compute | sudo tee -a
/install_neutron.log
```

```
$ pcocc ssh builder@vm2
vm2 $ sudo /usr/local/bin/install_neutron_compute | sudo tee -a
/install_neutron.log
```

Vérifications à fournir:

```
$ pcocc ssh builder@vm0
$ sudo -i
# source /etc/openstack_admin_openrc
# openstack network agent list
```

Répondez à la section 'Neutron' du Google forms.

Cinder : Block Storage Service

Le service **Cinder** permet de fournir des périphériques de stockage bloc à la demande appelés volumes. Un mécanisme de plugins permet à **Cinder** de fournir une à l'utilisateur une vue abstraite des moyens de stockage sous-jacent (NFS, SAN, iSCSI, Ceph, etc …).

L'API de **Cinder** sera installée sur le noeud contrôleur (**vm0)** alors que le stockage des données se fera sur un noeud dédié (**vm3**).

Tout d'abord, installons l'API:

```
$ pcocc ssh builder@vm0
vm0 $ sudo /usr/local/bin/install_cinder_controller | sudo tee -a
/install_cinder.log
```

Puis le serveur de stockage:

```
$ pcocc ssh builder@vm3
vm3 $ sudo /usr/local/bin/install_cinder_block_server | sudo tee -a
/install_cinder.log
```

Vérifications à fournir :

```
$ pcocc ssh builder@vm0
vm0 $ sudo -i
vm0 # source /etc/openstack_admin_openrc
vm0 # openstack volume service list
```

Répondez a la section 'Cinder' du Google forms.

Prise en main

Maintenant que les services **Keystone, Glance, Nova, Neutron et Cinder** sont installés, nous allons pouvoir réellement utiliser notre plateforme **IaaS**.

L'objectif est de créer 4 machines virtuelles partageant un même réseau virtuel et tenter de les faire communiquer. Il faudra ensuite accéder (via SSH) à l'une ou l'autre depuis l'extérieur de façon transparente. L'extérieur étant ici accessible depuis une autre VM (**vm4)**

Cette 'infrastructure' nécessite plusieurs choses:

- Un réseau virtuel : celui qui sera partagé entre nos 2 machines virtuelles
- Un sous-réseau IP : il allouera une IP dans le réseau virtuel à chacune de nos VMs
- Un routeur IP L3 : pour que les VMs accèdent à des ressources extérieures (DNS) et que l'extérieur puisse accéder aux services rendus par les VMs
- 2 VMs
- Un moyen d'y accéder via SSH:
    - Une IP
    - Une bi-clé RSA

Réseau

Nous allons mettre en place un réseau dit *self-service*. Celui-ci permettra aux end-user de créer autant de réseau virtuels que souhaité sans toutefois requérir des changement sur le réseau physique sous-jacent ie. des réseaux à la demande.

Il faut d'abord mettre en place le réseau qui sera utilisé pour accéder à notre infrastructure. Celui-ci est nommé réseau *external* en référence à un réseau *extérieur* utilisé pour accéder aux service fourni par la plateforme **IaaS**.

```
$ pcocc ssh builder@vm0
vm0 $ sudo -i
vm0 # source /etc/openstack_admin_openrc
vm0 # openstack network create --external --provider-physical-network
provider --provider-network-type flat external
```

Puis assigner une sous-réseau IP à ce dernier:

```
vm0 # openstack subnet create --network external --allocation-pool
start=10.201.0.100,end=10.201.0.250 --dns-nameserver 10.201.255.253
--gateway 10.201.0.1 --subnet-range 10.201.0.0/24 external
```

On peut ensuite mettre en place le réseau *self-service* qui sera utilisé par nos 2 VMs

```
vm0 # source /etc/openstack_user_openrc
vm0 # openstack network create --mtu 1400 selfservice
```

Et lui associer un sous-réseau IP:

```
vm0 # openstack subnet create --network selfservice --subnet-range
172.16.1.0/24 selfservice
```

Il faut ensuite créer le routeur IP L3 au centre de ces 2 sous-réseaux IP en tant que passerelle par défaut. Puis des IP flottantes accessibles depuis l'extérieur du réseau virtuel.

```
vm0 # openstack router create router
vm0 # openstack router add subnet router selfservice
vm0 # openstack router set router --external-gateway external
vm0 # openstack floating ip create external --tag IP1
vm0 # openstack floating ip create external --tag IP2
vm0 # openstack floating ip create external --tag IP3
vm0 # openstack floating ip create external --tag IP4
```

Machines Virtuelles

Les machines virtuelles (compute) d'OpenStack sont toutes dérivées d'une *flavor* correspondant aux caractéristiques matérielles de la machine. Ici nous allons créer un modèle de VM possédant 1 CPU, 64 Mb de RAM et 1 Gb d'espace disque et un second, plus gros, 4 CPUs, 512 Mb de RAM et 16Gb d'espace disque.

```
$ pcocc ssh builder@vm0
vm0 $ sudo -i
vm0 # source /etc/openstack_admin_openrc
vm0 # openstack flavor create --vcpus 1 --ram 64 --disk 1 m1.nano
vm0 # openstack flavor create --vcpus 1 --ram 512 --disk 16 m1.small
```

Il faut ensuite renseigner nos identifiants SSH (bi-clé) dans OpenStack:

```
vm0 # source /etc/openstack_user_openrc
vm0 # ssh-keygen -q -N "" -f /root/.ssh/id_rsa_os
vm0 # openstack keypair create --public-key /root/.ssh/id_rsa_os.pub
mykey
# Déploiement sur la VM “client”
vm0 # scp /root/.ssh/id_rsa_os* builder@vm4:.ssh/
```

Et ensuite donner les ports utilisés pour la connection SSH (entre autres)

```
vm0 # openstack security group rule create --proto icmp default
vm0 # openstack security group rule create --proto tcp --dst-port 22
default
```

Et enfin créer les machines virtuelles :

```
vm0 # openstack server create --flavor m1.nano --image cirros --network
selfservice --security-group default --key-name mykey cirros1
vm0 # openstack server create --flavor m1.nano --image cirros --network
selfservice --security-group default --key-name mykey cirros2
vm0 # openstack server create --flavor m1.small --image centos7.5
--network selfservice --security-group default --key-name mykey centos1
vm0 # openstack server create --flavor m1.small --image centos7.5
--network selfservice --security-group default --key-name mykey centos2
```

Assigner les IP flottantes aux différents serveurs lancés:

```
vm0 # openstack server add floating ip cirros1 $(openstack floating ip
list --tags IP1 -c 'Floating IP Address' -f value)
vm0 # openstack server add floating ip cirros2 $(openstack floating ip
list --tags IP2 -c 'Floating IP Address' -f value)
vm0 # openstack server add floating ip centos1 $(openstack floating ip
list --tags IP3 -c 'Floating IP Address' -f value)
vm0 # openstack server add floating ip centos2 $(openstack floating ip
list --tags IP4 -c 'Floating IP Address' -f value)
```

Puis créer et assigner des volumes aux VMs:

```
# openstack volume create --size 20 volume1
# openstack volume create --size 20 volume2
# openstack volume create --size 20 volume3
# openstack volume create --size 20 volume4
# openstack server add volume cirros1 volume1
# openstack server add volume cirros2 volume2
```

Récupérez l'IP flottante (`10.201.0.???`) d'un des serveurs *cirros:*

```
# openstack server list -c Name -c Networks
+----------+---------------------------------------+
| Name     | Networks                              |
+----------+---------------------------------------+
| cirros-1 | selfservice=172.16.1.15, 10.201.0.117 |
| centos-2 | selfservice=172.16.1.4, 10.201.0.113  |
| centos-1 | selfservice=172.16.1.3, 10.201.0.111  |
| cirros-2 | selfservice=172.16.1.22, 10.201.0.101 |
+----------+---------------------------------------+
```

Tentez de vous y connecter et vérifiez que le volume est bien présent (**vdb**) et de la bonne taille (**20GB**):

```
$ pcocc ssh builder@vm4
# ssh -i .ssh/id_rsa_os cirros@10.201.0.101
$ hostname
cirros-2
$ sudo fdisk -l /dev/vda /dev/vdb
Disk /dev/vda: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 127BB530-4FDD-4855-B653-77C39F7AE9C4

Device     Start      End Sectors  Size Type
/dev/vda1  18432 2097118 2078687 1015M Linux filesystem
/dev/vda15  2048   18431   16384 8M EFI System

Partition table entries are not in disk order.
Disk /dev/vdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Vérifications à fournir dans le google form:

```
$ pcocc ssh vm0
vm0 $ sudo -i
vm0 # source /etc/openstack_user_openrc
vm0 # openstack server list
vm0 # openstack floating ip list
vm0 # openstack volume list
vm0 # openstack router list
vm0 # openstack subnet list
```

Découverte du fonctionnement du réseau

Le fait qu'**OpenStack** permet d'instancier des réseau à la demande est extrêmement pratique mais ceci ne vient pas sans contrepartie : la complexité. En effet, pour fournir tout ceci de façon sûre et fiable **OpenStack** utilise des outils tout à fait classique mais certaines interactions entres-eux peuvent sembler difficile à appréhender au premier abord.

Les composants systèmes utilisés ici sont :

- Des **bridges :** équivalents à des switchs au sein du Kernel. Ne sert à rien tout seul, il faut y attacher des ports.
- Des **TAP:** ports ethernet kernel, usuellement utilisés pour fournir un accès externe à des machines virtuelles
- Des **vETH** : paire de ports Ethernet connectés par un câble virtuel.
- Des tunnels **VXLAN** : des tunnel L2 (Ethernet) permettant de connecter un ensemble de noeuds distants par un hub ethernet virtuel.
- Des **NetNS** (Network Namespace): piles réseau indépendantes entre elles et indépendante du système. Permet d'héberger plusieurs réseaux isolés sur un même OS.

Commandes utiles:

Pour lister les bridges réseau:

```
# brctl show
bridge name     bridge id       STP enabled     interfaces
brq1f4bb8d4-01  8000.525450000000     no              eth1
                                                    tap591076ac-13
                                                    tap98d1043c-af
brq5ad9706c-09  8000.0ef4a31ded69     no              tap4a34c7ea-aa
                                                    tapb4a98bdd-88
                                                    vxlan-46
```

Pour lister les namespace réseau et les IP associées :

```
# ip netns
qdhcp-5ad9706c-098a-4ca3-bf55-8c1a44cc0e3f (id: 2)
qdhcp-1f4bb8d4-01d6-4986-bc9e-6bfcfb568e8f (id: 1)
qrouter-ae118dd5-34f9-49ae-8d3c-f6ea25d65a9d (id: 0)

# ip netns exec qdhcp-5ad9706c-098a-4ca3-bf55-8c1a44cc0e3f ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
     valid_lft forever preferred_lft forever
     inet6 ::1/128 scope host
     valid_lft forever preferred_lft forever
2: ns-b4a98bdd-88@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc
noqueue state UP group default qlen 1000
     link/ether fa:16:3e:77:9c:3a brd ff:ff:ff:ff:ff:ff link-netnsid 0
     inet 169.254.169.254/16 brd 169.254.255.255 scope global
ns-b4a98bdd-88
     valid_lft forever preferred_lft forever
     inet 172.16.1.2/24 brd 172.16.1.255 scope global ns-b4a98bdd-88
     valid_lft forever preferred_lft forever
     inet6 fe80::f816:3eff:fe77:9c3a/64 scope link
     valid_lft forever preferred_lft forever
```

Pour lister les TAP et les MACs des VM associées à un tap:

```
# lsof /dev/net/tun
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
qemu-kvm 3020 qemu   30u   CHR 10,200      0t0 1324 /dev/net/tun
qemu-kvm 3104 qemu   27u   CHR 10,200      0t0 1324 /dev/net/tun
qemu-kvm 3321 qemu   28u   CHR 10,200      0t0 1324 /dev/net/tun
qemu-kvm 3345 qemu   29u   CHR 10,200      0t0 1324 /dev/net/tun
# pgrep -f fa:16:3e:99:58:c9
3104
# grep iff /proc/3104/fdinfo/27
iff:   tap9fff5e49-af
```

Pour les tunnel VXLAN:

```
# bridge fdb | grep vxlan
92:b7:ec:a4:2d:e6 dev vxlan-14 master brq9ede2adb-56 permanent
92:b7:ec:a4:2d:e6 dev vxlan-14 vlan 1 master brq9ede2adb-56 permanent
00:00:00:00:00:00 dev vxlan-14 dst 10.200.0.1 self permanent
fa:16:3e:b4:b9:56 dev vxlan-14 dst 10.200.0.1 self permanent
fa:16:3e:a4:a1:66 dev vxlan-14 dst 10.200.0.1 self permanent
```

Et une paire de **vETH:**

```
# ip netns exec qrouter-8d475e00-14ff-469c-af2f-b6c037a7f445 ip link list
type veth | grep -o '^[0-9]: [^ ]*'
2: qr-a5417962-99@if10:
3: qg-5e85026e-ea@if11:
## @if10 indique l'id de l'autre extrémité
# ip l l type veth | grep -o '^10: [^ ]*'
10: tapa5417962-99@if2:
```

Les règles de NAT du routeur L3 pour un namespace donné::

```
# ip netns exec qrouter-ae118dd5-34f9-49ae-8d3c-f6ea25d65a9d iptables -n
-v -t nat
```

Pour trouver la route empruntée pour une IP donnée:

```
# ip route get 10.201.0.101
10.201.0.101 dev eth1 src 10.201.0.1
     cache
```

Exercice:

Prenez l'IP d'une des VMs dans la sortie de :

```
# openstack floating ip list -c 'Floating IP Address'
+---------------------+
| Floating IP Address |
+---------------------+
| 10.201.0.123        |
| 10.201.0.118        |
| 10.201.0.103        |
| 10.201.0.107        |
+---------------------+
```

Et tentez une connection TCP depuis la **vm4**:

```
$ pcocc ssh builder@vm4
vm4 $ nc -v 10.201.0.123 22 < /dev/null > /dev/null
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 10.201.0.123:22.
SSH-2.0-OpenSSH_7.4
Ncat: 0 bytes sent, 21 bytes received in 0.21 seconds.
```

A l'aide des commandes données au dessus, à partir de l'IP et la MAC de destination tracez le cheminement du paquet TCP jusqu'à la VM associée à l'IP flottante. Répondez dans le google form

La réponse devra comporter les commandes utilisées et une description du cheminement (réseaux physiques/virtuels empruntés, namespace, interfaces, machine, routage).

Installation manuelle de Designate (Bonus)

Designate est un service OpenStack permettant d'insérer des entrée DNS de façon dynamique. Un exemple d'utilisation serait de donner un nom générique aux IP flottantes afin de ne pas utiliser directement d'IPs.

Tout en adaptant à l'architecture mise en place dans ce TP, suivez la documentation d'installation de **Designate** et mettez en place le service.

https://docs.openstack.org/designate/rocky/install/index.html

Les scripts d'installation utilisés au début peuvent vous être utile également.

Il est peut-être nécessaire de donner certains droits à **named** :

```
# chmod g+w -R /var/named
```

De même, il faudra configurer named pour écouter sur tous les ports:

```
options {
      ...
      listen-on port 35 { any; };
      allow-query { any; };
      dnssec-enable no;
      dnssec-validation no;
}
```

Pour intégrer **designate** et **neutron**, ajoutez ce qu'il suit dans */etc/neutron/neutron.conf*:

```
[DEFAULT]
dns_domain = pcocc.c-hpc.pedago.ensiie.fr
external_dns_driver = designate

[designate]
url = http://controller:9001/v2
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
region_name = RegionOne
username = designate
password = >>>>DESIGNATE PASS<<<<
```

Ainsi que ceci dans */etc/neutron/plugins/ml2/ml2_conf.ini*

```
[ml2]
extension_drivers=port_security,dns,dns_domain_ports
```

Puis relancez neutron-server:

```
systemctl restart neutron-server
```

Vérifications à fournir dans le google form:

```
$ pcocc ssh builder@vm0
vm0 $ sudo -i
vm0 # source /etc/openstack_admin_openrc
# openstack dns service list -c id -c hostname -c service_name
```

Test :

```
# openstack server create --flavor m1.nano --image cirros --network
selfservice dns_test
[...]
# openstack floating ip create external
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2018-10-16T16:23:55Z                 |
| dns_domain          |                                      |
| dns_name            |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 10.201.0.118                         |
| floating_network_id | 58aa4b73-231d-4cb2-9175-6ca1e151b3c1 |
| id                  | 6f58846c-4758-47b2-9e6d-0fcfa96f6055 |
| name                | 10.201.0.118                         |
| port_details        | None                                 |
| port_id             | None                                 |
| project_id          | e415b49feb234ee593b426c2cbba1c1c     |
| qos_policy_id       | None                                 |
| revision_number     | 0                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| subnet_id           | None                                 |
| tags                | []                                   |
| updated_at          | 2018-10-16T16:23:55Z                 |
+---------------------+--------------------------------------+
# openstack server add floating ip dns_test 10.201.0.118
# openstack recordset list pcocc.c-hpc.pedago.ensiie.fr. -c name -c
records -c status --name dns-test.pcocc.c-hpc.pedago.ensiie.fr.
+---------------------------------------+--------------+--------+
| name                                  | records      | status |
+---------------------------------------+--------------+--------+
| dns-test.pcocc.c-hpc.pedago.ensiie.fr. | 10.201.0.118 | ACTIVE |
+---------------------------------------+--------------+--------+
```