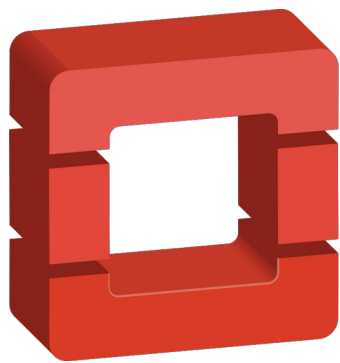


# Formation ICLQ : Openstack Utilisation



openstack™



## Sommaire

Trois transformations profondes.....	6
Le Cloud.....	6
Conteneurisation.....	7
Le mouvement DevOps.....	8
Objectifs du DevOps.....	8
.....	8
Apports des orchestrateurs de conteneurs pour le DevOps.....	9
Qu'est-ce que le Cloud ?.....	10
OpenStack.....	11
Introduction.....	11
Composants d'OpenStack.....	15
Architecture d'OpenStack.....	16
Déployer Openstack.....	18
Kolla-ansible.....	18
Caractéristiques clés de Kolla-Ansible.....	18
Tp -Déployer Openstack avec kolla.....	19
Keystone.....	28
Introduction.....	28
.....	31
TP – Gestion des projets, utilisateurs et rôles.....	33
Création d'un projet.....	33

## Formation ILCQ : Openstack Utilisateur

Création de l'utilisateur membre.....	33
Création de l'utilisateur administrateur.....	34
Se connecter au dashboard.....	34
Le service Glance.....	36
Introduction.....	36
TP – Gestion des images.....	38
Utilisation de l'outil QEMU-NDB.....	38
Le service Neutron.....	43
Introduction.....	43
Principe.....	43
Les réseaux provider Neutron.....	45
le réseau "flat".....	45
TP – Création de la topologie réseau.....	47
Ajout d'un réseau externe.....	48
Les groupes de sécurité.....	48
Les adresses IP flottantes (floating IP).....	50
TP – Création de la topologie réseau pour le projet formation.....	52
Le service Compute Nova.....	54
Introduction.....	54
Les instances.....	55
Les flavors.....	57
TP – Création d'instance.....	58
Cloud init.....	60
Introduction.....	60

configuration initiale.....	61
Le format cloud-config.....	62
TP – Personnaliser votre instance avec Cloud-init.....	67
Le service de stockage block Cinder.....	68
Introduction.....	68
Les types de volumes.....	69
Les backend de stockage.....	71
Les snapshots et clones.....	72
Tp -Gestion des volumes.....	74
Création de volume type lvm et nfs.....	74
Attacher les volumes a l'instance.....	74
Utilisation du volume dans les instances.....	74
TP – Réaliser un cloud-init pour configurer automatiquement les volumes Cinder.....	75
Utilisation du volume dans les instances.....	75
Heat.....	76
Introduction à Heat.....	76
Le modèle HOT.....	77
TP - Déploiement avec Heat.....	79
Déploiement d'une instance avec Heat.....	79
Qu'est-ce qu'une application Cloud-Ready/Native ?.....	83
Trove.....	88
Introduction.....	88
TP – Database As Service.....	92

Octavia.....	94
Introduction.....	94
Configuration d'Octavia.....	94
Déploiement d'un load balancer avec Octavia.....	94
Magnum.....	95
Introduction.....	95
Le reseau avec magnum.....	97
TP- déploiement d' un cluster avec Magnum.....	99

## Trois transformations profondes de l'informatique

Le Cloud se trouve au cœur de trois transformations profondes techniques, humaines et économiques de l'informatique :

- L' Informatique As Service
- La conteneurisation logicielle
- Le mouvement DevOps

Il est un des projets qui symbolise et supporte techniquement ces transformations. D'où son omniprésence dans les discussions informatiques actuellement.

### Le Cloud

- Au-delà du flou dans l'emploi de ce terme, le cloud est un mouvement de réorganisation technique et économique de l'informatique.
- On retourne à la consommation de "temps de calcul" et de services après une "aire du Personnel Computer".
- Pour organiser cela on définit trois niveaux à la fois techniques et économiques de l'informatique :
  - **Software as a Service** : location de services à travers internet pour les usagers finaux
  - **Platform as a Service** : location d'un environnement d'exécution logiciel flexible à destination des développeurs
  - **Infrastructure as a Service** : location de ressources "matérielles" à la demande pour des VMs et de conteneurs sans avoir à maintenir un data center.

## Conteneurisation

La conteneurisation est permise par l'isolation au niveau du noyau du système d'exploitation du serveur : les processus sont isolés dans des namespaces au niveau du noyau. Cette innovation permet d'isolation des applications et leur dépendances sans ajouter une couche de virtualisation.

Ainsi les conteneurs permettent d'avoir des performances d'une application traditionnelle tournant directement sur le système d'exploitation hôte et ainsi d'optimiser les ressources.

Les technologies de conteneurisation permettent donc d'exécuter des applications dans des «boîtes» isolées, ceci pour apporter l'uniformisation du déploiement :

- Une façon standard de packager un logiciel (basée sur l'image)
- Cela réduit la complexité grâce:
  - À l'intégration de toutes les dépendance déjà dans la boîte
  - Au principe d'immutabilité qui implique de jeter les boîtes ( automatiser pour lutter contre la culture de prudence). Rend l'infra prédictible.

## Le mouvement DevOps

- Dépasser l'opposition culturelle et de métier entre les développeurs et les administrateurs système.
- Intégrer tout le monde dans une seule équipe et ...
- Calquer les rythmes de travail sur l'organisation agile du développement logiciel
- Rapprocher techniquement la gestion de l'infrastructure du développement avec l'infrastructure as code.



- Concrètement on écrit des fichiers de code pour gérer les éléments d'infra
- L'état de l'infrastructure est plus claire et documentée par le code
- La complexité est plus gérable car tout est déclaré et modifiable au fur et à mesure de façon centralisée
- L'usage de git et des branches/tags pour la gestion de l'évolution d'infrastructure

## **Objectifs du DevOps**

- Rapidité (velocity) de déploiement logiciel (organisation agile du développement et livraison jusqu'à plusieurs fois par jour)
  - Implique l'automatisation du déploiement et ce qu'on appelle la CI/CD c'est à dire une infrastructure de déploiement continu à partir de code.
- Passage à l'échelle (horizontal scaling) des logiciels et des équipes de développement (nécessaire pour les entreprises du cloud qui doivent servir pleins d'utilisateurs)
- Meilleure organisation des équipes
  - Meilleure compréhension globale du logiciel et de son installation de production car le savoir est mieux partagé
  - Organisation des équipes par thématique métier plutôt que par spécialité technique (l'équipe scale mieux)

## **Apports des orchestrateurs de conteneurs pour le DevOps**

- Abstraction et standardisation des infrastructures :

- Langage descriptif et incrémental : on décrit ce qu'on veut plutôt que la logique complexe pour l'atteindre
- Logique opérationnelle intégrée dans l'orchestrateur : la responsabilité de l'état du cluster est laissée au contrôleur k8s ce qui simplifie le travail

Aux vues des transformations humaines et techniques précédentes, l'organisation des orchestrateurs de conteneurs prend vraiment sens pour le développement d'applications microservices ou Cloud ready :

- Des applications avec de nombreux de "petits" services.
- Chaque service a des problématiques très limitées (gestion des utilisateurs = une application qui ne fait que ça)
- Les services communiquent par le réseaux selon différents modes API (REST, gRPC, job queues, GraphQL)

Les microservices permettent justement le DevOps car :

- Ils peuvent être déployés séparément

**Une petite équipe gère chaque service ou groupe thématique de services**

## Qu'est-ce que le Cloud ?

Le cloud (ou cloud computing) est un modèle de prestation de services informatiques via Internet ou le réseau d'entreprise. Au lieu de stocker et de traiter des données sur un ordinateur local, le cloud permet aux utilisateurs d'accéder à des ressources informatiques telles que des serveurs, des bases de données, des applications et des services via Internet.

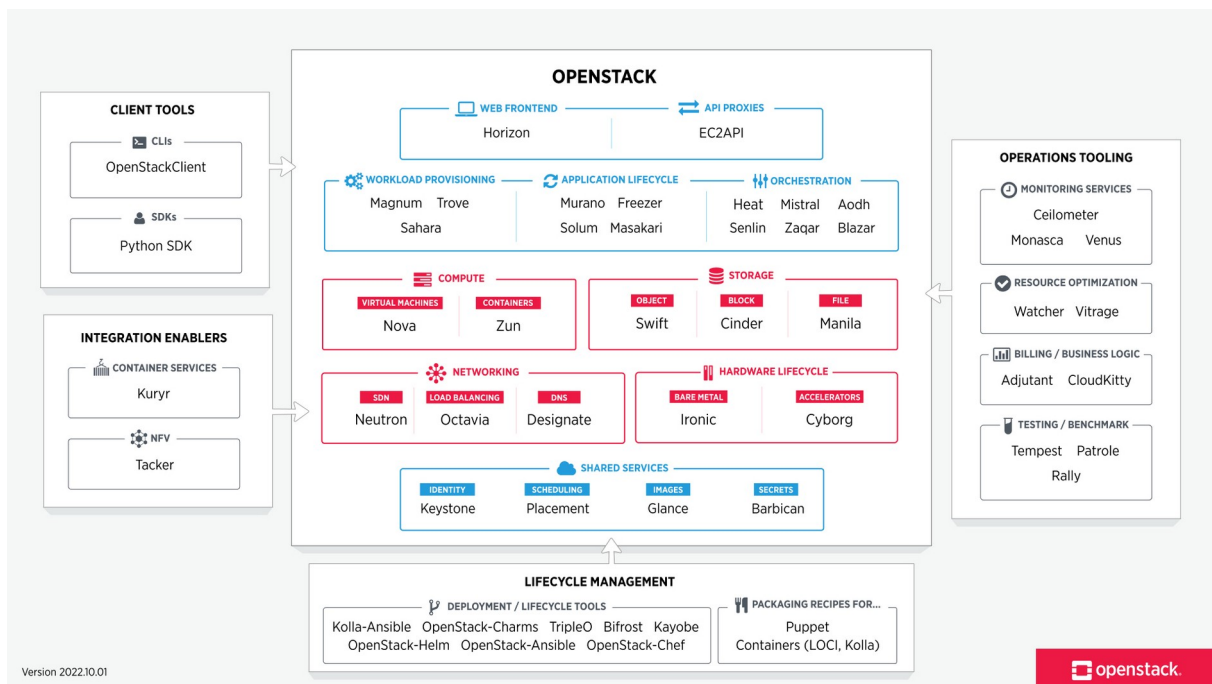
Le cloud computing offre une variété de services, y compris des services d'infrastructure (IaaS), des plates-formes de développement (PaaS) et des services de logiciels (SaaS). Les services IaaS fournissent aux utilisateurs des ressources de calcul, de stockage et de réseau, tandis que les services PaaS fournissent des environnements de développement pour la création et le déploiement d'applications. Les services SaaS offrent des applications et des logiciels prêts à l'emploi, tels que les services de messagerie électronique ou les suites bureautiques, via Internet.

Les avantages du cloud computing sont nombreux, notamment la flexibilité, l'évolutivité, la sécurité et l'accessibilité. Le cloud permet aux utilisateurs de facilement accéder à des ressources informatiques à la demande, en fonction de leurs besoins, sans avoir à investir dans des infrastructures coûteuses. De plus, le cloud peut évoluer pour répondre aux besoins croissants des utilisateurs, en ajoutant ou en supprimant des ressources selon les besoins.

En somme, le cloud est un modèle de prestation de services informatiques via Internet, offrant une variété de services tels que les services d'infrastructure, de développement de plateformes et de logiciels prêts à l'emploi. Les avantages du cloud computing sont la flexibilité, l'évolutivité, la sécurité et l'accessibilité, permettant aux utilisateurs d'accéder facilement à des ressources informatiques à la demande, en fonction de leurs besoins.

# OpenStack

## Introduction



OpenStack est un système d'exploitation cloud open source qui permet de créer et de gérer des infrastructures cloud. Il fournit un ensemble de services pour la gestion des ressources, du stockage, du réseau et des machines virtuelles dans un environnement cloud.

OpenStack est considéré comme un système d'exploitation cloud car il fournit une infrastructure cloud complète, similaire à la façon dont un système d'exploitation fournit une plateforme pour exécuter des applications. En utilisant OpenStack, les utilisateurs peuvent déployer, gérer et automatiser l'ensemble de leur infrastructure cloud, y compris le stockage, le réseau, les ressources de calcul et les conteneurs.

OpenStack est open source, ce qui signifie que son code source est disponible librement et peut être utilisé, modifié et distribué par quiconque. Cette nature open source permet à un grand nombre d'utilisateurs de contribuer au développement et à l'amélioration d'OpenStack, ce qui a conduit à une communauté mondiale dynamique et active autour du projet.

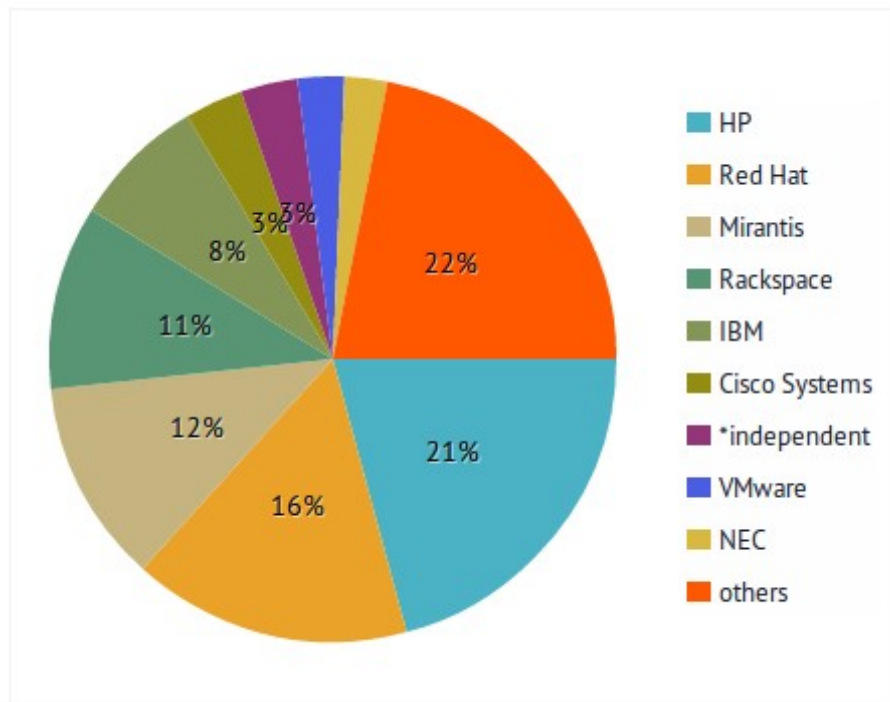
En outre, étant open source, OpenStack est modulaire, ce qui permet aux utilisateurs de choisir les services spécifiques qu'ils souhaitent utiliser en fonction de leurs besoins spécifiques. Cette modularité offre une flexibilité et une extensibilité accrues par rapport aux solutions propriétaires.

OpenStack ne se limite pas à la virtualisation. En plus de la virtualisation, OpenStack prend également en charge d'autres technologies pour fournir des fonctionnalités de cloud computing, telles que le stockage distribué, la gestion de réseau, la sécurité, la planification de capacité, l'orchestration et bien plus encore.

OpenStack fournit un ensemble de services pour la gestion des ressources de calcul, de stockage et de réseau, y compris des services tels que Keystone pour l'authentification et l'autorisation, Neutron pour la gestion de réseau, Cinder pour le stockage de blocs, et Swift pour le stockage d'objets. En plus de cela, OpenStack fournit également des services d'orchestration avec Heat, des services de base de données avec Trove, et des services de conteneurs avec Magnum.

En somme, OpenStack fournit un ensemble complet de services pour la gestion des ressources de calcul, de stockage, de réseau et bien plus encore, avec une architecture extensible qui permet aux utilisateurs de personnaliser et d'adapter OpenStack à leurs besoins spécifiques.

OpenStack est utilisé par un large éventail d'entreprises, d'organisations et d'institutions. Voici les principaux contributeurs



OpenStack est utilisé par un large éventail d'entreprises, d'organisations et d'institutions, allant des startups aux grandes entreprises, des fournisseurs de services cloud aux universités et aux gouvernements. Voici quelques exemples d'entreprises et d'organisations qui utilisent OpenStack :

- Cisco : Cisco utilise OpenStack pour alimenter ses services de cloud computing, notamment pour la fourniture d'infrastructures de cloud privé et public.
- AT&T : AT&T utilise OpenStack pour fournir des services de réseau virtualisé à ses clients, ainsi que pour gérer son propre cloud privé.
- Volkswagen : Volkswagen utilise OpenStack pour créer une infrastructure de cloud privé pour son environnement de développement et de test.
- CERN : Le CERN, l'Organisation européenne pour la recherche nucléaire, utilise OpenStack pour gérer son infrastructure de cloud computing, qui alimente des projets tels que le Grand collisionneur de hadrons (LHC).

- NASA : La NASA utilise OpenStack pour gérer ses propres ressources de cloud computing, ainsi que pour soutenir les projets scientifiques et les collaborations avec d'autres organisations.
- eBay : eBay utilise OpenStack pour fournir une infrastructure de cloud privé à ses équipes de développement et de test.
- Commonwealth Bank of Australia : La Commonwealth Bank of Australia utilise OpenStack pour fournir une infrastructure de cloud privé pour ses charges de travail de développement et de test.

OVH utilise également OpenStack pour fournir des services de cloud computing à ses clients. OVH est l'un des plus grands fournisseurs de services de cloud en Europe, offrant des services de cloud public et privé à des entreprises de toutes tailles.

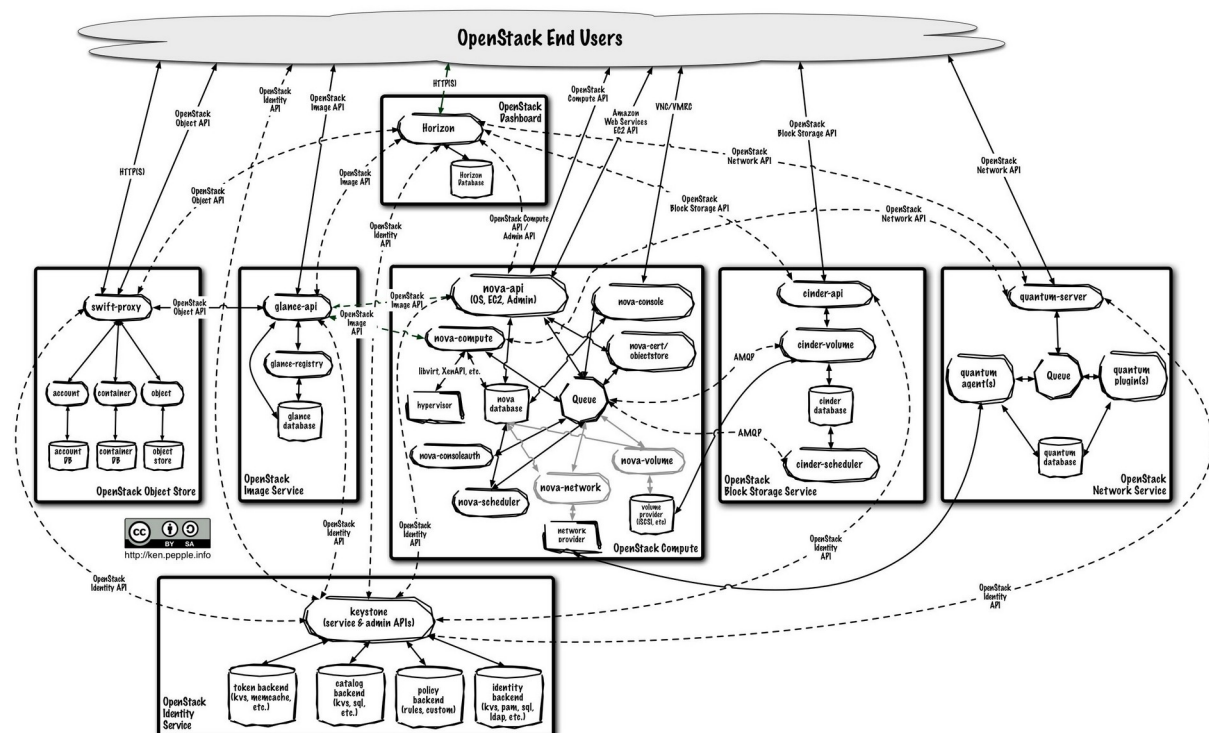
En utilisant OpenStack, OVH a été en mesure de fournir une infrastructure de cloud computing flexible et évolutive, offrant une variété de services tels que le calcul, le stockage et la gestion de réseau. En plus de cela, OVH a également créé une version personnalisée d'OpenStack, appelée OVHcloud OpenStack, qui offre une expérience utilisateur améliorée et des fonctionnalités supplémentaires.

OVH est un exemple de la façon dont les fournisseurs de services de cloud peuvent utiliser OpenStack pour offrir des services de cloud computing évolutifs, flexibles et abordables à leurs clients.

Ces exemples ne représentent qu'une fraction des entreprises et des organisations qui utilisent OpenStack dans le monde. En raison de sa nature open source et de sa flexibilité, OpenStack est largement adopté dans de nombreux secteurs et industries pour fournir des infrastructures cloud privées et publiques.

## Composants d'OpenStack

OpenStack est composé de plusieurs projets qui travaillent ensemble pour fournir des fonctionnalités complètes pour les infrastructures cloud. Voici quelques-uns des projets les plus couramment utilisés:



**Nova** : Il fournit des services de calcul en nuage pour le déploiement et la gestion des machines virtuelles.

**Glance** : Il fournit un service de gestion des images pour stocker et gérer les images de machines virtuelles.

**Cinder** : Il fournit des services de stockage en nuage pour le stockage persistant des blocs de données.



**Neutron** : Il fournit un service de réseau pour la configuration et la gestion des réseaux virtuels.

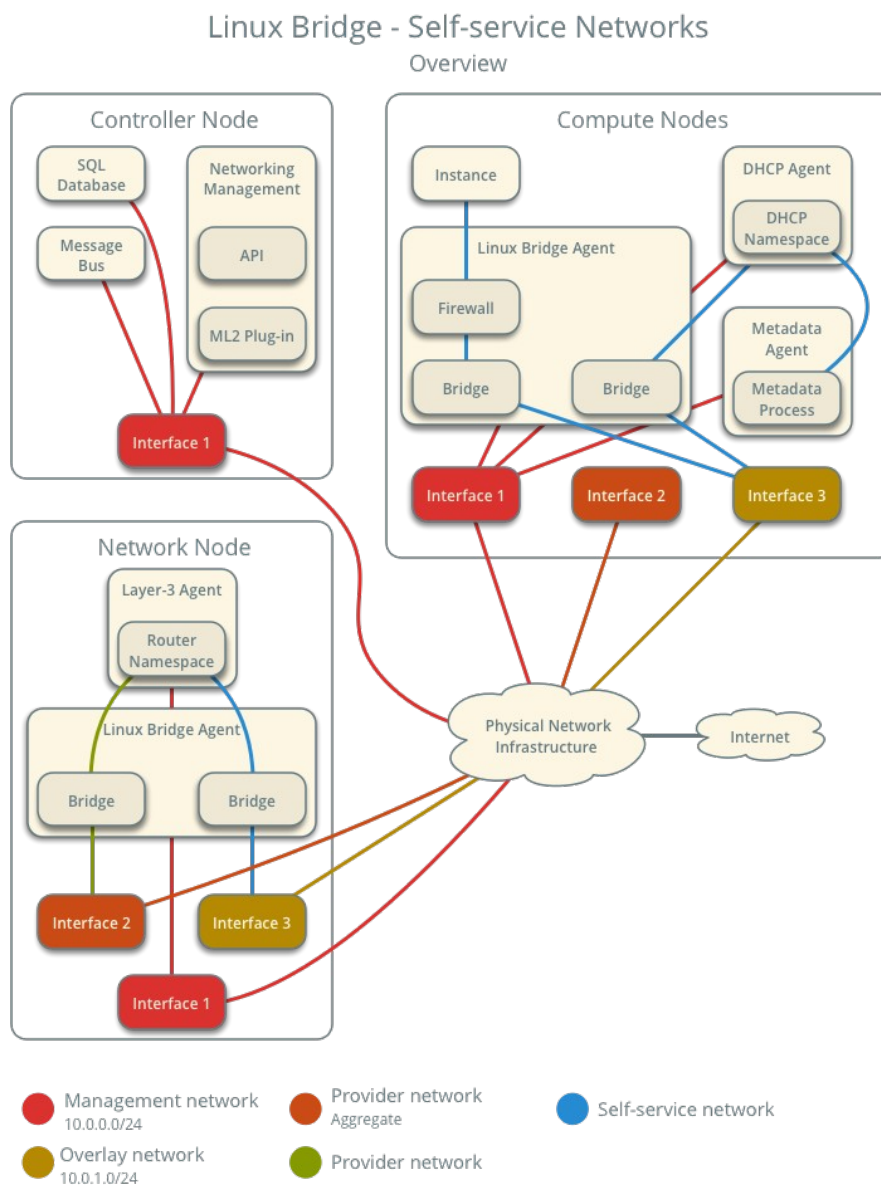
**Keystone** : Il fournit un service d'authentification et d'autorisation pour les utilisateurs et les applications.

**Horizon** : Il fournit une interface utilisateur Web pour gérer les ressources et les instances OpenStack.

**Swift** : Il fournit un service de stockage d'objets pour stocker et récupérer des objets de données de manière distribuée.

## Architecture d'OpenStack

L'architecture d'OpenStack est conçue pour être distribuée et évolutive horizontalement. Elle est composée de différents composants qui travaillent ensemble pour fournir une infrastructure cloud complète. Voici quelques-uns des composants principaux de l'architecture d'OpenStack:



**Le contrôleur :** Il est responsable de la gestion des services OpenStack, de la gestion de la base de données et de la communication entre les différents services.

**Les nœuds de calcul :** Ils fournissent la puissance de calcul pour les instances de machines virtuelles et exécutent les charges de travail des utilisateurs.

**Le nœud de stockage :** Il stocke les images de machines virtuelles, les disques et les fichiers.

**Le nœud réseau :** Il fournit une connectivité entre les différentes machines virtuelles et les différents services.

L'architecture d'OpenStack est conçue pour être modulaire, ce qui permet aux utilisateurs de choisir les composants qu'ils souhaitent utiliser en fonction de leurs besoins spécifiques.

## Déployer Openstack

### Kolla-ansible

Kolla-Ansible est un projet open source faisant partie de l'écosystème OpenStack. Il vise à simplifier le déploiement et la gestion d'OpenStack en utilisant Ansible, un outil de gestion de configuration et d'orchestration largement utilisé.

<https://github.com/openstack/kolla-ansible>

Kolla-Ansible automatise le processus de déploiement d'OpenStack en utilisant des conteneurs Docker. Il utilise Ansible pour orchestrer et gérer les tâches nécessaires à la configuration et au déploiement des différents composants d'OpenStack dans des conteneurs. Cela permet de rendre le déploiement d'OpenStack plus rapide, plus reproductible et plus facile à gérer.

### Caractéristiques clés de Kolla-Ansible

**Conteneurisation** : Kolla-Ansible utilise des conteneurs Docker pour encapsuler les différents services et composants d'OpenStack. Chaque composant d'OpenStack, tel que Nova, Neutron, Cinder, etc., est exécuté dans un conteneur séparé. Cela permet d'isoler les services et de simplifier la gestion et la mise à l'échelle.

**Gestion de configuration déclarative** : Kolla-Ansible utilise des fichiers de configuration YAML pour décrire l'état désiré du déploiement OpenStack. Ces fichiers décrivent les services à déployer, les images Docker à utiliser, les paramètres de configuration, etc. Ansible utilise ces fichiers pour orchestrer et gérer les tâches de déploiement.

**Personnalisation** : Kolla-Ansible permet de personnaliser le déploiement en fournissant des options de configuration flexibles. Vous pouvez spécifier les paramètres de configuration spécifiques à votre environnement, tels que les adresses IP, les noms d'hôte, les réseaux, etc. Cela vous permet d'adapter le déploiement d'OpenStack en fonction de vos besoins.

**Extensibilité** : Kolla-Ansible est conçu pour être extensible. Vous pouvez ajouter ou personnaliser des images Docker, des services ou des configurations spécifiques pour répondre à vos besoins particuliers. Cela permet une plus grande flexibilité lors du déploiement et de la gestion d'OpenStack.

En utilisant Kolla-Ansible, vous pouvez simplifier le processus de déploiement d'OpenStack, réduire les erreurs et faciliter la gestion à long terme de votre infrastructure OpenStack. Il offre une approche basée sur des conteneurs et utilise Ansible comme outil d'orchestration pour automatiser les tâches complexes associées au déploiement d'OpenStack.

## Tp -Déployer Openstack avec kolla

### Exigences relatives à la machine hôte

Une VM Ubuntu ou type Redhat (Rocky, centos)

La machine hôte doit satisfaire aux exigences minimales suivantes :

- 2 interfaces réseau
- 8 Go de mémoire principale
- 40 Go d'espace disque

### Installer les dépendances

En général, les commandes qui utilisent le gestionnaire de paquets du système dans cette section doivent être exécutées avec les privilèges de l'utilisateur root.

Il est généralement recommandé d'utiliser un environnement virtuel pour installer Kolla Ansible et ses dépendances, afin d'éviter les conflits avec les paquets du système

Pour Debian ou Ubuntu, mettez à jour l'index des paquets.

```
ludo@openstack:~$ sudo apt update
```

### Installez les dépendances de construction de Python :

CentOS, Rocky ou openEuler, exécutez :

```
ludo@openstack:~$ sudo dnf install git python3-devel libffi-devel gcc  
openssl-devel python3-libselinux
```

Pour Debian ou Ubuntu, exécutez

```
ludo@openstack:~$ sudo apt install git python3-dev libffi-dev gcc  
libssl-dev
```

Installer les dépendances à l'aide d'un environnement virtuel

Pour CentOS, Rocky, vous n'avez rien à faire.

Pour Debian ou Ubuntu, lancez

```
ludo@openstack:~$ sudo apt install python3-venv
```

Créez un environnement virtuel et activez-le :

```
ludo@openstack:~$ python3 -m venv /path/to/venv  
ludo@openstack:~$ source /chemin/vers/venv/bin/activate
```

L'environnement virtuel doit être activé avant d'exécuter des commandes qui dépendent de paquets installés dans cet environnement.

Assurez-vous que la dernière version de pip est installée :

```
ludo@openstack:~$ pip install -U pip
```

Installer Ansible. Kolla Ansible nécessite au moins une version 4 et supporte jusqu'à 5.

```
ludo@openstack:~$ pip install 'ansible-core>=2.13,<=2.14.2'  
ludo@openstack:~$ pip install 'ansible>=6,<8'
```

## Installer Kolla-ansible

### Installer Kolla-ansible pour le déploiement ou l'évaluation

```
ludo@openstack:~$ pip install git+https://opendev.org/openstack/kolla-ansible@master
```

Créez le répertoire /etc/kolla.

```
ludo@openstack:~$ sudo mkdir -p /etc/kolla
ludo@openstack:~$ sudo chown $USER:$USER /etc/kolla
```

Copiez les fichiers globals.yml et passwords.yml dans le répertoire /etc/kolla.

```
ludo@openstack:~$ cp -r share/kolla-ansible/etc_examples/kolla/* \
    /etc/kolla
```

Copiez les fichiers d'inventaire tout-en-un et multinode dans le répertoire actuel.

```
ludo@openstack:~$ cp share/kolla-ansible/ansible/inventory/* .
```

## Installer Kolla pour le développement

Clonez le dépôt kolla-ansible depuis git.

```
ludo@openstack:~$ git clone --branch master \
    https://opendev.org/openstack/kolla-ansible
```

Installez les dépendances de kolla et de kolla-ansible :

```
ludo@openstack:~$ pip install ./kolla-ansible
```

## Installer les dépendances d'Ansible Galaxy¶

```
ludo@openstack:~$ kolla-ansible install-deps
```

## Configurer Ansible

Pour de meilleurs résultats, la configuration d'Ansible doit être adaptée à votre environnement. Par exemple, ajoutez les options suivantes au fichier de configuration d'Ansible **/etc/ansible/ansible.cfg** :

```
[defaults]
host_key_checking=False
pipelining=True
forks=100
```

## Préparer la configuration initiale

### Inventaire

L'étape suivante consiste à préparer un fichier d'inventaire. Un inventaire est un fichier Ansible dans lequel nous spécifions les hôtes et les groupes auxquels ils appartiennent. Nous pouvons l'utiliser pour définir les rôles des nœuds et les identifiants d'accès.

Kolla Ansible propose des exemples de fichiers d'inventaire all-inone et multinode. La différence entre eux est que le premier est prêt pour le déploiement d'un seul nœud OpenStack sur localhost. Si vous avez besoin d'utiliser un hôte séparé ou plus d'un nœud, éditez l'inventaire multinode :

Modifiez la première section de multinode avec les détails de connexion de votre environnement, par exemple :

### Mots de passe Kolla

Les mots de passe utilisés dans notre déploiement sont stockés dans le fichier **/etc/kolla/passwords.yml**. Tous les mots de passe sont vides dans ce



fichier et doivent être remplis soit manuellement, soit en utilisant un générateur de mots de passe aléatoires :

Pour le déploiement ou l'évaluation, exécutez :

```
ludo@openstack:~$ kolla-genpwd
```

## Kolla globals.yml

globals.yml est le fichier de configuration principal de Kolla Ansible. Il y a quelques options qui sont nécessaires pour déployer Kolla Ansible :

```
ludo@openstack:~$ vim /etc/kolla/globals.yml
kolla_base_distro: "ubuntu"
kolla_internal_vip_address: "10.0.0.200"
network_interface: "enp1s0"
neutron_external_interface: "enp7s0"
enable_ceilometer: "yes"
enable_cinder: "yes"
enable_cinder_backend_lvm: "yes"
enable_cinder_backend_nfs: "yes"
enable_designate: "yes"
enable_gnocchi: "yes"
enable_magnum: "yes"
enable_neutron_vpnaas: "yes"
enable_octavia: "yes"
enable_skyline: "yes"
enable_trove: "yes"
gnocchi_backend_storage: "{% if enable_swift | bool %}swift{% else %}file{%
endif %}"
octavia_auto_configure: no
octavia_certs_country: FR
octavia_certs_state: Bretagne
octavia_certs_organization: OpenStack
octavia_certs_organizational_unit: Octavia
```

Kolla offre le choix entre plusieurs distributions Linux dans les conteneurs :

## Réseaux

Kolla Ansible nécessite quelques options de mise en réseau. Nous devons définir les interfaces réseau utilisées par OpenStack.

La première interface à définir est "network\_interface". C'est l'interface par défaut pour plusieurs réseaux de type gestion.

```
network_interface : "enp1s0"
```

La deuxième interface requise est dédiée aux réseaux externes (ou publics) de Neutron, elle peut être vlan ou plate, cela dépend de la façon dont les réseaux sont créés. Cette interface doit être active sans adresse IP. Si ce n'est pas le cas, les instances ne pourront pas accéder aux réseaux externes.

```
neutron_external_interface: "enp7s0"
```

Ensuite, nous devons fournir une IP flottante pour le trafic de gestion. Cette IP sera gérée par keepalived pour fournir une haute disponibilité, et devrait être configurée pour être une adresse non utilisée dans le réseau de gestion qui est connecté à notre interface\_réseau.

```
kolla_internal_vip_address: "10.0.0.200"
```

## Activer des services supplémentaires

Par défaut, Kolla Ansible fournit le compute nu, mais il prend en charge une vaste sélection de services supplémentaires. Pour les activer, définissez enable\_\* à "yes". Par exemple, pour activer le service Block Storage :

```
enable_magnum: "yes"  
enable_neutron_vpnaas: "yes"  
enable_octavia: "yes"
```

il y a une liste des services disponibles. Pour plus d'informations sur la configuration des services, veuillez vous référer au Guide de référence des services.

### Creation du nfs et lvm pour le stockage

```
ludo@openstack:~$ sudo apt install -y nfs-server

ludo@openstack:~$ sudo mkdir -p /srv/cinder

ludo@openstack:~$ sudo echo "/srv/cinder \
*(rw,sync,no_root_squash,no_subtree_check) " > /etc/exports

ludo@openstack:~$ sudo systemctl start nfs-utils

ludo@openstack:~$ sudo mkdir /etc/kolla/config

ludo@openstack:~$ echo "192.168.232.100:/srv/cinder" > \
/etc/kolla/config/nfs_shares
```

### Creation du nfs et lvm pour le stockage

```
ludo@openstack:~$ sudo vgcreate cinder-volumes /dev/sdb
```

## Déploiement

Une fois la configuration effectuée, nous pouvons passer à la phase de déploiement. Tout d'abord, nous devons configurer les dépendances de base au niveau de l'hôte, comme docker.

Kolla Ansible fournit un playbook qui installera tous les services requis dans les bonnes versions.

Pour le développement, exécutez :

Démarrer les serveurs avec les dépendances de kolla deploy :

```
ludo@openstack:~$ source kolla/bin/activate
ludo@openstack:~$ kolla-ansible -i all-in-one bootstrap-servers
```

Effectuer les vérifications de pré-déploiement pour les hôtes :

```
ludo@openstack:~$ kolla-ansible -i all-in-one prechecks
```

Enfin, procédez au déploiement OpenStack :

```
ludo@openstack:~$ kolla-ansible -i all-in-one deploy
```

Lorsque ce playbook se termine, OpenStack devrait être opérationnel et fonctionnel ! Si une erreur survient pendant l'exécution, reportez-vous au guide de dépannage.

### Utilisation de la CLI d'OpenStack

Installez le client CLI d'OpenStack :

```
ludo@openstack:~$ pip install python-openstackclient
```

OpenStack nécessite un fichier `clouds.yaml` dans lequel les informations d'identification de l'utilisateur admin sont définies. Pour générer ce fichier :

Pour le déploiement ou l'évaluation, exécutez :

```
ludo@openstack:~$ kolla-ansible post-deploy
```

Le fichier sera généré dans `/etc/kolla/clouds.yaml`, vous pouvez l'utiliser en le copiant dans `/etc/openstack` ou `~/.config/openstack` ou en définissant la variable d'environnement `OS_CLIENT_CONFIG_FILE`.

Selon la façon dont vous avez installé Kolla Ansible, il y a un script qui créera des exemples de réseaux, d'images, etc.

Vous êtes libre d'utiliser le script **init-runonce** à des fins de démonstration, mais notez qu'il n'est pas nécessaire de l'exécuter pour utiliser votre cloud. En fonction de vos personnalisations, il se peut qu'il ne fonctionne pas ou qu'il entre en conflit avec les ressources que vous souhaitez créer. Vous êtes prévenu.

```
ludo@openstack:~$ kolla-ansible/tools/init-runonce
```

Assurez vous que votre interface enp7s0 et UP

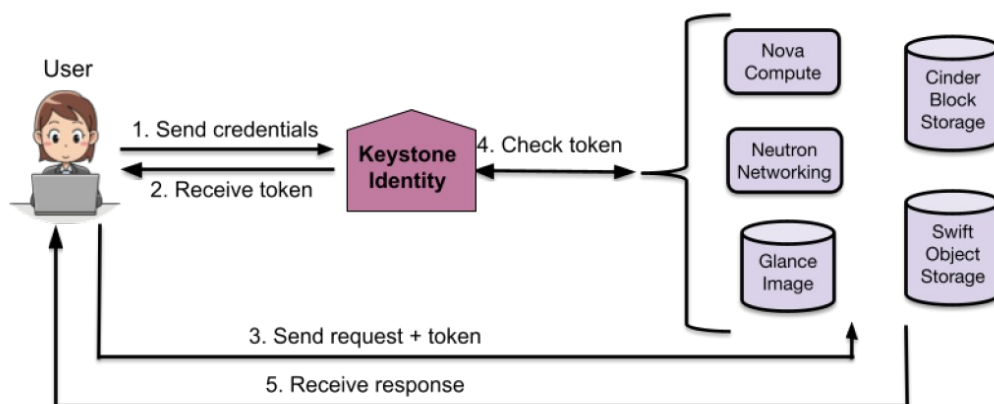
```
ludo@openstack:~$ sudo ip link set enp7s0 up
```

## Keystone

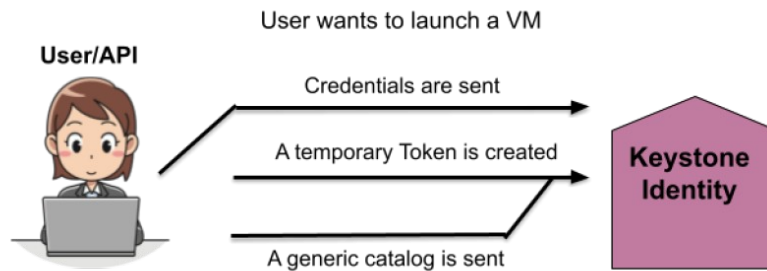


### Introduction

Keystone est le service d'identité dans le projet OpenStack. Il joue un rôle essentiel en fournissant l'authentification et l'autorisation aux utilisateurs, ainsi qu'en gérant les informations d'identification, les rôles et les projets au sein du système OpenStack.

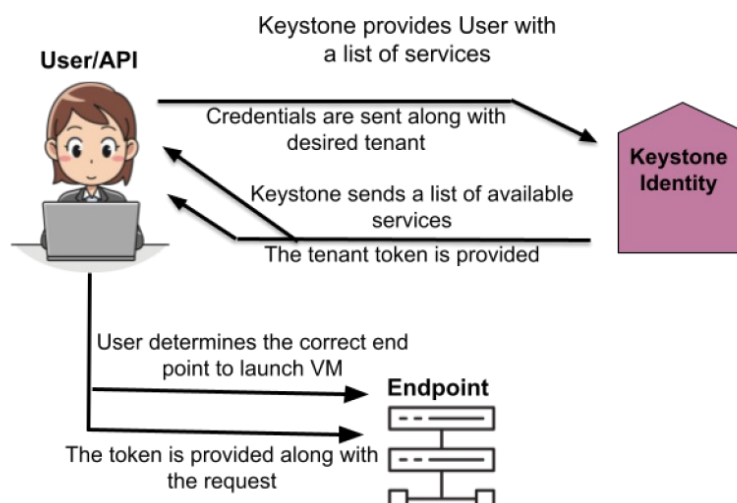


**Authentification des utilisateurs :** Keystone permet aux utilisateurs de s'authentifier auprès du système OpenStack en utilisant différents mécanismes d'authentification, tels que l'authentification par nom d'utilisateur/mot de passe, l'authentification par clé API, l'authentification basée sur un jeton, l'authentification à deux facteurs, etc. Il vérifie les informations d'identification fournies par les utilisateurs et délivre un jeton d'authentification valide en cas de succès.

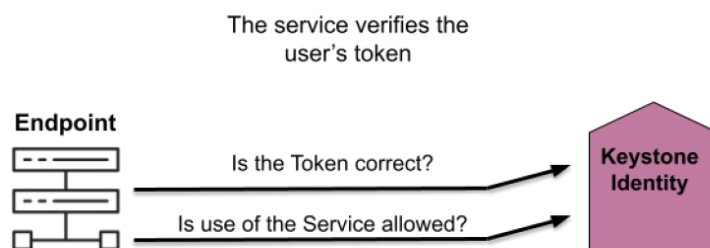


**Autorisation des utilisateurs :** Une fois authentifiés, les utilisateurs doivent obtenir une autorisation appropriée pour accéder aux ressources et effectuer des actions dans le système OpenStack. Keystone gère les rôles et les politiques d'accès pour contrôler les autorisations des utilisateurs. Les rôles définissent les ensembles de privilèges associés à des utilisateurs spécifiques, tandis que les politiques définissent les règles qui déterminent les actions autorisées pour chaque utilisateur en fonction de son rôle et de son projet.

**Gestion des utilisateurs, des groupes et des projets :** Keystone permet la création, la mise à jour et la suppression des utilisateurs, des groupes et des projets dans OpenStack. Il maintient une base de données d'utilisateurs qui comprend leurs informations d'identification, leurs rôles, leurs projets et d'autres attributs pertinents. Cela facilite la gestion centralisée des utilisateurs et de leurs permissions à travers le système OpenStack.



**Services d'identification des endpoints** : Keystone fournit également un catalogue des services disponibles dans le déploiement OpenStack. Il enregistre les endpoints (points d'extrémité) de chaque service, qui sont les URL permettant d'accéder aux différentes fonctionnalités fournies par les services OpenStack tels que Compute (Nova), Networking (Neutron), Image (Glance), etc. Ce catalogue permet aux utilisateurs et aux services de découvrir les fonctionnalités disponibles dans le déploiement OpenStack.



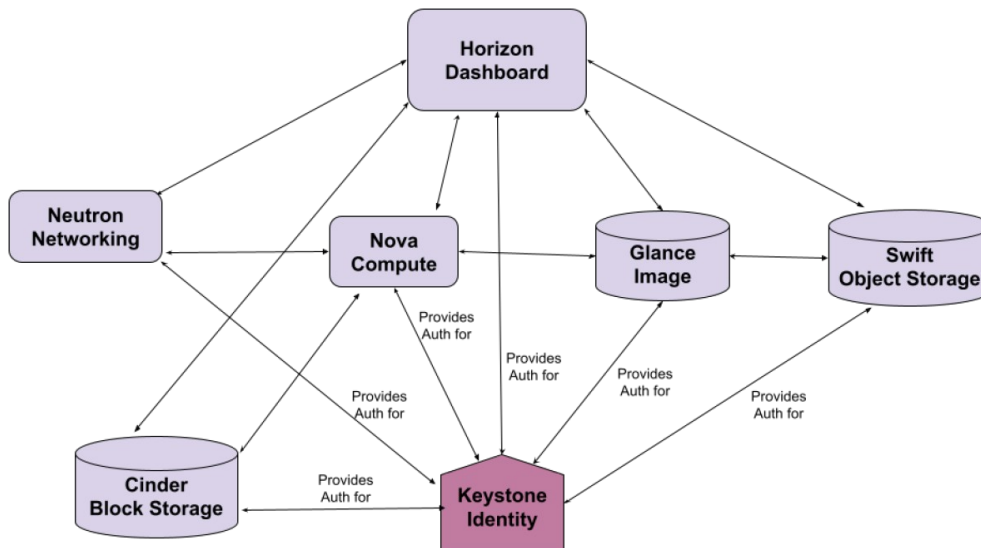
**Intégration avec d'autres services** : Keystone joue un rôle d'intégration clé en permettant l'interopérabilité entre les différents services OpenStack. Il fournit un point d'entrée commun pour l'authentification et l'autorisation, permettant aux autres services de s'appuyer sur lui pour gérer ces fonctionnalités. Ainsi, les services tels que Nova, Neutron, Glance, Cinder, etc., peuvent s'intégrer avec Keystone pour garantir une gestion centralisée et cohérente des identités et des autorisations.

Keystone est un service d'identité essentiel dans OpenStack qui fournit l'authentification, l'autorisation et la gestion des identités. Il permet aux utilisateurs de se connecter au système, d'obtenir les autorisations appropriées et de gérer leurs informations d'identification. Il joue également un rôle clé dans l'intégration des services OpenStack en fournissant un point d'entrée commun pour l'authentification et l'autorisation.

Les applications peuvent discuter avec Keystone, le service d'identité d'OpenStack, en utilisant son API RESTful. L'API Keystone expose un ensemble d'endpoints (URL) auxquels les applications peuvent envoyer des requêtes HTTP pour interagir avec le service.



Les principales étapes pour qu'une application communique avec Keystone :



**Récupération du token d'authentification** : Avant de pouvoir accéder aux autres fonctionnalités d'OpenStack, l'application doit obtenir un token d'authentification valide auprès de Keystone. Cela se fait généralement en envoyant une requête d'authentification à l'API Keystone, fournissant les informations d'identification appropriées (par exemple, nom d'utilisateur/mot de passe) ou un jeton existant. Une fois l'authentification réussie, Keystone renvoie un token d'authentification à l'application.

**Inclusion du token d'authentification** : Une fois que l'application dispose du token d'authentification, elle doit l'inclure dans chaque requête envoyée à Keystone. Cela se fait généralement en ajoutant l'en-tête HTTP "X-Auth-Token" avec la valeur du token d'authentification. Ainsi, Keystone peut identifier et valider l'identité de l'application lorsqu'elle interagit avec les autres services OpenStack.

**Envoi des requêtes aux endpoints Keystone** : L'application peut envoyer des requêtes HTTP aux différents endpoints de l'API Keystone pour effectuer des opérations telles que la gestion des utilisateurs, des projets, des rôles, des politiques, etc. Les requêtes sont généralement formatées en utilisant des méthodes HTTP standard telles que GET, POST, PUT, DELETE, etc., avec les paramètres appropriés et les corps de requête JSON.

**Traitement des réponses** : Après avoir envoyé une requête à Keystone, l'application reçoit une réponse HTTP contenant les résultats de la demande. Les réponses peuvent inclure des informations telles que les détails de l'utilisateur, les projets disponibles, les rôles assignés, les politiques d'accès, etc. L'application peut analyser et traiter les réponses reçues pour effectuer les actions nécessaires.

Il est important de noter que pour interagir avec Keystone, l'application doit être capable de construire et d'envoyer des requêtes HTTP, ainsi que de traiter les réponses HTTP reçues. Pour faciliter cette interaction, de nombreux langages de programmation fournissent des bibliothèques ou des SDK OpenStack qui encapsulent les appels à l'API Keystone et fournissent des méthodes simplifiées pour effectuer les opérations courantes. Ces bibliothèques peuvent faciliter le processus de communication avec Keystone pour les développeurs d'applications.

## TP – Gestion des projets, utilisateurs et rôles

Connexion à OpenStack Keystone

Ouvrez un terminal.

Utilisez les informations d'identification de votre installation OpenStack pour vous connecter à Keystone en utilisant la commande suivante :

```
ludo@openstack:~$ source admin-openrc.sh
```

### Création d'un projet

Utilisez la commande suivante pour créer un projet :

```
openstack project create --description "Mon projet Formation" formation
```

### Création de l'utilisateur membre

Créez un utilisateur administrateur :

```
openstack user create --project formation --password rootroot ludo
```

Accordez les privilèges Membre à l'utilisateur :

```
openstack role list

openstack role add --project formation --user ludo member

openstack role assignment list --user ludo --project formation

for assignment in $(openstack role assignment list --user ludo --project formation -f value -c Role); do openstack role show $assignment -f value -c name; done
```

## Création de l'utilisateur administrateur


Créez un utilisateur administrateur :

```
openstack user create --project formation --password rootroot adminforma
```

Accordez les privilèges d'administrateur à l'utilisateur :

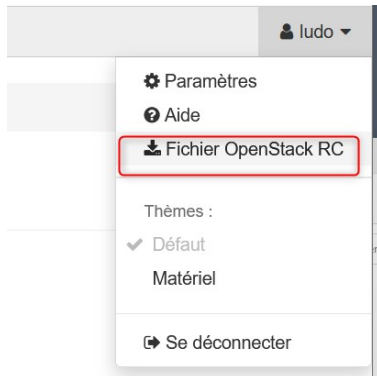
```
openstack role list  
  
openstack role add --project formation --user adminform admin
```

## Se connecter au dashboard



The image shows the OpenStack login interface. At the top is the OpenStack logo, which consists of a red square with a white 'O' inside, followed by the word 'openstack' in a bold, sans-serif font. Below the logo is the text 'Se connecter'. Underneath this is a form with two input fields. The first field is labeled 'Nom d'utilisateur' and contains the text 'ludo'. The second field is labeled 'Mot de passe' and contains a series of dots, indicating a password. To the right of the password field is an eye icon. At the bottom right of the form is a blue button with the text 'Se connecter'.

## Créer un fichier rc



```
#!/usr/bin/env bash
export OS_AUTH_URL=http://10.0.0.200:5000
export OS_PROJECT_ID=3355e91b79c94db9987ad47f6978e221
export OS_PROJECT_NAME="formation"
export OS_USER_DOMAIN_NAME="Default"
if [ -z "$OS_USER_DOMAIN_NAME" ]; then unset OS_USER_DOMAIN_NAME; fi
export OS_PROJECT_DOMAIN_ID="default"
if [ -z "$OS_PROJECT_DOMAIN_ID" ]; then unset OS_PROJECT_DOMAIN_ID; fi
unset OS_TENANT_ID
unset OS_TENANT_NAME
export OS_USERNAME="ludo"
echo "Please enter your OpenStack Password for project $OS_PROJECT_NAME
as user $OS_USERNAME: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
export OS_REGION_NAME="RegionOne"
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3
```

Le copier sur sa machine et le «sourcer»

```
ludo@openstack:~$ source ludorc.sh
```

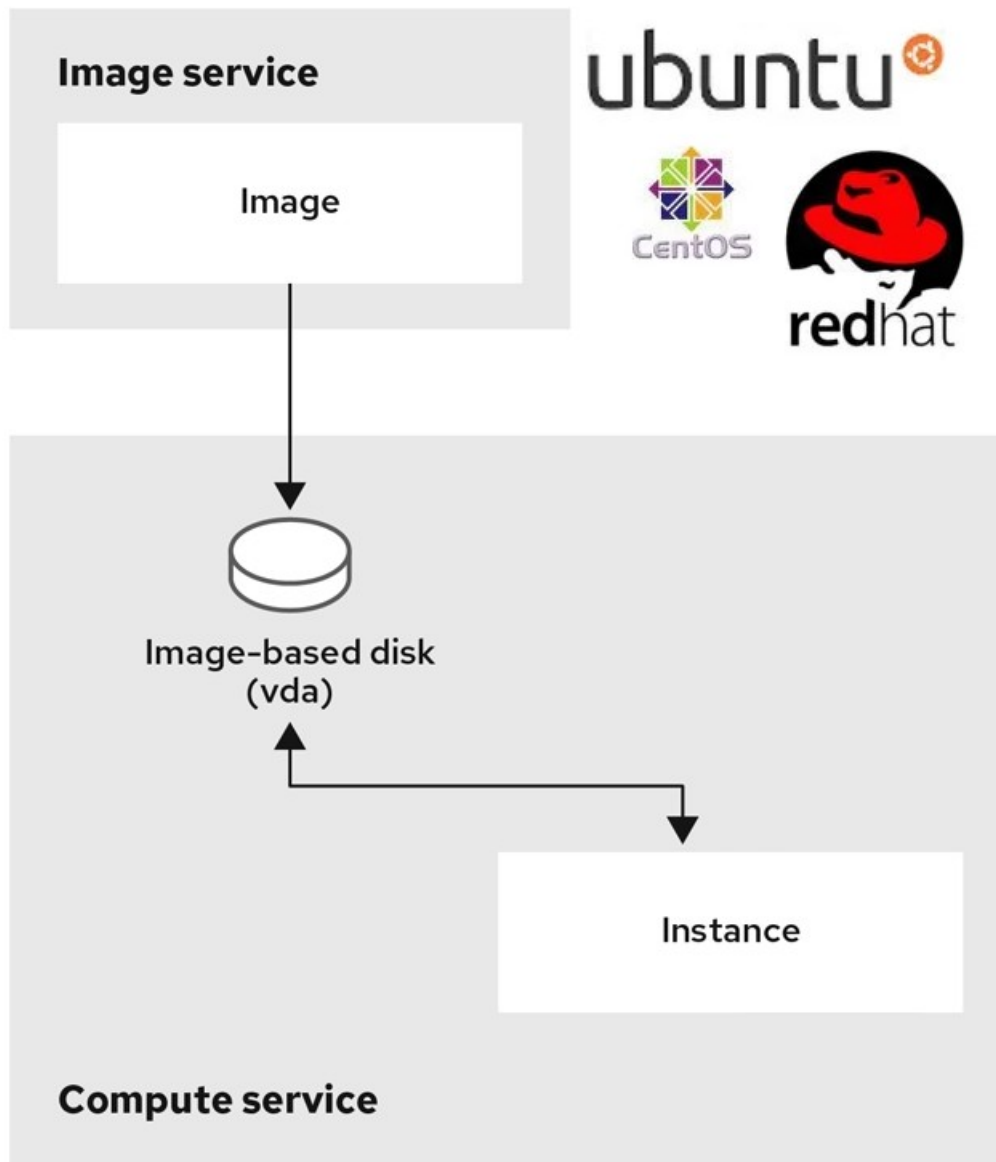
## Le service Glance



## Introduction

Le service Glance est un composant essentiel d'OpenStack qui gère les images de machines virtuelles (VM) et les snapshots de disques. Il fournit un référentiel centralisé pour stocker, gérer et récupérer les images utilisées dans les déploiements OpenStack. Voici une explication détaillée du service Glance et de ses fonctionnalités :

## OpenStack



**Stockage d'images** : Glance est responsable du stockage des images de machines virtuelles (VM) utilisées dans un environnement OpenStack. Ces images peuvent être des images de système d'exploitation, des images d'application ou des images personnalisées créées par les utilisateurs. Glance peut stocker les images localement ou les intégrer à des systèmes de stockage

externes tels que Swift (le service de stockage objet d'OpenStack) ou Ceph (un système de stockage distribué).

**Formats d'image pris en charge :** Glance prend en charge différents formats d'images, y compris les formats couramment utilisés tels que RAW, QCOW2, VMDK, VDI, et VHD. Cela permet aux utilisateurs de créer, importer et utiliser des images dans le format approprié pour leurs machines virtuelles.

**Métadonnées d'image :** Glance permet d'associer des métadonnées aux images stockées. Les métadonnées sont des informations supplémentaires qui décrivent l'image, telles que le nom, la description, la version, les propriétés spécifiques, etc. Ces métadonnées facilitent la recherche et la gestion des images dans Glance.

**Versions et snapshots d'images :** Glance prend en charge la gestion des versions des images. Cela permet aux utilisateurs de mettre à jour ou de créer de nouvelles versions d'une image tout en conservant les versions précédentes. Glance permet également de créer des snapshots d'images, qui sont des instantanés de l'état d'une image à un moment donné. Les snapshots d'images sont utiles pour capturer l'état d'une VM à un instant précis, par exemple, pour effectuer des sauvegardes ou pour créer des clones.

**Catalogue d'images :** Glance maintient un catalogue d'images, également appelé "Glance image catalog", qui répertorie les images disponibles dans le déploiement OpenStack. Ce catalogue permet aux utilisateurs et aux services OpenStack de découvrir et d'accéder facilement aux images disponibles. Le catalogue d'images peut être consulté et géré via l'API Glance.

**Intégration avec d'autres services :** Glance joue un rôle clé dans l'intégration avec d'autres services OpenStack. Par exemple, le service Nova (Compute) utilise Glance pour récupérer les images nécessaires pour lancer des instances de machines virtuelles. Glance fournit également une API qui peut être utilisée par d'autres services pour interagir avec les images et récupérer les métadonnées associées.



Le service Glance est responsable du stockage, de la gestion et de la récupération des images de machines virtuelles dans un environnement OpenStack. Il offre des fonctionnalités telles que le stockage d'images, la prise en charge de différents formats d'images, la gestion des versions et des snapshots, ainsi qu'un catalogue d'images pour faciliter la découverte et l'accès aux images disponibles. Glance joue un rôle crucial dans le déploiement des machines

## TP – Gestion des images

### Utilisation de l'outil QEMU-NBD

QEMU NBD (Network Block Device) est un outil utilisé dans le contexte de la virtualisation pour accéder et manipuler des images de disques virtuels (VDI) au niveau du bloc. Il permet de monter des images de disques virtuels et de les rendre accessibles comme des périphériques de blocs locaux sur le système d'exploitation hôte.

Voici quelques points importants à savoir sur QEMU NBD :

**Accès aux images de disques virtuels :** QEMU NBD fournit une interface permettant d'accéder aux images de disques virtuels utilisées par les machines virtuelles. Ces images peuvent être stockées dans divers formats tels que QEMU Copy-On-Write (qcow), Virtual Hard Disk (VHD), Raw (raw), etc. L'outil QEMU NBD permet de monter ces images pour accéder à leur contenu et effectuer des opérations de lecture/écriture.

**Montage des images de disques virtuels :** Avec QEMU NBD, vous pouvez monter une image de disque virtuel sur le système d'exploitation hôte en tant que périphérique de blocs. Cela permet d'accéder aux fichiers et aux systèmes de fichiers contenus dans l'image comme s'ils étaient présents sur un disque local. Cette fonctionnalité est utile pour effectuer des opérations de maintenance, de récupération de données ou d'exploration des fichiers à l'intérieur de l'image.

**Accès en mode lecture-écriture ou en mode lecture seule :** QEMU NBD prend en charge à la fois le mode lecture-écriture et le mode lecture seule pour monter les images de disques virtuels. Vous pouvez choisir le mode approprié en fonction des besoins. Le mode lecture seule permet de garantir l'intégrité de l'image en empêchant toute modification accidentelle ou non autorisée.

**Utilisation avec d'autres outils de virtualisation :** QEMU NBD est souvent utilisé en conjonction avec d'autres outils de virtualisation tels que QEMU (Quick Emulator) et libvirt. Par exemple, vous pouvez utiliser QEMU NBD pour monter une image de disque virtuel avant de lancer une machine virtuelle avec

QEMU, ou vous pouvez utiliser libvirt pour gérer et contrôler le processus de montage des images.

Téléchargement d'une image Ubuntu cloud

```
ludo@openstack:~$ wget
http://cloud-images.ubuntu.com/releases/22.04/release/ubuntu-22.04-
server-cloudimg-amd64.img
```

Montage de l'image Ubuntu

```
ludo@openstack:~$ sudo modprobe nbd

ludo@openstack:~$ sudo qemu-nbd --connect=/dev/nbd0 ubuntu-22.04-server-
cloudimg-amd64.img

ludo@openstack:~$ sudo mount /dev/nbd0p1 /mnt
```

Le fichier cloud.cfg est un fichier de configuration utilisé dans l'image Ubuntu Cloud pour la personnalisation et la configuration de l'instance lors de son déploiement dans un environnement cloud. Il est généralement utilisé avec les images Ubuntu fournies spécifiquement pour les déploiements sur des plateformes cloud comme OpenStack, AWS, Azure, etc.

Les principales sections et paramètres présents dans le fichier cloud.cfg :

**DataSource** : Cette section définit la source de données utilisée pour récupérer la configuration initiale de l'instance. Par exemple, le paramètre "None" indique qu'aucune source de données n'est utilisée, tandis que le paramètre "Ec2" indique l'utilisation de la source de données EC2 d'Amazon Web Services.

**MimeType** : Cette section spécifie les types de contenu MIME pris en charge lors du traitement des fichiers de configuration cloud-init. Par exemple,

"text/cloud-config" indique que les fichiers avec ce type de contenu seront interprétés comme des fichiers de configuration au format cloud-config.

**CloudConfigModules** : Cette section spécifie les modules cloud-init à exécuter lors de la configuration initiale de l'instance. Chaque module est représenté par un nom de module suivi de ses paramètres éventuels. Par exemple, "locale: en\_US.UTF-8" définit le module "locale" avec le paramètre "en\_US.UTF-8" pour définir la configuration régionale.

**Users** : Cette section permet de définir les utilisateurs qui seront créés sur l'instance lors de son déploiement. Vous pouvez spécifier le nom d'utilisateur, le mot de passe, les clés SSH, les groupes, les identifiants utilisateur, etc.

**SSH** : Cette section spécifie les paramètres de configuration SSH pour l'instance. Par exemple, vous pouvez définir les autorisations pour les clés SSH, activer ou désactiver l'authentification par mot de passe, etc.

**Packages** : Cette section permet de spécifier les paquets logiciels supplémentaires à installer sur l'instance lors de son déploiement.

**Bootcmd** : Cette section spécifie les commandes à exécuter au démarrage de l'instance. Vous pouvez utiliser cette section pour exécuter des scripts, des commandes d'initialisation, etc.

Ces exemples représentent seulement quelques-unes des sections et des paramètres disponibles dans le fichier cloud.cfg. Le fichier cloud.cfg permet une personnalisation détaillée de l'instance lors de son déploiement, en spécifiant des informations telles que les utilisateurs, les clés SSH, les configurations régionales, les paquets à installer, etc. Ces configurations sont prises en compte par le service cloud-init lors de la création de l'instance, lui permettant d'être configurée selon les besoins spécifiques de l'utilisateur.

### Edition du fichier cloud.cfg

```
ludo@openstack:~$ sudo vi /mnt/etc/cloud/cloud.cfg
# line 13 : ajouter

ssh_pwauth: true

# line 99 : changer

system_info:
  distro: ubuntu
  # Default user name + that default users groups (if added/used)
  default_user:
    name: ubuntu
    lock_passwd: False
    gecos: Ubuntu
```

### On démonte l'iso

```
ludo@openstack:~$ sudo umount /mnt

ludo@openstack:~$ sudo qemu-nbd --disconnect /dev/nbd0p1

/dev/nbd0p1 disconnected
```

### Ajouter l'image dans Glance

```
ludo@openstack:~$ openstack image create "Ubuntu2204" --file ubuntu-22.04-server-cloudimg-amd64.img --disk-format qcow2 --container-format bare --public
```

La commande "openstack image create" est utilisée pour créer une nouvelle image dans OpenStack. Voici une explication des différents paramètres utilisés dans la commande :

**"Ubuntu2204"** : C'est le nom donné à l'image. Vous pouvez choisir un nom significatif pour identifier l'image, tel que "Ubuntu 22.04".

**"--file ubuntu-22.04-server-cloudimg-amd64.img"** : Cela spécifie le chemin d'accès et le nom du fichier d'image à importer pour créer l'image dans

OpenStack. Dans cet exemple, le fichier est "ubuntu-22.04-server-cloudimg-amd64.img".

**"--disk-format qcow2"** : Cela indique le format du disque de l'image à importer. Dans cet exemple, le format est qcow2, qui est un format de disque couramment utilisé dans OpenStack pour les images.

**"--container-format bare"** : Cela spécifie le format de conteneur pour l'image. Dans ce cas, le format est "bare", ce qui signifie que l'image n'est pas encapsulée dans un conteneur supplémentaire.

**"--public"** : Cette option indique que l'image créée sera publique, ce qui signifie qu'elle sera accessible à tous les utilisateurs du projet ou du domaine OpenStack.

## Le service Neutron



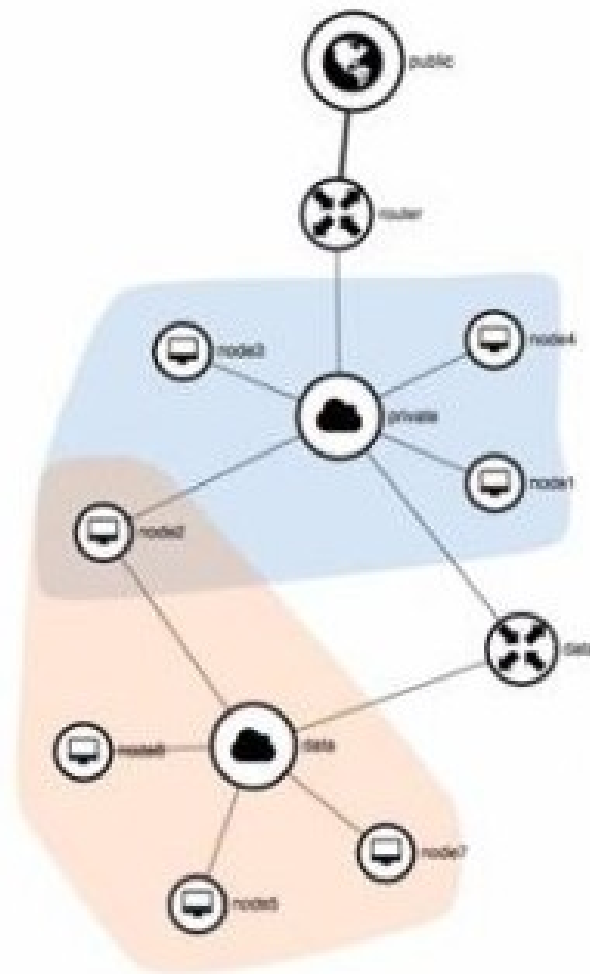
### Introduction

Neutron est un composant clé d'OpenStack qui se charge de la gestion des réseaux et de la connectivité dans un déploiement OpenStack. Il fournit des fonctionnalités avancées de réseau défini par logiciel (SDN) pour la création et la gestion de réseaux virtuels.

Neutron joue un rôle crucial dans la connectivité des instances de machines virtuelles (VM) dans un environnement OpenStack, en leur fournissant un accès réseau et en leur permettant de communiquer avec d'autres instances et ressources.

### Principe

**Réseaux virtuels** : Neutron permet de créer des réseaux virtuels isolés au sein d'un déploiement OpenStack. Ces réseaux virtuels fournissent une isolation logique entre les différentes instances de machines virtuelles (VM) et permettent d'organiser les ressources réseau de manière indépendante.



**Sous-réseaux :** Dans Neutron, vous pouvez créer des sous-réseaux à l'intérieur des réseaux virtuels. Les sous-réseaux définissent les plages d'adresses IP qui peuvent être utilisées par les instances VM dans le réseau virtuel.

**Adresses IP :** Neutron gère l'allocation des adresses IP aux instances VM. Il assure également la résolution des adresses IP (DHCP) pour les instances qui en ont besoin.

**Adresses IP flottantes :** permettent aux instances de machines virtuelles (VM) d'accéder à des réseaux externes et d'être accessibles depuis l'extérieur du déploiement OpenStack.



**Routage** : Neutron permet de configurer des routes entre les réseaux virtuels et de gérer les tables de routage pour permettre la communication entre les instances et les réseaux.

**Pare-feu et règles de sécurité** : Neutron propose des fonctionnalités de pare-feu et de règles de sécurité pour contrôler le trafic réseau. Vous pouvez définir des règles de sécurité spécifiques pour autoriser ou bloquer certains types de trafic réseau entrant ou sortant des instances VM.

**Mécanismes de connectivité** : Neutron offre différents mécanismes de connectivité pour relier les instances VM aux réseaux sous-jacents. Cela inclut des mécanismes tels que les commutateurs virtuels, les VLAN, VXLAN, Geneve.

**Extensions** : Neutron prend en charge des extensions qui permettent d'ajouter des fonctionnalités supplémentaires. Par exemple, l'extension "Load Balancer as a Service" (LBaaS) permet d'intégrer des services d'équilibrage de charge au sein du déploiement OpenStack.

**API Neutron** : Le service Neutron expose une API RESTful qui permet aux utilisateurs et aux services externes d'interagir avec les fonctionnalités de gestion du réseau. L'API Neutron permet de créer, de configurer et de supprimer des réseaux, des sous-réseaux, des règles de sécurité, etc.

Neutron travaille en étroite collaboration avec d'autres composants d'OpenStack, tels que Nova (pour la gestion des instances VM), Cinder (pour le stockage), Keystone (pour l'authentification et l'autorisation), etc. Ensemble, ces composants permettent de créer un environnement OpenStack complet et fonctionnel, avec des fonctionnalités de calcul, de stockage et de réseau intégrées.

## Les réseaux provider Neutron

### le réseau "flat"

le réseau "flat" est l'un des mécanismes de réseau disponibles pour la création de réseaux virtuels dans un déploiement OpenStack. Le réseau flat est utilisé lorsqu'aucune segmentation VLAN ou VXLAN n'est requise et que les instances VM doivent être connectées directement à un réseau physique existant sans isolation supplémentaire.

**Connectivité directe** : Avec le réseau flat, les instances VM sont connectées directement à un réseau physique externe sans utiliser de VLAN ou d'encapsulation VXLAN. Cela signifie que les instances partagent le même domaine de diffusion (broadcast domain) que le réseau physique, et elles obtiennent généralement leurs adresses IP du même sous-réseau que le réseau physique.

**Aucune isolation** : Contrairement aux autres mécanismes de réseau, tels que VLAN ou VXLAN, le réseau flat ne fournit pas d'isolation entre les différentes instances VM. Les instances sur le même réseau flat peuvent communiquer directement entre elles et partager le même trafic réseau.

**Configuration de l'interface de réseau** : Lorsque vous créez un réseau flat, vous devez spécifier l'interface réseau physique sur le nœud de calcul (compute node) qui sera utilisée pour la connectivité du réseau. Cela permet à Neutron de configurer l'interface de réseau physique pour connecter les instances VM au réseau flat.

**Utilisation de ponts** : Pour permettre la connectivité entre les instances VM et le réseau physique, Neutron utilise des ponts réseau sur les nœuds de calcul. Les ponts réseau servent de passerelle entre le réseau physique et le réseau virtuel, permettant ainsi aux paquets de passer entre les deux.

**Configuration du réseau physique** : Lors de l'utilisation du réseau flat, vous devez vous assurer que votre réseau physique est configuré correctement pour permettre la connectivité avec les instances VM. Cela peut impliquer la configuration de l'interface réseau physique, l'attribution d'adresses IP, la configuration des passerelles, etc.

Le réseau flat est utilisé dans des cas d'utilisation où aucune isolation supplémentaire n'est nécessaire entre les instances VM et où les instances doivent être directement connectées à des réseaux physiques existants. Cependant, il convient de noter que le réseau flat ne fournit pas les mêmes fonctionnalités d'isolation et de segmentation que d'autres mécanismes de réseau, tels que VLAN ou VXLAN.

## TP – Création de la topologie réseau

### Création d'un routeur

```
ludo@openstack:~$ openstack router create router
ludo@openstack:~$ openstack router list
ludo@openstack:~$ openstack router show router
```

### Création d' un réseau privé, du sous réseau et connexion au routeur

```
ludo@openstack:~$ openstack network create private
ludo@openstack:~$ openstack network list
```

```
ludo@openstack:~$ openstack subnet create --subnet-range 10.0.2.0/24 \
--network private --gateway 10.0.2.1 --dns-nameserver 8.8.8.8 \
private-subnet
```

```
ludo@openstack:~$ openstack router add subnet router private-subnet
```

### Ajout d'un réseau externe

```
ludo@openstack:~$ openstack network create --external \
--provider-physical-network physnet1 \
--provider-network-type flat public

ludo@openstack:~$ openstack subnet create --no-dhcp \
--allocation-pool start=10.0.0.150,end=10.0.0.200 \
--network public --subnet-range 10.0.0.0/24 \
--dns-nameserver 1.1.1.1 --gateway 10.0.0.254 public-subnet

ludo@openstack:~$ openstack router set --external-gateway public router
```

## Les groupes de sécurité

Les groupes de sécurité (security groups) sont une fonctionnalité importante de Neutron dans OpenStack. Ils permettent de définir des règles de sécurité pour contrôler le trafic réseau entrant et sortant des instances de machines virtuelles (VM) dans un déploiement OpenStack.

**Définition des règles :** Un groupe de sécurité est un ensemble de règles de sécurité qui spécifient les types de trafic réseau autorisés ou bloqués pour les instances VM. Les règles peuvent être basées sur les protocoles (comme TCP, UDP, ICMP), les adresses IP source et destination, les ports source et destination, etc.

**Association avec les instances VM :** Les groupes de sécurité sont associés aux instances VM lors de leur création. Chaque instance peut être associée à un ou plusieurs groupes de sécurité. Lorsqu'une instance est associée à un groupe de sécurité, les règles de ce groupe sont appliquées au trafic réseau de l'instance.

**Filtrage du trafic :** Lorsqu'un paquet de données entre ou sort d'une instance VM, le filtrage du trafic est effectué en fonction des règles définies dans les groupes de sécurité associés à l'instance. Les paquets qui correspondent aux règles autorisées sont acceptés, tandis que ceux qui ne correspondent à aucune règle sont généralement rejetés.

**Règles par défaut :** Les groupes de sécurité peuvent avoir des règles par défaut, qui spécifient le comportement par défaut pour le trafic réseau. Par exemple, un groupe de sécurité peut avoir une règle par défaut qui bloque tout le trafic entrant, sauf celui spécifiquement autorisé par d'autres règles.

**Règles prioritaires :** Lorsqu'une instance est associée à plusieurs groupes de sécurité, les règles sont appliquées dans un ordre de priorité. Les règles les plus spécifiques ou les plus restrictives sont généralement appliquées en premier.

**Modification des règles :** Les règles des groupes de sécurité peuvent être modifiées à tout moment pour ajuster les autorisations de trafic réseau. Vous

pouvez ajouter, supprimer ou mettre à jour les règles existantes en utilisant la CLI OpenStack ou l'interface utilisateur.

Les groupes de sécurité offrent un moyen puissant de contrôler la connectivité et la sécurité des instances VM dans un déploiement OpenStack. Ils permettent de limiter l'accès réseau aux ressources et de prévenir les vulnérabilités potentielles en bloquant le trafic non autorisé.

### Création des security groups

```
ludo@openstack:~$ openstack security group create secgroup01

ludo@openstack:~$ openstack security group rule create --protocol \
icmp --ingress secgroup01

ludo@openstack:~$ openstack security group rule create --protocol tcp \
--dst-port 22:22 secgroup01

ludo@openstack:~$ openstack security group rule list secgroup01
```

## Les adresses IP flottantes (floating IP)

Les floating ip sont gérées par le service Neutron dans OpenStack. Les adresses IP flottantes permettent aux instances de machines virtuelles (VM) d'accéder à des réseaux externes et d'être accessibles depuis l'extérieur du déploiement OpenStack.

**Création de l'adresse IP flottante** : Un utilisateur peut créer une adresse IP flottante en utilisant l'API Neutron ou une interface utilisateur (UI) OpenStack. L'adresse IP flottante est associée à un projet et est généralement obtenue à partir d'un pool d'adresses IP configuré.

**Association avec une instance VM :** L'adresse IP flottante peut être associée à une instance VM spécifique. Cela permet à l'instance d'être accessible à partir du réseau externe associé à l'adresse IP flottante. L'association peut être réalisée via l'API Neutron ou l'interface utilisateur.

**Mappage d'adresse réseau (NAT) :** Lorsqu'une adresse IP flottante est associée à une instance VM, Neutron effectue un mappage d'adresse réseau (NAT) pour rediriger le trafic réseau vers l'instance. Le trafic destiné à l'adresse IP flottante est redirigé vers l'adresse IP interne de l'instance VM.

**Connectivité externe :** Une fois l'adresse IP flottante associée à une instance VM, l'instance est accessible depuis le réseau externe associé à l'adresse IP flottante. Cela permet aux utilisateurs d'accéder à l'instance VM depuis l'extérieur du déploiement OpenStack, par exemple pour accéder à une application web ou à un service exécuté sur l'instance.

L'utilisation d'adresses IP flottantes est courante pour les cas d'utilisation tels que l'hébergement de sites web, l'accès à distance aux instances VM, l'équilibrage de charge, etc. Cela permet une connectivité flexible et contrôlée entre les instances VM et les réseaux externes.

### Création d' ip flotantes

```
ludo@openstack:~$ openstack floating ip create public
ludo@openstack:~$ openstack floating ip list
```

## TP – Création de la topologie réseau pour le projet formation

Réaliser la configuration réseau pour le projet formation.

- Création d' un routeur
- Creation d' un réseau privé et son subnet
- Attachement du reseau privé au routeur
- Ajout du routeur aux réseau public
- Ajout de groupe de sécurité
- Création d'ip flottantes pour le réseau

L' utilisateur est «adminforma»

```
ludo@openstack:~$ openstack router list
+-----+-----+-----+-----+
| ID                               | Name           | Status | State |
+-----+-----+-----+-----+
| 6635e702-a667-4067-85d1-9f3277d94de0 | router-forma   | ACTIVE | UP    |
+-----+-----+-----+-----+

ludo@openstack:~$ openstack network list
+-----+-----+-----+
| ID                               | Name           | Subnets
+-----+-----+-----+
| 1d09e85d-4be9-4a53-a18a-97c0dcc32cc7 | public         | 4df92d3f-1942-40c5-
| 6bb511b0-f438-4b81-b4fe-3ed18cf7b396 | formation      | ccb06f2c-e5ef-405a-

ludo@openstack:~$ openstack subnet list
+-----+-----+-----+
+-----+-----+-----+
| ID                               | Name           | Network
| Subnet                          |
+-----+-----+-----+
+-----+-----+-----+
```



## Formation ILCQ : Openstack Utilisateur

```
| ccb06f2c-e5ef-405a-8027-8961672590af | formation-subnet |
```

```
ludo@openstack:~$ openstack subnet show formation-subnet
```

Field	Value
allocation_pools	192.168.100.2-192.168.100.254
cidr	192.168.100.0/24

```
ludo@openstack:~$ openstack security group list
```

ID	Name	Description
968fdd9f-bfea-4535-b89d-0529c298a66b	secgroup-forma	secgroup-forma

```
ludo@openstack:~$ openstack floating ip list
```

ID	Floating IP Address
ea5f794b-01d7-4730-a119-7842a8c72815	10.0.0.167

## Le service Compute Nova



### Introduction

Nova est l'un des principaux services d'OpenStack. Il est responsable de la gestion et de l'orchestration des instances de machines virtuelles (VM) dans un déploiement OpenStack. Nova fournit une interface unifiée pour le déploiement et la gestion des ressources de calcul dans un environnement de cloud computing.

**Gestion des instances :** Nova permet de créer, démarrer, arrêter, redémarrer, supprimer et gérer les instances de machines virtuelles dans OpenStack. Il fournit une API qui permet aux utilisateurs et aux services externes d'interagir avec le service pour effectuer ces opérations.

**Orchestration des ressources de calcul :** Nova gère la planification et l'allocation des ressources de calcul, telles que les CPU, la mémoire et le stockage, pour les instances VM. Il assure également l'équilibrage de charge en répartissant les instances sur différents nœuds de calcul (compute nodes) pour optimiser l'utilisation des ressources disponibles.

**Isolation des instances** : Nova offre une isolation entre les instances VM en s'assurant qu'elles s'exécutent dans des environnements isolés les uns des autres. Cela permet aux instances d'exécuter différents systèmes d'exploitation et de garantir la sécurité et la confidentialité des charges de travail des utilisateurs.

**Évolutivité horizontale** : Nova permet l'évolutivité horizontale en permettant le dimensionnement automatique des instances VM en fonction des besoins de charge de travail. Il est possible de définir des politiques d'évolutivité horizontale pour augmenter ou diminuer le nombre d'instances en fonction de critères prédéfinis, tels que l'utilisation du CPU ou la charge réseau.

**Interactions avec d'autres services** : Nova interagit avec d'autres services d'OpenStack pour fournir une expérience de cloud computing complète. Par exemple, il utilise le service de réseau Neutron pour la connectivité réseau des instances, le service de stockage Cinder pour la gestion des volumes de stockage, et le service d'identité Keystone pour l'authentification et l'autorisation.

Nova joue un rôle essentiel dans la gestion des instances de machines virtuelles dans un déploiement OpenStack. Il offre une interface unifiée pour le déploiement, l'orchestration, la gestion et l'évolutivité des ressources de calcul, offrant ainsi une plateforme de cloud computing puissante et flexible.

## Les instances

Une instance OpenStack, également appelée instance de machine virtuelle (VM) ou server, est une entité informatique qui représente un système d'exploitation et des ressources matérielles virtuelles exécutées sur un nœud de calcul (compute node) d'un déploiement OpenStack. Les instances sont

créées à partir d'images de machine virtuelle et peuvent être configurées pour répondre aux besoins spécifiques des utilisateurs.

**Création d'une instance** : Les instances sont créées à l'aide de l'API OpenStack, de la CLI (interface de ligne de commande) ou de l'interface utilisateur OpenStack. Lors de la création d'une instance, vous devez spécifier des détails tels que l'image de machine virtuelle à utiliser, la taille des ressources (CPU, mémoire, disque) et les réseaux auxquels l'instance sera connectée.

**Les images de machine virtuelle** : Les instances sont créées à partir d'images de machine virtuelle, qui sont des instantanés du système d'exploitation et des logiciels installés. OpenStack prend en charge différents formats d'image, tels que QCOW2, RAW, VHD, etc. Vous pouvez utiliser des images prédéfinies fournies par OpenStack ou créer vos propres images personnalisées.

**Les flavors** : sont des descriptions prédéfinies de la taille et de la configuration des ressources, telles que le nombre de CPU, la quantité de mémoire et la taille du disque, disponibles pour les instances. Ils simplifient le processus de déploiement en permettant aux utilisateurs de spécifier facilement les caractéristiques des instances sans avoir à fournir des détails techniques spécifiques.

**Gestion des cycles de vie** : Les instances OpenStack peuvent être gérées à travers différentes étapes de leur cycle de vie. Vous pouvez démarrer, arrêter, redémarrer, suspendre, reprendre et supprimer des instances. De plus, vous pouvez effectuer des opérations de mise à l'échelle horizontale pour augmenter ou diminuer le nombre d'instances en fonction des besoins de votre application.

**Accès aux instances** : Une fois qu'une instance est créée, vous pouvez y accéder à distance pour effectuer des tâches de configuration, d'administration et de

développement. Cela peut être réalisé via des protocoles tels que SSH (Secure Shell) pour les systèmes Linux ou RDP (Remote Desktop Protocol) pour les systèmes Windows.

Les instances OpenStack offrent une flexibilité et une extensibilité permettant aux utilisateurs de déployer et de gérer des systèmes d'exploitation virtualisés dans un environnement de cloud computing. Elles sont essentielles pour exécuter des applications, héberger des services et fournir des ressources informatiques à la demande.

## **Les flavors**

Les flavors sont un mécanisme pratique pour configurer les ressources d'une instance dans OpenStack. Ils permettent aux utilisateurs de spécifier rapidement et facilement les caractéristiques des instances, en fournissant des profils prédéfinis ou en créant des flavors personnalisés. Cela facilite le déploiement et la gestion des ressources de calcul dans un environnement OpenStack.

Les flavors sont définis par l'administrateur du cloud OpenStack. Ils représentent différents profils de ressources qui peuvent être utilisés pour créer des instances. Par exemple, un flavor peut être défini avec 2 vCPU, 4 Go de mémoire et 50 Go de disque.

Les flavors offrent une flexibilité en matière d'évolutivité horizontale des instances. Vous pouvez redimensionner une instance en changeant son flavor, par exemple en augmentant le nombre de CPU ou en augmentant la quantité de mémoire. Cela permet de s'adapter aux besoins changeants de la charge de travail.

Les flavors disponibles peuvent varier en fonction de la configuration de votre déploiement OpenStack. L'administrateur du cloud peut restreindre les flavors disponibles ou définir des règles spécifiques pour leur utilisation.

### TP – Création d'instance

#### Création des flavors

```
ludo@openstack:~$ openstack flavor create --id 1 --ram 512 --disk 1 \
  --vcpus 1 m1.tiny

ludo@openstack:~$ openstack flavor create --id 2 --ram 2048 --disk 20 \
  --vcpus 1 m1.small

ludo@openstack:~$ openstack flavor create --id 3 --ram 4096 --disk 40 \
  --vcpus 2 m1.medium

ludo@openstack:~$ openstack flavor list
```

#### Création des clés SSH

```
ludo@openstack:~$ ssh-keygen -q -N ""
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):

ludo@openstack:~$ openstack keypair create --public-key \
  ~/.ssh/id_rsa.pub my_ssh_key
```

#### Création d'une instance

```
ludo@openstack:~$ netID=$(openstack network list | grep private \
  | awk '{ print $2 }')

ludo@openstack:~$ openstack server create --flavor m1.small --image
Ubuntu2204 --security-group secgroup01 --nic net-id=$netID --key-name
my_ssh_key Ubuntu-2204

ludo@openstack:~$ openstack server list
```

### Ajout d'une ip flottante

```
ludo@openstack:~$ openstack server add floating ip Ubuntu-2204 \  
10.0.0.167
```

```
ludo@openstack:~$ openstack floating ip show 10.0.0.167
```

```
ludo@openstack:~$ openstack server list
```

### Acces a notre instance

```
ludo@openstack:~$ ping 10.0.0.252
```

```
ludo@openstack:~$ ssh ubuntu@10.0.0.252
```

Nous venons de créer notre premier instance:)

## Cloud init

### Introduction

Cloud-init est un outil puissant pour automatiser la configuration et la personnalisation des instances de machines virtuelles dans les environnements de cloud computing. Il simplifie le processus de démarrage des instances en exécutant des tâches de configuration spécifiques, ce qui permet aux utilisateurs de gagner du temps et d'assurer une cohérence dans le déploiement de leurs applications.

Cloud-init est un largement utilisé dans les environnements de cloud computing pour la configuration initiale et la personnalisation des instances de machines virtuelles. Il s'agit d'un standard, largement pris en charge par de nombreuses plateformes de cloud, y compris OpenStack, AWS, Azure, Google Cloud, et d'autres.

Le but principal de Cloud-init est de faciliter la configuration automatisée des instances de machines virtuelles dès leur démarrage. Il fonctionne en lisant les données de configuration fournies lors de la création de l'instance, telles que les métadonnées, les scripts de démarrage, les clés SSH, etc., et en exécutant ces actions de configuration sur le système d'exploitation invité.

**Métadonnées** : Cloud-init peut extraire les métadonnées fournies par la plateforme de cloud et les utiliser pour personnaliser la configuration de l'instance. Les métadonnées peuvent inclure des informations telles que l'identifiant de l'instance, l'adresse IP, les tags, les informations de réseau, etc.

**Scripts de démarrage** : Cloud-init permet l'exécution de scripts de démarrage personnalisés lors du lancement de l'instance. Ces scripts peuvent contenir des commandes et des configurations supplémentaires spécifiques à l'application ou à l'environnement.



**Configuration système** : Cloud-init peut effectuer diverses tâches de configuration système, telles que la configuration du réseau, la gestion des utilisateurs et des groupes, la personnalisation des fichiers de configuration, l'installation de packages, etc.

**Clés SSH** : Cloud-init peut prendre en charge l'injection de clés SSH, permettant ainsi un accès sécurisé aux instances de machines virtuelles.

Cloud-init utilise généralement des formats de données standard tels que le format YAML ou JSON pour spécifier les actions de configuration. Les fournisseurs de cloud permettent souvent de fournir ces données de configuration lors de la création de l'instance ou en les associant à des métadonnées spécifiques.

### **configuration initiale**

La configuration initiale fait référence aux étapes et processus nécessaires pour préparer une instance de machine virtuelle fraîchement créée avant de pouvoir l'utiliser pleinement. Lorsque vous déployez une nouvelle instance de machine virtuelle, elle est généralement vierge et n'a pas encore été configurée pour répondre à vos besoins spécifiques.

**Configuration du réseau** : Vous devez définir les paramètres réseau de l'instance, tels que l'adresse IP, le masque de sous-réseau, la passerelle par défaut, les DNS, etc. Cela permet à l'instance d'accéder au réseau et à Internet.

**Création des utilisateurs et des autorisations** : Vous pouvez créer des utilisateurs, définir leurs mots de passe et leurs autorisations, et configurer les groupes d'utilisateurs appropriés.

**Installation de logiciels et de packages :** Vous pouvez installer les logiciels, les packages et les dépendances nécessaires à votre application ou à votre environnement.

**Configuration des services :** Vous pouvez configurer les services système, tels que les serveurs web, les bases de données, les serveurs de messagerie, etc., en ajustant leurs paramètres de configuration selon vos besoins.

**Personnalisation des fichiers de configuration :** Vous pouvez modifier les fichiers de configuration de l'instance pour refléter les paramètres spécifiques de votre application ou de votre environnement.

La configuration initiale est essentielle pour préparer l'instance de machine virtuelle à exécuter les tâches spécifiques pour lesquelles elle a été déployée. Cloud-init est un outil qui facilite cette configuration en automatisant ces étapes de configuration initiale, en utilisant les données et les scripts fournis lors de la création de l'instance. Cela permet d'accélérer le processus de mise en service de l'instance et de garantir une configuration cohérente et reproductible.

### **Le format cloud-config**

Le format cloud-config est un format de configuration spécifique utilisé par l'outil Cloud-init pour définir les actions de configuration lors de la création d'instances de machines virtuelles dans les environnements de cloud computing. Il permet de spécifier des directives de configuration de manière déclarative, simplifiant ainsi la personnalisation de l'instance.

Le format cloud-config utilise une syntaxe YAML (YAML Ain't Markup Language) pour décrire les actions de configuration. Voici quelques éléments clés du format cloud-config :

**Directives** : Les directives sont les principales instructions de configuration spécifiées dans le format cloud-config. Elles peuvent inclure des actions telles que la définition des utilisateurs, l'installation de packages, la configuration du réseau, la personnalisation des fichiers de configuration, etc. Chaque directive est identifiée par une clé et peut avoir des valeurs associées.

**Clés et valeurs** : Les clés et les valeurs sont utilisées pour spécifier les propriétés de chaque directive. Les clés représentent les actions ou les paramètres de configuration, tandis que les valeurs leur attribuent des valeurs spécifiques. Par exemple, la clé "packages" peut avoir une valeur qui spécifie une liste de packages à installer.

**Syntaxe de bloc** : Le format cloud-config prend en charge la syntaxe de bloc, permettant d'organiser les directives de manière hiérarchique et structurée. Les blocs sont délimités par des tirets ("-") et des indentations. Les directives peuvent être regroupées dans des blocs pour faciliter la gestion et la lecture du fichier de configuration.

**Commentaires** : Le format cloud-config permet d'inclure des commentaires dans le fichier de configuration en utilisant le caractère dièse ("#"). Les commentaires sont ignorés lors de l'interprétation du fichier et servent à documenter ou à ajouter des notes explicatives.

Voici un exemple simple de fichier de configuration cloud-config en YAML :

```
# Exemple de fichier cloud-config
users:
  - name: monutilisateur
    password: motdepasse
    ssh_authorized_keys:
      - clé_ssh_publique
packages:
  - package1
  - package2
```

```
runcmd:  
- echo "Configuration terminée."
```

Dans cet exemple, nous créons un utilisateur, spécifions des clés SSH autorisées, installons des packages et exécutons une commande.

Le format cloud-config est flexible et permet de spécifier diverses actions de configuration selon les besoins spécifiques de l'instance de la machine virtuelle. En utilisant ce format, vous pouvez personnaliser et configurer facilement vos instances de machines virtuelles dans les environnements de cloud computing.

En plus du format cloud-config utilisé par Cloud-init, il existe d'autres formats de configuration couramment utilisés dans les environnements de cloud computing. Voici quelques-uns d'entre eux :

**JSON (JavaScript Object Notation)** : JSON est un format de données léger et largement utilisé. Il permet de représenter des objets et des données de manière structurée à l'aide de paires clé-valeur. De nombreux outils et services de cloud computing prennent en charge la spécification de la configuration dans un format JSON.

**YAML (YAML Ain't Markup Language)** : YAML est un autre format de données qui est fréquemment utilisé dans les environnements de cloud computing. Il est similaire à JSON mais offre une syntaxe plus concise et lisible par les humains. Cloud-init utilise le format YAML pour spécifier le fichier de configuration cloud-config, mais il peut également être utilisé dans d'autres contextes de configuration.

**Shell scripting** : Les scripts shell sont couramment utilisés pour automatiser les tâches de configuration dans les environnements de cloud computing. Les scripts shell permettent d'exécuter des commandes et des actions spécifiques sur l'instance de la machine virtuelle. Ils peuvent être utilisés pour installer des packages, configurer des services, effectuer des modifications de fichiers, etc.

**Infrastructure-as-Code (IaC)** : L'IaC est une approche de configuration et de gestion des ressources d'infrastructure en utilisant des langages de programmation spécifiques. Des outils tels que **Terraform, AWS CloudFormation, Azure Resource Manager, et Google Cloud Deployment Manager** utilisent des formats spécifiques pour décrire l'infrastructure et les ressources souhaitées, ce qui permet une gestion déclarative et automatisée de l'infrastructure.

Ces formats de configuration offrent une flexibilité pour décrire les actions et les paramètres de configuration dans les environnements de cloud computing. Le choix du format dépend souvent de l'outil ou de la plateforme spécifique utilisé, ainsi que des préférences personnelles et de la facilité d'utilisation.

Déploiement d'une instance OpenStack avec l'utilisateur "ludo", l'installation du serveur Mariadb et démarrage du service Mariadb

```
# Exemple de fichier cloud-config
users:
  - default
ssh_pwauth : true

package_update: true

packages:
  - mariadb-server
  - mariadb-client
system_info:
  distro: ubuntu
  # Default user name + that default users groups (if added/used)
  default_user:
    name: ludo
    lock_passwd: False
    gecos: Ludo cloud User
    groups: [adm, audio, cdrom, dialout, dip, floppy, netdev, plugdev, sudo]
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
runcmd:
  - systemctl enable mariadb-server
  - systemctl start mariadb-server
  - echo "Configuration terminée."
```

## TP – Personnaliser votre instance avec Cloud-init

Utiliser la CLI (Interface de ligne de commande) d'OpenStack pour personnaliser une instance avec Cloud-init implique les étapes suivantes :

- Préparez votre fichier de configuration cloud-init : Créez un fichier de configuration cloud-init en utilisant un éditeur de texte, en vous basant sur l'exemple fourni précédemment ou en personnalisant selon vos besoins.
- Enregistrez le fichier de configuration cloud-init : Enregistrez le fichier de configuration cloud-init dans un emplacement accessible, par exemple sur votre machine locale.
- Connectez-vous à votre environnement OpenStack via la CLI : Ouvrez une session dans votre terminal et utilisez la commande ``openstack`` pour vous connecter à votre environnement OpenStack. Vous devrez fournir les informations d'identification appropriées, telles que le nom d'utilisateur, le mot de passe et l'URL de l'API d'OpenStack.
- Créez une instance en spécifiant la configuration cloud-init : Utilisez la commande ``openstack server create`` pour créer une instance, en spécifiant les détails nécessaires tels que le nom, le flavors, l'image, le réseau, le groupe de sécurité, la clé etc. l'option ``--user-data`` pour spécifier le chemin vers votre fichier de configuration cloud-init pour spécifier la configuration à utiliser. Voici un exemple de commande :
- Attendez que l'instance soit en état "ACTIVE" avant de continuer.
- Ajouter une adresse ip floatante
- Se connecter a l' instance avec votre utilisateur et tester votre serveur apache
- Votre serveur Web doit etre accessible de l'extérieur de l' instance (secgroup)

## Le service de stockage block Cinder



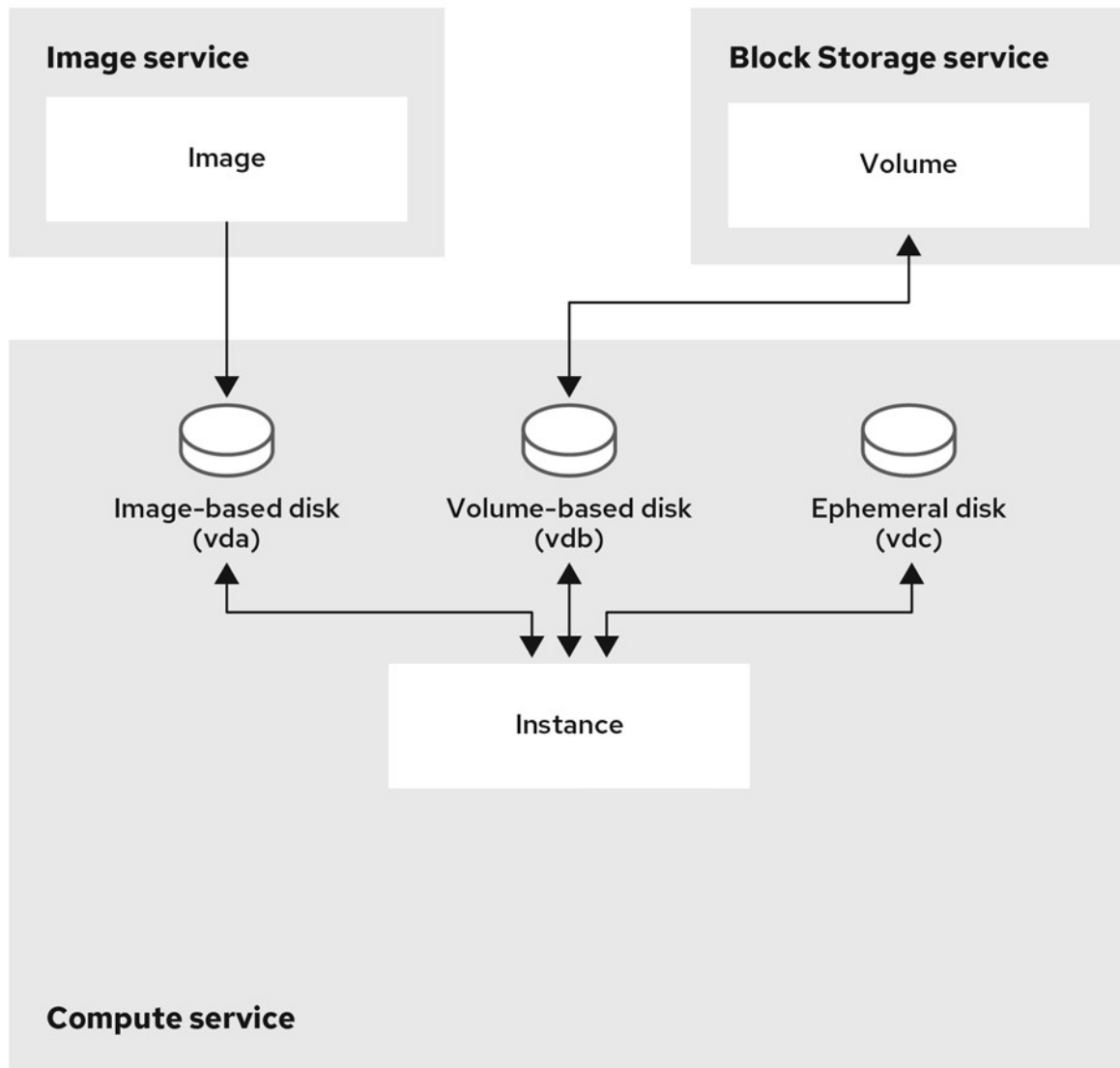
### Introduction

Cinder est le projet qui fournit des services de stockage block pour les machines virtuelles dans un environnement OpenStack.

Cinder permet aux utilisateurs de créer et de gérer des volumes de stockage persistants, qui peuvent être attachés ou détachés à des instances de machines virtuelles. Ces volumes peuvent être utilisés pour stocker des données de manière indépendante des instances de calcul, ce qui offre une grande flexibilité et une meilleure gestion des données.



## OpenStack



**Abstraction du stockage** : Cinder offre une abstraction du stockage sous forme de volumes, permettant aux utilisateurs de demander des ressources de stockage sans se soucier des détails sous-jacents du stockage physique.

**Prise en charge de multiples fournisseurs** : Cinder offre une architecture modulaire qui prend en charge plusieurs fournisseurs de stockage, ce qui permet aux utilisateurs de choisir le fournisseur de stockage qui convient le mieux à leurs besoins.

**Snapshots et clones** : Cinder prend en charge la création de snapshots (instantanés) des volumes existants, ce qui permet de capturer un état donné d'un volume à un instant précis. Il permet également de créer des clones de volumes à partir de ces snapshots, ce qui facilite la création de copies de volumes existants.

**Intégration avec d'autres services OpenStack** : Cinder est étroitement intégré avec d'autres composants d'OpenStack, tels que Nova (service de calcul) et Glance (service d'image). Cela permet d'associer facilement des volumes à des instances de machines virtuelles et de créer des images de volumes.

**Évolutivité et haute disponibilité** : Cinder est conçu pour être évolutif et hautement disponible. Il utilise des mécanismes de répartition de charge et de redondance pour assurer une disponibilité élevée des volumes de stockage.

OpenStack Cinder est un projet important au sein de la plateforme OpenStack, offrant des services de stockage block flexibles et évolutifs pour les machines virtuelles. Il facilite la gestion des ressources de stockage et permet aux utilisateurs de choisir parmi différents fournisseurs de stockage.

## Les types de volumes

Cinder prend en charge plusieurs types de volumes, qui peuvent être spécifiés lors de la création d'un volume. Voici quelques-uns des types de volumes couramment utilisés avec Cinder :

**Volume standard (Generic volume)** : C'est le type de volume par défaut pris en charge par Cinder. Il s'agit d'un volume générique qui utilise les fonctionnalités de stockage de base du système sous-jacent. Ce type de volume est approprié pour un large éventail de cas d'utilisation et est généralement utilisé lorsque des fonctionnalités spécifiques ne sont pas nécessaires.

**Volume basé sur le réseau (Network-based volume) :** Ce type de volume utilise le stockage en réseau (par exemple, iSCSI ou Fibre Channel) pour fournir une connectivité entre les nœuds de calcul et le stockage. Il permet une gestion plus centralisée et une meilleure séparation entre le stockage et les nœuds de calcul.

**Volume avec sauvegarde (Backup volume) :** Ce type de volume est utilisé pour créer une sauvegarde de données à partir d'un volume existant. Il permet de créer des copies de sauvegarde des données stockées dans un volume afin de garantir la disponibilité et la reprise après sinistre.

**Volume de démarrage (Bootable volume) :** Ce type de volume est utilisé comme volume de démarrage pour les instances de machines virtuelles. Il contient le système d'exploitation et les fichiers nécessaires pour démarrer l'instance. Les volumes de démarrage permettent une plus grande flexibilité dans la gestion des images et des instances.

**Volume SSD (SSD volume) :** Ce type de volume utilise des disques SSD (Solid-State Drive) pour offrir des performances de stockage élevées. Les volumes SSD sont adaptés aux charges de travail intensives en termes de lecture/écriture et aux applications nécessitant des temps d'accès rapides aux données.

Il est important de noter que la disponibilité des différents types de volumes dépend de votre environnement OpenStack, des pilotes de stockage utilisés et des fonctionnalités prises en charge par votre système de stockage sous-jacent. Vous devrez consulter la documentation spécifique à votre déploiement OpenStack pour connaître les types de volumes disponibles dans votre cas particulier.

## Les backend de stockage

Cinder, propose la possibilité d'utiliser différents types de stockage en fonction des besoins et de la configuration de votre environnement. Quelques-uns des types de stockage couramment utilisés avec Cinder :

**Stockage local** : Vous pouvez utiliser le stockage local des nœuds de calcul (compute nodes) dans votre environnement OpenStack. Cela implique d'utiliser le stockage directement attaché aux serveurs physiques qui exécutent les instances de machines virtuelles. Le stockage local offre des performances élevées, mais il n'est pas partageable entre différentes instances.

**Stockage en réseau** : Cinder prend en charge plusieurs protocoles de stockage en réseau, tels que iSCSI (Internet Small Computer System Interface) et Fibre Channel. Ces protocoles permettent de connecter les nœuds de calcul aux dispositifs de stockage dédiés via un réseau. Vous pouvez utiliser des systèmes de stockage externes, tels que des baies de stockage SAN (Storage Area Network) ou NAS (Network Attached Storage), pour fournir des capacités de stockage partagé entre les instances.

**Solutions de stockage tierces** : Cinder offre une architecture modulaire qui permet l'intégration de fournisseurs de stockage tiers. Cela signifie que vous pouvez utiliser des solutions de stockage tierces qui sont compatibles avec Cinder. Certains fournisseurs de stockage populaires, tels que NetApp, Dell EMC, IBM, et bien d'autres, proposent des pilotes Cinder pour intégrer leurs solutions de stockage avec OpenStack.

Il est important de noter que la disponibilité des différents types de stockage dépend de la configuration de votre environnement OpenStack, des pilotes de stockage pris en charge et des plugins installés. Vous devrez vérifier la

documentation spécifique à votre déploiement OpenStack pour connaître les options de stockage disponibles dans votre cas particulier.

## Les snapshots et clones

Les snapshots et les clones sont des fonctionnalités essentielles de Cinder pour la gestion des volumes de stockage.

### Snapshots (Instantanés) :

Les snapshots permettent de capturer un état donné d'un volume à un instant précis. Un snapshot est une copie des données du volume à un moment précis, mais il ne duplique pas réellement les données. Lorsque vous prenez un snapshot d'un volume, les modifications ultérieures apportées au volume d'origine ne sont pas reflétées dans le snapshot. Les snapshots sont généralement utilisés pour des besoins tels que la sauvegarde, la reprise après sinistre et la création de volumes dérivés.

fonctionnement des snapshot avec Cinder :

- Vous créez un snapshot à partir d'un volume existant en utilisant l'API ou le tableau de bord d'OpenStack.
- Le snapshot est créé en enregistrant les métadonnées et les informations nécessaires pour identifier l'état du volume.
- Une fois le snapshot créé, vous pouvez utiliser cette copie pour diverses opérations, telles que la restauration du volume à son état précédent, la création d'un nouveau volume basé sur le snapshot, ou la récupération des données spécifiques à partir du snapshot.

### Clones :

Les clones permettent de créer une copie complète d'un volume existant à partir d'un snapshot. Contrairement au snapshot qui conserve une référence aux données d'origine, un clone est une réplique indépendante du volume

d'origine. Les clones sont utiles lorsque vous avez besoin de plusieurs instances de données identiques pour les tests, le développement ou d'autres cas d'utilisation.

fonctionnement du clonage avec Cinder :

- Vous créez un clone à partir d'un snapshot existant en utilisant l'API ou le tableau de bord d'OpenStack.
- Le clone est créé en copiant entièrement les données du snapshot dans un nouveau volume.
- Le nouveau volume clone est complètement indépendant du snapshot et du volume d'origine. Vous pouvez l'attacher à une instance de machine virtuelle et y accéder de manière autonome.

Il est important de noter que les snapshots et les clones consomment de l'espace de stockage supplémentaire. Les snapshots ne dupliquent pas réellement les données, mais ils conservent les données existantes jusqu'à la création du snapshot. Les clones, en revanche, créent une copie complète des données, ce qui peut augmenter les besoins en espace de stockage.

En résumé, les snapshots permettent de capturer un état donné d'un volume à un instant précis, tandis que les clones permettent de créer une copie complète et indépendante d'un volume à partir d'un snapshot. Ces fonctionnalités offrent une grande flexibilité pour la gestion des volumes de stockage dans OpenStack Cinder.

## Tp -Gestion des volumes

Création de volume type lvm et nfs

```
ludo@openstack:~$ openstack volume type create lvm
ludo@openstack:~$ openstack volume type list
```

```
ludo@openstack:~$ openstack volume type create nfs
ludo@openstack:~$ openstack volume type list
```

Création d' un volume comme utilisateurs mebre du projet formation

```
ludo@openstack:~$ openstack volume create --type lvm --size 10 disk-lvm
ludo@openstack:~$ openstack volume create --type nfs --size 10 disk-nfs
ludo@openstack:~$ openstack volume list
```

Attacher les volumes a l'instance

```
ludo@openstack:~$ openstack server add volume Ubuntu-2204 disk-lvm
ludo@openstack:~$ openstack server add volume Ubuntu-2204 disk-nfs
ludo@openstack:~$ openstack server add volume Ubuntu-2204 disk-nfs
ludo@openstack:~$ openstack volume list
```

Utilisation du volume dans les instances

```
ludo@openstack:~$ ssh ludo@10.0.0.174
ludo@openstack:~$ lsblk
ludo@openstack:~$ vcgreate cinder /dev/vdb
ludo@openstack:~$ lvcreate -l +5G -n home cinder
ludo@openstack:~$ mkfs.xfs /dev/cinder/home
ludo@openstack:~$ mount /dev/cinder/home /home/
```

## **TP – Réaliser un cloud-init pour configurer automatiquement les volumes Cinder**

Utilisation du volume dans les instances

Ajouter dans votre Cloud-init créé précédemment pour votre instance :

la gestion du volume



## Heat



### Introduction à Heat

Heat est un composant essentiel de la plateforme de cloud computing OpenStack. Il fournit des fonctionnalités d'orchestration et d'automatisation pour déployer et gérer des applications et des services sur l'infrastructure OpenStack.

Heat permet aux utilisateurs de décrire leur infrastructure en utilisant un langage de modèle déclaratif appelé Heat Orchestration Template (HOT). Ce modèle décrit les ressources nécessaires pour déployer une application, telles que les instances de machines virtuelles, les réseaux, les volumes de stockage, etc. Le modèle HOT est généralement écrit en format YAML (YAML Ain't Markup Language).

Une fois le modèle défini, Heat déploie et configure automatiquement les ressources spécifiées dans le modèle sur l'infrastructure OpenStack. Il prend en charge la gestion des dépendances entre les ressources, ce qui permet de définir des relations et des flux de travail complexes.

Heat simplifie le processus de déploiement et de gestion des applications sur OpenStack en automatisant les tâches répétitives et en permettant aux utilisateurs de décrire leur infrastructure de manière déclarative. Cela favorise la flexibilité, la reproductibilité et l'efficacité opérationnelle dans le déploiement de l'infrastructure cloud.

Le modèle HOT (Heat Orchestration Template) est un langage de modèle déclaratif utilisé dans OpenStack Heat pour décrire l'infrastructure et les ressources nécessaires au déploiement d'une application. Le modèle HOT est écrit en utilisant le format YAML (YAML Ain't Markup Language), qui est un format simple et lisible par les humains.

### Le modèle HOT

Le modèle HOT permet aux utilisateurs de spécifier les ressources, leurs propriétés et leurs relations de dépendance. Voici quelques concepts clés du modèle HOT :

**Ressources** : Le modèle HOT permet de définir différents types de ressources, tels que des instances de machines virtuelles, des réseaux, des volumes de stockage, des groupes de sécurité, etc. Chaque ressource est décrite avec ses propriétés spécifiques, telles que la taille, l'image, les règles de sécurité, etc.

**Paramètres** : Les paramètres permettent aux utilisateurs de fournir des valeurs personnalisées lors du déploiement de l'infrastructure. Ils peuvent être utilisés pour configurer des valeurs telles que les noms d'utilisateur, les mots de passe, les adresses IP, etc.

**Relations de dépendance** : HOT permet de définir les relations de dépendance entre les ressources. Par exemple, une instance de machine virtuelle peut dépendre d'un réseau existant ou d'un volume de stockage. Cela permet à Heat

de gérer l'ordre de déploiement des ressources et de s'assurer que les dépendances sont satisfaites.

**Groupes et itérations** : Le modèle HOT permet de regrouper des ressources similaires et de les traiter de manière itérative. Cela permet de déployer plusieurs instances de machines virtuelles, par exemple, en utilisant les mêmes propriétés et les mêmes dépendances.

En utilisant le modèle HOT, les utilisateurs peuvent décrire leur infrastructure de manière déclarative, ce qui signifie qu'ils spécifient ce qu'ils veulent plutôt que comment le faire. Heat se charge ensuite d'interpréter le modèle et de gérer les opérations de déploiement, de mise à jour et de suppression des ressources sur l'infrastructure OpenStack. Cela simplifie considérablement la gestion et l'automatisation des déploiements d'applications sur OpenStack.

## TP - Déploiement avec Heat

Déploiement d'une instance avec Heat

Simple stack :

```
ludo@openstack:~$ vi simple-stack.yml
heat_template_version: 2013-05-23
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: my_key_pair_1
      image: cirros-0.3.1-x86_64
      flavor: m1.small

ludo@openstack:~$ openstack stack create -t simple-stack.yml simple-stack

ludo@openstack:~$ openstack stack list

ludo@openstack:~$ openstack server list
ludo@openstack:~$ openstack stack delete --yes Sample-Stack

ludo@openstack:~$ openstack stack list
```

Déploiement d'une stack avec paramètres :

```
ludo@openstack:~$ vi param-stack.yml

heat_template_version: 2021-04-16
description: Heat Sample Template
parameters:
  ImageID:
    type: string
    description: Image used to boot a server
  NetID:
    type: string
    description: Network ID for the server
resources:
  server1:
    type: OS::Nova::Server
```

```
properties:
  name: "Heat_Deployed_Server"
  image: { get_param: ImageID }
  flavor: "m1.small"
  networks:
    - network: { get_param: NetID }
outputs:
  server1_private_ip:
    description: IP address of the server in the private network
    value: { get_attr: [ server1, first_address ] }
```

ludo@openstack:~\$ openstack image list

ludo@openstack:~\$ openstack network list

ludo@openstack:~\$ Int\_Net\_ID=\$(openstack network list | grep private \
| awk '{ print \$2 }')

ludo@openstack:~\$ openstack stack create -t sample-stack.yml \
--parameter "ImageID=Ubuntu2204;NetID=\$Int\_Net\_ID" param-Stack

ludo@openstack:~\$ openstack stack list

ludo@openstack:~\$ openstack server list

ludo@openstack:~\$ openstack stack delete --yes Sample-Stack

ludo@openstack:~\$ openstack stack list

### Ajout de notre utilisateur dans Heat

```
ludo@openstack:~$ openstack role list
```

ludo@openstack:~\$ openstack role add --project formation --user ludo \
heat\_stack\_owner

ludo@openstack:~\$ openstack stack list



## Qu'est-ce qu'une application Cloud-Ready/Native ?

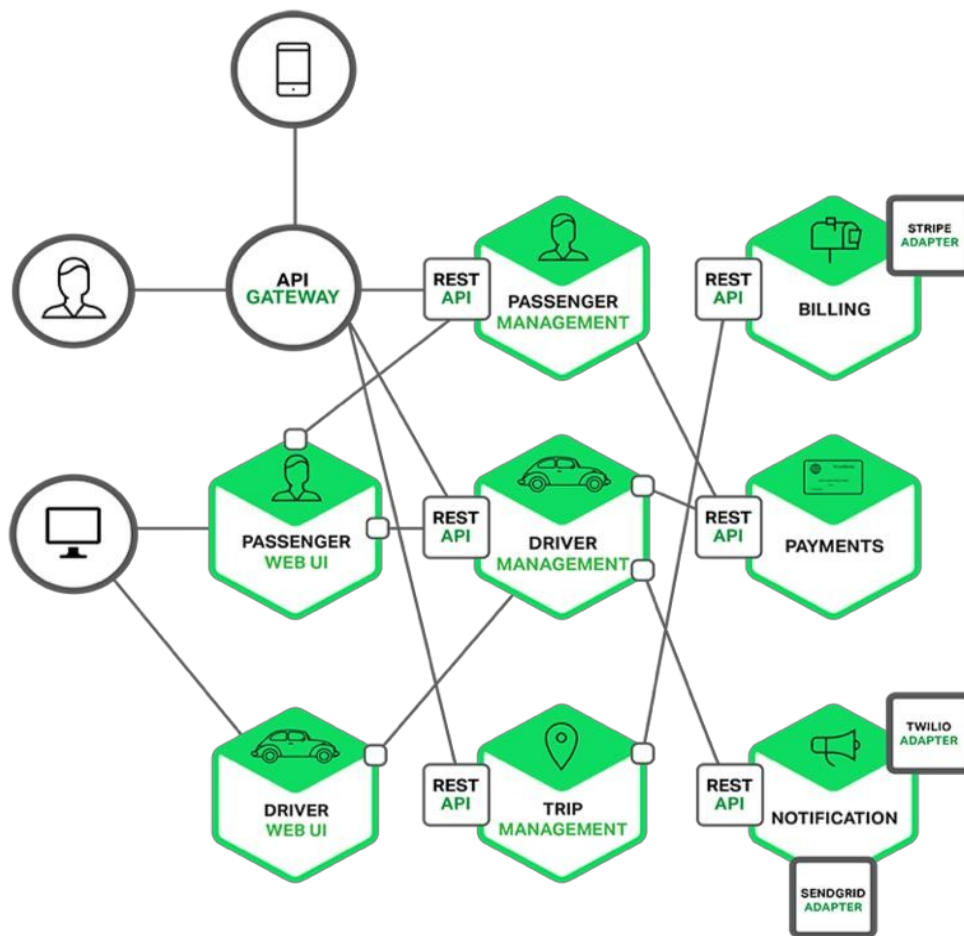
Une application Cloud-Native (ou Cloud-Ready) est une application qui est conçue spécifiquement pour fonctionner sur une infrastructure de cloud computing. Elle est développée pour être distribuée sur des environnements de cloud, tels que OpenStack, AWS, Azure, Google Cloud, etc. Elle est conçue pour exploiter les avantages offerts par le cloud computing, tels que l'évolutivité, la résilience et la haute disponibilité.

Une application Cloud-Ready est similaire à une application Cloud-Native, mais elle est généralement une application existante qui a été modifiée pour fonctionner sur un environnement de cloud computing. Elle a été adaptée pour être déployée sur un environnement de cloud, sans nécessiter de modifications majeures.

Différences entre une application traditionnelle et une application Cloud-Native ou Cloud-Ready

Les applications traditionnelles sont généralement conçues pour être installées et exécutées sur des serveurs physiques locaux. Elles sont souvent monolithiques et ne peuvent pas être facilement scindées en microservices. Ces applications peuvent ne pas être conçues pour fonctionner sur un environnement de cloud computing, et peuvent donc nécessiter des modifications importantes pour être déployées sur le cloud.

Les applications Cloud-Natives sont conçues pour fonctionner sur un environnement de cloud computing. Elles sont généralement développées en utilisant des architectures de microservices, ce qui les rend plus agiles et plus faciles à maintenir. Elles sont également conçues pour exploiter les avantages offerts par le cloud computing, tels que l'évolutivité, la résilience et la haute disponibilité.



### Avantages d'une application Cloud-Native ou Cloud-Ready

Les avantages d'une application Cloud-Native ou Cloud-Ready comprennent :

- **Évolutivité** : Les applications Cloud-Natives sont conçues pour être hautement évolutives, ce qui signifie qu'elles peuvent facilement s'adapter à des charges de travail fluctuantes, sans compromettre les performances.
- **Haute disponibilité** : Les applications Cloud-Natives sont conçues pour être hautement disponibles, ce qui signifie qu'elles peuvent fonctionner en continu sans interruption, même en cas de panne d'un serveur ou d'un nœud.



- **Résilience** : Les applications Cloud-Natives sont conçues pour être résilientes, ce qui signifie qu'elles peuvent résister aux erreurs et continuer à fonctionner de manière fiable.
- **Développement agile** : Les applications Cloud-Natives sont souvent développées en utilisant des architectures de microservices, ce qui les rend plus agiles et plus faciles à maintenir.
- **Économique** : Les applications Cloud-Natives peuvent être plus économiques, car elles peuvent être facilement déployées sur des environnements de cloud computing, sans nécessiter des infrastructures coûteuses.

Les applications Cloud-Natives ou Cloud-Ready sont conçues pour fonctionner sur des environnements de cloud computing. Elles sont hautement évolutives, disponibles, résilientes et agiles, et peuvent être plus économiques que les applications traditionnelles.

Une application Cloud-Native est conçue pour être déployée et exécutée sur une infrastructure de cloud computing. Elle est développée à l'aide d'une architecture de microservices, qui consiste en une suite de services autonomes et distribués. Chaque service est conçu pour accomplir une tâche spécifique et communique avec les autres services à travers une interface bien définie.

Caractéristiques techniques d'une application Cloud-Native :

- **Conteneurisation** : les services d'une application Cloud-Native sont souvent déployés dans des conteneurs, ce qui permet une mise en production rapide et une portabilité entre différents environnements de cloud computing.
- **Évolutivité horizontale** : les services d'une application Cloud-Native peuvent facilement s'adapter à une charge de travail fluctuante grâce à une évolutivité horizontale. Cela signifie que les services peuvent être

dupliqués pour répondre à une demande accrue, puis supprimés lorsque la demande diminue.

- **Gestion de la configuration** : les applications Cloud-Natives sont souvent configurées à l'aide d'outils de gestion de la configuration tels que Chef, Puppet ou Ansible. Ces outils permettent une gestion centralisée de la configuration et une mise en production rapide et reproductible.
- **Équilibrage de charge** : les services d'une application Cloud-Native peuvent être équilibrés de manière dynamique pour optimiser les performances et la disponibilité de l'application.
- **Sécurité** : les applications Cloud-Natives sont souvent conçues avec la sécurité à l'esprit dès le départ. Les services sont isolés les uns des autres et sont souvent protégés par des pare-feu et d'autres mécanismes de sécurité.
- **Monitoring et journalisation** : les applications Cloud-Natives sont souvent équipées de mécanismes de monitoring et de journalisation pour surveiller les performances et la disponibilité des services, ainsi que pour faciliter le débogage en cas de problème.

Une application Cloud-Native est conçue pour être résiliente, évolutive, portable et facilement gérable dans un environnement de cloud computing.

## Trove



### Introduction

Trove est le projet OpenStack qui offre un service de base de données en tant que service (Database-as-a-Service ou DBaaS). Il fournit une plateforme de déploiement et de gestion simplifiée des bases de données relationnelles et NoSQL dans un environnement OpenStack. Trove permet aux utilisateurs de provisionner, de gérer et d'accéder facilement à des bases de données, sans avoir à se soucier de la configuration et de la gestion des infrastructures sous-jacentes.

**Provisionnement de bases de données :** Trove permet de provisionner rapidement et facilement des bases de données, telles que MySQL, PostgreSQL, MongoDB, Cassandra, etc. Il fournit une API et une interface utilisateur qui permettent aux utilisateurs de spécifier les détails de la base de données souhaitée, tels que le type, la version, la taille, etc., et Trove se charge de la création et de la configuration de l'infrastructure nécessaire.

**Gestion des bases de données :** Trove offre des fonctionnalités de gestion des bases de données, telles que la sauvegarde, la restauration, la mise à l'échelle et la surveillance des performances. Il permet également d'automatiser

certaines tâches courantes, telles que la réplication des données, la gestion des utilisateurs et des autorisations, etc.

**Isolation et sécurité** : Trove assure l'isolation des bases de données entre les utilisateurs en utilisant des mécanismes de virtualisation et d'isolation des ressources. Il garantit également la sécurité des données en fournissant des fonctionnalités telles que le chiffrement des données, l'authentification et l'autorisation basées sur les rôles, etc.

**Intégration avec d'autres services OpenStack** : Trove s'intègre étroitement avec d'autres services OpenStack, tels que Nova pour le provisionnement des instances, Neutron pour la gestion du réseau, Cinder pour le stockage des données, et Keystone pour l'authentification et l'autorisation. Cela permet d'utiliser Trove de manière transparente dans un environnement OpenStack existant.

**Évolutivité et haute disponibilité** : Trove prend en charge l'évolutivité horizontale et verticale des bases de données. Il permet d'ajuster automatiquement les ressources allouées à une base de données en fonction des besoins de charge de travail. De plus, Trove offre des mécanismes de haute disponibilité pour garantir la continuité des services en cas de panne ou de défaillance matérielle.

Trove simplifie le déploiement, la gestion et l'accès aux bases de données dans un environnement OpenStack. Il permet aux développeurs et aux utilisateurs de se concentrer sur leurs applications plutôt que sur la gestion de l'infrastructure sous-jacente. En offrant un service de base de données en tant que service, Trove contribue à accélérer le développement d'applications, à réduire les coûts d'exploitation et à améliorer l'agilité dans les environnements de cloud computing.

**Trove** permet de créer des clusters de bases de données en utilisant des fonctionnalités spécifiques du moteur de base de données sous-jacent, telles

que la réplication, la partitionnement ou le clustering propre au moteur. Le processus de création d'un cluster de base de données avec Trove varie en fonction du moteur de base de données utilisé. Voici un aperçu général du fonctionnement :

**Choix du moteur de base de données** : Avant de créer un cluster, vous devez choisir un moteur de base de données compatible avec le clustering, tels que MySQL Galera Cluster, Percona XtraDB Cluster, PostgreSQL avec la réplication, etc. Chaque moteur de base de données peut avoir ses propres exigences et configurations spécifiques pour la création du cluster.

**Configuration du cluster** : Une fois le moteur de base de données choisi, vous devez configurer les paramètres spécifiques du cluster. Cela peut inclure des détails tels que le nombre de nœuds dans le cluster, les adresses IP, les rôles de réplication, la configuration de la répartition de charge, etc. Ces configurations dépendent du moteur de base de données et de ses fonctionnalités de clustering.

**Création des instances** : La prochaine étape consiste à créer les instances de base de données qui constitueront le cluster. Vous spécifiez le nombre d'instances souhaitées et les configurations nécessaires, telles que le flavor, la taille du disque, etc. Trove se charge de provisionner les instances et de les préparer pour la formation du cluster.

**Formation du cluster** : Une fois que les instances sont créées, Trove initie le processus de formation du cluster. Cela peut impliquer la configuration des connexions réseau entre les nœuds, l'initialisation de la réplication, l'élection du nœud principal, etc. Ces étapes sont gérées par le moteur de base de données spécifique au cluster.

**Gestion du cluster** : Une fois que le cluster est formé, Trove fournit des fonctionnalités de gestion pour le cluster de bases de données. Cela peut inclure la surveillance de l'état du cluster, la gestion des performances, les opérations de mise à l'échelle, la gestion des sauvegardes et des restaurations,

etc. Ces opérations sont effectuées à l'aide des API Trove ou de l'interface utilisateur.

Il est important de noter que la création et la gestion des clusters de bases de données dépendent des fonctionnalités spécifiques du moteur de base de données utilisé. Trove fournit une couche d'abstraction pour faciliter le déploiement et la gestion des clusters, mais les détails techniques et les fonctionnalités spécifiques sont gérés par le moteur de base de données lui-même. Il est recommandé de consulter la documentation spécifique au moteur de base de données pour obtenir des informations détaillées sur la création et la gestion des clusters avec Trove.

## TP – Database As Service

### Ajout de l'image pour Trove

```
ludo@openstack:~$ wget
https://tarballs.opendev.org/openstack/trove/images/trove-master-guest-
ubuntu-focal.qcow2

ludo@openstack:~$ openstack image create Trove-Ubuntu --file=trove-
master-guest-ubuntu-focal.qcow2 --disk-format=qcow2 --container-
format=bare --tag=trove --private

ludo@openstack:~$ openstack image list
```

### Création du volume Cinder pour Trove

```
ludo@openstack:~$ openstack volume type create lvm-trove --private
```

### Ajout des modeles de bases de données

```
ludo@openstack:~$ openstack datastore version create 10.5 mariadb \
mariadb "" --image-tags trove --active --default \
--version-number 10.5

ludo@openstack:~$ openstack datastore version create 10.3 mariadb \
mariadb "" --image-tags trove --active --default \
--version-number 10.3
```

### Lister les datastores disponibles

```
ludo@openstack:~$ openstack datastore list

ludo@openstack:~$ openstack datastore version list mariadb
```

## Création d'une nstances de Base de données

```
ludo@openstack:~$ openstack network list | grep formation

ludo@openstack:~$ openstack database instance create MariaDB-103 \
--flavor 1 \
--size 5 \
--nic net-id=ccb06f2c-e5ef-405a-8027-8961672590af \
--databases MyDB --users ludo:password \
--datastore mariadb --datastore-version 10.3
--allowed-cidr 192.168.100.0/24

ludo@openstack:~$ openstack database instance list
```

## On connexe la base de données de notre instance

```
ludo@openstack:~$ ssh ubuntu@10.0.0.167

ubuntu@ubuntu:~$ sudo apt update  && sudo apt install mariadb-client

ubuntu@ubuntu:~$ mysql -u ludo -h IP_MYSQL
```



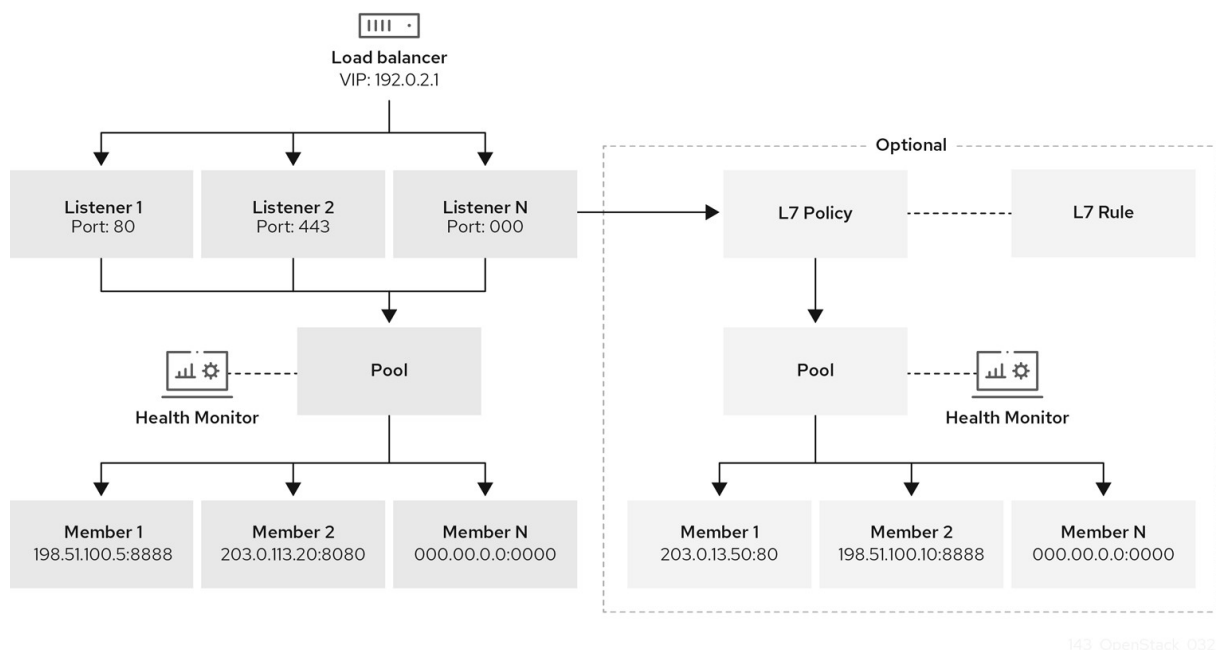
## Octavia

### Introduction

Octavia est le projet d'équilibreur de charge. Il fournit un service de répartition de charge (load balancing) hautement évolutif et performant pour les applications déployées dans un environnement OpenStack.

Octavia a été conçu pour remplacer le service de répartition de charge d'OpenStack appelé Neutron LBaaS (Load Balancer as a Service). Il offre une solution plus robuste, évolutive et flexible pour gérer la distribution de charge sur les applications web et les services dans un environnement cloud.

Octavia utilise maintenant haproxy pour fournir de l'équilibrage de charge



**Architecture :** Octavia utilise une architecture distribuée pour fournir une haute disponibilité et une évolutivité horizontale. Il utilise un modèle d'agent distribué où chaque nœud de calcul (compute node) exécute un agent Octavia

qui est responsable de la gestion des équilibreurs de charge (load balancers) et des pools.

**quilibrage de charge** : Octavia prend en charge plusieurs algorithmes d'équilibrage de charge, tels que le round-robin, le poids, la session persistante et le ratio. Ces algorithmes permettent de distribuer intelligemment les requêtes entre les instances de l'application pour améliorer la performance et la résilience.

**Sécurité** : Octavia offre des fonctionnalités de sécurité avancées. Il peut effectuer le déchargement SSL/TLS (SSL/TLS offloading) pour décharger le traitement des certificats SSL/TLS des instances de l'application. De plus, il prend en charge l'utilisation de certificats SSL/TLS pour sécuriser les communications entre les clients et les équilibreurs de charge.

**Intégration avec OpenStack** : Octavia s'intègre étroitement avec d'autres services OpenStack. Il utilise Keystone pour l'authentification et l'autorisation des utilisateurs. Il interagit avec Neutron pour la gestion des réseaux et des sous-réseaux. De plus, il utilise Nova pour la création et la gestion des instances virtuelles nécessaires pour les équilibreurs de charge.

**API et interface utilisateur** : Octavia fournit une API RESTful qui permet aux utilisateurs de gérer les équilibreurs de charge et les pools. Il est également possible d'utiliser une interface utilisateur graphique (Horizon) pour gérer Octavia et configurer les équilibreurs de charge.

Octavia est le projet OpenStack qui fournit un service de répartition de charge hautement évolutif et performant pour les applications déployées dans un environnement OpenStack. Il offre des fonctionnalités avancées, une

intégration étroite avec les autres services OpenStack et une interface conviviale pour la gestion des équilibres de charge.

## **Configuration d'Octavia**

La configuration d'Octavia implique plusieurs étapes pour déployer et configurer correctement les équilibres de charge dans un environnement OpenStack. Voici un aperçu des principales étapes de configuration :

**Installation et configuration d'Octavia** : La première étape consiste à installer et configurer les composants d'Octavia. Cela comprend l'installation des packages Octavia sur les contrôleurs et les nœuds de calcul, ainsi que la configuration des fichiers de configuration d'Octavia.

**Configuration du réseau** : Avant de déployer les équilibres de charge, vous devez configurer le réseau pour Octavia. Cela implique la création de sous-réseaux, de réseaux flottants (floating networks) et d'autres ressources réseau nécessaires pour les équilibres de charge.

**Création du fournisseur de services d'équilibrage de charge** : Dans OpenStack, Octavia utilise des fournisseurs de services d'équilibrage de charge pour interagir avec les services d'équilibrage de charge sous-jacents (tels que HAProxy, NGINX, etc.). Vous devez créer un fournisseur de services d'équilibrage de charge pour spécifier le type de service d'équilibrage de charge que vous souhaitez utiliser.

**Configuration des certificats SSL/TLS** : Si vous souhaitez utiliser des certificats SSL/TLS pour sécuriser les communications avec les équilibres de charge, vous devez générer et configurer les certificats appropriés. Cela peut inclure la création de certificats autosignés ou l'acquisition de certificats auprès d'une autorité de certification (CA).

**Création des pools et des membres :** Les pools sont des groupes d'instances d'application auxquelles les requêtes sont distribuées. Vous devez créer des pools et spécifier les membres qui y sont associés. Chaque membre représente une instance de l'application et est associé à une adresse IP et un port.

**Création des équilibreurs de charge :** Une fois les pools et les membres configurés, vous pouvez créer les équilibreurs de charge. Un équilibreur de charge est associé à un pool et reçoit les requêtes clients, puis les distribue aux membres du pool.

**Gestion des règles et des listes de contrôle d'accès (ACL) :** Octavia offre la possibilité de configurer des règles et des ACL pour contrôler le flux des requêtes. Vous pouvez définir des règles pour rediriger les requêtes, modifier les en-têtes HTTP, filtrer les adresses IP, etc.

**Surveillance et gestion des équilibreurs de charge :** Octavia fournit des outils de surveillance et de gestion des équilibreurs de charge. Vous pouvez surveiller les performances des équilibreurs de charge, visualiser les statistiques et effectuer des opérations de gestion telles que la mise à l'échelle, la mise à jour du firmware, etc.

## TP- Déploiement d'un load balancer avec Octavia

Création du load balancing

```
ludo@openstack:~$ openstack subnet list

ludo@openstack:~$ openstack loadbalancer create --name lb01 \
--vip-subnet-id private-subnet
```

```
ludo@openstack:~$ openstack loadbalancer list
```

### Création des Listener et ajout du pool

```
ludo@openstack:~$ openstack loadbalancer listener create -name \
listener01 --protocol TCP --protocol-port 80 lb01

ludo@openstack:~$ openstack loadbalancer pool create --name pool01 \
--lb-algorithm ROUND_ROBIN --listener listener01 --protocol TCP
```

### Ajout des membres du lb

```
ludo@openstack:~$ openstack server list
Web02      | ACTIVE | private=192.168.100.215
Web01      | ACTIVE | private=192.168.100.222

ludo@openstack:~$ openstack loadbalancer member create --subnet-id
private-subnet --address 192.168.100.215 --protocol-port 80 pool01

ludo@openstack:~$ openstack loadbalancer member create --subnet-id
private-subnet --address 192.168.100.222 --protocol-port 80 pool01

ludo@openstack:~$ openstack loadbalancer member list pool01

ludo@openstack:~$ openstack floating ip create public

ludo@openstack:~$ VIPPORT=$(openstack loadbalancer show lb01 | grep
vip_port_id | awk {'print $4'})

ludo@openstack:~$ openstack floating ip set --port $VIPPORT 10.0.0.245

ludo@openstack:~$ curl 10.0.0.245
Web Server on Instance01
```

```
ludo@openstack:~$ curl 10.0.0.245
Web Server on Instance02

ludo@openstack:~$ curl 10.0.0.245
Web Server on Instance01
```

## Magnum

### Introduction

Magnum est un projet d'orchestration de conteneurs dans l'écosystème OpenStack. Il fournit des fonctionnalités permettant de provisionner, gérer et orchestrer des clusters de conteneurs, en utilisant des technologies de conteneurisation populaires telles que Docker, Kubernetes et Mesos.

**Provisionnement de clusters :** Magnum permet de provisionner facilement des clusters de conteneurs à l'aide d'une interface unifiée. Il prend en charge plusieurs moteurs d'orchestration de conteneurs, notamment Kubernetes, Docker Swarm et Apache Mesos. Les utilisateurs peuvent choisir le moteur d'orchestration de leur choix lors de la création d'un cluster.

**Gestion des clusters :** Magnum offre des fonctionnalités de gestion avancées pour les clusters de conteneurs. Il fournit des opérations telles que la mise à l'échelle automatique des nœuds, la mise à jour des versions du moteur d'orchestration, la gestion des politiques de redémarrage, la gestion des certificats de sécurité, etc.

**Intégration avec OpenStack :** Magnum est étroitement intégré à l'écosystème OpenStack. Il utilise les services existants d'OpenStack, tels que Keystone pour l'authentification et l'autorisation, Nova pour la gestion des instances virtuelles, Neutron pour la gestion des réseaux, etc. Cette intégration permet une expérience cohérente et une utilisation harmonieuse des ressources d'OpenStack avec les clusters de conteneurs.

**Extensibilité :** Magnum est conçu de manière à être extensible et à permettre l'intégration de nouveaux moteurs d'orchestration de conteneurs. Cela permet

aux utilisateurs de choisir le moteur qui correspond le mieux à leurs besoins et de tirer parti des fonctionnalités spécifiques de chaque moteur.

**Sécurité** : Magnum prend en charge la sécurité des clusters de conteneurs en intégrant des fonctionnalités de sécurité d'OpenStack, telles que la gestion des certificats et des clés, l'isolation des réseaux et des ressources, l'authentification basée sur des tokens, etc. Cela permet de garantir un environnement sécurisé pour les applications conteneurisées.

les utilisateurs peuvent facilement créer et gérer des clusters de conteneurs dans leur environnement OpenStack, ce qui facilite le déploiement d'applications basées sur des conteneurs et leur gestion à grande échelle.



## Le reseau avec magnum

Magnum fournit des fonctionnalités pour gérer les orchestrateurs de conteneurs créés à l'aide Template Heat.

Magnum facilite le déploiement, la mise à jour d'orchestrateur de conteneurs, parmi Kubernetes, Swarm, Mesos.

Magnum va fournir:

**Réseau pour les nœuds du cluster :** Lors de la création d'un cluster avec Magnum, vous pouvez spécifier le réseau sur lequel les nœuds du cluster seront déployés. Cela peut être un réseau existant dans OpenStack ou vous pouvez demander à Magnum de créer un nouveau réseau pour le cluster. Magnum s'occupera de la création des sous-réseaux, des adresses IP et des autres ressources réseau nécessaires pour les nœuds.

**Connectivité entre les nœuds :** Magnum configure automatiquement les règles de sécurité et les groupes de sécurité d'OpenStack pour permettre la connectivité entre les nœuds du cluster. Cela garantit que les nœuds peuvent communiquer entre eux et échanger des données.

**Connectivité externe :** Les conteneurs et les applications s'exécutant dans les clusters de Magnum peuvent nécessiter une connectivité externe avec d'autres réseaux, services ou ressources. Magnum prend en charge l'attribution d'adresses IP flottantes (floating IPs) aux nœuds du cluster, ce qui permet d'accéder aux conteneurs depuis l'extérieur du cluster. Vous pouvez également configurer des règles de pare-feu et des balancers de charge pour gérer le trafic entrant et sortant.

**Plugins réseau :** Magnum prend en charge les plugins réseau d'OpenStack, tels que le plugin Neutron ML2 (Modular Layer 2), qui permet d'utiliser différents pilotes de réseau sous-jacents, tels que Open vSwitch, Linux Bridge, etc. Cela permet de personnaliser et d'adapter le réseau en fonction des besoins spécifiques du cluster.

**Isolation du réseau :** Magnum utilise les fonctionnalités d'isolation de réseau de Neutron pour garantir que les conteneurs et les applications s'exécutant dans les clusters sont isolés les uns des autres et des autres réseaux. Cela permet d'assurer la sécurité et la performance du réseau.

## déploiement d' un cluster avec Magnum

Vérification du service Magnum et création des volume pour le cluster

```
ludo@openstack:~$ openstack coe list

ludo@openstack:~$ openstack volume type create lvm-magnum --public
```

Créer une flavors m1.large pour Magnum:

8 Go de ram

20 Go de disque

4 CPU

Ajout de l' image des noeuds

```
ludo@openstack:~$ wget
https://builds.coreos.fedoraproject.org/prod/streams/stable/builds/
35.20220313.3.1/x86_64/fedora-coreos-35.20220313.3.1-
openstack.x86_64.qcow2.xz

ludo@openstack:~$ xz -dv \
fedora-coreos-35.20220313.3.1-openstack.x86_64.qcow2.xz

ludo@openstack:~$ penstack image create Fedora-CoreOS \
--file=fedora-coreos-35.20220313.3.1-openstack.x86_64.qcow2 \
--disk-format=qcow2 --container-format=bare \
--property os_distro='fedora-coreos' --public
```

Création du modele de cluster

```
ludo@openstack:~$ openstack flavor list

ludo@openstack:~$ openstack keypair list
```

```
ludo@openstack:~$ openstack network list
ludo@openstack:~$ openstack subnet list
```

```
ludo@openstack:~$ openstack coe cluster template create k8s-cluster-
template \
--image Fedora-CoreOS \
--external-network public \
--fixed-network private \
--fixed-subnet private-subnet \
--dns-nameserver 10.0.0.230 \
--network-driver calico
--docker-storage-driver overlay2 \
--docker-volume-size 10 \
--master-flavor m1.large \
--flavor m1.large \
--coe kubernetes
```

### Création du cluster avec deux noeuds

```
ludo@openstack:~$ openstack coe cluster create k8s-cluster \
--cluster-template k8s-cluster-template \
--master-count 1 \
--node-count 1 \
--keypair my-key

ludo@openstack:~$ openstack coe cluster list

ludo@openstack:~$ openstack stack list

ludo@openstack:~$ openstack stack list --nested | grep k8s-cluster

ludo@openstack:~$ openstack coe cluster list

ludo@openstack:~$ openstack server list
```

### Installation de la cli Kubernetes

```
ludo@openstack:~$ sudo snap install kubectl --classic
```

```
ludo@openstack:~$ openstack coe cluster config k8s-cluster  
export KUBECONFIG=/root/config  
ludo@openstack:~$ export KUBECONFIG=/root/config
```

## TP – Gestion des application avec Magnum

Afficher les noeuds du cluster

```
ludo@openstack:~$ kubectl get nodes
```

Afficher les pods de kubernetes

```
ludo@openstack:~$ kubectl get pods -n kube-system
```

Création d' un simple déploiement

```
ludo@openstack:~$ kubectl create deployment test-nginx --image=nginx \
--replicas=2
```

Afficher les ressources créées

```
ludo@openstack:~$ kubectl get pods
```

Supprimer le déploiement

```
ludo@openstack:~$ kubectl delete deployment test-nginx
```

## TP – Deploiement d' application

Création de son espace de nom

```
ludo@openstack:~$ vim namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: formation #ON CHANGE L'ESPACE DE NOM
```

On applique le manifest

```
ludo@openstack:~$ kubectl apply -f namespace.yaml
```

On s' enferme dans son espace de nom

```
ludo@openstack:~$ kubectl config set-context --current --namespace
MON_ESPACE_DE_NOM
```

Déploiement d'un serveur Web nginx

```
ludo@openstack:~$ cd k8s/pod
```

Editer le pod nginx et modifier le namespace

```
ludo@openstack:~$ vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: MON_ESPACE_DE_NOMS
  labels:
    app: web
    version: '1.9'
spec:
  containers:
  - image: nginx
    name: nginx
```

Déploiement du pod

```
ludo@openstack:~$ kubectl apply -f nginx-pod.yaml
```

on affiche le pod

```
ludo@openstack:~$ kubectl get pod
```

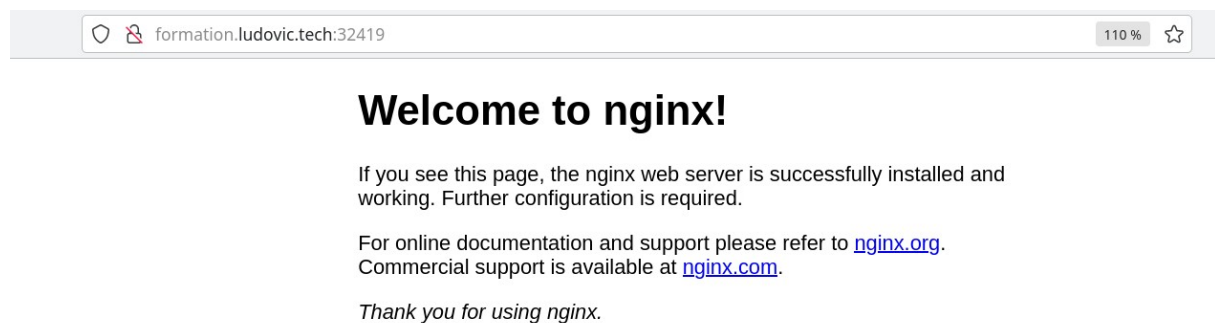
On ajoute du réseau a notre pod et on se connecte

```
ludo@openstack:~$ kubectl expose pod nginx --type NodePort --port 80

ludo@openstack:~$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx	NodePort	10.254.156.219	<none>	80:32419/TCP	11s

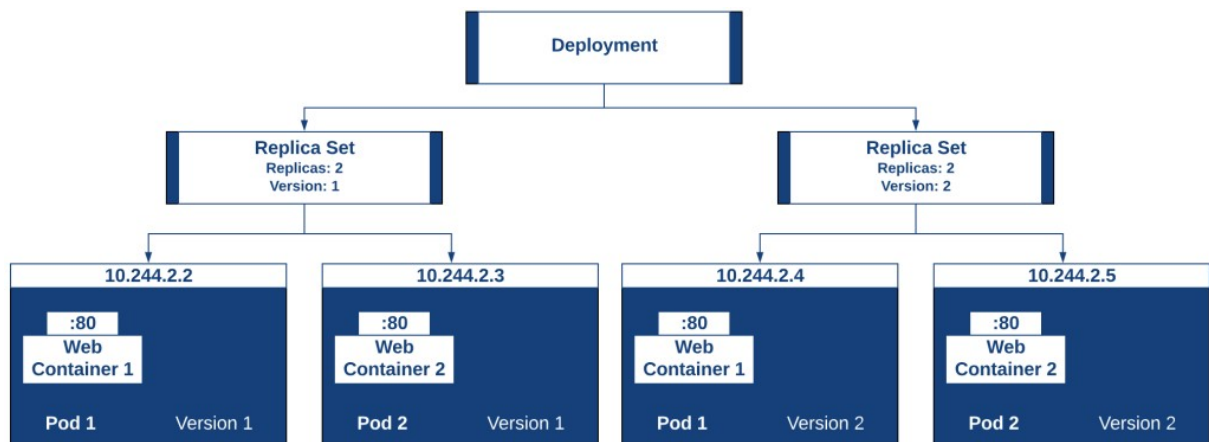
On peut utiliser son numéro de port pour acceder a son application



On supprime son pod et le réseau

```
ludo@openstack:~$ kubectl delete -f nginx-pod.yaml

ludo@openstack:~$ kubectl delete service nginx
```



## Déploiement d'une application Apache PHPMariadb

```
ludo@openstack:~$ cd k8s/apache-php
```

### Créer le stockage pour les serveurs Web

```
ludo@openstack:~$ vim apache-storage.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: apache-data
  namespace: #votre namespace
spec:
  storageClassName: nfs-client
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

### On applique le manifest

```
ludo@openstack:~$ kubectl apply -f apache-storage.yaml
```



On crée le réseau pour les serveur Web

```
ludo@openstack:~$ vim apache-network.yaml
apiVersion: v1
kind: Service
metadata:
  name: apache-service
  namespace: # votre espace de nom
spec:
  type: NodePort
  selector:
    app: web
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

On affiche le stockage et le réseau

```
ludo@openstack:~$ kubectl get service,pvc
NAME                                TYPE             CLUSTER-IP      EXTERNAL-IP
PORT(S)          AGE
service/apache-service  NodePort         10.254.179.244   <none>
80:32714/TCP      53s

persistentvolumeclaim/apache-data  Bound          ac503a9144      50Mi      RWX
nfs-client                        116s
```

On peut alors déployer son application

```
ludo@openstack:~$ vim apache-php-vol.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
    name: apache-php
spec:
  replicas: 2
  ....

ludo@openstack:~$ kubectl apply -f apache-php-vol.yaml
kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/apache-php-69fd4d7fdc-tbvhh    1/1     Running   0           108s
pod/apache-php-69fd4d7fdc-tqnnb    1/1     Running   0           111s
```

