90 minutes avec OpenStack

Laurent Foucher
email: u03@u03.fr
blog: https://blog.u03.fr/
git : https://github.com/U03

Version : 2019/11/11

A propos de ce livre

J'ai écrit ce livre comme une introduction à OpenStack en français, car il est important pour moi de disposer de ressources en français, même s'il faut bien l'admettre la « langue officielle » de l'informatique est l'anglais.

Il est probable qu'il vous faille plus de 90 minutes pour lire ce livre, en particulier si vous reproduisez les différents exemples, réalisez les quelques exercices et surtout expérimentez par vous même.

Table des matières

Présentation de OpenStack

OpenStack est un ensemble de logiciels qui permettent de déployer des infrastructures virtuelles (Infrastructure As A Service).

Chaque infrastructure (projet ou tenant) est composée de serveurs virtuels (instances ou serveurs) qui fournissent la puissance de traitement, utilisent du stockage (disques virtuels, fichiers) et s'intègrent dans des réseaux virtuels (sous-réseaux, routeurs, pare-feu).

Les projets sont indépendants les uns des autres, les adresses IP à l'intérieur d'un projet sont privées, elles sont choisies librement par l'administrateur du projet, elles ne sont accessibles ni de l'extérieur, ni des autres projets.

L'administrateur du système OpenStack définit des adresses IP spécifiques qui sont routables depuis l'extérieur du Cloud, ces adresses sont appelées 'adresses IP flottantes'. Les administrateurs des différents projets peuvent demander que des adresses leur soient allouées pour rendre accessibles des instances depuis l'extérieur de leur projet.

L'administration et l'utilisation de OpenStack se fait à travers d'une collection d'API (Application Programming Interface) qui donnent à OpenStack sa puissance et sa souplesse. Ces API sont utilisables grâce à une Interface en Ligne de Commande (CLI), grâce à une interface web spécifique appelée Dashboard, ou grâce à tout programme ou script capable d'appeler les API directement.

Architecture de OpenStack

Il y a 4 groupes de services :

- Control
- Network
- Storage
- Compute

L'élément Control fait fonctionner les API, l'interface web, la base de données (MySQL, MariaDB ou PostgreSQL), et le bus de messages (RabbitMQ, Qpid ou ActiveMQ).

Les principaux éléments constituant OpenStack sont les suivants :

- Dashboard : L'interface web, elle utilise l'API d'OpenStack, elle est développée sur le framework 'Horizon' (ce qui lui vaut d'être parfois appelée 'Horizon')
- Keystone : Composant de gestion des identités, il gère l'authentification mais également il assure la partie catalogue : pour les points d'accès (endpoints) aux autres API, les projets, utilisateurs, rôles et tous les composants. Keystone est la pierre angulaire du système, tous les composants doivent appartenir à un projet (également appelé tenant ou propriétaire)
- Nova : gère les instances (serveurs virtuels)

- Glance : gère les images qui permettent de lancer les instances, il s'agit d'images de disques avec l'OS préinstallé et non d'images de DVD d'installation. Ces images sont rendues anonymes (suppression des adresses MAC, clés SSH…) la personnalisation est faite par le script '`cloud-init`' lors du boot de l'instance
- Neutron : Gestion du réseau virtuel (SDN : Software Defined Network)
- Cinder : Gestionnaire de stockage en mode blocs, il permet de créer des volumes (des disques durs virtuels) qui seront utilisables par les instances
- Swift : Gestionnaire de stockage d'objets, il permet de stocker des objets (fichiers) qui seront accessibles via une URL, avec ou sans authentification
- Ceilometer : Métrologie
- Heat : Orchestration (automatisation du déploiement des architectures).

L'orchestration est un point important dans les infrastructures Cloud, elle permet de déployer des infrastructures de façon automatique, mais surtout de les redéployer autant de fois que nécessaire, en particulier pour réaliser la mise en production des applications.

La CI/CD (Intégration Continue / Déploiement Continu) est une méthode et un ensemble d'outils qui permettent à partir d'un référentiel (de programmes, de paramètres) d'automatiser la création (intégration) d'artefacts (paquetages livrables), puis d'automatiser le déploiement de ces artefacts vers différents environnements (intégration, qualification, recette, pré-production, production).

L'orchestration et la CI/CD associées aux technologies du Cloud rendent les infrastructures « jetables » puisque facilement déployables, que ce soient les serveurs, les logiciels installés, le paramétrage du système d'exploitation et des logiciels associés, et même l'application.

Ainsi seules données sont conservées de façon perenne.

Installation de DevStack

DevStack est une distribution simplifiée d'OpenStack qui fait fonctionner les éléments nécessaires à un environnement de développement sur un serveur unique.

La distribution DevStack déploie les éléments principaux d'OpenStack, elle est destinée à la formation et au développement, elle est déconseillée en production.

Il est conseillé d'avoir au moins 16Go de RAM disponibles sur la machine cible.

Nous commençons par préparer le serveur physique en installant le système d'exploitation « Ubuntu Server 18.04.3 LTS » (ou la dernière version LTS disponible sur le site de Ubuntu):

- https://www.ubuntu.com/download/server

LTS signifie Long Term Support afin de différencier les versions de production qui sont supportées durant 5 ans minimum, des versions dites de développement qui servent à mettre au point la release LTS suivante.

Les versions de développement contiennent des versions plus récentes des produits par rapport aux versions LTS mais elles ne disposent d'aucun support et ne sont pas nécessairement stables.

Une installation par défaut de Ubuntu est suffisante, ne pas oublier d'ajouter l'option '`SSH Server`' afin de pouvoir accéder à votre serveur en SSH. La procédure d'installation de Ubuntu vous propose de créer un compte utilisateur, créez un compte nommé '`stack`' avec le mot de passe de votre choix (ce compte aura droit d'utiliser la commande '`sudo`' en spécifiant son mot de passe),

L'installation de DevStack est simple, elle ne nécessite que l'installation du client Git, les autres packages nécessaires seront installés automatiquement. Un fichier de configuration minimal permet de procéder au déploiement de DevStack.

Nous réalisons les étapes suivantes :

- Modification du fichier `sudoers` pour autoriser l'utilisateur `stack` à passer des commandes à la place de `root` sans avoir à spécifier de mot de passe.
- Mise à jour de l'index des packages de `apt`, puis installation du package `git`
- Clone du repository Git de DevStack
- Création du fichier de configuration permettant l'installation de DevStack. Modifiez le paramètre `HOST_IP` pour mettre l'adresse IP du serveur sur lequel vous procédez à l'installation. Tous les mots de passe auront pour valeur '`topsecret`'.
- Le paramètre `VOLUME_BACKING_FILE_SIZE` permet de spécifier l'espace disque qui sera alloué à Cinder pour stocker les différents volumes. L'allocation est faite en mode 'thin provisionning', l'espace utilisé sur le disque physique dépend de l'utilisation réelle qui est faite par Cinder.
- Installation de DevStack grâce à la commande '`./stack.sh`' (la commande '`./unstack.sh`' permettra de procéder à la désinstallation si nécessaire).

```
stack@microwave:~$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee -a /etc/sudoers
```

```
stack@microwave:~$ sudo apt-get update
stack@microwave:~$ sudo apt-get -y install git

stack@microwave:~$ git clone https://git.openstack.org/openstack-dev/devstack

stack@microwave:~$ cd devstack
stack@microwave:~/devstack$ cat << EOF > local.conf
[[local|localrc]]
HOST_IP=192.168.1.211
GIT_BASE=http://git.openstack.org
ADMIN_PASSWORD=topsecret
DATABASE_PASSWORD=\$ADMIN_PASSWORD
RABBIT_PASSWORD=\$ADMIN_PASSWORD
SERVICE_PASSWORD=\$ADMIN_PASSWORD
#
# Activation de Swift sans replication
#
enable_service s-proxy s-object s-container s-account
SWIFT_HASH=66a3d6b56c1f479c8b4e70ab5c2000f5
SWIFT_REPLICAS=1
SWIFT_DATA_DIR=$DEST/data/swift
#
# Activation du composant HEAT et de l'onglet HEAT dans le dashboard
#
enable_plugin heat https://git.openstack.org/openstack/heat
enable_plugin heat-dashboard https://git.openstack.org/openstack/heat-dashboard
#
# Agrandissement volume pour Cinder
#
VOLUME_BACKING_FILE_SIZE=100G
EOF

stack@microwave:~/devstack$ ./stack.sh
    […/…]
=========================
DevStack Component Timing
 (times are in seconds)
=========================
run_process            53
test_with_retry         4
apt-get-update          2
osc                   359
wait_for_service       26
git_timed            1100
dbsync                107
pip_install           982
apt-get               736
-------------------------
Unaccounted time      848
=========================
Total runtime        4217

This is your host IP address: 192.168.1.211
This is your host IPv6 address: ::1
Horizon is now available at http://192.168.1.211/dashboard
Keystone is serving at http://192.168.1.211/identity/
The default users are: admin and demo
The password: topsecret
```

L'installation de DevStack dure jusqu'à 1 heure en fonction de la puissance de votre serveur mais aussi de la vitesse de votre connexion à Internet pour télécharger les éléments nécessaires.

Une fois DevStack installée vous pouvez autoriser vos futures machines virtuelles à se connecter à Internet (et à votre réseau privé) en paramétrant le NAT comme ceci :

```
stack@microwave:~$ 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: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 94:18:82:38:17:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.211/24 brd 192.168.1.255 scope global dynamic eno1
       valid_lft 73654sec preferred_lft 73654sec
    inet6 fe80::9618:82ff:fe38:1730/64 scope link
       valid_lft forever preferred_lft forever
stack@microwave:~$ sudo iptables -t nat -A POSTROUTING -o eno1 -j MASQUERADE
```

Il arrive que le réseau privé créé défaut créé par DevStack ne comporte pas de DNS, vous pouvez utiliser ceux de Google, ou de votre réseau privé comme ceci :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack subnet set private-subnet --dns-nameserver 8.8.8.8 --dns-nameserver 8.8.4.4
```

Premier contact avec OpenStack

Les projets et les utilisateurs

Toutes les ressources doivent appartenir à un projet, un utilisateur ne peut utiliser les ressources d'un projet que si un rôle lui a été attribué sur ce projet.

Certaines ressources ne sont pas liées à un projet, mais à un utilisateur, c'est le cas par exemple des clés SSH que nous verrons un peu plus loin.

L'API et l'interface en ligne de commande

Les différents composants de OpenStack déploient des API (Application Programing Interface) avec lesquelles il est possible de communiquer en utilisant le protocole TCP via des points d'entrée (endpoint) catalogués dans KeyStone.

Un utilitaire en ligne de commande (CLI – Command Line Interface) est fourni pour s'interfacer avec les composants OpenStack de façon simple. Il est également téléchargeable séparément pour pouvoir gérer son infrastructure depuis une machine distante.

Ces API sont également utilisables par des utilitaires d'automatisation (orchestration) qui permettent de déployer des infrastructures à la demande de façon reproductible.

On l'a dit, Keystone est le point d'entrée du système, il gère l'authentification ainsi que le catalogue des différents projets (tenants) et endpoints.

Pour interagir avec OpenStack il faut disposer du endpoint de KeyStone, d'un nom d'utilisateur, d'un mot de passe et du nom d'un projet sur lequel on souhaite travailler (ou du nom du projet d'admin).

Après authentification KeyStone va fournir un jeton d'authentification (token), ainsi que la liste des endpoints des autres API parmi lesquels on peut choisir le endpoint nécessaire (par exemple Glance pour la gestion des images…) Ce jeton permet ensuite de s'authentifier auprès des autres services, un jeton a une durée de vie limitée qui est donnée par KeyStone en même temps que la valeur du jeton.

Le client CLI de OpenStack est appelé '`openstack`', il utilise des variables d'environnement ou des paramètres en ligne de commande pour se connecter à l'infrastructure OpenStack. Cette commande permet de passer des commandes à tous les composants de OpenStack.

DevStack est fourni avec script appelé '`openrc`' qui positionne les variables nécessaires, il prend 2 paramètres facultatifs, le premier est le nom d'utilisateur ('`demo`' par défaut), le second est le nom du projet ('`demo`' par défaut).

Ci-dessous nous appelons le script `openrc` pour nous placer avec le user `demo` dans le projet `demo`, dans les variables positionnées par le script on note l'URL d'accès KeyStone qui est le point d'entrée du système :

```
stack@microwave:~$ cd ~/devstack
stack@microwave:~/devstack$ source openrc demo demo
stack@microwave:~/devstack$ set | grep -P "^OS_"
OS_AUTH_TYPE=password
OS_AUTH_URL=http://192.168.1.211/identity
OS_CACERT=
OS_IDENTITY_API_VERSION=3
```

```
OS_PASSWORD=topsecret
OS_PROJECT_DOMAIN_ID=default
OS_PROJECT_NAME=demo
OS_REGION_NAME=RegionOne
OS_TENANT_NAME=demo
OS_USERNAME=demo
OS_USER_DOMAIN_ID=default
OS_VOLUME_API_VERSION=2
```

La syntaxe générale de la commande '`openstack`' est :

`openstack` *`objet opération paramètres`*:

Maintenant que les variables d'environnement nécessaires sont positionnées nous pouvons appeler l'utilitaire '`openstack`' pour avoir la liste des projets sur lesquels l'utilisateur a des droits :

```
stack@microwave:~$ openstack project list
+----------------------------------+--------------------+
| ID                               | Name               |
+----------------------------------+--------------------+
| 8eb8bd6dcd2447b2bc8224b4ceb64672 | demo               |
| 8feed042b5504d3fa074fca9ae4694f5 | invisible_to_admin |
+----------------------------------+--------------------+
```

Les images

L'installation d'un serveur physique est réalisée en installant le système d'exploitation sur un volume (disque dur) en démarrant le serveur sur un DVD d'installation.

La création d'une instance (serveur virtuel) se fait en utilisant une image spécifique préparée à partir d'un disque sur lequel un système d'exploitation a été installé.

Les éditeurs de systèmes d'exploitation fournissent ces images 'Cloud Ready' au même titre que les images de média d'installation.

Les images 'Cloud Ready' sont anonymisées (suppression des adresses MAC des cartes réseau, des clés SSH) et un script spécifique appelé '`cloud-init`' est installé dessus, il prendra en charge le paramétrage des instances lors de leur démarrage.

Les volumes

Les volumes sont des disques durs virtuels (on parle de stockage en mode 'blocs'), ils sont créés par exemple lors de la création d'une instance pour y placer le système d'exploitation présent sur l'image utilisée.

Dans le Cloud l'infrastructure est virtuelle, habituellement elle est également éphémère car détruite et redéployée par l'Orchestration à chaque fois que cela est nécessaire (par exemple pour installer une mise à jour de l'application ou du système d'exploitation).

C'est pourquoi on déploie le système d'exploitation sur un volume qui ne contiendra aucune donnée, il sera détruit et recréé à chaque mise en production en même temps que l'instance qui l'utilisait.

On crée séparément des volumes sur lesquels on va stocker les données de façon pérenne, lors des mises en production ces volumes seront détachés des instances auxquelles ils étaient connectés, ils seront ensuite reconnectés aux instances créées par l'Orchestration lors du déploiement de la mise en production.

Les clés SSH

L'utilisation de mots de passe pour se connecter à un serveur n'est pas sécurisée, lors de la création d'une instance on spécifie une clé publique qui sera ajoutée par `cloud-init` aux clés autorisées de l'utilisateur par défaut pour permettre la connexion en SSH à l'instance.

Le nom de l'utilisateur par défaut est le nom de la distribution (`centos` pour CentOS, `debian` pour Debian et `ubuntu` pour Ubuntu)

Il existe deux possibilités concernant les clés SSH :

- Utiliser un bi-clé existant, et ajouter la clé publique dans la base de données de OpenStack
- Faire tirer un bi-clé par la commande OpenStack, la clé publique sera automatiquement ajoutée dans la base de donnés de OpenStack, la clé privée associée devra être conservée pour être utilisée lors de la connexion à l'instance

Les clés SSH de type DSA ne sont pas utilisables avec OpenStack, en effet ce type de clés est reconnu comme non sûr. Pour pouvoir utiliser une clé existante elle doit être obligatoirement de type RSA.

Utilisation d'une clé existante

La commande `openstack keypair create` avec le paramètre `--public-key` permet d'ajouter une clé publique SSH sans la base de données OpenStack.

La clé appartient à utilisateur, elle n'appartient pas à un projet en particulier, elle est utilisable pour tous les projets de l'utilisateur. Les autres utilisateurs qui ont accès aux mêmes projets que vous n'ont pas accès à vos clés.

```
stack@microwave:~$ cd devstack/
stack@microwave:~/devstack$ source openrc demo
stack@microwave:~/devstack$ openstack keypair create --public-key ~/.ssh/id_rsa.pub "Foucher Laurent RSA"
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | 71:1d:81:03:3c:46:97:b8:e8:88:eb:e2:b2:15:6c:f5 |
| name        | Foucher Laurent RSA                             |
| user_id     | 7436b216d6ee4a1b809c723579c0e2bc                |
+-------------+-------------------------------------------------+
stack@microwave:~/devstack$ openstack user show 7436b216d6ee4a1b809c723579c0e2bc
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| email               | demo@example.com                 |
| enabled             | True                             |
| id                  | 7436b216d6ee4a1b809c723579c0e2bc |
| name                | demo                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

Pour information un bi-clé SSH de type RSA s'obtient comme ceci :

```
stack@microwave:~$ ssh-keygen -t rsa -b 4096 -C "Ma cle SSH"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/stack/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stack/.ssh/id_rsa.
Your public key has been saved in /home/stack/.ssh/id_rsa.pub.
The key fingerprint is:
```

```
SHA256:MvDIiOz6TPR+ACWZn5DT1vfyLLBGtMyog54UPX3I118 Ma cle SSH
The key's randomart image is:
+---[RSA 4096]----+
|    = .          |
|  B + o .        |
|   .B+*.o..      |
|.o.=+B*o...  E   |
|.o+o+.*oS+. .    |
|ooo.. oo. o.     |
|o.o..o   .       |
|.= .  .          |
|..o ..           |
+----[SHA256]-----+
```

Création d'un bi-clé par la commande OpenStack

La commande `openstack keypair create` avec le paramètre `--private-key` permet de tirer un bi-clé et d'ajouter la clé publique SSH dans la base de données OpenStack. La clé privée est sauvegardée dans le fichier spécifié en paramètre, il faut faire très attention en manipulant ce fichier, il est à traiter avec la même prudence qu'un mot de passe en clair. En particulier attention aux droits sur le fichier qui sont ceux par défaut de l'utilisateur (umask) et sont trop permissifs :

```
stack@microwave:~/devstack$ openstack keypair create --private-key ~/cle_privee "Cle Generee par OpenStack"
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| fingerprint | ea:b3:47:9a:de:0d:a4:61:37:e3:1e:6a:5d:ce:ba:bd |
| name        | Cle Generee par OpenStack                       |
| user_id     | 7436b216d6ee4a1b809c723579c0e2bc                |
+-------------+-------------------------------------------------+
stack@microwave:~/devstack$ ls -l ~/cle_privee
-rw-rw-r-- 1 stack stack 1675 Feb 11 22:19 /home/stack/cle_privee
stack@microwave:~/devstack$ chmod 600 ~/cle_privee
-rw------- 1 stack stack 1675 Feb 11 22:19 /home/stack/cle_privee
```

Les quotas et les gabarits

A chaque projet sont associés des quotas qui définissent la quantité maximale de chaque type de ressources qui peuvent être allouées par le projet :

- Instances: le nombre de serveurs virtuels
- vCPU : le nombre de processeurs virtuels
- RAM : la mémoire vive totale qui peut être allouée aux différentes instances
- Adresses IP flottantes (adresses IP accessibles depuis l'extérieur du projet
- Groupes de sécurité : les pare-feu placés en frontal des instances pour filtrer les flux entrants et sortants des instances
- Volumes : les disques durs virtuels qui permettent de stocker le système d'exploitation ou les données des instances (en nombre et en taille totale)

Les administrateurs des projets peuvent commander des instances de serveurs virtuels parmi différents modèles (vCPU, RAM…) appelés gabarits (ou flavor en anglais dans le dashboard et le CLI). Seul l'administrateur OpenStack peut définir ces gabarits.

```
stack@microwave:~$ openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 42 | m1.nano   |    64 |    0 |         0 |     1 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
```

```
| 84 | m1.micro  |   128 |    0 |         0 |     1 | True      |
| c1 | cirros256 |   256 |    0 |         0 |     1 | True      |
| d1 | ds512M    |   512 |    5 |         0 |     1 | True      |
| d2 | ds1G      |  1024 |   10 |         0 |     1 | True      |
| d3 | ds2G      |  2048 |   10 |         0 |     2 | True      |
| d4 | ds4G      |  4096 |   20 |         0 |     4 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

Les réseaux

OpenStack permet de construire des infrastructures complètes, y compris au niveau réseau.

En créant des routeurs et des réseaux on crée des infrastructures qui se rapprochent des infrastructures 'legacy' (héritées).

Les réseaux correspondent à des réseaux physiques, un réseau est composé d'au moins un sous-réseau.

Les sous-réseaux dans Neutron correspondent à des sous-réseaux au niveau IP, leur définition est composée des éléments suivants :

- La plage d'adresse, elle est définie en notation CIDR : adresse de réseau + longueur de masque (par exemple : `192.168.1.0/24`)
- Une plage d'adresse DHCP permettant d'attribuer dynamiquement des adresses aux instances
- La passerelle par défaut du sous-réseau
- Les DNS qui seront utilisés par les serveurs

Des routeurs permettent d'interconnecter les réseaux entre eux.

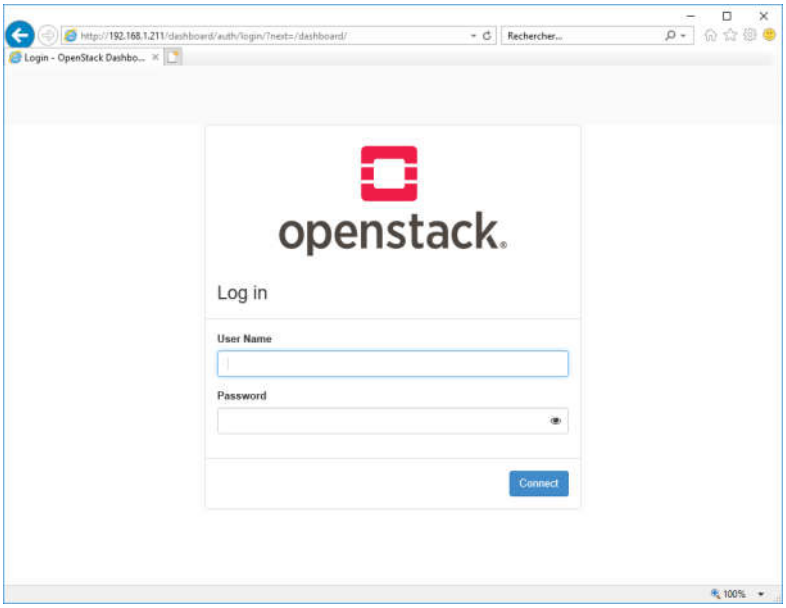
Des ports, qui correspondent à des prises réseaux (carte réseau de serveur, port de routeur…)

Nous verrons comment construire des réseaux dans la partie consacrée à l'orchestration Heat.

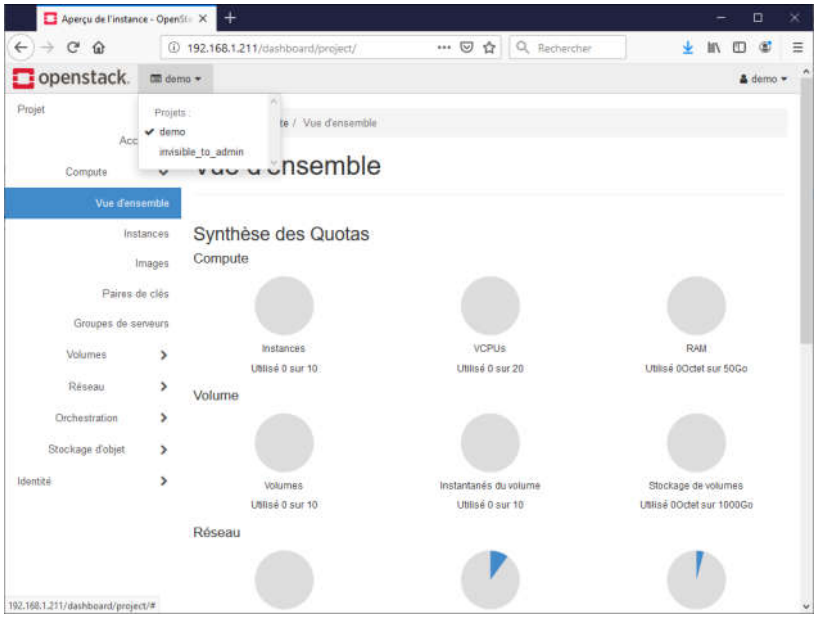
Connexion au Dashboard

On pourrait traduire ‘Dashboard’ par ‘Tableau de bord’, il est également appelé ‘Horizon’. Il s’agit de l’interface graphique de OpenStack.

Le Dashboard est utilisable avec n’importe quel navigateur moderne. Par défaut lors de l’installation de DevStack les utilisateurs ‘`demo`’ et ‘`admin`’ sont créés avec le mot de passe choisi lors de l’installation (il est rappelé à la fin de l’installation de DevStack en même temps que l’URL d’accès au Dashboard).

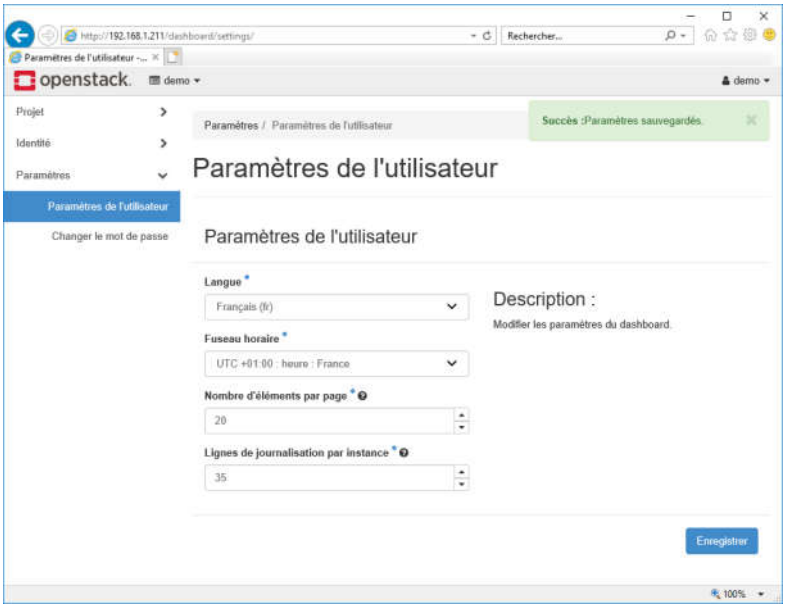


Lors de la connexion au Dashboard un premier écran s’affiche avec les différents quotas dont nous avons parlé. A gauche du bandeau supérieur une liste déroulante permet de passer d’un projet à l’autre parmi les différents projets sur lesquels nous avons des droits. A droite est rappelé quel utilisateur est connecté, un menu déroulant permet de modifier certains réglages (en particulier la langue d’affichage du Dashboard et le fuseau horaire).



Nous utilisons l'option «Paramètres»du menu déroulant de droite pour change la langue de l'interface et nous placer dans le fuseau horaire de notre choix. Il est possible de se placer en « UTC » (Temps Universel) ou « UTC+1 : France ».

Souvent les serveurs sont réglés sur le fuseau horaire UTC, pour des raisons de commodité dans les grandes infrastructures Cloud qui s'étalent sur plusieurs fuseaux horaires, dans ce cas choisir UTC dans le Dashboard permet de rester cohérent avec l'heure des serveurs.



Notre première instance

A savoir avant de lancer notre première instance

Nous allons lancer notre première instance grâce au Dashboard, mais auparavant nous allons discuter d'un point essentiel sur les volumes et sur l'infrastructure d'OpenStack.

Nous avons dit qu'OpenStack est (entre autres) composé de nœuds (serveurs) de type « Compute » (ce sont ceux sur lesquels s'exécutent nos instances) et de nœuds de type « Storage » (ce sont ceux qui ont en charge le stockage des volumes de données).

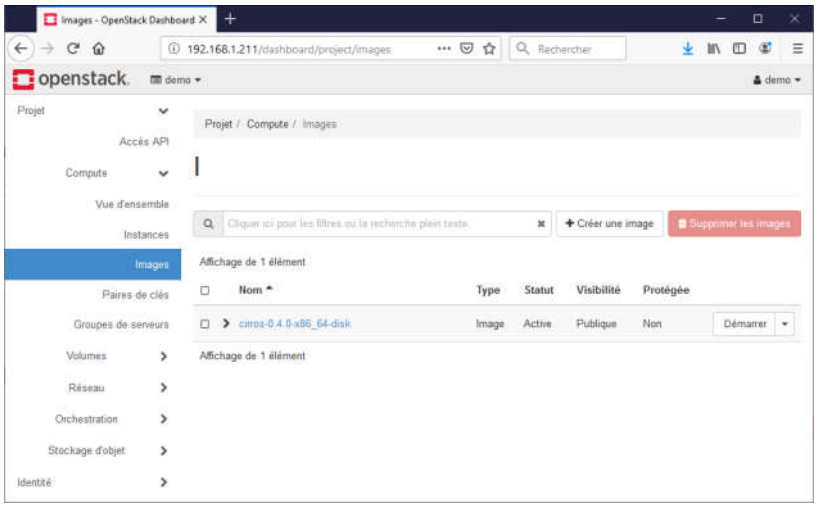
Il possible de lancer une instance en laissant OpenStack créer un volume pour le système d'exploitation directement sur l'espace de stockage du nœud « compute ». Cependant en cas de panne du nœud (ou si l'administrateur décide de l'arrêter pour une mise à jour par exemple) alors l'instance et les données présentes sur ce volume sont perdues…

La bonne façon de faire est de créer un volume à partir de l'image du système d'exploitation que nous souhaitons utiliser, ce volume sera utilisé pour lancer notre instance.

Pour nos premiers tests nous allons utiliser une image de système d'exploitation spécifiquement faite pour les tests, elle est légère, ne nécessite pas beaucoup de ressources, n'a pas de nombreuses fonctionnalités et ne doit pas être utilisée en production pour des raisons de sécurité. Son nom est 'Cirros', elle a été téléchargée depuis Internet et ajoutée à OpenStack par la procédure d'installation de DevStack.

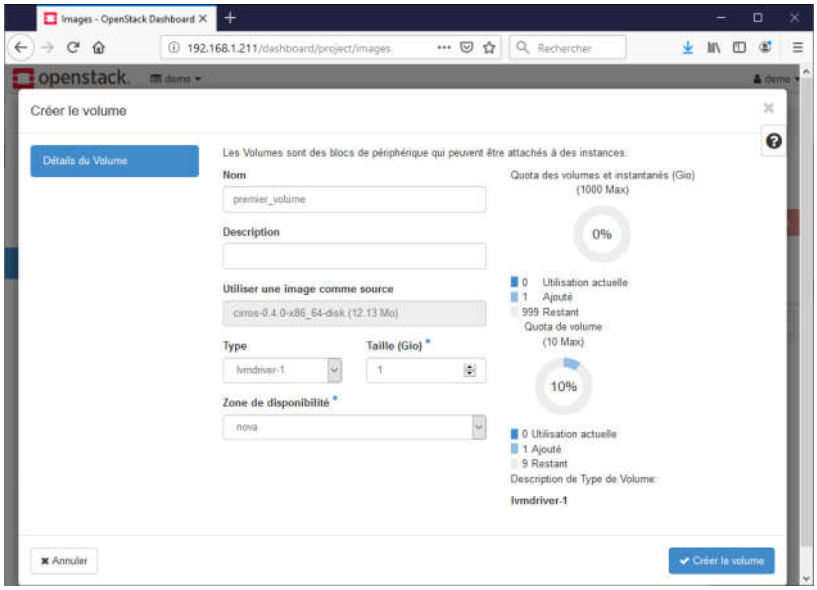
Lancement de notre première instance

Dans le Dashboard nous affichons l'onglet « Projet → Compute → Images », ici nous voyons l'image de Cirros. A droite de l'image 'cirros' nous avons un menu déroulant et nous choisissons « Créer le volume »

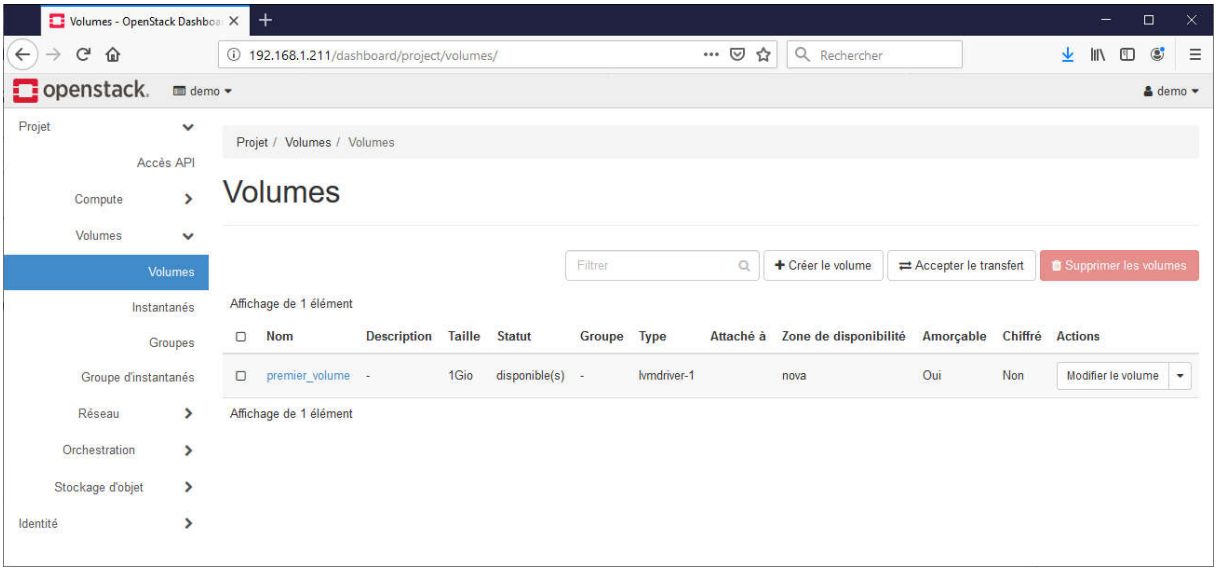


Nous appelons notre volume « premier_volume », la source est déjà renseignée puisque nous appelons cette fonction depuis l'écran « Images », nous laissons la taille à 1Go (c'est largement

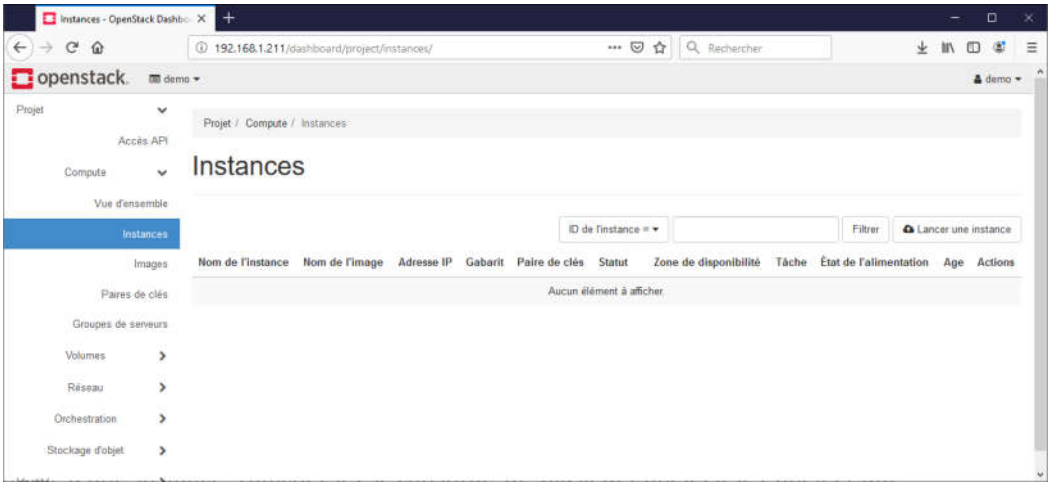
suffisant pour Cirros), les paramètres « Type » et « Zone de disponibilité » doivent être laissés à leur valeur par défaut, cliquons sur « Créer le volume ».



Nous affichons l'onglet « Projet → Volumes → Volumes », nous voyons le volume que nous venons de créer, il fait 1Go, son statut est « disponible » cela signifie qu'il n'est attaché à aucune instance, il est « amorçable » car il a été créé à partir d'une image de système d'exploitation, ça veux dire qu'on peut s'en servir pour lancer une instance.

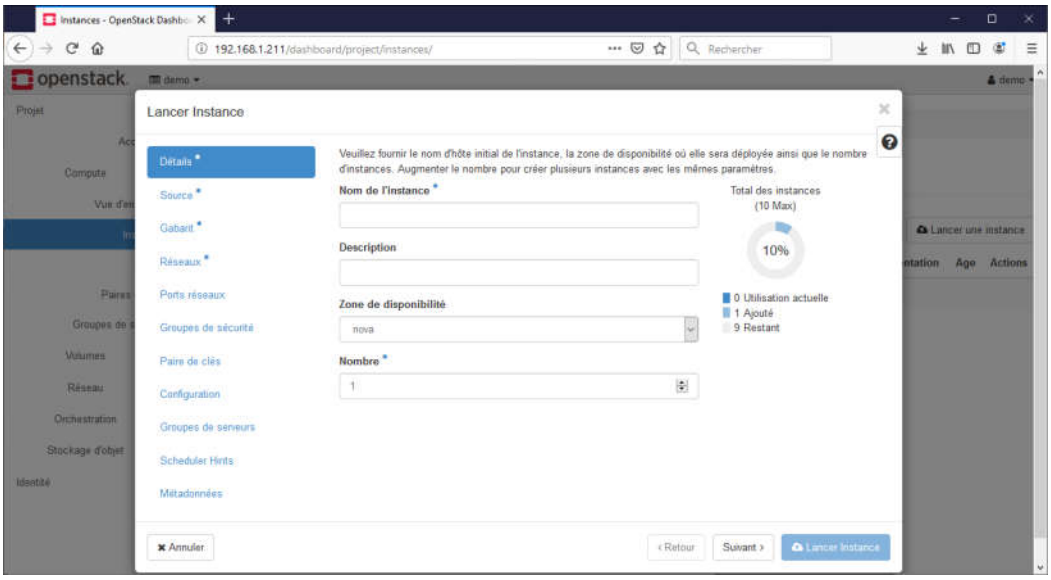


Nous allons pouvoir créer l'instance en nous servant de ce volume, nous allons dans l'onglet « Projet → Compute → Instances » et nous cliquons sur « Lancer une instance » :

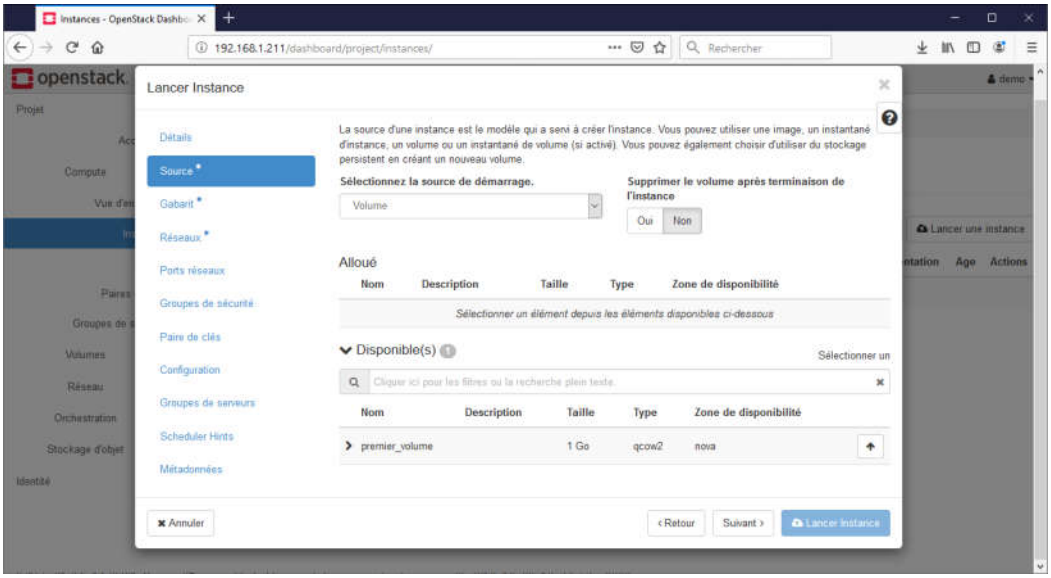


L'écran de lancement d'une instance s'affiche, il est constitué de plusieurs onglets, les onglets qui doivent être remplis sont suivis d'une étoile « * », le bouton « Lancer l'Instance » reste en grisé tant que tous les onglets requis ne sont pas complétés.

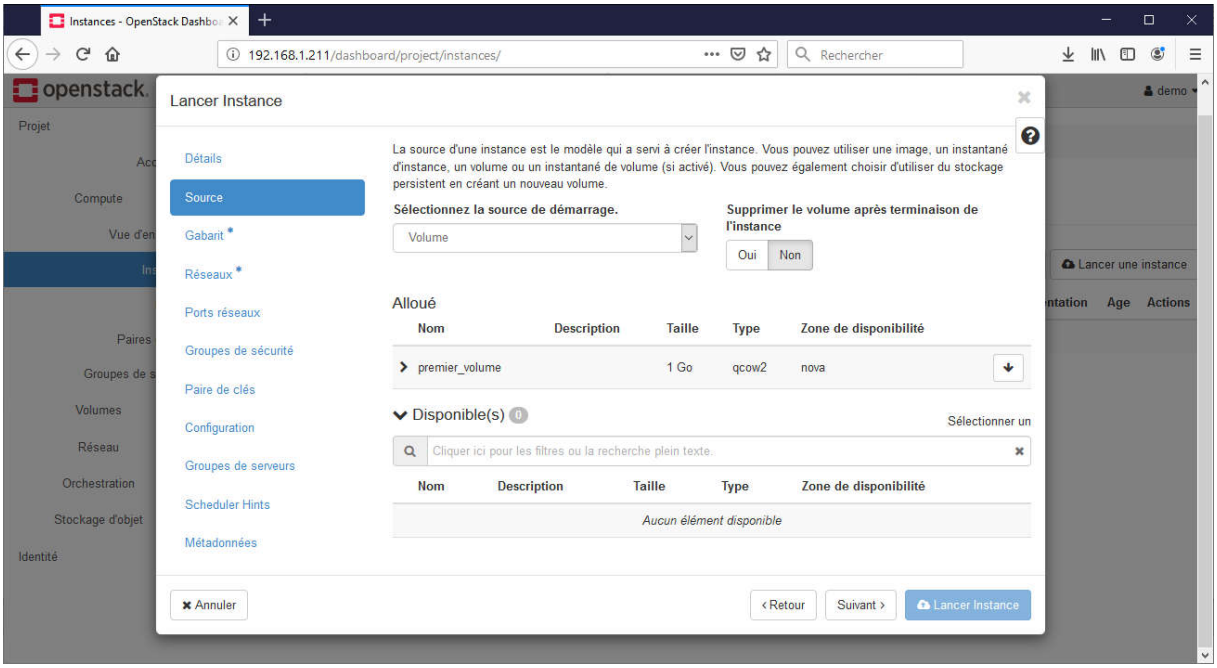
Nous entrons « premiere_instance » dans le champ « Nom de l'instance » et nous cliquons sur l'onglet « Source » :



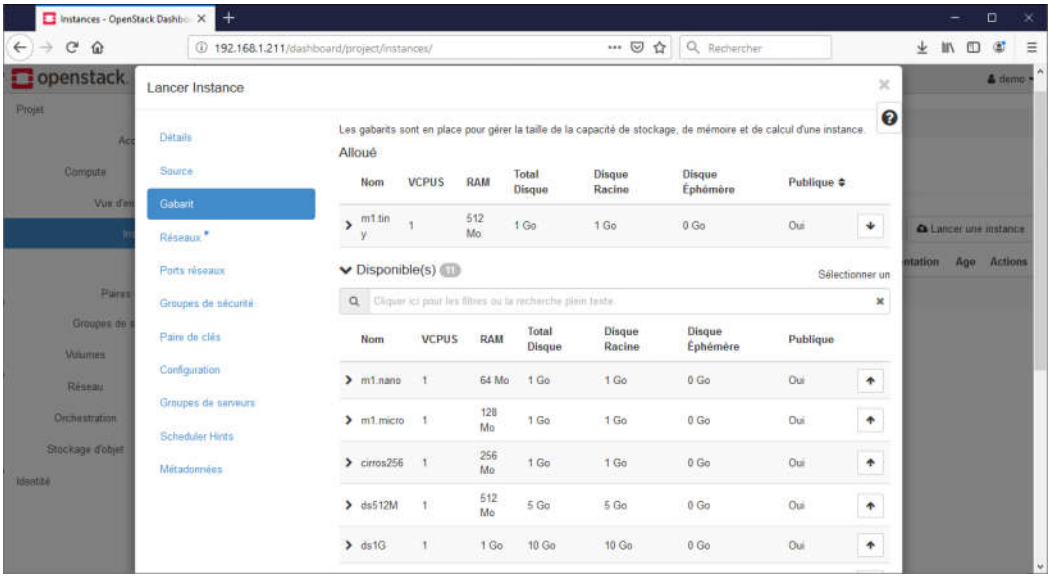
Dans le champ « Sélectionner le source de démarrage » nous choisissons « Volume », dans la zone inférieure de l'onglet la liste des volumes amorçables disponibles s'affiche, nous cliquons sur la flèche à droite du volume « premier_volume » :



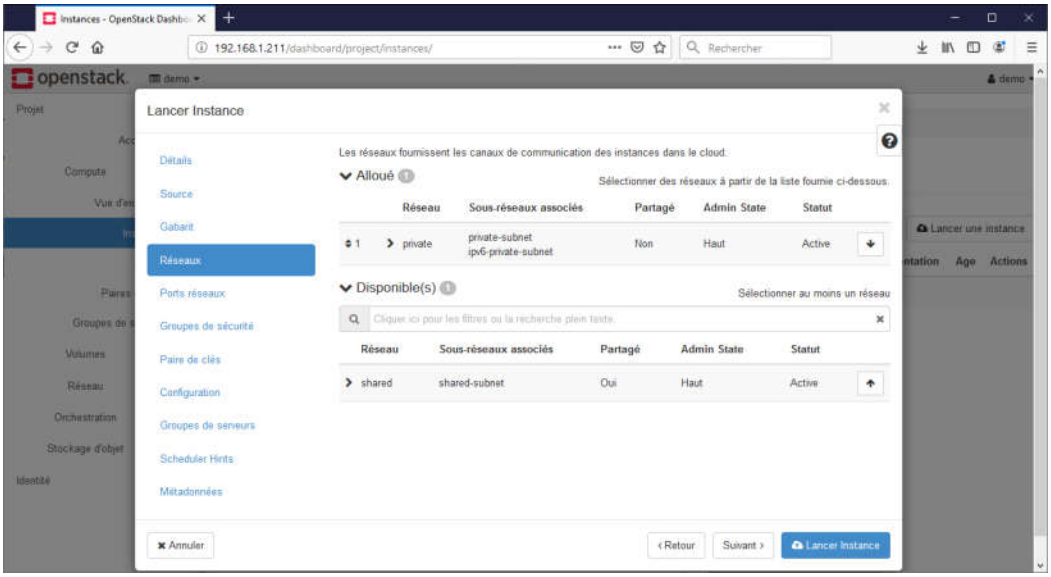
Le volume « premier_volume » passe de la zone « Disponibles » à « Alloué », on clique sur l'onglet « Gabarit » :



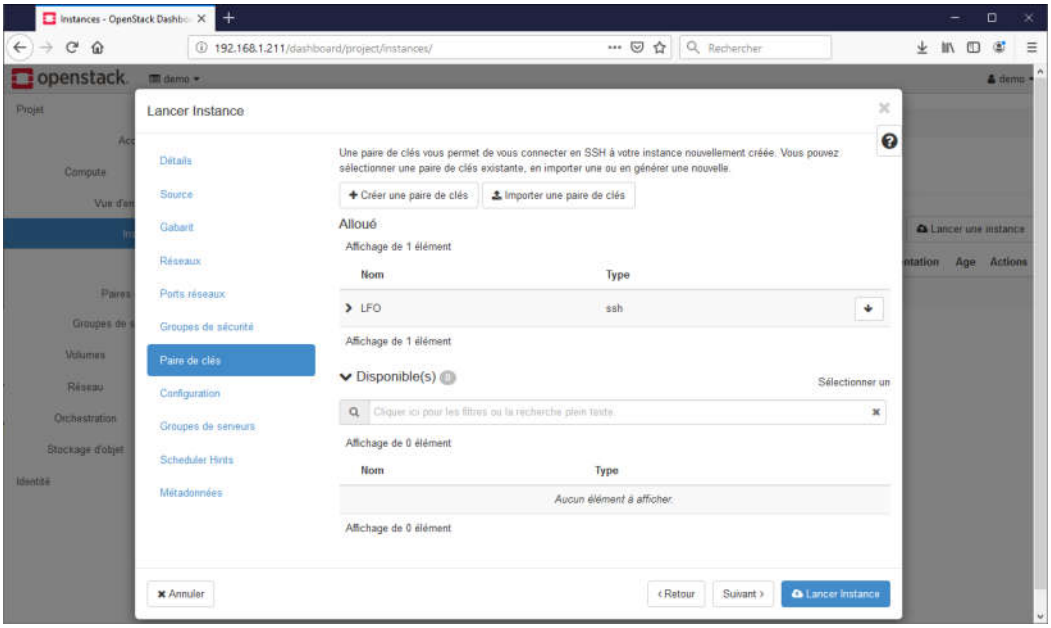
La liste des gabarits s'affiche, on sélectionne le gabarit « m1.tiny » qui est le plus petit gabarit avec 1vCPU et 512Mo de mémoire :



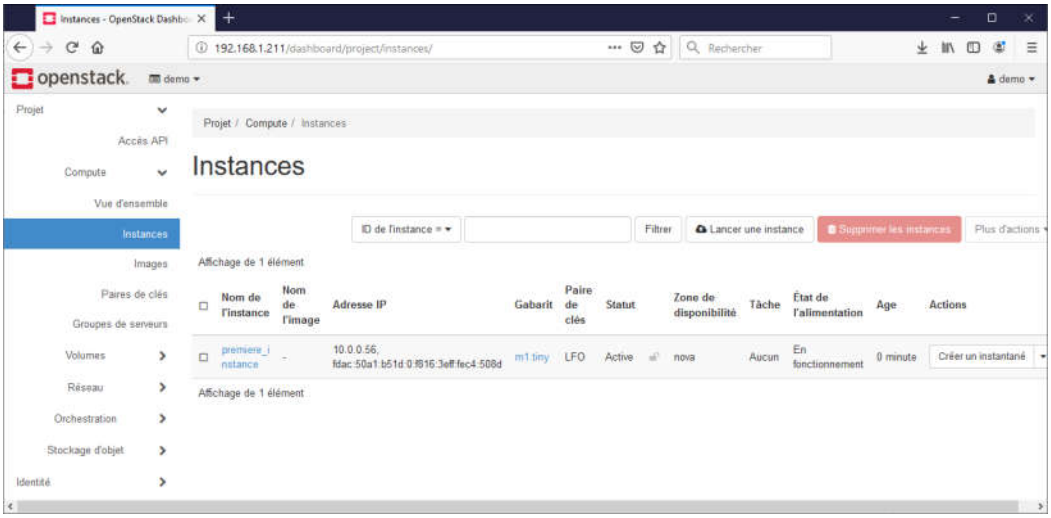
Il faut choisir dans l'onglet 'Réseaux' à quel réseau attacher notre instance, nous choisissons '`private`' :



Si on clique sur l'onglet « Paire de clés » on voit que comme une seule clé est disponible elle a été sélectionnée par défaut (c'est aussi le cas d'autres éléments sur d'autres onglets). Il est déjà possible de lancer l'instance, nous la lançons en cliquant sur 'Lancer Instance' :



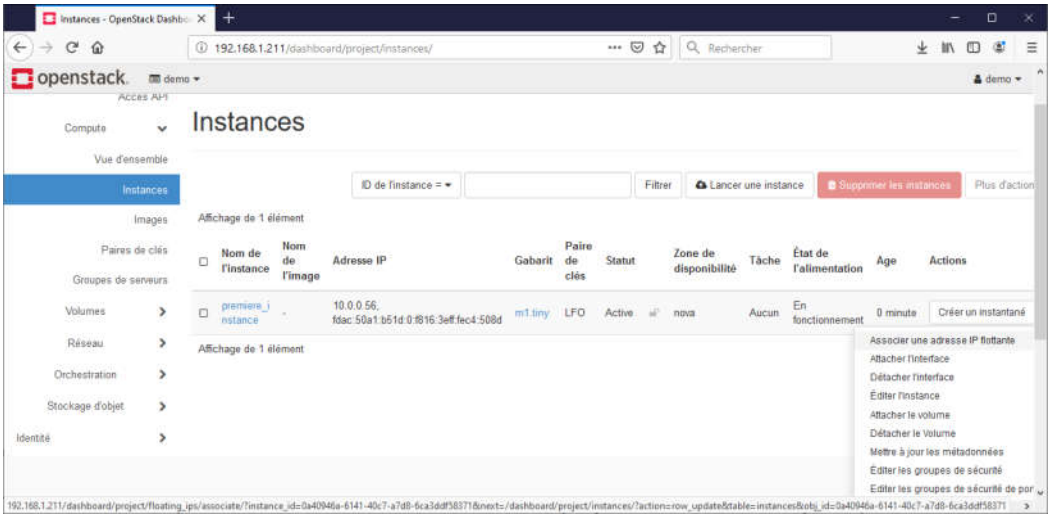
L'instance que nous venons de créer apparaît dans l'onglet « Projet → Compute → Instances », l'instance a une adresse IP qui lui a été allouée, cette adresse IP est une adresse IP privée, elle n'est pas accessible depuis l'extérieur de notre projet OpenStack.



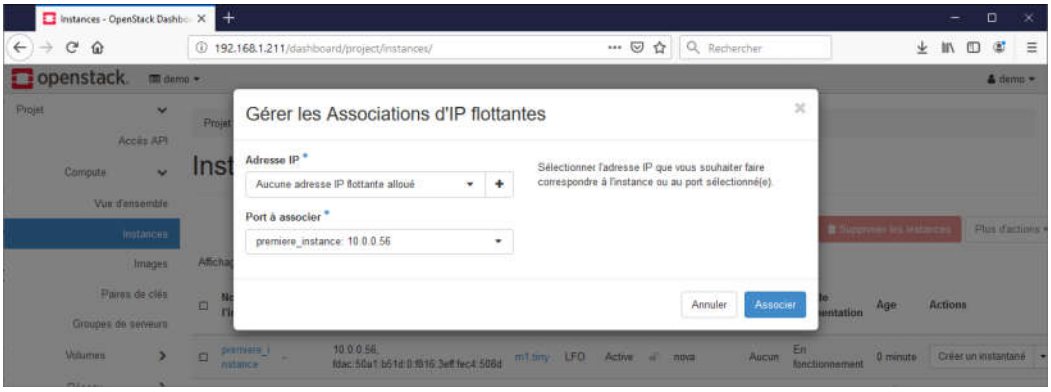
Association d'une adresse IP flottante

Le Cloud OpenStack dispose d'un certain nombre d'adresses IP flottantes, l'administrateur d'un projet peut demander à se faire allouer des adresses IP flottantes (dans la limite des quotas du projet), quand une adresse IP flottante est allouée au projet l'administrateur peut l'associer à une instance.

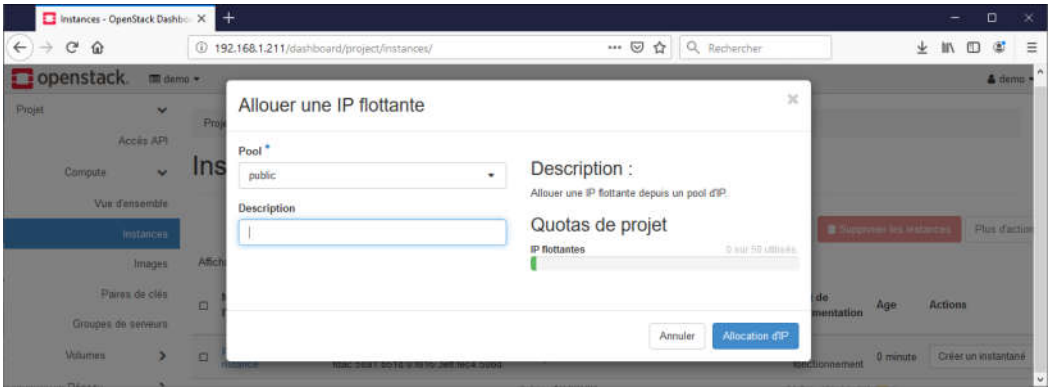
Nous allons attribuer une adresse IP flottante à l'instance afin qu'elle soit accessible en dehors de notre projet. Nous choisissons « Allouer une adresse IP flottante » dans le menu déroulant en face de notre instance :



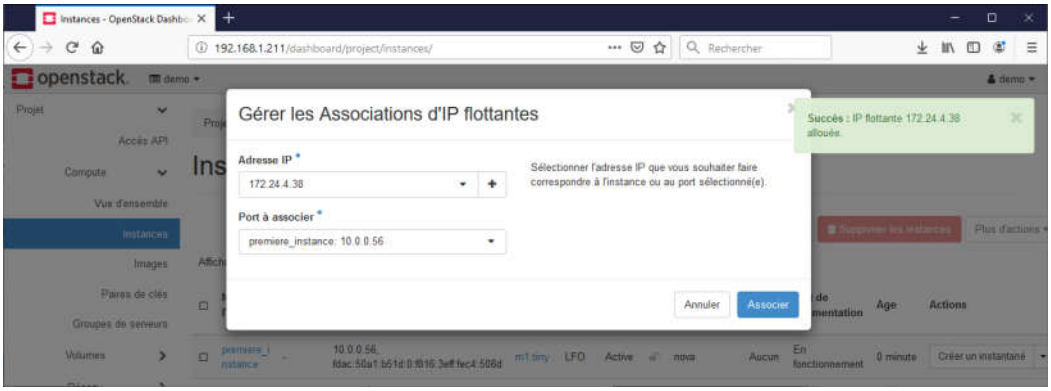
Pour le moment nous n'avons pas d'IP flottante allouée à notre projet. Nous commençons par demander l'allocation d'une IP flottante en cliquant sur le « + » en face de « Aucune IP flottante allouée » :



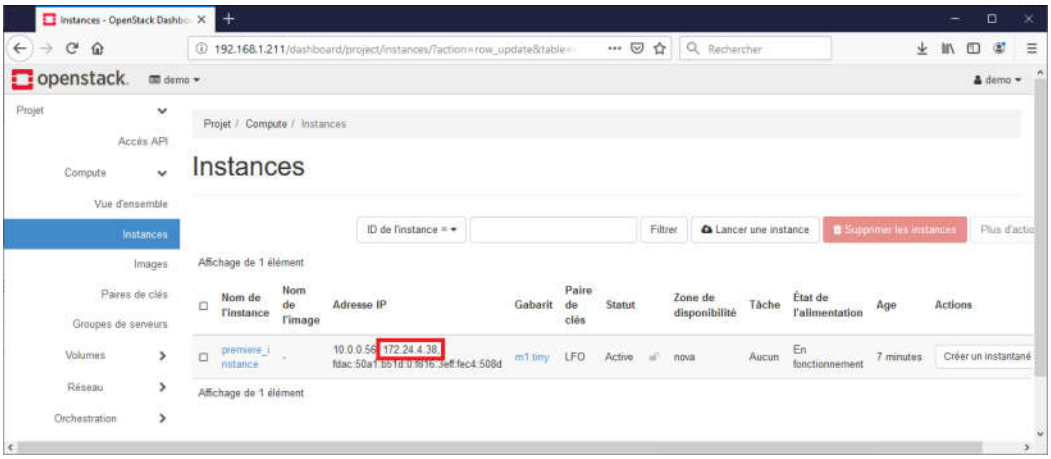
Nous demandons l'allocation d'une adresse IP dans le réseau « public » et nous cliquons sur « Allocation d'IP » :



Une adresse IP flottante nous est allouée, nous pouvons maintenant l'associer à notre instance en cliquant sur « Associer » :



L'adresse IP flottante qui a été associée apparaît maintenant en face de notre instance :



Si après avoir dissocié une adresse IP flottante d'une instance vous la libérez elle sera rendue au pool créé par les administrateurs du système et il n'y a aucun moyen de vous la réattribuer.

Ceci peut être un problème si jamais vous avez référencé cette adresse à l'extérieur (DNS, Firewall, applications tierces..)

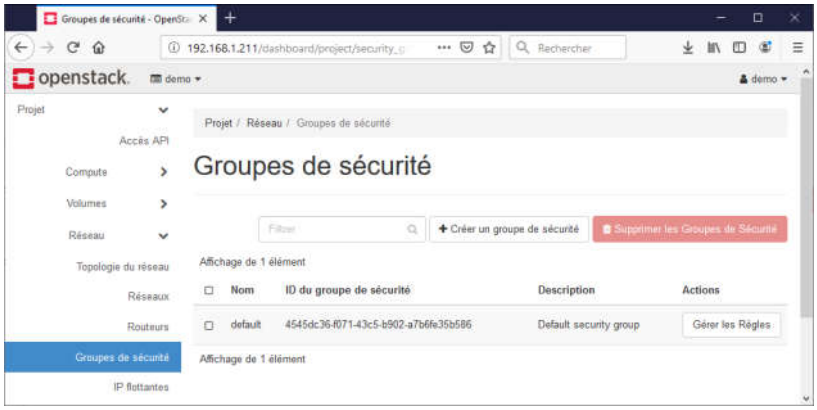
Modification du groupe de sécurité

Les groupes de sécurité sont des pare-feu qui se placent devant les instances pour filtrer les flux réseau entrants et sortants. Le pare-feu par défaut autorise tous les flux sortants de l'instance, et bloque tous les flux entrants.

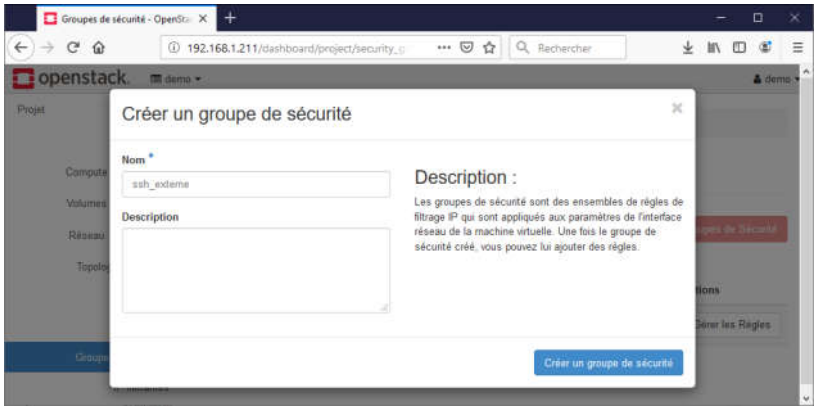
L'administrateur du projet peut créer des groupes de sécurité en fonction de ses besoins (dans la limite des quotas de son projet)

Nous allons modifier le groupe de sécurité par défaut afin d'autoriser les connexions SSH (protocole TCP port 22) vers notre instance.

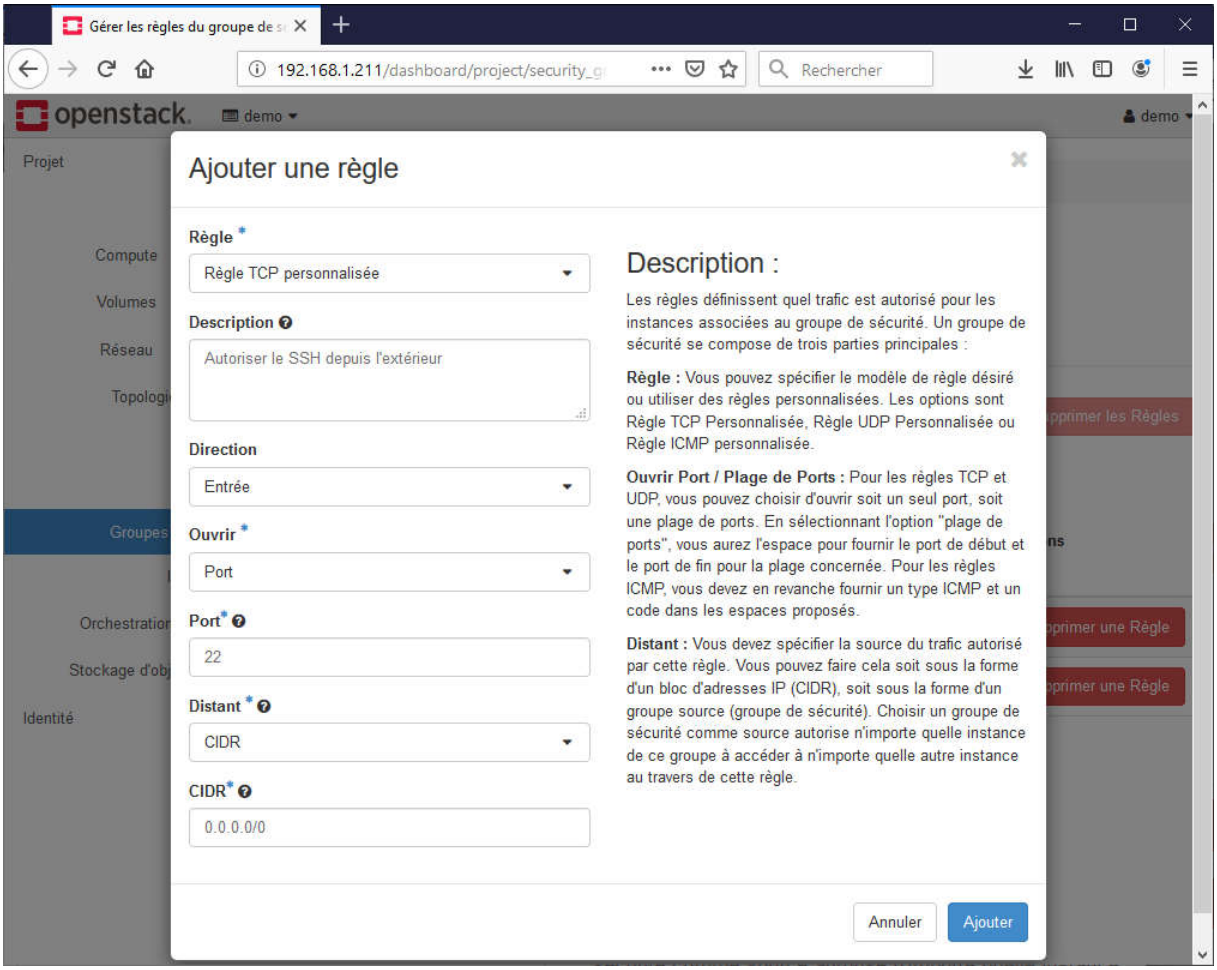
Les groupes de sécurité sont dans l'onglet « Projet → Réseau → Groupes de sécurité », pour le moment nous n'avons que le groupe « default » qu'il ne faut surtout pas modifier :

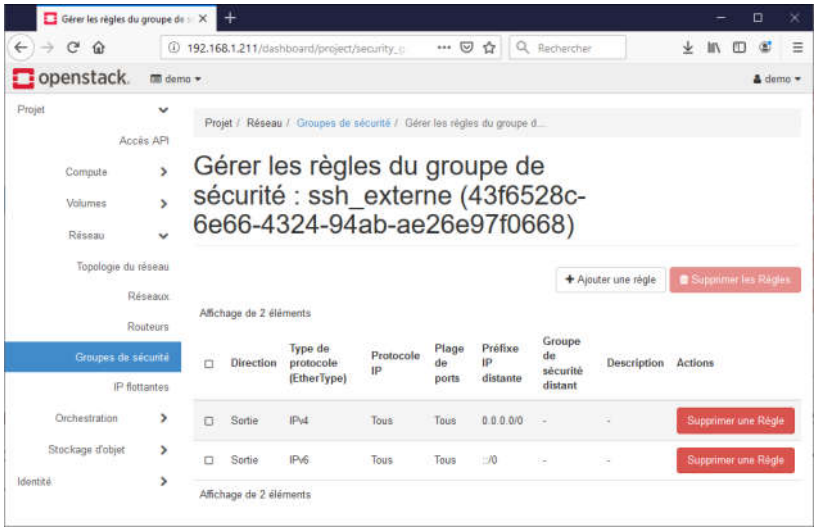


Nous allons créer un nouveau groupe de sécurité en cliquant sur ‘Créer un groupe de sécurité’, nous l’appelons ‘`ssh_externe`’ et nous validons la création en cliquant sur le bouton en bas à droite :

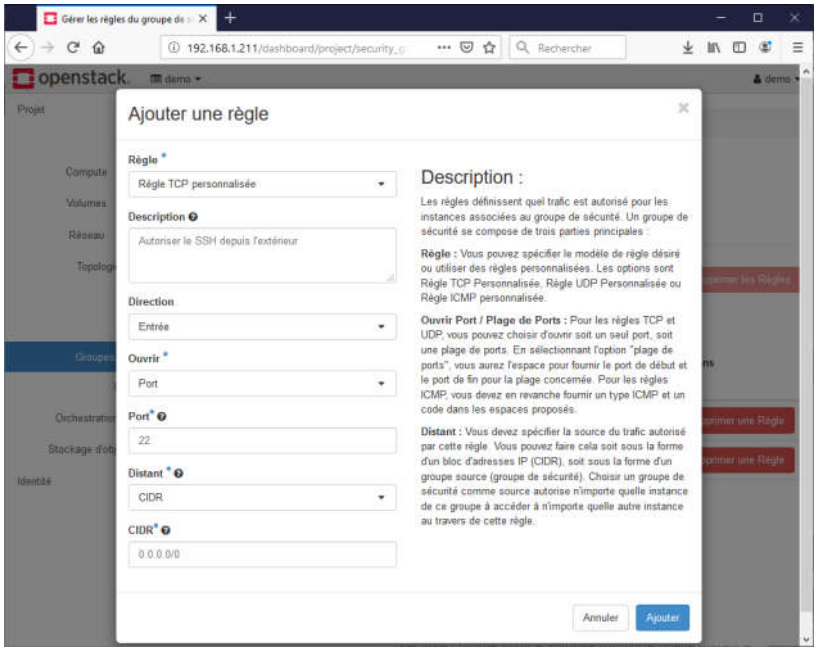


Suite à la création du groupe de sécurité nous arrivons directement sur l’écran de gestion des règles :

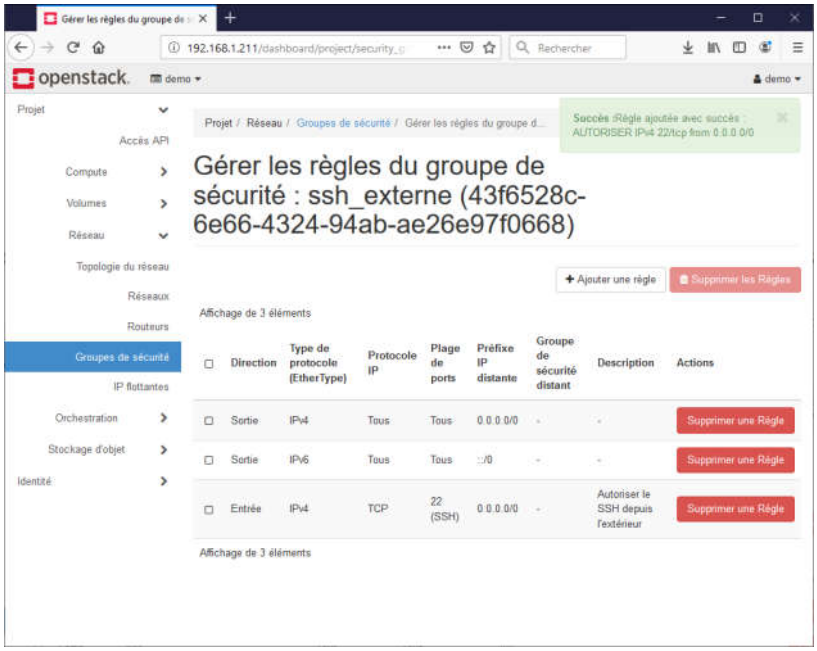




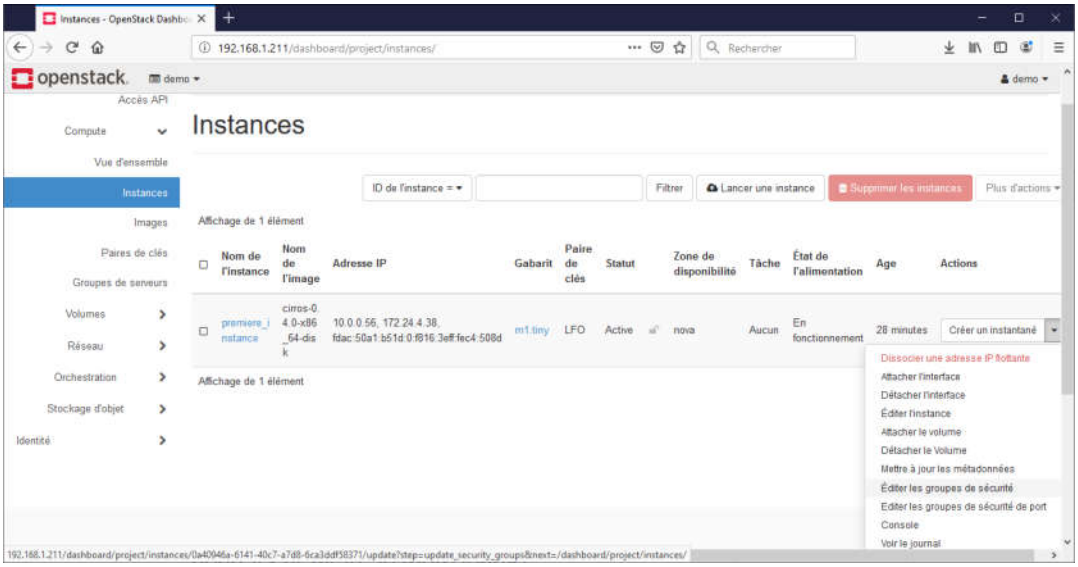
En cliquant sur « Ajouter une règle » nous allons ajouter la règle dont nous avons besoin pour autoriser les connexions SSH, nous créons une « Règle TCP personnalisée , en « Entrée», pour le « Port » numéro « 22 » (SSH), quelle que soit l'origine « CIDR 0.0.0.0/0) :



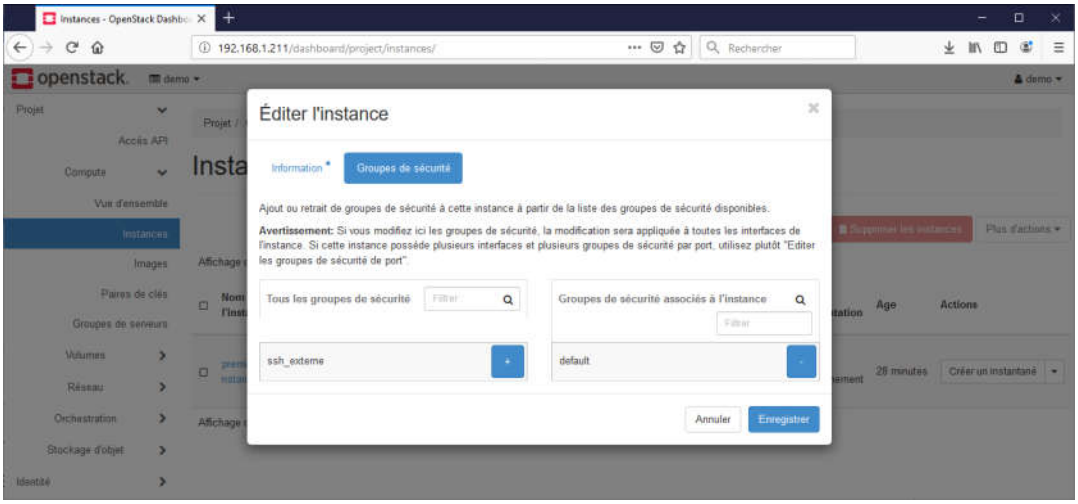
Une fois la modification validée les différentes règles qui composent le groupe de sécurité s'affichent :



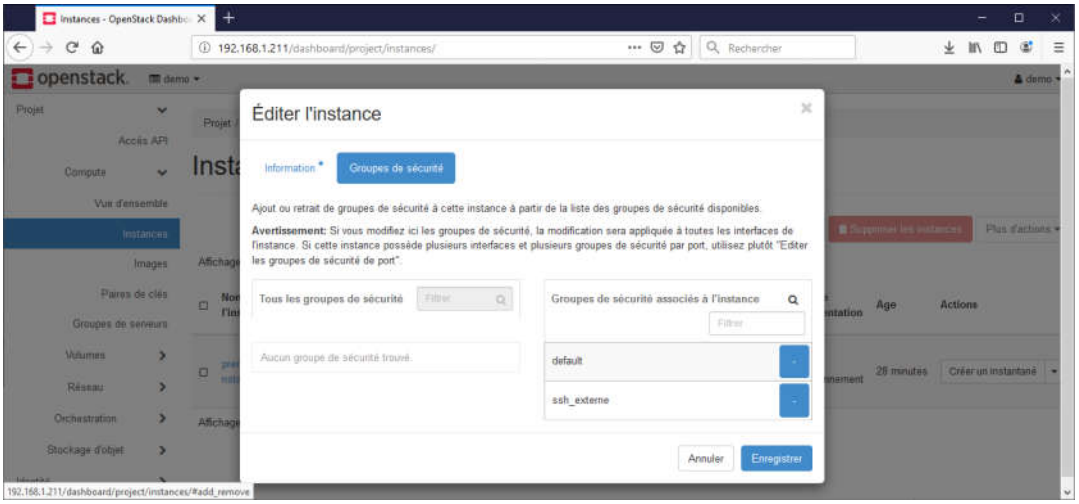
Il faut désormais attribuer le groupe de sécurité que nous venons de créer à notre instance afin que vous puissions nous connecter. Ceci se fait depuis l'écran « Projet → Compute → Instances », en cliquant sur 'éditer les groupes de sécurité' dans le menu déroulant en face de notre instance :



Les groupes de sécurité disponibles mais non associés à l'instance apparaissent dans la colonne de gauche, ceux associés à l'instance dans la colonne de droite. Cliquer sur le '+' ou le '-' en face d'un groupe le fait passer d'une colonne à l'autre :



Nous associons le groupe 'ssh_externe' à l'instance et nous enregistrons:



Connexion à l'instance

Nous pouvons désormais nous connecter en SSH à notre instance depuis notre serveur DevStack en utilisant l'adresse IP flottante de notre instance et le user « cirros », l'adresse IP flottante est affichée en face de l'instance dans le Dashboard :

```
stack@microwave:~$ ssh cirros@172.24.4.38
The authenticity of host '172.24.4.38 (172.24.4.38)' can't be established.
ECDSA key fingerprint is SHA256:Lq/GLmCuTBJrUqJ7GpB/q8sz6VqGfrqQbZ0kIzSpp3o.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.24.4.38' (ECDSA) to the list of known hosts.
$ uname -a
Linux premiere-instance 4.4.0-28-generic #47-Ubuntu SMP Fri Jun 24 10:09:13 UTC 2016 x86_64 GNU/Linux
```

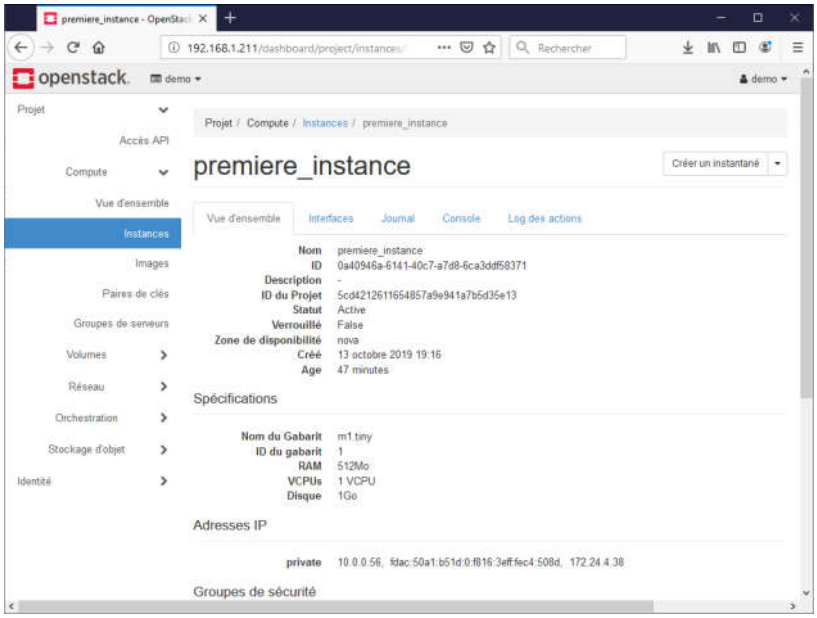
Comme nous avons lancé instance en spécifiant notre clé SSH nous avons pu nous connecter sans avoir à spécifier de mot de passe. Si nous avons un problème avec notre clé SSH nous pouvons utiliser le mot passe « `cubswin:)` » ou « `gocubsgo` » (à partir de cirros v0.4)

Notez que le nom de notre instance est '`premiere_instance`' et que le hostname de l'instance est '`premiere-instance`' car le caractère '_' est interdit dans les entrées DNS (et donc dans les hostnames)

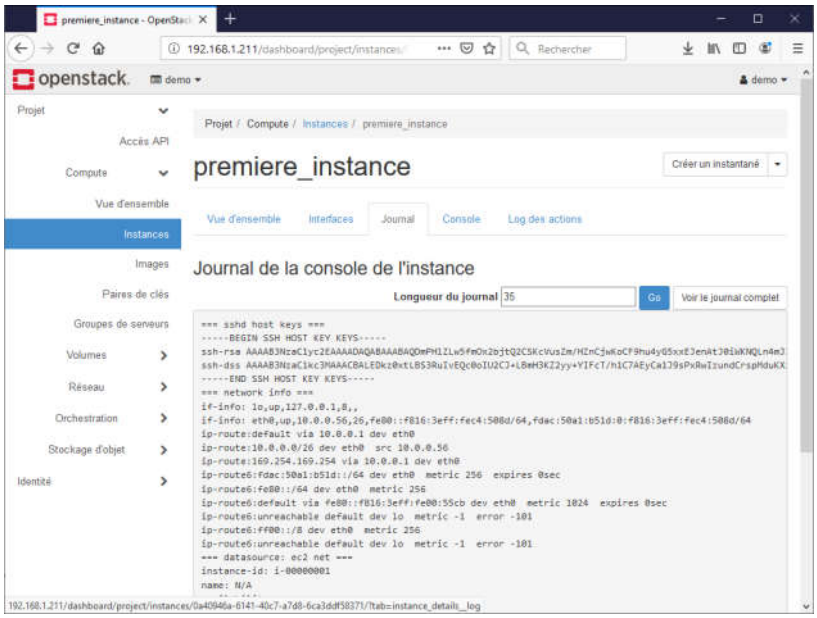
Affichage des informations sur l'instance

Depuis la liste des instances dans le Dashboard (projet → compute → instances) nous pouvons afficher les informations sur une instance, sa log système ainsi que sa console (écran / clavier).

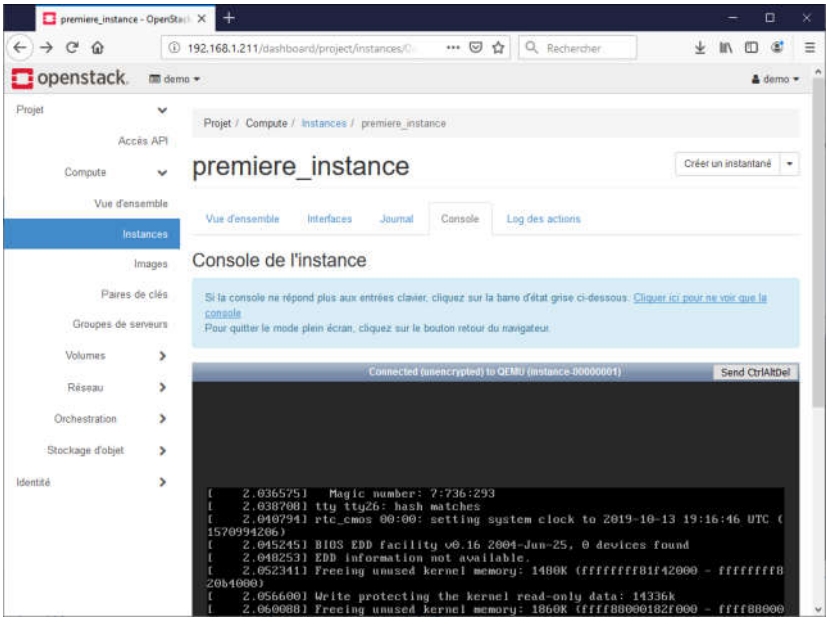
En cliquant sur le nom du serveur, nous affichons 4 onglets (Vue d'ensemble, Journal, Console, Log des actions) :



L'onglet « Journal » donne la log système de l'instance (sur écran cliquer 'Voir le journal complet' permet d'afficher le journal complet et pas seulement les 35 dernières lignes) :



L'onglet « Console » permet d'avoir l'écran et le clavier de l'instance :



Il peut être nécessaire de cliquer sur 'Cliquer ici pour ne voir que la console' afin que la console soit utilisable.

La disposition des touches du clavier peut ne pas être correcte (AZERTY/QWERTY)

Suppression des éléments créés

Depuis le Dashboard nous pouvons supprimer les éléments crées :

- L'instance depuis le menu « Projet → Compute → Instances » en cliquant sur le menu déroulant à droite de l'instance concernée et en sélectionnant « Supprimer l'instance ».
- Le volume associé de la même façon depuis le menu « Projet → Volumes → Volumes »
- Le groupe de sécurité depuis l'écran « Réseau → Groupes de sécurité »
- L'adresse IP flottante depuis l'écran « Réseau → IP flottantes »,choisir « Libérer l'IP flottante » dans le menu déroulant à droite de l'adresse

Utilisation des images

Les éditeurs de systèmes d'exploitation fournissent des images « cloud ready », contrairement aux images classiques qui sont des images de DVD d'installation de l'OS les images « Cloud Ready » sont des images de disques sur lesquels l'OS a été installé.

Téléchargement de l'image Ubuntu « Cloud Image »

Nous allons télécharger l'image cloud de Ubuntu 18.04, les images cloud de Ubuntu 16.04 sont disponibles à l'adresse suivante:

```
https://cloud-images.ubuntu.com/releases/18.04/release/
```

Il y a deux points auxquels il faut faire attention :

- Tout comme pour les serveurs réels, les images cloud sont dépendantes du processeur, nous allons prendre une image ‘amd64’ qui correspond aux architectures Intel/AMD en 64 bits
- Ensuite le format de l’image dépend de l’hyperviseur (le système de cloud) utilisé : HyperV (Microsoft Azure), OVF (VMware), QCOW2 (QEMU et KVM). DevStack est basé sur l’hyperviseur KVM, nous prenons donc l’image QCOW2

Nous allons prendre le fichier `ubuntu-18.04-server-cloudimg-amd64.img` (le libellé ‘Cloud image for 64-bit computers (USB image)’ pour cette image prête à confusion)

```
stack@microwave:~$ wget https://cloud-images.ubuntu.com/releases/bionic/release/ubuntu-18.04-server-cloudimg-amd64.img
--2019-10-13 20:31:01--  https://cloud-images.ubuntu.com/releases/bionic/release/ubuntu-18.04-server-cloudimg-amd64.img
Resolving cloud-images.ubuntu.com (cloud-images.ubuntu.com)... 91.189.88.89, 2001:67c:1560:8001::8001
Connecting to cloud-images.ubuntu.com (cloud-images.ubuntu.com)|91.189.88.89|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 344588288 (329M) [application/octet-stream]
Saving to: 'ubuntu-18.04-server-cloudimg-amd64.img'

ubuntu-18.04-server-cloudimg-amd64.img             100%
[===================================================================================>] 328.62M   827KB/s    in 6m 49s  > ] 312.99M
2019-10-13 20:37:50 (823 KB/s) - 'ubuntu-18.04-server-cloudimg-amd64.img' saved [344588288/344588288]
```

Contrôle de l’intégrité de l’image téléchargée

Une fois l’image téléchargée nous contrôlons la somme de contrôle du fichier téléchargé (les sommes de contrôle sont dans des fichiers `SHA512SUMS` et `MD5SUMS` sur la page de téléchargement).

Plusieurs algorithmes de hashage sont utilisés, nous pouvons utiliser sha512sum pour contrôler l’intégrité du fichier téléchargé, MD5 n’est plus considéré comme un algorithme sûr, mais c’est celui utilisé par Glance pour le contrôle d’intégrité (on retrouve cette somme de contrôle dans le champ `checksum` des commandes `openstack image`).

```
stack@microwave:~$ sha512sum ubuntu-18.04-server-cloudimg-amd64.img
e39393b4b347790b18c…4c0fc4be4cbcf084c56d  ubuntu-18.04-server-cloudimg-amd64.img

stack@microwave:~$ md5sum ubuntu-18.04-server-cloudimg-amd64.img
33e7b1e0d309877a3f6c27bc52d1bd68  ubuntu-18.04-server-cloudimg-amd64.img
```

Chargement de l’image dans Glance

Nous chargeons l’image dans Glance qui est en charge du catalogage et du stockage des images dans OpenStack.

Tout utilisateur d’un projet peut charger une image dans Glance, elle ne sera visible que dans son projet (ce comportement est modifiable par l’administrateur).

L’administrateur du système OpenStack (ou un utilisateur disposant des droits nécessaires) peut charger une image dans Glance et la rendre publique c’est à dire visible depuis tous les projets.

Nous chargeons les variables d’environnement nécessaires pour nous authentifier comme user `admin` dans le projet `admin` et nous chargeons l’image dans Glance en la rendant publique :

```
stack@microwave:~$ source devstack/openrc admin admin
stack@microwave:~$ openstack image create --public \
                                          --disk-format qcow2 \
                                          --container-format bare \
                                          --file ubuntu-18.04-server-cloudimg-amd64.img \
                                          ubuntu-18.04

+------------------+------------------------------------------------------------------------------------------------------+
| Field            | Value                                                                                                |
```

```
+------------------+-------------------------------------------------------------------------------------------------------------+
| checksum         | 33e7b1e0d309877a3f6c27bc52d1bd68                                                                            |
| container_format | bare                                                                                                        |
| created_at       | 2019-10-13T20:48:03Z                                                                                        |
| disk_format      | qcow2                                                                                                       |
| file             | /v2/images/7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5/file                                                        |
| id               | 7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5                                                                        |
| min_disk         | 0                                                                                                           |
| min_ram          | 0                                                                                                           |
| name             | ubuntu-18.04                                                                                                |
| owner            | 0302dd0df6e74588a279d139e452b219                                                                            |
| properties       | os_hash_algo='sha512', os_hash_value='e39393b4b347790b18c…4c0fc4be4cbcf084c56d', os_hidden='False'          |
| protected        | False                                                                                                       |
| schema           | /v2/schemas/image                                                                                           |
| size             | 344588288                                                                                                   |
| status           | active                                                                                                      |
| tags             |                                                                                                             |
| updated_at       | 2019-10-13T20:48:11Z                                                                                        |
| virtual_size     | None                                                                                                        |
| visibility       | public                                                                                                      |
+------------------+-------------------------------------------------------------------------------------------------------------+

stack@microwave:~$ openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| 67d1e1b2-a9f8-451f-afd2-f4697da2b6c7 | cirros-0.4.0-x86_64-disk | active |
| 7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5 | ubuntu-18.04             | active |
+--------------------------------------+--------------------------+--------+
```

Certaines images sont compressées (en `.zip` ou `.xz`) il faut les décompresser avant de les charger dans Glance.

D'une manière générale Glance ne fait aucun contrôle sur l'image : charger une image avec les mauvais paramètres, charger une image compressée ou d'un format incorrect produira des erreurs inattendues au moment du lancement d'une instance utilisant cette image.

Vérification de la disponibilité de l'image

Nous vérifions que l'image est disponible dans le projet `demo`, on note que l'utilisateur `demo` n'a pas les autorisations nécessaires pour supprimer cette image car il s'agit d'une image publique :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| 67d1e1b2-a9f8-451f-afd2-f4697da2b6c7 | cirros-0.4.0-x86_64-disk | active |
| 7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5 | ubuntu-18.04             | active |
+--------------------------------------+--------------------------+--------+
stack@microwave:~$ openstack image delete ubuntu-18.04
Failed to delete image with name or ID 'ubuntu-18.04': HTTP 403 Forbidden: You are not permitted to delete this image.
Failed to delete 1 of 1 images.
```

Lancement d'une instance en ligne de commandes

Nous pouvons lancer une instance grâce à la commande `openstack` de la même manière que nous l'avons fait en utilisant précédemment en utilisant le Dashboard.

Nous commençons par créer le volume système que nous allons appeler `ubuntu-18.04_vol` à partir de l'image `ubuntu-18.04`, cette opération peut prendre une ou plusieurs minutes, et l'image passe par plusieurs états avant d'être disponible.

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack image list
+--------------------------------------+--------------------------+--------+
| ID                                   | Name                     | Status |
+--------------------------------------+--------------------------+--------+
| 67d1e1b2-a9f8-451f-afd2-f4697da2b6c7 | cirros-0.4.0-x86_64-disk | active |
```

```
| 7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5 | ubuntu-18.04             | active |
+--------------------------------------+--------------------------+--------+
stack@microwave:~$ openstack volume create --size 10 --image ubuntu-18.04  ubuntu-18.04_vol
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2019-10-13T21:02:45.000000           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | edc26441-2f67-410d-8b24-a1e85844d8dc |
| multiattach         | False                                |
| name                | ubuntu-18.04_vol                     |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 10                                   |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | lvmdriver-1                          |
| updated_at          | None                                 |
| user_id             | 57d02b0938d149a1aac96ce6757a3a14     |
+---------------------+--------------------------------------+
stack@microwave:~$ openstack volume show ubuntu-18.04_vol
+------------------------------+--------------------------------------+
| Field                        | Value                                |
+------------------------------+--------------------------------------+
| attachments                  | []                                   |
| availability_zone            | nova                                 |
| bootable                     | false                                |
| consistencygroup_id          | None                                 |
| created_at                   | 2019-10-13T21:02:45.000000           |
| description                  | None                                 |
| encrypted                    | False                                |
| id                           | edc26441-2f67-410d-8b24-a1e85844d8dc |
| multiattach                  | False                                |
| name                         | ubuntu-18.04_vol                     |
| os-vol-tenant-attr:tenant_id | 5cd4212611654857a9e941a7b5d35e13     |
| properties                   |                                      |
| replication_status           | None                                 |
| size                         | 3                                    |
| snapshot_id                  | None                                 |
| source_volid                 | None                                 |
| status                       | downloading                          |
| type                         | lvmdriver-1                          |
| updated_at                   | 2019-10-13T21:03:21.000000           |
| user_id                      | 57d02b0938d149a1aac96ce6757a3a14     |
| volume_image_metadata        | {u'signature_verified': u'False'}    |
+------------------------------+--------------------------------------+
stack@microwave:~$ openstack volume show ubuntu-18.04_vol
+------------------------------+-----------------------------------------------------------------------------+
| Field                        | Value                                                                       |
+------------------------------+-----------------------------------------------------------------------------+
| attachments                  | []                                                                          |
| availability_zone            | nova                                                                        |
| bootable                     | true                                                                        |
| consistencygroup_id          | None                                                                        |
| created_at                   | 2019-10-13T21:02:45.000000                                                  |
| description                  | None                                                                        |
| encrypted                    | False                                                                       |
| id                           | edc26441-2f67-410d-8b24-a1e85844d8dc                                        |
| multiattach                  | False                                                                       |
| name                         | ubuntu-18.04_vol                                                            |
| os-vol-tenant-attr:tenant_id | 5cd4212611654857a9e941a7b5d35e13                                            |
| properties                   |                                                                             |
| replication_status           | None                                                                        |
| size                         | 10                                                                          |
| snapshot_id                  | None                                                                        |
| source_volid                 | None                                                                        |
| status                       | available                                                                   |
| type                         | lvmdriver-1                                                                 |
| updated_at                   | 2019-10-13T21:04:42.000000                                                  |
| user_id                      | 57d02b0938d149a1aac96ce6757a3a14                                            |
| volume_image_metadata        | {u'checksum': u'33e7b1e0d309877a3f6c27bc52d1bd68', u'min_ram': u'0',        |
|                              |  u'disk_format': u'qcow2', u'image_name': u'ubuntu-18.04',                  |
|                              |  u'image_id': u'7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5',                      |
|                              |  u'signature_verified': u'False', u'container_format': u'bare',             |
|                              |  u'min_disk': u'0', u'size': u'344588288'}                                  |
+------------------------------+-----------------------------------------------------------------------------+
```

Création d'un groupe de sécurité

Nous allons créer notre propre « security group » qui autorise tout le trafic en sortie (c'est le comportement par défaut) et autorise le port 22 (SSH) en entrée.

```
stack@microwave:~$ openstack security group create bastion_ssh
+-----------------+----------------------------------------+
| Field           | Value                                  |
+-----------------+----------------------------------------+
| created_at      | 2019-10-13T21:12:20Z                   |
| description     | bastion_ssh                            |
| id              | 9116395f-92dc-47c8-bd38-9424f6b7f88f   |
| name            | bastion_ssh                            |
| project_id      | 5cd4212611654857a9e941a7b5d35e13       |
| revision_number | 1                                      |
| rules           | direction='egress', ethertype='IPv6'   |
|                 | direction='egress', ethertype='IPv4'   |
| tags            | []                                     |
| updated_at      | 2019-10-13T21:12:20Z                   |
+-----------------+----------------------------------------+
stack@microwave:~$ openstack security group rule create --ingress --remote-ip 0.0.0.0/0 \
                                                        --protocol tcp --dst-port 22 bastion_ssh
+-------------------+----------------------------------------+
| Field             | Value                                  |
+-------------------+----------------------------------------+
| created_at        | 2019-10-13T21:12:34Z                   |
| description       |                                        |
| direction         | ingress                                |
| ether_type        | IPv4                                   |
| id                | 08a8650e-1b28-4f8e-beb7-2c21bbff7b99   |
| name              | None                                   |
| port_range_max    | 22                                     |
| port_range_min    | 22                                     |
| project_id        | 5cd4212611654857a9e941a7b5d35e13       |
| protocol          | tcp                                    |
| remote_group_id   | None                                   |
| remote_ip_prefix  | 0.0.0.0/0                              |
| revision_number   | 0                                      |
| security_group_id | 9116395f-92dc-47c8-bd38-9424f6b7f88f   |
| tags              | []                                     |
| updated_at        | 2019-10-13T21:12:34Z                   |
+-------------------+----------------------------------------+
stack@microwave:~$ openstack security group show bastion_ssh
+-----------------+-------------------------------------------------------------+
| Field           | Value                                                       |
+-----------------+-------------------------------------------------------------+
| created_at      | 2019-10-13T21:12:20Z                                        |
| description     | bastion_ssh                                                 |
| id              | 9116395f-92dc-47c8-bd38-9424f6b7f88f                        |
| name            | bastion_ssh                                                 |
| project_id      | 5cd4212611654857a9e941a7b5d35e13                            |
| revision_number | 2                                                           |
| rules           | direction='ingress', ethertype='IPv4',                      |
|                 |   port_range_max='22', port_range_min='22', protocol='tcp', |
|                 |   remote_ip_prefix='0.0.0.0/0'                              |
|                 | direction='egress',  ethertype='IPv6'                       |
|                 | direction='egress',  ethertype='IPv4'                       |
| tags            | []                                                          |
| updated_at      | 2019-10-13T21:12:34Z                                        |
+-----------------+-------------------------------------------------------------+
```

Lancement de l'instance

Pour lancer notre instance nous avons besoin des éléments suivantes :

- Le volume système que nous venons de créer
- Un gabarit (flavor), par exemple m1.small
- Une clé publique pour pouvoir nous connecter en SSH à l'instance, elle est déjà disponible
- Le groupe de sécurité qui autorise tout le trafic en sortie et le port 22 en entrée
- Le réseau privé auquel attacher l'instance

```
stack@microwave:~$ openstack volume list
+--------------------------------------+-----------------+-----------+------+-------------+
| ID                                   | Name            | Status    | Size | Attached to |
+--------------------------------------+-----------------+-----------+------+-------------+
| edc26441-2f67-410d-8b24-a1e85844d8dc | ubuntu-18.04_vol | available |   10 |             |
+--------------------------------------+-----------------+-----------+------+-------------+
stack@microwave:~$ openstack security group list
+--------------------------------------+-------------+------------------------+----------------------------------+------+
| ID                                   | Name        | Description            | Project                          | Tags |
+--------------------------------------+-------------+------------------------+----------------------------------+------+
| 4545dc36-f071-43c5-b902-a7b6fe35b586 | default     | Default security group | 5cd4212611654857a9e941a7b5d35e13 | []   |
| 9116395f-92dc-47c8-bd38-9424f6b7f88f | bastion_ssh | bastion_ssh            | 5cd4212611654857a9e941a7b5d35e13 | []   |
+--------------------------------------+-------------+------------------------+----------------------------------+------+
```

```
stack@microwave:~$ openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 42 | m1.nano   |    64 |    1 |         0 |     1 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
| 84 | m1.micro  |   128 |    1 |         0 |     1 | True      |
| c1 | cirros256 |   256 |    1 |         0 |     1 | True      |
| d1 | ds512M    |   512 |    5 |         0 |     1 | True      |
| d2 | ds1G      |  1024 |   10 |         0 |     1 | True      |
| d3 | ds2G      |  2048 |   10 |         0 |     2 | True      |
| d4 | ds4G      |  4096 |   20 |         0 |     4 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
stack@microwave:~$ openstack keypair list
+------+-------------------------------------------------+
| Name | Fingerprint                                     |
+------+-------------------------------------------------+
| LFO  | 8c:56:76:56:d9:de:50:81:8b:4d:6c:b1:ee:55:b2:ae |
+------+-------------------------------------------------+
stack@microwave:~$ openstack network list
+--------------------------------------+---------+--------------------------------------+
| ID                                   | Name    | Subnets                              |
+--------------------------------------+---------+--------------------------------------+
| 0e6e2f0c-c4cc-4eca-8428-8117864b6e4a | private | 10b8dba4-4613-4462-b8f4-500bd6c7fc84 |
| 12ceb0b7-9ed6-4bfd-9160-ab44b4f868b2 | public  | b3ee49a7-80b9-4898-8f53-a58cb1e3908c |
| 38f2e146-43e0-4f93-b3db-87fc25a4be19 | shared  | 29ea72f8-7ac3-4d2a-a2f8-e4b2f2946718 |
+--------------------------------------+---------+--------------------------------------+
stack@microwave:~$ openstack server create --volume ubuntu-18.04_vol        \
                                           --security-group bastion_ssh     \
                                           --flavor m1.small                \
                                           --key-name "LFO"                 \
                                           --network private                \
                                             ubuntu-18.04_inst
+-----------------------------+------------------------------------------------+
| Field                       | Value                                          |
+-----------------------------+------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                         |
| OS-EXT-AZ:availability_zone |                                                |
| OS-EXT-STS:power_state      | NOSTATE                                        |
| OS-EXT-STS:task_state       | scheduling                                     |
| OS-EXT-STS:vm_state         | building                                       |
| OS-SRV-USG:launched_at      | None                                           |
| OS-SRV-USG:terminated_at    | None                                           |
| accessIPv4                  |                                                |
| accessIPv6                  |                                                |
| addresses                   |                                                |
| adminPass                   | YdPdfs86o22C                                   |
| config_drive                |                                                |
| created                     | 2019-10-13T21:23:44Z                           |
| flavor                      | m1.small (2)                                   |
| hostId                      |                                                |
| id                          | 01e719a0-abd4-44dd-99e1-27076de3dce5           |
| image                       |                                                |
| key_name                    | LFO                                            |
| name                        | ubuntu-18.04_inst                              |
| progress                    | 0                                              |
| project_id                  | 5cd4212611654857a9e941a7b5d35e13               |
| properties                  |                                                |
| security_groups             | name='9116395f-92dc-47c8-bd38-9424f6b7f88f'    |
| status                      | BUILD                                          |
| updated                     | 2019-10-13T21:23:44Z                           |
| user_id                     | 57d02b0938d149a1aac96ce6757a3a14               |
| volumes_attached            |                                                |
+-----------------------------+------------------------------------------------+

stack@microwave:~$ openstack server show ubuntu-18.04_inst
+-----------------------------+------------------------------------------------------------+
| Field                       | Value                                                      |
+-----------------------------+------------------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                                     |
| OS-EXT-AZ:availability_zone | nova                                                       |
| OS-EXT-STS:power_state      | Running                                                    |
| OS-EXT-STS:task_state       | None                                                       |
| OS-EXT-STS:vm_state         | active                                                     |
| OS-SRV-USG:launched_at      | 2019-10-13T21:23:57.000000                                 |
| OS-SRV-USG:terminated_at    | None                                                       |
| accessIPv4                  |                                                            |
| accessIPv6                  |                                                            |
| addresses                   | private=fdac:50a1:b51d:0:f816:3eff:fec8:2ed5, 10.0.0.43    |
| config_drive                |                                                            |
| created                     | 2019-10-13T21:23:43Z                                       |
| flavor                      | m1.small (2)                                               |
| hostId                      | b3ce8aa16af22651ac4ab1291280fd727ff9c016c9068dcfcc861a75   |
| id                          | 01e719a0-abd4-44dd-99e1-27076de3dce5                       |
| image                       |                                                            |
| key_name                    | LFO                                                        |
| name                        | ubuntu-18.04_inst                                          |
```

```
| progress                    | 0                                                                |
| project_id                  | 5cd4212611654857a9e941a7b5d35e13                                 |
| properties                  |                                                                  |
| security_groups             | name='bastion_ssh'                                               |
| status                      | ACTIVE                                                           |
| updated                     | 2019-10-13T21:23:58Z                                             |
| user_id                     | 57d02b0938d149a1aac96ce6757a3a14                                 |
| volumes_attached            | id='edc26441-2f67-410d-8b24-a1e85844d8dc'                        |
+-----------------------------+------------------------------------------------------------------+
```

Le lancement d'une instance tout comme la création d'un volume n'est pas immédiat, la commande de création est lancée puis exécutée de façon asynchrone par les différents composants de Openstack. La commande `openstack server show` permet de savoir quand l'instance est fonctionnelle (« `running` »)

Ajout d'une adresse IP flottante à notre instance

Par défaut dans notre projet nous avons au minimum deux réseaux, un réseau public (externe à notre projet, il appartient au système OpenStack) et un réseau privé (interne à notre projet) :

```
stack@microwave:~$ openstack network list
+--------------------------------------+---------+--------------------------------------+
| ID                                   | Name    | Subnets                              |
+--------------------------------------+---------+--------------------------------------+
| 0e6e2f0c-c4cc-4eca-8428-8117864b6e4a | private | 10b8dba4-4613-4462-b8f4-500bd6c7fc84 |
| 12ceb0b7-9ed6-4bfd-9160-ab44b4f868b2 | public  | b3ee49a7-80b9-4898-8f53-a58cb1e3908c |
| 38f2e146-43e0-4f93-b3db-87fc25a4be19 | shared  | 29ea72f8-7ac3-4d2a-a2f8-e4b2f2946718 |
+--------------------------------------+---------+--------------------------------------+
```

Pour le moment notre instance a une adresse IP dans le réseau « private » :

```
stack@microwave:~$ openstack server show ubuntu-18.04_inst
+-----------------------------+------------------------------------------------------------+
| Field                       | Value                                                      |
+-----------------------------+------------------------------------------------------------+
| OS-DCF:diskConfig           | MANUAL                                                     |
| OS-EXT-AZ:availability_zone | nova                                                       |
| OS-EXT-STS:power_state      | Running                                                    |
| OS-EXT-STS:task_state       | None                                                       |
                              …/…
| id                          | 01e719a0-abd4-44dd-99e1-27076de3dce5                       |
| name                        | ubuntu-18.04_inst                                          |
                              …/…
| addresses                   | private=fdac:50a1:b51d:0:f816:3eff:fec8:2ed5, 10.0.0.43    |
                              …/…
| volumes_attached            | id='edc26441-2f67-410d-8b24-a1e85844d8dc'                  |
+-----------------------------+------------------------------------------------------------+
```

Nous créons une adresse IP flottante pour notre serveur, nous la créons dans le réseau « public » :

```
stack@microwave:~$ openstack floating ip create public
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2019-10-14T18:42:24Z                 |
| description         |                                      |
| dns_domain          | None                                 |
| dns_name            | None                                 |
| fixed_ip_address    | None                                 |
| floating_ip_address | 172.24.4.105                         |
| floating_network_id | 12ceb0b7-9ed6-4bfd-9160-ab44b4f868b2 |
| id                  | 168bd864-8489-4d50-96dd-24bd93f316e4 |
| name                | 172.24.4.105                         |
| port_details        | None                                 |
| port_id             | None                                 |
| project_id          | 5cd4212611654857a9e941a7b5d35e13     |
| qos_policy_id       | None                                 |
| revision_number     | 0                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| subnet_id           | None                                 |
| tags                | []                                   |
| updated_at          | 2019-10-14T18:42:24Z                 |
+---------------------+--------------------------------------+
stack@microwave:~$ openstack floating ip list
+--------------------------------------+---------------------+------------------+------+
| ID                                   | Floating IP Address | Fixed IP Address | Port |
+--------------------------------------+---------------------+------------------+------+
| 168bd864-8489-4d50-96dd-24bd93f316e4 | 172.24.4.105        | None             | None |
```

```
+--------------------------------------+---------------------+------------------+------+
```

Nous attachons maintenant l'adresse IP flottante à notre instance :

```
stack@microwave:~$ openstack server add floating ip ubuntu-18.04_inst 172.24.4.105
stack@microwave:~$ openstack floating ip list
+--------------------------------------+---------------------+------------------+--------------------------------------+
| ID                                   | Floating IP Address | Fixed IP Address | Port                                 |
+--------------------------------------+---------------------+------------------+--------------------------------------+
| 168bd864-8489-4d50-96dd-24bd93f316e4 | 172.24.4.105        | 10.0.0.43        | 3db24397-cd2e-4cba-853a-e48ebcbd71e2 |
+--------------------------------------+---------------------+------------------+--------------------------------------+
stack@microwave:~$ openstack server list
+--------------------------------------+------------------+--------+----------------------------------+-------+----------+
| ID                                   | Name             | Status | Networks                         | Image | Flavor   |
+--------------------------------------+------------------+--------+----------------------------------+-------+----------+
| 01e719a0-abd4-44dd-99e1-27076de3dce5 | ubuntu-18.04_inst | ACTIVE | private=10.0.0.43, 172.24.4.105 |       | m1.small |
+--------------------------------------+------------------+--------+----------------------------------+-------+----------+
```

Connexion à notre instance

Nous pouvons désormais nous connecter à notre instance :

```
stack@microwave:~$ ssh ubuntu@172.24.4.105
The authenticity of host '172.24.4.105 (172.24.4.105)' can't be established.
ECDSA key fingerprint is SHA256:i8oP2aoZrUSWHrpZ3mZBozhD4pG4GL9CCqizOMgHrGg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.24.4.105' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)
                                        …/…
ubuntu@ubuntu-18:~$ uname -a
Linux ubuntu-18 4.15.0-65-generic #74-Ubuntu SMP Tue Sep 17 17:06:04 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

Les volumes de données

Nous l'avons vu, le contenu du volume système, c'est à dire le système d'exploitation et les logiciels associés, sont « jetables », seules les données de l'application (bases de données, fichiers…) sont à conserver de façon perenne.

Le volume système est celui sur lequel a été démarré l'instance, nous l'avons créé à partir d'une image système mise à notre disposition, les logiciels systèmes peuvent être installés et paramétrés de façon automatique, nous verrons plus loin un exemple.

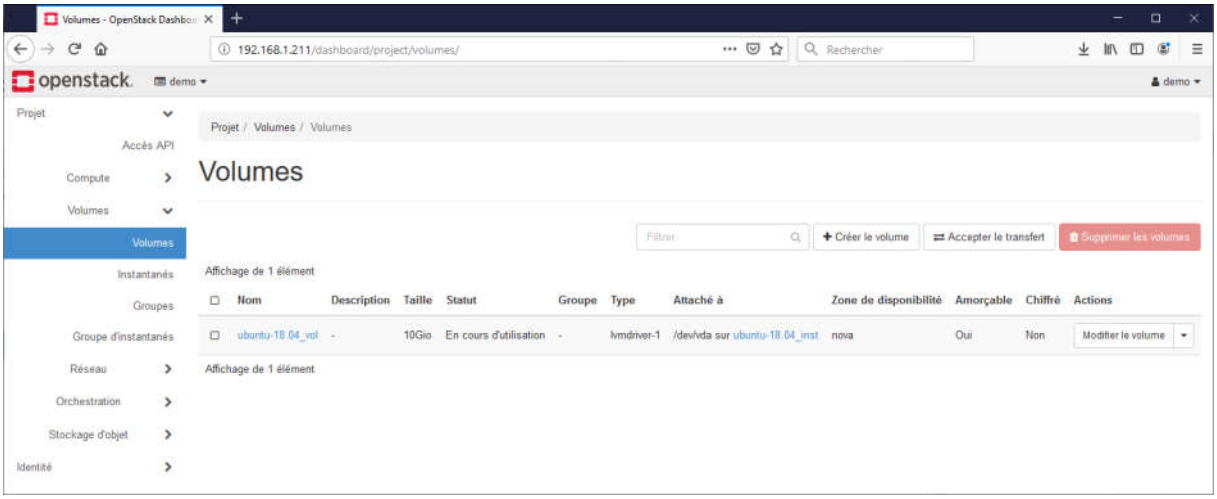
Nous créons un volume pour stocker des données, ce volume est créé de façon similaire au volume système (sauf qu'il est créé vide, et non à partir d'une image). Ce volume est ensuite attaché à une instance.

Création d'un volume de données via le Dashboard

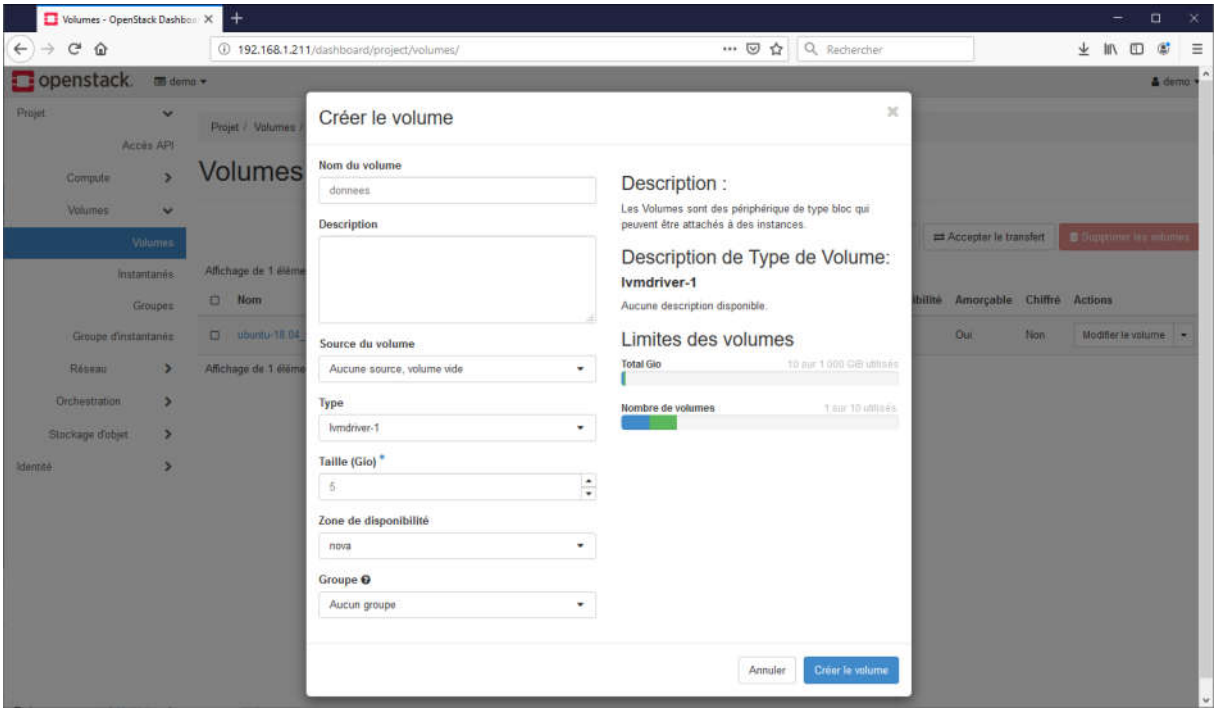
Les volumes tout comme les instances appartiennent à un projet, il convient donc de se connecter avec le bon utilisateur et de se placer dans le bon projet.

Dans le menu « Projet → Volumes → Volumes » nous voyons le volume système de notre instance, son statut est « En cours d'utilisation », il est attaché à notre instance sur le 'device' `/dev/vda`

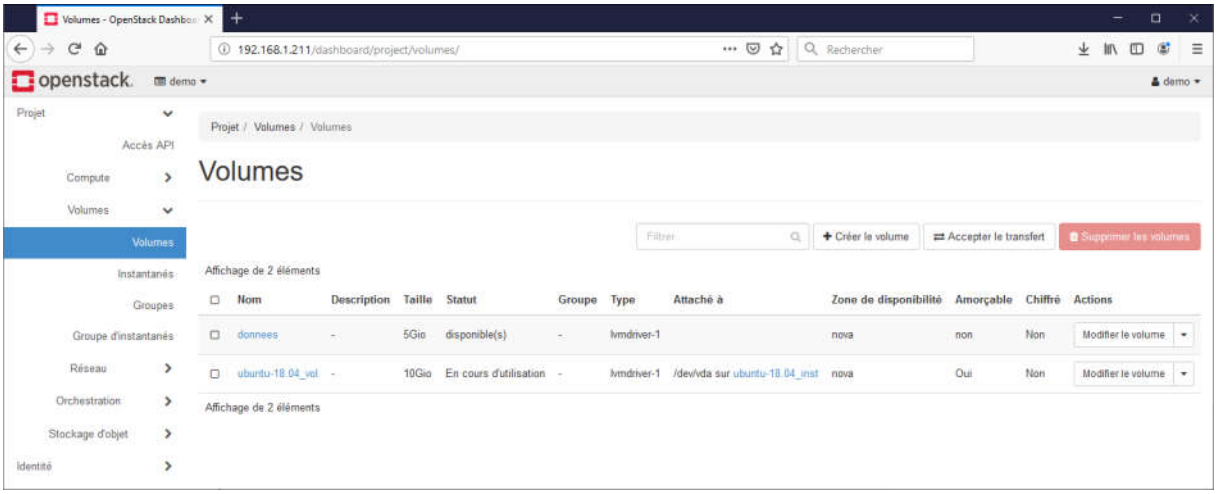
Nous cliquons sur « Créer le volume » pour créer un nouveau volume :



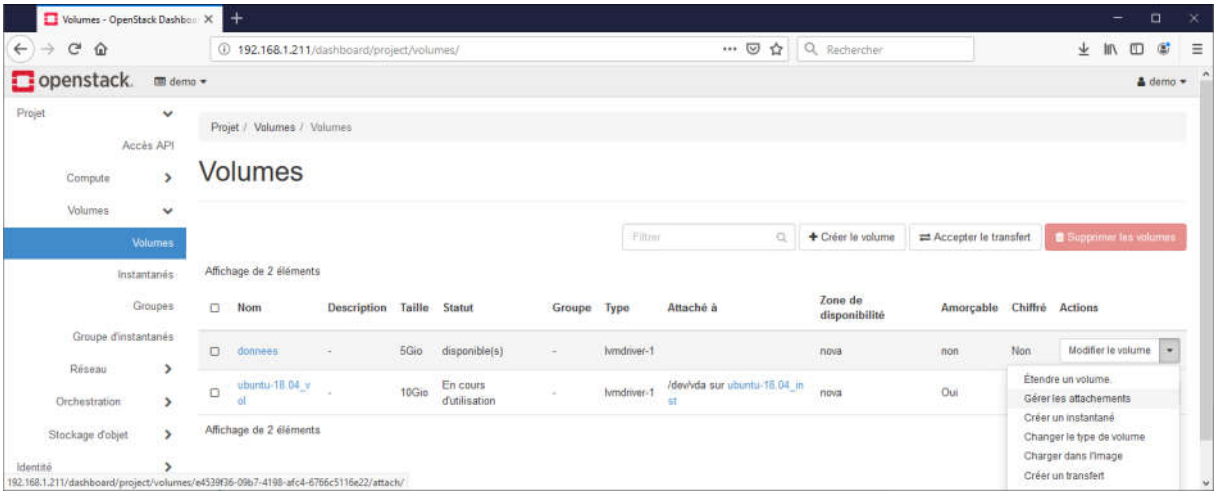
Nous mettons « donnees » comme nom de volume, dans le champ « Source du volume » nous mettons « Aucune source, volume vide », comme taille nous mettons 5 Go par exemple, puis nous cliquons sur « Créer le volume » :



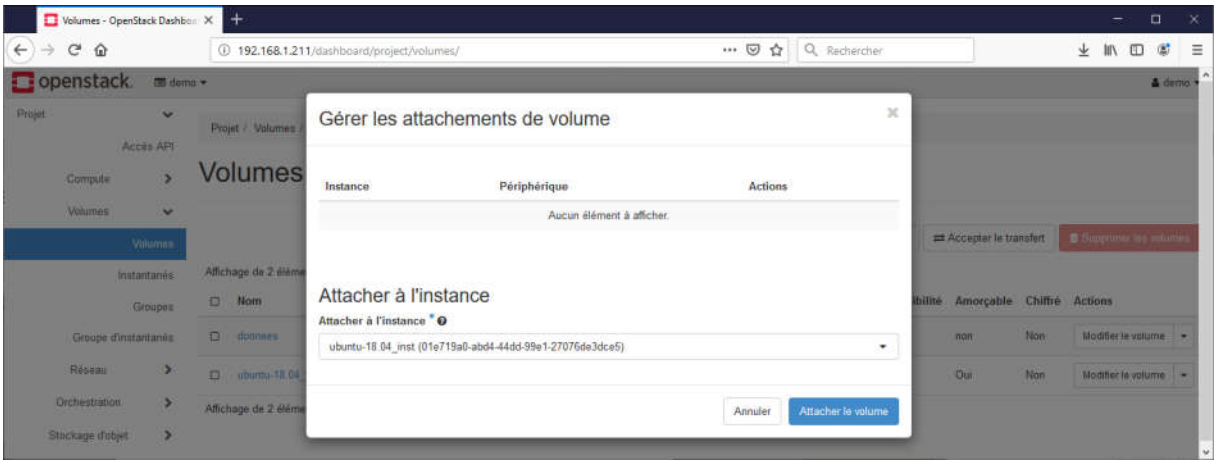
Le statut de notre nouveau volume est « disponible » :



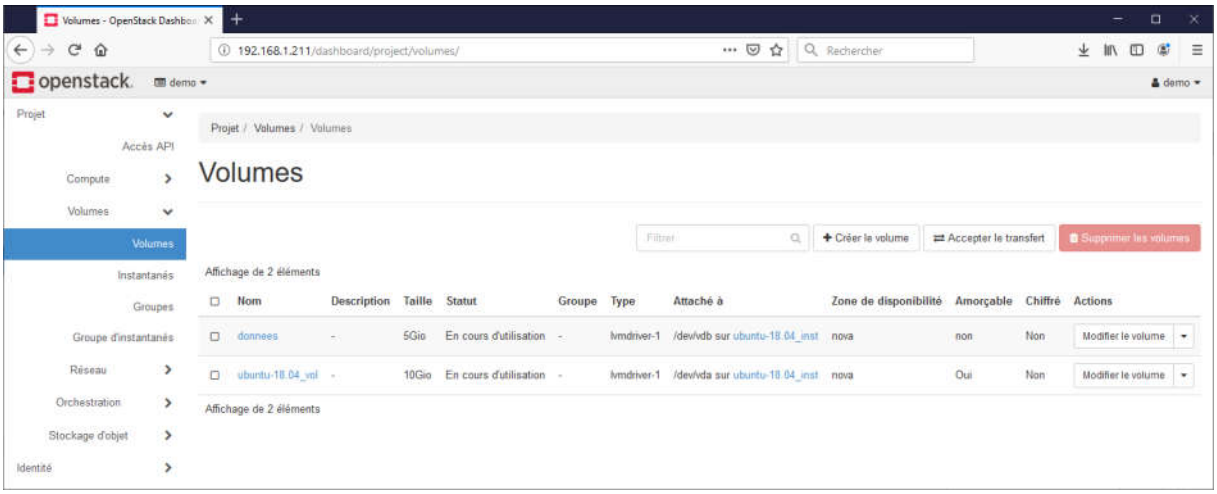
Dans la liste déroulante à droite du volume « donnees »nous sélectionnons « Gérer les attachements » :



Dans la liste « Attacher à l'instance » nous sélectionnons notre instance « Ubuntu-18.04_inst », puis nous cliquons sur « Attacher le volume » :



L'attachement du volume prend quelques secondes, une fois l'attachement terminé nous voyons qu'il est attaché en `/dev/vdb` :



Préparation (partitionnement et formatage) du volume

Nous nous connectons à notre instance, nous voyons le device `/dev/vbd` :

```
stack@microwave:~$ ssh ubuntu@172.24.4.105
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

ubuntu@ubuntu-18:~$ ls -l /dev/vd*
brw-rw---- 1 root disk 252,  0 Oct 13 21:24 /dev/vda
brw-rw---- 1 root disk 252,  1 Oct 13 21:24 /dev/vda1
brw-rw---- 1 root disk 252, 14 Oct 13 21:24 /dev/vda14
brw-rw---- 1 root disk 252, 15 Oct 13 21:24 /dev/vda15
brw-rw---- 1 root disk 252, 16 Oct 15 19:08 /dev/vdb
```

Afin de faciliter un éventuel agrandissement nous allons mettre notre disque en LVM, nous commençons par partitionner le disque avec la commande `fdisk`, nous voyons ensuite le device correspondant à notre partition (`/dev/vdb1`) :

```
ubuntu@ubuntu-18:~$ sudo fdisk /dev/vdb

Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table. ⇒ LE VOLUME N'EST PAS ENCORE PARTITIONNE
Created a new DOS disklabel with disk identifier 0xabdc88e0.

Command (m for help): n  ⇒ n pour new
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1 ⇒ Partition 1
First sector (2048-10485759, default 2048): ⇒ Appuyer sur Entrée
Last sector, +sectors or +size{K,M,G,T,P} (2048-10485759, default 10485759): ⇒ Appuyer sur Entrée

Created a new partition 1 of type 'Linux' and of size 5 GiB.

Command (m for help): t ⇒ t pour type
Selected partition 1
Partition type (type L to list all types): 8e ⇒ 8e pour Linux LVM
Changed type of partition 'Linux' to 'Linux LVM'.

Command (m for help): p ⇒ p pour print (pour vérifier)
Disk /dev/vdb: 5 GiB, 5368709120 bytes, 10485760 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: dos
Disk identifier: 0xabdc88e0

Device     Boot Start      End  Sectors Size Id Type
/dev/vdb1        2048 10485759 10483712   5G 8e Linux LVM

Command (m for help): w ⇒ w pour write, provoque la sortie du programme
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

ubuntu@ubuntu-18:~$ ls -l /dev/vd*
brw-rw---- 1 root disk 252,  0 Oct 13 21:24 /dev/vda
brw-rw---- 1 root disk 252,  1 Oct 13 21:24 /dev/vda1
brw-rw---- 1 root disk 252, 14 Oct 13 21:24 /dev/vda14
brw-rw---- 1 root disk 252, 15 Oct 13 21:24 /dev/vda15
brw-rw---- 1 root disk 252, 16 Oct 15 19:14 /dev/vdb
brw-rw---- 1 root disk 252, 17 Oct 15 19:14 /dev/vdb
```

Nous créons un Physical Volume, un Logical Volume et nous le formatons en `ext4` :

```
ubuntu@ubuntu-18:~$ sudo pvcreate /dev/vdb1
  Physical volume "/dev/vdb1" successfully created

ubuntu@ubuntu-18:~$ sudo vgcreate PV_DONNEES /dev/vdb1
  Volume group "PV_DONNEES" successfully created

ubuntu@ubuntu-18:~$ sudo lvcreate -l 100%FREE -n lv_donnees PV_DONNEES
  Logical volume "lv_donnees" created.

ubuntu@ubuntu-18:~$ sudo mkfs.ext4 /dev/mapper/PV_DONNEES-lv_donnees
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 1309696 4k blocks and 327680 inodes
Filesystem UUID: eae55076-b853-4d6d-9a83-3d2258b5633b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736
```

```
Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Nous créons le point montage, modifions le fichier `/etc/fstab`, montons le système de fichiers et nous créons un fichier de test :

```
ubuntu@ubuntu-18:~$ sudo mkdir /DONNEES

ubuntu@ubuntu-18:~$ sudo tee /etc/fstab << EOF
/dev/mapper/PV_DONNEES-lv_donnees /DONNEES                ext4    defaults        1 2
EOF
/dev/mapper/PV_DONNEES-lv_donnees /DONNEES                ext4    defaults        1 2

ubuntu@ubuntu-18:~$ sudo mount /DONNEES

ubuntu@ubuntu-18:~$ df -h /DONNEES
Filesystem                        Size  Used Avail Use% Mounted on
/dev/mapper/PV_DONNEES-lv_donnees  4.9G   20M  4.6G   1% /DONNEES

ubuntu@ubuntu-18:~$ sudo tee /DONNEES/hello.txt << EOF
Hello
EOF
```

Le partitionnement du volume et le formatage du volume logique provoquent la perte des données existantes. Les commandes correspondantes demandent confirmation s'il y a un risque que des données soient présentes.

Détachement du volume via le Dashboard

Depuis le Dashboard il est possible de détacher le volume de l'instance à laquelle il est attaché en utilisant le menu déroulant à droite du volume dans la liste des volumes.

Détacher un volume revient à débrancher un disque dur d'un serveur physique, s'il était encore monté par le système d'exploitation cela risquerai de provoquer une perte ou une corruption de données.

Cinder et Nova n'ont pas possibilité de savoir si le volume est monté par le système d'exploitation de l'instance, aussi il convient de vérifier avant de procéder au démontage.

Au cas où vous souhaiteriez démonter le volume, il faut démonter la partition et il est préférable de procéder à la désactivation du volume group :

```
ubuntu@ubuntu-18:~$ sudo umount /DONNEES
ubuntu@ubuntu-18:~$ sudo vgchange /dev/mapper/PV_DONNEES -an
  0 logical volume(s) in volume group "PV_DONNEES" now active
```

Suppression d'un volume via le Dashboard

Depuis le Dashboard il est possible de supprimer un volume en utilisant le menu déroulant à droite du volume dans la liste des volumes, pour ceci il ne doit plus être attaché à aucun serveur.

La suppression d'un volume est irréversible, les données qu'il contenait sont définitivement perdues

Nous ne détruisons pas le volume de données que nous venons de créer car nous allons le

réutiliser plus loin.

Création et attachement d'un volume en ligne de commande

Il est également possible de créer un volume et de l'attacher à un serveur en ligne de commande.

On voit là aussi que la création d'un volume (même vide) n'est pas instantanée.

Le nom des volumes n'est pas unique, plusieurs volumes peuvent porter le même nom, si on passe la commande en utilisant le nom et qu'il n'est pas unique une erreur va se produire, dans ce cas il est obligatoire d'utiliser l'ID unique du volume.

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack volume create --size 1 donnees_bis
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| attachments         | []                                   |
| availability_zone   | nova                                 |
| bootable            | false                                |
| consistencygroup_id | None                                 |
| created_at          | 2019-10-15T19:21:36.000000           |
| description         | None                                 |
| encrypted           | False                                |
| id                  | 140b7ad2-0485-4fea-9d63-16e91c5707a8 |
| multiattach         | False                                |
| name                | donnees_bis                          |
| properties          |                                      |
| replication_status  | None                                 |
| size                | 1                                    |
| snapshot_id         | None                                 |
| source_volid        | None                                 |
| status              | creating                             |
| type                | lvmdriver-1                          |
| updated_at          | None                                 |
| user_id             | 57d02b0938d149a1aac96ce6757a3a14     |
+---------------------+--------------------------------------+
stack@microwave:~$ openstack volume list
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| ID                                   | Name            | Status    | Size | Attached to                                 |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| 140b7ad2-0485-4fea-9d63-16e91c5707a8 | donnees_bis     | available |    1 |                                             |
| e4539f36-09b7-4198-afc4-6766c5116e22 | donnees         | in-use    |    5 | Attached to ubuntu-18.04_inst on /dev/vdb   |
| edc26441-2f67-410d-8b24-a1e85844d8dc | ubuntu-18.04_vol | in-use    |   10 | Attached to ubuntu-18.04_inst on /dev/vda   |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
+stack@microwave:~$ openstack server list
+--------------------------------------+-----------------+--------+-----------------------------------+-------+----------+
| ID                                   | Name            | Status | Networks                          | Image | Flavor   |
+--------------------------------------+-----------------+--------+-----------------------------------+-------+----------+
| 01e719a0-abd4-44dd-99e1-27076de3dce5 | ubuntu-18.04_inst | ACTIVE | private=10.0.0.43, 172.24.4.105 |       | m1.small |
+--------------------------------------+-----------------+--------+-----------------------------------+-------+----------+
stack@microwave:~$ openstack server add volume ubuntu-18.04_inst donnees_bis
stack@microwave:~$ openstack volume list
+--------------------------------------+-----------------+--------+------+---------------------------------------------+
| ID                                   | Name            | Status | Size | Attached to                                 |
+--------------------------------------+-----------------+--------+------+---------------------------------------------+
| 140b7ad2-0485-4fea-9d63-16e91c5707a8 | donnees_bis     | in-use |    1 | Attached to ubuntu-18.04_inst on /dev/vdc   |
| e4539f36-09b7-4198-afc4-6766c5116e22 | donnees         | in-use |    5 | Attached to ubuntu-18.04_inst on /dev/vdb   |
| edc26441-2f67-410d-8b24-a1e85844d8dc | ubuntu-18.04_vol | in-use |   10 | Attached to ubuntu-18.04_inst on /dev/vda   |
+--------------------------------------+-----------------+--------+------+---------------------------------------------+
```

L'option '`-f json`' de la commande `openstack` permet d'afficher le résultat de la commande au format JSON au lieu d'afficher le résultat sous forme de tableau. Le format est humainement lisible et le résultat plus complet que sous forme de tableau :

```
stack@microwave:~$ openstack volume show donnees_bis -f json
{
  "status": "in-use",
  "properties": {},
  "user_id": "57d02b0938d149a1aac96ce6757a3a14",
  "description": null,
  "availability_zone": "nova",
  "bootable": "false",
  "encrypted": false,
  "created_at": "2019-10-15T19:21:36.000000",
```

```
  "multiattach": false,
  "updated_at": "2019-10-15T19:25:18.000000",
  "source_volid": null,
  "name": "donnees_bis",
  "snapshot_id": null,
  "consistencygroup_id": null,
  "replication_status": null,
  "os-vol-tenant-attr:tenant_id": "5cd4212611654857a9e941a7b5d35e13",
  "size": 1,
  "type": "lvmdriver-1",
  "id": "140b7ad2-0485-4fea-9d63-16e91c5707a8",
  "attachments": [
    {
      "server_id": "01e719a0-abd4-44dd-99e1-27076de3dce5",
      "attachment_id": "30162df5-3788-4aff-a3b7-a743beec55a9",
      "attached_at": "2019-10-15T19:25:17.000000",
      "host_name": "microwave",
      "volume_id": "140b7ad2-0485-4fea-9d63-16e91c5707a8",
      "device": "/dev/vdc",
      "id": "140b7ad2-0485-4fea-9d63-16e91c5707a8"
    }
  ]
}
```

Détachement et suppression d'un volume en ligne de commande

Nous pouvons détacher le deuxième volume du serveur puis détruire le volume :

```
stack@microwave:~$ openstack server remove volume ubuntu-18.04_inst donnees_bis
stack@microwave:~$ openstack volume delete donnees_bis
stack@microwave:~$ openstack volume list
+--------------------------------------+------------------+--------+------+-----------------------------------------+
| ID                                   | Name             | Status | Size | Attached to                             |
+--------------------------------------+------------------+--------+------+-----------------------------------------+
| e4539f36-09b7-4198-afc4-6766c5116e22 | donnees          | in-use |    5 | Attached to ubuntu-18.04_inst on /dev/vdb  |
| edc26441-2f67-410d-8b24-a1e85844d8dc | ubuntu-18.04_vol | in-use |   10 | Attached to ubuntu-18.04_inst on /dev/vda  |
+--------------------------------------+------------------+--------+------+-----------------------------------------+
```

Il vous est laissé à titre d'exercice de vérifier si le filesystem du volume '`donnees`' a été démonté en vous connectant à l'instance puis de détacher le volume de l'instance (s'il n'a pas été démonté la commande de démontage et de désactivation du volumegroup est indiquée un peu plus haut).

Ne détruisez pas le volume '`donnees`' il va resservir.

L'orchestration

L'orchestration est une des fonctionnalités puissantes du Cloud, elle permet de déployer des infrastructures de manière automatique et reproductible, elle se base sur un langage de description des infrastructures et un langage de template (modèles).

L'orchestration permet aussi d'assurer l'élasticité, c'est à dire de faire varier les infrastructures en fonction de la charge.

Différents systèmes d'orchestration existent, ceci grâce à l'ouverture des API OpenStack.

L'orchestration avec Heat

Heat est un des projets de OpenStack.

Heat permet de décrire les infrastructures sous forme de fichier texte, l'infrastructure peut comprendre la plupart des types de ressources (instances, volumes, routeurs, réseaux…)

La description d'une infrastructure pour Heat s'appelle un template (modèle). Déployer un template dans un tenant (éventuellement avec des paramètres) va créer une « pile » de ressources.

Heat est composé :

- d'un service en charge de synchroniser les opérations nécessaires à la création de l'infrastructure décrite
- d'une API
- d'une interface en ligne de commande qui s'intègre à la commande CLI `openstack`.

Un template de base est composé de 4 parties :

- Une description du template, à titre purement documentaire
- Un ensemble de paramètres qui permettent d'adapter le template lors de chaque déploiement et de créer une pile adaptée au client et à l'environnement, ces paramètres ont une valeur par défaut qui peut être écrasée lors de la création de la pile, soit depuis des champs de saisie dans le Dashboard, soit par des paramètres passés en ligne de commande lors de la création de la pile à l'aide du CLI `openstack`
- Un ensemble de ressources qui vont être créées et liées les unes aux autres (des volumes aux serveurs, des interfaces à des réseaux…)
- Des sorties qui vont nous permettre de récupérer des informations sur la pile créée (adresses IP, URL...), ces données sont décrites et mises en forme dans la pile

Les ressources

On l'a dit, une pile est composée de ressources, ce sont les objets que l'on peut créer dans OpenStack, la pile décrit également les liaisons entre ces ressources.

Ces ressources sont définies par un type, par exemple une instance est de type `OS::Nova::Server` (`OS` car il s'agit d'un objet de OpenStack, `Nova` car Nova est le composant de OpenStack qui gère les serveurs), `OS::Nova::KeyPair` est une paire de clé. Il existe également des objets de type `AWS::` pour assurer la compatibilité avec AWS, le cloud de Amazon.

Dans le Dashboard on retrouve les différents types d'objets disponibles ainsi que leur documentation succincte dans le menu « Projet → Orchestration → Type de ressources ».

Une première pile

Notre premier template va être simple, il va créer une instance à partir de l'image Ubuntu que nous avons chargée dans OpenStack, et un script bash va être utilisé pour installer deux packages logiciels et exécuter deux commandes.

Recopiez le template ci-dessous dans un fichier que vous appellerez `premiere_stack.yaml`, il est également disponible à cette adresse : https://u03.fr/90openstack/premiere_stack.yaml

```
heat_template_version: 2013-05-23

description: >
  Notre premiere pile Heat

parameters:
  nom_de_cle:
    type: string
    description: Nom de la cle

  gabarit_instance:
    type: string
    description:
    default: m1.small

  image_instance:
    type: string
    description: Nom de l'image
    default: ubuntu-18.04

  reseau_instance:
    type: string
    description: Nom du reseau de raccordement de l'instance
    default: private

resources:
  instance_heat:
    type: OS::Nova::Server
    properties:
      image: { get_param: image_instance }
      flavor: { get_param: gabarit_instance }
      key_name: { get_param: nom_de_cle }
      networks:
       - network: { get_param: reseau_instance }
      user_data: |
        #!/bin/bash -v

        apt-get update
        apt-get install -y cowsay fortune
        /usr/games/fortune | /usr/games/cowsay
        /usr/games/fortune | /usr/games/cowsay > /dev/tty1
        cat << EOF >> /home/ubuntu/.bashrc
        /usr/games/fortune | /usr/games/cowsay
        EOF

outputs:
  salutations:
    description: La salutation du jour
    value: Hello World!!!
```

On y retrouve les 4 parties (« `description` », « `parameters` », « `ressources` » et « `outputs` »).

Dans la section « `ressources` » nous n'avons qu'une ressource, il s'agit d'une instance de serveur, son identifiant est « `instance_heat` », grâce à cet identifiant, utilisable uniquement dans la définition de pile, la ressource pourra être référencée ailleurs dans la pile (dans la définition d'autres ressources ou dans la section « `outputs` »).

Le type de la ressource est « `OS::Nova::Server` », ses propriétés sont :

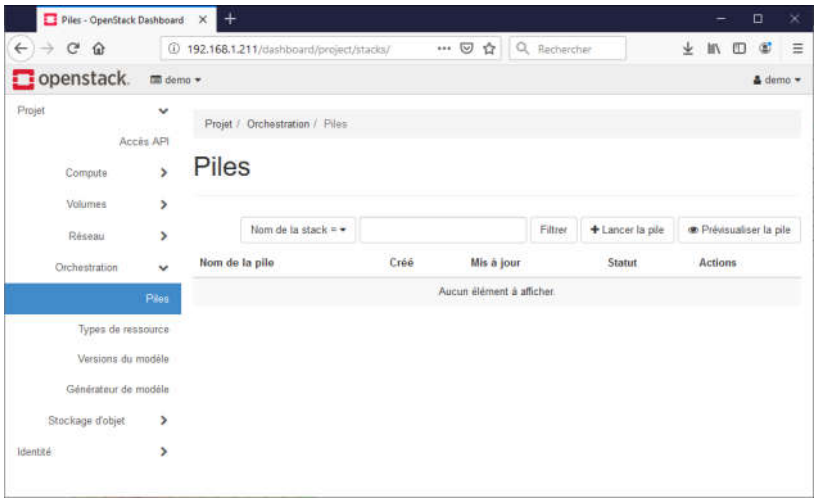
- `image` : le nom de l'image, ce nom provient du paramètre « `image_instance` »
- `flavor` : il s'agit du gabarit, ce nom provient du paramètre « `gabarit_instance` »

- `key_name` : le nom de la paire de clé utilisée pour se connecter en SSH à l'instance, une clé avec ce nom doit être enregistrée dans le projet
- `user_data` : ce champ s'étend sur plusieurs lignes (grâce au « | »), il s'agit du script qui va s'exécuter au moment du lancement de l'instance, ici ce champ est constant mais nous verrons qu'il est possible de modifier ce champ en fonction des valeurs des paramètres passés lors de la création de la pile.

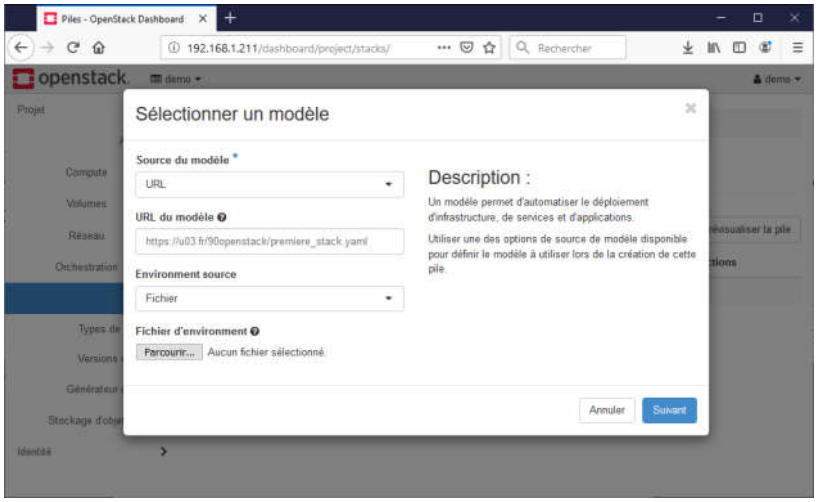
La ligne **`heat_template_version: 2013-05-23`** ne correspond pas à la version du template, mais à la version du langage de template utilisée, ce qui correspond à la version d'OpenStack pour laquelle le template a été initialement écrit. N'essayez donc pas d'y mettre la date du jour par exemple.

Utilisation de la pile

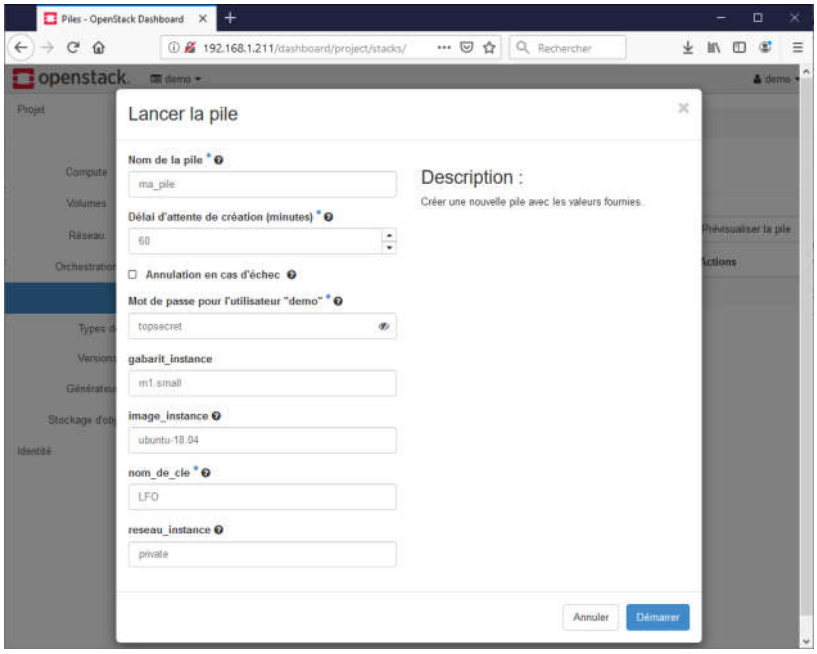
L'utilisation des piles se fait dans le Dashboard dans le menu « Projet → Orchestration → Piles », pour lancer une pile il faut cliquer sur le bouton « Lancer la pile » :



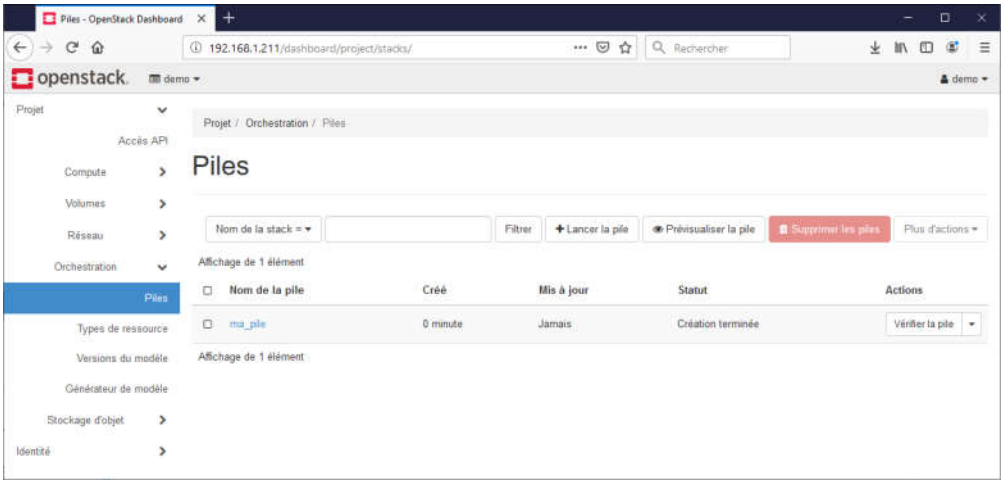
La source du modèle (le template) est `premiere_stack.yaml`, (disponible un peu plus haut dans le document ou à l'adresse https://u03.fr/90openstack/premiere_stack.yaml). Vous pouvez au choix utiliser une copie locale sur votre poste de travail (source du modèle 'Fichier' et cliquez sur « Parcourir »), soit utiliser directement le fichier via son URL. Le « champ environnement » permet de sélectionner un fichier de paramètres qui vont venir écraser les paramètres par défaut spécifiés dans le template, nous ne l'utilisons pas pour le moment. Cliquez sur « Suivant » :



Nous allons nommer notre pile « `ma_pile` », nous tapons notre mot de passe dans le champ prévu à cet effet, et nous tapons le nom de notre clé SSH dans le champ « `nom_de_cle` » puis nous cliquons sur « Démarrer » :

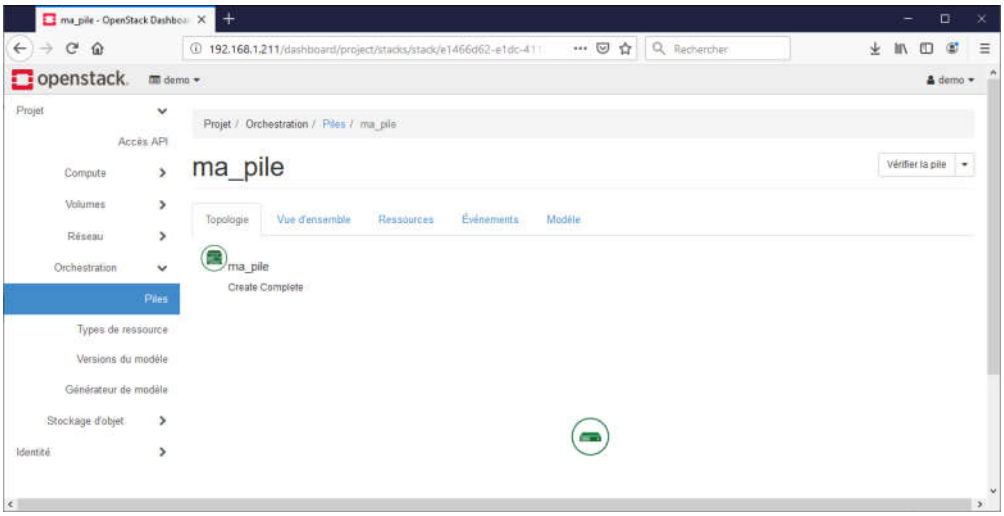


La création de la pile va demander environ une minute :

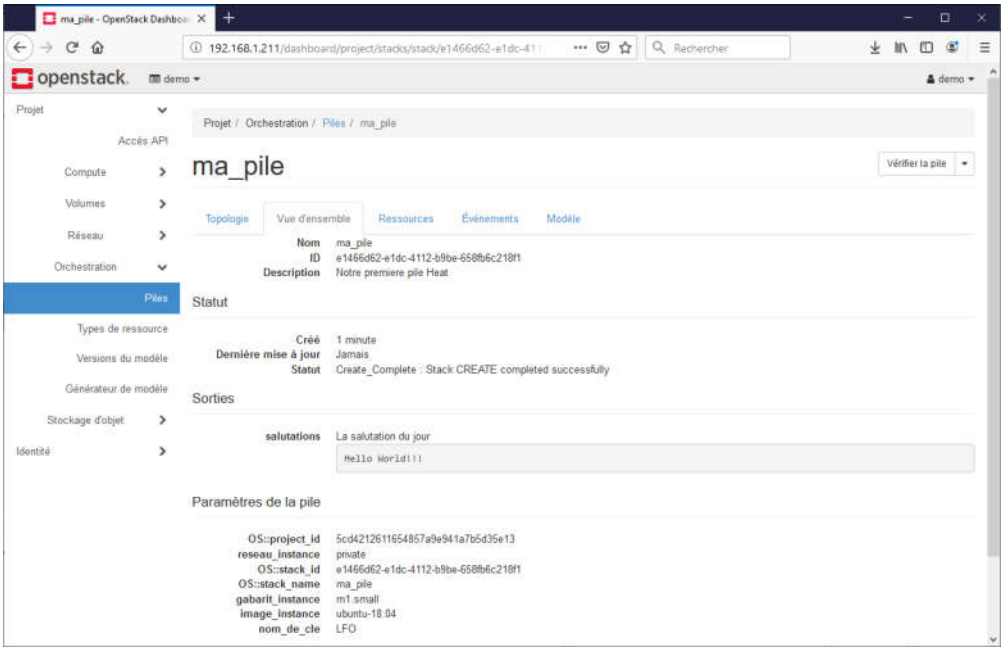


En cliquant sur « ma_pile » nous affichons par défaut la « topologie » de notre pile, elle est simple, il n'y a que notre serveur.

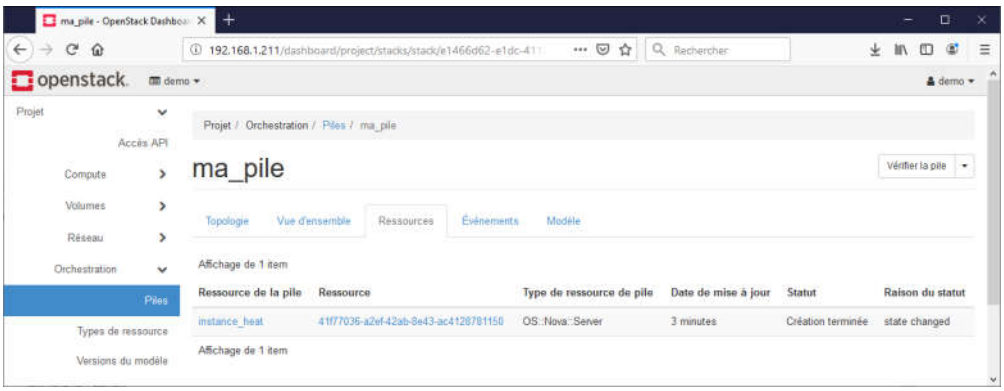
Nous voyons que l'écran « ma_pile » est composé de 5 onglets « Topologie » , « Vue d'ensemble », « Ressources », « Événements » et « Modèle ».



L'onglet « Vue d'ensemble » nous donne les informations essentielles sur la pile, on y retrouve la description présente dans la définition de la pile, l'état de la pile, les paramètres qui ont été passés et les sorties (Outputs) qui ont été définies dans la pile dans la section « outputs », nous retrouvons ici nos salutations :

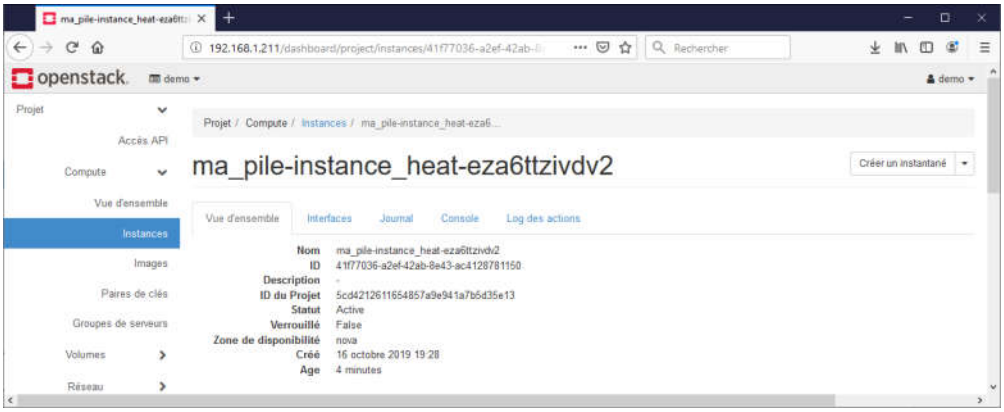


L'onglet « Ressouces » donne les différentes ressources qui composent notre pile, ici nous n'avons qu'un serveur, dont le nom est « `instance_heat` » dans la définition de notre pile :

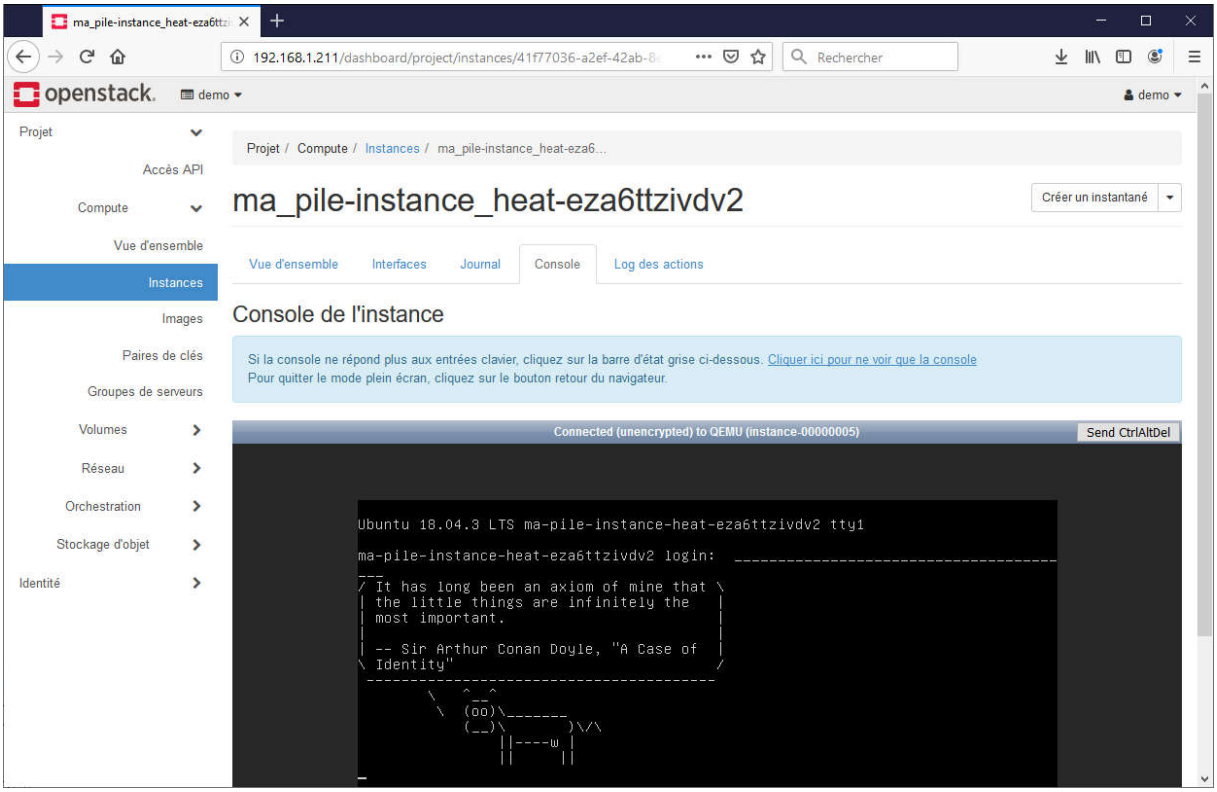


Si nous cliquons sur l'identifiant en hexadécimal de notre instance nous allons nous retrouver sur l'écran de vue d'ensemble de notre instance, de la même manière que si vous avions cliqué dessus dans l'écran « Projet → Compute → Instances ».

On aperçoit au passage que le nom de notre instance est composé du nom de la pile, du nom de la ressource dans la pile et d'une chaîne de caractères aléatoires, nous aurions pu donner un nom à cette instance en utilisant l'attribut « `name` » dans la définition de la ressource :



Cliquer sur l'onglet « console » nous affiche la console, on peut y voir une vache qui prononce une phrase amusante en anglais (cliquez dans la partie noire de l'écran puis tapez sur la touche espace si l'écran de la console est entré en veille) :



Il vous est laissé à titre d'exercice depuis le Dashboard d'attacher une adresse IP flottante à l'instance, d'autoriser la connexion SSH dans les groupes de sécurité et de vous connecter à l'instance avec l'utilisateur ubuntu, vous serez également accueilli par la vache.

Suppression de la pile

Dans le menu « Projet → Orchestration → Piles » il est possible de supprimer la pile en utilisant le menu déroulant à droite de la pile.

La suppression d'une pile entraîne la suppression de la totalité des éléments créés par la pile, il s'agit d'une opération irréversible.

Autres types de ressources de Heat

Nous avons créé une première pile qui comportait juste une instance de base et utilisait quelques paramètres.

D'autres types de ressources permettent de construire des infrastructures plus évoluées.

Création d'une groupe de sécurité et d'une adresse IP flottante

Nous allons voir comment créer un groupe de sécurité et l'attacher ainsi qu'une adresse IP flottante à notre instance.

Le groupe de sécurité de sécurité est créé en même temps que les règles (ici il y a une seule règle qui autorise les accès en SSH (tcp port 22), en entrée (ingress), depuis toutes les adresses IP (0.0.0.0/0).

Le groupe de sécurité est ensuite attachée à l'instance avec la propriété `security_groups` de l'instance grâace à `get_resource` :

```
  instance_heat_security_group:
    type: OS::Neutron::SecurityGroup
    properties:
      rules:
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
          direction: ingress

  […/…]
  instance_heat:
    type: OS::Nova::Server
      security_groups: [{ get_resource: instance_heat_security_group }]
```

La gestion de l'adresse IP flottante est différente, on crée une adresse IP flottante (`OS::Neutron::FloatingIP`) dans le réseau '`public`' et on crée une deuxième ressource qui est une association d'adresse IP flottante (`OS::Nova::FloatingIPAssociation`)

```
  floating_ip_instance_heat:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: { get_param: reseau_public }

  association_reverseproxy:
    type: OS::Nova::FloatingIPAssociation
    properties:
      floating_ip: { get_resource: floating_ip_instance_heat }
      server_id: { get_resource: instance_heat }
```

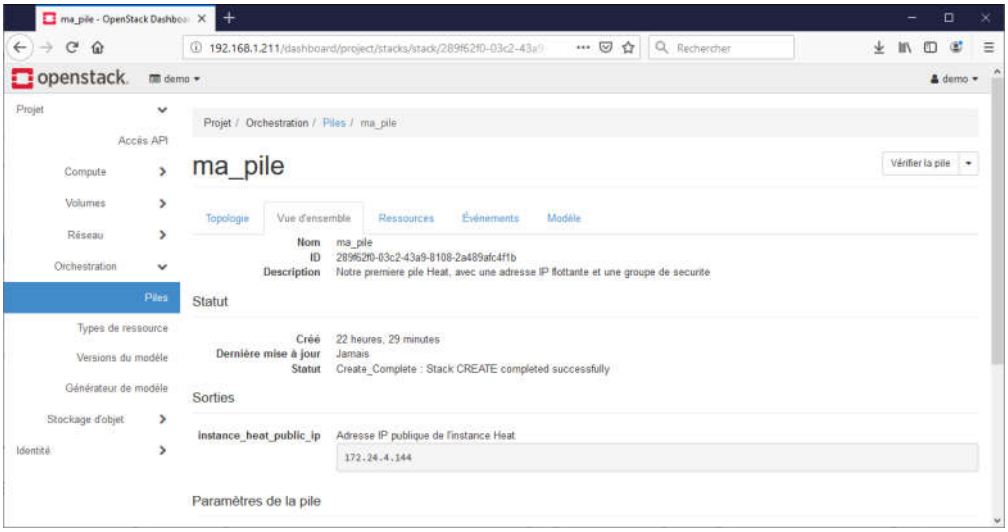
L'adresse IP flottante du serveur va être attribuée par le système, nous pouvons modifier la section « outputs » pour afficher l'adresse IP qui a été attribuée :

```
outputs:
   instance_heat_public_ip:
      description: Adresse IP publique de l'instance Heat
      value: { get_attr: [ floating_ip_instance_heat, floating_ip_address ] }
```

La pile complète est disponible à cette adresse :

- https://u03.fr/90openstack/groupe_securite_ip.yaml

Une fois la création de la pile terminée nous avons l'adresse IP flottante qui a été attribuée dans l'onglet « Vue d'ensemble de la pile » :



Nous pouvons désormais nous connecter en SSH à notre serveur en utilisant l'adresse IP flottante qui lui a été allouée :

```
stack@microwave:~$ ssh ubuntu@172.24.4.144
The authenticity of host '172.24.4.6 (172.24.4.6)' can't be established.
ECDSA key fingerprint is SHA256:Bw5AKL7SQk+AfMXSv1WUQAAMt9u3eQHC3W2Dx8vEjhY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.24.4.144' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

 _____________________________________
/ You are confused; but this is your \
\ normal state.                      /
 -------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ubuntu@ip-flottante-instance-heat-a4sp3gehehc4:~$
```

L'inconvénient de créer l'adresse IP flottante dans la pile est que celle-ci est libérée lors de la destruction de la pile, et lors de la recréation de la pile une nouvelle adresse IP sera allouée.

Cela rend impossible la création d'entrées DNS qui pointent vers l'application.

C'est pour cela qu'on alloue les IP flottantes en dehors de la pile et qu'on les associe au serveur dans la pile.

Il vous est laissé à titre d'exercice de supprimer la pile.

Utilisation d'une adresse IP flottante

Ci-dessous la création de l'adresse IP flottante, la définition du paramètre dans la pile, et l'associaton de l'adresse IP flottante au serveur. Globalement on utilise '`get-param`' au lieu d'un '`get-ressource`'.

```
stack@microwave:~$ openstack floating ip create public
+---------------------+--------------------------------------+
```

```
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2018-03-26T17:38:04Z                 |
| description         |                                      |
| fixed_ip_address    | None                                 |
| floating_ip_address | 172.24.4.14                          |
| floating_network_id | 087bc8a1-65e2-4747-b96e-4dd421c542e5 |
| id                  | dd92bcbc-77a5-4624-bf3b-b02489da4ee1 |
| name                | 172.24.4.14                          |
| port_id             | None                                 |
| project_id          | 3f36cd33d14e468293caa7bf311c00de     |
| qos_policy_id       | None                                 |
| revision_number     | 0                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| subnet_id           | None                                 |
| updated_at          | 2018-03-26T17:38:04Z                 |
+---------------------+--------------------------------------+
```

```
parameters:
  id_floating_ip_reverseproxy:
    type: string
    description: ID de la floating IP pour le reverse-proxy
    default: dd92bcbc-77a5-4624-bf3b-b02489da4ee1

resources:
  association_reverseproxy:
    type: OS::Nova::FloatingIPAssociation
    properties:
      floating_ip: { get_param: id_floating_ip_reverseproxy }
      server_id: { get_resource: instance_reverseproxy }
```

Création d'un volume à partir d'une image

Comme nous l'avons vu, l'orchestration permet de déployer automatiquement des infrastructures, en particulier pour des mises à jour du système ou de l'application.

Pour ceci il est possible de redéployer des serveurs à partir d'une image système mise à jour. Tout comme nous l'avions fait à partir du Dashboard :

```
  instance_heat_volume:
    type: OS::Cinder::Volume
    properties:
      size: 10
      name: instance_heat_volume
      image: { get_param: image_instance }
```

Le volume est ensuite une ressource utilisable dans la définition d'une instance, grâce à la propriété `block_device_mapping` de l'instance, « `get_ressource` » permettant de référencer la ressource correspondant au volume par son identifiant :

```
  instance_heat:
    type: OS::Nova::Server
    properties:
      name: une_instance
      flavor: { get_param: instance_type_heat }
      block_device_mapping: [{ device_name: "vda", volume_id : { get_resource : instance_heat_volume } }]
```

La pile est disponible à cette adresse : https://u03.fr/90openstack/volume_systeme.yaml

Il vous est laissé à titre d'exercice créer la pile, de vous connecter au serveur et de supprimer la pile.

Attachement d'un volume de données à une instance

Les volumes de données contiennent les données persistantes de l'application, lors du redéploiement de l'application il faut les ré-attacher aux instances concernées.

L'attachement du volume système dans l'exemple précédent se faisait en utilisant `get_resource` pour récupérer l'`id` du volume créé par la pile.

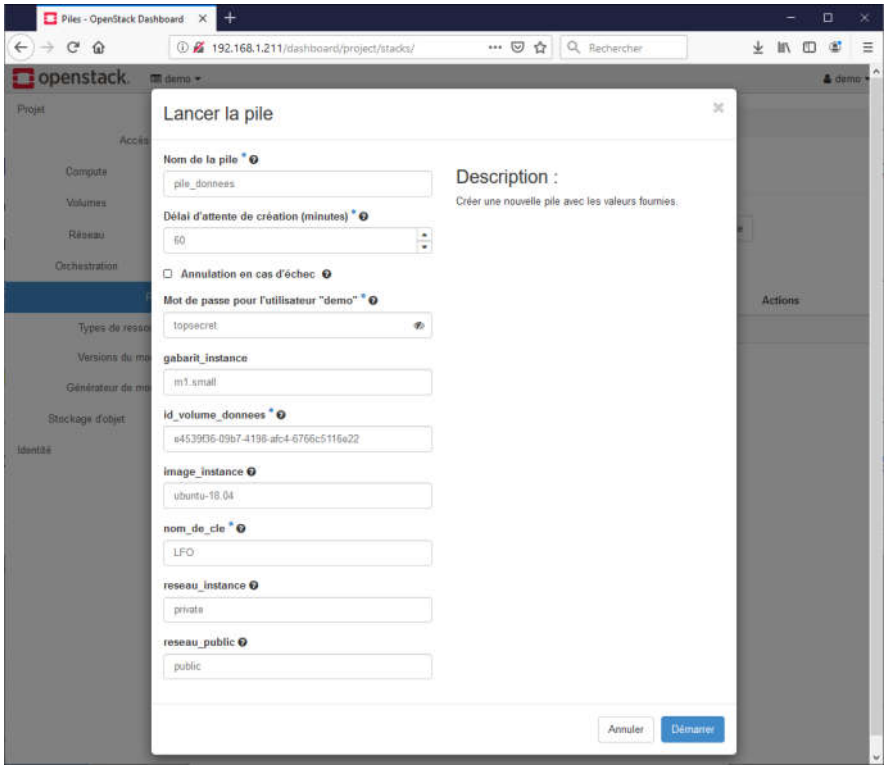
L'attachement est le même que pour le volume système, sauf que la valeur de l'`id` va être passée en paramètre :

```
parameters:
  id_volume_donnees:
    type: string
    description: ID du volume de donnees a attacher

  […/…]
```

Nous allons réutiliser le volume que nous avions créé précédemment et qui s'appelait « donnees », l'attachement se faisant par id du volume (car seuls les id sont uniques dans le système), nous obtenons son identifiant depuis le Dashboard ou en ligne de commande :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack volume list
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| ID                                   | Name            | Status    | Size | Attached to                                 |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| e4539f36-09b7-4198-afc4-6766c5116e22 | donnees         | available |    5 |                                             |
| edc26441-2f67-410d-8b24-a1e85844d8dc | ubuntu-18.04_vol | in-use    |   10 | Attached to ubuntu-18.04_inst on /dev/vda   |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
```

La pile complète est disponible à cette adresse : https://u03.fr/90openstack/volume_donnees.yaml



Nous avions créé un fichier de test dans le volume de données lors de sa création, nous le retrouvons dans notre instance :

```
ubuntu@pile-instance-heat-vfn4bx77nznj:~$ df -h /DONNEES/
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/PV_DONNEES-lv_donnees  4.8G   10M  4.6G   1% /DONNEES
ubuntu@pile-instance-heat-vfn4bx77nznj:~$ cat /DONNEES/hello.txt
Hello
```

Utilisation des piles à l'aide du CLI openstack

Toutes les opérations sous OpenStack sont effectuées grâce aux API, le Dashboard et le CLI OpenStack sont deux des moyens d'utiliser ces API, il est donc possible d'utiliser le CLI OpenStack pour gérer les piles.

Lancement d'une pile

Nous spécifions les différents paramètres sur la ligne de commande, la création d'une pile est asynchrone et prend un certain temps en fonction des ressources à créer :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack stack create --wait                                                    \
                                          --parameter nom_de_cle="LFO"                              \
                                          --parameter id_volume_donnees="e4539f36-09b7-4198-afc4-6766c5116e22"  \
                                          -t https://u03.fr/90openstack/volume_donnees.yaml         \
                                          ma_pile_donnees
2019-10-19 20:31:04Z [ma_pile_donnees]: CREATE_IN_PROGRESS  Stack CREATE started
2019-10-19 20:31:04Z [ma_pile_donnees.instance_heat_volume]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:06Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:06Z [ma_pile_donnees.instance_heat_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:07Z [ma_pile_donnees.instance_heat_security_group]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.instance_heat_volume]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:38Z [ma_pile_donnees.instance_heat]: CREATE_COMPLETE  state changed
2019-10-19 20:31:38Z [ma_pile_donnees.association_reverseproxy]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:45Z [ma_pile_donnees.association_reverseproxy]: CREATE_COMPLETE  state changed
2019-10-19 20:31:45Z [ma_pile_donnees]: CREATE_COMPLETE  Stack CREATE completed successfully
+---------------------+--------------------------------------------------------------------------------+
| Field               | Value                                                                          |
+---------------------+--------------------------------------------------------------------------------+
| id                  | 41325438-8de5-473e-b310-b54f42bd586f                                           |
| stack_name          | ma_pile_donnees                                                                |
| description         | Creation d'un volume systeme a partir d'une image et attachement a l'instance |
|                     |                                                                                |
| creation_time       | 2019-10-19T20:31:04Z                                                           |
| updated_time        | None                                                                           |
| stack_status        | CREATE_COMPLETE                                                                |
| stack_status_reason | Stack CREATE completed successfully                                            |
+---------------------+--------------------------------------------------------------------------------+
```

Une fois la création de la pile terminée nous retrouvons les données présentes dans la section « outputs » de la définition de notre pile :

```
stack@microwave:~$ openstack stack show ma_pile_donnees
+-----------------------+--------------------------------------------------------------------------------+
| Field                 | Value                                                                          |
+-----------------------+--------------------------------------------------------------------------------+
| id                    | 41325438-8de5-473e-b310-b54f42bd586f                                           |
| stack_name            | ma_pile_donnees                                                                |
| description           | Creation d'un volume systeme a partir d'une image et attachement a l'instance |
|                       |                                                                                |
| creation_time         | 2019-10-19T20:31:04Z                                                           |
| updated_time          | None                                                                           |
| stack_status          | CREATE_COMPLETE                                                                |
| stack_status_reason   | Stack CREATE completed successfully                                            |
| parameters            | OS::project_id: 5cd4212611654857a9e941a7b5d35e13                               |
|                       | OS::stack_id: 41325438-8de5-473e-b310-b54f42bd586f                             |
|                       | OS::stack_name: ma_pile_donnees                                                |
|                       | gabarit_instance: m1.small                                                     |
|                       | id_volume_donnees: e4539f36-09b7-4198-afc4-6766c5116e22                        |
|                       | image_instance: ubuntu-18.04                                                   |
|                       | nom_de_cle: LFO                                                                |
|                       | reseau_instance: private                                                       |
|                       | reseau_public: public                                                          |
|                       |                                                                                |
| outputs               | - description: Adresse IP publique de l'instance Heat                          |
|                       |   output_key: instance_heat_public_ip                                          |
|                       |   output_value: 172.24.4.166                                                   |
|                       |                                                                                |
| parent                | None                                                                           |
| disable_rollback      | True                                                                           |
| deletion_time         | None                                                                           |
| stack_user_project_id | 18bddd0b6d314b2faf056b0b3ebc50f0                                               |
| capabilities          | []                                                                             |
| notification_topics   | []                                                                             |
| stack_owner           | None                                                                           |
| timeout_mins          | None                                                                           |
| tags                  | None                                                                           |
+-----------------------+--------------------------------------------------------------------------------+
```

Il est possible de lister les différentes ressources qui composent notre pile, notez au passage comment il est possible de sélectionner les colonnes affichées:

```
stack@microwave:~$ openstack stack resource list ma_pile_donnees \
                       -c resource_name -c physical_resource_id -c resource_type -c resource_status
+-----------------------------+--------------------------------------+--------------------------------+-----------------+
| resource_name               | physical_resource_id                 | resource_type                  | resource_status |
+-----------------------------+--------------------------------------+--------------------------------+-----------------+
| instance_heat               | 2d8cad6c-e75d-44f1-acc9-48e61459b0a7 | OS::Nova::Server               | CREATE_COMPLETE |
| instance_heat_volume        | 79f521fa-daca-4699-80a8-0ceaf8e941d5 | OS::Cinder::Volume             | CREATE_COMPLETE |
| floating_ip_instance_heat   | b88cf79f-1a54-4971-9cd9-f2b7f50fa08f | OS::Neutron::FloatingIP        | CREATE_COMPLETE |
| association_reverseproxy    | 19                                   | OS::Nova::FloatingIPAssociation | CREATE_COMPLETE |
| instance_heat_security_group | 503bfcc6-892b-45d3-ad43-ea033e7a5ff4 | OS::Neutron::SecurityGroup     | CREATE_COMPLETE |
+-----------------------------+--------------------------------------+--------------------------------+-----------------+
```

Il est ensuite possible d'afficher une ressource particulière à l'aide de la commande ci-dessous :

```
stack@microwave:~$ openstack stack resource show ma_pile_donnees instance_heat_volume
```

Vous pouvez aussi obtenir la liste des évènements liées à la pile :

```
stack@microwave:~$ openstack stack event list ma_pile_donnees
2019-10-19 20:31:04Z [ma_pile_donnees]: CREATE_IN_PROGRESS  Stack CREATE started
2019-10-19 20:31:04Z [ma_pile_donnees.instance_heat_volume]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:06Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:06Z [ma_pile_donnees.instance_heat_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:07Z [ma_pile_donnees.instance_heat_security_group]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.instance_heat_volume]: CREATE_COMPLETE  state changed
2019-10-19 20:31:09Z [ma_pile_donnees.instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:38Z [ma_pile_donnees.instance_heat]: CREATE_COMPLETE  state changed
2019-10-19 20:31:38Z [ma_pile_donnees.association_reverseproxy]: CREATE_IN_PROGRESS  state changed
2019-10-19 20:31:45Z [ma_pile_donnees.association_reverseproxy]: CREATE_COMPLETE  state changed
2019-10-19 20:31:45Z [ma_pile_donnees]: CREATE_COMPLETE  Stack CREATE completed successfully
```

Mise à jour d'une pile

Il est possible de mettre à jour une pile (lorsque certains paramètres ou définitions ont été modifiés), lors de la mise à jour de la pile les ressources dont la définition a été modifiée sont détruits et recréés.

Nous ajoutons une règle dans le groupe de sécurité pour ajouter le port 80 :

```
  instance_heat_security_group:
    type: OS::Neutron::SecurityGroup
    properties:
      rules:
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
          direction: ingress
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 80
          port_range_max: 80
          direction: ingress
```

La pile mise à jour est disponible à cette adresse :

- https://u03.fr/90openstack/volume_donnees_update.yaml

Nous mettons à jour la pile :

```
stack@microwave:~$ openstack stack update --wait                                                \
                                          --parameter nom_de_cle="LFO"                          \
                                          --parameter id_volume_donnees="e4539f36-09b7-4198-afc4-6766c5116e22"  \
                                          -t https://u03.fr/90openstack/volume_donnees_update.yaml  \
                                          ma_pile_donnees
2019-10-19 20:41:08Z [ma_pile_donnees]: UPDATE_IN_PROGRESS  Stack UPDATE started
2019-10-19 20:41:10Z [ma_pile_donnees.instance_heat_security_group]: UPDATE_IN_PROGRESS  state changed
2019-10-19 20:41:12Z [ma_pile_donnees.instance_heat_security_group]: UPDATE_COMPLETE  state changed
2019-10-19 20:41:13Z [ma_pile_donnees]: UPDATE_COMPLETE  Stack UPDATE completed successfully
+--------------------+-------------------------------------------------------------------------------+
| Field              | Value                                                                         |
+--------------------+-------------------------------------------------------------------------------+
```

```
| id                  | 41325438-8de5-473e-b310-b54f42bd586f                                                    |
| stack_name          | ma_pile_donnees                                                                         |
| description         | Creation d'un volume systeme a partir d'une image et attachement a l'instance          |
|                     |                                                                                         |
| creation_time       | 2019-10-19T20:31:04Z                                                                    |
| updated_time        | 2019-10-19T20:41:08Z                                                                    |
| stack_status        | UPDATE_COMPLETE                                                                         |
| stack_status_reason | Stack UPDATE completed successfully                                                     |
+---------------------+-----------------------------------------------------------------------------------------+
stack@microwave:~$ openstack stack list
+--------------------------------------+----------------+-----------------+----------------------+----------------------+
| ID                                   | Stack Name     | Stack Status    | Creation Time        | Updated Time         |
+--------------------------------------+----------------+-----------------+----------------------+----------------------+
| 41325438-8de5-473e-b310-b54f42bd586f | ma_pile_donnees | UPDATE_COMPLETE | 2019-10-19T20:31:04Z | 2019-10-19T20:41:08Z |
+--------------------------------------+----------------+-----------------+----------------------+----------------------+
```

Vérifions la bonne mise à jour du groupe de sécurité :

```
stack@microwave:~$ openstack stack resource list ma_pile_donnees \
                        -c resource_name -c physical_resource_id -c resource_type -c resource_status
+-----------------------------+--------------------------------------+-------------------------------+-----------------+
| resource_name               | physical_resource_id                 | resource_type                 | resource_status |
+-----------------------------+--------------------------------------+-------------------------------+-----------------+
| instance_heat               | 2d8cad6c-e75d-44f1-acc9-48e61459b0a7 | OS::Nova::Server              | CREATE_COMPLETE |
| instance_heat_volume        | 79f521fa-daca-4699-80a8-0ceaf8e941d5 | OS::Cinder::Volume            | CREATE_COMPLETE |
| floating_ip_instance_heat   | b88cf79f-1a54-4971-9cd9-f2b7f50fa08f | OS::Neutron::FloatingIP       | CREATE_COMPLETE |
| association_reverseproxy    | 19                                   | OS::Nova::FloatingIPAssociation | CREATE_COMPLETE |
| instance_heat_security_group | 503bfcc6-892b-45d3-ad43-ea033e7a5ff4 | OS::Neutron::SecurityGroup    | UPDATE_COMPLETE |
+-----------------------------+--------------------------------------+-------------------------------+-----------------+
stack@microwave:~$ openstack security group show 503bfcc6-892b-45d3-ad43-ea033e7a5ff4
+-----------------+-------------------------------------------------------------------------------+
| Field           | Value                                                                         |
+-----------------+-------------------------------------------------------------------------------+
| created_at      | 2019-10-19T20:31:06Z                                                          |
| description     |                                                                               |
| id              | 503bfcc6-892b-45d3-ad43-ea033e7a5ff4                                          |
| name            | ma_pile_donnees-instance_heat_security_group-u2b577tcgwak                     |
| project_id      | 5cd4212611654857a9e941a7b5d35e13                                              |
| revision_number | 9                                                                             |
| rules           | direction='egress', ethertype='IPv4'                                          |
|                 | direction='egress', ethertype='IPv6'                                          |
|                 | direction='ingress', ethertype='IPv4',                                        |
|                 |                      protocol='tcp', port_range_max='22', port_range_min='22', |
|                 |                      remote_ip_prefix='0.0.0.0/0'                             |
|                 | direction='ingress', ethertype='IPv4',                                        |
|                 |                      protocol='tcp', port_range_max='80', port_range_min='80', |
|                 |                      remote_ip_prefix='0.0.0.0/0'                             |
|                 |                                                                               |
| tags            | []                                                                            |
| updated_at      | 2019-10-19T20:41:11Z                                                          |
+-----------------+-------------------------------------------------------------------------------+
```

Destruction d'une pile

Nous pouvons détruite une pile en ligne de commande, une confirmation est demandée car la destruction d'un pile peut provoquer la destruction d'un grand nombre de ressources, le paramètre « `-y` » permet d'outrepasser la confirmation. La destruction d'une pile est irréversible même si elle n'est pas immédiate :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack stack delete ma_pile_donnees --wait
Are you sure you want to delete this stack(s) [y/N]? y
2019-10-19 20:49:26Z [ma_pile_donnees]: DELETE_IN_PROGRESS  Stack DELETE started
2019-10-19 20:49:26Z [ma_pile_donnees.association_reverseproxy]: DELETE_IN_PROGRESS  state changed
2019-10-19 20:49:28Z [ma_pile_donnees.association_reverseproxy]: DELETE_COMPLETE  state changed
2019-10-19 20:49:28Z [ma_pile_donnees.floating_ip_instance_heat]: DELETE_IN_PROGRESS  state changed
2019-10-19 20:49:28Z [ma_pile_donnees.instance_heat]: DELETE_IN_PROGRESS  state changed
2019-10-19 20:49:32Z [ma_pile_donnees.floating_ip_instance_heat]: DELETE_COMPLETE  state changed
2019-10-19 20:49:38Z [ma_pile_donnees.instance_heat]: DELETE_COMPLETE  state changed
2019-10-19 20:49:38Z [ma_pile_donnees.instance_heat_volume]: DELETE_IN_PROGRESS  state changed
2019-10-19 20:49:38Z [ma_pile_donnees.instance_heat_security_group]: DELETE_IN_PROGRESS  state changed
2019-10-19 20:49:39Z [ma_pile_donnees.instance_heat_security_group]: DELETE_COMPLETE  state changed
2019-10-19 20:49:40Z [ma_pile_donnees.instance_heat_volume]: DELETE_COMPLETE  state changed
2019-10-19 20:49:40Z [ma_pile_donnees]: DELETE_COMPLETE  Stack DELETE completed successfully
stack@microwave:~$ openstack stack list
   ⇒ Vide
```

Lancement d'une pile avec un fichier de paramètres

Nous créons fichier de paramètres que nous appelons `env.yaml`, adaptez le fichier à votre environnement (nom de la clé et ID de votre volume de données) :

```
parameter_defaults:
    nom_de_cle: LFO
    id_volume_donnees: e4539f36-09b7-4198-afc4-6766c5116e22
```

Nous utilisons le paramètre « `-e` » qui permet de spécifier un fichier d'environnement, le paramètre « `--wait` » permet d'attendre la fin de la création de la pile et de suivre sa progression :

```
stack@microwave:~$ openstack stack create --wait                                    \
                                   -e env.yaml -t https://u03.fr/90openstack/volume_donnees.yaml \
                                   ma_pile_donnees
2019-10-20 12:07:08Z [ma_pile_donnees]: CREATE_IN_PROGRESS  Stack CREATE started
2019-10-20 12:07:08Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:07:09Z [ma_pile_donnees.instance_heat_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:07:10Z [ma_pile_donnees.instance_heat_security_group]: CREATE_COMPLETE  state changed
2019-10-20 12:07:10Z [ma_pile_donnees.instance_heat_volume]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:07:11Z [ma_pile_donnees.floating_ip_instance_heat]: CREATE_COMPLETE  state changed
2019-10-20 12:07:15Z [ma_pile_donnees.instance_heat_volume]: CREATE_COMPLETE  state changed
2019-10-20 12:07:15Z [ma_pile_donnees.instance_heat]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:07:42Z [ma_pile_donnees.instance_heat]: CREATE_COMPLETE  state changed
2019-10-20 12:07:42Z [ma_pile_donnees.association_reverseproxy]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:07:47Z [ma_pile_donnees.association_reverseproxy]: CREATE_COMPLETE  state changed
2019-10-20 12:07:47Z [ma_pile_donnees]: CREATE_COMPLETE  Stack CREATE completed successfully
+---------------------+--------------------------------------------------------------------------------+
| Field               | Value                                                                          |
+---------------------+--------------------------------------------------------------------------------+
| id                  | 1b8ac52a-51af-43de-b40b-532bd377eda3                                           |
| stack_name          | ma_pile_donnees                                                                |
| description         | Creation d'un volume systeme a partir d'une image et attachement a l'instance |
|                     |                                                                                |
| creation_time       | 2019-10-20T12:07:07Z                                                           |
| updated_time        | None                                                                           |
| stack_status        | CREATE_COMPLETE                                                                |
| stack_status_reason | Stack CREATE completed successfully                                            |
+---------------------+--------------------------------------------------------------------------------+
```

Le paramètre « `-c` » sur les commandes de type 'show' permet de n'afficher que les champs listés :

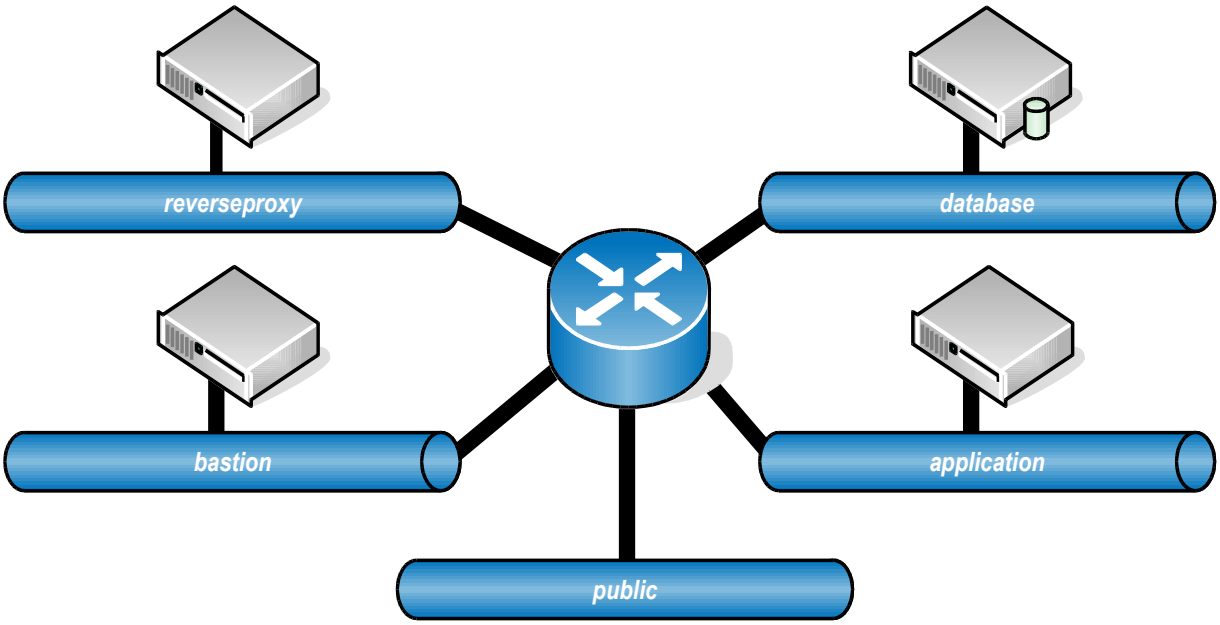
```
stack@microwave:~$ openstack stack show ma_pile_donnees -c outputs
+---------+-------------------------------------------------------+
| Field   | Value                                                 |
+---------+-------------------------------------------------------+
| outputs | - description: Adresse IP publique de l'instance Heat |
|         |   output_key: instance_heat_public_ip                 |
|         |   output_value: 172.24.4.58                           |
|         |                                                       |
+---------+-------------------------------------------------------+
stack@microwave:~$ ssh ubuntu@172.24.4.58
The authenticity of host '172.24.4.58 (172.24.4.58)' can't be established.
ECDSA key fingerprint is SHA256:V+SovoRNlIpMGsXR3UxlaSRxAtsE7uBOUtsCyFuGwYY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.24.4.58' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

ubuntu@ma-pile-donnees-instance-heat-c3odmlldiekq:~$ cat /DONNEES/hello.txt
Hello
```

Création d'infrastructures réseau

Nous avons dit que l'infrastructure réseau est constituée de :

- réseaux
- sous-réseaux
- routeurs
- ports (qui sont généralement créés de façon implicite, sur les serveurs, les routeurs…)

Ci-dessous une architecture réseau typique d'une application 'Cloud' avec 4 réseaux :



La pile complète est disponible à cette adresse :

- https://u03.fr/90openstack/4_reseaux.yaml

Il va nous falloir deux adresses IP flottantes (une pour le bastion SSH de votre tenant, et une pour le reverse-proxy). Nous avons normalement une adresse IP allouée mais elle est déjà associée à un serveur, nous en allouons deux nouvelles :

```
stack@microwave:~$ openstack floating ip list -c ID -c 'Floating IP Address' -c 'Fixed IP Address'
+--------------------------------------+---------------------+------------------+
| ID                                   | Floating IP Address | Fixed IP Address |
+--------------------------------------+---------------------+------------------+
| 168bd864-8489-4d50-96dd-24bd93f316e4 | 172.24.4.105        | 10.0.0.43        |
+--------------------------------------+---------------------+------------------+
stack@microwave:~$ openstack floating ip create public -c 'floating_ip_address' -c id
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| floating_ip_address | 172.24.4.239                         |
| id                  | 3fe5d827-a1aa-47d9-a73f-cb22cdaecc27 |
+---------------------+--------------------------------------+
stack@microwave:~$ openstack floating ip create public -c 'floating_ip_address' -c id
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| floating_ip_address | 172.24.4.123                         |
| id                  | 311cbd22-c4f4-4190-b342-d0d233906ccc |
+---------------------+--------------------------------------+
stack@microwave:~$ openstack floating ip list -c ID -c 'Floating IP Address' -c 'Fixed IP Address'
```

```
+--------------------------------------+---------------------+------------------+
| ID                                   | Floating IP Address | Fixed IP Address |
+--------------------------------------+---------------------+------------------+
| 168bd864-8489-4d50-96dd-24bd93f316e4 | 172.24.4.105        | 10.0.0.43        |
| 311cbd22-c4f4-4190-b342-d0d233906ccc | 172.24.4.123        | None             |
| 3fe5d827-a1aa-47d9-a73f-cb22cdaecc27 | 172.24.4.239        | None             |
+--------------------------------------+---------------------+------------------+
```

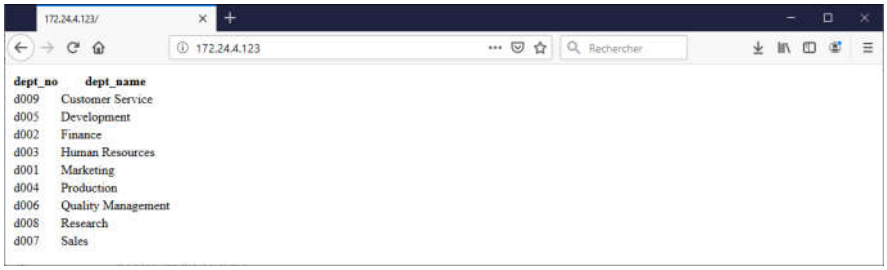
Nous pouvons désormais instancier notre pile :

```
stack@microwave:~$ openstack stack create --wait                                                                \
                                          --parameter nom_de_cle="LFO"                                          \
                                          --parameter id_floating_ip_bastion="3fe5d827-a1aa-47d9-a73f-cb22cdaecc27"      \
                                          --parameter id_floating_ip_reverseproxy="311cbd22-c4f4-4190-b342-d0d233906ccc" \
                                          -t https://u03.fr/90openstack/4_reseaux.yaml          \
                                          ma_pile_4_reseaux
2019-10-20 12:46:39Z [ma_pile_4_reseaux]: CREATE_IN_PROGRESS  Stack CREATE started
2019-10-20 12:46:40Z [ma_pile_4_reseaux.instance_reverseproxy_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:41Z [ma_pile_4_reseaux.instance_reverseproxy_security_group]: CREATE_COMPLETE  state changed
2019-10-20 12:46:41Z [ma_pile_4_reseaux.reverseproxy_network]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:42Z [ma_pile_4_reseaux.reverseproxy_network]: CREATE_COMPLETE  state changed
2019-10-20 12:46:42Z [ma_pile_4_reseaux.router]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:42Z [ma_pile_4_reseaux.reverseproxy_subnet]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:43Z [ma_pile_4_reseaux.instance_bastion_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:44Z [ma_pile_4_reseaux.instance_database_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:45Z [ma_pile_4_reseaux.reverseproxy_subnet]: CREATE_COMPLETE  state changed
2019-10-20 12:46:45Z [ma_pile_4_reseaux.instance_bastion_volume]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:46Z [ma_pile_4_reseaux.instance_database_security_group]: CREATE_COMPLETE  state changed
2019-10-20 12:46:46Z [ma_pile_4_reseaux.instance_application_security_group]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:47Z [ma_pile_4_reseaux.database_network]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:46:48Z [ma_pile_4_reseaux.instance_reverseproxy_volume]: CREATE_IN_PROGRESS  state changed
                  …/…
2019-10-20 12:50:02Z [ma_pile_4_reseaux.association_bastion]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:50:05Z [ma_pile_4_reseaux.instance_reverseproxy]: CREATE_COMPLETE  state changed
2019-10-20 12:50:06Z [ma_pile_4_reseaux.association_reverseproxy]: CREATE_IN_PROGRESS  state changed
2019-10-20 12:50:28Z [ma_pile_4_reseaux.association_bastion]: CREATE_COMPLETE  state changed
2019-10-20 12:50:42Z [ma_pile_4_reseaux.association_reverseproxy]: CREATE_COMPLETE  state changed
2019-10-20 12:50:42Z [ma_pile_4_reseaux]: CREATE_COMPLETE  Stack CREATE completed successfully
+---------------------+--------------------------------------------------------------------------------+
| Field               | Value                                                                          |
+---------------------+--------------------------------------------------------------------------------+
| id                  | 44cf5b31-f2a5-4e52-9c6e-3d4bc4e765ce                                           |
| stack_name          | ma_pile_4_reseaux                                                              |
| description         | Template HEAT pour obtenir une infra apache + PHP + MariaDB.                  |
|                     | Nous construisons 4 instances de serveur:                                      |
|                     |   - Une instance bastion SSH                                                   |
                …/…
|                     | En pré-requis vous devez disposer des ID OpenStack des éléments suivants:      |
|                     |   - L'adresse IP flottante qui sera attachée au reverse-proxy                  |
|                     |   - L'adresse IP flottante qui sera attachée au basion SSH                     |
|                     |                                                                                |
|                     |                                                                                |
| creation_time       | 2019-10-20T12:46:38Z                                                           |
| updated_time        | None                                                                           |
| stack_status        | CREATE_COMPLETE                                                                |
| stack_status_reason | Stack CREATE completed successfully                                            |
+---------------------+--------------------------------------------------------------------------------+
```

Attention même si la pile est finie de créer il peut falloir plusieurs minutes pour que les différents packages s'installent sur les différentes instances, cela dépend de la puissance de votre machine DevStack.

A l'issue de la création de la pile (et de l'initialisation de scripts sur les différents serveurs) si vous tapez l'adresse IP du reverse-proxy dans votre navigateur web la page suivante, ceci signifie que vous avez atteint le reverse-proxy, que celui-ci a pu relayer votre requête au serveur d'application et que le serveur d'application a interrogé le serveur de bases de données :

```
stack@microwave:~$ openstack server list -c Name -c Networks
+------------------+-----------------------------------------------------------------------------+
| Name             | Networks                                                                    |
+------------------+-----------------------------------------------------------------------------+
| reverseproxy     | reverseproxy=10.99.2.194, 172.24.4.123                                      |
| bastion          | bastion=10.99.1.125, 172.24.4.239                                           |
| application      | application=10.99.3.13                                                      |
| database         | database=10.99.4.13                                                         |
| ubuntu-18.04_inst | private=fdac:50a1:b51d:0:f816:3eff:fec8:2ed5, 10.0.0.43, 172.24.4.105      |
+------------------+-----------------------------------------------------------------------------+
```



Création des réseaux et sous-réseaux

Le réseau 'public' est celui fourni par le Cloud, nous allons créer les 4 autres, chacun des réseaux est composé d'au moins un sous-réseau, pour chacun des sous-réseaux il faut choisir les paramètres suivants :

- Plage d'adresses IP (notation CIDR)
- Plage d'adresses IP pour attribuer dynamiquement les adresses IP en DHCP (notation : adresses IP début et fin)
- Adresse IP de la passerelle (cette adresse sera utilisée par le routeur lors de l'attachement du sous-réseau au routeur)
- Le ou les serveurs DNS qui seront utilisés par les serveurs connectés au sous-réseau

Dans le choix des adresses IP il faut garder à l'esprit qu'il s'agit de réseaux privés propres au projet, elles ne doivent être uniques qu'à l'intérieur du projet mais il faut éviter qu'elles entrent en collision avec des réseaux utilisés par les utilisateurs de votre application.

Pour chacun des réseaux nous commençons par créer le réseau (objet `OS::Neutron::Net`) puis le sous-réseau (`OS::Neutron::Subnet`).

La liaison entre le sous-réseau et le réseau se fait par la propriété `network_id` du sous-réseau, en utilisant `get_ressource` pour obtenir l'ID du réseau.

La propriété `allocation_pools` donne la ou les plages d'allocation par le serveur DHCP, la propriété `cidr` donne la plage d'adresses du réseau, `gateway_ip` est l'adresse IP choisie pour être celle de la passerelle par défaut du réseau.

```
  bastion_network:
    type: OS::Neutron::Net
    properties:
      name: bastion

  bastion_subnet:
    type: OS::Neutron::Subnet
    properties:
      network_id: { get_resource: bastion_network }
      allocation_pools:
      - {start: 10.99.1.100, end: 10.99.1.250}
      cidr: 10.99.1.0/24
      dns_nameservers: { get_param: dns }
      gatewayCréation du routeur
```

Nous créons le routeur et nous le rattachons au réseau public (`external_gateway_info`), ainsi le routeur routera vers le réseau public tout le trafic qui n'est pas destiné à un des réseau qu'il gère, il fera également de la translation d'adresse pour masquer les adresses privées mais surtout présenter une adresse routable sur le réseau public :

```
  router:
    type: OS::Neutron::Router
    properties:
      name: routeur
      external_gateway_info:
        network: { get_param: public_network }
```

Nous voyons ci-dessous l'adresse IP publique qui sera utilisée par le routeur pour faire de la translation d'adresse (`python -m json.tool` permet de mettre en forme du format JSON) :

```
stack@microwave:~$ openstack router show routeur -f json -c external_gateway_info | python -m json.tool
{
    "external_gateway_info": {
        "enable_snat": true,
        "external_fixed_ips": [
            {
                "ip_address": "172.24.4.18",
                "subnet_id": "b3ee49a7-80b9-4898-8f53-a58cb1e3908c"
            },
            {
                "ip_address": "2001:db8::2a2",
                "subnet_id": "d9a81841-daa0-4f6f-8cde-75de6fb2a343"
            }
        ],
        "network_id": "12ceb0b7-9ed6-4bfd-9160-ab44b4f868b2"
    }
}
```

Les interfaces sont créées comme ceci pour connecter les réseaux au routeur : :

```
  router_interface_bastion:
    type: OS::Neutron::RouterInterface
    properties:
      router_id: { get_resource: router }
      subnet: { get_resource: bastion_subnet }
```

Nous voyons ci-dessous les informations qui correspondent aux différentes interfaces du réseau une fois que le pile a été créée, nous retrouvons nos 4 réseaux et les 4 adresses que nous avons attribué à chacune des passerelles (`python -m json.tool` permet de mettre en forme du format JSON) :

```
stack@microwave:~$ openstack router show routeur -f json -c interfaces_info | python -m json.tool
{
    "interfaces_info": [
        {
            "ip_address": "10.99.1.254",
            "port_id": "0e31cacf-2fca-4cb8-9feb-19c8cc874eac",
            "subnet_id": "9997f78d-f7d5-45ce-b02f-5c42970c988d"
        },
        {
            "ip_address": "10.99.3.254",
            "port_id": "85101556-2d35-457e-921b-756ddbc87db1",
            "subnet_id": "c5e7e43e-66a3-4f00-a7b2-766496e7dc64"
        },
        {
            "ip_address": "10.99.4.254",
            "port_id": "87fe7141-457b-4cf5-aab2-92568f5ab1dc",
            "subnet_id": "e9611ee6-abbe-4310-92a2-eb9ccae23951"
        },
        {
            "ip_address": "10.99.2.254",
            "port_id": "ad4b5fda-134e-425e-b49e-f3b63ce5098f",
            "subnet_id": "bb3f4557-2c9a-467b-b237-3ba6b7458223"
        }
    ]
}
```

Attachement du serveur au réseau

La propriété `networks` est un tableau de réseaux auxquels doit être attaché le serveur. Il est possible au passage de fixer l'adresse IP du serveur, si `fixed_ip` n'est pas spécifié alors une adresse sera automatiquement allouée à partir des plages d'adresses DHCP des sous-réseaux concernés. :

```
  instance_database:
    properties:
      networks:
        - network: { get_resource: database_network }
          fixed_ip: 10.99.4.13
```

Techniquement le serveur est configuré en DHCP, et donc quand une adresse IP est fixée il s'agit d'une réservation DHCP effectuée au niveau de Neutron, et non d'une adresse IP configurée de façon statique dans les fichiers de configuration de l'interface.

Si vous fixez une adresse IP il vous est conseillé de ne pas le faire dans la plage d'adresse que vous avez assigné au DHCP lors de la création du sous-réseau (`allocation_pools`).

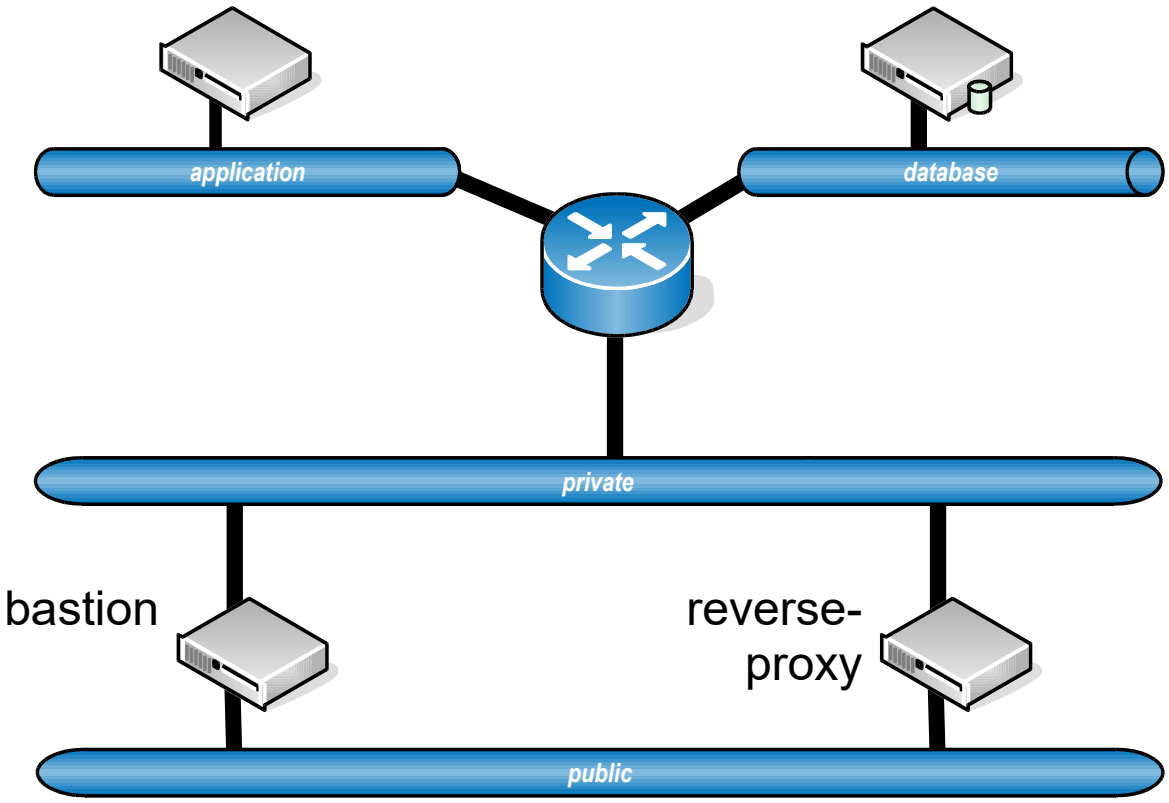
Lorsque que nous regardons les paramètres réseau d'un de nos serveurs nous retrouvons les propriétés que nous avons définies pour notre sous-réseau :

```
ubuntu@bastion:~$ ip addr
   [.../…]
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc pfifo_fast state UP group default qlen 1000
    link/ether fa:16:3e:04:bb:32 brd ff:ff:ff:ff:ff:ff
    inet 10.99.1.13/24 brd 10.99.1.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe04:bb32/64 scope link
       valid_lft forever preferred_lft forever

ubuntu@bastion:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.99.1.254     0.0.0.0         UG        0 0          0 ens3
10.99.1.0       0.0.0.0         255.255.255.0   U         0 0          0 ens3

ubuntu@bastion:~$ nslookup u03.fr
Server:         8.8.8.8
Address:        8.8.8.8#53
Non-authoritative answer:
Name:   u03.fr
Address: 37.187.126.87
```

Attachement à plusieurs réseaux

Il est possible de spécifier plusieurs réseaux en spécifiant plusieurs entrées `network` sous la propriété `networks`.

C'est par exemple utilisable pour un bastion ou un reverse-proxy afin d'isoler le réseau public et le réseau privé :


application
database
private
bastion
reverse-
proxy
public

Le stockage d'objets avec Swift

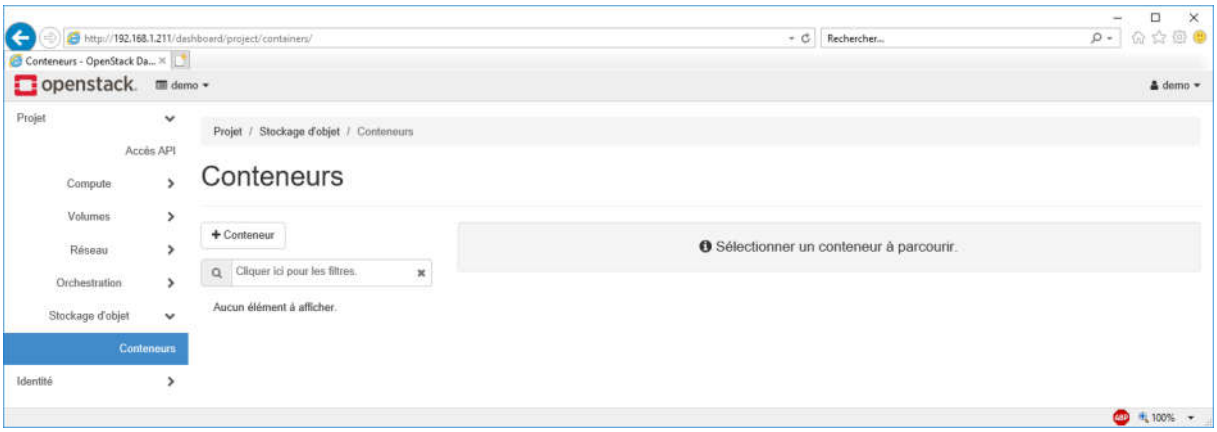
Swift est un composant de OpenStack qui permet de stocker des objets (fichiers) qui seront accessibles en HTTP/HTTPS :

- de façon publique sans authentification
- à des utilisateurs authentifiés par Keystone et autorisés
- à des utilisateurs disposant d'une URL temporaire donnant des droits spécifiques pour une période de temps limitée.

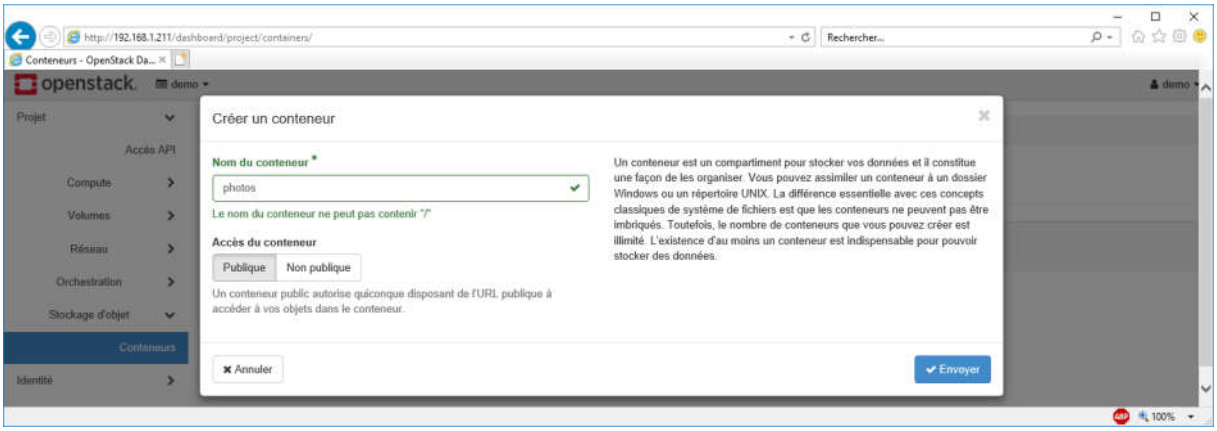
Chaque projet peut créer des containers qui vont permettre de stocker des objets, à l'intérieur de chaque container il n'y a pas de réelle structure arborescente, les objets sont donc tous stockés au même niveau.

Création d'un conteneur Swift et chargement d'objets depuis le Dashboard

Le Dashboard permet de gérer les conteneurs Swift (création, chargement de fichiers, destruction).



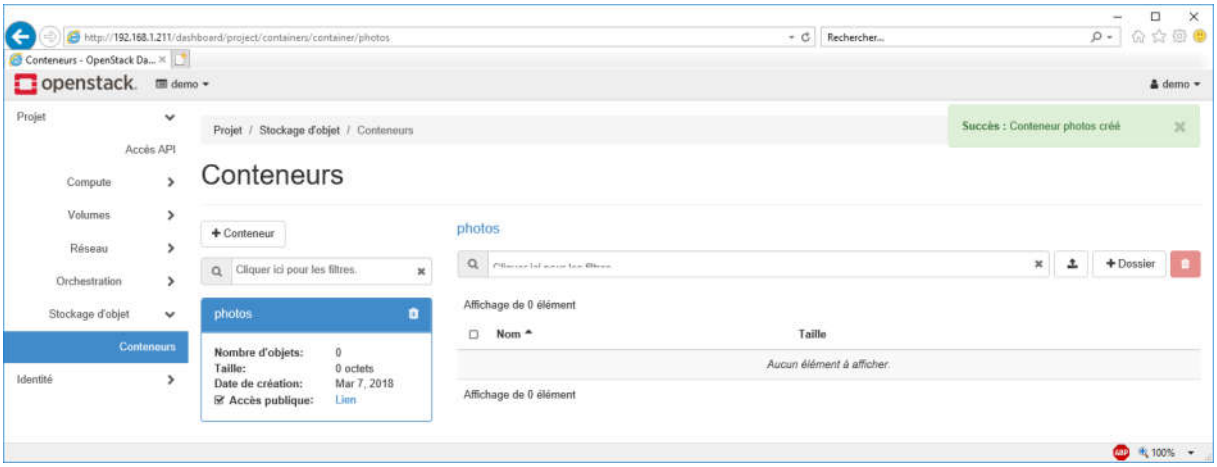
En cliquant sur le bouton « +Conteneur » nous affichons l'écran de création de conteneur, nous l'appelons « photos », nous déclarons le conteneur public.



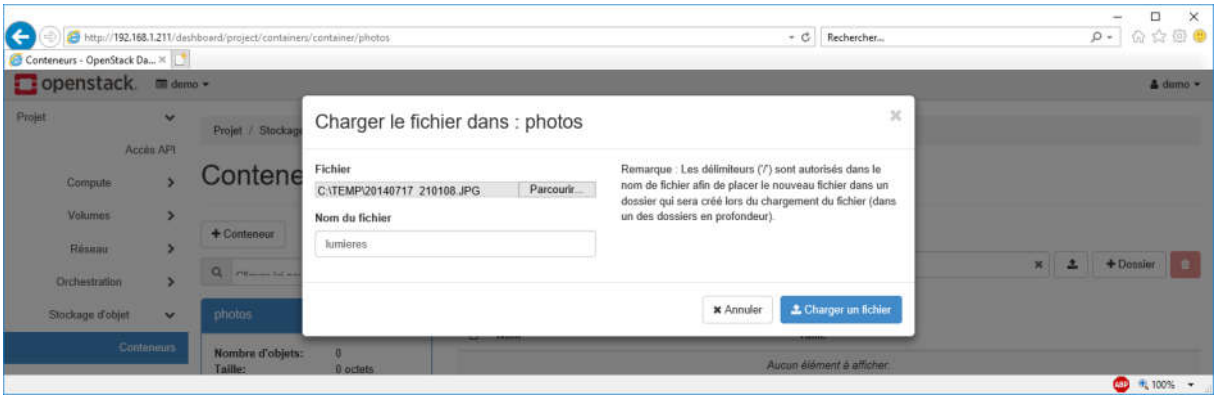
Swift ne gère pas les répertoires mais le Dashboard permet de simuler la création d'un répertoire (bouton « +Dossier »), ceci est simulé en chargeant un fichier à 0 octets avec le nom du pseudo-dossier.

Il vous est laissé à titre d'exercice à la fin du chapitre la création et l'utilisation de répertoires depuis le Dashboard.

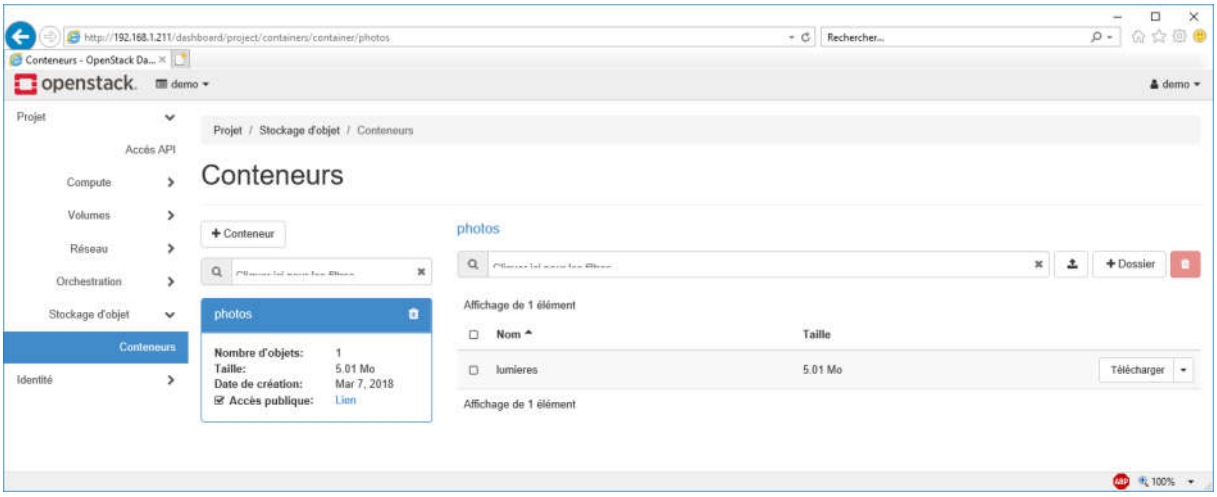
Le conteneur est en accès public, il se présente sous la forme d'un répertoire accessible en http, il est possible d'obtenir l'URL du conteneur en cliquant avec le bouton de droite sur « Lien » en face de « Accès public » :



Le bouton « ⬆ » permet de télécharger un fichier dans le conteneur en cours, en cliquant sur « Parcourir » nous sélectionnons le fichier sur notre poste local, nous rentrons le nom qu'aura le fichier dans Swift :



Une fois chargé le fichier apparaît dans la liste, il est possible de le télécharger, d'obtenir les informations sur le fichier ou de le supprimer :

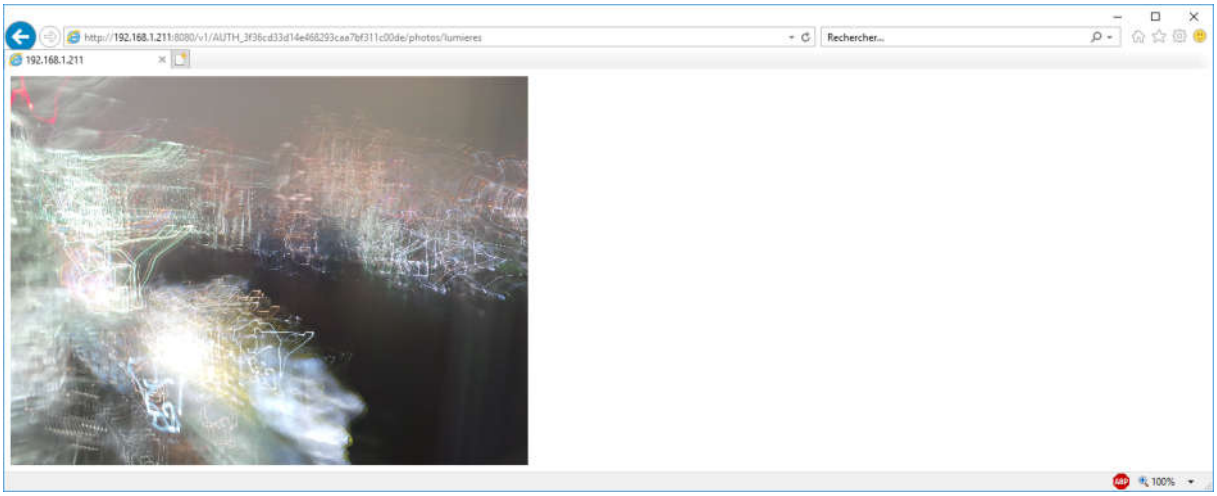


Le lien obtenu pour le conteneur est:

- http://192.168.1.211:8080/v1/AUTH_3f36cd33d14e468293caa7bf311c00de/photos

Le nom du fichier donné lors du chargement est '`lumieres`', l'URL du fichier est donc :

- http://192.168.1.211:8080/v1/AUTH_3f36cd33d14e468293caa7bf311c00de/photos/lumieres



Manipulation des conteneurs et objets en ligne de commande

Il est possible de lister les conteneurs et leur contenu à l'aide de l'utilitaire en ligne de commande :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack container list
+---------+
| Name    |
+---------+
| photos  |
+---------+
stack@microwave:~$ openstack container show photos
+--------------+----------------------------------------+
| Field        | Value                                  |
+--------------+----------------------------------------+
| account      | AUTH_3f36cd33d14e468293caa7bf311c00de |
| bytes_used   | 5249087                                |
| container    | photos                                 |
| object_count | 1                                      |
| read_acl     | .r:*,.rlistings                        |
+--------------+----------------------------------------+
```

Il est également possible de lister le contenu des conteneurs ainsi que les informations sur un objet à l'aide de l'utilitaire en ligne de commande :

```
stack@microwave:~$ openstack object list photos
+----------+
| Name     |
+----------+
| lumieres |
+----------+
stack@microwave:~$ openstack object show photos lumieres
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de |
| container      | photos                               |
| content-length | 5249087                              |
| content-type   | application/octet-stream             |
| etag           | f66ca1814ff66b249ee973b01139207e     |
| last-modified  | Wed, 07 Mar 2018 21:10:44 GMT        |
| object         | lumieres                             |
| properties     | Orig-Filename='20140717_210108.JPG'  |
+----------------+--------------------------------------+
```

Stockage et récupération d'un fichier

La commande « `openstack object create` » permet charger un objet dans Swift, on note que le champ « `etag` » est en fait le hash md5 du fichier :

```
stack@microwave:~$ openstack object create --name takeshiba.jpg photos /tmp/IMG_20140718_015943.jpg
+--------------+-----------+----------------------------------+
| object       | container | etag                             |
+--------------+-----------+----------------------------------+
| takeshiba.jpg | photos    | 584175f6b846a13f34477d08d1ed9a0c |
+--------------+-----------+----------------------------------+
stack@microwave:~$ openstack object show photos takeshiba.jpg
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de |
| container      | photos                               |
| content-length | 2357220                              |
| content-type   | image/jpeg                           |
| etag           | 584175f6b846a13f34477d08d1ed9a0c     |
| last-modified  | Sat, 10 Mar 2018 21:27:25 GMT        |
| object         | takeshiba.jpg                        |
+----------------+--------------------------------------+
stack@microwave:~$ md5sum /tmp/IMG_20140718_015943.jpg
584175f6b846a13f34477d08d1ed9a0c  /tmp/IMG_20140718_015943.jpg
```

La commande « `openstack object save` » permet télécharger un objet présent dans Swift, on note que par défaut la commande sauve l'objet avec le nom sous lequel il a été stocké dans Swift, il est possible d'utiliser le paramètre « `--file` » pour spécifier le nom (et le chemin) du fichier sous lequel l'objet doit être sauvegardé localement :

```
stack@microwave:~$ openstack object save photos lumieres
stack@microwave:~$ file lumieres
lumieres: JPEG image data, Exif standard: [TIFF image data, little-endian, direntries=12, height=3456, manufacturer=SAMSUNG,
model=EK-GC100, orientation=upper-left, xresolution=208, yresolution=216, resolutionunit=2, software=GC100XXBMC4,
datetime=2014:07:17 21:01:07, width=4608], baseline, precision 8, 4608x3456, frames 3
stack@microwave:~$ openstack object save photos lumieres --file /tmp/lumieres.jpg
stack@microwave:~$ ls -l /tmp/lumieres.jpg
-rw-rw-r-- 1 stack stack 5249087 Mar 10 22:36 /tmp/lumieres.jpg
```

Ajout de propriétés

Lors de l'ajout de l'objet via le Dashboard une propriété avait été ajoutée automatiquement (le nom réel du fichier initial), ce que n'a pas fait la commande « `openstack object create` » :

```
stack@microwave:~$ openstack object show photos lumieres
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de |
| container      | photos                               |
| content-length | 5249087                              |
| content-type   | application/octet-stream             |
```

```
| etag           | f66ca1814ff66b249ee973b01139207e      |
| last-modified  | Wed, 07 Mar 2018 21:10:44 GMT         |
| object         | lumieres                              |
| properties     | Orig-Filename='20140717_210108.JPG'   |
+----------------+---------------------------------------+
stack@microwave:~$ openstack object show photos takeshiba.jpg
+----------------+---------------------------------------+
| Field          | Value                                 |
+----------------+---------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de |
| container      | photos                                |
| content-length | 2357220                               |
| content-type   | image/jpeg                            |
| etag           | 584175f6b846a13f34477d08d1ed9a0c      |
| last-modified  | Sat, 10 Mar 2018 21:27:25 GMT         |
| object         | takeshiba.jpg                         |
+----------------+---------------------------------------+
```

Il est possible d'ajouter les propriétés de son choix à un objet sous la forme de paires « `clé=valeur` » :

```
stack@microwave:~$ openstack object set photos takeshiba.jpg --property annee=2014 \
                                                             --property categorie=vacances \
                                                             --property lieu='Tokyo'
stack@microwave:~$ openstack object show photos takeshiba.jpg
+----------------+-----------------------------------------------------+
| Field          | Value                                               |
+----------------+-----------------------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de                |
| container      | photos                                              |
| content-length | 2357220                                             |
| content-type   | image/jpeg                                          |
| etag           | 584175f6b846a13f34477d08d1ed9a0c                    |
| last-modified  | Sat, 10 Mar 2018 21:55:17 GMT                       |
| object         | takeshiba.jpg                                       |
| properties     | Annee='2014', Categorie='vacances', Lieu='Tokyo'    |
+----------------+-----------------------------------------------------+
```

La commande « `set` » efface les propriétés déjà présentes sur un objet, supposons que nous désirions modifier la propriété '`annee`' pour mettre '`2015`', il ne suffit pas de spécifier la propriété '`annee`', sinon toutes les autres propriétés seront supprimées.

```
stack@microwave:~$ openstack object set photos takeshiba.jpg --property annee=2015
stack@microwave:~$ openstack object show photos takeshiba.jpg
+----------------+---------------------------------------+
| Field          | Value                                 |
+----------------+---------------------------------------+
| account        | AUTH_3f36cd33d14e468293caa7bf311c00de |
| container      | photos                                |
| content-length | 2357220                               |
| content-type   | image/jpeg                            |
| etag           | 584175f6b846a13f34477d08d1ed9a0c      |
| last-modified  | Sat, 10 Mar 2018 21:56:52 GMT         |
| object         | takeshiba.jpg                         |
| properties     | Annee='2015'                          |
+----------------+---------------------------------------+
```

La commande « `unset` » permet d'effacer une ou plusieurs propriétés, il vous est laissé à titre d'exercice l'ajout et la suppression de propriétés sur un objet présent dans Swift.

Terraform

On l'a vu, les API OpenStack sont très puissantes, elles sont utilisables dans la plupart des langages de programmation et donc de nombreux logiciels ont été développés pour tirer partie de OpenStack et de ses API.

Terraform permet de faire de l'orchestration comme Heat, il permet d'utiliser de nombreuses solutions de Cloud, dont OpenStack :

- https://www.terraform.io/docs/providers/openstack/

Même si le format des fichiers est différent entre Heat et Terraform la logique est la même, en effet Heat et Terraform sont tributaires des API OpenStack et de la façon dont elles s'articulent.

Les descriptions d'infrastructures dans Terraform s'appellent '*configurations*'.

Appliquer la configuration (`terraform apply`) va effectuer les actions nécessaires pour réaliser l'infrastructure décrite dans la configuration, soit en la construisant (lors du premier appel à `terraform apply`), soit en mettant à jour l'infrastructure (lors appels suivants à `terraform apply`). Pour ceci Terraform va bâtir un '*plan*' des opérations à réaliser.

Un appel à `terraform destroy` va permettre de détruire l'infrastructure construite.

Installation et initialisation

Terraform est téléchargeable à cette adresse : https://www.terraform.io/downloads.html

Terraform est disponible pour Windows et Linux ('i386', 'amd64' et processeurs 'arm'). Il se présente sous la forme d'un fichier '.zip' contenant un unique programme exécutable en commande ligne.

Les processeurs 'arm' sont utilisés sur la plupart des téléphones portables, des tablettes, de certains équipements réseaux et ordinateurs 'monocarte' tels que les Raspberry Pi

Dans cette exemple nous utilisons un Raspberry Pi (qui fait partie de mes machines de tests), il s'agit d'une mini machine qui coûte environ 40 euros.

https://blog.u03.fr/la-paillasse-de-linformaticien/

Téléchargez la version qui correspond à votre plateforme, dézippez le fichier et placez-le dans un répertoire vide que vous appelez '`tf`' (par exemple).

```
foucher@raspberrypi:~/tf $ wget https://releases.hashicorp.com/terraform/0.12.12/terraform_0.12.12_linux_arm.zip
--2019-10-20 14:36:39--  https://releases.hashicorp.com/terraform/0.12.12/terraform_0.12.12_linux_arm.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 151.101.1.183, 151.101.65.183, 151.101.129.183, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|151.101.1.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15115877 (14M) [application/zip]
Saving to: 'terraform_0.12.12_linux_arm.zip'

terraform_0.12.12_linux_arm.zip                   100%
[=================================================================================>]  14.42M   790KB/s    in 19s

2019-10-20 14:36:59 (785 KB/s) - 'terraform_0.12.12_linux_arm.zip' saved [15115877/15115877]

foucher@raspberrypi:~/tf $ unzip terraform_0.12.12_linux_arm.zip
Archive:  terraform_0.12.12_linux_arm.zip
```

```
  inflating: terraform
```

Dans le répertoire ‘tf’ créez un fichier que vous appellerez ‘provider.tf’ (par exemple, mais le nom doit finir par ‘.tf’) :

```
provider "openstack" {
  user_name   = "demo"
  tenant_name = "demo"
  password    = "topsecret"
  auth_url    = "http://192.168.1.211/identity"
  region      = "RegionOne"
}
```

Lancez la commande ‘terraform init’, Terraform va charger tous les fichiers ‘.tf’ présents dans le répertoire (ici uniquement le fichier ‘provider.tf’) et il va télécharger depuis le site de l’éditeur le plugin nécessaire pour utiliser OpenStack :

```
foucher@raspberrypi:~/tf $ ./terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "openstack" (terraform-providers/openstack) 1.23.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.openstack: version = "~> 1.23"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Éléments d’une configuration

Nous avons déjà vu le provider, il permet à Terraform de récupérer le bon plugin et permet de lui fournir les paramètres associés (les mêmes que dans les variables d’environnement positionnées par le script openrc).

Terraform est multi-provider, il permet par exemple de déployer à la fois de l’OpenStack, de l’AWS et du VMware…

Dans Terraform les variables (variable) sont l’équivalent des paramètres dans Heat, elle permettent d’adapter la configuration à l’environnement dans lequel elle est déployée.

Les ressources (resource) sont les éléments de l’infrastructure, comme dans Heat.

Les sources de données (data) sont des constructions syntaxiques qui permettent de récupérer l’ID unique d’éléments préexistants dans OpenStack à partir de différents paramètres (par exemple récupérer l’ID d’une image pour construire un volume de boot...)

Une première configuration

Nous allons faire une première configuration qui classiquement va créer une instance à partir d'une image.

Nous créons un fichier que nous appelons `premiere_configuration.tf` qui est également disponible à l'adresse https://u03.fr/90openstack/premiere_configuration.tf :

```
variable "nom_de_cle" {
  description = "Nom de la clé publique SSH permettant d'accéder aux serveurs"
}

variable "gabarit_instance" {
  description = "Gabarit qui sera utilisé par l'instance"
  default     = "ds1G"
}

variable "image_instance" {
  description = "Nom de l'image"
  default     = "ubuntu-18.04"
}

variable "reseau_prive" {
  description = "Nom du reseau prive"
  default     = "private"
}

resource "openstack_compute_instance_v2" "instance_terraform" {
  name            = "instance_terraform"
  flavor_name     = "${var.gabarit_instance}"
  key_pair        = "${var.nom_de_cle}"
  image_name      = "${var.image_instance}"
  network {
    name = "${var.reseau_prive}"
  }
}
```

Vous avons 4 éléments `variable` qui correspondent à nos paramètres, il y a une description et il est possible de mettre une valeur par défaut.

La valeur des paramètres est récupérée par la construction syntaxique suivante :

- `"${var.image_instance}"`

Comme dans Heat il est possible de passer un fichier avec la valeur des paramètres. Nous créons le fichier `terraform.tfvars` avec la valeur de nos éléments `variable` (nous conservons la valeur par défaut pour les variables `reseau_prive` et `image_instance`) :

```
nom_de_cle       =  "LFO"
gabarit_instance = "ds1G"
```

La commande `terraform plan` nous permet de visualiser les opérations nécessaires à la construction de l'infrastructure décrite dans la configuration :

```
foucher@raspberrypi:~/tf $ ./terraform plan
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
------------------------------------------------------------------------
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # openstack_compute_instance_v2.instance_terraform will be created
  + resource "openstack_compute_instance_v2" "instance_terraform" {
      + access_ip_v4        = (known after apply)
      + access_ip_v6        = (known after apply)
      + all_metadata        = (known after apply)
      + availability_zone   = (known after apply)
      + flavor_id           = (known after apply)
      + flavor_name         = "ds1G"
      + force_delete        = false
      + id                  = (known after apply)
```

```
      + image_id            = (known after apply)
      + image_name          = "ubuntu-18.04"
      + key_pair            = "LFO"
      + name                = "instance_terraform"
      + power_state         = "active"
      + region              = (known after apply)
      + security_groups     = (known after apply)
      + stop_before_destroy = false

      + network {
          + access_network = false
          + fixed_ip_v4    = (known after apply)
          + fixed_ip_v6    = (known after apply)
          + floating_ip    = (known after apply)
          + mac            = (known after apply)
          + name           = "private"
          + port           = (known after apply)
          + uuid           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

La commande `terraform apply` va déployer la configuration, une confirmation est demandée :

```
foucher@raspberrypi:~/tf $ ./terraform apply

An execution plan has been generated and is shown below.

   […/…] Rappel du plan

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

openstack_compute_instance_v2.instance_terraform: Creating...
openstack_compute_instance_v2.instance_terraform: Still creating... [10s elapsed]
openstack_compute_instance_v2.instance_terraform: Creation complete after 19s [id=c8e4a067-d449-42ba-b5cf-ada95f2ff248]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Nous retrouvons notre serveur en utilisant la commande CLI `openstack` :

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ openstack server show instance_terraform
stack@microwave:~$ openstack server show instance_terraform
+------------------------------+------------------------------------------------------------+
| Field                        | Value                                                      |
+------------------------------+------------------------------------------------------------+
| OS-DCF:diskConfig            | MANUAL                                                     |
| OS-EXT-AZ:availability_zone | nova                                                       |
| OS-EXT-STS:power_state       | Running                                                    |
| OS-EXT-STS:task_state        | None                                                       |
| OS-EXT-STS:vm_state          | active                                                     |
| OS-SRV-USG:launched_at       | 2019-10-20T15:58:54.000000                                 |
| OS-SRV-USG:terminated_at     | None                                                       |
| accessIPv4                   |                                                            |
| accessIPv6                   |                                                            |
| addresses                    | private=fdac:50a1:b51d:0:f816:3eff:fe23:e552, 10.0.0.24    |
| config_drive                 |                                                            |
| created                      | 2019-10-20T15:58:42Z                                       |
| flavor                       | ds1G (d2)                                                  |
| hostId                       | b3ce8aa16af22651ac4ab1291280fd727ff9c016c9068dcfcc861a75   |
| id                           | c8e4a067-d449-42ba-b5cf-ada95f2ff248                       |
| image                        | ubuntu-18.04 (7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5)        |
| key_name                     | LFO                                                        |
| name                         | instance_terraform                                         |
| progress                     | 0                                                          |
| project_id                   | 5cd4212611654857a9e941a7b5d35e13                           |
| properties                   |                                                            |
| security_groups              | name='default'                                             |
| status                       | ACTIVE                                                     |
| updated                      | 2019-10-20T15:58:55Z                                       |
| user_id                      | 57d02b0938d149a1aac96ce6757a3a14                           |
| volumes_attached             |                                                            |
+------------------------------+------------------------------------------------------------+
```

La commande `terraform destroy` va permettre de détruire l'infrastructure :

```
foucher@raspberrypi:~/tf $ ./terraform destroy
openstack_compute_instance_v2.instance_terraform: Refreshing state... [id=c8e4a067-d449-42ba-b5cf-ada95f2ff248]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # openstack_compute_instance_v2.instance_terraform will be destroyed
  - resource "openstack_compute_instance_v2" "instance_terraform" {
      - access_ip_v4        = "10.0.0.24" -> null
      - access_ip_v6        = "[fdac:50a1:b51d:0:f816:3eff:fe23:e552]" -> null
      - all_metadata        = {} -> null
      - availability_zone   = "nova" -> null
      - flavor_id           = "d2" -> null
      - flavor_name         = "ds1G" -> null
      - force_delete        = false -> null
      - id                  = "c8e4a067-d449-42ba-b5cf-ada95f2ff248" -> null
      - image_id            = "7ab813dd-6937-4f96-8cd1-ed0ec64e8ad5" -> null
      - image_name          = "ubuntu-18.04" -> null
      - key_pair            = "LFO" -> null
      - name                = "instance_terraform" -> null
      - power_state         = "active" -> null
      - region              = "RegionOne" -> null
      - security_groups     = [
          - "default",
        ] -> null
      - stop_before_destroy = false -> null

      - network {
          - access_network = false -> null
          - fixed_ip_v4    = "10.0.0.24" -> null
          - fixed_ip_v6    = "[fdac:50a1:b51d:0:f816:3eff:fe23:e552]" -> null
          - mac            = "fa:16:3e:23:e5:52" -> null
          - name           = "private" -> null
          - uuid           = "0e6e2f0c-c4cc-4eca-8428-8117864b6e4a" -> null
        }
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

openstack_compute_instance_v2.instance_terraform: Destroying... [id=c8e4a067-d449-42ba-b5cf-ada95f2ff248]
openstack_compute_instance_v2.instance_terraform: Still destroying... [id=c8e4a067-d449-42ba-b5cf-ada95f2ff248, 10s elapsed]
openstack_compute_instance_v2.instance_terraform: Destruction complete after 10s

Destroy complete! Resources: 1 destroyed.
```

Création d'un volume système et d'un groupe de sécurité :

Nous allons déployer une infrastructure qui comprend les éléments suivants :

- Un volume de boot créé à partir d'une image
- Une instance qui démarre sur ce volume en utilisant un script de démarrage (`user_data`)
- Un groupe de sécurité autorisant le trafic SSH

Dans les piles Heat il est possible de créer un volume de boot à partir du nom de l'image ou de son ID unique. L'unicité des noms d'images n'est pas garantie par Glance et lorsque deux images portent le même nom cela provoque l'échec de création de la pile Heat.

Afin de se prémunir contre ce genre d'échecs Terraform utilise l'ID de l'image au lieu de son nom, une source de donnés (`data`) permet d'interroger la catalogue Glance pour récupérer l'ID à partir du nom de l'image passé dans une variable.

```
variable "image_instance" {
  description = "Nom de l'image"
  default     = "ubuntu-18.04"
}

data "openstack_images_image_v2" "image_boot" {
```

```
  name = "${var.image_instance}"
  most_recent = true
}
```

Il est ensuite possible de créer le volume à partir de cette image.

Le `volume_type` est également récupéré à partir d'une variable (il s'agit habituellement de `lvmdriver-1` pour DevStack).

L'ID de l'image est récupéré à partir de la source de données grâce à la construction suivante :

- `"${data.openstack_images_image_v2.image_boot.id}"`

```
resource "openstack_blockstorage_volume_v2" "volume_boot_terraform" {
  name        = "volume_terraform"
  description = "Le volume de boot"
  size        = 10
  volume_type = "${var.type_volume}"
  image_id    = "${data.openstack_images_image_v2.image_boot.id}"
}
```

Il faut créer le groupe de sécurité puis lui attacher les règles de sécurité, ici celle qui autorise le trafic SSH (tcp port 22) :

```
resource "openstack_networking_secgroup_v2" "secgroup_tf" {
  name        = "secgroup_ssh"
  description = "Autoriser le flux SSH"
}

resource "openstack_networking_secgroup_rule_v2" "secgrouprule_secgroup_tf_ssh" {
  direction         = "ingress"
  ethertype         = "IPv4"
  protocol          = "tcp"
  port_range_min    = 22
  port_range_max    = 22
  remote_ip_prefix  = "0.0.0.0/0"
  security_group_id = "${openstack_networking_secgroup_v2.secgroup_tf.id}"
}
```

Nous pouvons désormais créer notre serveur en y attachant les différents éléments que nous avons construits. Le script exécuté au moment du lancement du serveur (`user_data`) est recopié depuis un fichier, contrairement aux piles Heat dans lesquelles il est inclus dans le fichier de la pile :

```
resource "openstack_compute_instance_v2" "instance_terraform" {
  name            = "instance_terraform"
  flavor_name     = "${var.gabarit_instance}"
  key_pair        = "${var.nom_de_cle}"
  security_groups = ["${openstack_networking_secgroup_v2.secgroup_tf.name}"]
  user_data       = "${file("configuration_evoluee_user_data.sh")}"

  block_device {
    uuid                  = "${openstack_blockstorage_volume_v2.volume_boot_terraform.id}"
    source_type           = "volume"
    destination_type      = "volume"
    boot_index            = 0
    delete_on_termination = false
  }
}
```

Le fichier `configuration_evoluee_user_data.sh` provient d'une de nos piles Heat précédentes :

```
#!/bin/bash -v

apt-get update
apt-get install -y cowsay fortune
/usr/games/fortune | /usr/games/cowsay
/usr/games/fortune | /usr/games/cowsay > /dev/tty1
cat << EOF >> /home/ubuntu/.bashrc
/usr/games/fortune | /usr/games/cowsay
EOF
```

Les fichiers pour cette configuration sont disponibles à cette adresse :

- https://u03.fr/90openstack/configuration_evoluee.tf
- https://u03.fr/90openstack/configuration_evoluee_user_data.sh

Le fichier `terraform.tfvars` avec la valeur de nos éléments `variable` contiendra :

```
nom_de_cle       =  "LFO"
gabarit_instance = "ds1G"
reseau_public = "public"
type_volume = "lvmdriver-1"
```

Vous devez effacer le fichier premiere_configuration.tf, sinon Terraform va également tenter de déployer le serveur décrit dedans.

Il vous est laissé à titre d'exercice d'afficher le plan, de déployer la configuration, de vous connecter à l'instance avec l'utilisateur ubuntu (vous serez également accueilli par la vache) puis de détruire la configuration.

Note : N'oubliez pas qu'il faut un peu de temps au serveur pour se lancer et exécuter son script de démarrage.

Création et utilisation de réseaux

Tout comme avec Heat on commence par créer un réseau, puis on y attache un ou plusieurs sous-réseaux.

Nous voyons dans l'exemple ci-dessous que le DNS est une variable, il est en effet important que le DNS soit adapté à notre infrastructure, sinon de nombreuses fonctionnalités risquent de ne pas fonctionner dans les instances attachées au réseau.

Les différents paramètres sont les mêmes que pour Heat, en particulier les plages d'adresses IP dynamiques pour DHCP (`allocation_pools`) :

```
variable "dns" {
  type        = "list"
  description = "Adresse IP du serveur DNS"
}


resource "openstack_networking_network_v2" "net_bastion" {
  name           = "net_bastion"
  admin_state_up = "true"
}

resource "openstack_networking_subnet_v2" "subnet_bastion" {
  name            = "subnet_bastion"
  network_id      = "${openstack_networking_network_v2.net_bastion.id}"
  cidr            = "10.99.1.0/24"
  allocation_pools = {
    start = "10.99.1.100"
    end   = "10.99.1.250"
  }  dns_nameservers = "${var.dns}"
  gateway_ip      = "10.99.1.254"
  ip_version      = 4
}
```

La variable DNS est une liste de serveurs DNS.

Il est nécessaire de spécifier `type = "list"` dans la déclaration de la variable.

Dans le fichier `.tfvars` la valeur de la varible se défini comme ceci :

```
dns = ["8.8.8.8", "8.8.4.4"]
```

Une fois les différents réseaux définis il est possible de créer le routeur pour les relier entre eux et les relier au réseau public. Là encore, le nom du réseau public doit correspondre à l'infrastructure et il s'agit donc d'une variable pour rendre notre configuration Terraform partageable.

```
resource "openstack_networking_router_v2" "routeur" {
  name                = "routeur"
  admin_state_up      = true
  external_network_id = "${var.id_reseau_public}"
}

resource "openstack_networking_router_interface_v2" "router_interface_bastion" {
  router_id = "${openstack_networking_router_v2.routeur.id}"
  subnet_id = "${openstack_networking_subnet_v2.subnet_bastion.id}"
}

resource "openstack_networking_router_interface_v2" "router_interface_pio" {
  router_id = "${openstack_networking_router_v2.routeur.id}"
  subnet_id = "${openstack_networking_subnet_v2.subnet_pio.id}"
}
```

Notez comment le serveur est rattaché à son réseau, ici une adresse fixe lui est donnée :

```
resource "openstack_compute_instance_v2" "instance_bastion" {
   [.../...]
  network {
    fixed_ip_v4 = "${var.ip_interne_bastion}"
    name = "net_bastion"
    access_network = true
  }
   [.../...]
}
```

Script d'initialisation d'instance simple

Terraform dispose de la possibilité de faire exécuter un script au moment du lancement de l'instance (ce qui est normal, les `user_data` étant une fonctionnalité de OpenStack).

Le cas le plus simple est l'inclusion d'un fichier externe qui sera exécuté au lancement de l'instance :

```
resource "openstack_compute_instance_v2" "instance_terraform" {
  name            = "instance_terraform"
  flavor_name     = "${var.gabarit_instance}"
  key_pair        = "${var.nom_de_cle}"
  security_groups = ["${openstack_networking_secgroup_v2.secgroup_tf.name}"]
  user_data       = "${file("cowsay.sh")}"

  block_device {
    uuid                  = "${openstack_blockstorage_volume_v2.volume_boot_terraform.id}"
    source_type           = "volume"
    destination_type      = "volume"
    boot_index            = 0
    delete_on_termination = false
  }
}
```

Le fichier `cowsay.sh` est un script shell classique :

```
#!/bin/bash -v

apt-get update
apt-get install -y cowsay fortune
/usr/games/fortune | /usr/games/cowsay
cat << EOF >> /home/ubuntu/.bashrc
/usr/games/fortune | /usr/games/cowsay
EOF
```

Script d'initialisation d'instance complexe (template)

Terraform dispose d'un système de templates qui permet, entre autres, de personnaliser le script d'initialisation des instances en remplaçant des motifs dans un fichier modèle

Un élément 'data template_file' est utilisé pour désigner le fichier modèle et faire la liaison entre les motifs et leurs valeurs de remplacement.

```
data "template_file" "userdata_bastion" {
  template = "${file("bastion.tpl")}"
  vars {
    ip_docker         = "${var.ip_interne_docker}"
    ip_pio            = "${var.ip_interne_pio}"
  }
}
```

Voici un exemple de fichier template '`bastion.tpl`' et de fichier résultant :

```
#!/bin/bash

tee -a /etc/hosts << EOF

${ip_docker}  docker
${ip_pio}  pio reverseproxy
EOF
```

```
#!/bin/bash

tee -a /etc/hosts << EOF

10.10.10.1  docker
10.12.12.1  pio reverseproxy
EOF
```

La liaison avec l'instance se fait comme ceci :

```
resource "openstack_compute_instance_v2" "instance_bastion" {
  name = "bastion"
  flavor_name = "${var.gabarit_instance_bastion}"
  user_data = "${data.template_file.userdata_bastion.rendered}"
  network {
     [.../...]
}
```

Utilisation directe des API

Jusqu'ici nous avons utilisé les API de OpenStack à travers l'interface en commande de ligne, le Dashboard ou un programme externe comme Terraform

La puissance de OpenStack est l'ouverture des API.

Les API utilisent le protocole HTTPS (ou HTTP pour DevStack), le formalisme JSON en entrée et en sortie.

Les opérations offertes par l'API sont de type CRUD (Create, Read, Update, Delete), les verbes HTTP utilisés sont les suivants :

- Create : POST
- Read : GET
- Update : PUT
- Detele : DELETE

Les paramètres sont passés soit dans l'URL (path) soit dans le corps de la requête (body) au format JSON.

Obtention d'un Token et de la liste des endpoints

Nous l'avons dit, KeyStone est la pierre angulaire du système, il permet l'authentification et contient la base de données des utilisateurs, des projets et les points d'entrée pour les autres API.

La première chose à faire pour utiliser les API est de s'authentifier auprès de KeyStone, en retour on obtient un token, on peut aussi obtenir les points d'entrée pour contacter les autres API. On utilise ensuite ce token pour prouver notre identité aux autres API (attention le token obtenu a une durée de validité).

Le détail de l'API KeyStone pour l'authentification sur un domaine est disponible à l'adresse suivante :

- https://developer.openstack.org/api-ref/identity/v3/#token-authentication-with-scoped-authorization

Nous utilisons `curl` pour dialoguer avec l'API de KeyStone, le protocole est HTTP, nous envoyons l'entête '`Content-Type`' pour informer l'API que le format des données envoyées est JSON, les données à envoyer sont passées après le paramètre '`data`' (et donc la méthode HTTP sera `POST`), nous utilisons les variables d'environnement valorisées par le script `openrc`

```
stack@microwave:~$ source devstack/openrc demo demo
stack@microwave:~$ curl --silent \
     --request POST \
     --include \
     --header "Content-Type: application/json" \
     --data '
{
  "auth": {
    "identity": {
      "methods": [ "password" ],
      "password": { "user": { "domain": { "id": "'$OS_PROJECT_DOMAIN_ID'" },
                              "name": "'$OS_USERNAME'",
                              "password": "'$OS_PASSWORD'" } }
    },
    "scope": {
       "project": { "name": "'$OS_PROJECT_NAME'", "domain": { "id": "'$OS_PROJECT_DOMAIN_ID'" } }
    }
  }
```

```
}
'  \
  "$OS_AUTH_URL/v$OS_IDENTITY_API_VERSION/auth/tokens"; echo
```

La réponse est en deux parties :

- les entêtes HTTP qui vont contenir le code retour (ici '`201 Created`' qui signifie qu'un token a été créé), ainsi que le token après l'entête '`X-Subject-Token`'
- les données qui contiennent la date d'expiration du token ainsi que les différents points d'entrée des autres API :

```
HTTP/1.1 201 Created
Date: Wed, 14 Mar 2018 21:07:53 GMT
Server: Apache/2.4.18 (Ubuntu)
X-Subject-Token:
gAAAAABarDI1U6c4sJJW8WRYQ4jJ1IYxNom8ee8wrzaVgHlQOChh0ZrpNA0VwIK6ObZ3eP0F20vkACAjAFpxurl5p6N3XYdDKtkICE5qE20aLFCAHseYtFpwqAQom
QGMfXV73cpckXUz1GUfAKKtPmrykqKHBezi8iFTHZicYkp4dq4_JZj9XDQ
Vary: X-Auth-Token
Content-Type: application/json
Content-Length: 5150
x-openstack-request-id: req-67b012fd-8a25-4a42-a034-d5640f48c209
Connection: close

{"token": {"is_domain": false, "methods": ["password"], "roles": [{"id": "c4f81ba8d77b49d3a3d55712dcf42b11", "name":
"Member"}, {"id": "e1d0b86682854ef1b56878f07775b3e4", "name": "anotherrole"}], "expires_at": "2018-03-14T22:07:53.000000Z",
"project": {"domain": {"id": "default", "name": "Default"}, "id": "3f36cd33d14e468293caa7bf311c00de", "name": "demo"},
"catalog": [{"endpoints": [{"region_id": "RegionOne", "url":
"http://192.168.1.211/heat-api/v1/3f36cd33d14e468293caa7bf311c00de", "region": "RegionOne", "interface": "public", "id":
"9ecdad764d374f54bff9faed8fc395d9"}, {"region_id": "RegionOne", "url": "http://192.168.1.211/heat-
   […/…]
"nova"}], "user": {"password_expires_at": null, "domain": {"id": "default", "name": "Default"}, "id":
"cf025806abdb476f936a72573a8ae66f", "name": "demo"}, "audit_ids": ["tH_IWiaRTh6z5WPRB9R8Fw"], "issued_at": "2018-03-
14T21:07:53.000000Z"}}
```

La partie JSON n'est pas très lisible par un être humain, il est possible de le rendre lisible humainement, nous devons supprimer l'affichage des entêtes pour ceci (et donc le token ne sera pas visible).

Nous voyons au passage le endpoint du service '`cinderv3`' ('`volumev3`') qui gère les volumes, celui de '`neutron`' qui gère le réseau ('`network`') et du service '`nova`' ('`compute`') qui gère les instances:

```
stack@microwave:~$ curl --silent \
     --request POST \
     --header "Content-Type: application/json" \
     --data '
{
  "auth": {
    "identity": {
      "methods": [ "password" ],
      "password": { "user": { "domain": { "id": "'$OS_PROJECT_DOMAIN_ID'" },
                              "name": "'$OS_USERNAME'",
                              "password": "'$OS_PASSWORD'" } }
    },
    "scope": {
       "project": { "name": "'$OS_PROJECT_NAME'", "domain": { "id": "'$OS_PROJECT_DOMAIN_ID'" } }
    }
  }
}
'    "$OS_AUTH_URL/v$OS_IDENTITY_API_VERSION/auth/tokens" | python -m json.tool | pygmentize -l json

{
    "token": {
        "audit_ids": [
            "lg6y1QFKT0Czcy26OlwjKA"
        ],
        "catalog": [
            {
   […/…)
            {
                "endpoints": [
                    {
                        "id": "17069516fedb40cc8e44f15ea4b02dba",
                        "interface": "public",
                        "region": "RegionOne",
                        "region_id": "RegionOne",
```

```
                        "url": "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de"
                    }
                ],
                "id": "9e2e4494f10a45b8b2f5000874c1822e",
                "name": "cinderv3",
                "type": "volumev3"
            },
    […/…]
            {
                "endpoints": [
                    {
                        "id": "57953f4651274b2da61f32c0757a7697",
                        "interface": "public",
                        "region": "RegionOne",
                        "region_id": "RegionOne",
                        "url": "http://192.168.1.211:9696/"
                    }
                ],
                "id": "77e4245aa4e84492901cc89c7280e99c",
                "name": "neutron",
                "type": "network"
            },
    […/…]
            {
                "endpoints": [
                    {
                        "id": "c813bc135d5143a0b7327960ab3d92a7",
                        "interface": "public",
                        "region": "RegionOne",
                        "region_id": "RegionOne",
                        "url": "http://192.168.1.211/compute/v2.1"
                    }
                ],
                "id": "fb15a47673b74f64b16e57b97aae0c8d",
                "name": "nova",
                "type": "compute"
            }
    […/…]
        "expires_at": "2018-03-17T22:19:27.000000Z",
        "is_domain": false,
        "issued_at": "2018-03-17T21:19:27.000000Z",
    […/…]
}
```


L'URL du endpoint de Cinder renvoyée par KeyStone est :

- `http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de`

La fin de l'URL du endpoint de Cinder est l'identifiant du projet :

```
stack@microwave:~$ openstack project show demo
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 3f36cd33d14e468293caa7bf311c00de |
| is_domain   | False                            |
| name        | demo                             |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

Utilisation de l'API Cinder : Obtenir la liste des volumes

Nous allons utiliser l'API Cinder pour obtenir la liste des volumes de notre projet, à titre de comparaison la liste des volumes obtenue en utilisant le CLI est la suivante :

```
stack@microwave:~$ openstack volume list
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| ID                                   | Name            | Status    | Size | Attached to                                 |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
| 609fd47f-0c07-46b6-925a-46e8ababdeb89 | donnees         | available |    5 |                                             |
| 417394d7-dc15-48ec-9f22-09b034019459 | ubuntu-16.04_vol | in-use    |   10 | Attached to ubuntu-16.04_inst on /dev/vda   |
+--------------------------------------+-----------------+-----------+------+---------------------------------------------+
```

Il s'agit d'un des cas les plus simples d'utilisation d'une API, la documentation de cette API est disponible à cette adresse :

- https://developer.openstack.org/api-ref/block-storage/v3/index.html#list-accessible-volumes-with-details

Pour obtenir la liste des volumes de notre projet il nous suffit de faire un '`GET`' en HTTP et d'utiliser comme URL celle du endpoint retournée par KeyStone (qui contient l'`id` de notre projet) et de rajouter '`/volumes/detail`' à la fin :

- http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/detail

Nous récupérons le token KeyStone en appelant l'API KeyStone et nous le plaçons dans la variable MON_TOKEN, puis nous appelons l'API de Cinder en utilisant le endpoint renvoyé par KeyStone en même temps que le token, nous ajoutons un entête HTTP '`X-Auth-Token`' :

```
stack@microwave:~$ export MON_TOKEN=gAAAAABarDI1U6c4sJJW8WRYQ4jJ1 […/…] qKHBezi8iFTHZicYkp4dq4_JZj9XDQ

stack@microwave:~$ curl --silent \
            --request GET \
            --header "X-Auth-Token: $MON_TOKEN" \
            "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/detail" | python -m json.tool
{
    "volumes": [
 ⇒ PREMIER VOLUME
        {
            "attachments": [],
            "availability_zone": "nova",
            "bootable": "false",
            "consistencygroup_id": null,
            "created_at": "2018-03-03T16:07:40.000000",
            "description": "",
            "encrypted": false,
            "id": "609fd47f-0c07-46b6-925a-46e8ababdeb89",
            "links": [
                {
                    "href": "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/609fd47f-0c07-46b6-925a-
46e8ababdeb89",
                    "rel": "self"
                },
                {
                    "href": "http://192.168.1.211/volume/3f36cd33d14e468293caa7bf311c00de/volumes/609fd47f-0c07-46b6-925a-
46e8ababdeb89",
                    "rel": "bookmark"
                }
            ],
            "metadata": {
                "readonly": "False"
            },
            "multiattach": false,
            "name": "donnees",
            "os-vol-tenant-attr:tenant_id": "3f36cd33d14e468293caa7bf311c00de",
            "replication_status": null,
            "size": 5,
            "snapshot_id": null,
            "source_volid": null,
            "status": "available",
            "updated_at": "2018-03-04T21:15:53.000000",
            "user_id": "cf025806abdb476f936a72573a8ae66f",
            "volume_type": "lvmdriver-1"
        },
 ⇒ SECOND VOLUME
        {
            "attachments": [
                {
                    "attached_at": "2018-02-18T16:46:57.000000",
                    "attachment_id": "75c0ad36-7aa9-475d-b830-ec2d6a93123b",
                    "device": "/dev/vda",
                    "host_name": null,
                    "id": "417394d7-dc15-48ec-9f22-09b034019459",
                    "server_id": "39ea09c2-659a-4b2a-9552-71fcee89ef4a", ⇒ ID du serveur  ubuntu-16.04_inst
                    "volume_id": "417394d7-dc15-48ec-9f22-09b034019459"
                }
            ],
            "availability_zone": "nova",
            "bootable": "true",
            "consistencygroup_id": null,
            "created_at": "2018-02-18T16:08:57.000000",
            "description": null,
            "encrypted": false,
            "id": "417394d7-dc15-48ec-9f22-09b034019459",
            "links": [
                {
                    "href": "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/417394d7-dc15-48ec-9f22-
09b034019459",
                    "rel": "self"
                },
                {
```

```
                    "href": "http://192.168.1.211/volume/3f36cd33d14e468293caa7bf311c00de/volumes/417394d7-dc15-48ec-9f22-
09b034019459",
                    "rel": "bookmark"
                }
            ],
            "metadata": {
                "attached_mode": "rw",
                "readonly": "False"
            },
            "multiattach": false,
            "name": "ubuntu-16.04_vol",
            "os-vol-tenant-attr:tenant_id": "3f36cd33d14e468293caa7bf311c00de",
            "replication_status": null,
            "size": 10,
            "snapshot_id": null,
            "source_volid": null,
            "status": "in-use",
            "updated_at": "2018-02-18T16:46:57.000000",
            "user_id": "cf025806abdb476f936a72573a8ae66f",
            "volume_image_metadata": {
                "checksum": "9cb8ed487ad8fbc8b7d082968915c4fd",
                "container_format": "bare",
                "disk_format": "qcow2",
                "image_id": "0885d2ea-9445-4a6a-ab45-61ecd42e01d8",
                "image_name": "ubuntu-16.04",
                "min_disk": "0",
                "min_ram": "0",
                "size": "289603584"
            },
            "volume_type": "lvmdriver-1"
        }
    ]
}
```


Si vous utilisez un token incorrect ou expiré vous allez avoir une erreur d'authentification (HTTP 401) :


```
{
    "error": {
        "code": 401,
        "message": "The request you have made requires authentication.",
        "title": "Unauthorized"
    }
}
```


Utilisation de l'API Cinder : Création d'un volume

Nous allons créer un volume à partir d'une image, comme nous l'avions fait en utilisant le CLI OpenStack.

La documentation de l'API correspondante est disponible à cette adresse :

- https://developer.openstack.org/api-ref/block-storage/v3/index.html#create-a-volume

Cette opération est réalisée par un ordre POST, le nom du projet est passé dans l'URL (path) de la même façon que nous avions obtenu la liste des volumes, les autres paramètres sont passés dans le corps de la requête au format JSON.

Nous commençons par obtenir l'`id` de l'image Ubuntu à partir de laquelle nous allons créer le volume, nous pouvons utiliser le CLI OpenStack pour bien sûr utiliser l'API pour obtenir cette information :


```
stack@microwave:~$ curl --silent \
     --request GET \
     --header "X-Auth-Token: $MON_TOKEN" \
     "http://192.168.1.211/image/v2/images" | python -mjson.tool
{
   […/…]
            "id": "0885d2ea-9445-4a6a-ab45-61ecd42e01d8",
            "min_disk": 0,
```

```
            "min_ram": 0,
            "name": "ubuntu-16.04",
            "owner": "f5863f4d4266400cbb2562001ea22eba",
    […/…]
}
```

Nous allons créer un volume de 10Go, qui aura pour nom ‘ubuntu_volapi’, tout comme lorsque nous avions créé le volume en utilisant le CLI openstack nous voyons que le statut du volume est ‘creating’ et qu’il n’est pas bootable.

```
stack@microwave:~$ curl --silent \
     --request POST \
     --header "X-Auth-Token: $MON_TOKEN" \
     --header "Content-Type: application/json" \
     --data '
{"volume": {
    "size": "10",
    "name": "ubuntu_volapi",
    "imageRef": "0885d2ea-9445-4a6a-ab45-61ecd42e01d8"
  }
}' \
 "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes" | python -mjson.tool
{
    "volume": {
        "attachments": [],
        "availability_zone": "nova",
        "bootable": "false",
        "consistencygroup_id": null,
        "created_at": "2018-03-17T17:24:55.000000",
        "description": null,
        "encrypted": false,
        "id": "160ce17d-df05-43b5-b262-0327f3d0d994",
        "links": [
            {
                "href": "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/160ce17d-df05-43b5-b262-
0327f3d0d994",
                "rel": "self"
            },
            {
                "href": "http://192.168.1.211/volume/3f36cd33d14e468293caa7bf311c00de/volumes/160ce17d-df05-43b5-b262-
0327f3d0d994",
                "rel": "bookmark"
            }
        ],
        "metadata": {},
        "multiattach": false,
        "name": "ubuntu_volapi",
        "replication_status": null,
        "size": 10,
        "snapshot_id": null,
        "source_volid": null,
        "status": "creating",
        "updated_at": null,
        "user_id": "cf025806abdb476f936a72573a8ae66f",
        "volume_type": "lvmdriver-1"
    }
}
```

Nous interrogeons l’API d’OpenStack pour vérifier quand notre volume est disponible, on note au passage qu’il devient bootable, ce qui nous permettra de lancer une instance à partir de ce volume.

- https://developer.openstack.org/api-ref/block-storage/v3/index.html#show-a-volume-s-details

```
stack@microwave:~$ curl --silent \
     --request GET \
     --header "X-Auth-Token: $MON_TOKEN" \
     "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/160ce17d-df05-43b5-b262-0327f3d0d994" \
    | python -mjson.tool
{
    "volume": {
        "attachments": [],
        "availability_zone": "nova",
        "bootable": "true",
        "consistencygroup_id": null,
        "created_at": "2018-03-17T17:24:55.000000",
        "description": null,
        "encrypted": false,
        "id": "160ce17d-df05-43b5-b262-0327f3d0d994",
        "links": [
            {
                "href": "http://192.168.1.211/volume/v3/3f36cd33d14e468293caa7bf311c00de/volumes/160ce17d-df05-43b5-b262-
0327f3d0d994",
                "rel": "self"
```

```
            },
            {
                "href": "http://192.168.1.211/volume/3f36cd33d14e468293caa7bf311c00de/volumes/160ce17d-df05-43b5-b262-
0327f3d0d994",
                "rel": "bookmark"
            }
        ],
        "metadata": {},
        "multiattach": false,
        "name": "ubuntu_volapi",
        "os-vol-tenant-attr:tenant_id": "3f36cd33d14e468293caa7bf311c00de",
        "replication_status": null,
        "size": 10,
        "snapshot_id": null,
        "source_volid": null,
        "status": "available",
        "updated_at": "2018-03-17T17:24:57.000000",
        "user_id": "cf025806abdb476f936a72573a8ae66f",
        "volume_image_metadata": {
            "checksum": "9cb8ed487ad8fbc8b7d082968915c4fd",
            "container_format": "bare",
            "disk_format": "qcow2",
            "image_id": "0885d2ea-9445-4a6a-ab45-61ecd42e01d8",
            "image_name": "ubuntu-16.04",
            "min_disk": "0",
            "min_ram": "0",
            "size": "289603584"
        },
        "volume_type": "lvmdriver-1"
    }
}
```

Utilisation de l'API Compute: Création d'une instance

Nous allons utiliser le volume que nous venons de créer pour créer une instance, nous avons déjà l'`id` du volume, il nous faut l'`id` d'un gabarit (flavor), l'`id` de notre réseau privé ainsi que le nom d'une clé publique.

L'API pour obtenir la liste des gabarits est la suivante :

- https://developer.openstack.org/api-ref/compute/#list-flavors

Il vous est laissé à titre d'exercice d'utiliser l'API compute pour récupérer l'`id` du gabarit '`m1.small`'

Indice: Il vous faut le endpoint de l'API 'compute'.

La description de l'API pour obtenir les détails d'un gabarit est la suivante :

- https://developer.openstack.org/api-ref/compute/#show-flavor-details

Il vous est laissé à titre d'exercice d'utiliser l'API compute pour récupérer les caractéristiques du gabarit '`m1.small`' (à partir de son `id`)

Il est également nécessaire de spécifier à quel réseau connecter le serveur, nous allons le connecter au réseau privé ('`private`'), nous devons obtenir son `id` grâce à l'API Neutron :

- https://developer.openstack.org/api-ref/network/v2/index.html#list-networks

```
stack@microwave:~$ curl --silent \
     --request GET \
     --header "X-Auth-Token: $MON_TOKEN" \
     "http://192.168.1.211:9696/v2.0/networks.json" | python -mjson.tool
{
    "networks": [
        {
   […/…]
            "id": "087bc8a1-65e2-4747-b96e-4dd421c542e5",
            "name": "public",
```

```
    […/…]
            "id": "8e7ebaa8-a44e-4a94-9060-9d5fc6143ddf",
            "name": "private",
    […/…]
}
```

La dernière information importante est le nom de la clé publique à utiliser :

- https://developer.openstack.org/api-ref/compute/#list-keypairs

```
stack@microwave:~$ curl --silent \
     --request GET \
     --header "X-Auth-Token: $MON_TOKEN" \
     "http://192.168.1.211/compute/v2.1/os-keypairs" | python -mjson.tool
{
    "keypairs": [
        {
            "keypair": {
                "fingerprint": "8c:56:76:56:d9:de:50:81:8b:4d:6c:b1:ee:55:b2:ae",
                "name": "Foucher Laurent RSA",
                "public_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDSjnEGxaX1FsDqp6x2TrYACUWCQgHiq5YGtfdZ5O1z7F5ic9z57Cf5zYgRme40WUpB+6JW8Qv6Jbw8mswyTLjgYGxh97Hp3
5vg3SBIrVfEoGw7jYwRmo2Xf5zMx1orxNcTUnvSGRusoY6CdR28129sfzsIxV7Zh9EMisDt6lXra+CdescSQG3UNiQ0KcIoS24N/
zyImdkXkxrDeeqHPiw9FFNpAbFnwNQx7VCwiwJgjII2wVSYYYBcnpTRzqU8PL0HSU9xcDKn5n9iuD8pFAieF1nSP6k2e6E7hOuPEkIkSkemUedUS6t2BS+pUZqg4b
bMDcSCz7OL0acjyvdHmlkcdjNJzPsEFdqZ7eKBTwkOFS3ot7brpRtG8AMjDKx0KQHUawm8S2vPW3RjhD4QB2osPpvmokKLyvP4mZy0II3RN9ga69nwntKEBGUbclA
wBEQwUE0l0MhknLymVbABtbmllO54X8R940LTyVfXC92RXL/F3j5iJB91WtUwf5ODLB96rp09I4IJI3loEkW7r0cV2NbvusYSZBaD/
PVZoV9h5i2oz8UospaGHjoce4cJG+ZOLKB/lUM778rXVL/
oFFm2TRMEOH4x2yBq0EyJo4o+cOquWhj29FFCB2yylpWhpi42Nf80qXM0d7jKxwaKkS7R7ZdM1eAJ44B/gV57Msqiiw== Foucher, Laurent\n"
            }
        }
    ]
}
```

Nous avons maintenant toutes les informations nécessaires pour créer une instance,

La description de l'API qui permet de créer une instance est disponible à cette adresse :

- https://developer.openstack.org/api-ref/compute/#create-server

```
stack@microwave:~$ curl --silent \
     --request POST \
     --header "X-Auth-Token: $MON_TOKEN" \
     --header "Content-Type: application/json" \
     --data '
{"server": {
    "name": "ubuntu_instapi",
    "flavorRef": "2",
    "networks": [{
      "uuid": "8e7ebaa8-a44e-4a94-9060-9d5fc6143ddf"
    }],
    "key_name": "Foucher Laurent RSA",
    "block_device_mapping_v2": [{
      "source_type": "volume",
      "destination_type": "volume",
      "boot_index": "0",
      "uuid": "160ce17d-df05-43b5-b262-0327f3d0d994"
    }]
  }
}' \
 "http://192.168.1.211/compute/v2.1/servers" | python -mjson.tool
{
    "server": {
        "OS-DCF:diskConfig": "MANUAL",
        "adminPass": "4d8L5QtyWvAV",
        "id": "564754fa-3bf7-44fb-96ce-41a2cf9f6ee2",
        "links": [
            {
                "href": "http://192.168.1.211/compute/v2.1/servers/564754fa-3bf7-44fb-96ce-41a2cf9f6ee2",
                "rel": "self"
            },
            {
                "href": "http://192.168.1.211/compute/servers/564754fa-3bf7-44fb-96ce-41a2cf9f6ee2",
                "rel": "bookmark"
            }
        ],
        "security_groups": [
            {
                "name": "default"
            }
        ]
    }
}
```

Nous utilisons l'API compute pour vérifier si notre serveur est disponible :

- https://developer.openstack.org/api-ref/compute/#show-server-details

```
stack@microwave:~$ curl --silent \
    --request GET \
    --header "X-Auth-Token: $MON_TOKEN" \
    "http://192.168.1.211/compute/v2.1/servers/564754fa-3bf7-44fb-96ce-41a2cf9f6ee2" | python -mjson.tool
{
    "server": {
        "OS-DCF:diskConfig": "MANUAL",
        "OS-EXT-AZ:availability_zone": "nova",
        "OS-EXT-STS:power_state": 1,
        "OS-EXT-STS:task_state": null,
        "OS-EXT-STS:vm_state": "active",
        "OS-SRV-USG:launched_at": "2018-03-18T11:28:35.000000",
        "OS-SRV-USG:terminated_at": null,
    […/…]
        "id": "564754fa-3bf7-44fb-96ce-41a2cf9f6ee2",
        "image": "",
        "key_name": "Foucher Laurent RSA",
        "name": "ubuntu_instapi",
        "os-extended-volumes:volumes_attached": [
            {
                "id": "160ce17d-df05-43b5-b262-0327f3d0d994"
            }
        ],
    […/…]
        "status": "ACTIVE",
        "tenant_id": "3f36cd33d14e468293caa7bf311c00de",
        "updated": "2018-03-18T11:28:36Z",
        "user_id": "cf025806abdb476f936a72573a8ae66f"
    }
}
```

Il vous est laissé à titre d'exercice d'utiliser les API pour associer une adresse IP flottante au serveur, le groupe de sécurité qui nous avions créé et qui autorise les connexions en SSH

Vous devriez ensuite être capable de vous connecter à votre serveur en SSH.

Utilisation du client CLI openstack en mode debug

Vous pouvez utiliser le client CLI openstack en mode debug pour avoir la liste des appels à l'API avec leurs paramètres et les données renvoyées par l'API.

Ceci peut vous permettre de récupérer les appels API nécessaires à vos projets (notez au passage que le token n'est jamais affiché en clair en mode debug, seulement son hash SHA-1) :

```
stack@microwave:~$ openstack --debug server list
START with options: [u'--debug', u'server', u'list']
    […/…]
REQ: curl -g -i -X GET http://192.168.1.211/compute/v2.1/servers/detail -H "User-Agent: python-novaclient" -H "Accept:
application/json" -H "X-Auth-Token: {SHA1}2bc21fc4fc2075c05d796fa0e72c2d5895da3ece"
http://192.168.1.211:80 "GET /compute/v2.1/servers/detail HTTP/1.1" 200 3196
RESP: [200] Date: Sun, 18 Mar 2018 13:25:20 GMT Server: Apache/2.4.18 (Ubuntu) Content-Length: 3196 Content-Type:
application/json OpenStack-API-Version: compute 2.1 X-OpenStack-Nova-API-Version: 2.1 Vary: OpenStack-API-Version,X-
OpenStack-Nova-API-Version x-openstack-request-id: req-3f13e308-890e-4681-ba83-d66e2e188800 x-compute-request-id: req-
3f13e308-890e-4681-ba83-d66e2e188800 Connection: close
RESP BODY: {"servers": [{"OS-EXT-STS:task_state": null, "addresses": {"private": [{"OS-EXT-IPS-MAC:mac_addr":
"fa:16:3e:9d:db:6c", "version": 4, "addr": "10.0.0.7", "OS-EXT-IPS:type": "fixed"}, {"OS-EXT-IPS-MAC:mac_addr":
"fa:16:3e:9d:db:6c", "version": 6, "addr": "fdb0:453a:3cd:0:f816:3eff:fe9d:db6c", "OS-EXT-IPS:type": "fixed"}, {"OS-EXT-IPS-
MAC:mac_addr": "fa:16:3e:9d:db:6c", "version": 4, "addr": "172.24.4.13", "OS-EXT-IPS:type": "floating"}]}, "links": [{"href":
"http://192.168.1.211/compute/v2.1/servers/564754fa-3bf7-44fb-96ce-41a2cf9f6ee2", "rel": "self"}, {"href":
"http://192.168.1.211/compute/servers/564754fa-3bf7-44fb-96ce-41a2cf9f6ee2", "rel": "bookmark"}], "image": "", "OS-EXT-
STS:vm_state": "active", "OS-SRV-USG:launched_at": "2018-03-18T11:28:35.000000", "flavor": {"id": "2", "links": [{"href":
"http://192.168.1.211/compute/flavors/2", "rel": "bookmark"}]}, "id": "564754fa-3bf7-44fb-96ce-41a2cf9f6ee2",
    […/…]
+--------------------------------------+-------------------+--------+----------- ... -+----------+
| ID                                   | Name              | Status | Networks        | Flavor   |
+--------------------------------------+-------------------+--------+----------- ... -+----------+
| 564754fa-3bf7-44fb-96ce-41a2cf9f6ee2 | ubuntu_instapi    | ACTIVE | private=10      | m1.small |
| 39ea09c2-659a-4b2a-9552-71fcee89ef4a | ubuntu-16.04_inst | ACTIVE | private=10      | m1.small |
+--------------------------------------+-------------------+--------+----------- ... -+----------+
clean_up ListServer:
END return value: 0
```

Conclusion

J'espère que cette introduction à l'utilisation de OpenStack vous a plu.

N'oubliez pas de visiter régulièrement mon blog : https://blog.u03.fr/

N'hésitez pas à me faire part de vos remarques, soit en commentant sur mon blog ou directement par email : u03@u03.fr