**Auteurs :**
ALDAKAR Omar : omar.aldakar@imt-atlantique.net
BELGHITH Hamouda : hamouda.belghith@imt-atlantique.net
BORDES Aurélia : aurelia.bordes@imt-atlantique.netles
HSAIRI Khouloud : khouloud.hsairi@imt-atlantique.net
ODJE Olloe Ernest-Abel : olloe-ernest-abel.odje@imt-atlantique.net

**Tuteur méthodologique** : BOUABDALLAH Ahmed
**Expert technique** : LE NARZUL Jean-Pierre

**Destinataires :**
Tuteur entreprise : PHAN Cao-Thanh
Collaborateur du tuteur entreprise : MORIN Cédric
Chef du projet pour l'entreprise : BEAUCHAMP Frédéric

# PROJET S3 – Document technique

## K8S AS A SERVICE – PARTENAIRE B<>COM

Version 2.0

13 décembre 2020

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# SOMMAIRE

# Table des illustrations

# Table des acronymes

ETSI : European Telecommunications Standards Institute
K8s : Kubernetes
KNF : Kubernetes-based Network Function
MANO : MANagement and Orchestration
NFV : Network Function Virtualization
NFVI : Network Function Virtualization Infrastructure
NFVO : Network Function Virtualization Orchestrator
OS : Operating System
OSS/BSS : Operation Support Subsystem/Business Support Subsystem
VIM : Virtual Infrastructure Management
VM : Virtual Machine
VNF : Virtual Network Function
VNFD : Virtual Network Function Descriptor
VNFM : Virtualized Network Function Management

# I. Résumé

Le but de ce document est de présenter l'installation technique réalisée au cours du projet « K8s as a service » ainsi que l'architecture mise en place. Ce document doit pouvoir servir de guide à qui voudrait réaliser la même installation.

# II. Abstract

The goal of this document is to present the technical realization of the project "K8s as a service" and to explain the architecture that has been set up. This document should be able to serve as a guide for someone who would like to realize the same installation.

# III. Présentation générale

## III A. Présentation du Projet entreprise

Le projet « Commande entreprise » est une UE (Unité d'Enseignement) faisant partie du tronc commun de 2ème année à IMT Atlantique. L'objectif de cette UE est de répondre à une problématique réelle d'entreprise formulée selon un cahier des charges. Ce projet dure un trimestre ; il débute en septembre et se finit en décembre. Il s'effectue en équipe de 5 à 6 étudiants et est encadré par un tuteur méthodologique, un référent scientifique et au moins un membre de l'entreprise partenaire qui représente le client.

L'entreprise client est, dans le cas de ce document, b<>com.

## III B. Présentation de b<>com

B<>com est un institut de recherche technologique français basé à Cesson-Sévigné créé en 2012 dont l'objectif est d'innover dans les technologies du numérique. L'entreprise s'intéresse principalement aux domaines de l'intelligence artificielle, de la vidéo et l'audio immersif, de la protection de contenus, des réseaux 5G, de l'internet des objets et des technologies cognitives.

Le cadre du projet mené dans ce document s'inscrit notamment dans le domaine des réseaux 5G.

# IV. Contexte et objectifs du projet

## IV A. Contexte du projet

Ce projet s'inscrit dans le contexte suivant pour b<>com :

- **Se préparer à l'arrivée de la 5G**
    - L'autorité de régulation des télécoms (ARCEP) est censée autoriser l'utilisation des fréquences 5G vers fin 2020. La 5G faisant partie des domaines d'expertise de b<>com, il est essentiel que l'entreprise soit prête pour son arrivée.

- **Innover dans le domaine du Cloud Computing**
    - Mise en œuvre d'une nouvelle architecture de virtualisation légère.

- **Optimiser le rapport coût-performance**
    - La virtualisation légère offre un meilleur rapport coût-performance que la virtualisation lourde.

## IV B. Objectifs scientifiques

### IV B 1) La virtualisation

#### *IV B 1) a. Les VNF*

Une fonction réseau (NF pour Network Functions) est une brique fonctionnelle d'une architecture réseau qui effectue une tâche (=fonction) particulière. Traditionnellement, une fonction réseau est associée à un équipement physique.

Quelques exemples de fonctions réseau :

- Un firewall
- Un routeur
- Un switch

Une VNF ou Virtual Network Function est une fonction de réseau virtualisée. Il s'agit par exemple d'un routeur virtuel ou d'un firewall virtuel. Les VNF permettent de réduire la dépendance d'une fonction réseau à un matériel physique en particulier. Par exemple, le routeur virtuel ne nécessite plus l'équipement matériel spécifique du routeur ; il a juste besoin d'un serveur physique. Ce dernier peut d'ailleurs faire tourner différentes fonctions réseau qui auraient auparavant nécessitées chacune du matériel spécialisé.

#### *IV B 1) b. L'approche NFV des réseaux [1]*

La NFV (Network Function Virtualization) est la virtualisation des fonctions réseau. C'est une approche réseau qui tend à substituer les équipements physiques et spécifiques

par des équipements logiciels sur des serveurs normalisés. L'approche NFV se base donc sur des VNF.

L'approche NFV se base sur 3 concepts :

- La softwarisation
- La virtualisation
- L'orchestration et l'automatisation

**La softwarisation**

La softwarisation caractérise la prépondérance croissante du software dans les réseaux. Avant, les équipements réseaux ont toujours été indissociables d'un environnement matériel spécifique. Quant à elle, l'architecture NFV a pour objectif de déployer des composants logiciels indépendants de l'infrastructure physique.

**La virtualisation**

La virtualisation consiste en l'émulation d'équipement matériel. C'est le concept à la base de la technologie cloud.

**L'orchestration et l'automatisation**

L'automatisation désigne l'utilisation de technologies pour « automatiser des tâches », c'est-à-dire faire en sorte qu'elles s'effectuent avec le moins d'intervention humaine possible.

L'orchestration désigne l'utilisation de technologies pour automatiser un processus de tâches elles-mêmes automatisées.

La principale différence entre les 2 termes est que l'automatisation concerne souvent une unique tâche tandis que l'orchestration concerne l'ensemble d'un processus.

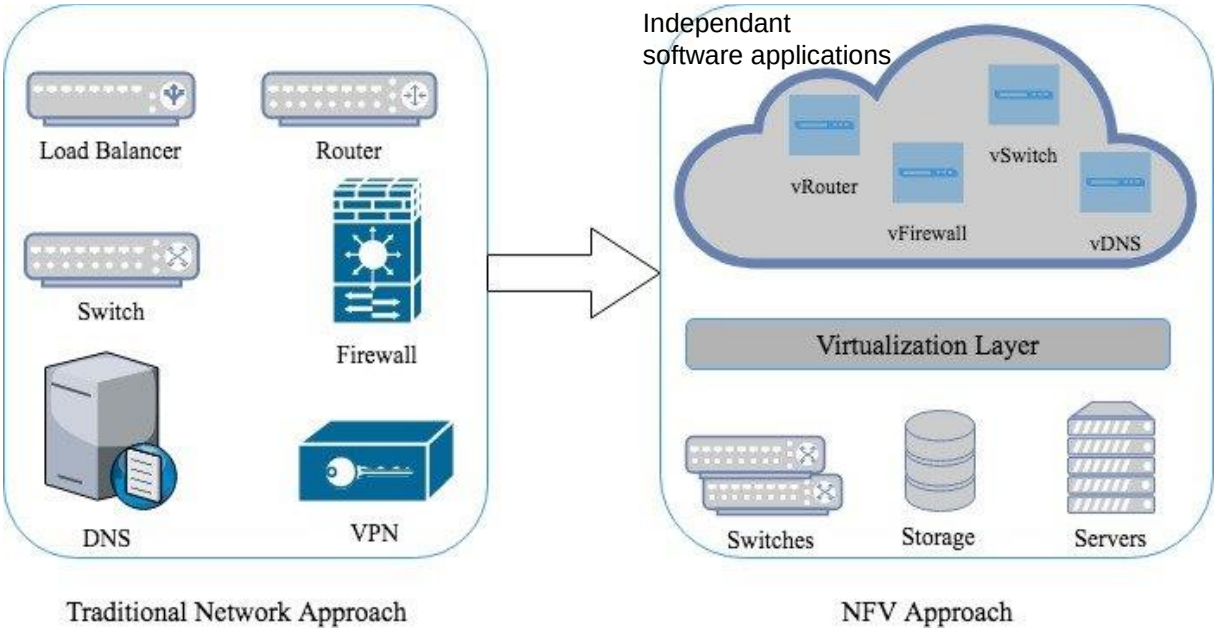


*Figure 1 Différence entre l'approche réseau traditionnelle et l'approche NFV*[a]

[a]https://www.researchgate.net/profile/Ahmed_Alwakeel/publication/334699708/figure/fig1/AS:791565404475392@1565735307473/Difference-between-traditional-network-approach-and-NFV-approach.jpg

Comme l'illustre la Figure 1, l'approche NFV permet de passer de beaucoup de composants informatiques physiques et spécifiques à des composants standardisés moins nombreux qui font le même travail, voire plus, en virtualisant les fonctions réseau de manière indépendantes. Ainsi, un des objectifs principaux de l'architecture NFV est de séparer les fonctions réseau du matériel et de leur capacité.

### *IV B 1) c. Les intérêts de l'architecture NFV [2]*

| Avantages de l'architecture NFV | Explications |
|---|---|
| Réduction des dépenses financières | Moins de coûts de déploiement de matériel physique<br>Moins de coûts liés à l'espace physique à allouer au matériel physique<br>Moins de coûts de maintenance du matériel physique |
| Réduction de la dépendance des opérateurs réseau | Vis-à-vis de matériels physiques spécifiques |
| Meilleure scalabilité et flexibilité | Les équipements réseau virtuels peuvent être créés et déployés selon les besoins du réseau grâce à l'orchestration et l'automatisation. |

### *IV B 1) d. L'architecture ETSI pour l'approche NFV [3]*

ETSI (*European Telecommunications Standards Institute) est un organisme de normalisation dans le domaine de l'informatique et des technologies de communication. ETSI a développé une architecture standardisée pour l'approche NFV (NFV architectural framework) et c'est celle sur laquelle le projet de ce document est basé.*

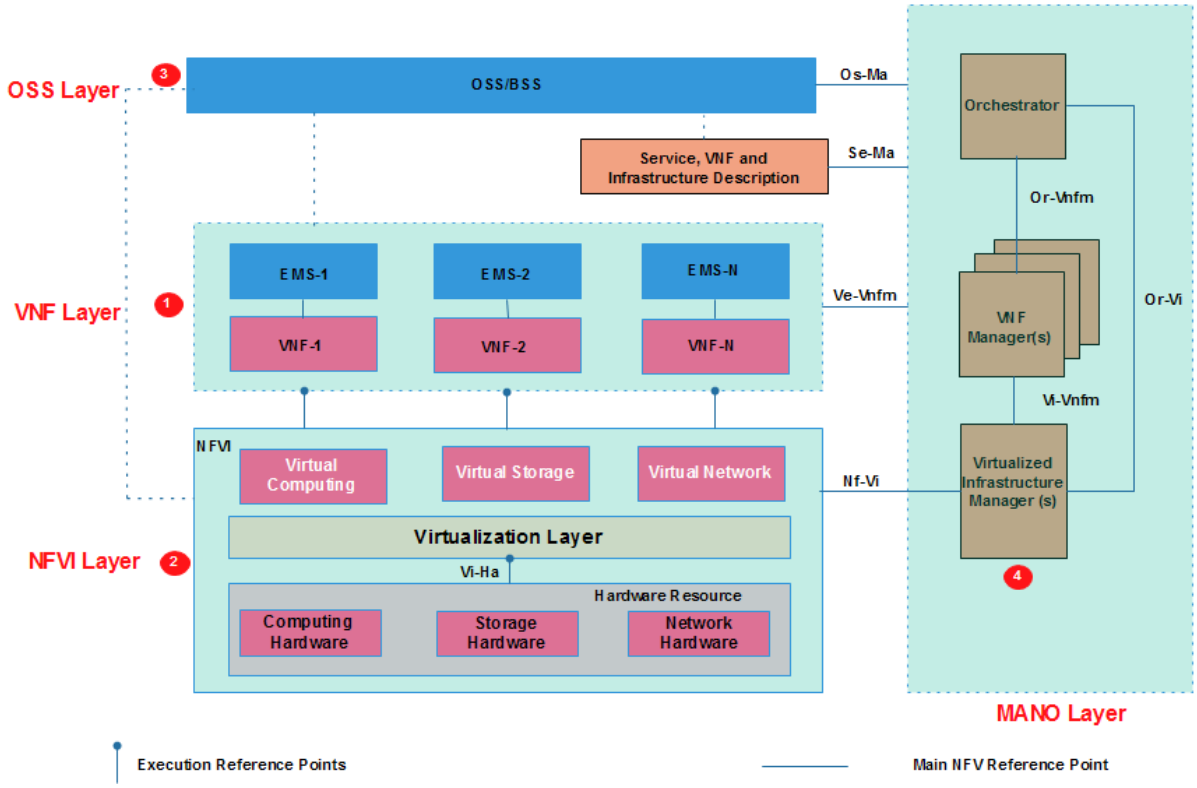


*Figure 2 Architecture NFV définie par ETSI*[b]

Comme l'illustre la Figure 2, cette architecture comprend 4 couches distinctes.

## La couche NFVI (Network Function Virtualization Infrastructure)

Le couche NFVI comprend :

**Le matériel informatique utilisé**

| Les 3 grands types de ressources physiques | Exemples |
|---|---|
| Computing (Calcul) hardware | Serveurs<br>RAM |
| Storage (Stockage) hardware | Stockage sur le disque<br>NAS |
| Network (Réseau) hardware | Switch<br>Pare-feu<br>Routeur |

**Une couche de virtualisation**

La couche de virtualisation aussi appelée hyperviseur s'occupe de virtualiser les ressources physiques présentées précédemment et de les allouer aux VM (Virtual Machines).

**Les ressources virtualisées**

Les ressources virtualisées correspondent aux 3 ressources physiques (Calcul, stockage, réseau) virtualisée par l'hyperviseur. Ces ressources sont distribuées entre les VM.

[b] http://www.techplayon.com/wp-content/uploads/2017/08/nfv-arc-3.png

### La couche MANO (MANagement and Orchestration)

MANO est parfois utilisé pour désigner uniquement la partie supérieure de la couche MANO, c'est-à-dire les parties VNFM (Virtual Network Function Manager) et Orchestrateur (aussi appelée NFVO pour Network Function Virtualization Orchestrator), en excluant la partie VIM (Virtual Infrastructure Manager). En effet, même si ces blocs font partie de la même couche, ils peuvent être assurés par des logiciels différents. C'est le cas dans ce projet puisque OSM MANO s'occupe de la partie « haute » de la couche MANO et Openstack de la partie VIM.

### Le couche OSS/BSS (Operation Support Subsystem/Business Support Subsystem)

OSS désigne l'ensemble des composants s'occupant de la maintenance opérationnelle du réseau de télécommunications déployé par un opérateur. C'est un ensemble de matériel physique et logiciel qui permet de surveiller, analyser, configurer et gérer les activités du réseau.

BSS est un système informatique s'occupant de la partie business orientée client du réseau.

## IV B 2) Les technologies de virtualisation [4]

### *IV B 2) a. Les VM*

Une **VM** (Virtual Machine) est une émulation d'un appareil informatique créée par un hyperviseur. La virtualisation à l'aide de VM s'appelle virtualisation lourde.

### *IV B 2) b. Les conteneurs*

Un **conteneur** est un espace d'exécution dédié à une application logicielle. De nombreuses applications se déploie sur plusieurs conteneurs. L'outil de création de conteneurs le plus utilisé est Docker. La virtualisation à l'aide de conteneurs s'appelle virtualisation légère

### *IV B 2) c. Les différences entre une VM et un conteneur*

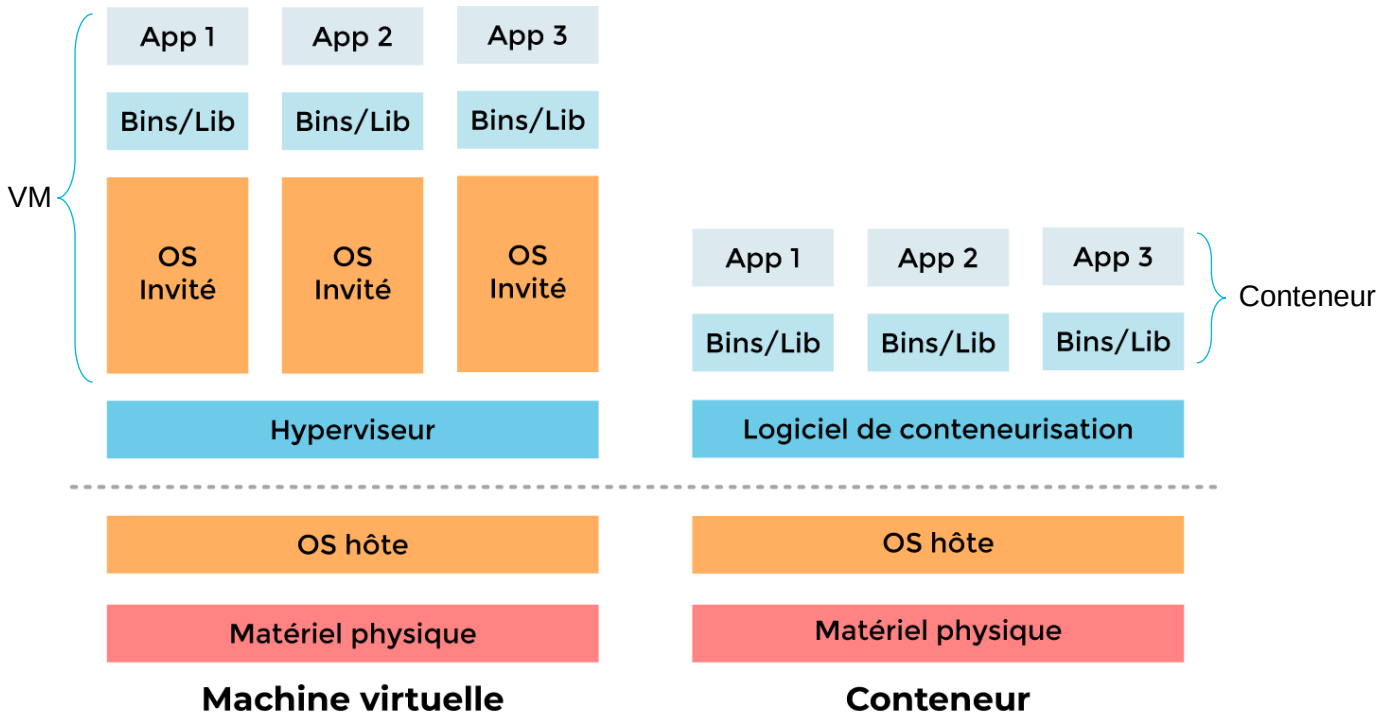


*Figure 3 Différence d'architecture entre une VM et un conteneur*[c]

Comme le montre la *Figure 3,* la différence principale entre les 2 technologies réside dans le fait que les conteneurs utilisent l'OS (Operating System, système d'exploitation) de la machine hôte tandis que les VM nécessitent la virtualisation d'un OS chacune. De plus, une VM exécute une copie de tout le matériel physique dont l'OS a besoin pour fonctionner (RAM, CPU…) grâce à l'hyperviseur.

| Avantages des conteneurs | Explications |
|---|---|
| Légèreté | Un conteneur utilise le noyau de l'OS hôte et le partage avec les autres conteneurs en prenant une place semblable à n'importe quel autre exécutable contrairement au VM. On peut donc mettre plus d'applications sur une même machine physiques à l'aide de conteneurs que de VM. |
| Facilité de la maintenance | Des VM impliquent des maintenances individualisées de chacune d'entre elles même si ce sont des copies identiques car elles sont isolées. Les conteneurs, partageant le même noyau, n'ont pas besoin de faire de mises à jour redondantes. |

| Inconvénients de conteneurs | Explications |
|---|---|
| Compatibilité avec les OS | Un conteneur Linux ne peut pas être exécuté sur Windows et inversement |
| Sécurité | Beaucoup de conteneurs sont prêts à être téléchargés sur Internet mais certains d'entre eux peuvent contenir des virus. Il faut donc bien s'assurer de la sécurité du conteneur que l'on télécharge. |

[c] https://user.oc-static.com/upload/2019/05/13/15577645779374_vm-vs-conteneur.png

### IV B 3) Le network slicing [5]

Le network slicing est une approche réseau de la 5G qui consiste à découper virtuellement le réseau en « tranches ». Chaque « tranche » est allouée à un usage bien spécifique. Par exemple, il y a une « tranche » pour le traitement des données des véhicules autonomes et une autre pour les données des communications mobiles. Le network slicing permet d'allouer les ressources nécessaires par « tranche » en étant au plus près des besoins de chaque cas d'usage. C'est une technologie qui repose sur une approche NFV des réseaux.

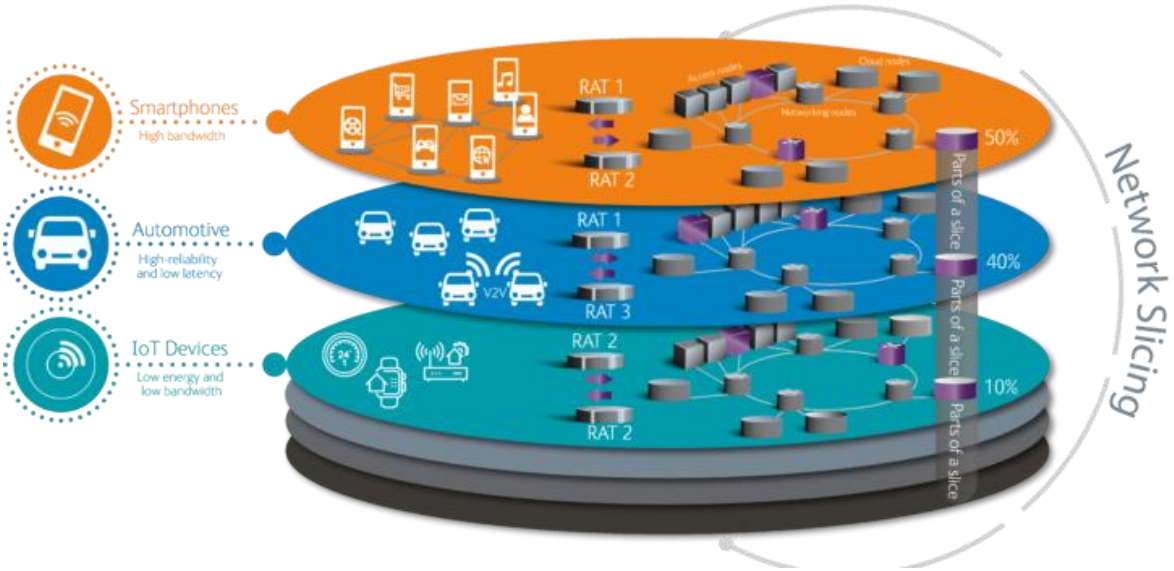


*Figure 4 Network Slicing*[d]

## IV C. Reformulation du besoin

La 5G offre avec le concept de Network Slicing de la flexibilité pour fournir des services personnalisés utilisant une infrastructure partagée et ainsi permettre de réduire les dépenses liées à l'achat et l'exploitation du matériel. Pour ce faire, les différents opérateurs seront amenés à utiliser des VNFs.

Dans ce projet, nous allons nous intéresser à un opérateur souhaitant déployer un service sur son infrastructure réseau virtualisée obéissant aux principes du modèle NFV.

Ce service comprend dans un premier temps une seule VNF. La mise en œuvre actuelle de NFV permet comme seule possibilité pour cet opérateur, de déployer avec MANO cette VNF qui s'exécutera dans une VM sur OpenStack (virtualisation lourde). Il s'agira dans ce projet d'introduire Kubernetes comme orchestrateur de conteneurs linux (virtualisation légère), puis de l'exécuter sur OpenStack en l'articulant avec MANO. Il deviendra ainsi possible à l'opérateur pour une VNF donnée, de choisir entre une virtualisation lourde ou légère. Cela se concrétisera par un déploiement automatique avec exécution dans un conteneur linux via Kubernetes ou dans un VM via MANO.

---

[d] *https://www.viavisolutions.com/sites/default/files/styles/800_wide/public/network_slicing_0.png?itok=0w5g2l8w*

L'objectif de ce projet est de permettre à un opérateur télécom exploitant un réseau 5G, de créer dynamiquement une infrastructure cloud-native (K8s cluster) sur une infrastructure de virtualisation de machines (Openstack) afin de déployer des fonctions réseau virtualisées (VNF) à base de container. Le modèle d'information ETSI NFV est utilisé pour abstraire les services réseaux.

La plateforme expérimentale réalisée est définie par un empilement de produits open source dont le socle est constitué par Openstack qui sert d'hyperviseur. La gestion des VNF est effectuée par l'orchestrateur de référence de l'ETSI, Open-Source MANO. L'originalité de ce projet consiste à articuler aux deux briques précédentes, l'orchestrateur de conteneurs Kubernetes, qui permet d'introduire la virtualisation légère.

En reprenant le schéma de l'approche NFV des réseaux, Openstack joue le rôle d'hyperviseur et de VIM pour les VM, Kubernetes joue le rôle de VIM pour les conteneurs et OSM MANO joue le rôle de de VNFM et NFVO.

# V. Plateforme d'expérimentations

## V A. Interconnexion des machines physiques

### V A. 1) Gabarit des différentes machines utilisées

Deux machines physiques ont été utilisées pour l'installation. Elles respectent les exigences suivantes :

- Machine1 : 1 processeur, 64 Go de RAM, et ... Go de stockage
- Machine2 : 1 processeur, 8 Go de RAM, et ....Go de stockage

### V A. 2) Système d'exploitation utilisé

Le système d'exploitation utilisé pour l'installation est ubuntu version 18.04.3 lts car il dispose de :

- LXD 3.0, qui propose le native clustering ou encore la migration physique vers conteneur via lxd-p2c
- Chrony, qui remplace ntpd comme serveur NTP par défaut
- Cloud-init 18.2, qui prend notamment en charge les plateformes 64 bits pour VMware et améliore les performances pour le pré-previsionning Azure

Pour réduire l'encombrement et laisser plus de ressources pour OpenStack, C'est Ubuntu 64 bits qui a été choisi.

### V A. 3) Aspect Réseau des machines

Les différentes machines utilisées sont connectées au réseau de l'IMT Atlantique grâce à un lien filaire (câble Ethernet sur les ordinateurs du laboratoire). La connexion aux machines se fait via ssh en deux étapes :

- Se connecter à la passerelle de l'école
- Se connecter à la machine

# VI. Openstack

Openstack est un ensemble de logiciels qui permet de déployer une infrastructure de cloud. C'est une solution IaaS (Infrastructure as a Service).

## VI A. Pourquoi avoir choisi Openstack ?

Openstack est très populaire, c'est une des solutions de cloud les plus utilisées, et dispose d'une communauté très active.

## VI B. Installation d'Openstack

L'installation d'Openstack se trouve en annexe car elle est très longue et n'est pas au cœur de ce projet mais elle est incontournable car sans un OpenStack opérationnel, il ne pourra pas y avoir de projet.

# VII. L'orchestrateur OSM MANO

## VII A. Pourquoi avoir choisi OSM MANO ?

Le côté open source permet à OSM MANO d'être dans le temps en termes d'innovation et réduit la dépendance à un prestataire spécifique. De plus, OSM MANO dispose d'une communauté active.

En outre, OSM MANO est compatible avec d'autres logiciels qui peuvent servir de VIM comme VMware, par exemple. Cela réduit donc la dépendance à un VIM en particulier.

## VII B. Installation de l'orchestrateur OSM MANO

Référence : [https://osm.etsi.org/docs/user-guide/03-installing-osm.html]

OSM peut être installé dans une machine virtuelle ayant accès au réseau de management d'Openstack. Pour ce faire, le site d'OSM met à disposition un script d'installation qui :

- Crée un nouveau gabarit sur Openstack (4 CPU, 8Go de RAM, 40 Go de disque)
- Ajoute une image d'Ubuntu 18.04 à Openstack
- Génère un couple de clé SSH

- Met en place les groupes de sécurités appropriés sur Openstack
- Déploie une VM sous Ubuntu 18.04 et installe OSM sur cette dernière

L'utilisateur Openstack lançant ce script doit posséder des droits **« admin » ou équivalents**.

Même si cela est possible, il n'est pas conseillé de lancer ce script directement sur le controller Openstack. En effet ce script installe des paquets qui peuvent **rentrer en conflit avec l'installation d'Openstack.** C'est pourquoi il est conseillé de le lancer sur une VM ou machine ayant accès au réseau de management d'Openstack.

Dans le cas de l'installation présentée dans ce guide, nous lancerons ce script sur une VM d'Openstack « osm-launcher ».

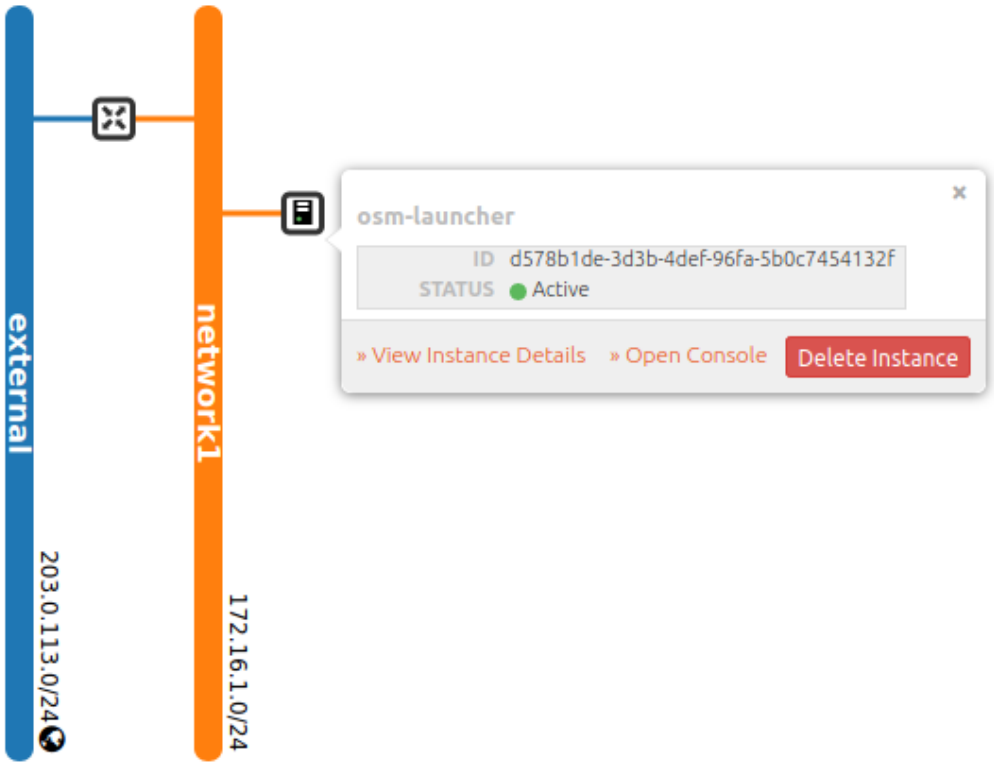


*Figure 5 Schéma du réseau avant l'installation d'OSM*

La machine virtuelle osm-launcher est une machine virtuelle sous ubuntu 18.04. Sur celle-ci, il faut ajouter dans le fichier /etc/hosts « 10.51.0.232 controller » en remplaçant 10.51.0.232 par l'adresse IP du controller. Cette étape est utile si l'installation d'Openstack utilise le mot controller dans ses urls.

Exécuter le script d'installation sur « osm-launcher » :

```
wget https://osm-download.etsi.org/ftp/osm-8.0-eight/install_osm.sh
chmod +x install_osm.sh
./install_osm.sh -O pcloud-openrc -N network1
```

Le fichier pcloud-openrc est un fichier openrc décrivant un utilisateur du projet pcloud ayant les droits administrateur et network1 est un réseau virtuel accessible sur ce projet (cf figure 7).

Il est possible que l'erreur suivante apparaisse : ERROR: Cannot uninstall 'PyYAML'. It is a distutils installed project and thus we cannot accurately determine which files belong to it which would lead to only a partial uninstall. Dans ce cas la commande suivante permet de la corriger

```
sudo -H pip3 install --ignore-installed PyYAML
```

puis relancer le script d'installation

```
./install_osm.sh -O pclouduser -N network1
```

Cette erreur vient du fait que pip désinstalle d'abord l'ancienne version de PyYAML et c'est cette étape de désinstallation qui échoue puisque le paquet n'est pas installé. L'installation est assez longue.

La paire de clés SSH se trouve dans le répertoire ~/.ssh sous les noms ansible-key.pub et ansible-key. Il est possible de les récupérer et de supprimer la VM osm-launcher puisque le script d'installation a créé une nouvelle machine virtuelle "server-osm" où OSM est installé.

Dans le groupe de sécurité network1_access, le serveur OSM accepte les connexions entrantes en TCP venant seulement du réseau network1, pour pouvoir accéder à l'interface web du serveur osm ou pour pouvoir se connecter en SSH à cette dernière. Il faut donc ajouter une règle acceptant toutes les connexions entrantes en TCP

Il est possible de se connecter en SSH au server OSM avec la commande :

```
ssh -i ansible-key ubuntu@203.0.113.239
```

en remplaçant 203.0.113.239 par l'adresse l'IP flottante de l'osm-server.

Il est aussi possible d'accéder à l'interface web d'OSM en allant à l'adresse http://203.0.113.239 (toujours en remplaçant 203.0.113.239 par l'adresse l'IP flottante de l'osm-server) où il est demandé un identifiant (admin) et un mot de passe (admin).

### VII C. Ajout d'Openstack en tant que VIM à l'orchestrateur OSM MANO

Référence : https://osm.etsi.org/docs/user-guide/04-vim-setup.html section 4.1.2

Pour ajouter openstack en tant que VIM à l'OSM il faut d'abord s'assurer que le conteneur osm_ro de la VM osm-server a accès au réseau de management d'openstack. Si ce n'est pas le cas il suffit d'ajouter dans le fichier /etc/hosts la ligne « 10.51.0.232 controller » en remplaçant 10.51.0.232 par l'adresse du controller. Il est possible d'accéder à un conteneur docker en faisant docker exec -it nom_du_conteneur /bin/bash

Ensuite il suffit de faire cette commande sur le serveur OSM pour plus d'informations sur les options voir la section 4.1.2 https://osm.etsi.org/docs/user-guide/04-vim-setup.html,

```
osm vim-create --name openstack-site --user pclouduser --password stack --auth_url http://controller:5000/v3
--tenant pcloud --account_type openstack --config='{security_groups: network1_access,
management_network_name: network1 , insecure : true,keypair: mykey}'
```

- name indique le nom sous lequel le VIM sera enregistré dans l'OSM
- user est le nom de l'utilisateur openstack qu'utilisera OSM pour utiliser openstack
- password est le mot de passe de user
- tenant est le projet openstack qu'utilisera OSM
- auth_url correspond à l'adresse d'authentification d'Openstack que l'on trouve dans le fichier openrc
- account_type indique à OSM que notre VIM est openstack
- config indique des paramètres de personnalisation :
    - security_groups : Les groupes de sécurités à déployer sur les VMs
    - management_network_name : Le réseau sur lequel les VMs seront lancées
    - insecure : autorise à utiliser ou non le protocole http pour l'authentification (dans notre cas cette options est obligatoire)
    - keypair : la clé ssh qui sera déployée sur les VMs

L'ajout d'openstack en tant que VIM peut être testé à la section 1.4.1 du lien suivant https://osm.etsi.org/docs/user-guide/01-quickstart.html

# VIII. L'orchestrateur Kubernetes

### VIII A. Présentation de K8s [6]

Kubernetes ou K8s est une plateforme open source d'orchestration de conteneurs. Il est à noter que Kubernetes ne permet pas de créer des conteneurs. Dans le cadre de ce projet, Docker est utilisé pour créer des conteneurs.

Les objectifs principaux de K8s sont d'automatiser :

1. Le déploiement
2. La mise à l'échelle
3. La gestion

d'applications conteneurisées.

### VIII A. 1) La terminologie K8s

Un **cluster** k8s est un ensemble de machines physiques ou virtuelles appelées **nœuds** sur lesquelles s'exécutent des applications conteneurisées. Un cluster k8s désigne ainsi un déploiement fonctionnel de Kubernetes ; exécuter Kubernetes revient à exécuter un cluster.

Un cluster se compose d'au moins 2 nœuds : un nœud de gestion, le **Master** et un (en pratique plusieurs) nœud de calcul, le **Worker**. Le Master est responsable du maintien du cluster dans un état défini. Les Workers sont sous le contrôle du Master et c'est sur eux que se trouvent les applications conteneurisées.

Un état défini précise :

- Les applications à exécuter
- Les images à utiliser
- Les ressources allouées à chaque application
- …

Ces informations de configuration sont présentées sous la forme de fichiers JSON ou YAML.

Un **pod** est la plus petite unité de déploiement et d'exécution sous K8s. Il a pour objectif d'exécuter une instance d'une application. Il se compose d'un ou plusieurs conteneurs déployés sur un même nœud. L'ensemble des conteneurs d'un pod partagent les mêmes IP, ports, stockage et autres ressources disponibles dans ce pod. Cependant, chaque pod dispose de son propre réseau interne. Les pods sont particulièrement utiles pour scaler horizontalement. En effet, il n'y a qu'à augmenter le nombre d'instances de l'application, chacune sur un pod différent.

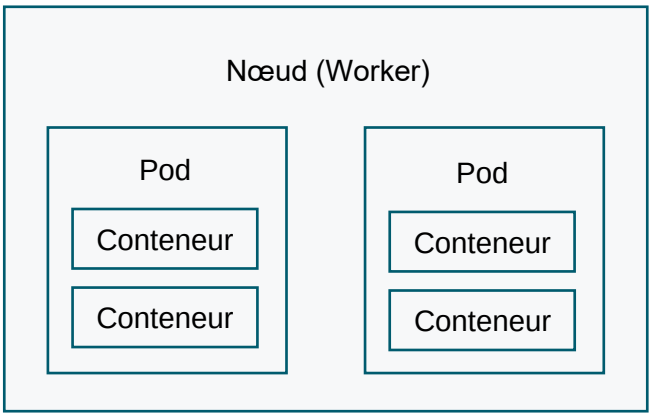


*Figure 6 Schéma d'un noeud worker K8s*

### VIII A. 2) L'architecture d'un cluster K8s

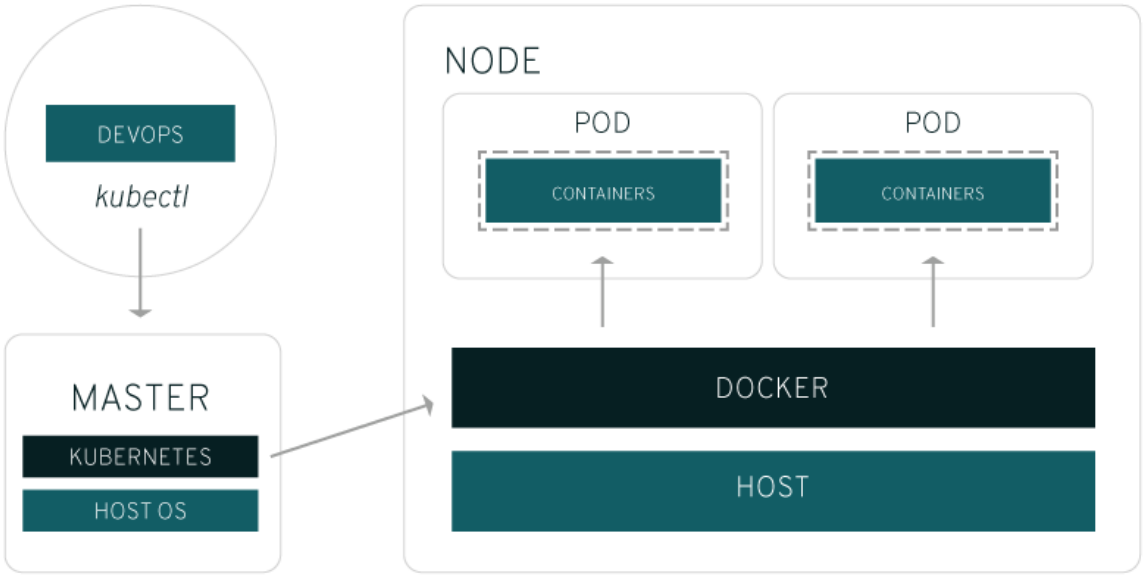


*Figure 7 Schéma de l'architecture d'un cluster K8s*[e]

## VIII B. Déploiement générique de K8s sur Openstack

Afin de préparer le déploiement d'un cluster Kubernetes contenant un master et un worker par le biais de descripteurs de VNF, nous avons fait le choix de commencer l'installation de ce cluster sur deux machines virtuelles que nous sauvegardons sur openstack et utilisons dans les descripteurs de VNF.

Pour cela, deux VMs sous Ubuntu 18.04 ayant 2 VCPUs, 4096 Mb de RAM et 20 GB d'espace disque sont utilisées.

### VIII B. 1) Mise en place de docker sur les deux nœuds

Référence :
[https://docs.docker.com/engine/install/ubuntu/?fbclid=IwAR2Tjc_8ormHaGhP9DV_fedvw-IduzMV-tf-uhvbdxglWS0tQnRkZU7ADjU ]

Kubernetes nécessite une installation Docker; il faut donc installer docker sur les deux machines virtuelles.

Pour installer Docker, mettre à jour l'index des paquets d'apt et installer les paquets suivants :

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificats curl gnupg-agent software-properties-common
```

[e] https://linuxcluster.files.wordpress.com/2020/06/kubernetes-diagram-2.png?w=640

Ajouter la clé GPG officielle de Docker :

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Puis utiliser la commande suivante pour mettre en place le dépôt stable :

```
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```

Remettre à jour l'index des paquets apt, et installer la dernière version de Docker Engine et containerd :

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Docker est maintenant installé, il suffit de l'activer au démarrage :

```
$ sudo systemct enable docker
```

### VIII B. 2) Installation de Kubernetes

Référence : [https://phoenixnap.com/kb/install-kubernetes-on-ubuntu]

Sur les deux machines virtuelles, ajouter la clé de signature de Kubernetes et les dépôts nécessaires à son installation.

Puis, toujours sur ces deux VMs, installer les outils permettant le déploiement d'un cluster Kubernetes :

```
$ sudo apt-get install kubeadm kubelet kubectl
$ sudo apt-mark hold kubeadm kubelet kubectl
```

La vérification de l'installation se fait avec la commande suivante :

```
$ kubeadm version
```

Vérifier que la mémoire swap est désactivée sur chaque nœud :

```
$ sudo swapoff -a
```

Puis renommer le nœud master et le nœud worker :

```
$ sudo hostnamectl set-hostname master-node
$ sudo hostnamectl set-hostname worker01
```

Sur le noeud master entrer la commande suivante :

```
$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

Une fois cette commande terminée, elle affiche un message “kubeadm join”. Il s’agit de la commande pour associer le worker au cluster (il est important de ne pas la perdre).

Toujours sur le nœud master, entrer ce qui suit :

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Puis sur le master déployer le réseau de pods virtuel flannel qui permet aux pods situés sur différents nœuds de communiquer entre eux :

```
$ sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

Vérifier que tout fonctionne et communique bien :

```
$ kubectl get pods --all-namespaces
```

Sur le worker, entrer la commande “kubeadm join”:

```
$ kubeadm join --discovery-token abcdef.1234567890abcdef --discovery-token-ca-cert-hash
sha256:1234..cdef 1.2.3.4:6443
```

Puis vérifier sur le master que tout s'est bien passé :

```
$ kubectl get nodes
```

# IX. Déploiement d'un cluster Kubernetes par le biais de VNFD/NSD

## IX A. Enregistrement du master et du worker en tant qu'image sur Openstack

L'objectif est de déployer un cluster Kubernetes par l'intermédiaire d'OSM. Ce déploiement est décrit dans des fichiers appelés "NFV Descriptor" et "NS Descriptor". Ils utiliseront l'image du Kubernetes master et du Kubernetes worker créées précédemment. Pour enregistrer ces deux VMs en tant qu'image sur Openstack entrer la commande suivante :

```
$ openstack server image create --name <nom de l'image> <nom de l'instance>
```

Les deux machines virtuelles peuvent ensuite être supprimées, **cependant il faut retenir leurs adresses IP qui seront utiles pour la suite**.

## IX B. Intégration des VNFDs

Créer un répertoire worker :

```
$ mkdir ~/worker
```

Ajouter le fichier worker.yaml se trouvant en annexe dans ce répertoire en remplaçant "image : k8sWorker" par le nom de l'image du worker et "hostname: worker01" par l'hostname du worker.

Compresser ensuite le dossier worker :

```
$ tar -czvf ~/worker.tar.gz ~/worker
```

Déposer ensuite cette archive sous la rubrique “VNF packages” de l’interface web d’OSM :

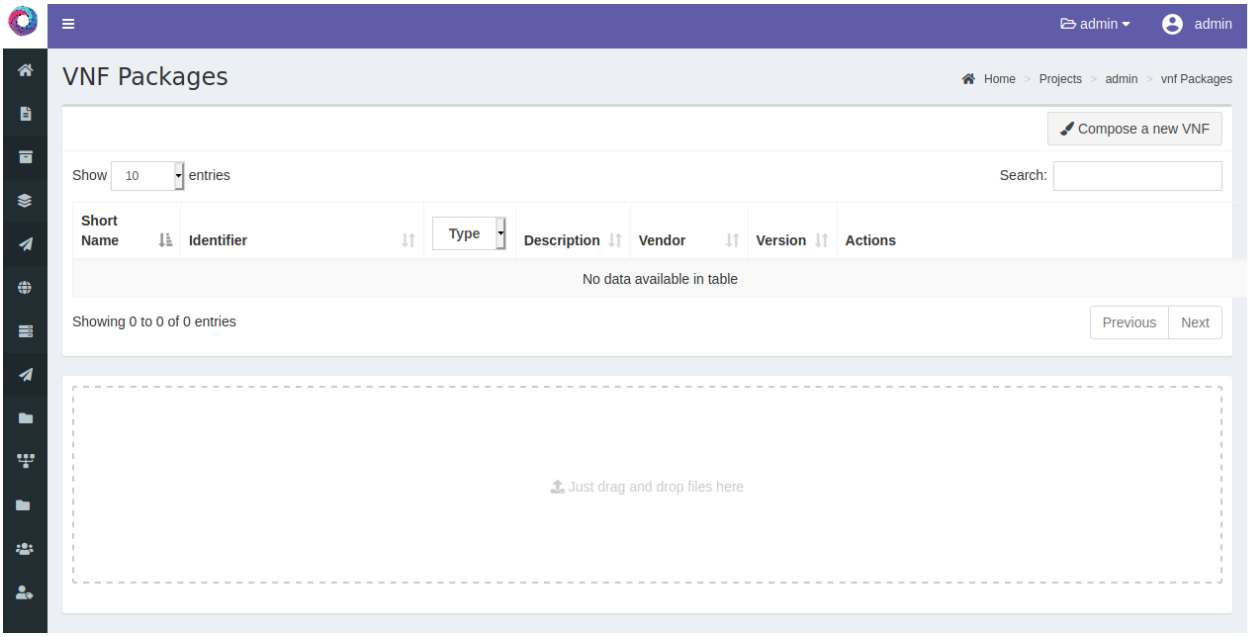


*Figure 8 Capture d'écran du dashboard de l'orchestrateur OSM Mano*

Répéter l’opération avec le fichier master.yaml

## IX C. Intégration du NSD

Créer un répertoire nsk8s :

```
$ mkdir ~/nsk8s
```

Ajouter le fichier nsk8s.yaml qui se trouve en annexe dans ce répertoire en remplaçant “ip-address: 172.16.1.193” par l’adresse IP du master et “ip-address: 172.16.1.159” par l’adresse IP du worker.

Compresser ensuite le dossier worker :

```
$ tar -czvf ~/nsk8s.tar.gz ~/nsk8s
```

Pour finir déposer cette archive sous la rubrique “NS packages” de l’interface web d’OSM.

## IX D. Déploiement du network service

Sous la rubrique “NS Instances” de l’interface web d’OSM, cliquer sur “New NS”, indiquer un nom, une description, sélectionner le network service “k8s-cluster” , puis selectionner le VIM souhaité.  Il suffit alors de cliquer sur “create” pour lancer le déploiement du cluster Kubernetes.

# X. Intégration de K8s à OSM

Référence : [https://osm.etsi.org/docs/user-guide/15-k8s-installation.html#method-3-manual-cluster-installation-steps-for-ubuntu]

## X A. Mise en place d’un load balancer

Référence : [https://starkandwayne.com/blog/k8s-and-metallb-a-loadbalancer-for-on-prem-deployments/]

Il faut désormais remplir les conditions de l'OSM, à commencer par l'installation d'un load balancer pour le cluster.

OpenStack dispose d’une protection qui empêche les VM d'utiliser les IP non configurées pour elles (antispoofing protection). Il est nécessaire de la désactiver en utilisant la commande suivante sur le port du master et du worker (dans openstack un port correspond à une interface réseau ce qui peut paraître ambiguë) :

```
openstack port set --disable-port-security id_port
```

Metallb est un load balancer facile à configurer pour kubernetes. Pour l'installer dans le cluster, il faut appliquer les manifestes suivants :

```
kubectl apply -f https://raw.githubusercontent.com/google/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/google/metallb/v0.9.3/manifests/metallb.yaml
```

Il faut également ajouter une clé secrète :

```
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"
```

À ce stade, il devrait y avoir un “controller pod” et un “speaker pod” par nœud qui devraient tous être en état de marche :

```
kubectl get pods -n metallb-system
```

Pour la suite, la configuration au niveau de la couche 2 est plus simple. Une configuration au niveau de la couche 3 avec le protocole BGP est aussi possible.

Pour la configuration au niveau de la 2ème couche, un ensemble d'adresses IPv4 allouées à MetalLB est requis.

Créer un fichier config.yml contenant ces informations en remplaçant 172.16.1.10-172.16.1.30 par le pool d'adresse IP désiré :

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - addresses:
      - 172.16.1.10-172.16.1.30
      name: default
      protocol: layer2
```

puis l'appliquer avec kubectl apply -f config.yml.

Pour tester votre configuration, appliquer le manifeste suivant :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-1
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1
        ports:
        - name: http
          containerPort: 80

---
```

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-1
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
```

Puis faire :

```
kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
kubernetes   ClusterIP      10.96.0.1      <none>         443/TCP        22d
nginx-1      LoadBalancer   10.110.150.9   172.16.1.10    80:32069/TCP   5h11m
```

Il y a un soucis avec la configuration si dans EXTERNAL-IP de nginx-1 il est écrit pending.

## X B. Définition d'une classe de stockage par défaut (storage class)

Référence : [https://docs.openebs.io/docs/next/installation.html]

L'autre configuration nécessaire est la création d'une classe de stockage par défaut. Pour ce faire, vérifier que les services iSCSI sont configurés :

```
sudo cat /etc/iscsi/initiatorname.iscsi
systemctl status iscsid
```

Si le statut du service est indiqué comme inactif, il faut l'activer et le démarrer :

```
sudo systemctl enable --now iscsid
```

Ensuite, il faut appliquer le manifeste d'openEBS :

```
kubectl apply -f https://openebs.github.io/charts/openebs-operator.yaml
```

Puis, il faut vérifier que les classes de stockage ont bien été ajoutées :

```
kubectl get storageclass
```

Enfin, il faut définir openebs-hostpath comme classe de stockage par défaut :

```
kubectl patch storageclass openebs-hostpath -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

### X C. Mise en place des droits de Tiller

Pour les clusters Kubernetes > 1.15, il faut une autorisation spéciale de Tiller qui peut être ajoutée par la commande suivante :

```
kubectl create clusterrolebinding tiller-cluster-admin --clusterrole=cluster-admin  --serviceaccount=kube-system:default
```

### X D. Ajout du cluster à l'OSM

Copier le fichier config sous le répertoire ~/.kube du master sur le serveur osm et entrer la commande suivante :

```
osm k8scluster-add cluster --creds config.yaml --vim openstack-site --k8s-nets '{"k8s_net1": "demoselfservice"}' --version "1.19" --description="Isolated K8s cluster in openstack"
```

# XI. Instanciation de KNF via Helm Chart

### XI A. Création des descripteurs de KNF

Référence : [https://osm.etsi.org/docs/vnf-onboarding-guidelines/07-knfwalkthrough.html]

Créer un répertoire fb_magma_knf sur le serveur osm :

```
$ mkdir ~/fb_magma_knf
```

Ajouter le fichier fb_magma_knfd.yaml se trouvant en annexe dans ce répertoire.

Compresser ensuite le dossier fb_magma_knf :

```
$ tar -czvf ~/fb_magma_knf.tar.gz ~/fb_magma_knf
```

Faire de même pour le fichier fb_magma_knf_nsd.yaml sous le répertoire fb_magma_knf_ns.

## XI B. Instanciation

Pour lancer le paquet KNF, d'abord il faut ajouter le dépôt qui contient le “Helm chart” :

```
osm repo-add --type helm-chart --description "Repository for Facebook Magma helm Chart" magma
https://felipevicens.github.io/fb-magma-helm-chart/
```

Puis intégrer les descripteurs à l’OSM :

```
osm nfpkg-create fb_magma_knf.tar.gz
osm nspkg-create fb_magma_ns.tar.gz
```

Il est maintenant possible d’instancier le knf :

```
osm ns-create --ns_name magma_orc8r --nsd_name fb_magma_ns --vim_account <vim_name>
```

# XII. Références bibliographiques

[1] https://www.redhat.com/fr/topics/virtualization/what-is-nfv
[2] https://www.ciena.fr/insights/articles/What-is-NFV-prx_fr_FR.html
[3] http://blogs.univ-poitiers.fr/f-launay/2018/02/04/network-functions-virtualisation-nfv-pour-le-reseau-4g5g/
[4] https://www.alibabacloud.com/fr/knowledge/difference-between-container-and-virtual-machine
[5] https://hellofuture.orange.com/fr/network-slicing-une-connectivite-5g-innovante-pour-les-vehicules-connectes/
[6] https://www.redhat.com/fr/topics/containers/what-is-kubernetes

# XIII. Annexes

## XIII A. Installation d'Openstack sur une seule machine

Référence : [http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/]

L'installation à été faite sur une machine sous Ubuntu 18.04 ayant 64 GB de RAM, 256 GB d'espace disque et 8 CPU. Cette machine a une interface réseau [enp0s31f6] connecté en fillaire d'ip 10.51.0.232 ayant accès à internet.

Dans le fichier /etc/hosts rajoutez la ligne suivante "10.51.0.232 controller" en remplaçant 10.51.0.232 par votre adresse ip.

Dans la suite du tutoriel dans les différents fichiers de config il faut remplacer l'adresse IP 172.16.50.21 par 10.51.0.232

### XIII A. 1) Etape 1 : Mise en place de l'environnement de travail

Commencer par mettre en place l'environnement de travail en suivant cette partie du tutoriel : http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/01-environnement/

### XIII A. 2) Etape 2 : Installation des paquets pré-requis

Continuer avec cette partie du tutoriel : http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/02-pre-requis/

- ATTENTION : Il faut la version de Maria DB indiqué [10.3]. Il faut faire attention à ce que la commande wget fonctionne bien; pour cela il ne faut pas taper la commande export https_proxy)
- Pour la gestion NTP le serveur ntp1.svc.enst-bretagne.fr n'est sûrement pas accessible en dehors du réseau d'IMT Atlantique

### XIII A. 3) Etape 3 : Authentification avec Keystone

Toujours en faisant attention à remplacer l'adresse IP 172.16.50.21 par l'adresse IP qui convient, suivre la suite de ce tutoriel : http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/03-keystone/

### XIII A. 4) Etape 4 : Création d'un premier projet et utilisateurs

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/04-projets-et-utilisateurs/

### XIII A. 5) Etape 5 : Service de placement

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/05_placement/

### XIII A. 6) Etape 6 : Gestionnaire d'image Glance

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/06-glance/

### XIII A. 7) Etape 7 : Service compute nova

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/07-nova/

### XIII A. 8) Etape 8 : Service de réseau nova

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/08-neutron/

- Dans le fichier /etc/netplan/51-br-ex.yaml remplacer l'adresse [203.0.113.2/24] par [203.0.113.1/24]
- Dans le fichier /etc/neutron/plugins/ml2/linuxbridge_agent.ini remplacer provider :<HOST-Interface> par provider:enp0s31f6 (le nom de votre interface réseau)

### XIII A. 9) Etape 9 : Activation du compute sur le controlleur

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/02_compute/03-nova-compute/

- Remplacer l'adresse IP 172.16.50.31 par 10.51.0.232

### XIII A. 10) Etape 10 : L'interface graphique horizon

http://formations.telecom-bretagne.eu/syst/cloud/openstack/TP/installation/01_controller/09-horizon/

### XIII A. 11) Etape 11 : Accès à internet aux machines virtuelles

On peut normalement déployer des instances les pinger depuis les instance la machine mais elle n'ont pas accès à internet.

Pour accéder à internet on configure le NAT sur le controller :

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o enps031f -j MASQUERADE
```

## XIII B. Fichier worker .yaml

```
vnfd:vnfd-catalog:
    vnfd:
    -   connection-point:
        -   name: eth0
            type: VPORT
        description: ''
        id: k8s-worker
        mgmt-interface:
            cp: eth0
        name: k8s-worker
        short-name: k8s-worker
        vdu:
        -   cloud-init: '#cloud-config

                hostname: worker01

                '
            count: '1'
            description: A kubernetes worker vnfd
            id: worker
            image: k8sWorker
            interface:
            -   external-connection-point-ref: eth0
                name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            name: worker
            vm-flavor:
                memory-mb: 4096
                storage-gb: 60
                vcpu-count: 2
        version: '1.0'
```

## XIII C. Fichier master.yaml

```
vnfd:vnfd-catalog:
    vnfd:
    -   connection-point:
        -    name: eth0
             type: VPORT
        description: ''
        id: k8s-master
        mgmt-interface:
            cp: eth0
        name: k8s-master
        short-name: k8s-master
        vdu:
        -   cloud-init: '#cloud-config

                hostname: master-node

                '
            count: '1'
            description: A kubernetes master vnfd
            id: masterVM
            image: k8sMaster
            interface:
            -   external-connection-point-ref: eth0
                name: eth0
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            name: masterVM
            vm-flavor:
                memory-mb: 4096
                storage-gb: 20
                vcpu-count: 2
        version: '1.0'
```

## XIII C. Fichier nsk8s.yaml

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: 1
      vnfd-id-ref: k8s-master
    - member-vnf-index: 2
      vnfd-id-ref: k8s-worker
    description: k8s-cluster descriptor
    id: k8s-cluster
    name: k8s-cluster
    short-name: k8s-cluster
    vendor: OSM Composer
    version: '1.0'
    vld:
    - id: ns_vl_xlmm
      mgmt-network: 'true'
      name: ns_vl1
      type: ELAN
      vnfd-connection-point-ref:
      - ip-address: 172.16.1.193
        member-vnf-index-ref: '1'
        vnfd-connection-point-ref: eth0
        vnfd-id-ref: k8s-master
      - ip-address: 172.16.1.159
        member-vnf-index-ref: '2'
        vnfd-connection-point-ref: eth0
        vnfd-id-ref: k8s-worker
```

## XIII D. Fichier fb_magma_knfd.yaml

```
vnfd-catalog:
  schema-version: '3.0'
```

```
vnfd:
- connection-point:
  - name: mgmt
  description: KNF with KDU using a helm-chart for Facebook magma orc8r
  id: fb_magma_knf
  k8s-cluster:
    nets:
    - external-connection-point-ref: mgmt
      id: mgmtnet
  kdu:
  - helm-chart: magma/orc8r
    name: orc8r
  mgmt-interface:
    cp: mgmt
  name: fb_magma_knf
  short-name: fb_magma_knf
  version: '1.0'
```

## XIII E. Fichier fb_magma_nsd.yaml.yaml

```
nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: orc8r
      vnfd-id-ref: fb_magma_knf
    description: NS consisting of a KNF fb_magma_knf connected to mgmt network
    id: fb_magma_ns
    name: fb_magma_ns
    short-name: fb_magma_ns
    version: '1.0'
    vld:
    - id: mgmtnet
      mgmt-network: true
      name: mgmtnet
      type: ELAN
      vim-network-name: mgmt
      vnfd-connection-point-ref:
      - member-vnf-index-ref: orc8r
        vnfd-connection-point-ref: mgmt
        vnfd-id-ref: fb_magma_knf
```