‘

Déploiement d’une Infrastructure Cloud Privée/Hybride IaaS OpenStack

I. Aperçu

 # Le Cloud ! Le Cloud !

Les clouds publics comme AWS d'Amazon, GCP de Google, et Azure de Microsoft semblent attirer toute l'attention ces jours-ci, mais n'oublions pas le héros méconnu du Cloud… le Cloud Privé. Soyons honnêtes, de nombreuses entreprises fonctionnent encore sur des Clouds Privés, que ce soit pour des raisons réglementaires ou en raison de restrictions budgétaires (les factures de cloud peuvent rapidement devenir ridiculement élevées).

Il existe deux types de Clouds Privés : internes et hébergés.

- Les Clouds Privés sont hébergés dans les bureaux ou centres de données d'une organisation;
- Les Clouds Privés Hébergés peuvent être mono-tenant (centres de données dédiés à une seule entreprise) ou multi-tenant (centres de données hébergeant plusieurs entreprises).

Des Cadres comme OpenStack, Apache CloudStack, et Azure Stack permettent aux organisations de déployer et gérer leurs Clouds Privés de manière efficace. Parmi ceux-ci, OpenStack se distingue comme une plateforme open-source puissante offrant flexibilité, évolutivité et contrôle sur votre infrastructure.

Modèles de Cloud Computing principaux:

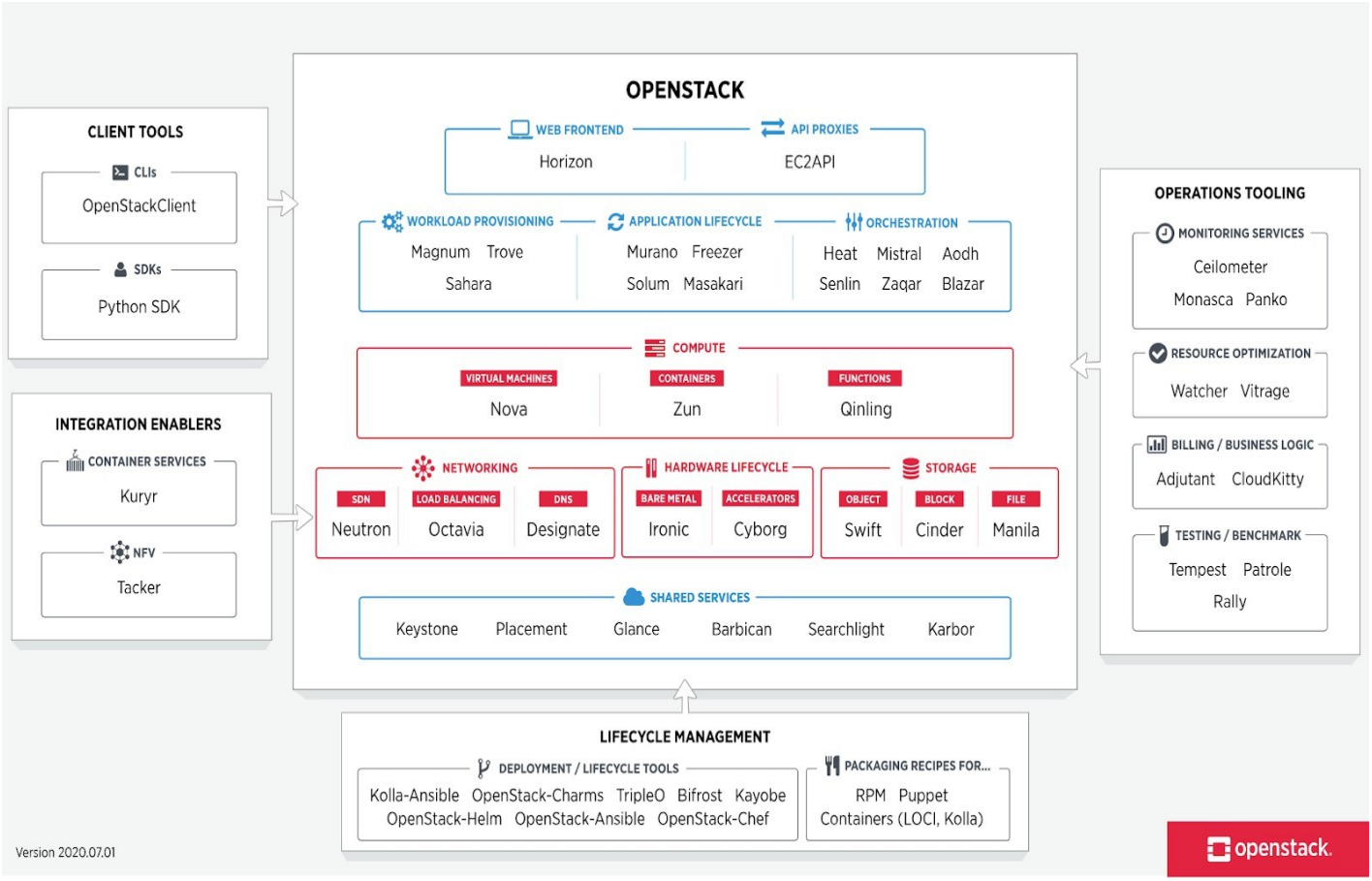
- **IaaS (Infrastructure en tant que Service)** : Fournit des ressources virtualisées telles que des machines virtuelles, du stockage et des réseaux via Internet, permettant aux utilisateurs de contrôler l'infrastructure sans matériel physique (par exemple, Amazon EC2, Google Compute Engine). Idéal pour les entreprises recherchant de la flexibilité sans posséder de serveurs.
- **PaaS (Plateforme en tant que Service)** : Offre une plateforme pour construire, déployer et gérer des applications sans se soucier de l'infrastructure. Utile pour les développeurs concentrés sur la programmation tandis que le fournisseur gère l'infrastructure (par exemple, Google App Engine, Heroku).
- **SaaS (Logiciel en tant que Service)** : Fournit des logiciels via Internet sur abonnement, accessibles par le biais de navigateurs sans nécessiter d'installation (par exemple, Google

Workspace, Salesforce). Idéal pour les utilisateurs ayant besoin d'un accès facile et de logiciels sans maintenance.

Alors, qu'est-ce qu' OpenStack ?

OpenStack est une plateforme de cloud computing open-source qui fournit un ensemble d'outils pour créer et gérer des infrastructures cloud évolutives. Déployé en tant que Infrastructure-as-a Service (IaaS), il permet aux utilisateurs de créer et de gérer des clouds publics et privés, offrant un déploiement flexible des machines virtuelles, du stockage et des ressources réseau.

Architecture d' OpenStack



Composants principaux d'OpenStack:

Nova

C’est le service de calcul d'OpenStack, gère le cycle de vie des machines virtuelles (VM) et d'autres ressources de calcul en prenant en charge des tâches telles que le provisionnement, la planification et la mise à l'échelle des instances.

Neutron,

C’est le service réseau d'OpenStack, offre le "Réseau en tant que service" (NaaS), permettant aux utilisateurs de créer et de gérer des réseaux, des sous-réseaux et des routeurs dans un environnement cloud. Avec des fonctionnalités telles que les IP flottantes, les VLAN, l'équilibrage de charge et les groupes de sécurité, Neutron garantit l'isolation des réseaux et l'évolutivité, offrant un cadre robuste de réseau défini par logiciel (SDN).

Cinder

C’est le service de stockage en bloc d'OpenStack, offre un stockage persistant qui reste disponible indépendamment des instances de calcul, ce qui le rend idéal pour les données critiques. Avec des fonctionnalités telles que la gestion des volumes, les instantanés et la prise en charge de divers systèmes de stockage, Cinder garantit un stockage fiable et évolutif pour les applications et les charges de travail.

Glance

le service d'images d'OpenStack, permet aux utilisateurs de stocker, découvrir et récupérer des images de machines virtuelles, qui servent de modèles pour les instances. Prenant en charge divers formats comme RAW et QCOW2, Glance agit comme un référentiel centralisé et s'intègre parfaitement à Cinder et Nova pour un approvisionnement efficace.

Keystone

C’est le service d'identité qui fournit l'authentification et l'autorisation pour tous les services OpenStack. Il gère les identifiants des utilisateurs, les rôles et les permissions, permettant un accès sécurisé aux ressources cloud.

Swift

C’est le service de stockage d'objets conçu pour stocker et gérer de grandes quantités de données non structurées, telles que des images, des vidéos et des sauvegardes. Il offre une solution de stockage évolutive, durable et hautement disponible, qui prend en charge une API RESTful pour un accès facile et une intégration avec d'autres services OpenStack.

Placement

C’est le service responsable du suivi des ressources disponibles dans l'infrastructure cloud et de l'optimisation de l'allocation des ressources pour les instances. Il fournit une API unifiée qui aide des services comme Nova et Cinder à prendre des décisions éclairées sur l'emplacement des charges de travail en fonction de la disponibilité des ressources et des politiques.

Heat

C’est le service d'orchestration utilisé pour automatiser le déploiement et la gestion des ressources cloud. Il permet aux utilisateurs de définir l'infrastructure sous forme de code à l'aide de modèles, facilitant la création, la mise à jour et la mise à l'échelle d'environnements complexes à la demande.

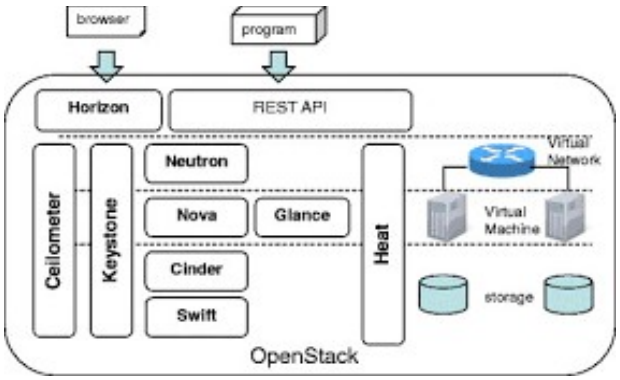
Horizon

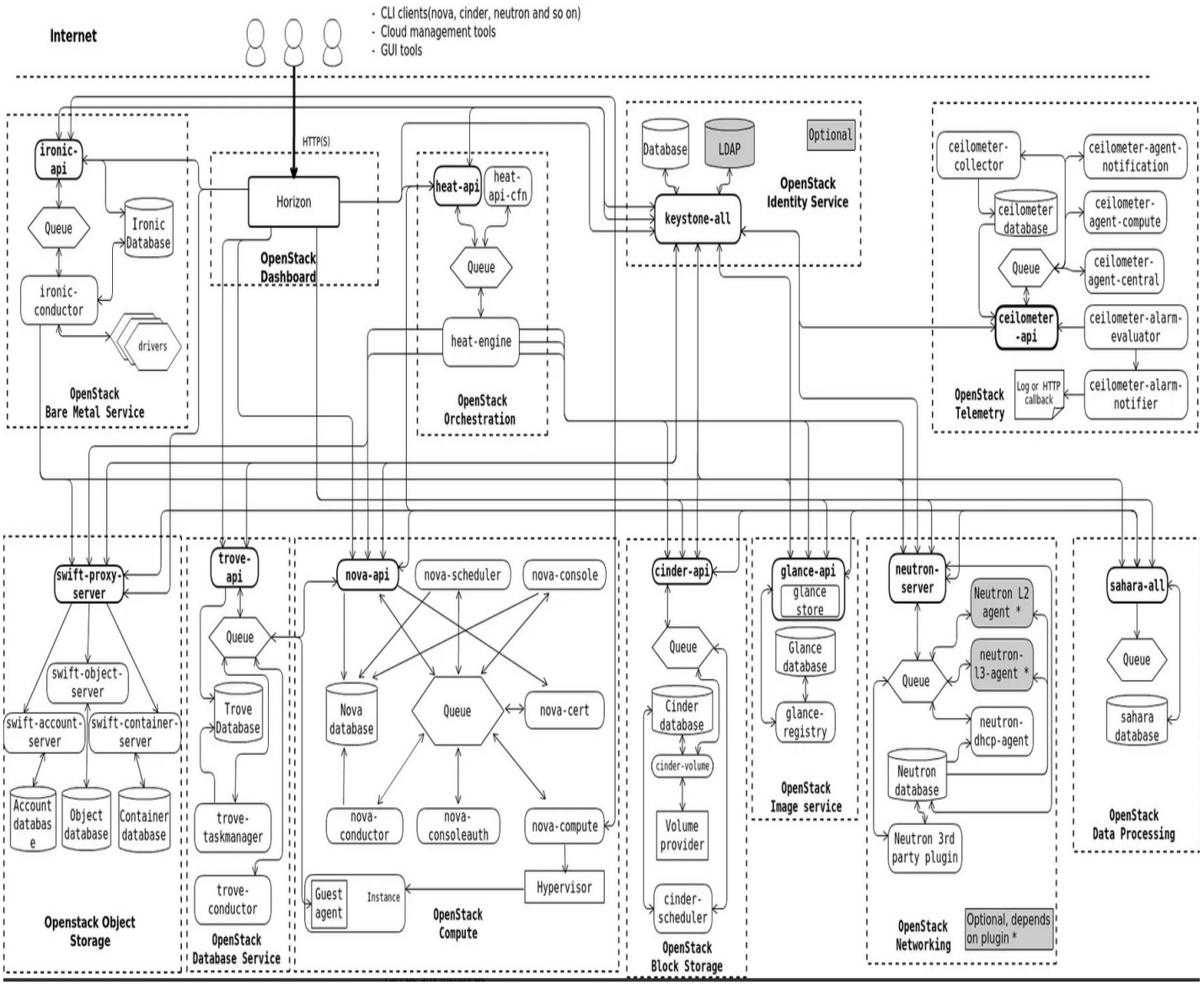
C’est le tableau de bord web officiel qui offre une interface utilisateur pour gérer et interagir avec les services OpenStack. Il permet aux administrateurs et aux utilisateurs d'effectuer des tâches telles que la provision des ressources, la gestion des identités et la surveillance de l'infrastructure cloud de manière simple et intuitive.

Ceilometer

C’est le service de télémétrie dans OpenStack, responsable de la surveillance et de la mesure de l'utilisation des ressources à travers l'environnement cloud.

OpenStack possède une architecture modulaire où chaque service (par exemple, Nova, Glance, Cinder, Swift) communique via une file de messages (comme RabbitMQ) et une API RESTful. Cette conception permet flexibilité et évolutivité, permettant aux utilisateurs de déployer uniquement les services nécessaires.





Concepts clés d'OpenStack:

- **Projet (Tenant)**: Un regroupement logique de ressources pour des utilisateurs ou organisations, permettant l'isolation des ressources et la multi-location. Plusieurs projets peuvent être utilisés pour différents environnements (par exemple, développement, test).
- **Instance**: Une machine virtuelle gérée par Nova, créée à partir d'images disque dans Glance. Les utilisateurs peuvent lancer, gérer et terminer des instances selon leurs besoins.
- **Flavor**: Une configuration prédéfinie d'instance (CPU, RAM, espace disque) qui détermine les caractéristiques de performance.

- **Security Group**: Un pare-feu virtuel avec des règles pour le trafic entrant et sortant, sécurisant l'accès aux instances.
- **Floating IP**: Une adresse IP qui peut être attribuée dynamiquement à une instance pour un accès externe.
- **Volume**: Un périphérique de stockage bloc qui fournit un stockage persistant pour les instances, gérable indépendamment des instances.

Flux de travail OpenStack et interactions des composants :

OpenStack fonctionne comme une plateforme cloud cohésive grâce à la collaboration de ses composants modulaires, permettant une gestion et un provisionnement efficaces des ressources :

1. **Authentification et Accès** : **Keystone** gère l'authentification et l'autorisation des utilisateurs, émettant des jetons qui accordent un accès sécurisé aux autres services OpenStack.
2. **Provisionnement des Ressources** : Les utilisateurs peuvent demander des ressources telles que des instances, des réseaux et du stockage via le tableau de bord **Horizon** ou **l'interface en ligne de commande OpenStack.**
3. **Intégration de l'Informatique et du Réseau** : **Nova** provisionne les instances de calcul, utilisant **Neutron** pour configurer la connectivité réseau, attribuer des IP et appliquer l'isolation.
4. **Gestion du Stockage** : **Cinder** fournit un stockage bloc persistant, en attachant des volumes aux instances, tandis que **Swift** gère le stockage de données non structurées à grande échelle.
5. **Gestion des Images** : **Nova** récupère les images et configurations du système d'exploitation depuis **Glance** pour lancer efficacement des machines virtuelles.
6. **Orchestration et Mise à l'Échelle** : **Heat** permet l'orchestration d'applications cloud complexes, en mettant à l'échelle les ressources selon des modèles prédéfinis.
7. **Surveillance et Facturation** : **Ceilometer** collecte les métriques d'utilisation à travers des services tels que **Nova, Cinder et Neutron**, aidant à la surveillance des performances et à la facturation.

Outils différents pour déployer OpenStack :

1. **DevStack:**

- **Description** : Un ensemble de scripts pour déployer rapidement OpenStack dans un environnement de développement.
- **Cas d'utilisation** : Idéal pour les développeurs et les testeurs.
- **Avantages** : Configuration rapide, facile à modifier, excellent pour les tests.
- **Inconvénients** : Pas pour la production, évolutivité limitée.

2. **OpenStack Ansible:**
    - **Description** : Utilise des playbooks Ansible pour un déploiement OpenStack plus adapté à la production.
    - **Cas d'utilisation** : Idéal pour des déploiements flexibles, évolutifs et automatisés.
    - **Avantages** : Hautement configurable, prend en charge les configurations multi-nœuds, adapté à la production.
    - **Inconvénients** : Nécessite des connaissances en Ansible, configuration plus complexe.
3. **Charms (Juju):**
    - **Description** : Déploiement model-driven utilisant Juju, emballé sous forme de charms.
    - **Cas d'utilisation** : Idéal pour les environnements à grande échelle et la gestion de services.
    - **Avantages** : Gestion déclarative, intégration facile, évolutif.
    - **Inconvénients** : Nécessite la compréhension de Juju et des charms, complexité de configuration initiale.
4. **Packstack:**
    - **Description** : Utilise des modules Puppet pour installer OpenStack sur des serveurs existants.
    - **Cas d'utilisation** : Bon pour les installations de taille petite à moyenne, preuve de concept.
    - **Avantages** : Configuration simple, prend en charge les configurations multi-nœuds.
    - **Inconvénients** : Flexibilité limitée, nécessite des connaissances en Puppet pour les personnalisations.
5. **Kolla-Ansible:**
    - **Description** : Utilise Ansible pour déployer et gérer les services OpenStack avec des conteneurs Docker.
    - **Cas d'utilisation** : Idéal pour les utilisateurs souhaitant des déploiements OpenStack conteneurisés.
    - **Avantages** : Prend en charge les services conteneurisés, flexible, évolutif pour les grands déploiements.
    - **Inconvénients** : Nécessite des connaissances en Docker et Ansible, peut être complexe à configurer au début.

OpenStack peut être déployé dans différentes configurations, principalement en déploiements **tout-en-un** ou **multi-nœuds**, en fonction de la taille et de la complexité de l'environnement.

- Un **déploiement tout-en-un** regroupe tous les services OpenStack (comme **compute**, **storage**, and **networking**) sur une seule machine, ce qui le rend idéal pour les environnements à petite échelle, le développement et les tests. Il offre simplicité et installation rapide, mais présente une évolutivité et des performances limitées en raison des contraintes de ressources d'un seul nœud.
- D'autre part, un **déploiement multi-nœuds** répartit les services OpenStack sur plusieurs machines physiques ou virtuelles, offrant une meilleure évolutivité, des performances accrues et une tolérance aux pannes. Cette configuration est plus adaptée aux environnements de production, où différents services peuvent fonctionner sur des nœuds dédiés pour garantir une haute disponibilité et une utilisation optimale des ressources.

Voyons maintenant le déploiement d'OpenStack avec Kolla-Ansible : un déploiement tout-en-un.

II. OpenStack Kolla-Ansible all-in-one Deployment Instructions

Prérequis :

- Ubuntu 22.04 LTS ou une version antérieure.

La machine hôte doit répondre aux exigences minimales suivantes :

- au moins 2 interfaces réseau (interne, externe), vous pouviez aussi ajouter un separer pour internet connexion.
- au moins 8 Go de mémoire principale,
- au moins 50 Go d'espace disque.

Ce guide explique comment déployer OpenStack avec Kolla-Ansible, un outil conçu pour simplifier le déploiement d'OpenStack à l'aide de conteneurs. Il comprend des instructions pour installer les dépendances, créer un environnement virtuel et configurer le système pour le déploiement.

Les étapes du déploiement seront exécutées à l'aide du script shell fourni. Mais ces étapes n'ont pas besoin d'être effectuées manuellement. Une fois que vous les avez comprises, vous pouvez simplement exécuter ce script, qui automatisera tout le processus.

Voici ce que vous devez faire :

1. **Télécharger le script :** Clonez ou téléchargez le dépôt contenant le script de déploiement.

2. Rendre le script exécutable :

```
sudo chmod +x ./opkdeployment_script.sh
```

puis exécutez-le :

```
./opkdeployment_cript.sh
```

3. Mise à jour et installation des dépendances

Cette section garantit que votre système est à jour et installe les dépendances nécessaires pour le déploiement d'OpenStack.

```
sudo apt update -y && sudo apt-get full-upgrade -y
sudo apt install -y python3-dev libffi-dev gcc libssl-dev python3-selinux python3-setuptools python3-venv net-tools git
```

4. Configuration de l'environnement virtuel Python

Nous allons créer un environnement virtuel pour isoler le processus de déploiement :

```
python3 -m venv "kolla-venv"
```

Activez l'environnement virtuel :

```
source kolla-venv/bin/activate
```

5. Mise à jour de pip et installation de Wheel

Nous mettons à jour pip et installons wheel, un package nécessaire pour la construction d'autres dépendances :

```
pip install -U pip  # Upgrade pip
pip install wheel  # Install wheel for package building
```

6. Configuration de Docker

Kolla-Ansible utilise Docker pour exécuter les services OpenStack dans des conteneurs. Les étapes suivantes installent Docker :

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-ke
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://downlo
sudo apt update
sudo apt install docker-ce -y
sudo usermod -aG docker ${USER}
```

vérifier l'installation de Docker

```
docker --version
```

7. Installation et configuration d'Ansible

Ansible est un outil d'automatisation IT open-source qui simplifie et automatise divers processus IT manuels, tels que l'approvisionnement, la gestion de configuration, le déploiement d'applications et l'orchestration. Nous l'installerons via pip :

```
pip install "ansible-core>=2.15,<2.16.99"
```

Configurez Ansible pour éviter la vérification des clés d'hôte et définissez le parallélisme pour les tâches en créant un fichier ansible.cfg :

```
sudo mkdir -p /etc/ansible
sudo nano /etc/ansible/ansible.cfg
```

Ajoutez la configuration suivante :

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

what is Host Key Checking:

- Qu'est-ce que la vérification des clés d'hôte ? : La vérification des clés d'hôte est une fonctionnalité de sécurité de SSH qui vérifie l'authenticité du serveur auquel vous vous connectez. Lorsque vous vous connectez à un serveur pour la première fois, SSH vous demandera de vérifier la clé d'hôte du serveur (c'est-à-dire son identification unique). Si

vous vous connectez à nouveau à l'avenir, SSH comparera la clé du serveur à celle stockée dans votre fichier ~/.ssh/known_hosts pour s'assurer qu'il s'agit du même serveur.

- Dans la configuration, nous l'avons définie sur false :

```
host_key_checking=False
```

- Pourquoi le désactiver dans Ansible ? Dans des environnements automatisés, comme lorsqu'on utilise Ansible pour gérer plusieurs serveurs, vous ne voudrez peut-être pas confirmer manuellement la clé d'hôte à chaque fois qu'Ansible se connecte à un serveur. Cela peut ralentir l'automatisation et être peu pratique dans des environnements à grande échelle où vous gérez un grand nombre d'hôtes.

Pipelining:

- Qu'est-ce que le **Pipelining** ? Le **pipelining** est une technique utilisée pour accélérer les commandes basées sur SSH. Par défaut, lorsque Ansible exécute une tâche, il envoie une connexion SSH distincte pour chaque commande ou tâche. Cela peut être lent, surtout lorsque vous exécutez de nombreuses tâches sur plusieurs hôtes.
- Avec le **pipelining**, plusieurs commandes peuvent être exécutées sur une seule connexion SSH, réduisant ainsi la surcharge liée à l'ouverture répétée de nouvelles connexions pour chaque tâche.
- Dans la configuration, nous l'avons définie sur true :

```
pipelining=True
```

- Pourquoi l'activer dans Ansible ? Activer le pipelining permet d'améliorer la vitesse des playbooks Ansible, en particulier dans les environnements où de nombreuses tâches sont exécutées sur plusieurs hôtes. Cela réduit la surcharge des connexions SSH et rend le processus plus efficace.

Forks:

- Qu'est-ce que les **Forks** ? Les **forks** font référence au nombre de processus parallèles qu'Ansible utilise pour exécuter des tâches sur plusieurs hôtes. Par défaut, **Ansible** exécute les tâches de manière séquentielle sur un hôte à la fois. Lorsque vous avez un grand nombre d'hôtes, cela peut rendre les playbooks lents.
- Dans la configuration, nous avons défini

```
forks=100
```

- Pourquoi augmenter les Forks dans Ansible ? En augmentant le nombre de forks, vous pouvez exécuter des tâches en parallèle sur plusieurs hôtes. Cela peut accélérer considérablement l'exécution des tâches lors de la gestion de nombreux serveurs. Cependant, cela nécessite également plus de ressources système (CPU et mémoire) sur la machine exécutant Ansible, il convient donc de le configurer en fonction des ressources disponibles.

8. Installation de Kolla-Ansible

Kolla-Ansible est un outil de déploiement open-source pour **OpenStack** qui utilise **Ansible** pour automatiser le déploiement et la gestion des services OpenStack. Il simplifie le processus d'installation, de configuration et de maintenance d'OpenStack sur diverses plateformes d'infrastructure, telles que les machines virtuelles ou les serveurs bare-metal.

Les principales fonctionnalités de Kolla-Ansible comprennent :

- **Déploiement containerisé : Kolla-Ansible** déploie les **services OpenStack** dans des conteneurs Docker, ce qui facilite la gestion et l'évolutivité.
- **Intégration avec Ansible :** Il utilise Ansible pour l'automatisation, offrant flexibilité et répétabilité dans le processus de déploiement.
- **Mises à jour progressives :** Kolla-Ansible facilite les mises à jour progressives des composants OpenStack, minimisant ainsi les temps d'arrêt.
- **Personnalisabilité** : Il offre un large éventail d'options de configuration, permettant la personnalisation pour s'adapter à différents cas d'utilisation et environnements.

Installez-le via pip :

```
pip install git+https://opendev.org/openstack/kolla-ansible@stable/2024.1
```

9. Configurez le répertoire Kolla-Ansible et assurez-vous de la propriété :

```
sudo mkdir -p /etc/kolla
sudo chown $USER:$USER /etc/kolla
```

10. Copie des fichiers de configuration

Copiez les fichiers de configuration d'exemple depuis l'environnement virtuel Kolla-Ansible vers le répertoire /etc/kolla :

```
cp -r "$HOME/kolla-venv/share/kolla-ansible/etc_examples/kolla/"* /etc/kolla
cp "$HOME/kolla-venv/share/kolla-ansible/ansible/inventory/"* .
```

Alors, quels sont ces fichiers :

Ces fichiers sont `globals.yml`, `password.yml`, et les inventaires tels que (all-in-one et multi-node). Commençons par comprendre ces fichiers, en commençant par le fichier de configuration `globals.yml` :

Alors, qu'est-ce que le fichier `globals.yml` :

Le fichier **`globals.yml`** est un fichier de configuration clé dans Kolla-Ansible qui définit les paramètres globaux pour les déploiements OpenStack. Il joue un rôle crucial dans la

configuration du déploiement en spécifiant des paramètres pour le réseau, la configuration des services, la gestion des ressources et la sécurité. Que ce soit pour un environnement de test simple ou une configuration de production complexe en multi-nœuds, le fichier `globals.yml` contrôle quels composants OpenStack sont activés ou désactivés, comment les services sont configurés et comment les conteneurs sont gérés. Il sert de base pour l'ensemble du processus de déploiement, garantissant que tous les services fonctionnent ensemble selon les paramètres définis.

Exemple des sections et paramètres clés :

- **Configuration du réseau :** Spécifie les paramètres réseau pour la communication interne et externe.

  Exemple:

  - `network_interface: "eth0"`

  - `neutron_external_interface: "eth1"`

- **Configuration des services :** Active ou désactive des services OpenStack spécifiques tels que Ceph, RabbitMQ, Cinder, Horizon, Haproxy, Glance, etc.

  Exemple:

  - `enable_ceph: "yes"`

  - `enable_rabbitmq: "yes"`

Cela permet de personnaliser votre déploiement OpenStack en choisissant uniquement les services (Nova, Neutron, etc.) dont vous avez besoin.

- **Sécurité et TLS :** Active le chiffrement TLS pour une communication sécurisée entre les services.

  Example:

  `enable_tls: "yes"ssl_certificate: "/etc/ssl/certs/my_cert.pem"`

- **Paramètres Docker et conteneurs :** Configure les paramètres Docker comme le pilote de stockage et le registre.

  Example:

  `docker_storage_driver: "overlay2"`

- **Fuseau horaire et paramètres régionaux :** Définit le fuseau horaire et les paramètres régionaux système.

  Example:

  `timezone: "UTC"locale: "en_US.UTF-8"

Et plus de configurations à mettre en place selon vos besoins, vous pouvez jeter un œil au fichier de configuration `globals.yml` à partir d'ici.

Un autre fichier de configuration à connaître est le fichier `passwords.yml` :

Qu'est-ce que le fichier `passwords.yml` ?

Le fichier **`passwords.yml`** est un fichier de configuration utilisé par **Kolla-Ansible** pour définir les mots de passe des services et composants clés d'OpenStack. Ces services incluent le système d'authentification (Keystone), les bases de données (par exemple, **MySQL, MongoDB**), les files d'attente de messages (**RabbitMQ**) et d'autres services OpenStack.

Dans OpenStack, les mots de passe sont essentiels pour sécuriser la communication entre les services OpenStack. Le fichier `passwords.yml` offre une manière sécurisée et automatisée de gérer ces identifiants.

Encore un autre fichier de configuration à connaître : les fichiers d'inventaire.

Dans le déploiement d'OpenStack, le fichier d'inventaire joue un rôle crucial en définissant les nœuds (machines ou VM) impliqués dans le déploiement d'OpenStack et en attribuant des rôles spécifiques à chacun.

- Pour un déploiement **all-in-one,** le fichier d'inventaire spécifie que tous **les services OpenStack,** tels que Keystone, Nova, Glance et Neutron, sont installés sur une seule machine. Cette configuration est idéale pour les tests, le développement ou les environnements à petite échelle où la simplicité et la rapidité de configuration sont importantes. Tous les services, y compris les rôles de contrôle, de calcul et de réseau, sont exécutés sur le même nœud, ce qui facilite la configuration, mais est moins adapté à la production en raison de l'absence de redondance et d'évolutivité.

  Exemple de fichier d'inventaire All-in-One :

```
# Inventory file for an all-in-one deployment

[all]
localhost ansible_connection=local

[control]
localhost

[compute]
localhost

[network]
localhost
```

- En revanche, un déploiement **multi-nœuds** répartit les **services OpenStack** sur plusieurs machines, chaque nœud se voyant attribuer des rôles spécifiques tels que **contrôleur**, **compute** et **neutron**. Cette configuration est idéale pour les environnements de production car elle offre évolutivité, redondance et haute disponibilité, avec la possibilité

d'ajouter davantage de nœuds selon les besoins. Le fichier d'inventaire dans une configuration multi-nœuds liste tous les nœuds et leurs rôles, garantissant une distribution efficace des services. Cette approche est plus complexe, mais nécessaire pour les déploiements OpenStack à grande échelle, où l'infrastructure doit supporter davantage d'utilisateurs et de charges de travail.

Exemple de fichier d'inventaire multi-nœuds :

```
# Inventory file for a multi-node deployment

[all]
controller1 ansible_host=192.168.0.1
controller2 ansible_host=192.168.0.2
compute1 ansible_host=192.168.0.3
compute2 ansible_host=192.168.0.4

[control]
controller1
controller2

[compute]
compute1
compute2

[network]
controller1
```

11. Vérification de la configuration d'Ansible

Vérifiez qu'Ansible est installé correctement et peut atteindre les hôtes :

```
ansible --version
ansible -i ./all-in-one all -m ping  # Ping the hosts to verify Ansible is working
```

12. Génération des mots de passe pour Kolla-Ansible

Générez les mots de passe pour Kolla-Ansible :

```
kolla-genpwd
```

13. Configurer les mots de passe

Changez le **`keystone_admin_password`** en "kolla" dans le fichier **`passwords.yml`** :

```
sed -i 's#keystone_admin_password:.*#keystone_admin_password: kolla#g' /etc/kolla/passwords.yml
```

14. Configuration du fichier Kolla `globals.yml`

```
sudo nano /etc/kolla/globals.yml
```

Copiez ceci, en sachant que vous découvrirez d'autres configurations dans le fichier `globals.yml` :

```
---
workaround_ansible_issue_8743: yes
kolla_base_distro: "ubuntu"
openstack_release: "2024.1"
network_interface: "ens20"
neutron_external_interface: "ens19"
kolla_internal_vip_address: "192.168.3.5"  # Set internal VIP address
enable_haproxy: "no"  # Disable HAProxy by default
nova_compute_virt_type: "qemu"  # Set default virtualization type for Nova

## fqdn
#kolla_external_fqdn: "opkext.test.link"  # Set external FQDN (optional)
#kolla_internal_fqdn: "opkint.test.link"  # Set internal FQDN (optional)

## cinder
#enable_cinder: "yes"  # Enable Cinder volume service (optional)
#enable_cinder_backend_lvm: "yes"  # Enable LVM backend for Cinder (optional
#cinder_volume_group: "cinder-volumes"  # Set Cinder volume group (optional)
#enable_cinder_backup: "no"  # Disable Cinder backup service (optional)

## tls
#kolla_enable_tls_internal: "yes"
#kolla_enable_tls_external: "yes"
#kolla_certificates_dir: "/etc/kolla/certificates"
#kolla_external_fqdn_cert: "{{ kolla_certificates_dir }}/haproxy.pem"
#kolla_internal_fqdn_cert: "{{ kolla_certificates_dir }}/haproxy-internal.pe
#kolla_admin_openrc_cacert: "/etc/ssl/certs/ca-certificates.crt"
#kolla_copy_ca_into_containers: "yes"
#openstack_cacert: "/etc/ssl/certs/ca-certificates.crt"


#kolla_enable_tls_backend: "yes"
#kolla_verify_tls_backend: "yes"
#kolla_tls_backend_cert: "{{ kolla_certificates_dir }}/backend-cert.pem"
#kolla_tls_backend_key: "{{ kolla_certificates_dir }}/backend-key.pem"
```

https://github.com/openstack/kolla-ansible/blob/master/etc/kolla/globals.ymlhttps://github.com/openstack/kolla-ansible/blob/master/etc/kolla/globals.yml.

14. Installation des dépendances d'Ansible Galaxy

```
kolla-ansible install-deps
```

15. Ajouter l'utilisateur actuel au groupe Docker

Assurez-vous que votre utilisateur fait partie du groupe Docker :

```
sudo usermod -aG docker $USER
```

Et voilà pour la configuration.

Maintenant, déployons OpenStack.

Maintenant, nous commençons le processus de déploiement proprement dit. Le processus de déploiement commence par l'amorçage des serveurs, l'exécution des pré-vérifications et le déploiement des services OpenStack à l'aide de Kolla-Ansible :

17. Détruire le déploiement précédent (le cas échéant)...

S'il y a un déploiement précédent, détruisez-le avant de poursuivre :

```
kolla-ansible destroy --yes-i-really-really-mean-it -i ./all-in-one
```

18. Générer les certificats (si TLS est configuré)

Si vous utilisez TLS, générez de nouveaux certificats :

```
kolla-ansible certificates -i ./all-in-one  # Optionally generate new certificates
sudo cp /etc/kolla/certificates/ca/root.crt /usr/local/share/ca-certificates/kolla-root.crt # important if making it
sudo update-ca-certificates
```

19. Amorcer les serveurs

Amorcez les serveurs (configurez-les pour préparer les services OpenStack) :

```
kolla-ansible bootstrap-servers -i ./all-in-one -e ansible_sudo_pass=yoursystempassword
```

20. Exécuter les pré-vérifications

Exécutez les pré-vérifications pour vous assurer que tout est en ordre avant le déploiement :

```
kolla-ansible prechecks -i ./all-in-one  # Run prechecks before deploying
```

21. Déployer OpenStack

Lancez le déploiement d'OpenStack :

```
kolla-ansible deploy -i ./all-in-one
```

22. Étape post-déploiement

Après le déploiement, exécutez les tâches post-déploiement :

```
kolla-ansible post-deploy
```

Et voilà, OpenStack est déployé sur votre système en mode all-in-one !

23. Installer le client OpenStack

Pour interagir avec OpenStack, installez les outils clients nécessaires :

```
pip install python-openstackclient -c https://releases.openstack.org/constraints/upper/2024.1

pip install python-neutronclient -c https://releases.openstack.org/constraints/upper/2024.1

pip install python-glanceclient -c https://releases.openstack.org/constraints/upper/2024.1

pip install python-heatclient -c https://releases.openstack.org/constraints/upper/2024.1
```

Vérifiez le serveur (instances en cours d'exécution) :

```
source /etc/kolla/admin-openrc.sh
source /etc/kolla/admin-opnerc-system.sh

openstack server list
```

III. Création d'instance

Maintenant qu'OpenStack en mode all-in-one est configuré, les sections suivantes expliquent les étapes pour créer une machine virtuelle dans OpenStack.

Créer une instance de machine virtuelle (VM) dans **OpenStack** implique plusieurs étapes préparatoires pour assurer un déploiement sans faille :

- Le processus commence par la définition des **flavors**, qui spécifient les ressources de calcul, de mémoire et de stockage allouées à l'instance.

- Ensuite, vous devez préparer les **images**, qui servent de modèles du système d'exploitation pour la VM.
- La configuration des **réseaux** est également essentielle pour permettre la communication entre l'instance et d'autres ressources.
- De plus, vous devrez créer ou sélectionner une **paire de clés** pour un accès SSH sécurisé et configurer les groupes de sécurité pour gérer les règles de trafic réseau de l'instance.

Ces éléments fondamentaux sont cruciaux pour une configuration réussie et sécurisée de l'instance VM dans OpenStack.

1. Création d'image :

Tout d'abord, l'image du système d'exploitation pour la machine virtuelle doit être téléchargée. Pour trouver les images de machines virtuelles compatibles avec OpenStack, visitez https://docs.OpenStack.org/image-guide/obtain-images.html. L'image peut être téléchargée dans OpenStack de deux manières :

1.1 Utilisation de la ligne de commande :

téléchargez l'image :

```
wget https://cloud-images.ubuntu.com/jammy/20240912/jammy-server-cloudimg-amd64.img
```

Téléchargez l'image sur OpenStack :

```
openstack image create --disk-format qcow2 --container-format bare --public \
    --property os_type=linux --file ./jammy-server-cloudimg-amd64.img jammy-cloud-img
```

```
(kolla-venv) grd@opk:~$ openstack image create --disk-format qcow2 --container-format bare --public \
    --property os_type=linux --file ./jammy-server-cloudimg-amd64.img jammy-cloud-img
+------------------+------------------------------------------------------------------------------------------------------+
| Field            | Value                                                                                                |
+------------------+------------------------------------------------------------------------------------------------------+
| container_format | bare                                                                                                 |
| created_at       | 2024-12-24T12:04:36Z                                                                                 |
| disk_format      | qcow2                                                                                                |
| file             | /v2/images/878543f4-d444-44f5-8da9-1cf28ab9e2f6/file                                                 |
| id               | 878543f4-d444-44f5-8da9-1cf28ab9e2f6                                                                 |
| min_disk         | 0                                                                                                    |
| min_ram          | 0                                                                                                    |
| name             | jammy-cloud-img                                                                                      |
| owner            | e7677ebfe40f43e7945848a1c91d3ee3                                                                     |
| properties       | os_hidden='False', os_type='linux', owner_specified.openstack.md5='', owner_specified.openstack.object='images/jammy-cloud-img', |
|                  | owner_specified.openstack.sha256=''                                                                  |
| protected        | False                                                                                                |
| schema           | /v2/schemas/image                                                                                    |
| status           | queued                                                                                               |
| tags             |                                                                                                      |
| updated_at       | 2024-12-24T12:04:36Z                                                                                 |
| visibility       | public                                                                                               |
+------------------+------------------------------------------------------------------------------------------------------+
```

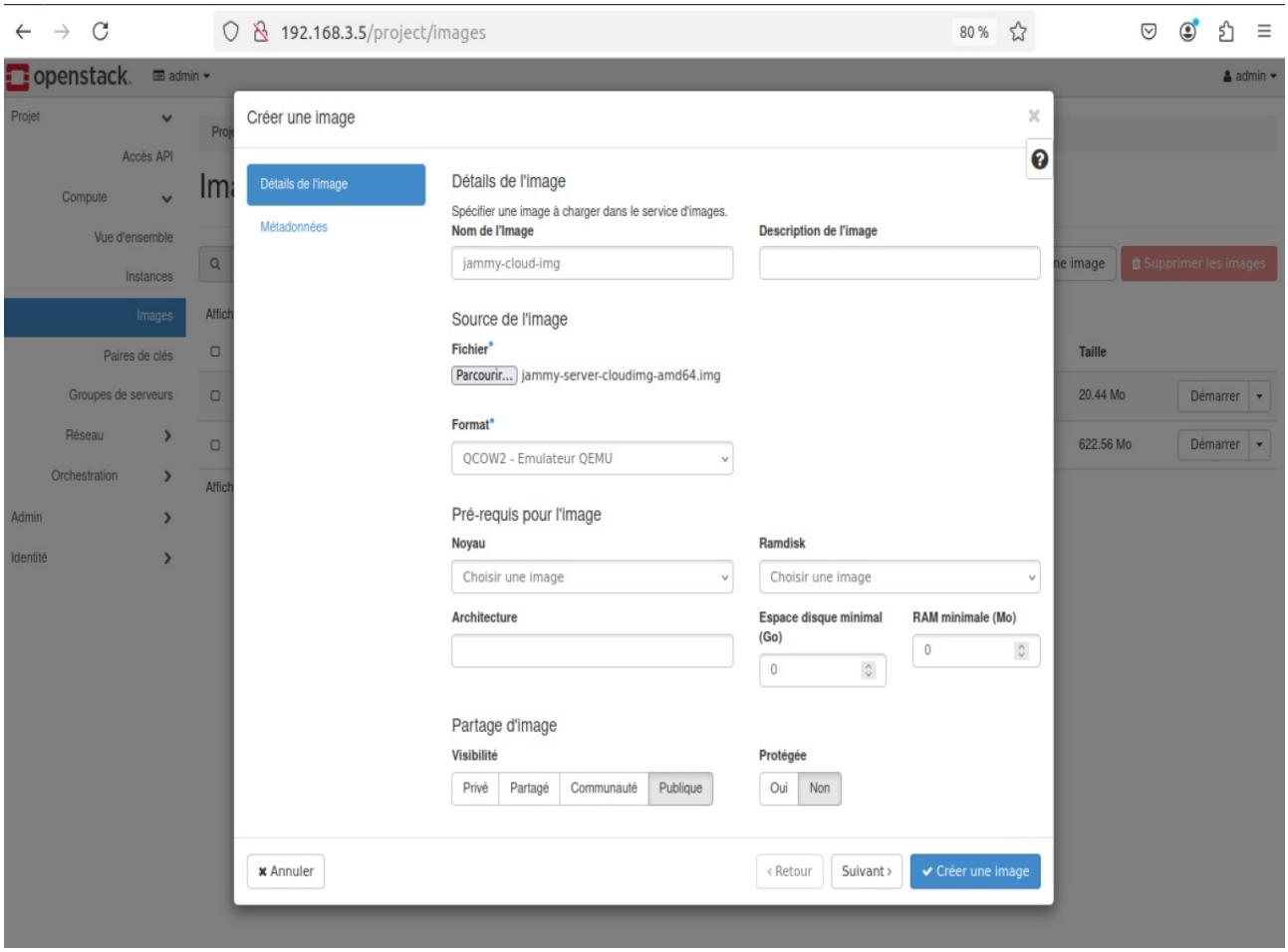
Listez toutes les images :

```
openstack image list
```

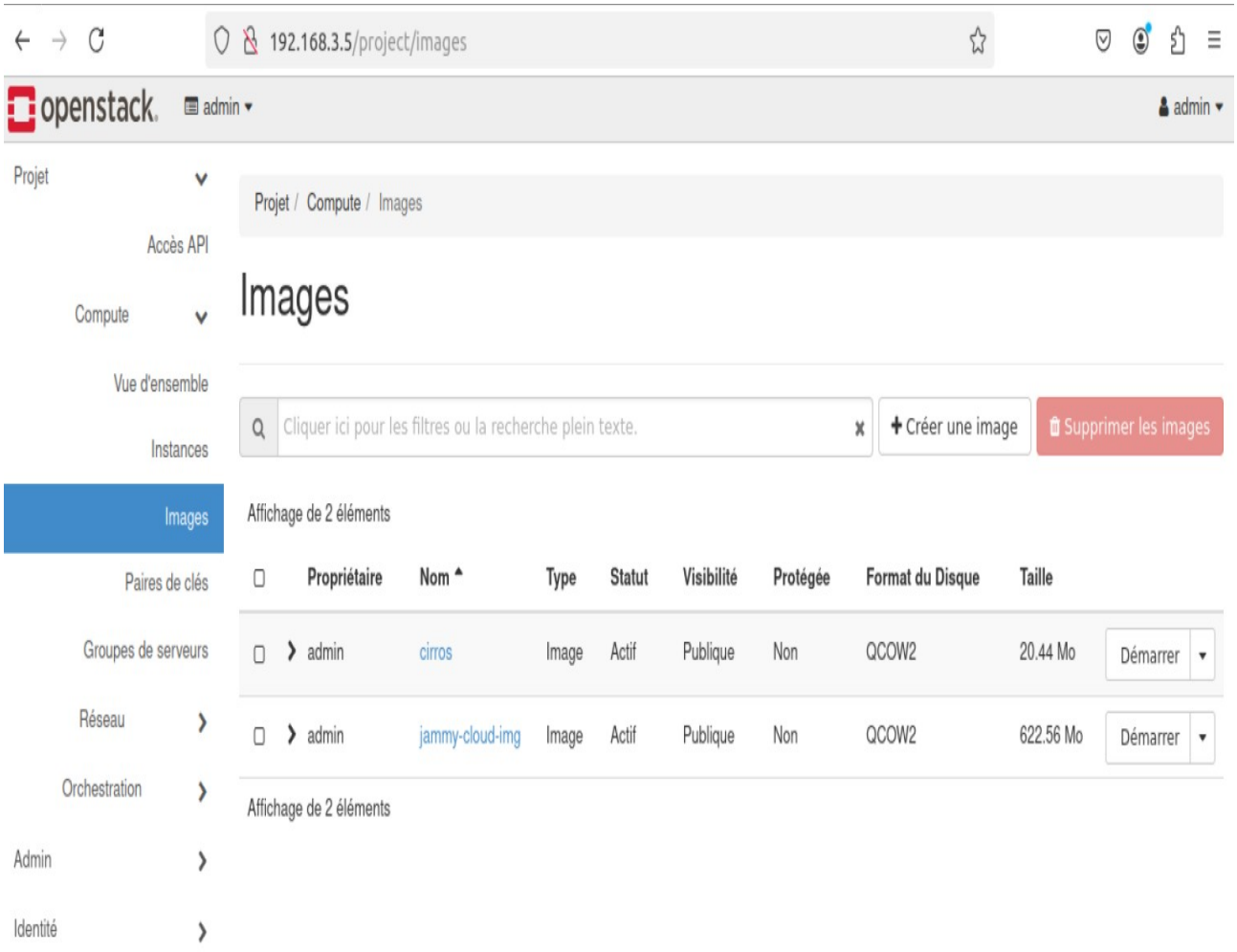
```
(kolla-venv) grd@opk:~$ openstack image list
+--------------------------------------+---------------+--------+
| ID                                   | Name          | Status |
+--------------------------------------+---------------+--------+
| d4fe4755-2450-49c3-a202-105a8f1e84ca | cirros        | active |
| 878543f4-d444-44f5-8da9-1cf28ab9e2f6 | jammy-cloud-img | active |
+--------------------------------------+---------------+--------+
(kolla-venv) grd@opk:~$
```

1.2 Utilisation du tableau de bord OpenStack :

- Télécharger l'image : Allez sur https://docs.OpenStack.org/image-guide/obtain-images.html et téléchargez l'image `jammy-cloud-image`.
- Créer une image : Allez dans l'onglet Images sous l'onglet Calcul, puis sélectionnez Créer une image comme montré dans la figure ci-dessous :



Listez maintenant toutes les images depuis Horizon :



2. Flavor:

Dans OpenStack, un **flavor** définit la capacité de calcul, de mémoire et de stockage des instances de machine virtuelle (VM). Considérez-le comme un modèle spécifiant la taille de votre instance, y compris des paramètres tels que les vCPU, la RAM, l'espace disque et des propriétés supplémentaires comme l'espace de swap ou les métadonnées. Les flavors sont essentiels pour adapter les ressources à différentes charges de travail, garantissant une utilisation efficace de votre infrastructure cloud.

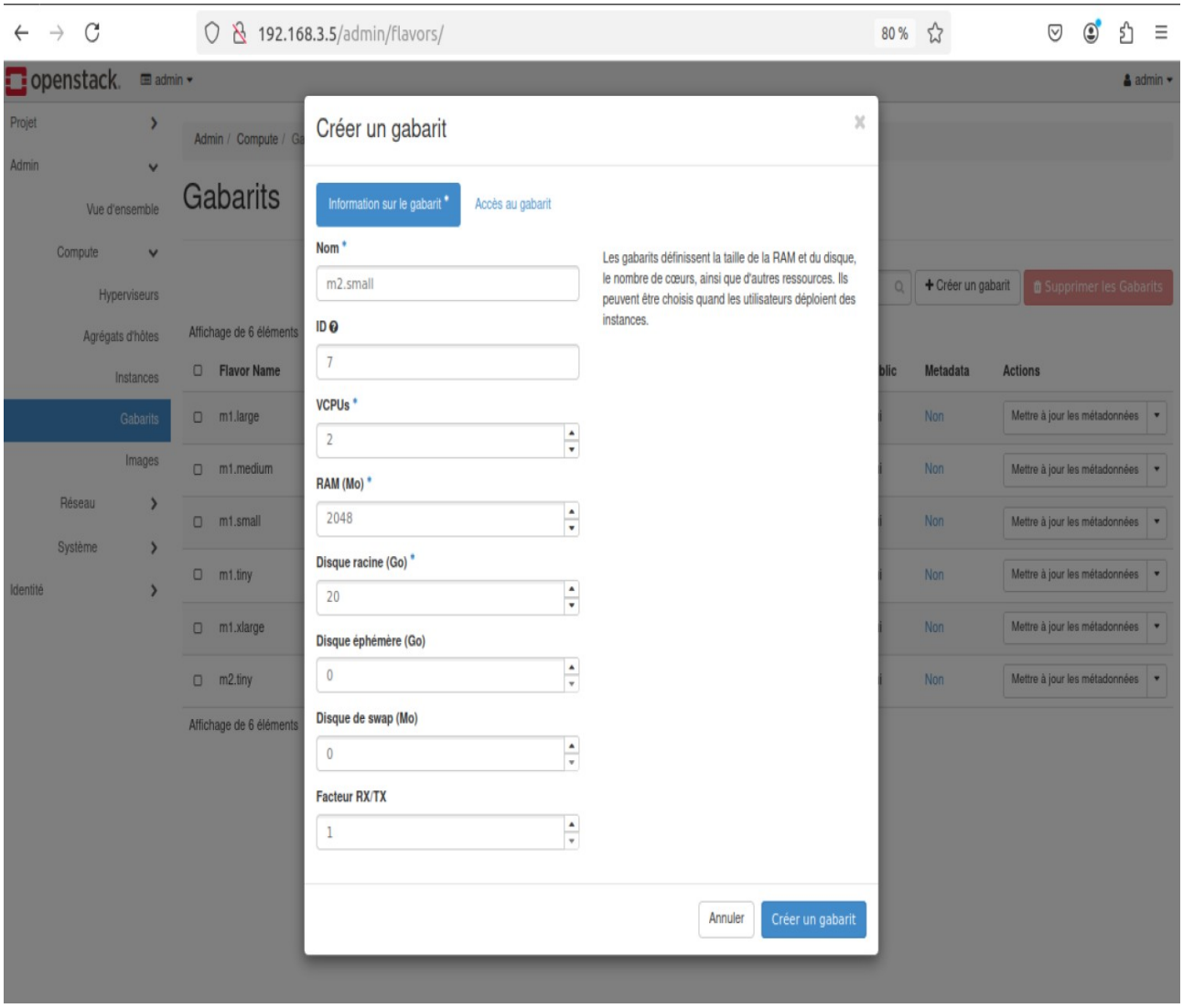
Pour créer un flavor, utilisez la CLI OpenStack ou le tableau de bord Horizon. Par exemple, avec la CLI, vous pouvez exécuter :

```
openstack flavor create --vcpus <num-cpus> --ram <memory-mb> --disk <disk-size-gb> <flavor-name>
```

```
openstack flavor create --id 6 --ram 512 --disk 1 --vcpus 2 m2.tiny
```

```
(kolla-venv) grd@opk:~$ openstack flavor create --id 6 --ram 512 --disk 1 --vcpus 2 m2.tiny
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| description                | None    |
| disk                       | 1       |
| id                         | 6       |
| name                       | m2.tiny |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 512     |
| rxtx_factor                | 1.0     |
| swap                       | 0       |
| vcpus                      | 2       |
+----------------------------+---------+
```

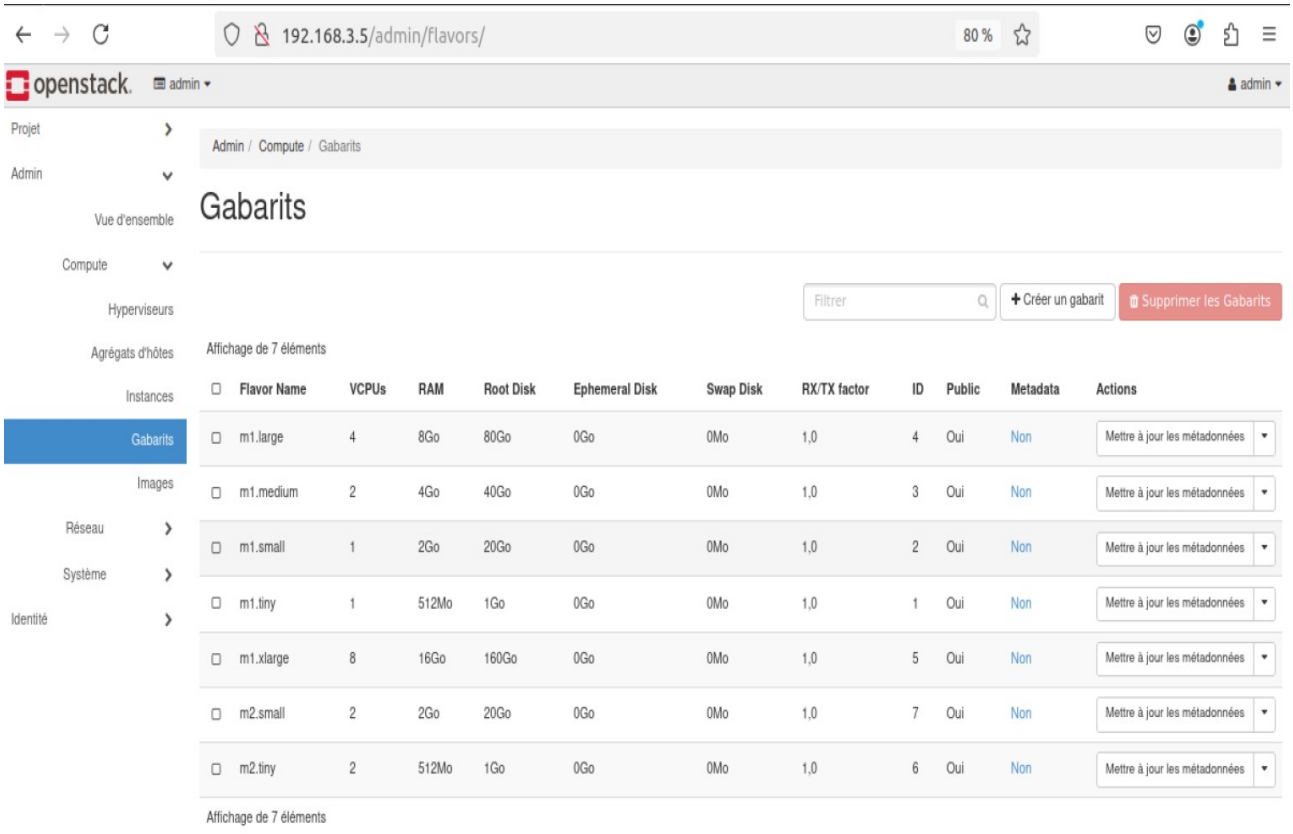
Créons-en un autre depuis Horizon :



Maintenant, listons tous les flavors :

```
(kolla-venv) grd@opk:~$ openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   40 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   80 |         0 |     4 | True      |
| 5  | m1.xlarge | 16384 |  160 |         0 |     8 | True      |
| 6  | m2.tiny   |   512 |    1 |         0 |     2 | True      |
| 7  | m2.small  |  2048 |   20 |         0 |     2 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

Voici également depuis Horizon :



3. Réseaux

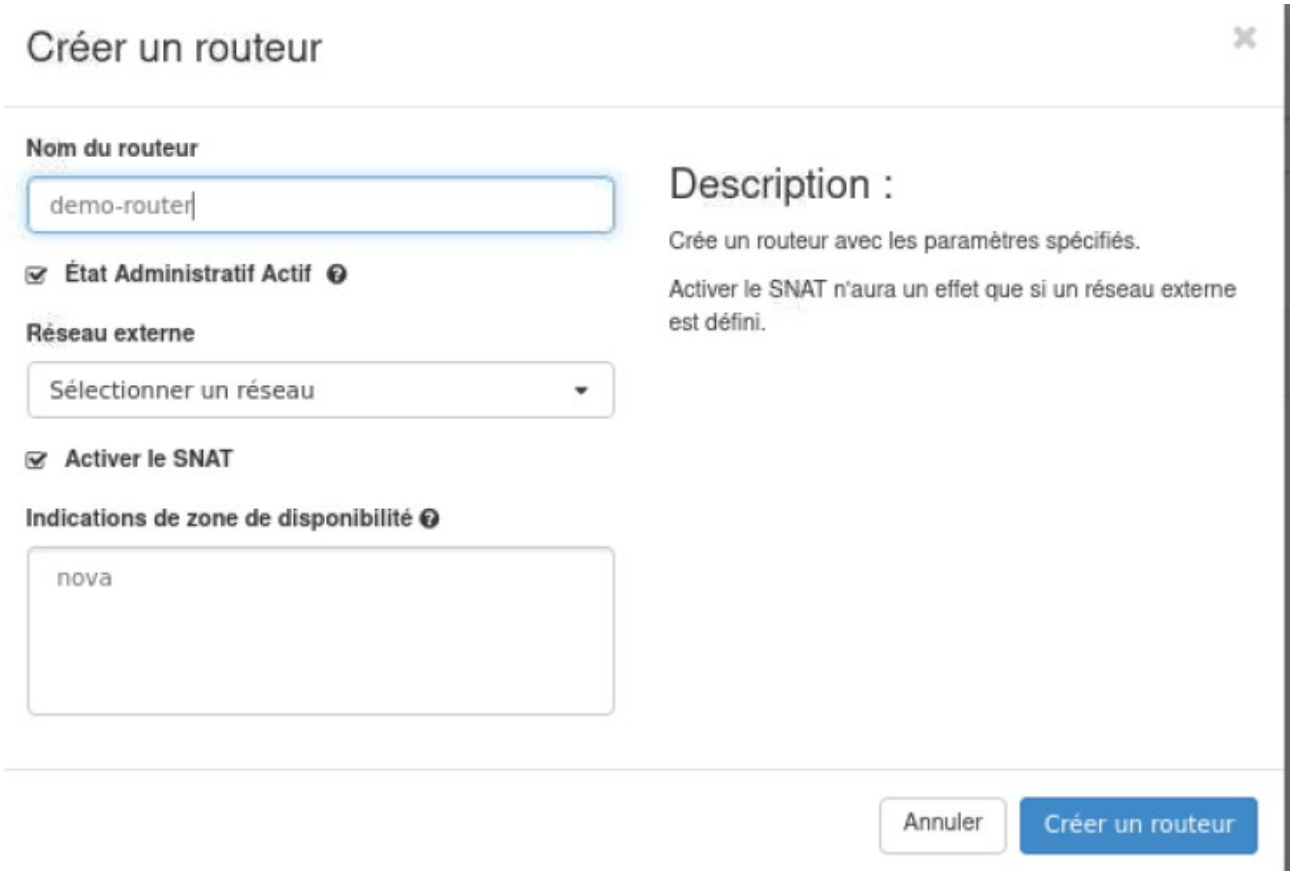
Dans OpenStack, un **réseau** fait référence à l'infrastructure réseau virtualisée qui connecte les instances et autres ressources au sein de l'environnement cloud. Il définit comment les machines virtuelles (VM) communiquent entre elles, avec les réseaux externes et l'internet. OpenStack utilise **Neutron**, le service de mise en réseau, pour gérer les réseaux, les routeurs, les sous-réseaux et les groupes de sécurité, offrant ainsi une flexibilité dans la configuration des topologies réseau pour votre environnement cloud.

Nous allons commencer par créer un routeur.

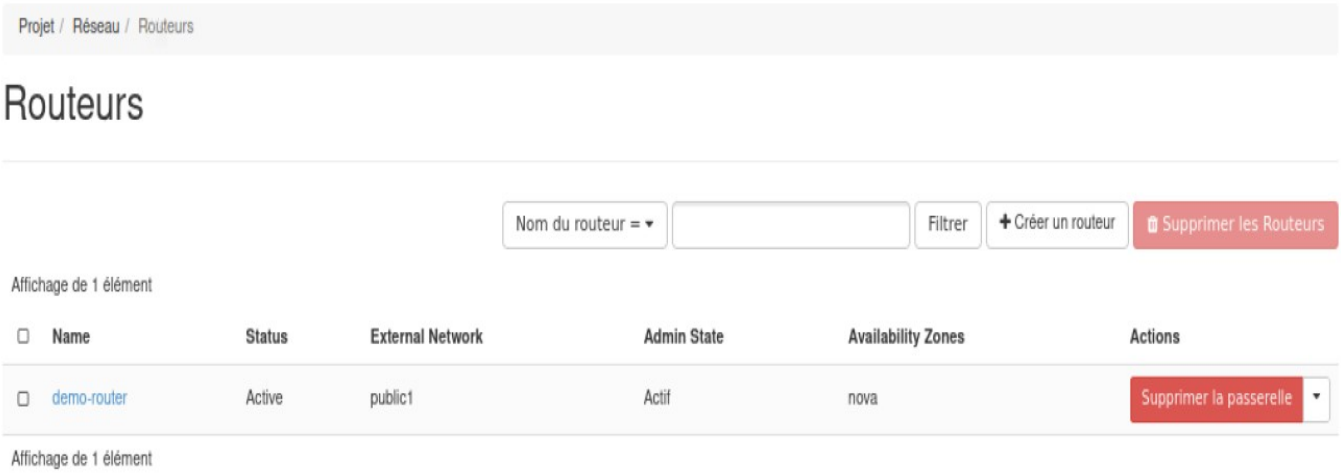
Router:

Dans OpenStack, un **routeur** est utilisé pour connecter différents réseaux, à la fois privés et externes, et acheminer le trafic entre eux.

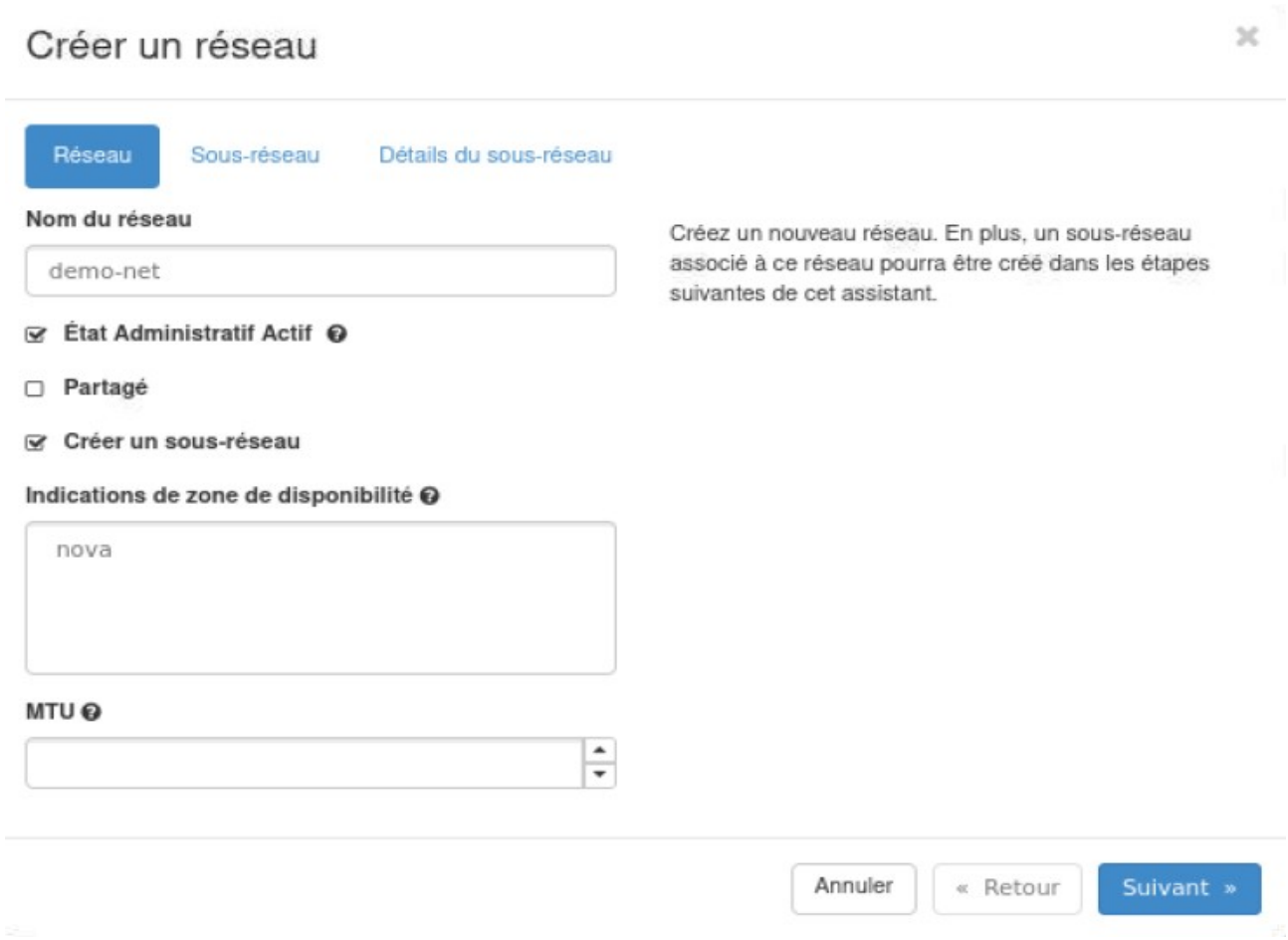
Créons-en un :

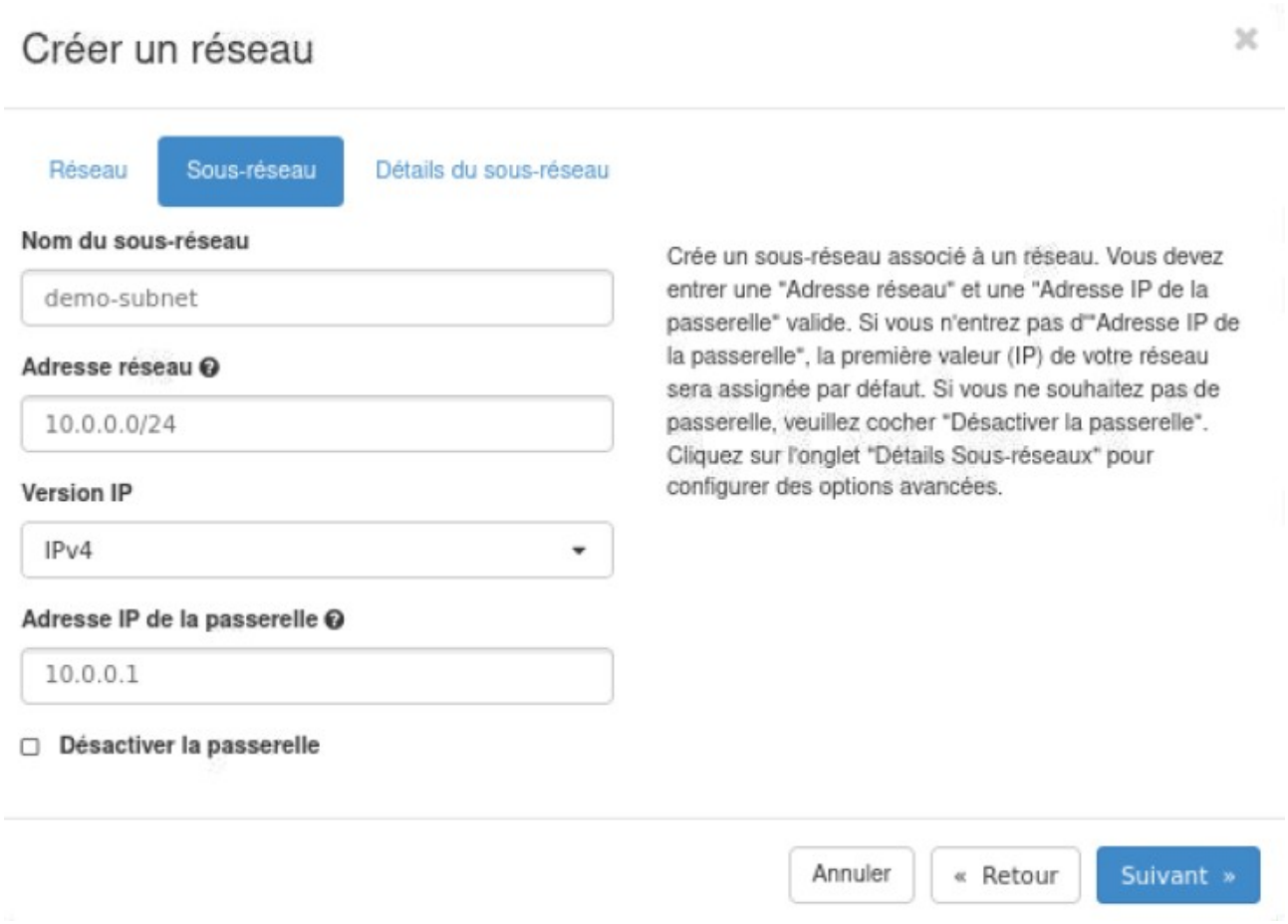


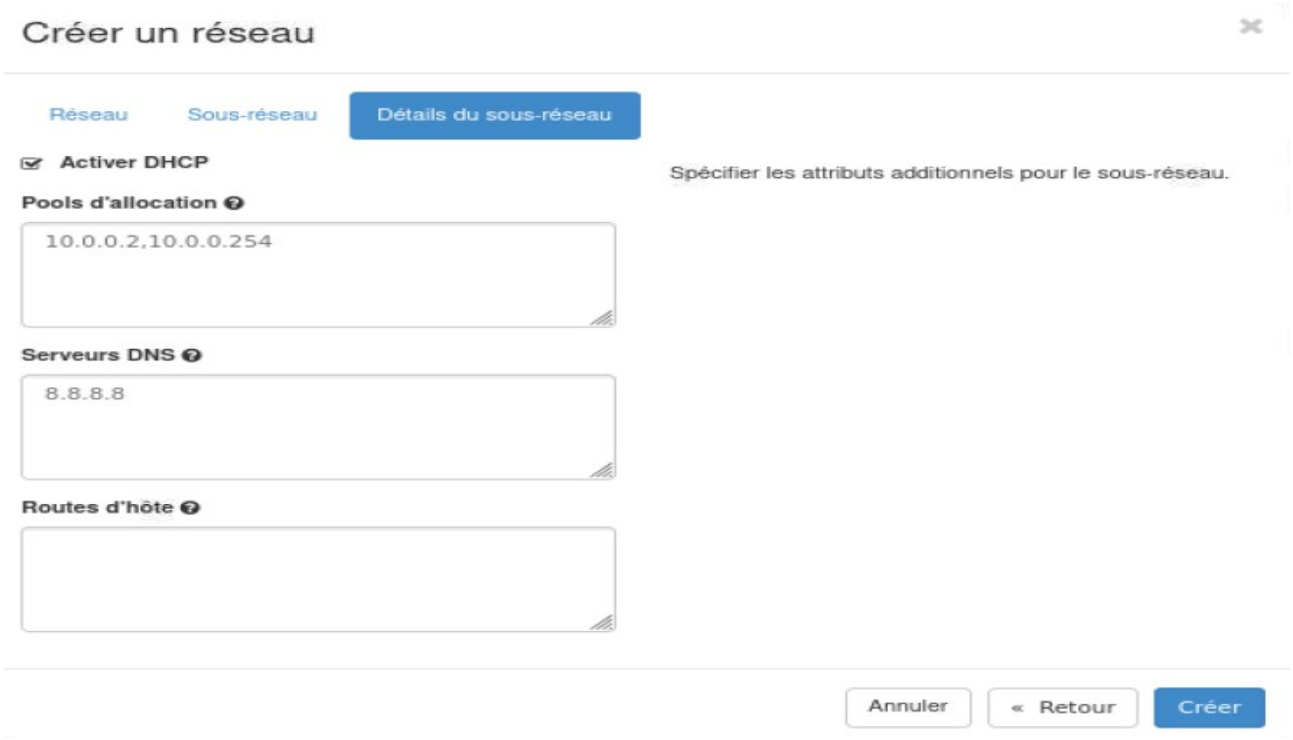
allons listons les routeurs:



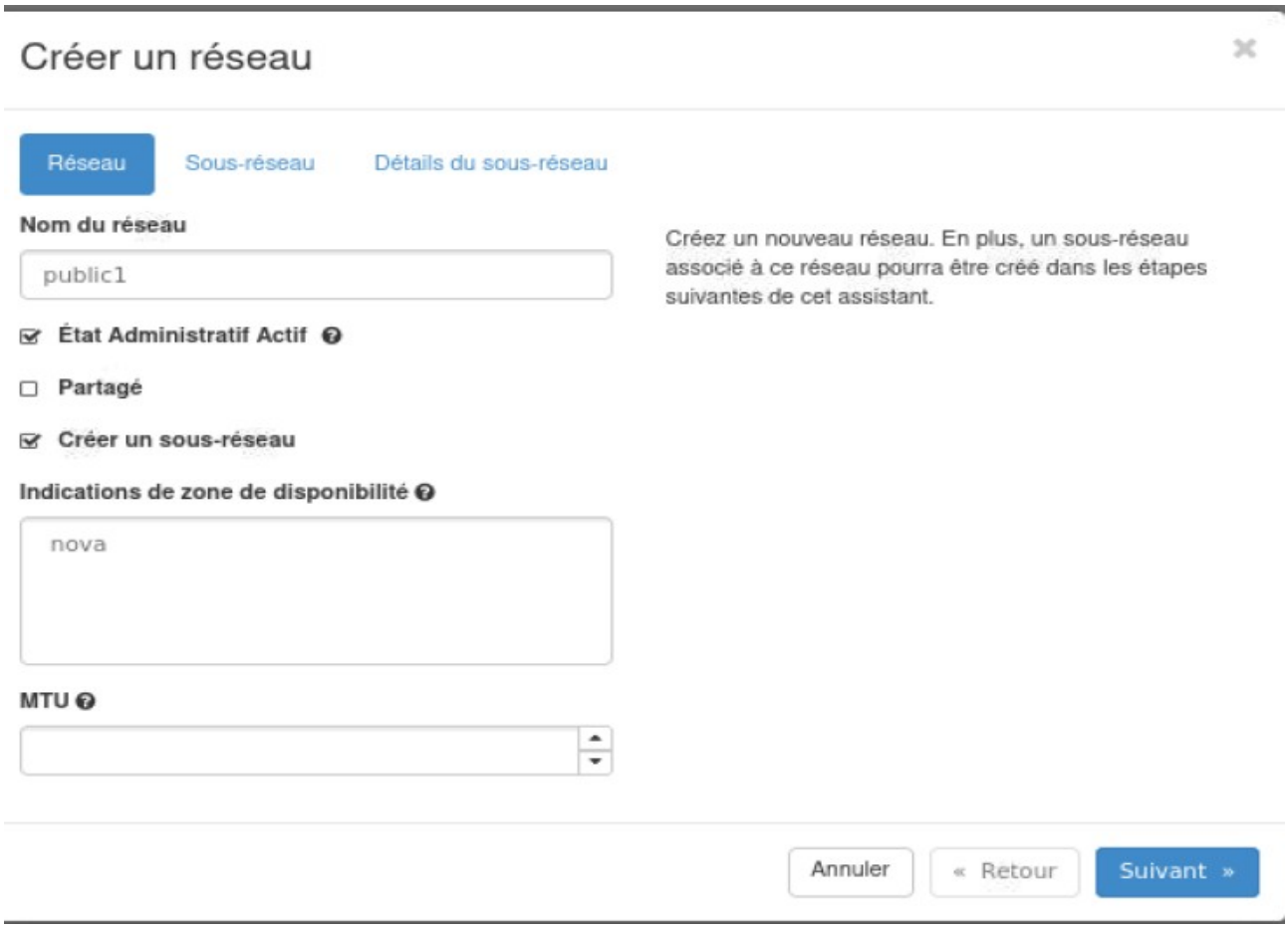
Maintenant, on'a besoin de creer un reseau pour les vm instances, allons lui creer:

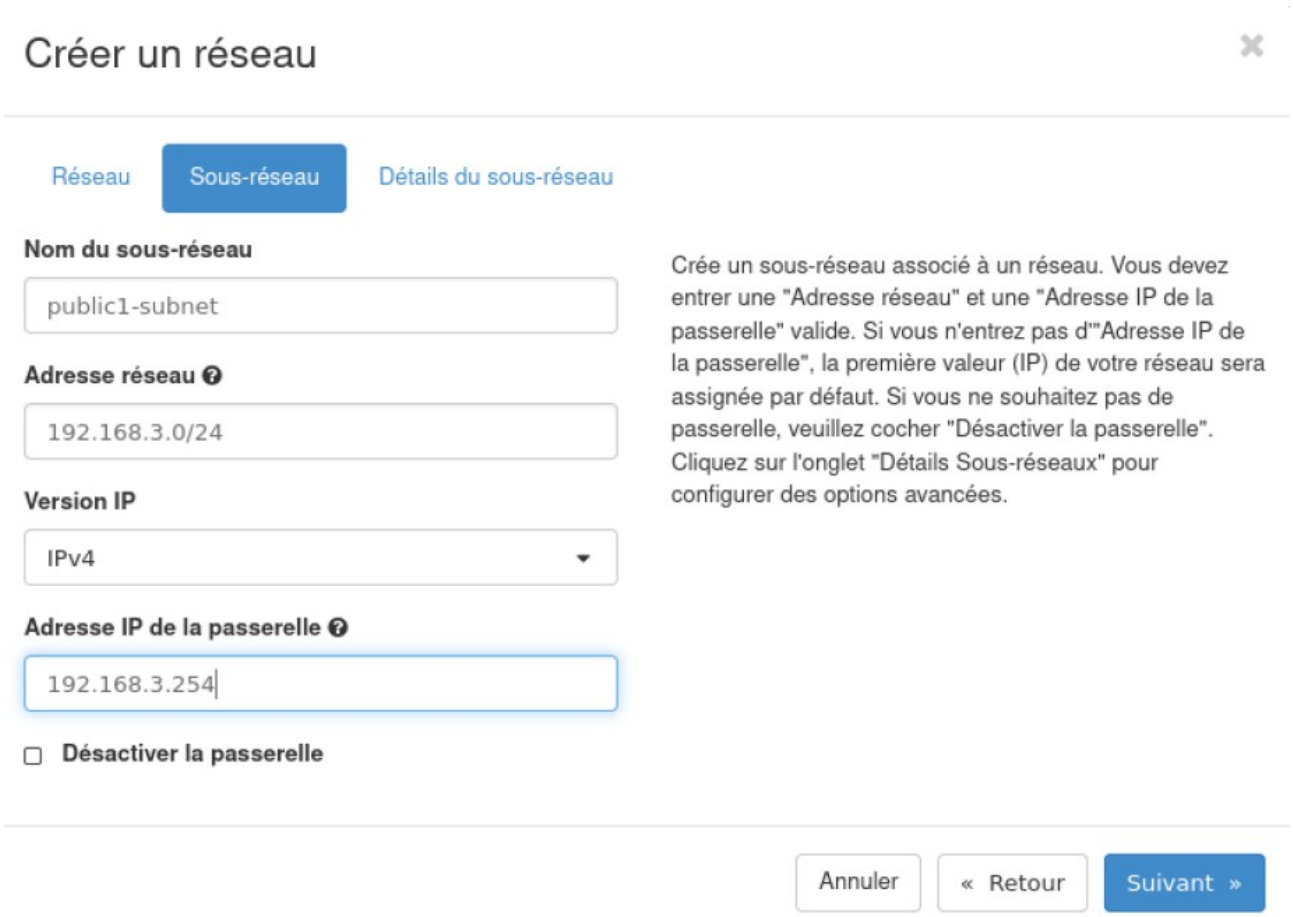


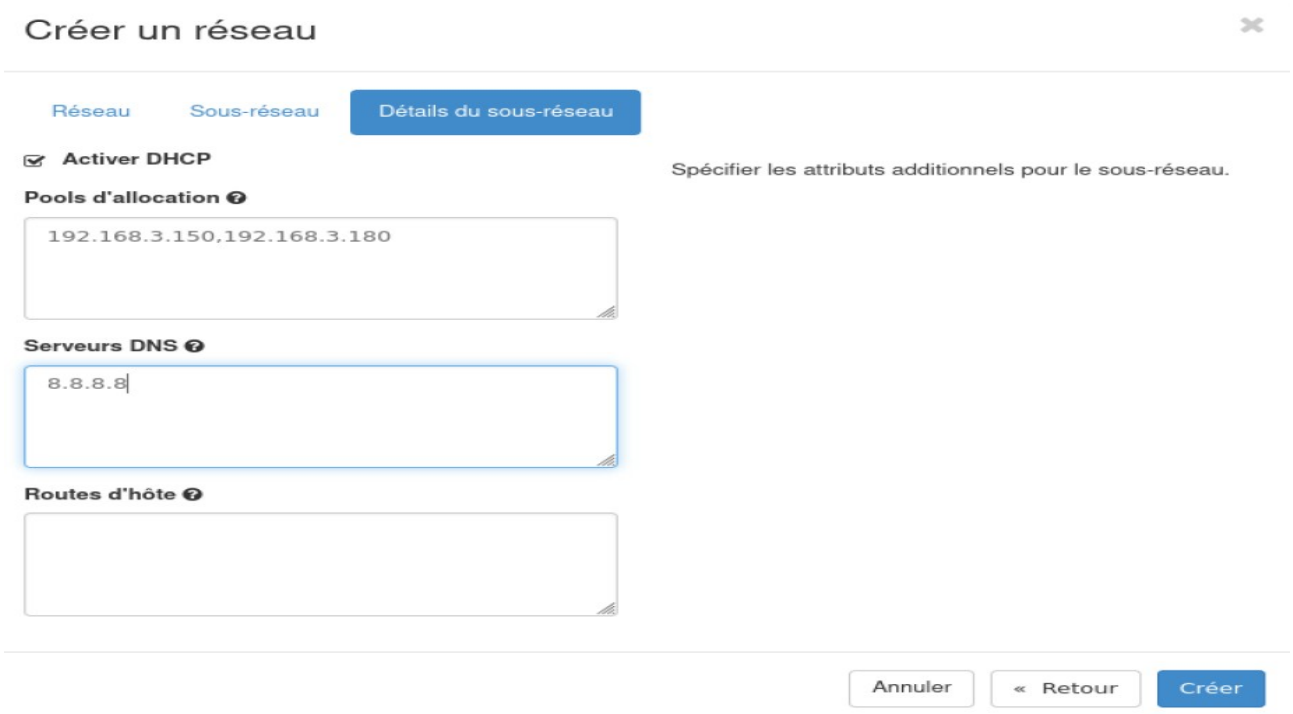

Créer un réseau
Réseau
Sous-réseau
Détails du sous-réseau
Nom du sous-réseau
demo-subnet
Adresse réseau
10.0.0.0/24
Version IP
IPv4
Adresse IP de la passerelle
10.0.0.1
Désactiver la passerelle
Crée un sous-réseau associé à un réseau. Vous devez entrer une "Adresse réseau" et une "Adresse IP de la passerelle" valide. Si vous n'entrez pas d'"Adresse IP de la passerelle", la première valeur (IP) de votre réseau sera assignée par défaut. Si vous ne souhaitez pas de passerelle, veuillez cocher "Désactiver la passerelle". Cliquez sur l'onglet "Détails Sous-réseaux" pour configurer des options avancées.
Annuler
« Retour
Suivant »


Créer un réseau
Réseau
Sous-réseau
Détails du sous-réseau
Activer DHCP
Pools d'allocation
10.0.0.2,10.0.0.254
Serveurs DNS
8.8.8.8
Routes d'hôte
Spécifier les attributs additionnels pour le sous-réseau.
Annuler
« Retour
Créer

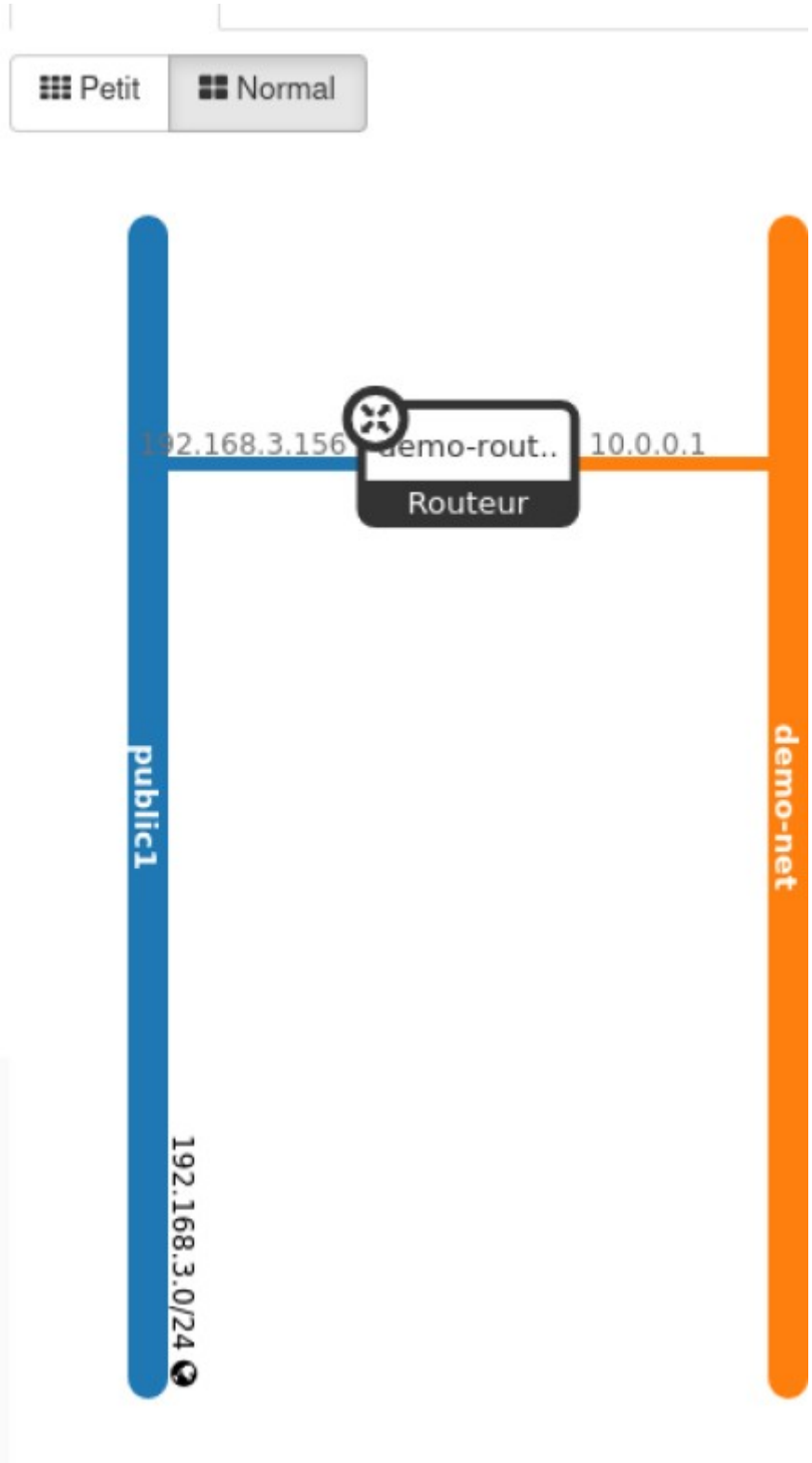
et maintenant on va creer un pour les reseau externes, qui sera public:




Créer un réseau
Réseau
Sous-réseau
Détails du sous-réseau
Nom du sous-réseau
public1-subnet
Adresse réseau
192.168.3.0/24
Version IP
IPv4
Adresse IP de la passerelle
192.168.3.254
Désactiver la passerelle
Crée un sous-réseau associé à un réseau. Vous devez entrer une "Adresse réseau" et une "Adresse IP de la passerelle" valide. Si vous n'entrez pas d'"Adresse IP de la passerelle", la première valeur (IP) de votre réseau sera assignée par défaut. Si vous ne souhaitez pas de passerelle, veuillez cocher "Désactiver la passerelle". Cliquez sur l'onglet "Détails Sous-réseaux" pour configurer des options avancées.
Annuler
« Retour
Suivant »


Créer un réseau
Réseau
Sous-réseau
Détails du sous-réseau
Activer DHCP
Pools d'allocation
192.168.3.150,192.168.3.180
Serveurs DNS
8.8.8.8
Routes d'hôte
Spécifier les attributs additionnels pour le sous-réseau.
Annuler
« Retour
Créer

Et voila le topologie que on a apres la configuration, on’a un reseau qui interconnectes les vm instances et un pour les reseau externes.



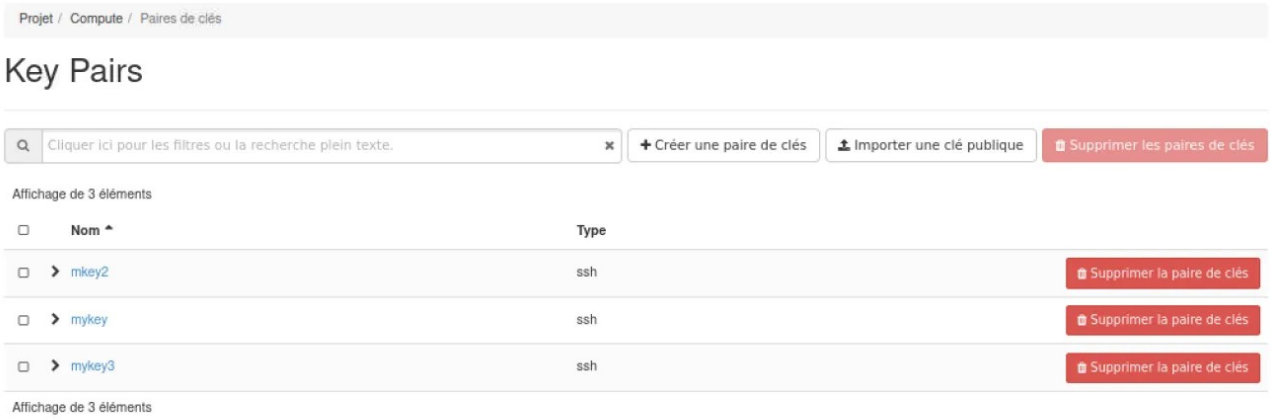
4. Key Pair:

Une **paire de clés** dans **OpenStack** est une paire de clés cryptographiques composée d'une clé publique et d'une clé privée. Elle est utilisée pour un accès SSH sécurisé aux instances OpenStack. La clé publique est téléchargée sur OpenStack et injectée dans les instances lorsqu'elles sont lancées, tandis que la clé privée reste avec l'utilisateur et est nécessaire pour établir une connexion SSH sécurisée. Cela élimine le besoin d'authentification par mot de passe, améliorant ainsi la sécurité.

Pour générer une paire de clés, nous procédons ainsi :

```
ssh-keygen -t ecdsa -N '' -f ~/.ssh/id_ecdsa
openstack keypair create --public-key ~/.ssh/id_ecdsa.pub mykey
```

```
(kolla-venv) grd@opk:~$ ssh-keygen -t ecdsa -N '' -f ~/.ssh/id_ecdsa3
Generating public/private ecdsa key pair.
Your identification has been saved in /home/grd/.ssh/id_ecdsa3
Your public key has been saved in /home/grd/.ssh/id_ecdsa3.pub
The key fingerprint is:
SHA256:KFkcqSefzpLnn+bSyHsmRQTH0O1E5wKz2bcJlRcm5Hc grd@opk
The key's randomart image is:
+---[ECDSA 256]---+
|      +*+o..=.o. |
|     ..o+*o=.o.  |
|     .o.oo+ +.. E|
|    oo... .+ + . |
|    o+.oS   o    |
|     .o .        |
|     = +         |
|    o O =.       |
|     +oX+        |
+----[SHA256]-----+
(kolla-venv) grd@opk:~$ openstack keypair create --public-key ~/.ssh/id_ecdsa3.pub mykey3
+-------------+-------------------------------------------------+
| Field       | Value                                           |
+-------------+-------------------------------------------------+
| created_at  | None                                            |
| fingerprint | 48:ca:5a:a8:ea:1d:52:d9:da:76:de:06:ff:58:2c:e0 |
| id          | mykey3                                          |
| is_deleted  | None                                            |
| name        | mykey3                                          |
| type        | ssh                                             |
| user_id     | dbcfdf72ca4d44d6883a4c806d51bc0d                |
+-------------+-------------------------------------------------+
```



5. Security Groups

Un **groupe de sécurité** dans **OpenStack** est un ensemble de règles de pare-feu qui définissent l'accès réseau aux instances. Il agit comme un pare-feu virtuel qui contrôle le trafic entrant et sortant des instances, offrant ainsi une sécurité au niveau du réseau. Lors de la création ou du lancement d'une instance OpenStack, vous pouvez lui associer un ou plusieurs groupes de sécurité. Ces règles peuvent spécifier le trafic autorisé ou refusé en fonction de paramètres tels que l'adresse IP, le port et le protocole. Les règles courantes à configurer pour une instance incluent généralement l'autorisation de SSH (port 22) pour l'accès à distance, HTTP (port 80) et HTTPS (port 443) pour l'accès web, et éventuellement ICMP (ping) pour les tests de connectivité de base. Les groupes de sécurité peuvent être personnalisés en fonction des exigences de sécurité du déploiement et peuvent être modifiés même après le lancement de l'instance.

Pour lister les groupes de sécurité :

```
openstack security group list
```

Pour ajouter une règle (par exemple, autoriser SSH sur le port 22) :

```
openstack security group rule create --protocol tcp --dst-port 22 <security-group-id>
```

Pour notre instance, nous allons configurer ces règles de groupe de sécurité :

```
# icmp
openstack security group rule create --ingress --ethertype IPv${IP_VERSION} \
    --protocol icmp ${ADMIN_SEC_GROUP}
# ssh
openstack security group rule create --ingress --ethertype IPv${IP_VERSION} \
    --protocol tcp --dst-port 22 ${ADMIN_SEC_GROUP}
# http
openstack security group rule create --ingress --ethertype IPv${IP_VERSION} \
    --protocol tcp --dst-port 8000 ${ADMIN_SEC_GROUP}
# http port 8080
openstack security group rule create --ingress --ethertype IPv${IP_VERSION} \
    --protocol tcp --dst-port 8080 ${ADMIN_SEC_GROUP}
```

```
(kolla-venv) grd@opk:~$ openstack security group list
+--------------------------------------+---------+-----------------------+----------------------------------+------+
| ID                                   | Name    | Description           | Project                          | Tags |
+--------------------------------------+---------+-----------------------+----------------------------------+------+
| 32709e82-34e2-4828-a45b-32aa894835c1 | default | Default security group | a66f7dc80b5145d494d0a34beae6cee2 | []   |
| 4d0e76cf-c6cc-4ca5-ab52-63d377ecbed3 | default | Default security group | e7677ebfe40f43e7945848a1c91d3ee3 | []   |
+--------------------------------------+---------+-----------------------+----------------------------------+------+
(kolla-venv) grd@opk:~$ openstack security group rule list
+--------------------------+-----------+----------+----------+-----------+----------+--------------------------+--------------------+--------------------------+
| ID                       | IP Protocol | Ethertype | IP Range | Port Range | Direction | Remote Security Group    | Remote Address Group | Security Group           |
+--------------------------+-----------+----------+----------+-----------+----------+--------------------------+--------------------+--------------------------+
| 12b07dee-d757-47fd-90b7- | None      | IPv4     | 0.0.0.0/0 |           | egress   | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 7765847b8697             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
| 29479430-51a6-4504-bcdf- | None      | IPv4     | 0.0.0.0/0 |           | ingress  | 4d0e76cf-c6cc-4ca5-ab52- | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 67eca50193d0             |           |          |          |           |          | 63d377ecbed3             |                    | 63d377ecbed3             |
| 39abc481-aef5-4809-b02b- | tcp       | IPv4     | 0.0.0.0/0 | 8080:8080 | ingress  | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 5b0d313607ad             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
| 3ed8f505-ee81-46b7-8bd8- | None      | IPv6     | ::/0     |           | ingress  | 32709e82-34e2-4828-a45b- | None               | 32709e82-34e2-4828-a45b- |
| 42e96270dca5             |           |          |          |           |          | 32aa894835c1             |                    | 32aa894835c1             |
| 5bce557b-22c6-4c46-b009- | icmp      | IPv4     | 0.0.0.0/0 |           | ingress  | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 7973567f3ddb             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
| 6d5c4009-e4f1-4c40-905e- | tcp       | IPv4     | 0.0.0.0/0 | 8000:8000 | ingress  | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| f2b8fb6aabad             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
| 77d69d5a-8079-4b71-932c- | None      | IPv4     | 0.0.0.0/0 |           | egress   | None                     | None               | 32709e82-34e2-4828-a45b- |
| 562b3ceac961             |           |          |          |           |          |                          |                    | 32aa894835c1             |
| 997374df-0363-467f-923e- | None      | IPv6     | ::/0     |           | egress   | None                     | None               | 32709e82-34e2-4828-a45b- |
| d16cb04f1498             |           |          |          |           |          |                          |                    | 32aa894835c1             |
| a6014646-6223-42e6-9daa- | None      | IPv6     | ::/0     |           | egress   | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 9fb8c85002e6             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
| c257e471-e489-4c65-b4b4- | None      | IPv4     | 0.0.0.0/0 |           | ingress  | 32709e82-34e2-4828-a45b- | None               | 32709e82-34e2-4828-a45b- |
| 82156f492a63             |           |          |          |           |          | 32aa894835c1             |                    | 32aa894835c1             |
| da8978c0-a79b-47f9-98a0- | None      | IPv6     | ::/0     |           | ingress  | 4d0e76cf-c6cc-4ca5-ab52- | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| 6c1a2834ae7f             |           |          |          |           |          | 63d377ecbed3             |                    | 63d377ecbed3             |
| df9c4b7d-7107-4ab7-96a4- | tcp       | IPv4     | 0.0.0.0/0 | 22:22     | ingress  | None                     | None               | 4d0e76cf-c6cc-4ca5-ab52- |
| c145e0e77795             |           |          |          |           |          |                          |                    | 63d377ecbed3             |
+--------------------------+-----------+----------+----------+-----------+----------+--------------------------+--------------------+--------------------------+
```

Et c'est tout, nous avons terminé toutes les configurations des ressources, maintenant créons une instance à la fois via la CLI OpenStack et le tableau de bord Horizon.

5. Instance Creation

```
(kolla-venv) grd@opk:~$ openstack server create \
    --image jammy-cloud-img \
    --flavor m1.small \
    --key-name mykey \
    --network demo-net \
    ubuntu-demo
+-------------------------------------+-------------------------------------------------------------+
| Field                               | Value                                                       |
+-------------------------------------+-------------------------------------------------------------+
| OS-DCF:diskConfig                   | MANUAL                                                      |
| OS-EXT-AZ:availability_zone         |                                                             |
| OS-EXT-SRV-ATTR:host                | None                                                        |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                                        |
| OS-EXT-SRV-ATTR:instance_name       |                                                             |
| OS-EXT-STS:power_state              | NOSTATE                                                     |
| OS-EXT-STS:task_state               | scheduling                                                  |
| OS-EXT-STS:vm_state                 | building                                                    |
| OS-SRV-USG:launched_at              | None                                                        |
| OS-SRV-USG:terminated_at            | None                                                        |
| accessIPv4                          |                                                             |
| accessIPv6                          |                                                             |
| addresses                           |                                                             |
| adminPass                           | bKMq4Y8ATcx2                                                |
| config_drive                        |                                                             |
| created                             | 2024-12-29T07:34:33Z                                        |
| flavor                              | m1.small (2)                                                |
| hostId                              |                                                             |
| id                                  | b90ac4c3-579d-4b1a-820f-215367915be7                        |
| image                               | jammy-cloud-img (4cb17836-1ccc-4752-9781-6a09f99aff08)      |
| key_name                            | mykey                                                       |
| name                                | ubuntu-demo                                                 |
| os-extended-volumes:volumes_attached | []                                                          |
| progress                            | 0                                                           |
| project_id                          | 3dbcba028a0046c7896e97b3ccc83f8d                            |
| properties                          |                                                             |
| security_groups                     | name='default'                                              |
| status                              | BUILD                                                       |
| updated                             | 2024-12-29T07:34:33Z                                        |
| user_id                             | 184eac219e1a47dfadf7acc19fbcf423                            |
+-------------------------------------+-------------------------------------------------------------+
```

Et tout comme sa nous avons une openstack VM instance en cours d'exécution sur notre OpenStack IaaS, listons les serveurs (instances) :

```
(kolla-venv) grd@opk:~$ openstack server list
+--------------------------------------+-------------+--------+---------------------+----------------+----------+
| ID                                   | Name        | Status | Networks            | Image          | Flavor   |
+--------------------------------------+-------------+--------+---------------------+----------------+----------+
| b90ac4c3-579d-4b1a-820f-215367915be7 | ubuntu-demo | ACTIVE | demo-net=10.0.0.194 | jammy-cloud-img | m1.small |
+--------------------------------------+-------------+--------+---------------------+----------------+----------+
(kolla-venv) grd@opk:~$
```

6. Floating Ips

Une **adresse IP flottante** dans OpenStack est une adresse IP publique qui peut être assignée dynamiquement à une instance, permettant ainsi un accès depuis l'extérieur du réseau privé. Les IP flottantes sont généralement utilisées pour les instances qui doivent être accessibles depuis Internet ou des réseaux externes, tels que les serveurs web ou les applications. Ces IP ne sont pas attribuées à une instance spécifique par défaut ; elles peuvent être associées à n'importe

quelle instance dans l'environnement OpenStack. Cette flexibilité permet aux instances de conserver une adresse IP publique cohérente, même si l'instance est arrêtée ou migrée. Une fois qu'une IP flottante est allouée, elle peut être associée à une instance, et le trafic dirigé vers cette IP est transféré à l'IP privée de l'instance, la rendant accessible publiquement.

Maintenant, créons une IP flottante pour l'instance que nous avons créée :

```
(kolla-venv) grd@opk:~$ openstack floating ip create public1
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| created_at          | 2024-12-29T07:36:27Z                 |
| description         |                                      |
| dns_domain          | None                                 |
| dns_name            | None                                 |
| fixed_ip_address    | None                                 |
| floating_ip_address | 192.168.3.169                        |
| floating_network_id | 0a907f27-2d51-4de1-b980-7f39fe2284d9 |
| id                  | ca6a8323-aed5-40e9-992c-2a1836f7b222 |
| name                | 192.168.3.169                        |
| port_details        | None                                 |
| port_id             | None                                 |
| project_id          | 3dbcba028a0046c7896e97b3ccc83f8d     |
| qos_policy_id       | None                                 |
| revision_number     | 0                                    |
| router_id           | None                                 |
| status              | DOWN                                 |
| subnet_id           | None                                 |
| tags                | []                                   |
| updated_at          | 2024-12-29T07:36:27Z                 |
+---------------------+--------------------------------------+
(kolla-venv) grd@opk:~$
```

Elle nous a donné l'IP publique 192.168.1.174, maintenant associons l'IP flottante à notre instance VM.

```
(kolla-venv) grd@opk:~$ openstack server add floating ip ubuntu-demo 192.168.3.169
(kolla-venv) grd@opk:~$
```

Et maintenant, notre instance est associée à l'IP flottante :

```
(kolla-venv) grd@opk:~$ openstack server list
+--------------------------------------+-------------+--------+----------------------------------+----------------+----------+
| ID                                   | Name        | Status | Networks                         | Image          | Flavor   |
+--------------------------------------+-------------+--------+----------------------------------+----------------+----------+
| b90ac4c3-579d-4b1a-820f-215367915be7 | ubuntu-demo | ACTIVE | demo-net=10.0.0.194, 192.168.3.169 | jammy-cloud-img | m1.small |
+--------------------------------------+-------------+--------+----------------------------------+----------------+----------+
```

7. Activer Internet dans l'instance

Le Problème :

Par défaut, les instances OpenStack ne sont pas connectées à Internet. Cela se produit car le trafic réseau pour la communication externe (en dehors de l'environnement OpenStack) doit être acheminé via une interface de pont, généralement br-ex (pont externe), qui se connecte à votre réseau externe. Sans une configuration adéquate, le trafic ne pourra pas atteindre Internet, car il pourrait essayer de passer par l'interface réseau privée de la machine virtuelle ou la passerelle par défaut de l'hôte, ce qui pourrait ne pas le router correctement.

Pour nous, nos configurations réseau étaient quelque chose comme ceci :

```
(kolla-venv) grd@opk:~$ sudo cat /etc/netplan/50-cloud-init.yaml
[sudo] password for grd:
network:
    ethernets:
        ens19:
            dhcp4: false
        ens20:
            addresses:
            - 192.168.3.5/24
            dhcp4: false
        ens18:
            addresses:
            - 192.168.1.101/24
            nameservers:
                addresses:
                - 8.8.8.8
                - 8.8.4.4
                - 1.1.1.1
                search: []
            routes:
            # Default route via 192.168.1.1
            - to: default
              via: 192.168.1.1
            # Route traffic for the 192.168.3.0/24 subnet via 192.168.3.254
            - to: 192.168.3.0/24
              via: 192.168.3.254
    version: 2
```

Les interfaces `ens19` et `ens20` étaient utilisées pour les connexions réseau internes et externes d'OpenStack, tandis que l'interface `ens18` fournissait l'accès à Internet à notre VM hôte.
Pour fournir l'accès à Internet à nos instances OpenStack VM, nous suivrons les étapes suivantes :

Les Étapes :

7.1 Activer l'interface de pont externe (`br-ex`) : Le pont `br-ex` est responsable de l'acheminement du trafic externe entre vos instances OpenStack et le monde extérieur. Vous devez configurer ce pont pour qu'il soit activé.

```
sudo ip addr add 192.168.3.254/24 dev br-ex
sudo ip link set br-ex up
```

7.2 Activer le transfert IP : Le transfert IP permet à votre hôte de router les paquets depuis le réseau interne OpenStack vers le réseau externe (Internet).

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

7.3 Configurer le NAT (Network Address Translation) : Pour permettre à vos instances VM d'accéder à Internet, vous devez configurer une règle NAT a l'interface connecter a l'internet(ens18 dans ce cas).

```
sudo iptables -t nat -A POSTROUTING -o ens18 -j MASQUERADE
```

7.4 Autoriser le transfert entre `br-ex` et votre interface Internet : Vous devez autoriser le trafic de `br-ex` à être transféré vers l'interface connectée à Internet (`ens18`).

```
sudo iptables -A FORWARD -i br-ex -o ens18 -j ACCEPT
sudo iptables -A FORWARD -i ens18 -o br-ex -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Explication :

- **br-ex :** Il s'agit d'un pont Open vSwitch (OVS) dans OpenStack utilisé pour acheminer le trafic externe vers/depuis vos instances VM.
- **ens18 :** Il s'agit de l'interface réseau connectée à Internet.
- **iptables :** Cet outil est utilisé pour configurer le pare-feu et définir les règles NAT pour un routage correct.
- **MASQUERADE :** Cette règle NAT masque les adresses IP internes de vos VMs OpenStack, les faisant apparaître comme si elles provenaient de l'IP externe de l'hôte lorsqu'elles accèdent à Internet.

Maintenant, voyons si nous pouvons pinguer et nous connecter en SSH à notre instance VM :

```
(kolla-venv) grd@opk:~$ openstack server list
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
| ID                                   | Name        | Status | Networks                        | Image          | Flavor   |
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
| b90ac4c3-579d-4b1a-820f-215367915be7 | ubuntu-demo | ACTIVE | demo-net=10.0.0.194, 192.168.3.169 | jammy-cloud-img | m1.small |
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
(kolla-venv) grd@opk:~$ ping 192.168.3.169
PING 192.168.3.169 (192.168.3.169) 56(84) bytes of data.
64 bytes from 192.168.3.169: icmp_seq=1 ttl=63 time=5.01 ms
64 bytes from 192.168.3.169: icmp_seq=2 ttl=63 time=3.43 ms
64 bytes from 192.168.3.169: icmp_seq=3 ttl=63 time=2.36 ms
64 bytes from 192.168.3.169: icmp_seq=4 ttl=63 time=2.36 ms
64 bytes from 192.168.3.169: icmp_seq=5 ttl=63 time=4.42 ms
^C
--- 192.168.3.169 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 2.357/3.514/5.008/1.070 ms
(kolla-venv) grd@opk:~$
```

Et oui, nous pouvons pinguer l'instance VM.
Maintenant, voyons si nous pouvons nous connecter en SSH :

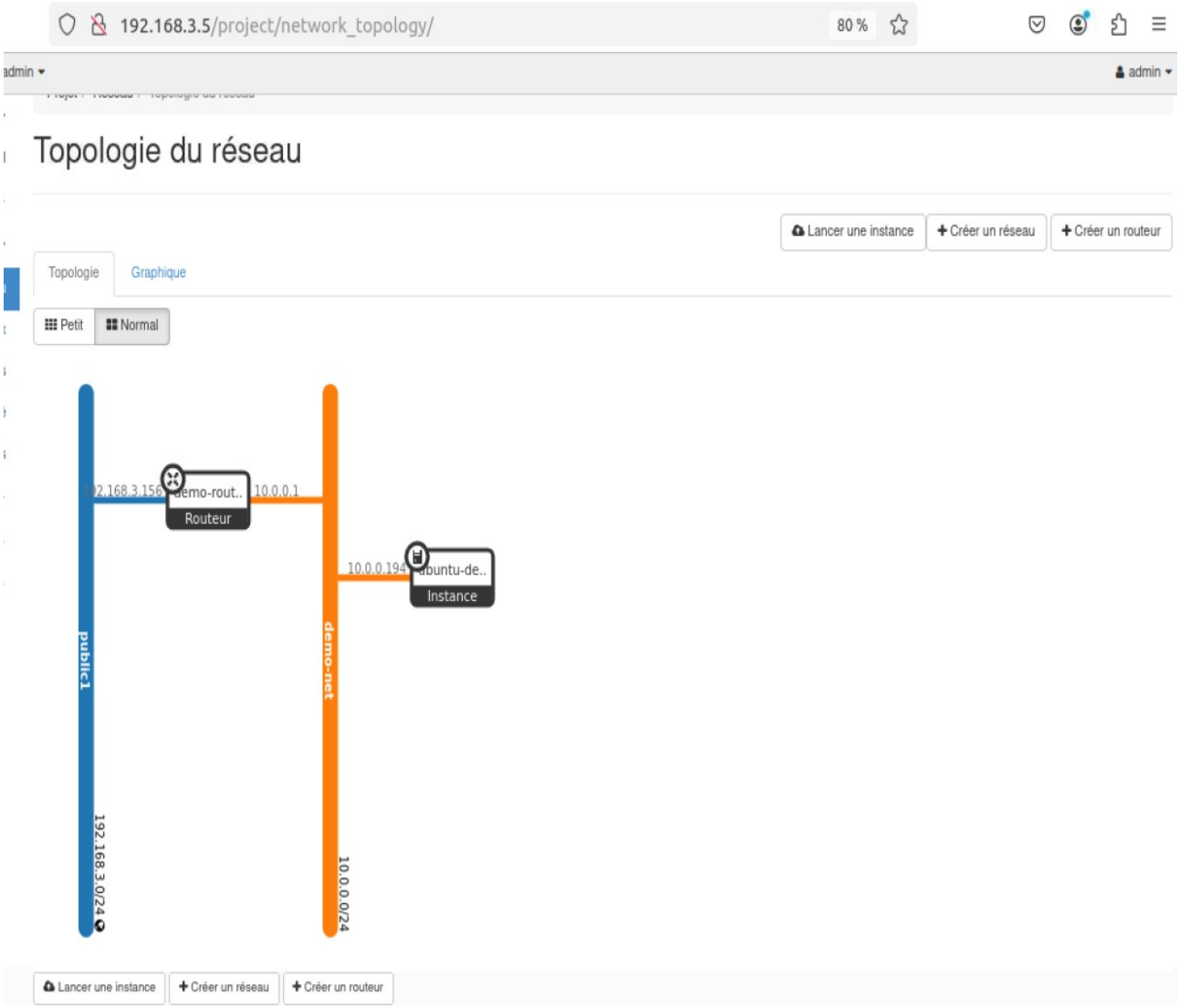
```
(kolla-venv) grd@opk:~$ openstack server list
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
| ID                                   | Name        | Status | Networks                        | Image          | Flavor   |
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
| b90ac4c3-579d-4b1a-820f-215367915be7 | ubuntu-demo | ACTIVE | demo-net=10.0.0.194, 192.168.3.169 | jammy-cloud-img | m1.small |
+--------------------------------------+-------------+--------+---------------------------------+----------------+----------+
(kolla-venv) grd@opk:~$ ssh ubuntu@192.168.3.169
The authenticity of host '192.168.3.169 (192.168.3.169)' can't be established.
ED25519 key fingerprint is SHA256:dK2F/7taYMY5f0n3l1BcUCMid1hTqJYbeBuFQE4P0/0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.3.169' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information disabled due to load higher than 1.0

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ubuntu-demo:~$
```

Et oui, nous pouvons nous connecter en SSH à notre openstack VM instance.

On a aussi accès à Internet dans notre instance OpenStack VM :

```
ubuntu@ubuntu-demo:~$ ping google.com
PING google.com (192.178.24.206) 56(84) bytes of data.
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=1 ttl=113 time=38.7 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=2 ttl=113 time=37.0 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=3 ttl=113 time=36.6 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=4 ttl=113 time=37.0 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=5 ttl=113 time=36.8 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=6 ttl=113 time=36.8 ms
64 bytes from mct04s03-in-f14.1e100.net (192.178.24.206): icmp_seq=7 ttl=113 time=36.7 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6015ms
rtt min/avg/max/mdev = 36.628/37.087/38.680/0.660 ms
ubuntu@ubuntu-demo:~$
```

Maintenant, voyons notre topologie réseau :



IV. Déploiement d'Application

L'instance créée peut être utilisée pour déployer des applications ou d'autres PaaS. Dans notre cas, nous allons installer une application en ligne de commande que j'ai développée précédemment, appelée Zeroward. Zeroward est un programme de chiffrement à connaissance zéro (preuve à connaissance zéro) qui sécurise les fichiers des utilisateurs à toutes les étapes (localement, lors de la transmission et au repos sur le stockage cloud). Il est conçu pour différents fournisseurs de stockage cloud, bien qu'il ne soit actuellement intégré qu'avec Yandex Cloud Storage.

Vous pouvez y jeter un œil ici.

1. Tout d'abord, téléchargez la dernière version de Zeroward :

```
wget https://github.com/Abdiooa/zeroward/releases/download/v1.0.47/zeroward_1.0.47_linux_amd64.deb
```

Pour nous, c'est Ubuntu, donc nous allons télécharger la distribution amd64.deb.

Alors téléchargeons-le :

```
ubuntu@ubuntu-demo:~$ wget https://github.com/Abdiooa/zeroward/releases/download/v1.0.47/zeroward_1.0.47_linux_amd64.deb
--2024-12-29 07:56:47--  https://github.com/Abdiooa/zeroward/releases/download/v1.0.47/zeroward_1.0.47_linux_amd64.deb
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/748131548/842e27f4-2c78-485f-9b91-799c5bdf12f6?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential
=releaseassetproduction%2F20241229%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241229T075648Z&X-Amz-Expires=300&X-Amz-Signature=031f519695a6f6613537fb52fa48b9097a9750f2a9e3725c3bd426
994fdd0ab7&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dzeroward_1.0.47_linux_amd64.deb&response-content-type=application%2Foctet-stream [following
]
--2024-12-29 07:56:48--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/748131548/842e27f4-2c78-485f-9b91-799c5bdf12f6?X-Amz-Algorithm=AWS4-HMAC-SHA256&X
-Amz-Credential=releaseassetproduction%2F20241229%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20241229T075648Z&X-Amz-Expires=300&X-Amz-Signature=031f519695a6f6613537fb52fa48b9097a9750f
2a9e3725c3bd426994fdd0ab7&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dzeroward_1.0.47_linux_amd64.deb&response-content-type=application%2Foctet-st
ream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10529304 (10M) [application/octet-stream]
Saving to: ‘zeroward_1.0.47_linux_amd64.deb’

zeroward_1.0.47_linux_amd64.deb             100%[=======================================================================================>]  10.04M  8.66MB/s    in 1.2s

2024-12-29 07:56:50 (8.66 MB/s) - ‘zeroward_1.0.47_linux_amd64.deb’ saved [10529304/10529304]

ubuntu@ubuntu-demo:~$
```

Vous pouvez suivre les étapes d'installation des autres plateformes ici.

2. Ensuite, installez le paquet téléchargé comme décrit dans le README.md :

```
ubuntu@ubuntu-demo:~$ sudo dpkg -i zeroward_1.0.47_linux_amd64.deb
Selecting previously unselected package zeroward.
(Reading database ... 64394 files and directories currently installed.)
Preparing to unpack zeroward_1.0.47_linux_amd64.deb ...
Unpacking zeroward (1.0.47) ...
Setting up zeroward (1.0.47) ...
ubuntu@ubuntu-demo:~$
```

Et l'application est installée sur notre instance VM OpenStack IAAS.

```
ubuntu@ubuntu-demo:~$ zeroward
Error creating CLSD folder: mkdir /home/ubuntu/.config/zeroward: no such file or directoryError writing config file: open /home/ubuntu/.config/zeroward/config.json: no such file or di
rectoryzeroward is a zero-knowledge-ecryption command-line application that secures client data at all stage(locally,
               during transmission to the cloud, and at rest). It provides a robust solution
             for encrypting client data locally before uploading it to a cloud storage server. The application
             implements secure transmission protocols, ensuring data remains encrypted during transfer.
             Once stored, it adheres to security policies set by the cloud service provider, including additional
             encryption layers, access management, and continuous monitoring. zeroward empowers users to actively
             participate in securing their data, offering full control and confidence in maintaining the
             confidentiality and integrity of their information.

Usage:
  zeroward [command]

Available Commands:
  buckets     List All Buckets(Folders) stored on the Cloud Storage
  completion  Generate the autocompletion script for the specified shell
  decrypt     Command to decrypt user encrypted file
  download    Download Command to download Files from the cloud.
  encrypt     Encrypt a file Locally
  help        Help about any command
  objects     List All Objects(Files/Images/Docs) stored on the Cloud Storage
  remove      Remove Command to delete a file from cloud storage
  upload      Upload Command to upload Files on a cloud

Flags:
  -i, --accessKeyID string      Access Key Id as your Login Key
  -b, --bucketName string       Bucket name out of all your existing buckets!
      --config string           config file (default is $HOME/.cobra.yaml)
  -f, --filePath string         Path of the file that you want to encrypt
  -h, --help                    help for zeroward
  -o, --objectkey string        objectkey refers to the unique identifier or name of the object(file) with a bucket, it is the path or where in the bucket the file should be stored.
  -p, --passphrase string       Passphrase for encryption required for the first encryption
  -s, --secretAccessKey string  Secret Access Key as your Password

Use "zeroward [command] --help" for more information about a command.
ubuntu@ubuntu-demo:~$
```