Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

# Cloud Computing & Open Stack

T2A Projet intégré (Analyse préliminaire)

Réseau et Sécurité

Marc Roten

2017/2018

# Table des matières

# Introduction

Ces derniers temps, on entend de plus en plus parler de cloud, de cloud computing. Tout le monde connaît ces mots mais peu de monde sait réellement de quoi il s'agit. Ce qui a pour effet si tout le monde parle de quelque chose qu'il ne connaisse pas, de rendre cette technologie opaque.

Des types de cloud, il en existe différentes variétés, celui qui nous intéresse dans le cadre de ce projet est le cloud IaaS. Les éléments manipulés au travers de ce type de services, sont tous les éléments d'un système informatique, tels que (des routeurs, des switches, des serveurs, des firewall etc…). Mais de manière plus générale, le cloud computing consiste en le fait d'exploiter en premier lieu de la puissance de calcul, du stockage de manière décentralisée, j'entends par là que l'utilisateur n'a pas à stocker ses 15'456 photos sur son HDD chez lui, il n'a pas à connaître le fonctionnement exact du système, ni ou sont stockées ses données.

Plusieurs solutions s'offrent à nous lors de la mise en place d'un Cloud de type IaaS, il y a par exemple AWS et Open Stack. Les deux offrant les mêmes services, le choix, dans le cadre de ce projet, s'est tourné vers Open Stack

# Open Stack

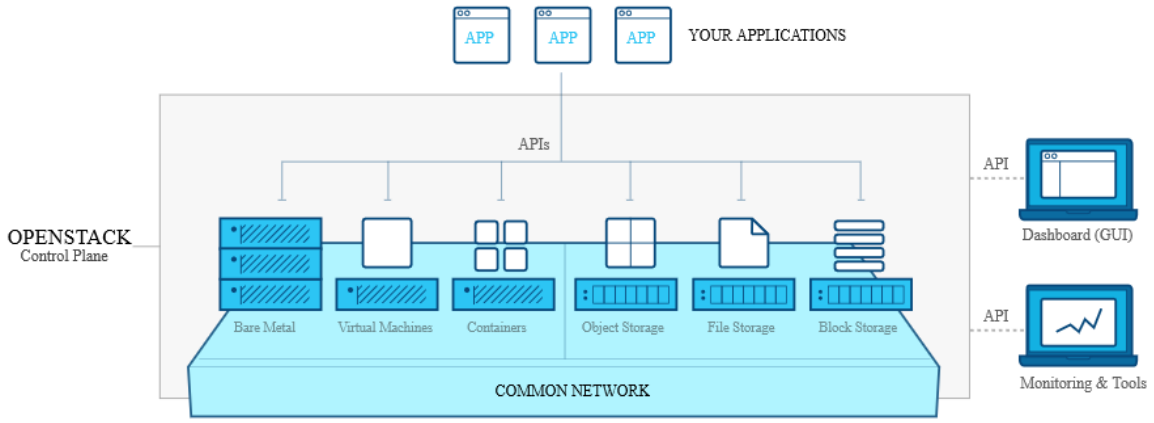



*Figure 1 Schéma concept d'Open Stack*

D'où vient Open Stack ? Open Stack, c'est avant tout un Projet open source. Lancé en juillet 2010, La Nasa et Racks Space, Rack Space étant un des leaders dans le cadre de la mise à disposition de clous, majoritairement de type IaaS. Tous les 6 mois, une nouvelle release d'Open Stack est mise à disposition des utilisateurs.

Alors Open Stack, c'est quoi ? Open Stack, c'est une plateforme open-source, de cloud computing. Cette plateforme permet le déploiement de cloud de type IaaS (infrastructure as a service). Open Stack, il faut voir ça un peu comme des legos, c'est une architecture modulaire. Il faut un certain nombre de bloc de base pour permettre le fonctionnement de cette infrastructure. Dans ce Type de cloud, on met à disposition des utilisateurs de notre cloud Open Stack, différents types de services, tels que de la puissance de calcul, de

l'architecture réseau, et du stockage, le tout au travers de « Nodes » on retrouve différents types de Nodes :

- Les Compute Nodes c'est basiquement une machine sur laquelle nos VM sont déployées. On exécute l'hyperviseur, l'hyperviseur de notre choix comme par exemple KVM, VMWare ou encore Hyper-V). Dans ce bloc, on retrouve aussi les blocs : neutron et nova. Chacun de ces nœuds requiert au minimum deux interfaces réseau.

- Les Network Nodes Ce sont ces Nodes qui offrent la connectivité vers l'extérieur de notre réseau, ces Nodes offrent des services tels que le DHCP, le NAT, OSPF, et tout autre type de protocoles. Chacun de ces nœuds requiert au minimum deux interfaces réseau.

- Les Storage Nodes Permet la création d'espace de stockage. Les Storage Nodes incluent les services de stockage de Snapshot (glance) le stockage d'objets (Swift) de d'autres qui seront abordés plus loin dans ce rapport. Chacun de ces nœuds requiert au minimum deux interfaces réseau.

- Les Controller Nodes C'est l'interface d'utilisation et de gestion. Au travers des Controller Nodes, on trouvera différentes applications d'orchestration, j'entends par là, la mise à disposition de l'utilisateur d'API pour la gestion de leur cloud IaaS. Par Exemple : l'interface utilisateur (horizon), et d'autres qui seront comme au point précédent abordés plus loin dans ce travail. Chacun de ces nœuds requiert au minimum deux interfaces réseau.

## Cloud Computing

Le cloud computing permet l'accès à différents services, sans y être directement connecté (j'entends par là, pas besoin d'aller mettre son RJ-45 dans le switch de l'infrastructure informatique). Par le biais du cloud computing, on peut souscrire à différents services tels que la mise à disposition de puissance de calcul, et de stockage.

Le fonctionnement du point de vue de l'utilisateur, l'utilisateur crée une application et dispose des ressources pour lesquelles il a souscrit. C'est donc un gros avantage des ISP de pouvoir proposer ce type de services, car adaptable à la consommation réelle du client en question.

Il existe 3 type de cloud, le cloud public, le cloud privé et le cloud hybride

### Cloud Public

Les Cloud public sont des cloud appartenant à une entreprise qui décide de mettre à disposition de clients son infrastructure. Les clients ont des tarifs intéressant dû au fait qu'ils n'ont pas besoin d'acheter par exemple des serveurs, des routeurs des switches etc…

*Figure 2 Data center de Google en Oklahoma*

Un autre avantage de ce type de cloud est notamment la scalabilité du réseau, en effet, si on souscrit à un cloud de type IaaS offert par exemple par Open Stack, on peut créer des machines Virtuelles, choisir la RAM, l'espace de stockage souhaité en quelques clics. La scalabilité est donc un des gros avantages de ce type de cloud. Pour les ISP, c'est un service facile à vendre, vu que le coût est adaptatif à l'utilisation réelle de l'utilisateur.

| Avantages | Désavantages |
|---|---|
| Rapide et peu couteux à la mise en place | Moins sécurisé? Ou sont stockées nos données? |
| Facilement adaptable | Pas forcément adapté aux besoin d'une entreprise |

*Figure 3 Avantages et limites des clouds publics*

## Cloud Privé

Le cloud privé permet d'avoir un contrôle total de ses données. Par exemple les entreprises manipulant des données sensibles. Si la sécurité est l'élément central de votre réseau, le cloud privé est la solution. Mais attention, la sécurité est accrue si les gens ayant mis en place le dit réseau sont compétents.

Le principal désavantage de ce type de réseau est le fait qu'il faille avoir l'infrastructure chez soi, ou dans un datacenter nous appartenant, ce qui peut être encombrant et très couteux

| Avantages | Désavantages |
| --- | --- |
| Adapté aux besoins plus spécifiques des entreprise<br><br>Sécurité des données accrue (voir désavantage n°1) | Sécurité amoindrie (si le personnel n'est pas compétent) |

*Figure 4 Avantages et limites du cloud privé*

# Type de Services fournit par les clouds standards

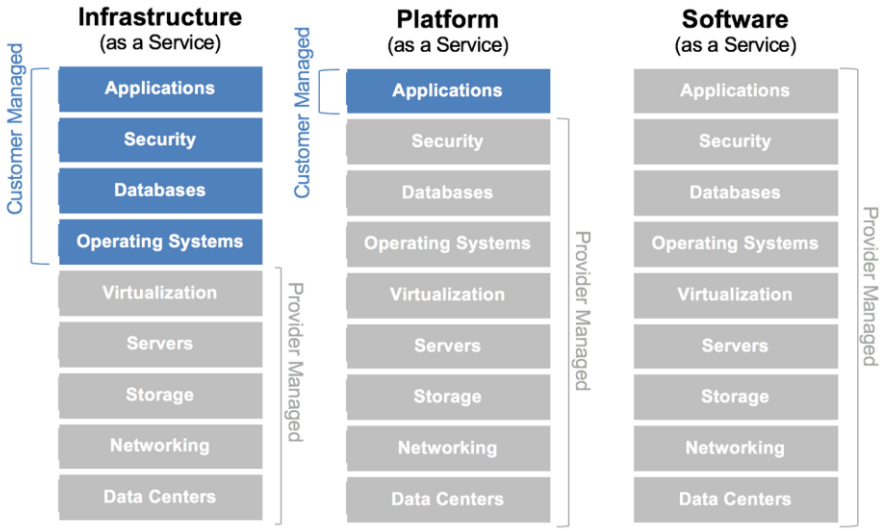



*Figure 5 Différents types de services*

## IaaS (Infrastructure as a Service)

Dans les cloud de type IaaS, on donne accès aux utilisateurs à différents services. Le client a une certaine liberté. Le client peut choisir son OS, qu'il veuille travailler sous différents linux sous son OS préféré. Le système d'open Stack fonctionne sur le concept de location de VM pour faire tourner différents Nodes, formant une infrastructure. L'ISP s'occupe de tout ce qui est des étages de virtualisation, de serveur de Storage de networking et de Datacenter.

## PaaS (Platform as a Service)

Ce type de service ne remplace pas l'intégralité d'une structure informatique d'entreprise, en effet, c'est l'ISP qui s'occupe de toutes les couches inférieures (voir figure 5). L'utilisateur dépend donc du fournisseur, il peut donc se retrouver avec un OS imposé sur un hyperviseur imposé. L'utilisateur installe ses applications qu'il souhaite utiliser.

## SaaS (Software as a Service)

Dans ce cas-ci, l'utilisateur n'a aucun souci de configuration, le fournisseur se charge de tout, c'est un peu la solution « clé en main ». Idéal pour les clients inexpérimentés ou les entreprises voulant des logiciels faciles à la prise en main et ne nécessitant pas de formation particulière

# Vue d'ensemble d'Open stack

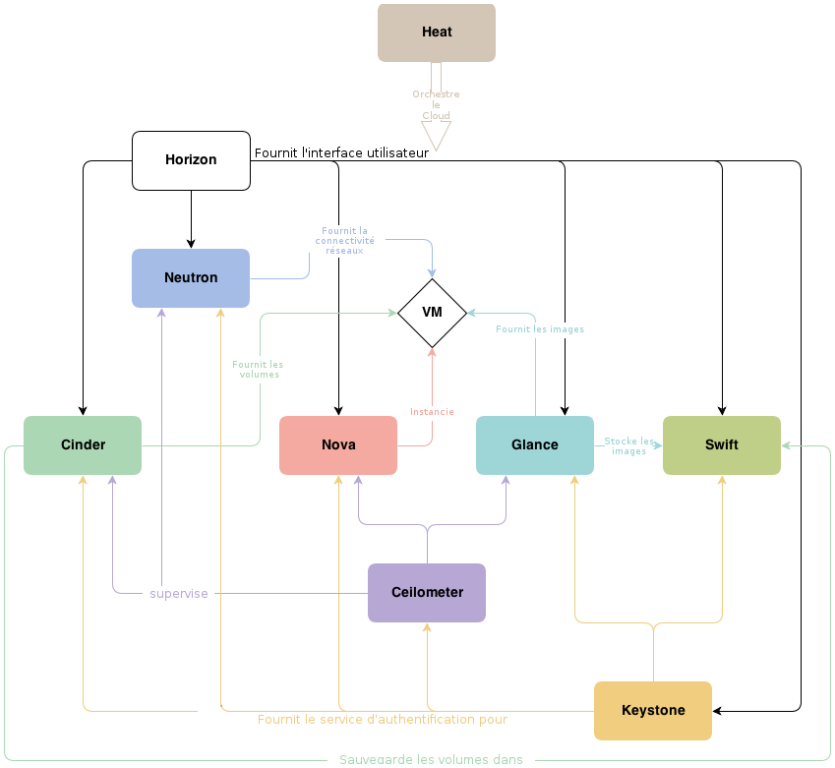



*Figure 6 concept d'Open Stack*

## Services principaux

### Nova

Nova est un des plus gros blocs, il prend les images que Glance lui fournit et les instancie.

C'est lui qui s'occupe de la gestion des VM, il s'occupe de générer des nouvelles instances et de shutdown les instances des VM. C'est en quelque sorte le cœur de notre infrastructure.

Nova fonctionne sur du matériel non spécialisé, j'entends par là qu'on peut utiliser des serveurs déjà présents. Pas besoin de serveurs plus chers, on utilise le matériel déjà à disposition.

Outre le fait que ça tourne sur du matériel standard, on peut aussi utiliser l'hyperviseur de son choix, que l'on veuille travailler avec Hyper-V, VMWare, ou encore KVM.

### Horizon

Horizon est le Dashboard, j'entends par là l'interface graphique destinée à l'utilisateur pour pouvoir gérer toute l'infrastructure qui lui est mise à disposition. Comme tous les modèles affiliés à open stack, on peut voir le code source et modifier, selon ses goûts.

## Swift (Object Storage)

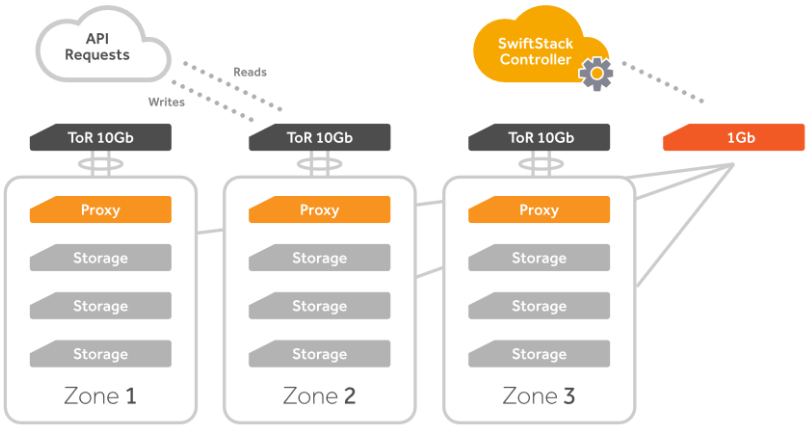



*Figure 7 concept de Swift*

La manière la plus élégante et la plus pratique est le stockage d'objet. Que ce soit pour stocker des PDF, des .zip des tars, ou encore des photos, l'Object Storage est le plus adapté. Swift est un service qui fonctionne sur le principe de la réplication (en principe 3 répliques) sont nécessaires dépendant de la taille de l'infrastructure mise en place. Si une panne survient dans notre réseau, l'accès aux données est toujours garanti, en effet, les données sont répliquées à plusieurs endroits distincts, l'accès est donc toujours garanti.

## Cinder (block Storage)

Cinder est un service en quelques point semblable à Swift mais les deux projets diffèrent sur certains points. Dans le titre, j'ai écrit block Storage, nos fichiers sont dissociés en petits blocs de même taille, et répartis à différentes adresses.

Le principal avantage de ce type de stockage est la rapidité. Mais Il y a un désavantage par rapport à Swift, c'est l'absence de métadonnée.

C'est le bloc utilisé pour tout ce qui concerne les espaces de stockage pour nos VM. Il est aussi utilisé pour ce qui concerne le stockage de base de données. En effet, le fait que la rapidité soit un de ses atouts, le fait de l'utiliser pour les bases de données est bénéfique.

## Neutron (NaaS Network as a Service)

Nova devenant trop grand, en termes de projet, ils ont décidé de créer Neutron pour la gestion du réseau. A la base c'était un module de nova qui s'occupait du NaaS.

Neutron se charge de fournir de la connectivité aux instances de nos VM. Neutron prend en charge la connexion Layer2-Layer3.

Quand on se trouve en couche 2, on reste dans le même réseau. La couche 3 gère le routage des paquets dans les différentes instances de nos VM si on possède deux réseaux distincts.

Pour tout ce qui conseille l'adressage par notre NaaS, on peut procéder é un adressage de manière manuelle ou en DHCP

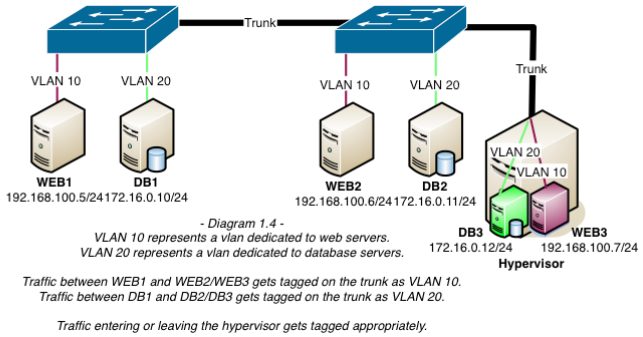



*Figure 8 Exemple de configuration avec VLAN*

On peut aussi créer via neutron des VLAN pour au sein d'une même entreprise ayant accès aux mêmes infrastructures, qu'il y ait une séparation entre les différents membres.

Neutron peut aussi servir de Firewall, FWaaS (Firewall as a Service), offrant les mêmes avantages niveau sécurité qu'un firewall standard.

### Glance

Glance est le service qui gère les Snapshots, c'est lui qui permet l'envoi et la distribution des images vers Nova. Glance ne stocke pas uniquement les snapshots et images pour nos VM, il contient aussi des métadonnées. Il travaille conjointement avec Swift dans lequel Glance peut stocker ses images disque

### Keystone

Keystone fournit le service d'authentification pour les autres blocs (nova, glance, Swift, Cinder, ceilometer, neutron). Tous les services doivent passer par Keystone pour s'authentifier. Les utilisateurs du cloud IaaS doivent aussi passer par Keystone pour pouvoir faire quoi que ce soit à l'intérieur de l'infrastructure.

### Heat

Il faut voir un peu Heat comme le chef d'orchestre. S'il détecte notamment par le système de métrique utilisé par Ceilometer, la nécessité de lancer de nouvelles instances si une application nécessite plus de puissance de calcul

### Ceilometer

Ce service permet de voir indépendamment de chaque utilisateur, l'utilisation du cloud, le nombre de machine virtuelle et la durée de vie des VM's. Ce service est utile à des fins de facturation. On peut adapter le prix à la consommation réelle. Ce qui est un réel avantage.

## Architecture

Architecture logique en annexe

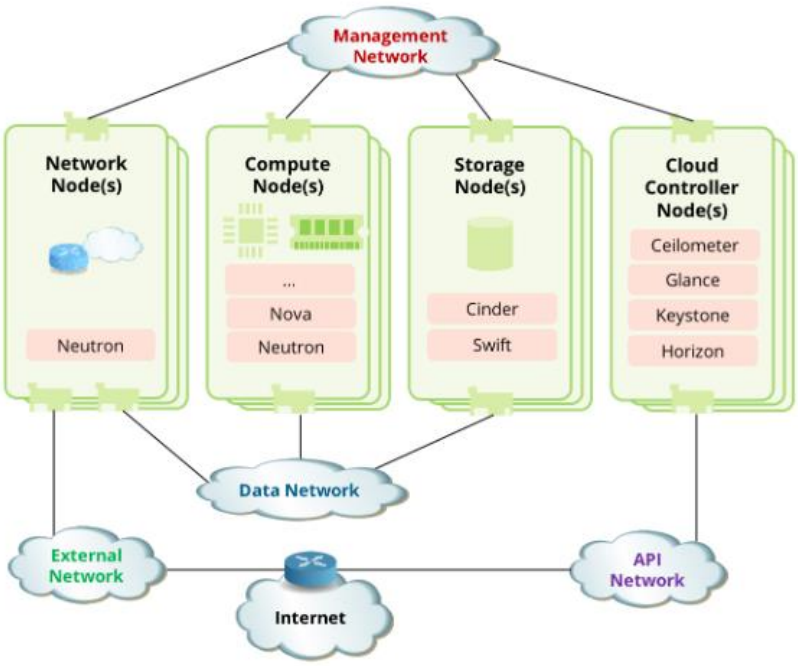



*Figure 9 Concept d'architecture*

## Cloud Controller Node

Le Cloud Controller Node est le regroupement des différents Node qui gèrent tout ce qui concerne l'accès, le contrôle des utilisateurs, la gestion et l'utilisation de notre infrastructure. Dans une infrastructure de petite taille, un cloud Controller peut suffire. Dans le cas d'une filiale plus grande que deux succursales, il est judicieux de disposer de deux cloud Controller. Selon la requête, le cloud Controller va accorder l'accès ou refuser l'accès à l'infrastructure. Si l'accès est accordé, le cloud Controller va ensuite envoyer les informations nécessaires aux autres (le Storage Node(s)) le Compute Node(s), et le network Node(s)).

Chaque Cloud Controller fournit un service de base de données de type MySQL. Dans notre cas, selon notre donnée, il nous faudra au minimum deux interfaces réseaux.

## Storage Node

On peut de manière optionnelle décider de créer des espaces de stockage sous forme de bloc. Il est possible de disposer d'autant de Storage nodes que l'on souhaite. La seule limite étant la limite fixée par l'ISP ou selon la limite de notre budget pour notre cloud. On doit passer par le réseau de Management lorsque l'on veut faire communiquer un Storage Node avec un nœud compute. (Voir illustration page suivante)

## Compute Node

C'est dans le Compute Node que les VM sont actuellement déployées, exécutées à cet endroit par l'hyperviseur de notre choix. Chaque fois qu'une VM est créé, l'utilisateur doit au travers du Dashboard Horizon choisir quel CPU, RAM, HDD, (Snapshot éventuellement) dont il a besoin

L'utilisateur ne parle pas au Compute Node directement. L'utilisateur passe par le Cloud Controller Node, qui lui fournit, ou non (si ce n'est pas un utilisateur référencé dans la base de données) l'API nécessaire pour qu'il puisse tout gérer depuis Horizon jusqu'au network Node. Toutes les requêtes de l'utilisateur se font au Cloud Controller et nulle part ailleurs.

## Network Node

Avant de lancer une instance de network Node, il est nécessaire de créer notre infrastructure virtuelle. Pour se faire, il y a plusieurs options citées juste ci-dessous.

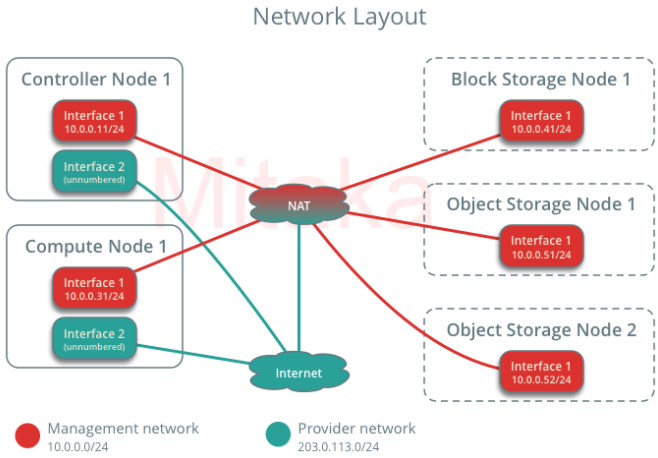



*Figure 10 Network overview*

## Provider Network

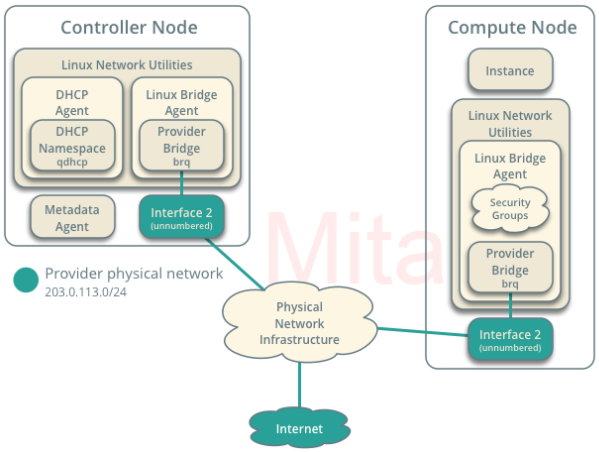



*Figure 11 network overview*

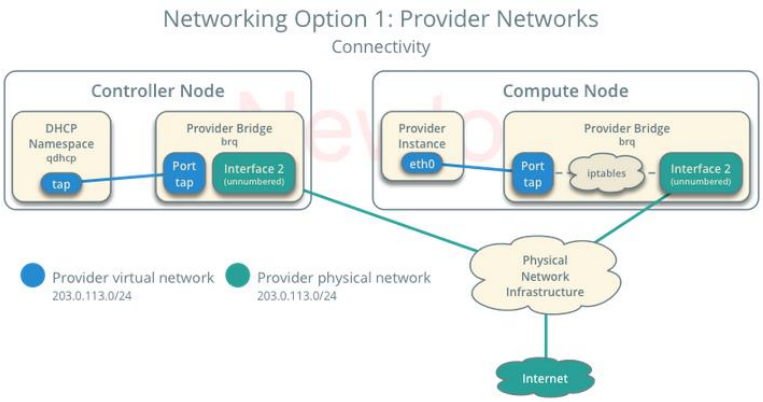



*Figure 12 Details provider network Connectivity*

L'option n°1 disponible est l'option Provider Network. On déploie dans ce cas-ci, open Stack de la manière la plus simple. Notamment par des mécanismes de couche 2 tels que le bridging et le switching. Pour ces derniers, on utilise l'infrastructure physique déjà en place.

Cette option utilise les VLAN's déjà mis en place, on dépend donc de l'infrastructure physique de notre réseau.

Pour la gestion des VLAN, avec cette option, on ne peut pas créer de nouveaux VLAN pour segmenter notre réseau en sous réseau. Avec cette option, on peut juste utiliser les VLAN déjà configurés. Pour l'adressage des instances un service DHCP est disponible. Mais si l'n souhaite, on peut aussi procéder à un adressage manuel.

## Self-Service Networks

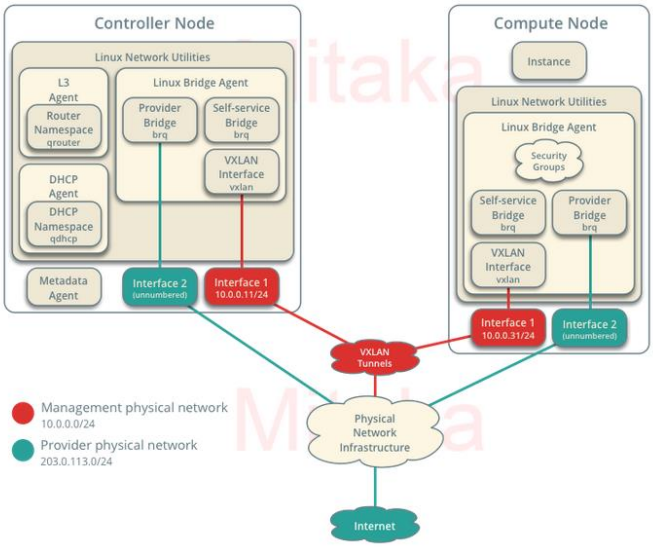



*Figure 13 Self-service network aperçu*

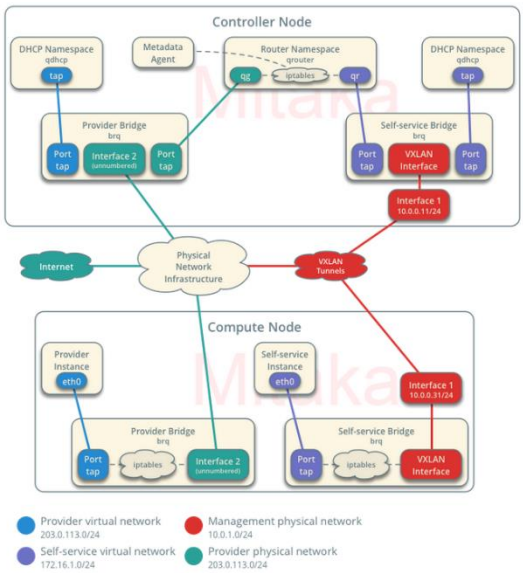



*Figure 14 Self-Service network Connectivity*

L'option n°2 disponible pour les Compute Nodes est le réseau libre-service. C'est en quelque sorte une version améliorée du Provider Network cité auparavant. L'avantage de ce type d'architecture est que l'on peut segmenter de manière quasi illimitée le nombre de réseau virtuel grâce à la technologie VXLAN.

Pour résumer l'avantage du self-service networking est qu'il permet aux utilisateurs de créer leur propre réseau virtuel, de sous-réseaux, de manière quasi illimitée. Ce type d'architecture : Network Node, utilisent le principe de tunneling en utilisant la technologie GRE ou VXLAN, en passant par un routeur Open Stack

Les routeurs Open Stack ont une interface dans le réseau externe et une autre interface sur le réseau externe. Selon si l'on se trouve sur la release actuelle (Mitaka, ou la release Kilo) on peut avoir plusieurs réseaux externes. On peut donc assigner des adresses IP flottantes à nos instances (FIP), ou on peut simplement assigner nos instances avec le provider-network (à condition d'avoir les droits d'admin).

# Architecture simulée dans ce TP

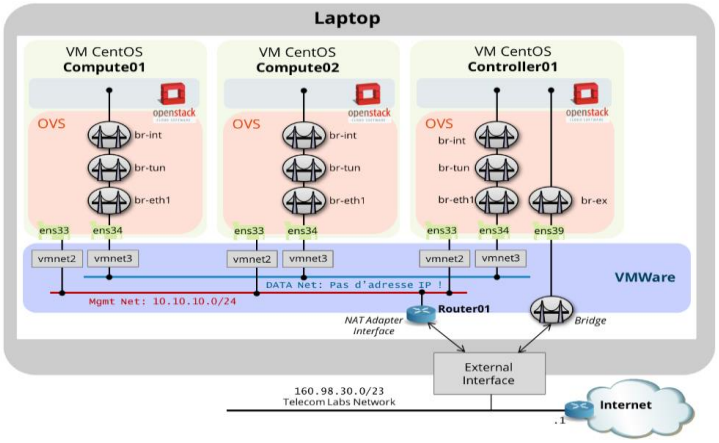



*Figure 15 Architecture de la donnée du TP*

## Open vSwitch (OVS)

En figure 10, on retrouve l'architecture simulée de notre infrastructure. En rouge, on trouve OVS, qui est l'abréviation d'open vSwitch. Open vSwitch qu'est-ce-que c'est ?

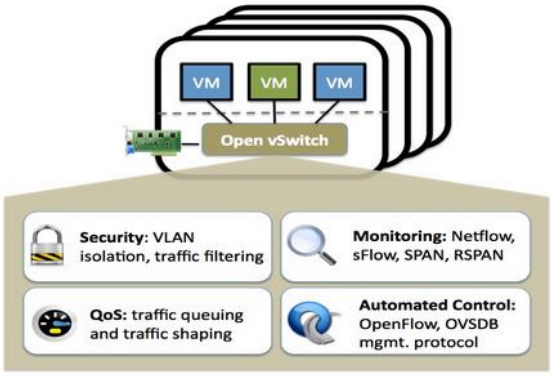



*Figure 16 concept OVS*

Open vSwitch est capable de simuler des switches utilisé typiquement avec les hyperviseurs pour interconnecter différentes machines virtuelles sans avoir besoin d'un nouvel hôte. Et faire le lien entre les VM et les différents hôtes au travers de notre réseau. Peut-être le maillon central dans une solution SDN.

Open vSwitch en plus de permettre l'interconnexion entre les différentes VM, et les différents Nodes. On ajoute encore une couche de sécurité. On peut ajouter via Open vSwitch, un firewall. Open vSwitch reprend toutes les notions de réseau commuté. Notamment les protocoles SPAN (Switched Port Analyzer) et les protocoles RSPAN (Remote Switched Port Analyzer),

Features apportées à notre réseau

- VLAN 802.1q
- Spanning Tree protocol
- Port mirroring (R)SPAN (Remote Port Analyzer)
- QoS control

Pour la communication inter-VM's, ainsi que la communication couche 2-couche 3, on utilise la notion de tunneling GRE (Generic Routing Encapsulation)

Spécificités d'Open vSwitch

- Pas de sécurité dans le Kernel, donc cheminement des données rapide
- Chaque composant de notre infrastructure (physique ou virtuelle) peut être ajoutée par un UpLink

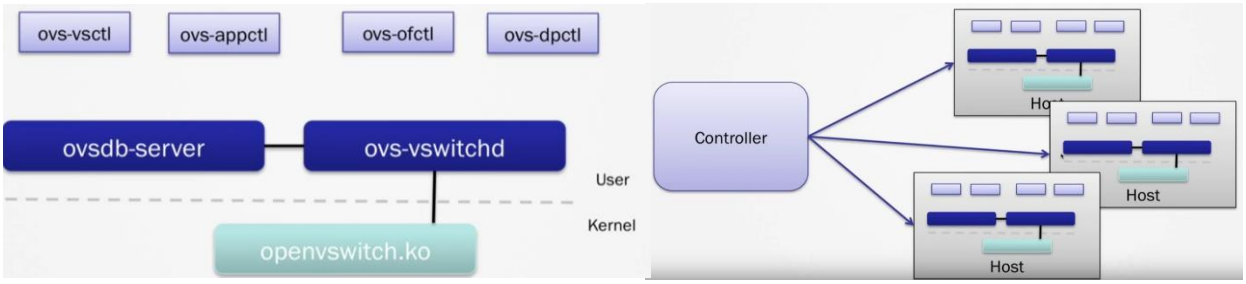



*Figure 17 architecture d'un composant*

Daemon : Logiciel ou processus ou ensemble de processus qui s'exécute en arrière-plan, et donc sans contrôle de l'utilisateur.

- Ovs-svctl ce bloc est nécessaire pour la configurer ovs-vswitchd en fournissant une interface haut-niveau à sa configuration. C'est ce bloc qui permet de maintenir une configuration possible à tout moment. On l'utilise pour ajouter, supprimer des ports de notre switch, il est donc utile pour tout ce qui concerne la configuration de notre switch.
- Ovs-vsswitchd C'est un Daemon qui va s'occuper de manager et de contrôler un certain nombre d'Open vSwitch le ovs-vswitch de la figure de gauche, est donc le contrôleur de la figure de droite.
- Ovs-appctl C'est lui qui dit au Ovs-svctl, quel Daemon il doit contacter. L'idée une fois automatisé est que l'utilisateur n'aie plus rien à faire. L'infrastructure informatique, mais plus précisément le cloud doit être une solution qui apporte un plus. Il faut donc qu'une fois mis en route, le réseau soit autonome.
- Ovs-ofctl la porte d'entrée vers nos switches, avec cette commande, on a un accès direct à la configuration du switch, aux tables de switching
- Ovs-dpctl c'est la commande pour ajouter de nouvelles entrées à nos tables de routage de nos switches.
- Ovsdb-server c'est le serveur qui s'occupe des bases de données pour les différents open vSwitch. Ce server fournit une interface RPC à un ou plusieurs Open vSwitch database. Dans le cas de nos open vSwitch, il s'occupe de nos tables de switching. Les clients extérieurs peuvent communiquer avec le ovsdb-server via json rpc

## VXLAN (Virtual extensible LAN) RFC 7348

VTEP = VXLAN Tunnel End Point = porte d'entrée ou sortie de notre tunnel VxLAN

Le VXLAN est en quelque sorte une version améliorée des VLAN, leur développement est du à la limite du nombre de VLAN en 802.1Q (limite de 4096). La technologie VXLAN est adaptée pour les Cloud, lieu ou la scalabilité est un point essentiel.

Ce sont des VLAN déployable de manière dynamique par-dessus un réseau IP routé. Ce système d'encapsulation VXLAN est un projet né d'un partenariat entre Cisco et VMware. Ce protocole vise à faire passer des trames layer 2 dans de l'UDP. On peut donc utiliser nos VLAN plus seulement dans notre domaine Ethernet, mais au-delà de notre réseau local. Cela est là principale caractéristique qui en fait un atout majeur pour la scalabilité des Cloud.

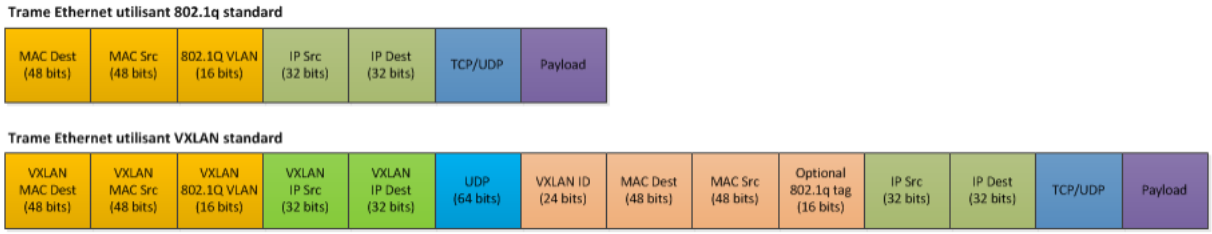



*Figure 18 différence trame Ethernet 802.1Q vs VxLAN*

Une des principales différences est le VxLAN ID de 24, on peut donc créer 2^16 vlan soit plus de 16'000'000 de VLAN.

On encapsule en quelque sorte des trames Ethernet dans de l'UDP. Il est possible de procéder à un routage des différents VTEP (Virtual Extensible LAN), on ajoute donc un niveau d'abstraction par rapport aux domaines Ethernet.

Dans de l'Ethernet standard, une adresse de broadCast est utilisée. Dans le cas de nos VXLAN, on utilise une adresse de multicast.

On déploie donc via routage en couche 3, différents VTEP sur notre VxLAN, c'est donc un avantage majeur, et du au fait que les adresses utilisées par nos VXLAN sont en multi-cast. Lors d'un déploiement d'un nouveau VTEP, il suffit simplement d'ajouter son adresse aux adresses de multicast du VXLAN.

Le VXLAN est une amélioration des VLAN, il permet donc de faire passer des données de broadcast entre les différents membres de son multicast.

Pour résumer, on a différents « tunnels » entre notre réseau et les différents nodes en Multicast. Et pour les compute Node entre eux, ils seront dans le même domaine de broadcast, leur permettant de ce fait de répondre à différents types de requêtes, notamment les requêtes de type ARP.

## RPC (Remote procedure call) RFC 1057

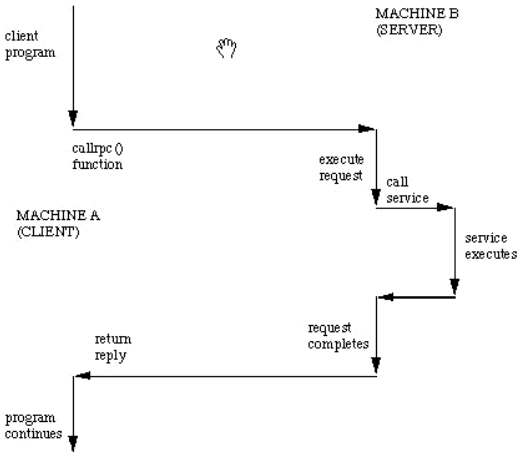



*Figure 19 Schéma RPC*

Le process Remote procedure Call est quand l'ordinateur réalise un process dans le but de s'exécuter dans un autre endroit. J'entends par là que dans un autre compute Node, ce dit

process est codé comme s'il était un « procedure call » local, sans que le programmeur aie à s'en soucier.

En résumé, le remonte procedure call est une commande externe à un programme, est une commande externe

## SDN (Software defined networking)

SDN est une solution qui nous offre une API, nous offre une manière de contrôler de manière distante de notre réseau, nos software/hardware. Développée à la base en réponse à la demande des larges datacenters. La majeure partie du SDN est gérée par Neutron, c'est en quelque sorte un supplément à neutron, pour rajouter des accès facilités à notre infrastructure

- Utile si le Traffic n'est pas prévisible.
- Possibilité de configurer le réseau rapidement

RPC SouthBound pour contrôler les différents « agents » (layer 3, DHCP, metadata). Le contrôleur SDN interagit avec nos open vSwitch, plus précisément avec ovs-vswitchd par des protocole openFlow.

Avantage de bénéficier d'une solution SDN

- Liberté de choix. Possibilité de connecter des appareils issus de différentes technologies (différents hyperviseurs supportés, possibilité de travailler avec docker, ou les concurrents de docker, un peu moins connus que sont Kubernetes (google) et Mesos(apache).
- Avoir un réseau commun pour tous nos services

Désavantage du SDN

- Nécessité d'avoir une infrastructure
- Nécessite une reconfiguration du réseau
- Formation du personnel

## DragonFlow

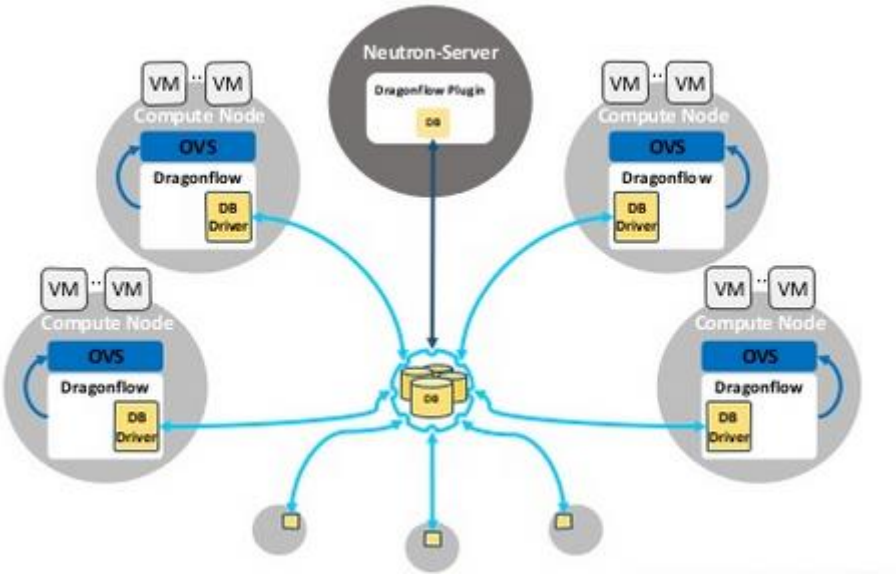



*Figure 4 DragonFlow concept*

DragonFlow est un peu une update de Neutron, rajouté il y a peu, il apporte tous les avantages qu'on peut attendre d'un switch Layer2-Layer3 et de routeurs. C'est-à-dire un accès distant, des fonctionnalités de routage, de switching de DHCP, et de différents protocoles.

DragonFlow est en quelque sort un contrôleur SDN. DragonFlow est le pilier central de la gestion SDN de notre réseau. Les autres solutions telles que openDayLght, contrail ont aussi leurs avantages, mais de manière standard, on a à disposition DragonFlow.

Les avantages de DragonFlow sont :

- Scalabilité
- Latences et performances

Grâce au plugin DragonFlow dans Neutron Server, si on veut augmenter la taille de notre réseau, on aura accès aux différentes bases de données. Des différents compute Node.

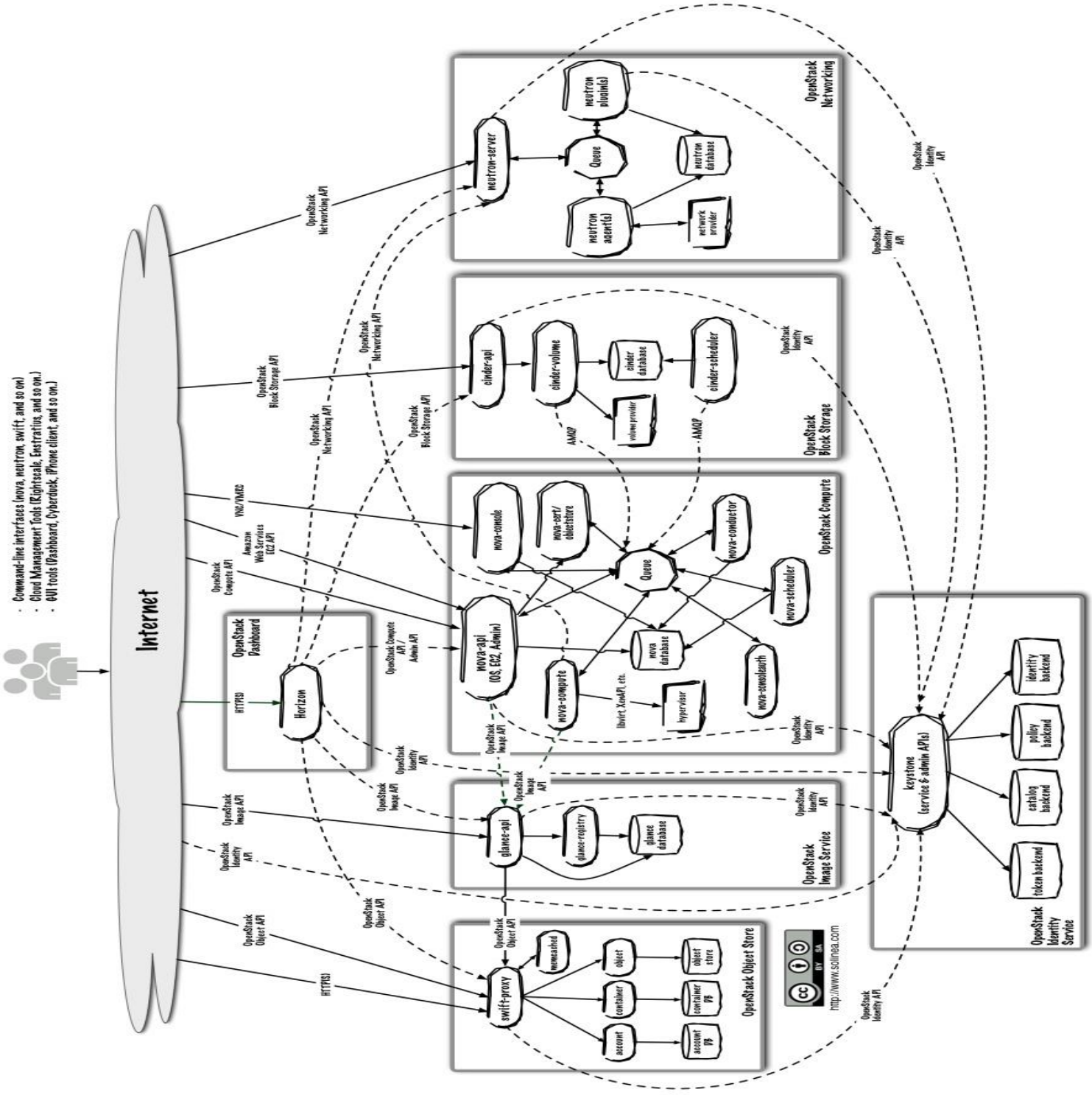
DragonFlow tourne localement sur chaque compude Node. C'est un contrôleur qui implémente l'API de Neutron en utilisant les principes et spécificités de SDN cité au chapitre précédent.

Toutes les bases de données de nos différents compute nodes sont mises en commun, sans pour autant les laisser en libre accès. Il faut procéder à une demande de type RPC, si aucune demande de ce type n'est effectuée, seuls les éléments à l'intérieur de notre compute Node.

Pour schématiser les compute Node, c'est un peu comme un sous-réseau de notre réseau principal.

# Annexe

## Architecture logique



# Conclusion

Au terme de cette analyse préliminaire, on a pu mieux comprendre les principes de cloud Computing, de notions nouvelles de Open Stack. Cette analyse préliminaire nous sera utile pour la suite du projet, pour avoir un meilleur compte rendu. Ces rapports préliminaires serviront à éviter aux autres membres du groupes de nombreuses heures de recherche, mais de quand même comprendre le concept, le fonctionnement de l'architecture d'Open Stack.

## Travail à venir

Après ce travail, il nous reste tous les aspects de conception, de design. La suite du projet sera contrairement à cette partie, un travail de groupe. Il sera nécessaire de transmettre ces copies aux différents membres du groupe pour l'avancement de ce projet. Après cette étape, il faudra rédiger le rapport d'expérience final et monter le réseau réel. Le travail s'annonce fort intéressant.

## Liste des Figures

Fig.20 DragonFlow concept https://www.slideshare.net/gampel/dragonflow-01-2016-tlv-meetup

## Reference


https://www.avineon.com/en/cloud-computing

https://www.openstack.org/software/

https://www.lebigdata.fr/definition-paas

https://blog.osones.com/le-naas-dopenstack-neutron-introduction.html

https://cyberlearn.hes-so.ch/pluginfile.php/2477951/mod_resource/content/1/OpenStack_Install_v1.2.pdf

https://www.swiftstack.com/product/hardware-infrastructure-configuration

https://docs.openstack.org/mitaka/fr/install-guide-ubuntu/launch-instance.html

https://ask.openstack.org/en/question/58926/what-is-a-cloud-controller/

https://ask.openstack.org/en/question/50263/what-is-openstack-compute-node/

https://docs.openvswitch.org/en/latest/internals/security/

https://www.openvswitch.org/

https://www.pica8.com/document/v2.3/html/ovs-commands-reference/

https://www.lemarson.com/thematique/1137/les-concurrents-de-docker-oui-ils-existent

https://www.alliedtelesis.com/

https://wiki.openstack.org/wiki/Dragonflow#What_is_Dragonflow.3F

https://www.slideshare.net/gampel/dragonflow-01-2016-tlv-meetup

https://users.cs.cf.ac.uk/Dave.Marshall/C/node33.html

https://docs.openstack.org/mitaka/fr/install-guide-ubuntu/environment-networking.html

https://docs.openstack.org/mitaka/install-guide-ubuntu/launch-instance-networks-provider.html

https://stackoverflow.com/questions/36747239/what-is-the-difference-between-provider-network-and-self-service-network-in-open

http://docs.openvswitch.org/en/latest/faq/vxlan/


Signature : _____