

RAPPORT DU PROJET DE FIN D'ÉTUDE

AU SEIN DE LA SOCIÉTÉ

CBI

(COMPAGNIE BUREAUTIQUE ET INFORMATIQUE)

DU 12/01/2019 A 31/05/2019

Mise en place d'une solution SDN sur Cloud OpenStack

Préparé par : EL annab Hamza

Encadré par : Mme. Berramdan Loubna

TABLE DES MATIERES

I. INTRODUCTION GENERALE.....	9
II. PRESENTATION DE L'ENTREPRISE D'ACCUEIL.....	10
1. Organigramme General.....	10
2. Histoire De CBI.....	11
3. Métiers Et Services De CBI.....	12
3.1. Les services.....	12
3.2. Métiers de CBI.....	12
3.2.1. Télécom.....	12
3.2.2. Systèmes	13
3.2.3. Software.....	13
3.2.4. Sécurité	13
3.2.5. Editique.....	14
4. Solution Office – Toshiba.....	14
5. Solution Production Printing – Canon.....	14
6. Organigramme Direction Des Operations.....	15

III. CLOUD COMPUTING	15
1. Cloud Computing en un mot.....	15
2. Modèles de service	16
3. Modèles de déploiement	18
IV. APERÇU sur OPSENSTACK	18
1. OpenStack - la naissance	18
2. Qu'est-ce que OpenStack?	18
3. OpenStack - évolution	18
V. ARCHITECTURE OPENSTACK	19
1. Keystone	20
2. Glance	22
3. Neutron	23
4. Nova.....	24
5. Horizon	24
6. Cinder.....	26
7. Swift	26
VI. Software-Defined Networking.....	27
1. Une brève histoire de SDN.....	28
2. OpenFlow	29
3. OpenvSwitch	31
4. OpenDaylight et OpenStack.....	31

VII. Mise en œuvre pratique	32
1. Topologie.....	33
2. Déploiement d'OpenStack.....	34
2.1. Mise en place de l'environnement	39
2.2. Installation et configuration de Keystone, Glance, Nova, Neutron et Horizon..	43
2.3. Lancer une instance	44
3. Intégration OpenDaylight dans OpenStack.....	45

I. INTRODUCTION GENERALE

Dans le nouveau monde des technologies de l'information, l'infrastructure est en train de passer de la méthode traditionnelle dédiée à l'approche en Cloud moderne et dynamique. La virtualisation des serveurs a amorcé cette transition vers le Cloud et des facteurs tels que la mondialisation et l'externalisation dans lesquels diverses équipes ont besoin de collaborer en temps réel ont accéléré l'adoption d'applications basées sur le Cloud.

La virtualisation des serveurs donne à l'infrastructure Cloud une partie de cette flexibilité. Pour exploiter pleinement la puissance du Cloud computing, la mise en réseau doit être dynamique et évolutive. Le réseau défini par logiciel (SDN) et la virtualisation de la fonction réseau (NFV) sont deux technologies émergentes qui font actuellement fureur dans l'industrie des réseaux, en particulier le SDN. Et ils sont réputés offrir la flexibilité et l'agilité requises par le Cloud computing. Sur le côté de la plate-forme Cloud, OpenStack occupe également une place importante. Depuis son lancement en 2010, OpenStack est discrètement devenue l'une des plates-formes open source à la croissance la plus rapide pour les infrastructures Cloud d'entreprise.

Lorsqu'OpenStack et SDN progressent rapidement et deviennent des plates-formes d'innovation, il est important de comprendre les technologies clés à leur intersection. L'objectif de ce projet est d'obtenir des informations sur la technologie et la mise en œuvre d'OpenStack et du SDN et de voir dans quelle mesure ces applications logicielles open source peuvent fonctionner ensemble.

II. Présentation de L'entreprise d'accueil

CBI, Premier acteur du conseil au Maroc pour les professionnels, se positionne avec une approche « Producteur de Productivité » pour accompagner les grandes entreprises dans l'amélioration de leur performance. Depuis sa création en 1970, CBI a été guidée par une culture du changement et de l'innovation. Ses plus de 150 consultants se différencient par la synergie multi- compétences tant sur les volets métier que sur le volet technologique. L'offre solution de CBI intègre des activités complémentaires à savoir la bureautique, l'informatique, l'intégration systèmes et les télécommunications. Cette offre est constituée de produits de haute technologie leaders sur leurs marchés. Les équipes de CBI sont formées en permanence aux technologies les plus récentes et possèdent tous les atouts et l'enthousiasme pour vous accompagner dans vos projets pour plus de productivité.

1. Organigramme General

L'organigramme ci-dessous présente les différentes directions de CBI :

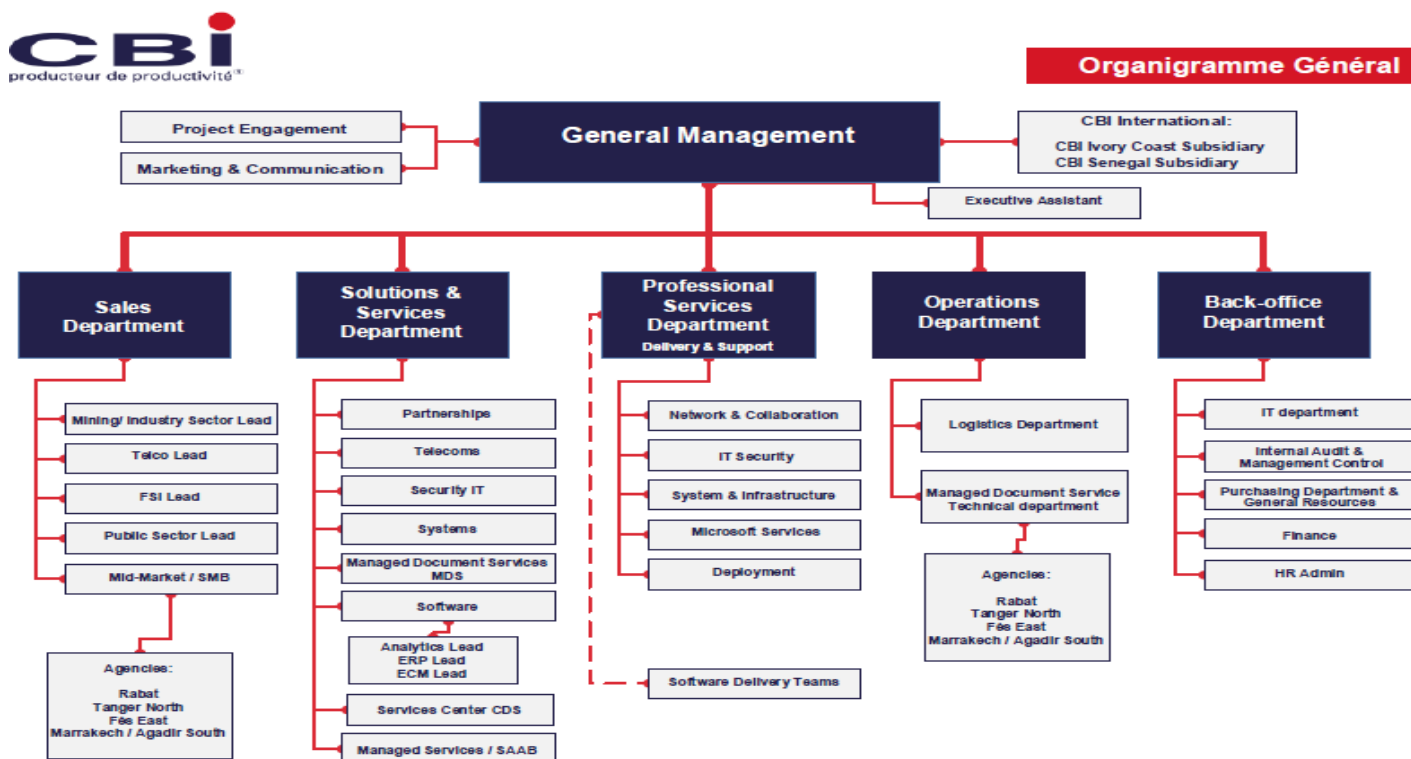


Figure 1. Organigramme au sein de direction CBI

2. Histoire De CBI

L'histoire de CBI se confond avec celle des technologies de l'information au Maroc et dans le monde. CBI a pu ainsi s'adapter en permanence aux différentes révolutions tant dans le secteur informatique que celui des télécommunications et comme ça initier dans notre pays les dernières évolutions technologiques. Ci-dessous quelques années phares qui marquent l'historique de CBI :

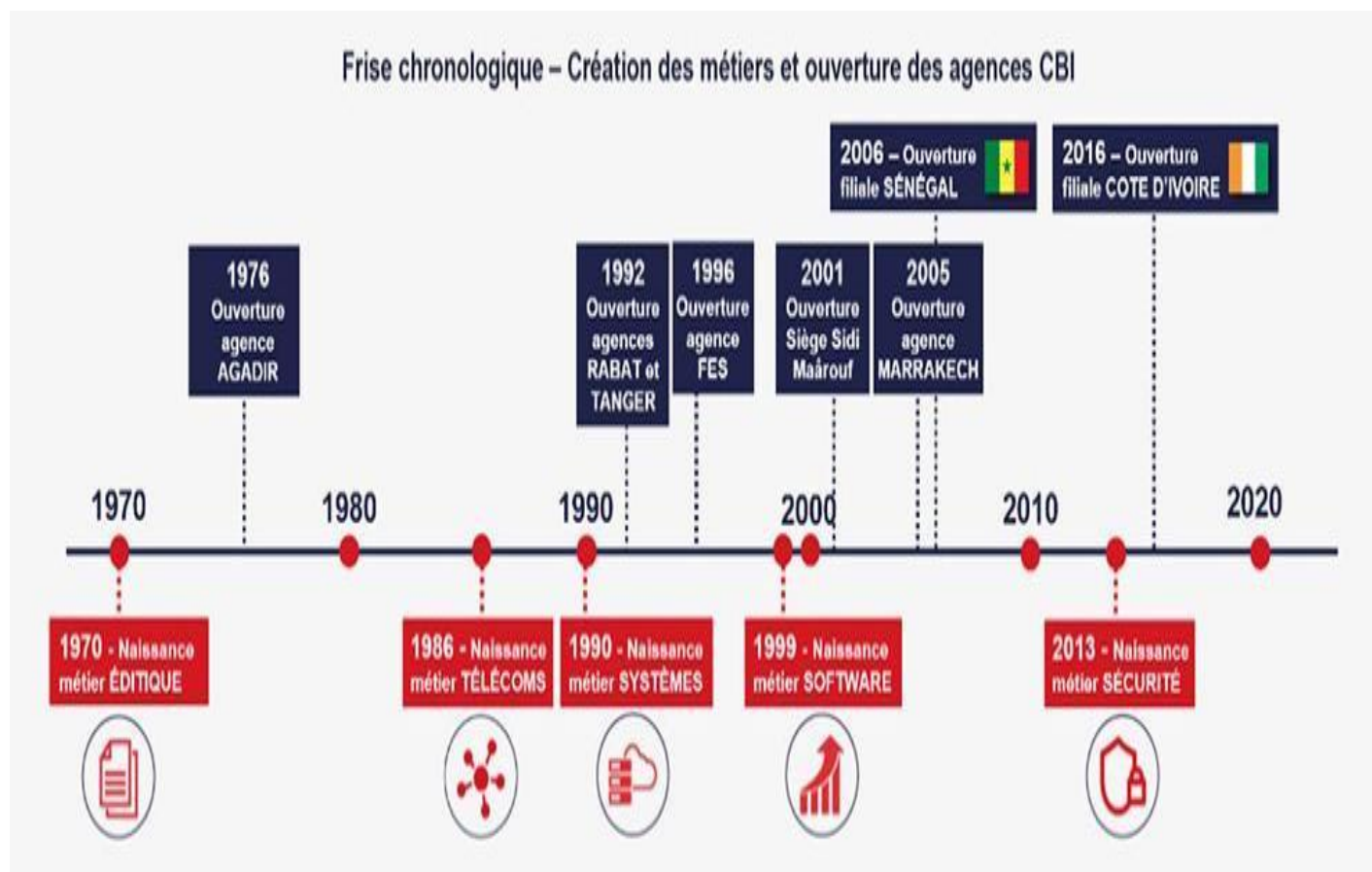


Figure 2. Chronologie de la création des métiers et agences CBI

3. Métiers Et Services De CBI

CBI se distingue par sa couverture globale des besoins d'Entreprise en matière de Systèmes d'Information (SI). Aujourd'hui encore cette culture lui permet d'être à la pointe de la convergence numérique et d'offrir aux entreprises nationales et africaines des solutions globales intégrant dans une synergie parfaite les compétences de ses métiers et ses services.

3.1 Les services

- Intégration projets
- Disponibilité et continuité
- Audit et assistance
- Outsourcing et services régies
- Formation
- Hébergement et offre Cloud

3.2 Métiers de CBI



3.2.1 Télécom

CBI TELECOMS s'est positionné comme le spécialiste dans la mise en place des solutions Réseau et Télécommunication. Par ses partenariats, CBI TELECOMS accompagne ses clients dans la mise en œuvre d'une véritable politique collaborative de la façon la plus efficiente possible. Précurseur dans la mise en œuvre de nombreuses technologies, le Pôle a su ainsi développer une expérience inédite dans toutes les étapes de transport de l'information, depuis son émission jusqu'à sa livraison et bien évidemment en prenant en compte les exigences des politiques de sécurité de ses clients.

3.2.2 Systèmes

CBI SYSTEMES a pour vocation de mettre à disposition de ses clients les meilleures Solutions répondant aux différents besoins suivants :

- Poste utilisateur : Poste de travail, périphériques, client léger...
- Serveurs : Consolidation, clustering...
- Stockage : Optimisation, Systèmes de sauvegarde, Système d'archivage
- Virtualisation : Serveur, Applications, Stockage, Poste utilisateur
- Datacenter : Armoires, Systèmes de câblage, accessoires, onduleurs.

3.2.3 Software

CBI SOFTWARE assure la mise en œuvre d'une infrastructure logicielle globale répondant à différentes problématiques de l'entreprise quant à la génération et la disponibilité de son information.

Les principales solutions qui se proposent sont :

- Customer Relationship Management (CRM): Traçabilité, archivage, Workflow
- Business Information Management: Dématérialisation des processus organisationnels et gestion des informations non structurées de l'entreprise (documents, mails, flux d'impression...)
- ERP: ERP Global, ERP Finances/Trésorerie.
- Décisionnel/ Business Intelligence
- EAM /GMAO
- Outil de support IT

3.2.4 Sécurité

CBI, à travers un écosystème de partenaires leaders dans les solutions de sécurité, propose une offre complète pour répondre à l'ensemble des risques et problématiques identifiés par les entreprises.

CBI accompagne ainsi ses clients dans la mise en place d'une politique de sécurité efficace visant à assurer un système d'information sûr et fiable, répondant aux exigences de la norme ISO 27 001, à savoir :

- Confidentialité,
- Traçabilité,
- Disponibilité,
- Intégrité,
- Authentification.

▪ Solution Office – Toshiba

Partenaire historique de Toshiba depuis plus de 48 ans, CBI propose une solution MDS – Managed Document Services « clés en main », économique et performante pour les impressions de bureau. De la définition de la politique documentaire la plus adaptée au structure du transfert de compétences, en passant par la fourniture de l'équipement, l'intégration et l'installation, la fourniture des consommables, la continuité de service et le remplacement de l'équipement, CBI prend intégralement en charge la gestion du parcs d'impression.

▪ Solution Production Printing – Canon

CBI a également développé un partenariat stratégique avec Canon pour les productions de documents et d'impressions moyens/hauts volumes.

Que ça soit éditeur, imprimeur de supports transactionnels ou promotionnels, ou encore le centre de reproduction interne d'une entreprise ou d'une administration, CBI propose des solutions qui s'adaptent aux besoins et aux impératifs de qualité et de rentabilité.

▪ Organigramme Direction Des Operations

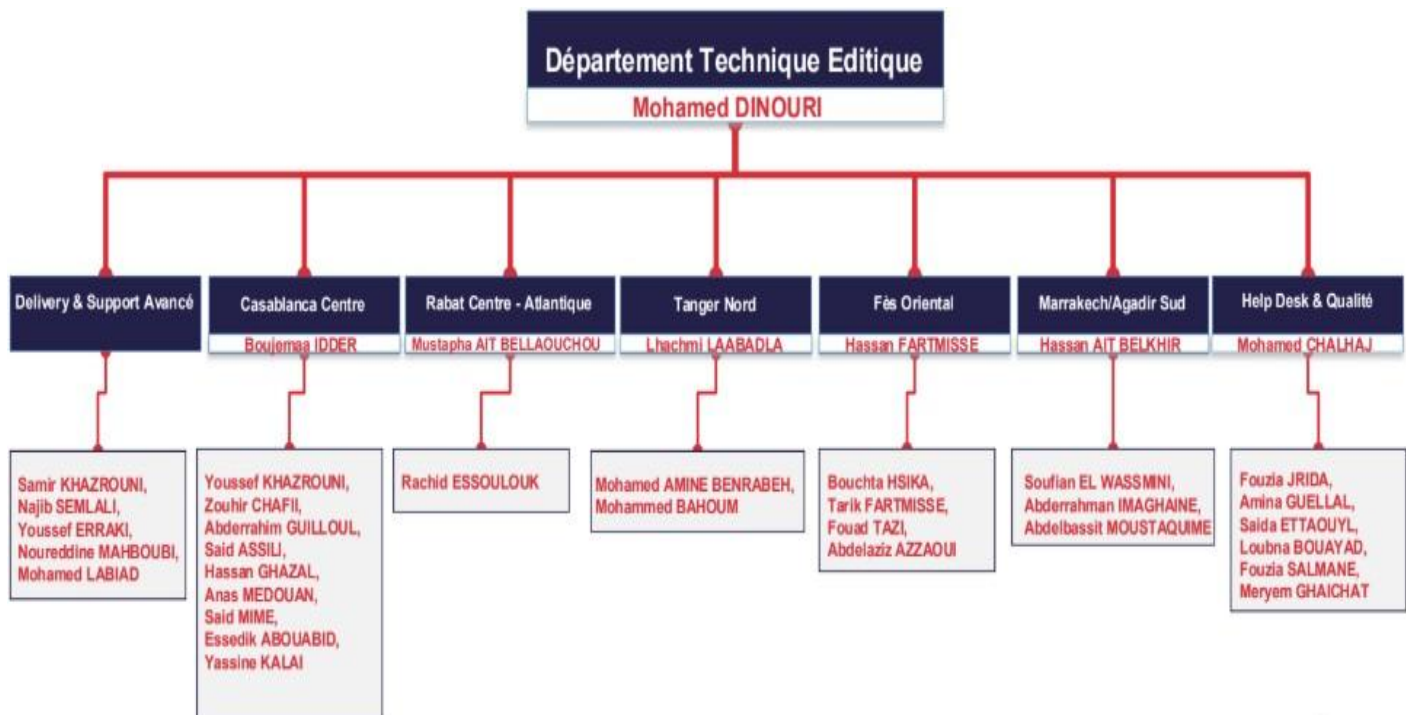


Figure 3. Organigramme De direction des opérations au sein de CBI

III. Cloud Computing

Ce chapitre est un aperçu sur les bases essentielles pour comprendre la signification de l'informatique en Cloud, son classement en fonction de modèles de service et de modèles de déploiement et la raison pour laquelle il joue un rôle important dans le secteur des technologies de l'information.

1. Cloud computing en un mot

Le Cloud computing est un modèle permettant un accès réseau pratique et sur demande à un pool partagé de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services) pouvant être rapidement mis en service et libérés avec un effort de gestion ou un service minimal.

Les cinq caractéristiques essentielles qui définissent un Cloud, à savoir :

- **Libre-service à la demande** : les utilisateurs finaux peuvent s'inscrire rapidement et obtenir les services souhaités sans les reports qui ont caractérisé l'informatique traditionnelle.
- **Accès réseau étendu** : les utilisateurs finaux peuvent accéder au service via des plateformes standard telles que des ordinateurs de bureau, des ordinateurs portables et des téléphones portables.
- **Regroupement de ressources** : plusieurs clients partagent un pool de ressources.
- **Élasticité rapide** : capacité à s'adapter à la demande des clients.
- **Service de mesure** : la facturation au compteur est livrée aux clients.

2. Modèles de service

La classification la plus courante utilise le modèle SPI (logiciel en tant que service, plate-forme en tant que service et infrastructure en tant que service). On les appelle souvent la pile d'informatique en nuage parce qu'elles sont superposées.

La figure 4 ci-dessous illustre une distinction claire entre SaaS, PaaS et IaaS, mais en réalité, les différences sont floues.



Figure 4. Pile de Cloud computing

SaaS (Logiciel en tant que service) fournit des applications logicielles sur Internet. On dit que c'est le modèle le plus mature. Tous les services fonctionnent sur l'infrastructure du fournisseur de Cloud. Les licences, la maintenance des applications, les mises à niveau logicielles et les correctifs de sécurité sont tous effectués par le fournisseur. Ce modèle implique un engagement significatif vis-à-vis d'un fournisseur de Cloud par rapport à IaaS et SaaS car il n'est pas simple de passer d'un fournisseur SaaS à un autre.

PaaS (Platform as a Service) est un service d'informatique en Cloud qui fournit aux développeurs des ressources pour développer, lancer, déployer et gérer des applications sans avoir à acheter, configurer et entretenir l'infrastructure sous-jacente. Le PaaS est analogue au SaaS, mais plutôt que de fournir des logiciels livrés sur le Web, il s'agit de la plate-forme sur laquelle les utilisateurs finaux créent des applications logicielles livrées sur le Web.

IaaS (L'infrastructure en tant que service) est le modèle de fourniture en nuage le plus répandu et le plus simple. Plutôt que de construire un centre de données, d'acheter des serveurs et du matériel de réseau, les clients louent des machines virtuelles, des systèmes de stockage, des réseaux et des systèmes d'exploitation entièrement externalisés par un fournisseur de Cloud. La frontière entre IaaS et PaaS est floue, mais PaaS offre généralement davantage de services destinés aux programmeurs, tels que des bibliothèques de code, des outils de développement, des middlewares, etc. Un autre facteur distinctif est qu'IaaS peut être obtenu en tant que public, privé ou une combinaison des deux : infrastructure hybride.

3. Modèles de déploiement

Le modèle SIP classe les services en fonction du type de contenu qu'ils offrent, mais les types de fournisseurs de services peuvent également être utilisés pour la classification. Dans un monde idéal, le consommateur devrait être indépendant des fournisseurs, il n'y a donc aucune raison de préférer l'un à l'autre. Cela étant dit, en ce qui concerne la sécurité, la gouvernance, la facturation et le règlement, les consommateurs doivent encore déterminer le type de modèle de déploiement fourni par un fournisseur de Cloud.

La figure 5 ci-dessous illustre quatre types de modèles de déploiement du Cloud computing, à savoir le Cloud public, le Cloud privé, le Cloud hybride et le Cloud communautaire.

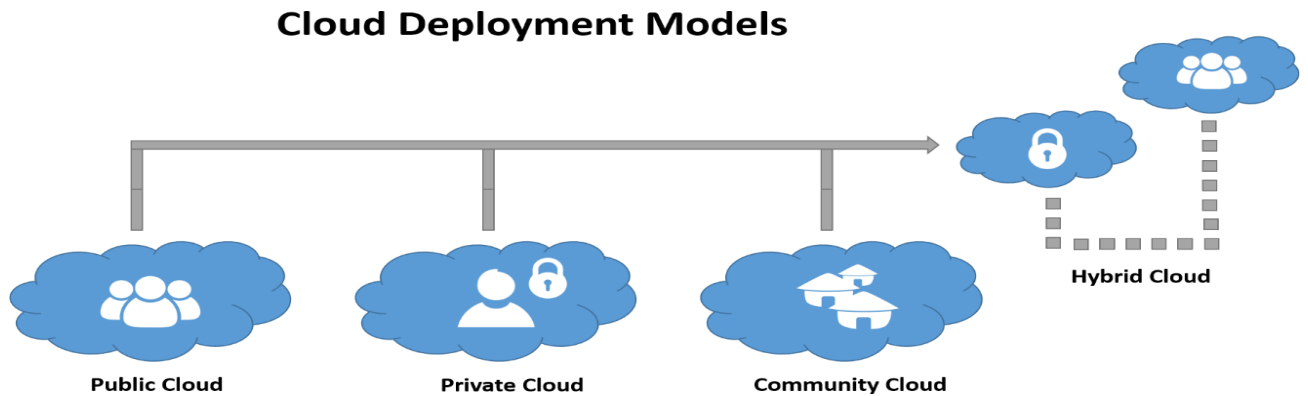


Figure 5. Modèles de déploiement du Cloud computing

Cloud public représente un véritable hébergement Cloud. Il fournit des services en Cloud sur un réseau pour un usage public et fournit des services et une infrastructure à divers clients. Le Cloud public est le meilleur choix pour les besoins professionnels dans lesquels la gestion des pics de charge, l'hébergement d'applications basées sur SaaS, l'utilisation d'une infrastructure provisoire pour le développement et le test des applications sont nécessaires.

Cloud privé est utilisé uniquement par une organisation car il est protégé par un pare-feu régi par un service informatique. Il offre un niveau de sécurité et de contrôle considérable, mais n'apporte pas grand-chose en termes de rentabilité, car l'entreprise doit toujours acheter, construire et entretenir sa propre infrastructure. Il peut être hébergé en interne ou en externe car il peut être détenu, géré et exploité par une organisation, un tiers ou une combinaison de ceux-ci. Le Cloud privé est un choix évident lorsque la sécurité, le contrôle et la confidentialité des données sont primordiaux pour une organisation.

Cloud hybride est la combinaison de Cloud public et privé qui sont liés mais qui restent toujours en tant qu'entités individuelles. En utilisant ce modèle, les clients peuvent tirer pleinement parti de la sécurité et de la confidentialité des données héritées d'un Cloud privé, tout en exploitant au mieux les coûts, en conservant les applications et les données partagées sur un Cloud public.

Cloud communautaire est partagé par plusieurs organisations, telles que des banques et des sociétés commerciales, avec les mêmes considérations de politique et de conformité, mais appartient à une communauté particulière. Ce modèle permet de réduire davantage les coûts par rapport à un Cloud privé, car il est partagé par un groupe plus important.

VI. Aperçue sur OpenStack

Ce chapitre un bref aperçu de la création du projet OpenStack, des personnes qui ont participé à la naissance d'OpenStack, de ce qu'est OpenStack et de son évolution.

1. OpenStack – la naissance

En 2009, le gouvernement des États-Unis avait demandé à la NASA d'exploiter les nouvelles technologies qui faciliteraient l'initiation du gouvernement ouvert, récemment adoptée. La NASA avait déjà développé NASA.net, une plate-forme technologique unifiée utilisée dans tous ses projets Web. Les développeurs de la NASA ont commencé à créer un ensemble de systèmes de stockage et de calcul génériques, à la demande, pilotés par API, à savoir Infrastructure as a Service, bien que l'informatique en nuage en soit encore à son stade de développement. L'équipe de NASA.net, qui a ensuite appelé leur projet NASA Nebula, a imaginé une chance importante de devenir un fournisseur de Cloud computing et a décidé de se concentrer sur la création d'un contrôleur de calcul Cloud open source.

2. Qu'est-ce que OpenStack

OpenStack est une plate-forme open source qui nous permet de créer un Cloud IaaS (Infrastructure as a Service) fonctionnant sur du matériel standard. C'est une couche de contrôle située au-dessus de toutes les couches virtualisées et qui offre un moyen cohérent d'accéder aux ressources, que la technologie d'hyperviseur utilisée en dessous soit KVM, Xen ou VMware. OpenStack combine un grand nombre d'outils Open Source interdépendants, appelés projets, pour fournir divers composants à une solution d'infrastructure Cloud. Six des projets qui gèrent les principaux services d'informatique en Cloud, notamment l'informatique, la mise en réseau, le stockage, l'identité et l'image, peuvent être associés à une douzaine de projets supplémentaires pour fournir le nuage souhaité.

OpenStack est écrit principalement en Python et disponible librement sous Apache Licence 2.0. Chaque service OpenStack fournit une API REST afin que toutes les ressources (calcul, stockage et mise en réseau) puissent être gérées via un tableau de bord. Il donne aux administrateurs le contrôle tout en permettant aux utilisateurs de provisionner des ressources via une interface Web ou un client à ligne de commande.

V. Architecture d'OpenStack

En bref, OpenStack est génial, mais on peut se demander quelle est la technologie derrière le projet OpenStack. Ce chapitre aborde de manière approfondie l'architecture OpenStack, ses services de base, leur nature et leur fonctionnement.

1. Keystone

Keystone est le service OpenStack Identité qui fournit des services d'authentification et d'autorisation à travers toute l'infrastructure du Cloud. Du point de vue architectural, Keystone est le service le plus simple de la composition OpenStack.

Keystone conserve un catalogue de services et de points de terminaison de tous les services OpenStack. Tous les services ont des points de terminaison API différents. C'est un avantage car un utilisateur final a seulement besoin de connaître l'adresse de Keystone pour interagir avec le Cloud

```
[root@dgnode3 ~]# source openrc
[root@dgnode3 ~]# openstack endpoint list
```

ID	Region	Service Name	Service Type	Enabled	Interface	URL
ce0f81cdd74341469736f2aa0a53085b	None	keystone	identity	True	public	http://dgnode:5000/v3
0c469862653e46c481afacee66d388d3	None	keystone	identity	True	internal	http://dgnode:35357/v3
a452badcc5714bb4b48c52930361c909	None	keystone	identity	True	admin	http://dgnode:35357/v3
fa3d45ebec4d49e982b3a3220587bb7d	None	swift	object-store	True	public	http://dgnode:8080/v1/AUTH_%(tenant_id)s
4cef3af47d6f4356a1125a868b6ee96f	None	swift	object-store	True	internal	http://dgnode:8080/v1/AUTH_%(tenant_id)s
2d6f2c7e0294425dbf6b30224090f73c	None	swift	object-store	True	admin	http://dgnode:8080

Figure 6. Catalogue Keystone

La figure 6 illustre comment Keystone décrit les services OpenStack, y compris lui-même. Il connaît tous les ID, régions, noms de service, types de service, interfaces et URL.

Keystone utilise un certain nombre de mécanismes d'authentification, notamment un nom d'utilisateur / mot de passe, basé sur un jeton. De plus, il prend également en charge l'intégration d'un répertoire principal existant, tel que LDAP ou PAM.

2. Glance

Glance est le service OpenStack Image qui sert de registre d'images. Une fois que nous sommes authentifiés et autorisés, nous devons tout d'abord disposer d'une image disque pour lancer une machine virtuelle. Cependant, installer un système d'exploitation sur chaque machine est fastidieux. Le Cloud computing a simplifié la procédure en créant un registre contenant des images de disque préinstallées. Glance est ce registre dans OpenStack

La figure 7 ci-dessous illustre l'architecture de Glance.

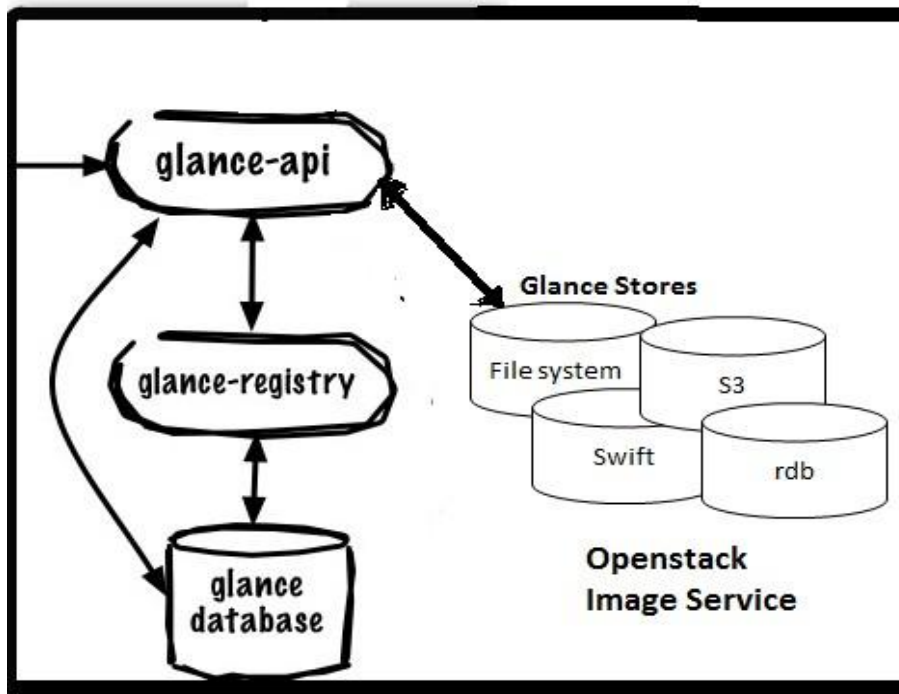


Figure 7. Glance OpenStack

Glance comprend les composants suivants:

- **glance-api** accepte les appels d'API pour les images d'utilisateurs finaux ou d'autres services.
- **Glance-registry** stocke, traite et récupère les métadonnées d'image telles que la taille, le type, le format, etc.
- **La base de données Glance** stocke les métadonnées de l'image. Il prend en charge plusieurs moteurs tels que MySQL, SQLite et MongoDB.
- **les backends** de stockage Glance prennent en charge les systèmes de fichiers normaux. Ils peuvent être Swift ou Ceph RBD.

3. Neutron

Neutron, auparavant connu sous le nom de Quantum avant la sortie de Havana, est un projet OpenStack qui fournit un réseau en tant que service (NaaS). Avant Quantum, OpenStack disposait d'une architecture réseau plate et simple contrôlée par Nova Network, un sous-composant de Nova. Neutron permet aux locataires de créer des topologies de réseau virtuel avancées incluant des services tels que des pare-feux, des équilibreurs de charge, des réseaux privés virtuels, etc. C'est l'un des composants les plus complexes des projets OpenStack, car il repose sur les concepts de réseau de base.

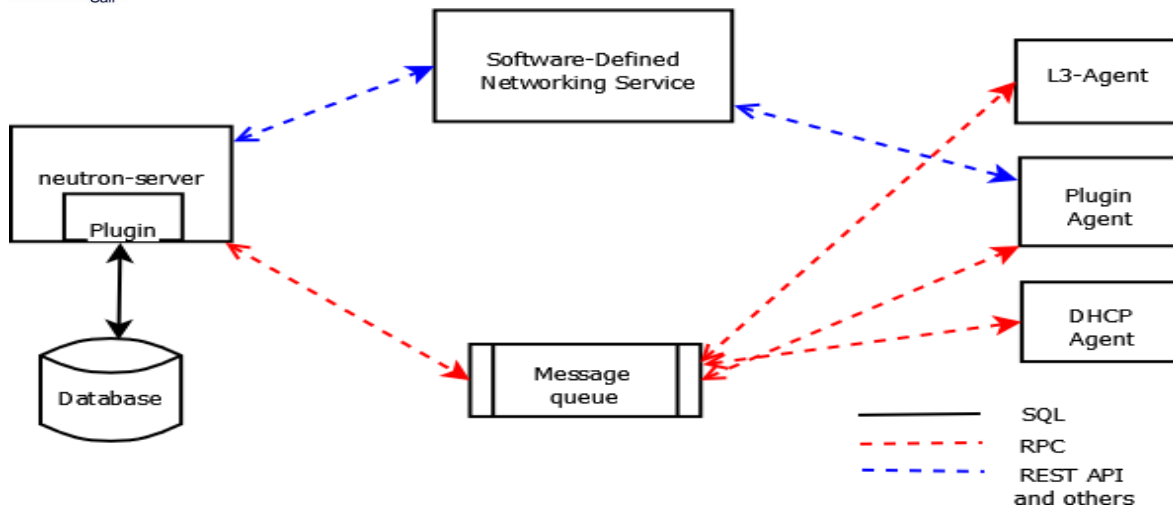


Figure 8. Neutron OpenStack

Le processus principal du service de réseau OpenStack s'appelle neutron-server. Il s'agit d'un démon Python qui expose l'API Neutron et transmet les demandes des clients hébergés à une suite de plug-ins pour un traitement supplémentaire.

Neutron fournit des réseaux, des sous-réseaux et des routeurs sous forme d'abstractions d'objets. Chacune d'entre elles a une fonctionnalité qui imite sa contrepartie physique.

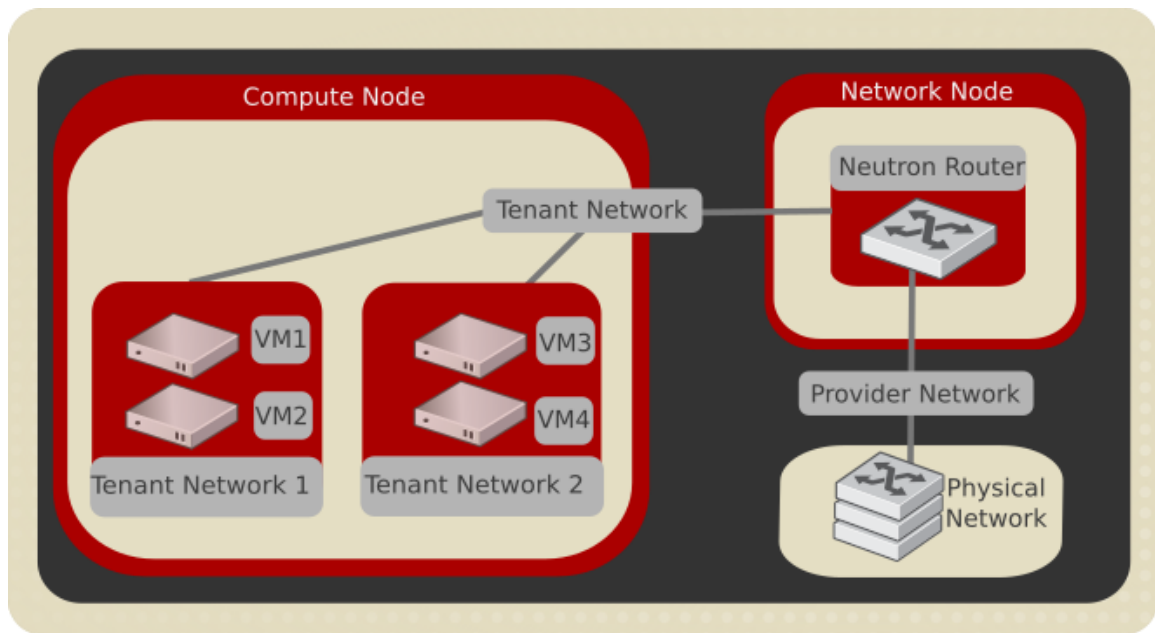


Figure 9. Réseaux de fournisseurs de neutrons et réseaux de locataires

ML2 (Modular Layer 2) fourni avec OpenStack est le plugin principal le plus important. Il prend en charge une grande variété de technologies de couche 2 et permet aux technologies de plusieurs fournisseurs de coexister. Avant que ML2 ne soit impliqué dans la grande image d'OpenStack, Neutron était limité à l'utilisation d'un seul plug-in à la fois. Le but de ML2 est de remplacer et de déconseiller deux plug-ins monolithiques: linuxbridge et openvswitch. Leurs agents de niveau 2 continuent toutefois de travailler avec le niveau 2

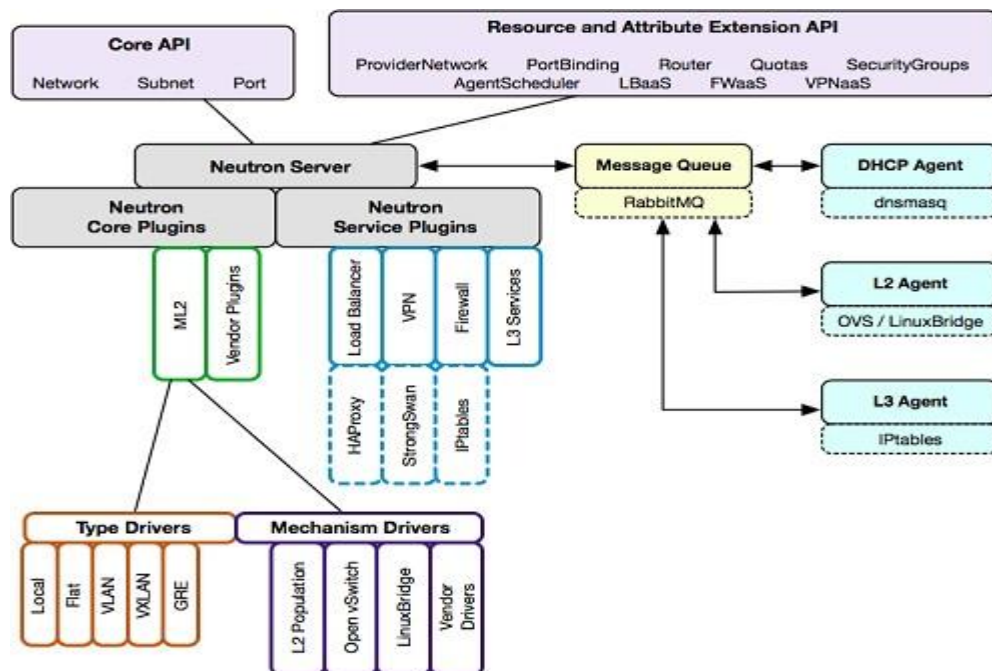


Figure 10. Architecture Modular Layer 2 (ML2)

4. Nova

Nova est le service de calcul et le composant le plus original d'OpenStack. Du point de vue architectural, Nova est le composant le plus compliqué et le plus distribué d'OpenStack.

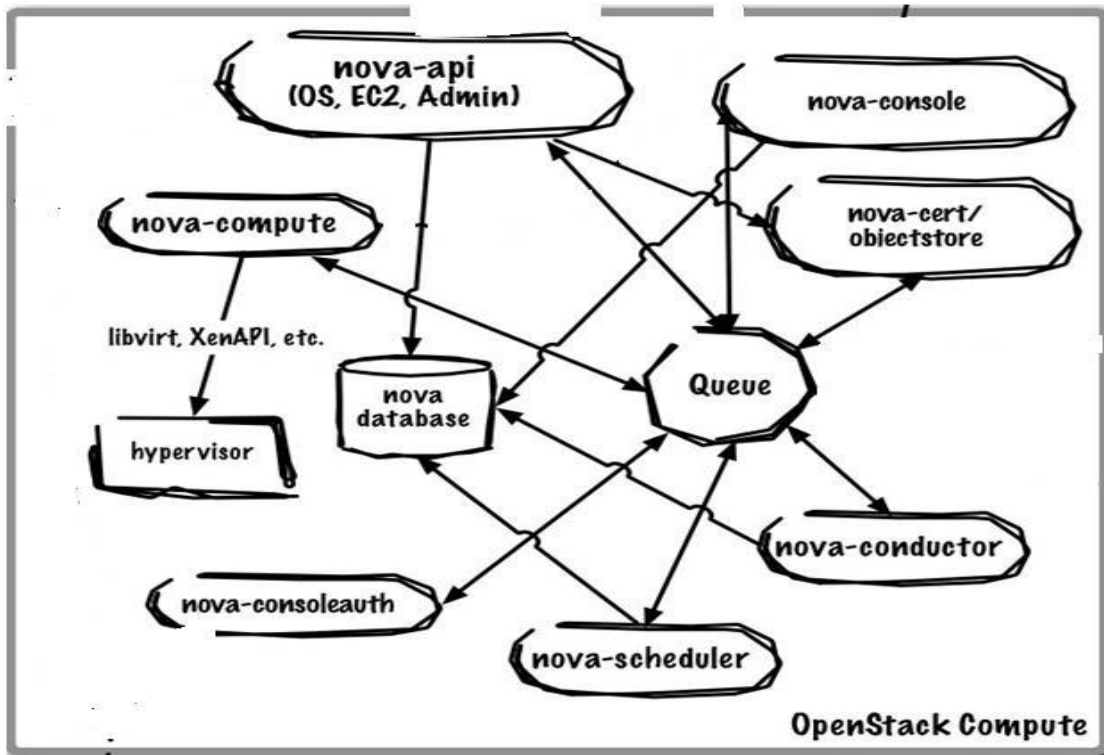


Figure 11. Nova OpenStack

Pour transformer la demande d'API d'un utilisateur final en une machine virtuelle en cours d'exécution, un grand nombre de processus, comme illustré à la figure 11, doivent coopérer. Comme identifié par Gupta (2013), ces processus sont:

- **nova-api** est un service API RESTful qui accepte les demandes des utilisateurs finaux et y répond
- **nova-cert** assure la gestion des certificats.
- **nova-compute** est principalement un démon de travail qui crée et met fin aux instances. Cela garantit que l'état des nouvelles instances est maintenu dans la base de données.
- **nova-console** permet aux utilisateurs finaux d'accéder à leur console d'instances via un proxy.
- **nova-consoleauth** fournit une authentification pour les consoles Nova.
- **nova-conductor** est un démon de serveur qui permet à OpenStack de fonctionner sans que les nœuds de calcul accèdent à la base de données. Il implémente

conceptuellement une nouvelle couche par-dessus nova-compute.

- **nova-scheduler** prend la demande d'une instance dans la file d'attente et détermine son emplacement, en particulier sur l'hôte de calcul sur lequel elle doit être exécutée.
- **la file d'attente** fournit un concentrateur central pour la transmission de messages entre démons. La file d'attente est généralement implémentée avec RabbitMQ.
- **La base de données Nova** stocke la plupart des états d'infrastructure d'exécution et d'exécution.

5. Horizon

Horizon est le service OpenStack Dashboard qui fournit une interface utilisateur Web aux services OpenStack, tels que Nova, Keystone, Neutron, comme illustré à la figure 12.

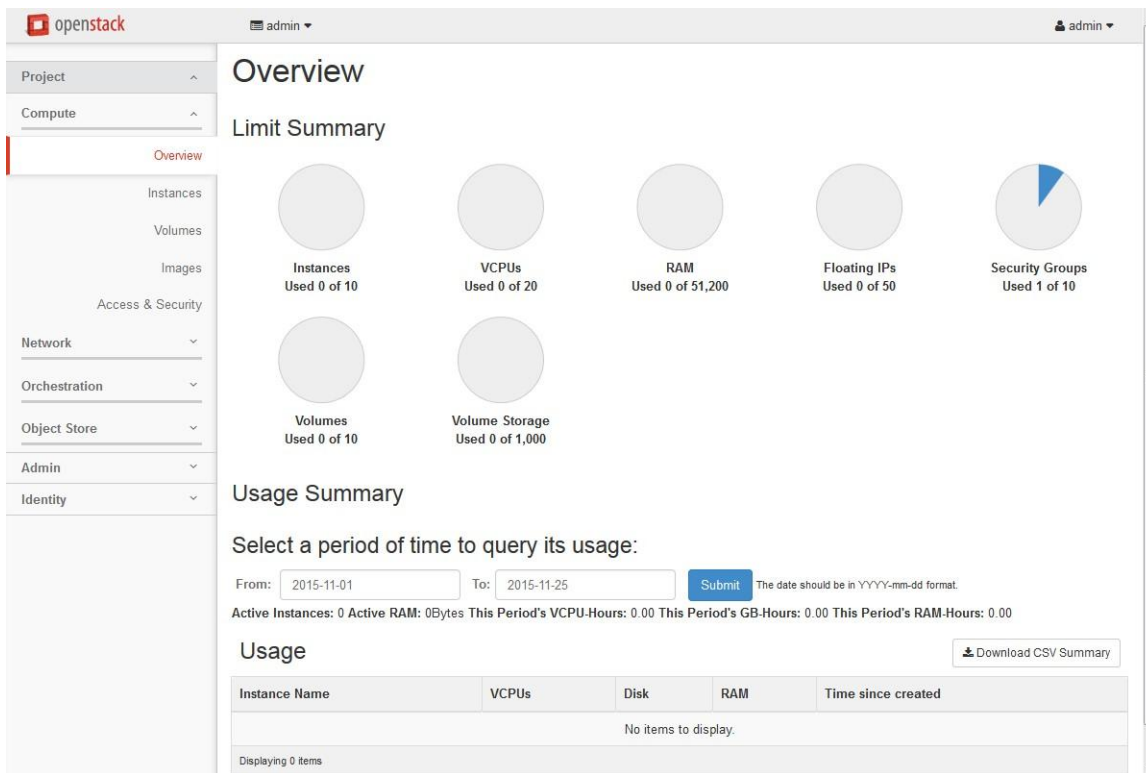


Figure 12. Horizon OpenStack

À l'origine, Horizon a été lancé en tant qu'application unique pour gérer le projet de calcul d'OpenStack. Par conséquent, il ne manquait plus qu'un ensemble de vues, de modèles et d'appels d'API. Par la suite, il a progressivement pris en charge plusieurs API et projets OpenStack. Il a donc été structuré de manière rigide en groupes de tirets et de systèmes.

6. Cinder

Cinder est le service OpenStack Block Storage. C'est le remplacement du service nova-volume depuis la sortie de Folsom.

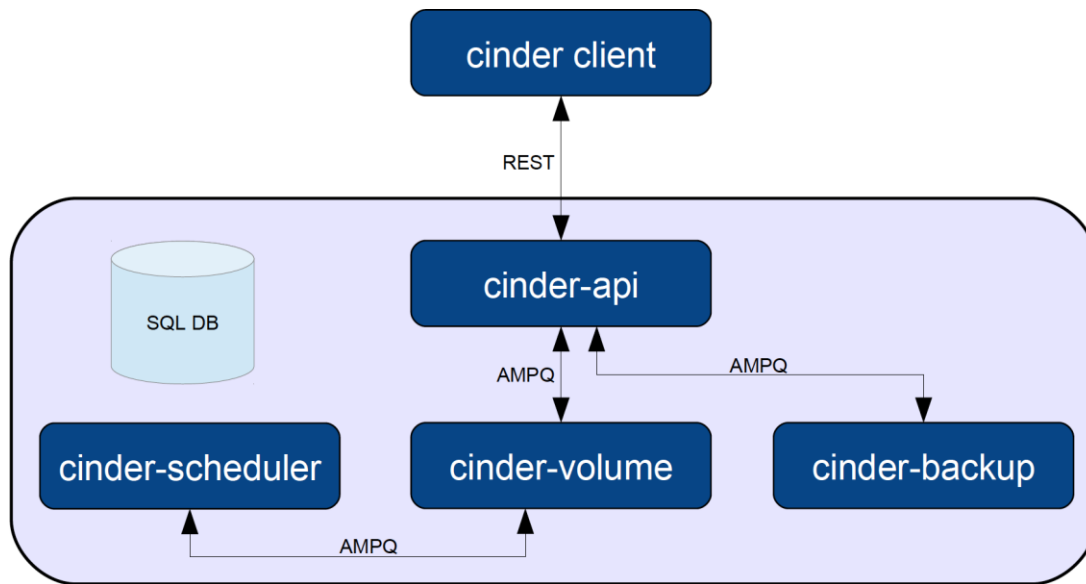


Figure 13. Cinder OpenStack

La figure 13 montre la structure de Cinder. Comme indiqué par Hui (2013), Cinder comprend les quatre démons suivants et deux autres composants:

- **cinder-api** gère, répond et place les demandes dans la file d'attente des messages.
- **cinder-scheduler** lit les demandes de la file de messages, les planifie et les achemine vers le service de volume approprié. Il peut s'agir d'une planification à tour de rôle simple ou plus sophistiquée via l'utilisation d'un planificateur de filtres, en fonction de sa configuration.
- **cinder-volume** gère l'interaction avec les périphériques de stockage en mode bloc.
- **Cinder-Backup** offre la possibilité de sauvegarder un volume Cinder sur diverses cibles de sauvegarde.
- **La base de données** fournit des informations d'état.
- **Le serveur RabbitMQ** fournit la file de messages AMQP.

7. Swift

Swift est le service OpenStack Object Storage. Il fournit un stockage d'objets distribués redondant et évolutif. Distribué signifie que chaque donnée est répliquée sur un cluster de nœuds de stockage. Swift est principalement utilisé pour stocker et récupérer des objets BLOB. Il s'agit de données statiques telles que des images, vidéos, photos, courriels, fichiers, sauvegardes et archives de machines virtuelles.

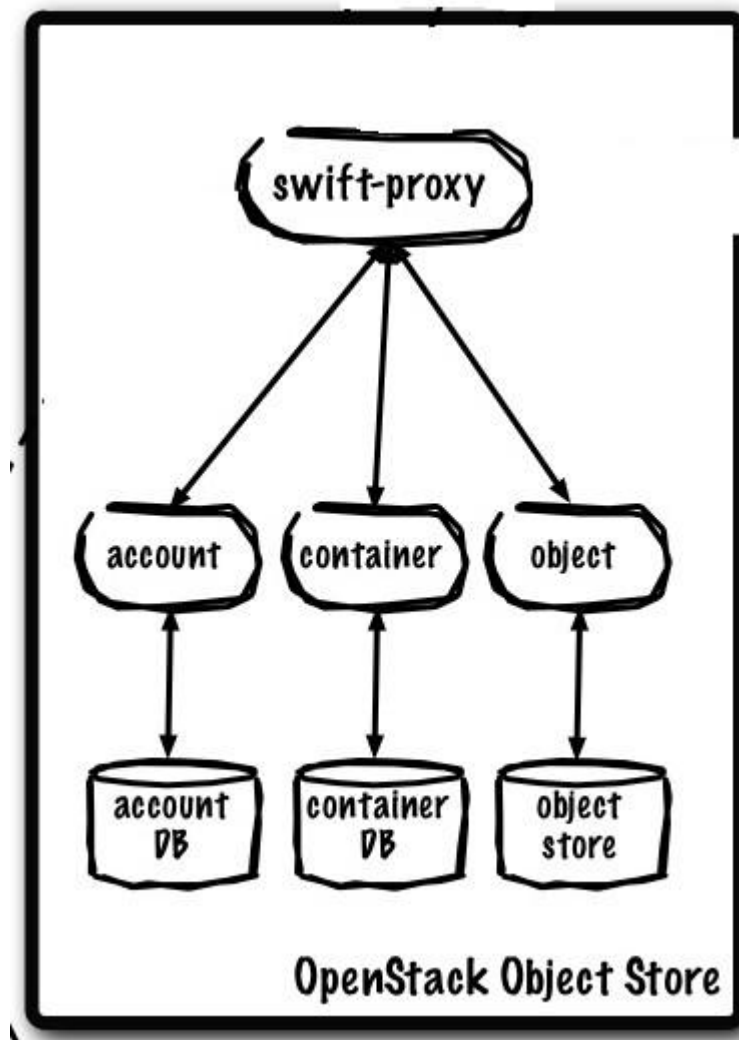


Figure 14. Swift OpenStack

Selon Gupta (2013), les principales composantes de Swift comprennent swift-proxy, swift-account, swift-container et swift-object, comme le montre la figure 14.

- **swift-proxy** expose l'API publique, fournit une authentification et est responsable du traitement et du routage des demandes entrantes.
- **swift-object** stocke, récupère et supprime des objets.
- **swift-account** est responsable de la liste des conteneurs.
- **swift-container** gère les listes d'objets.

7. Software-Defined Networking

Ce chapitre explique le développement de réseaux définis par logiciel et Il présente également plusieurs technologies liées au SDN, notamment OpenFlow, Open vSwitch et OpenDaylight. L'une des parties les plus importantes de ce chapitre est de savoir comment et où le SDN s'intègre dans l'image OpenStack.

1. Une brève histoire de SDN

Les réseaux définis par logiciel sont le fruit des travaux de recherche menés en 2004 dans le cadre de la recherche d'un nouveau paradigme de gestion de réseau. Le travail initial a été construit en 2008 par différents groupes. La startup Nicira, rachetée par VMware, a créé un système d'exploitation réseau appelé NOX. Parallèlement, Nicira a collaboré avec des équipes de l'Université de Stanford pour créer l'interface du commutateur OpenFlow. En 2011, l'Open Networking Foundation, l'organisme de normalisation de facto de l'espace SDN, bénéficiait déjà d'un large soutien de la part de nombreux grands noms de l'industrie des réseaux, tels que Cisco, Juniper Networks, Hewlett-Packard, Dell, Broadcom, IBM, etc. puis plus tard, Google, Verizon, Yahoo, Microsoft, Deutsche Telekom, Facebook et NTT.

La mise en réseau définie par logiciel est une architecture émergente dynamique, gérable, économique et adaptable, ce qui la rend idéale pour la nature dynamique à large bande passante des applications actuelles. Cette architecture dissocie les fonctions de contrôle de réseau et de transfert, ce qui permet à la commande de réseau de devenir directement programmable et d'abrèger l'infrastructure sous-jacente pour les applications et les services réseau. Le protocole OpenFlow est un élément fondamental pour la création de solutions SDN, l'architecture SDN se compose de trois couches, comme illustré à la figure 17

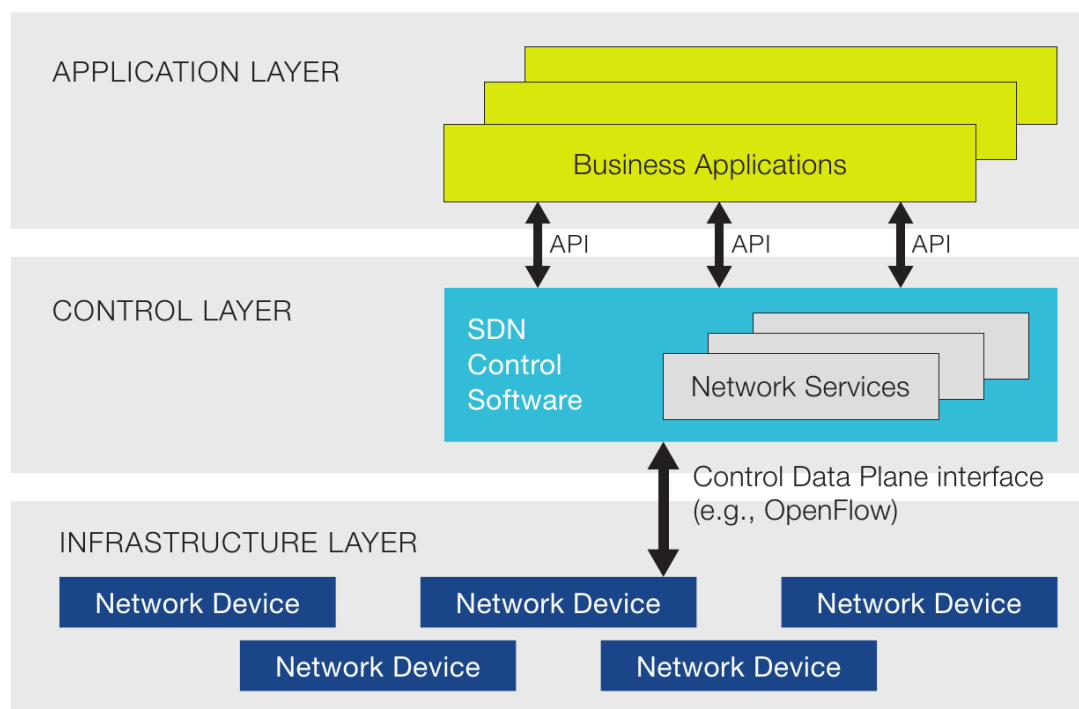


Figure 17. Architecture SDN

2. OpenFlow

OpenFlow n'est qu'une option parmi plusieurs protocoles de contrôle dans SDN, mais c'est le principal. OpenFlow est un protocole réseau programmable conçu pour gérer et diriger le trafic entre les routeurs et les commutateurs de plusieurs fournisseurs. OpenFlow sépare la programmation des routeurs et des commutateurs du matériel sous-jacent.

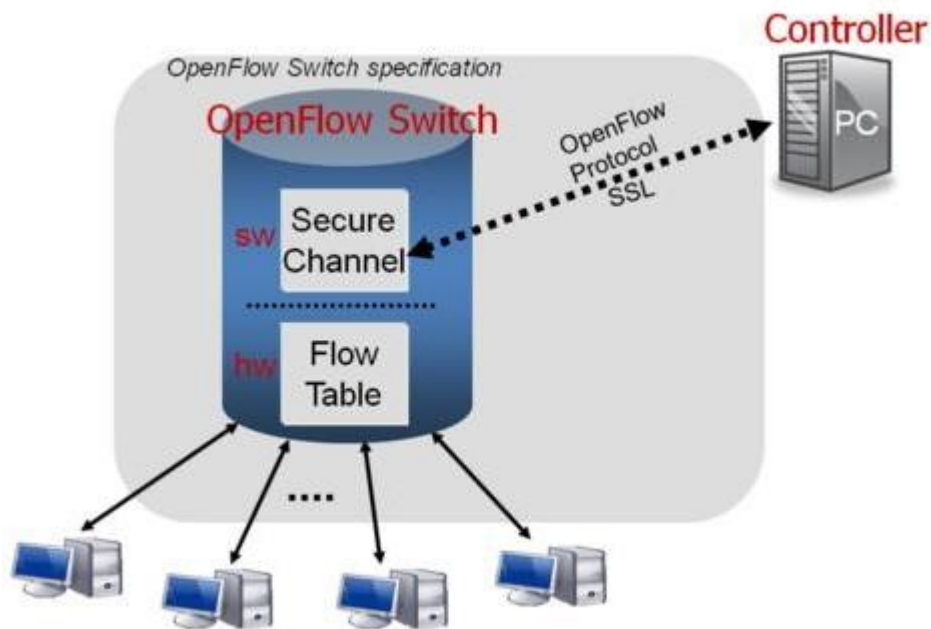


Figure 18. Architecture OpenFlow

L'architecture OpenFlow comprend trois parties : le plan de données construit par les commutateurs OpenFlow, le plan de contrôle contenant les contrôleurs OpenFlow et un canal de contrôle sécurisé reliant ces deux plans

Un commutateur OpenFlow est un dispositif de transmission de base qui transfère les paquets en fonction de son tableau de flux. La table de flux contient un ensemble d'entrées de table de flux, chacune d'entre elles étant constituée d'une règle (champs de correspondance), d'une action (instruction) et de stats (compteurs), comme illustré à la figure 19

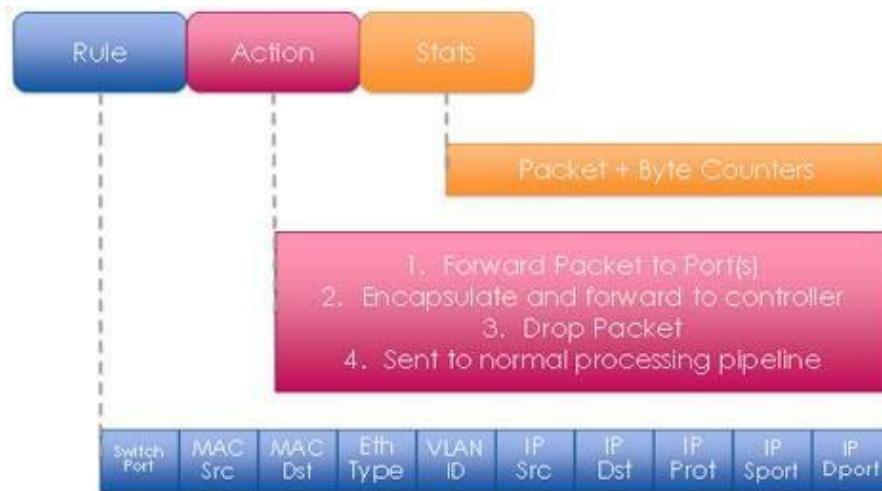


Figure 19. Entrée du tableau de flux

3. OpenVswitch

Open vSwitch est l'un des trois commutateurs virtuels les plus populaires aux côtés du commutateur virtuel VMware et du Cisco Nexus 1000V. Nicira, racheté par la suite par VMware, a créé Open vSwitch pour répondre aux besoins de la communauté open source car il n'existait aucune offre de commutateur virtuel riche en fonctionnalités conçue pour un hyperviseur basé sur Linux, tel que KVM et XEN. OVS est rapidement devenu le commutateur virtuel de facto pour les environnements XEN. Il joue désormais un rôle important dans le projet OpenStack et joue un rôle de premier plan dans les environnements SDN.

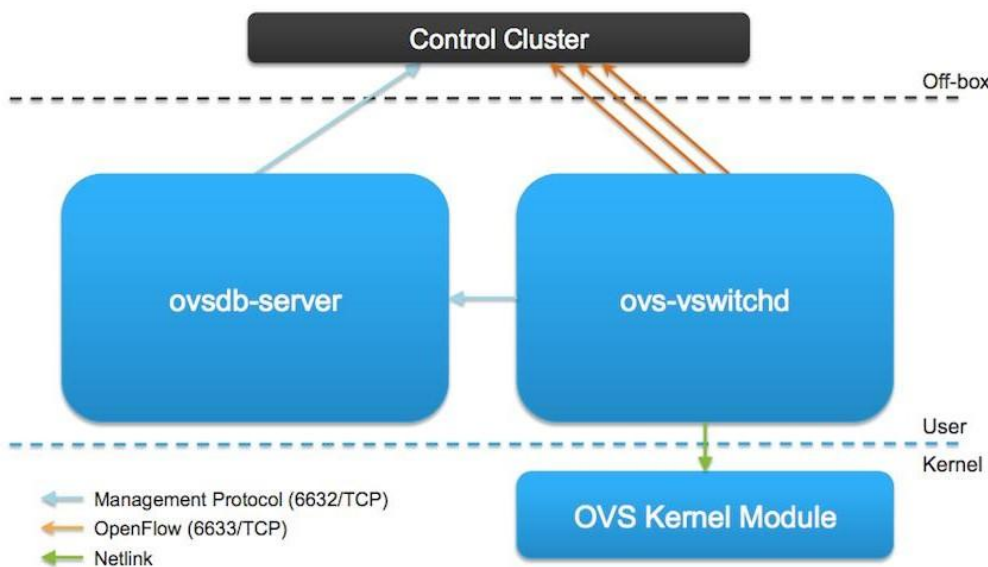


Figure 20. Composants vSwitch ouverts

La figure 20 illustre les composants OVS et leurs interactions. OVS est composé de trois composantes Principales:

- **ovs-vswitchd** est le démon Open vSwitch responsable de la logique de transmission, de la configuration à distance et de la visibilité
- **ovsdb-server** est le serveur de base de données Open vSwitch.
- **OVS Kernel Module** est le module du noyau responsable de la recherche, de la modification, du transfert et de l'encapsulation / décapsulation de tunnels.

4. OpenDaylight et OpenStack

Le projet OpenDaylight, annoncé en 2013, est un projet SDN open source hébergé par la Linux Foundation. Le projet est issu du mouvement SDN et son objectif est de faire progresser l'adoption du SDN et de créer la base d'un NFV fort.

Le contrôleur OpenDaylight, renommé OpenDaylight Platform, prend en charge le protocole OpenFlow et d'autres normes SDN ouvertes. Il expose les API ouvertes nord-utilisées qui sont utilisées par les applications pour collecter des informations sur le réseau, exécuter des algorithmes pour effectuer des analyses et créer de nouvelles règles sur l'ensemble du réseau.

La figure 21 ci-dessous illustre l'architecture de Beryllium, la quatrième version d'OpenDaylight.

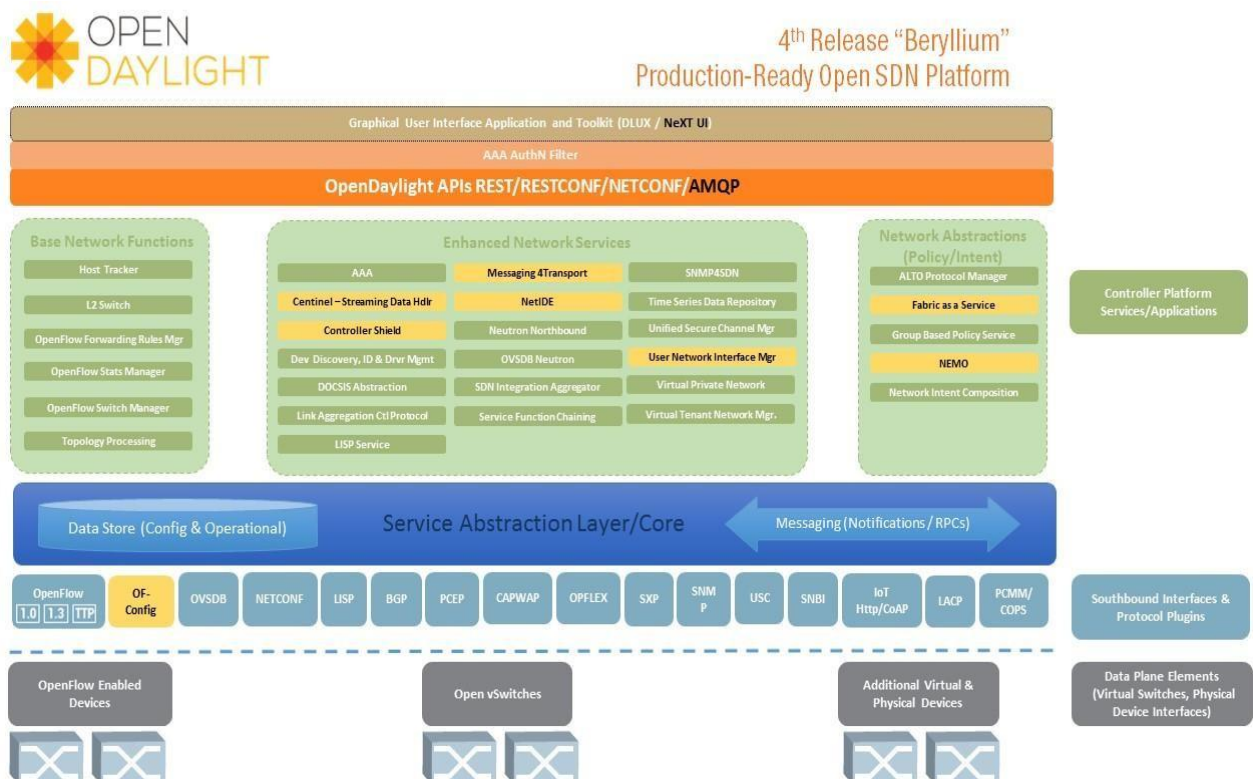


Figure 21. OpenDaylight Beryllium

Comme le montre la figure 21, de nombreuses fonctionnalités de Karaf sont prises en charge dans la version de Beryllium, telles que AAA, BGP, BMP, DLUX, FaaS, LACP, NETCONF, OVSDB, CONFIG (protocole de configuration OpenFlow, SNMP et VTN. ODL prend en charge une architecture en couches:

- **L'interface Northbound** fournit un riche ensemble d'API. Ces API REST sont principalement destinées à l'intégration avec des plates-formes Cloud telles qu'OpenStack. Ils peuvent également être utilisés pour créer une interface graphique pour ODL.
- **La couche de plate-forme de contrôleur** est chargée d'exploiter le modèle de données SAL et de fournir des fonctionnalités SDN fondamentales et des fonctions de mise en réseau telles que la topologie, la surveillance des performances, la gestion des commutateurs physiques et virtuels et la gestion des ARP. La couche de plate-forme de contrôleur relie les interfaces allant du nord aux interfaces sud et gère les API REST exposées. Il prend également en charge une fonctionnalité spécifique à chaque cas, qui s'avère être une fonctionnalité utile lors de l'intégration à OpenStack
- **SAL** est la couche la plus importante de l'architecture car son objectif principal est de cartographier un ensemble diversifié de technologies de mise en réseau.

La figure 22 ci-dessous montre un diagramme complet des blocs de construction de l'intégration OpenStack et OpenDaylight. L'idée est essentiellement que le plug-in ML2 de Neutron interagisse avec l'application OVSDB Neutron d'ODL, qui commande OVS à l'aide du protocole OVSDB ou du protocole OpenFlow. OVS prend en charge OpenFlow 1.0 et OpenFlow 1.3, et ce dernier prend en charge les fonctionnalités multi-tables qui optimisent le nombre de tunnels nécessaires entre OpenvSwitches.

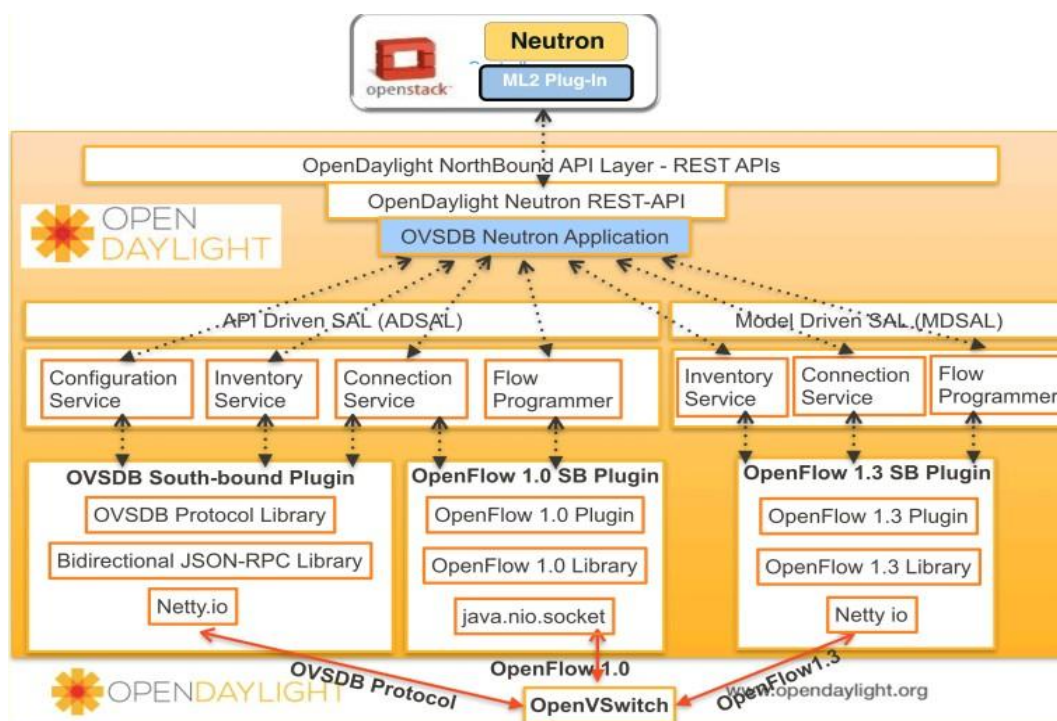


Figure 22. Blocs de construction impliqués dans l'intégration OpenStack et OpenDaylight

VII. Mise en œuvre Pratique

Ce chapitre montre comment un Cloud OpenStack à l'état de concept est construit avec OpenvSwitch en tant que plug-in Neutron ML2 principal. Et ensuite, OpenDaylight sera intégré à ce Cloud OpenStack pour créer un Cloud basé sur SDN.

1. Topologie

Pour un environnement de validation de concept, je construis un Cloud OpenStack avec quatre nœuds: un nœud de contrôleur, un nœud de réseau et deux nœuds de calcul. Les nœuds de contrôleur et de réseau sont des machines virtuelles s'exécutant sur ESXi, tandis que les nœuds de calcul sont des serveurs nus. Tous utilisent Ubuntu 14.04 comme système d'exploitation.

Le Cloud OpenStack contient la plupart de ses services principaux: Keystone, Glance, Nova, Neutron à l'aide du plug-in ML2 avec Open vSwitch et Horizon. Puisqu'il ne s'agit que d'un prototype, OpenStack est configuré pour utiliser des disques locaux pour les instances au lieu de Swift, Cinder ou Ceph; il n'a pas non plus de chaleur (orchestration) ni de ceilomètre (télémétrie).

Le nœud de contrôleur exécute les services Keystone et Glance, les parties de gestion de Nova et Neutron, Horizon, ainsi que des services de support tels qu'une base de données SQL, une file d'attente de messages et NTP. Le nœud de réseau dédié exécute les services Neutron. Les nœuds de calcul exécutent la partie d'hyperviseur de Nova qui gère les instances et les parties de gestion de Neutron, et utilisent l'hyperviseur KVM.

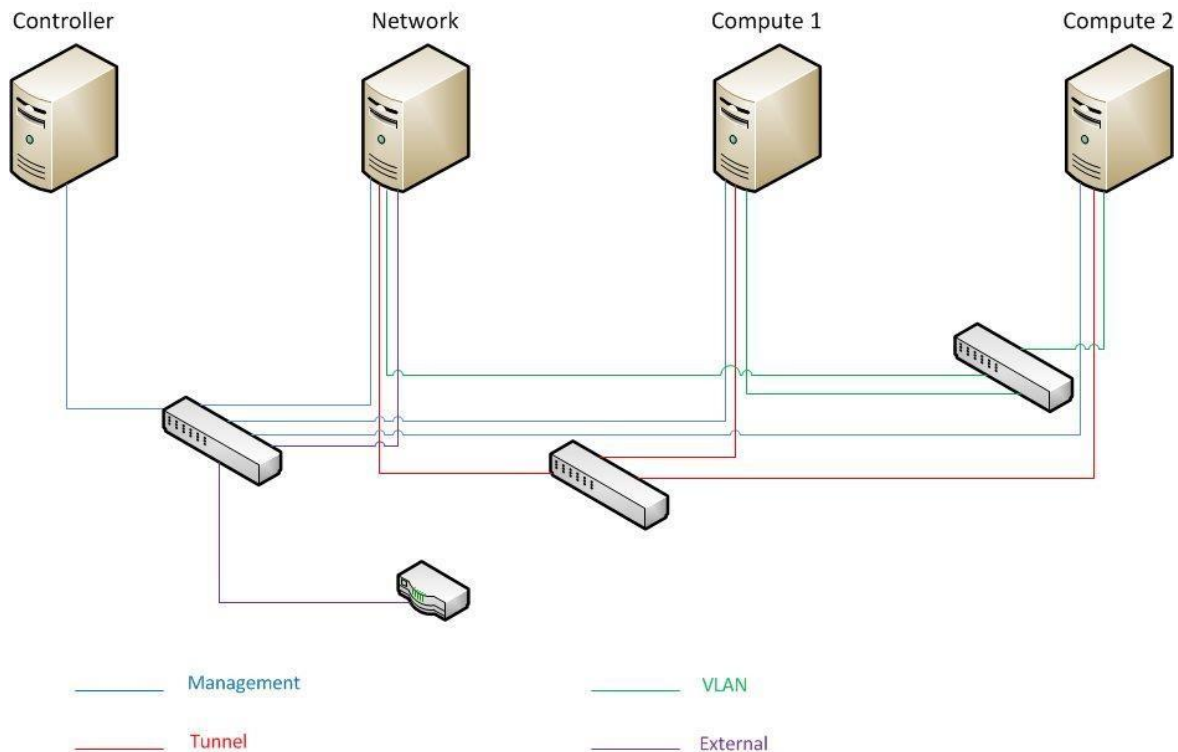


Figure 23. Topologie Réseau

La Figure 23 utilise quatre réseaux simples : le réseau de gestion NAT à des fins administratives, comme l'installation de packages, les mises à jour de sécurité, DNS et NTP, le réseau de tunnel pour le type de trafic interne de machine virtuelle GRE ou VXLAN, le réseau VLAN pour le réseau local virtuel de type VLAN de type VM, et le réseau externe fournissant un accès Internet aux instances (L'Internet). Les réseaux de gestion et externes utilisent le sous-réseau 172.16.0.0/21, le réseau de tunnel utilise le sous-réseau 10.10.10.0/24 et le réseau VLAN ne dispose d'aucune plage d'adresses IP, car il ne gère que la connectivité de couche 2.

Le nœud contrôleur n'a qu'une seule interface de gestion, les nœuds de calcul ont trois interfaces pour la gestion, les réseaux de tunnel et VLAN, et le nœud de réseau dispose de quatre interfaces.

2. Déploiement d'OpenStack

Cette partie explique comment le Cloud OpenStack est construit et comment vérifier la fonctionnalité OpenStack en lançant une instance. Déployer un OpenStack signifie beaucoup de répétition effectuée pour chaque service, je ne parlerai donc que de la façon dont j'ai installé et configuré Keystone et omettrai d'autres services par souci de brièveté.

2.1. Mise en place de l'environnement

L'interface externe de chaque nœud de calcul utilise une configuration spéciale sans adresse IP qui lui est affectée; elle doit donc être configurée pour ne pas utiliser d'adresse IP, comme ci-dessous:

```
auto eth4
iface eth4 inet manual
up ip link set dev $IFACE up down
ip link set dev $IFACE down
```

Le service NTP étant également requis sur tous les nœuds, j'installe et configure chrony de sorte que le fichier /etc/chrony/chrony.conf de chaque nœud contienne la ligne suivante:

```
server <NTP_SERVER/controller> iburst
```

Le nœud du contrôleur doit utiliser un nom d'hôte ou une adresse IP d'un serveur NTP, tandis que d'autres nœuds utilisent le nœud du contrôleur comme serveur NTP.

Ensuite, un référentiel OpenStack spécifique doit être activé. J'utilise la version Liberty, donc j'ajoute le référentiel pour Liberty et installe le client OpenStack sur tous les nœuds comme suit:

```
# apt-get install software-properties-common
# add-apt-repository cloud-archive:liberty
# apt-get install python-openstackclient
```

La plupart des services OpenStack utilisent une base de données SQL pour stocker des informations. Pour que le fichier /etc/mysql/conf.d/mysql_openstack.cnf demande aux autres nœuds d'accéder à la base de données via le réseau de gestion, j'installe et configure MariaDB sur le nœud du contrôleur, ainsi que définir le moteur de stockage par défaut et l'UTF. -8 caractères définis de la manière suivante:

```
[mysqld]
bind-address = <controller-ip>
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

# rabbitmqctl add_user openstack <rabbitmq_password>
# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

Pour le moment, l'environnement dispose de la mise en réseau de l'hôte, des packages NTP, OpenStack et du client, de la base de données SQL et de la file d'alarme. Et les composants centraux OpenStack sont prêts à être installés.

2.2. Installation et configuration de Keystone, Glance, Nova, Neutron et Horizon

Comme mentionné ci-dessus, Keystone est le service OpenStack Identity qui fournit des services d'authentification et d'autorisation. Les services OpenStack prennent en charge plusieurs options de sécurité, telles que le mot de passe, la stratégie et le chiffrement. Pour faciliter le processus d'installation dans ce travail, je n'utiliserai que la méthode du mot de passe. Tout au long du processus de déploiement, un jeton d'administrateur sera également nécessaire. Les mots de passe (GLANCE_DBPASS, KEYSTONE_DBPASS, NEUTRON_DBPASS, NOVA_DBPASS, etc.) et le jeton peuvent être créés manuellement ou générés de manière aléatoire à l'aide d'un outil tel que OpenSSL, comme ci-dessous:

```
openssl rand -hex 10
```

Nous devons créer une base de données pour Keystone avec les accès appropriés accordés comme suit:

```
CREATE DATABASE keystone ;
GRANT ALL PRIVILEGES ON keystone. * TO 'keystone'@'localhost'
IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone. * TO 'keystone'@'%'
IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Ensuite, le paquet Keystone et ses dépendances sont installés. Après cela, le fichier de configuration Keystone situé dans /etc/keystone/keystone.conf doit être modifié pour contenir le jeton admin généré ci-dessus, l'accès à la base de données et certaines autres configurations. La ligne suivante inclut les noms du package Keystone et ses dépendances :

```
# apt-get install keystone apache2 libapache2-mod-wsgi
memcached python-memcache
```

La base de données Keystone est vide pour le moment, nous devons donc la renseigner avec les données de base à l'aide de la commande suivante:

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

À ce stade, une base de données Keystone avec son ensemble complet de tables doit être créée. Par défaut, Keystone est fourni avec une base de données SQLite située dans /var/lib/keystone/keystone.db, mais nous n'en avons pas besoin, elle peut donc être supprimée.

Keystone gère un catalogue de services OpenStack. Nous devons donc créer des entités finales de service et des points de terminaison API pour Keystone. Keystone utilise le port 5000 pour l'accès public et interne et le port 35357 pour l'accès administrateur.

```
$ openstack service create --name keystone --description
"OpenStack Identity" identity
$ openstack endpoint create --region RegionOne identity
```

```
public http://controller:5000/v2.0
$ openstack endpoint create --region RegionOne identity
internal http://controller:5000/v2.0
$ openstack endpoint create --region RegionOne identity
admin http://controller:35357/v2.0
```

Keystone utilise une combinaison de domaines, projets, utilisateurs et rôles d'authentification, comme indiqué ci-dessus. Nous devons donc les créer, qui seront utilisés par l'administrateur pour les opérations administratives en exécutant les commandes suivantes :

```
$ openstack project create --domain default --description
"Admin Project" admin
$ openstack user create --domain default --password-prompt
admin
$ openstack role create admin
$ openstack role add --project admin --user admin admin
```

Nous devons également créer un projet appelé service composé d'un utilisateur unique pour chaque service OpenStack, comme ci-dessous:

```
$ openstack project create --domain default --description
"Service Project" service
```

Pour les opérations suivantes, nous aurons besoin d'un fichier d'identifiants d'administrateur avec le contenu suivant :

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<admin_password>
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
```

Pour Glance, Nova et Neutron, le processus d'installation et de déploiement est presque identique à celui de Keystone.

L'étape suivante consiste à créer des ponts OVS sur le réseau et des nœuds de calcul, puis à ajouter les interfaces physiques correspondantes en tant que ports sur les ponts OVS à l'aide des commandes suivantes :

```
# ovs-vsctl add-br br-tun
# ovs-vsctl add-port br-tun eth1
# ovs-vsctl add-br br-vlan
# ovs-vsctl add-port br-vlan eth2
# ovs-vsctl add-br br-ex
# ovs-vsctl add-port br-ex eth3
```


Les nœuds de calcul et de réseau doivent désormais disposer de piles de réseautage entièrement fonctionnelles, comme illustré aux figures 24 et 25 ci-dessous. En outre, chaque nœud de calcul aura un agent OVS en cours d'exécution, alors que le nœud de réseau aura un agent OVS, un agent L3, un agent DHCP et un agent de métadonnées fonctionnant comme décrit dans l'architecture OpenStack.

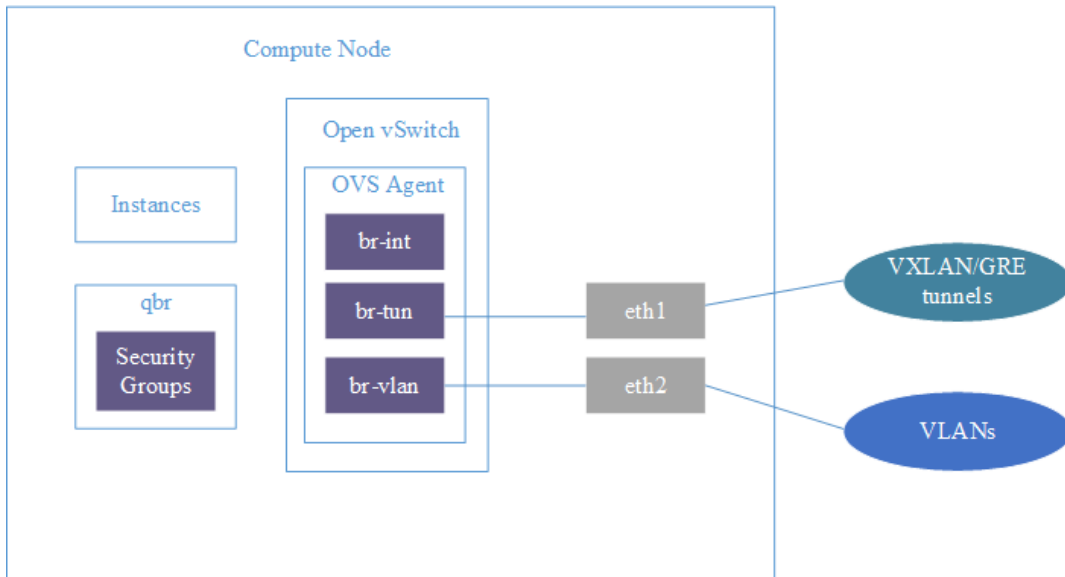


Figure 24. Disposition du réseau de nœuds de calcul

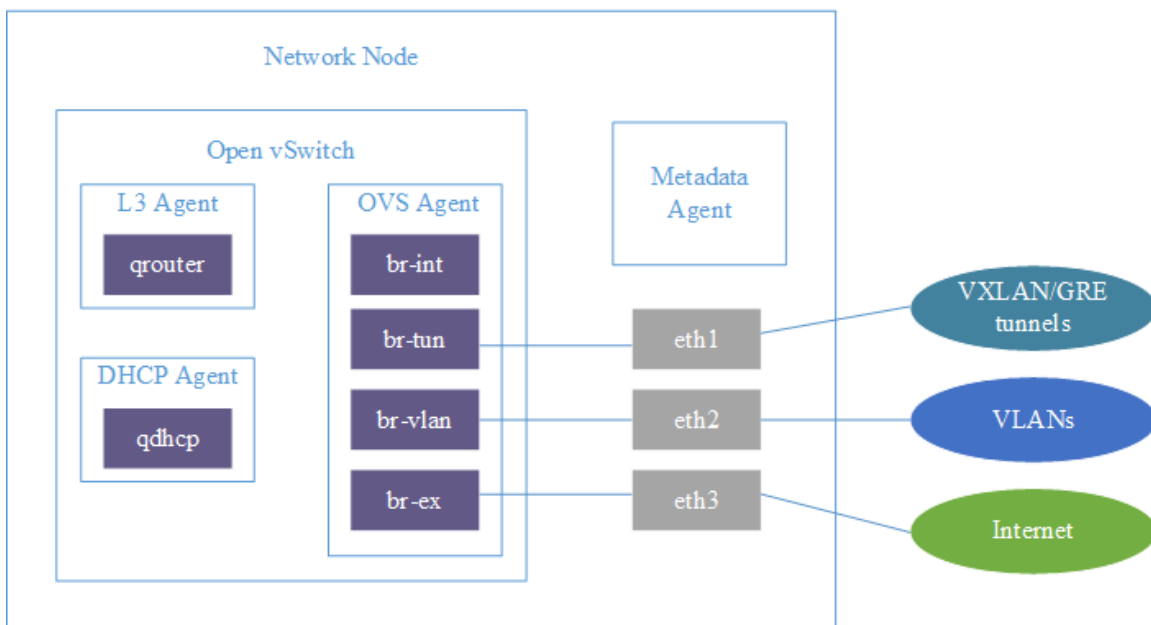


Figure 25. Disposition du réseau de nœuds de réseau

Nous pouvons également obtenir un bref aperçu du contenu de la base de données OVS sur les nœuds de calcul et de réseau en utilisant la commande `ovs-vsctl show`, comme illustré dans les figures 26 et 27 ci-dessous

```
root@network:/home/student# ovs-vsctl show
5a7f6a9e-54d5-45d8-ab72-113e15d6f22d
    Bridge br-tun
        fail_mode: secure
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port br-tun
            Interface br-tun
                type: internal
    Bridge br-int
        fail_mode: secure
        Port int-br-vlan
            Interface int-br-vlan
                type: patch
                options: {peer=phy-br-vlan}
        Port br-int
            Interface br-int
                type: internal
        Port int-br-ex
            Interface int-br-ex
                type: patch
                options: {peer=phy-br-ex}
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
    Bridge br-ex
        Port "eth3"
            Interface "eth3"
        Port br-ex
            Interface br-ex
                type: internal
        Port phy-br-ex
            Interface phy-br-ex
                type: patch
                options: {peer=int-br-ex}
    Bridge br-vlan
        Port phy-br-vlan
            Interface phy-br-vlan
                type: patch
                options: {peer=int-br-vlan}
        Port "eth2"
            Interface "eth2"
        Port br-vlan
            Interface br-vlan
                type: internal
    ovs_version: "2.4.1"
```

Figure 26. Ponts OVS sur un nœud de réseau

```
root@compute1:/home/student# ovs-vsctl show
dd0038ce-e5d1-4312-8aea-fe676e67a840
    Bridge br-tun
        fail_mode: secure
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port br-tun
            Interface br-tun
                type: internal
    Bridge br-vlan
        Port "eth2"
            Interface "eth2"
        Port br-vlan
            Interface br-vlan
                type: internal
        Port phy-br-vlan
            Interface phy-br-vlan
                type: patch
                options: {peer=int-br-vlan}
    Bridge br-int
        fail_mode: secure
        Port br-int
            Interface br-int
                type: internal
        Port int-br-vlan
            Interface int-br-vlan
                type: patch
                options: {peer=phy-br-vlan}
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
    ovs_version: "2.4.1"
```

Figure 27. Ponts OVS sur le nœud de calcul

Pour Horizon, nous devons installer le package qui fournit le tableau de bord en utilisant la commande suivante :

```
# apt-get install openstack-dashboard
```

Ensuite, le fichier de configuration Horizon situé dans / etc / openstack-dashboard / local_settings.py doit indiquer à Horizon d'utiliser les services OpenStack sur le nœud du contrôleur, afin de permettre à tous les hôtes d'accéder à Horizon, de configurer la version de l'API Keystone que nous souhaitons utiliser lorsque en vous connectant au tableau de bord, pour choisir les services Neutron à activer, etc. Le tableau de bord est disponible pour accéder à [http: // <controller-ip> / horizon](http://<controller-ip>/horizon) par la suite. L'écran de connexion est visible dans la figure 28 ci-dessous.

Figure 28. Écran de connexion OpenStack

2.3. Lancer une instance

Tout d'abord, nous devons télécharger et télécharger l'image CirrOS dans Glance comme ci-dessous. CirrOS est une distribution minimale de Linux conçue comme une image de test.

```
$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-  
x86_64-disk.img /tmp  
$ glance image-create --name "cirros" --file /tmp/cirros- 0.3.4-  
x86_64-disk.img --disk-format qcow2 --container-format bare --  
visibility public
```

Ensuite, nous devons créer un réseau externe plat et son sous-réseau en exécutant les commandes suivantes:

```
$ neutron net-create ext-net --router:external True --  
provider:physical_network external --provider:network_type flat  
$ neutron subnet-create ext-net --name ext-subnet --  
allocation-pool start=172.16.2.50,end=172.16.2.100 --  
disable-dhcp --gateway 172.16.0.1 172.16.2.0/24
```

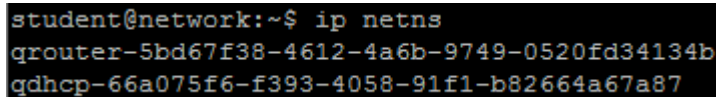
Un réseau VXLAN peut être créé de la même manière pour un projet nommé démo, comme suit:

```
$ neutron net-create demo-net --tenant-id <demo-project-id>
  --provider:network_type vxlan
$ neutron subnet-create demo-net --name demo-subnet --
gateway 200.200.200.1 200.200.200.0/24
```

Après cela, un routeur qui connecte le réseau de démonstration au réseau externe est créé à l'aide des commandes suivantes:

```
$ neutron router-create demo-router
$ neutron router-interface-add demo-router demo-subnet
$ neutron router-gateway-set demo-router ext-net
```

Plusieurs espaces de noms `qrouter` et `qdhcp` apparaîtront sur le nœud du réseau, comme illustré à la figure 29. Un espace de noms réseau est une copie logique de la pile de mise en réseau dotée de ses propres routeurs, règles de pare-feu et périphériques d'interface réseau. Un espace de noms `qdhcp` fournit des adresses IP aux instances via son service DHCP. Chaque réseau pour lequel DHCP est activé sur les sous-réseaux associés possède un espace de noms `qdhcp`. Un espace de noms `qrouter` représente un routeur virtuel et est responsable du routage du trafic depuis et vers les instances.



```
student@network:~$ ip netns
qrouter-5bd67f38-4612-4a6b-9749-0520fd34134b
qdhcp-66a075f6-f393-4058-91f1-b82664a67a87
```

Figure 29. Espaces de noms réseau Neutron

Il existe un groupe de sécurité par défaut nommé `default`. Nous pouvons ajouter des règles supplémentaires pour autoriser l'accès ping et SSH à l'instance, comme ci-dessous :

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

Une paire de clés dont la clé publique est injectée dans une instance nouvellement créée est nécessaire pour que nous puissions accéder aux instances. La clé publique peut être téléchargée avec la commande suivante:

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

Il est maintenant prêt à lancer une instance telle que celle illustrée à la figure 30. Les arômes disponibles peuvent être répertoriés à l'aide de la commande `nova flavour-list`, et `net-id` est l'ID de démon obtenu à l'aide de la commande `neutron net-list`.

```

-----+
student@controller:~$ nova boot --flavor ml.tiny --image cirros --nic net-id=66a
075f6-f393-4058-91f1-b82664a67a87 --security-group default --key-name demo-key v
ml
-----+
+-----+
| Property                                | Value
|-----+-----|
+-----+
| OS-DCF:diskConfig                       | MANUAL
|-----+-----|
| OS-EXT-AZ:availability_zone             |
|-----+-----|
| OS-EXT-STS:power_state                  | 0
|-----+-----|
| OS-EXT-STS:task_state                   | scheduling
|-----+-----|
| OS-EXT-STS:vm_state                     | building
|-----+-----|
| OS-SRV-USG:launched_at                  | -
|-----+-----|
| OS-SRV-USG:terminated_at                | -
|-----+-----|
| accessIPv4                              |
|-----+-----|
| accessIPv6                              |
|-----+-----|
| adminPass                               | u9XB0ifQ74Kv
|-----+-----|
| config_drive                            |
|-----+

```

Figure 30. Lancement d'une instance OpenStack

Après avoir associé une adresse IP flottante à l'instance en cours d'exécution, nous devrions pouvoir y accéder via SSH. Lorsque deux instances sont créées et se lancent l'une contre l'autre, nous pouvons constater qu'il existe plusieurs flux OVS. La base de données OVS est également mise à jour sur le réseau et les nœuds de calcul, comme illustré à la figure 31, à l'aide de la même commande `show ovs-vsctl`. Chaque base de données OVS sur ces nœuds sera tenue à jour avec les informations des autres nœuds, telles que les adresses IP et de nombreuses interfaces réseau virtuelles liées à des instances (unités de prise, paires veth, ponts Linux et ponts OVS; leurs préfixes correspondants sont: `taper`, `qvb`, `qvo`, `qbr`, `qr-`, `qg-` et `br`). Une discussion détaillée sur la nature de ces interfaces, leur fonctionnement et la manière dont elles se connectent dépasse le cadre de ce projet.

```

Bridge br-tun
    fail_mode: secure
    Port patch-int
        Interface patch-int
            type: patch
            options: {peer=patch-tun}
    Port "vxlan-0a0a141f"
        Interface "vxlan-0a0a141f"
            type: vxlan
            options: {df_default="true", in_key=flow, local_ip="10.10.20.21"
, out_key=flow, remote_ip="10.10.20.31"}
    Port br-tun
        Interface br-tun
            type: internal
Bridge br-int
    fail_mode: secure
    Port "tap1797ce2f-e2"
        tag: 3
        Interface "tap1797ce2f-e2"
            type: internal
    Port int-br-vlan
        Interface int-br-vlan
            type: patch
            options: {peer=phy-br-vlan}
    Port "qg-af31ba5c-68"
        tag: 4
        Interface "qg-af31ba5c-68"
            type: internal
    Port br-int
        Interface br-int
            type: internal
    Port int-br-ex
        Interface int-br-ex
            type: patch
            options: {peer=phy-br-ex}
    Port "qr-ecd9fe4c-78"
        tag: 3
        Interface "qr-ecd9fe4c-78"
            type: internal
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
Bridge br-ex
    Port "eth3"
        Interface "eth3"
    Port br-ex
        Interface br-ex
            type: internal
    Port phy-br-ex
        Interface phy-br-ex
            type: patch
            options: {peer=int-br-ex}
Bridge br-vlan
    Port phy-br-vlan
        Interface phy-br-vlan
            type: patch
            options: {peer=int-br-vlan}
    Port "eth2"
        Interface "eth2"
    Port br-vlan
        Interface br-vlan
            type: internal

```

Figure 31. Ponts, ports et flux OVS sur le nœud de réseau

3. Intégration OpenDaylight dans OpenStack

Le nœud ODL est également une machine virtuelle, à l'instar des nœuds de contrôleur et de réseau, mais il possède deux interfaces: l'une appartient au réseau de gestion pour pouvoir communiquer avec les autres nœuds, l'autre au réseau de tunnels.

Après le téléchargement et l'extraction de la dernière version d'OpenDaylight, Beryllium au moment de la rédaction, vous pouvez démarrer OpenDaylight en tant que processus serveur, puis vous connecter au Shell Karaf pour installer certaines fonctionnalités Karaf nécessaires et leurs dépendances à l'aide des commandes suivantes:

```
$ ./bin/start
$ ./bin/client
$ feature:install odl-base-all odl-aaa-authn odl-restconf
odl-nsf-all odl-adsal-northbound odl-mdsal-apidocs odl-
ovsdb-openstack odl-ovsdb-northbound odldlux-core
```

La figure 32 illustre l'architecture actuelle de l'intégration OpenDaylight et OpenStack.

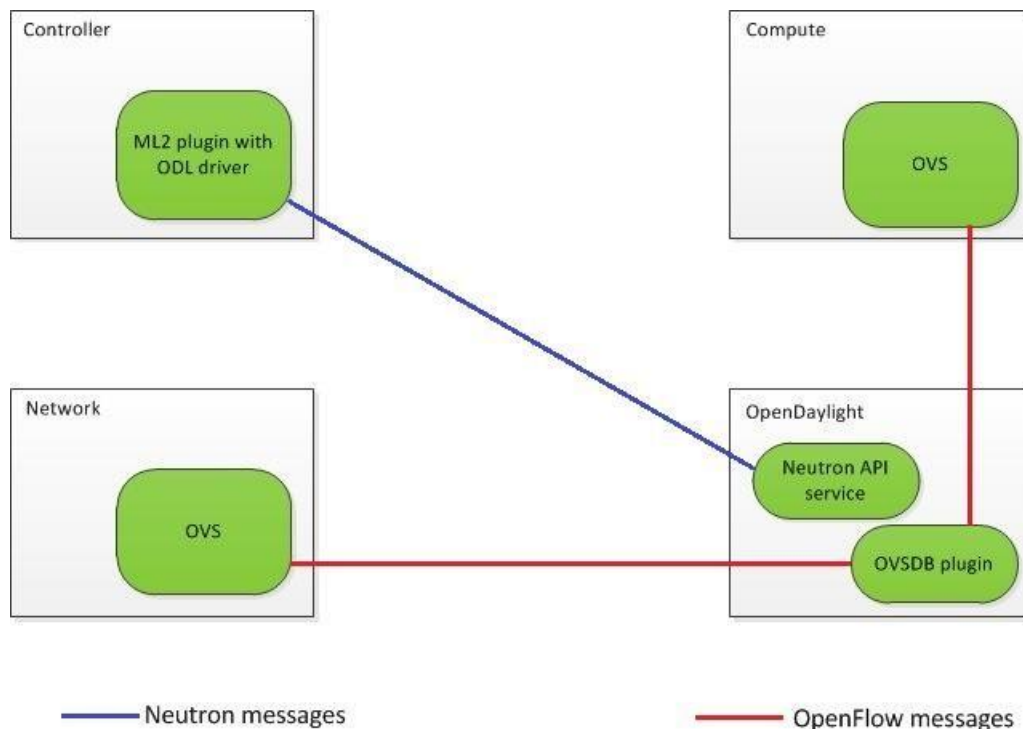


Figure 32. Intégration OpenStack et OpenDaylight

Étant donné qu'ODL est utilisé comme serveur principal Neutron, il est censé être la seule source de vérité pour la configuration OVS. Nous devons donc nettoyer toutes les configurations OpenStack et OVS existantes afin de nous assurer que l'ODL est propre. Le service de serveur à neutrons sur le nœud du contrôleur et le service neutron-openvswitch-agent sur tous les nœuds doivent être arrêté et l'agent OVS doit également être désactivé pour ne pas revenir après un redémarrage, comme suit:

```
# service neutron-server stop
# service neutron-openvswitch-agent stop
# service neutron-openvswitch-agent disable
```

Afin de permettre à OpenDaylight de gérer complètement OVS, vous devez supprimer la base de données OVSDb existante, puis configurer ODL de manière à gérer tous les nœuds, comme ci-dessous :

```
# service openvswitch-switch stop
# rm -rf /var/log/openvswitch/*
# rm -rf /etc/openvswitch/conf.db #
service openvswitch-switch start
# ovs-vsctl set-manager tcp:<odl_management_ip>:6640
```

Tous les nœuds avec OVS en cours d'exécution apparaissent dans DLUX, comme illustré à la Figure 33.

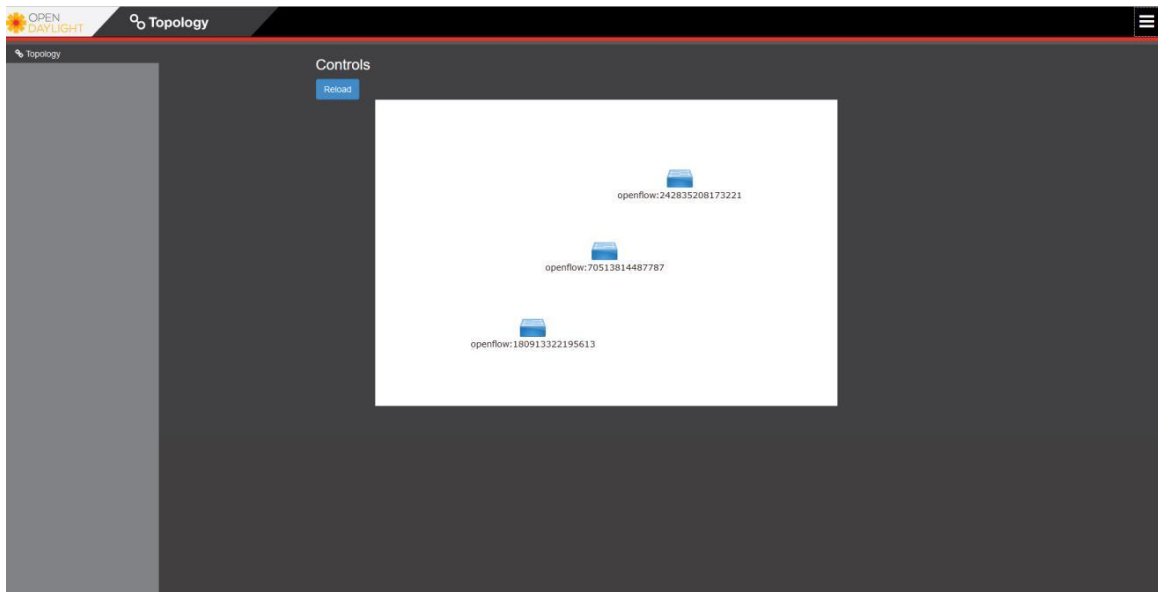


Figure 33. DLUX

Nous devons configurer Neutron pour utiliser ODL en ajoutant le pilote du mécanisme opendaylight à sa liste existante. La partie configuration pour ODL est visible ci-dessous:

```
[ml2]
mechanism_drivers = opendaylight
[ml2_odl]
password = admin
username = admin
```

```
url =
http://<odl_management_ip>:8080/controller/nb/v2/neutron
```

La base de données Neutron doit également être supprimée, recrée et repeuplée. Après avoir recréé des réseaux, des sous-réseaux et des routeurs, je crée deux instances, comme illustré à la figure 34.

```
openstack@controller:~$ nova list
+-----+-----+-----+-----+-----+
+-----+
| ID                                     | Name | Status | Task State | Power Stat
e | Networks                               |
+-----+-----+-----+-----+-----+
+-----+
| 8ca07264-3d85-4387-a323-b39002c73eed | vm1  | ACTIVE | -          | Running
| demo-net=200.200.200.3 |
| f02a2753-086d-4301-aa85-47e2cfbeb5eb | vm2  | ACTIVE | -          | Running
| demo-net=200.200.200.4 |
+-----+-----+-----+-----+-----+
+-----+
openstack@controller:~$
```

Figure 34. Exécution d'instances

J'accède à l'une des instances en utilisant SSH et en envoyant une requête ping à l'autre, et cela fonctionne comme le montre la figure 35. Cela signifie qu'ODL a été intégré avec succès dans le Cloud OpenStack.

```
$ ping 200.200.200.3
PING 200.200.200.3 (200.200.200.3): 56 data bytes
64 bytes from 200.200.200.3: seq=0 ttl=64 time=0.846 ms
64 bytes from 200.200.200.3: seq=1 ttl=64 time=0.138 ms
64 bytes from 200.200.200.3: seq=2 ttl=64 time=0.110 ms
64 bytes from 200.200.200.3: seq=3 ttl=64 time=0.109 ms
64 bytes from 200.200.200.3: seq=4 ttl=64 time=0.107 ms
64 bytes from 200.200.200.3: seq=5 ttl=64 time=0.129 ms
64 bytes from 200.200.200.3: seq=6 ttl=64 time=0.114 ms
64 bytes from 200.200.200.3: seq=7 ttl=64 time=0.116 ms

--- 200.200.200.3 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.107/0.208/0.846 ms
$ _
```

Figure 35. Vérification de ping

