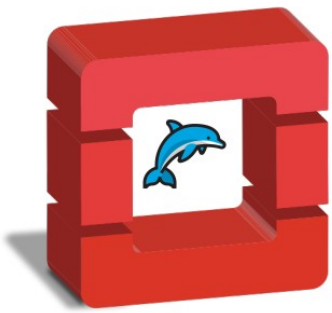
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et de l'Informatique
Département de Mathématiques et d'Informatique
Filière: Informatique

MÉMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option: **Réseaux et Systèmes**

THÈME: ÉTUDE ET IMPLÉMENTATION D'UNE SOLUTION CLOUD BASÉE SUR DOCKER POUR EXÉCUTER DES APPLICATIONS COMPOSITES



Étudiant : Mohamed El-amine FATIS

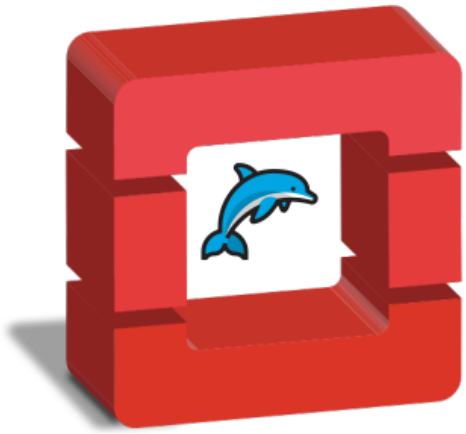
Jury
Encadrante : Dr. Fatima Zohra Filali
Examinateur : Dr. M Abid
Président : Dr. A.E.K BENAMOUR

Année Universitaire 2018-2019

UNIVERSITE ABDELHAMID IBN BADIS

Mémoire présenté pour l'obtention Du diplôme de master professionnalisant Réseaux et Systèmes

Étude et implémentation d'une solution cloud basé sur Docker pour exécuter des applications composites



Présenté par : Mr Mohamed El-amine FATIS

Encadreur : Dr Fatima Zohra Filali

Septembre 2018 — Juin 2019

Résumé

Au cours des dernières années, l'informatique en nuage est devenue la prochaine grande révolution des réseaux informatiques et de l'approvisionnement. Certaines solutions à code source ouvert gagnent dernièrement en popularité, telle que Open Stack, qui est actuellement largement utilisée dans le déploiement de cloud public et privé. Cela est dû au fait qu'il est ouvert et qu'il repose sur un écosystème puissant et en pleine croissance. Docker est la solution qui fait ses preuves dont la normalisation et la standardisation, il fournit un environnement de développement, de construction, de test et de production reproductible et isolée. Docker est rapide et permet d'effectuer rapidement des réplications et d'obtenir une redondance. En outre, il existe plusieurs projets qui s'engagent à intégrer Docker dans la plateforme Openstack. Dans notre travail, nous allons présenter l'intégration de Docker dans un environnement en nuage à travers le composant Zun.

Mots Clés : Cloud Computing, IaaS, SaaS, OpenStack, Conteneur Docker, Zun.

Abstract

In recent years, cloud computing has become the next major revolution in computer networks and procurement. Some open source code-based solutions are gaining in popularity lately, such as OpenStack, which is currently widely used in public and private cloud deployment. This is due to the fact that it is open and supported by a strong and growing ecosystem. Docker is the proven solution including standardization and normalization, it provides a reproducible and isolated development, construction, testing and production environment. Docker is fast and allows for quick replication and redundancy. In addition, there are several projects committed to integrating of Docker into the OpenStack platform. In our work, we will present the integration of Docker into a cloud environment through the Zun component.

Keywords : Cloud Computing, IaaS, SaaS, OpenStack, Docker Container, Zun.

Sommaire

Listes des figures

Liste des tableaux

Introduction

Les progrès technologiques en matière d'infrastructure matérielle (puissance de calcul, de stockage, etc.), de connexions réseau et technologies Internet ont permis l'émergence d'une nouvelle branche appelée l'informatique en nuage « Cloud Computing ». Le concept sous-jacent remonte à 1960 lorsque John McCarthy a estimé que « le calcul pourrait un jour être organisé comme une entreprise de service public ».

Le cloud computing est un concept relativement nouveau pour lequel les ressources sont étendues, virtualisées et fournies sous forme de service sur Internet. Il permet également aux fournisseurs de donner aux utilisateurs un accès à un nombre pratiquement illimité de ressources. Le cloud computing est un concept assez nouveau qui regroupe toutes les techniques (services Web, virtualisation, architecture orientée service, calcul en grille, etc.) et les modèles commerciaux utilisés pour fournir des fonctionnalités informatiques (logiciel, plate-forme, matériel) sous forme de demande de service, évolutive et élastique.

Plusieurs solutions de Cloud ont été utilisé ces dernières années. L'une des solutions Cloud open source est la plateforme Openstack. est une initiative de l'industrie basée sur une collaboration mondiale de développeurs et de technologues produisant le système d'exploitation en nuage standard. Il a été fondé par la NASA et Rackspace Hosting, qui est rapidement devenue une communauté logicielle mondiale de développeurs collaborant sur un système d'exploitation en nuage open source standard et massivement évolutif. OpenStack comprend principalement trois projets logiciels, tels que OpenStack Compute, OpenStack Objet Storage et OpenStack Image Service. La neuvième version du logiciel open source Openstack Icehouse est utilisé pour la création de clouds publics, privés et hybrides. De nouvelles fonctionnalités, environ 350, ont été ajoutées pour prendre en charge le développement de logiciels et la gestion de données.

Le développement actuel et l'adoption généralisée des techniques de cloud computing ont amené de nouvelles technologies alternatives utilisées pour accélérer et faciliter le lancement d'application dans le cloud. Ces solutions doivent relever le défi de fournir efficacement de nouvelles ressources afin de répondre aux demandes du nombre croissant d'utilisateurs de l'application. L'une de ces solutions et Docker. Docker est un moteur open source qui automatise le déploiement d'application dans des conteneurs. Il a été écrit par l'équipe de Docker, Inc (anciennement dot Cloud Inc, l'un des premiers acteurs du marché de la plate-forme en tant que service (PAAS)), et publié par cette dernière sous la licence Apache 2.0.

L'objectif de notre travail est d'intégrer l'outil Docker à un environnement de Cloud computing et plus exactement à la plateforme OpenStack. Pour cela, nous allons aborder dans les deux premiers chapitres les différents concepts et terminologies du Cloud computing et de la virtualisation. Dans le

troisième chapitre, nous présenterons les différents outils de notre implémentation qui sont : OpenStack et Docker. Enfin, dans le dernier chapitre on discutera les deux solution envisagées Nova-Docker et Zun et on se concentrera sur cette dernière.

Chapitre 1

Cloud Computing

1.1 Introduction

Le cloud computing, traduit par "l'informatique en nuage", a pour vision que les services informatiques arrivent sur le site de l'utilisateur comme de l'énergie électrique et sont également facturés en fonction de la consommation. Les services et ressources informatiques deviennent disponibles sur demande, ainsi que l'électricité, l'eau ou d'autres services de base[10].

Mais le cloud computing est aussi une idée commerciale, à savoir que les logiciels et les données ne sont plus traités ou stockés localement par l'utilisateur, mais via Internet en tant que médiateur auprès d'un tiers qui fournit le logiciel et les données de manière dynamique et en fonction des besoins. La fourniture en ligne de matériel, de logiciels et de données via le cloud computing est donc un service.

Avec la mise en place de l'informatique en nuage, les centres de données distribués plus petits et les ordinateurs autonomes à grande échelle, qui restent inutilisés la plupart du temps et entraînent des coûts de main-d'œuvre, d'énergie et de production considérables, prendront fin. La puissance de calcul, la capacité de stockage (services d'infrastructure), les plates-formes de développement de logiciels (plateforme en tant que service) ou les logiciels d'application (logiciels en tant que service) ne sont utilisés et payés par les utilisateurs que lorsque cela est nécessaire "à partir du Cloud".

Le cloud computing offre de nombreux avantages par rapport au déploiement informatique traditionnel. Le risque d'exploitation du logiciel est sous-traité à des tiers, une mise à jour des programmes de chaque ordinateur n'est pas nécessaire, mais se fait de manière centralisée, la disponibilité est accrue car souvent, plusieurs centres de données distribués, parfois globaux, sont disponibles, ce qui réduit le risque de pannes totales. Les frais de licence pour les licences utilisatrices individuelles ne sont plus appliqués et, de plus, les coûts de maintenance et donc de personnel pour le matériel et les logiciels sont réduits. Une nouvelle acquisition coûteuse de matériel et de logiciel n'est souvent plus nécessaire.[10].

1.2 Définition

L'Institut national des normes et de la technique (NIST) [1] a formulé une définition de l'informatique en nuage qui est devenue de plus en plus détaillée au fil du temps. La définition finale de la version 16 est : "le cloud computing est un modèle permettant un accès réseau à la demande omniprésent et pratique au pool de ressources informatiques configurables (réseaux, serveurs, stockage, applications et services, par exemple) qui peuvent être provisionnés et libérés avec un minimal d'effort de gestion ou interaction du fournisseur de services [11].

Comme indiqué plus haut, le cloud computing signifie l'utilisation de logiciels et d'autres services (stockage, puissance de calcul) sur Internet sur le matériel physique et virtuel du fournisseur de cloud. Le fournisseur fournit ces services au client (utilisateurs du nuage) moyennant des frais, lui permettant ainsi de s'abstenir presque totalement de ses propres matériels et logiciels ou d'utiliser les services du nuage en plus de son infrastructure informatique. Même avec une dispense de matériel, en particulier de serveurs ou de centres de données, l'utilisateur a besoin d'un ordinateur moins puissant doté d'un navigateur Web. Les services de cloud computing peuvent être utilisés par le client et ses employés.

1.3 Modèle de services

Une première caractéristique remarquable est le modèle en couches (ou modèle de services) du cloud computing. Comme indiqués dans l'introduction, nous faisons la distinction entre l'infrastructure en tant que service (IaaS), la plate-forme en tant que service (PaaS) et le logiciel en tant que service (Saas). La couche infrastructure représente la couche la plus basse, suivie de la couche plate-forme sur laquelle repose la couche application (logiciel). Plus la couche est haute, plus les services logiciels sont probables.

1.3.1 Infrastructure as a Service -IaaS

Au niveau de l'infrastructure, les fournisseurs fournissent des services de base tels que la puissance de calcul (puissance de traitement), le stockage de données (stockage et hébergement) et parfois des liaisons de communication. En raison de la nature matérielle de ces services, le terme matériel en tant que service (Haas) est également courant.

À l'aide de ces services matériels virtualisés, les deux couches Saas et PaaS peuvent être réalisées. Cependant, ce n'est pas obligatoire. Les utilisateurs peuvent également utiliser les services de base

1. NIST est une agence du département du commerce des États-Unis. Son but est de promouvoir l'économie en développant des techniques, la métrologie et des standards de concert avec l'industrie. Cette agence a pris la suite en 1988 du national Bureau of Standards fondé en 1901 avec substantiellement les mêmes missions.

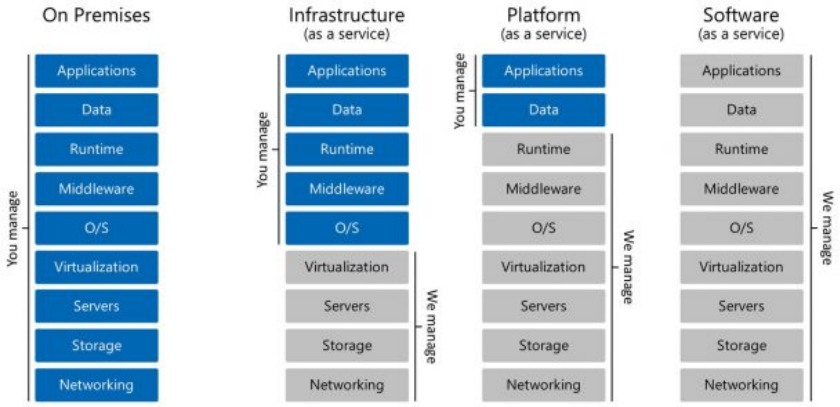


FIGURE 1.1 : Modèle de couche Cloud Computing [1]

pour leurs tâches, par exemple effectuer des opérations arithmétiques approfondies ou externaliser des données vers les serveurs du fournisseur. À savoir, avec les offres IaaS, l'utilisateur dispose d'un accès complet au matériel virtuel et peut également installer des applications lui-même. En contrepartie, il doit également gérer les serveurs lui-même et importer les documents de sécurité nécessaires pour le logiciel utilisé. De même, le fournisseur de cloud ne sera pas responsable des dommages résultant d'une mauvaise configuration par le client [11].

Le service d'infrastructure bien connue est proposé par Amazon Elastic compute Cloud (EC2). Le service de stockage Amazon simple storage Service (S3) est également affecté à la couche IaaS. Parmi les autres services d'hébergement en nuage populaires, citons GoGrid et Linode, ou l'offre de Google appelée Stockage Google [12].

1.3.2 Platform as a Service –PaaS

Plate-forme en tant que service signifie, d'une part, la fourniture de plates-formes de développement logiciel par des fournisseurs de cloud (environnement de programmation) et, d'autre part, la possibilité d'exécuter le logiciel développé sur le réseau. Les services PaaS s'adressent donc aux développeurs de logiciels indépendants, aux sociétés de développement de logiciels ou aux services informatiques d'autres sociétés.

L'approvisionnement centralisé dans le cloud présente l'avantage de simplifier la création d'un environnement de développement logiciel pour les développeurs. De plus, les développeurs n'ont plus à s'inquiéter de l'administration du serveur précédemment requise, mais peuvent le faire. Contrairement à Ias, les utilisateurs ne devraient pas pouvoir administrer les serveurs sous-jacents. Cette étape sera prise en charge par le fournisseur de plate-forme [11].

1.3.3 Software as a Service –SaaS

Saas représentent l'approvisionnement d'application en tant que service (Saas). La fonctionnalité fournit au consommateur consisté à utiliser les applications du fournisseur s'exécutant sur une infrastructure cloud. Les applications sont accessibles à partir de divers périphériques clients via une interface client léger, telle qu'un navigateur Web (par exemple, une messagerie Web) ou une interface de programme. Le consommateur ne gère ni ne contrôle l'infrastructure cloud sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, le stockage ou même les fonctionnalités d'applications individuelles, à l'exception possible des paramètres de configuration d'application spécifiques à un utilisateur [11].

1.3.4 Autres Service

Souvent, dans ce contexte, le terme "tout en tant que service" est utilisé, l'abréviation XaaS étant utilisée et le "X" désignant le service possible. Cependant, cette extension est plus technique que marketing car les services peuvent également être inclus dans les trois couches. Un exemple d'un tel service est la communication en tant que service (CaaS), le prétendu terme moderne des services de télécommunications classiques. L'accès classique à Internet et les dorsales Internet constituent la base de l'utilisation des services en nuage. De tels services de télécommunications classiques peuvent également être fournis par un seul fournisseur [11].

1.4 Modèles d'utilisation

Outre la distinction entre les trois couches, une distinction est également habituelle en fonction de la conception organisationnelle des offres de cloud. Ses différentes offres étant utilisées différemment, elles sont également décrites sous le terme générique de "modèles d'utilisation". Ces modèles d'utilisation font la distinction entre les clouds publics et externes accessibles et utilisables par tous, et ceux qui ne sont utilisés que dans l'entreprise ou l'organisation, c'est-à-dire qui ne sont pas publics (clouds privés et internes).

1.4.1 Cloud public

L'infrastructure en nuage est mise à la disposition du grand public, pour une utilisation ouverte. Elle peut être détenue, gérée et exploitée par une entreprise, un organisme universitaire ou un organisme gouvernemental, ou une combinaison de ceux-ci. Il existe chez le fournisseur de cloud [11].

1.4.2 Cloud Privé

Un cloud privé n'est pas destiné au public, mais est utilisé en interne par une organisation, telle qu'une entreprise ou un organisme public. Toutefois, "privé" ne signifie pas nécessairement que les serveurs

sont la propriété de l'entreprise, ni que l'entreprise est l'opérateur du cloud. Privé signifie simplement que cette unité organisationnelle utilise et contrôle un environnement cloud seul et exclusivement. L'infrastructure, en particulier les centres de calcul et les serveurs, peut également être louée par un fournisseur de ressources externes [11].

En matière de protection des données, toutefois, les serveurs sont toujours sous la responsabilité d'une seule unité de traitement de données, à savoir l'organisation qui utilise les ressources. Les clouds privés se caractérisent donc par le fait qu'ils sont contrôlés par l'organisation les utilisant [13].

1.4.3 Cloud Communauté

Dans les clouds communautaires, des infrastructures disparates sont initialement partagées entre plusieurs organisations. Les exigences collectives, telles que la sécurité ou la confidentialité, sont définies. Les nuages de communauté sont des nuages privés avec un nombre gérable d'organisations différentes en tant qu'utilisateurs. Avec les clouds communautaires, des économies peuvent être réalisées en utilisant mieux les ressources existantes [14].

Les nuages de communauté ressemblent aux nuages internes, en particulier dans l'administration publique et les grandes autorités, voire même avec des agences de renseignements [15].

1.4.4 Cloud hybrides

L'infrastructure est composée de deux couches (privées, communautaires ou publiques) qui restent, mais elle est liée à une norme ou à une technique appropriée, qui permet la portabilité des données et des applications (par exemple, le cloudbursting [2] pour équilibrer la charge entre deux couches)[13].

1.5 Acteurs de Cloud

Les organisations et les êtres humains peuvent représenter différents types d'acteurs selon leurs relations et/ou interaction avec un nuage et ses ressources informatiques hébergées. Chacun des acteurs participe et assume des responsabilités en relation avec l'activité basée sur le cloud. Les sections suivantes définissent ces acteurs et identifient leurs principales interactions.

1.5.1 Fournisseur de Cloud (Provider)

L'entité qui fournit des ressources informatiques basées sur le cloud s'appelle le fournisseur du cloud. En assumant le rôle de fournisseur de cloud, une organisation est responsable de mettre les services en

2. le cloud bursting est une configuration établie entre un cloud privé et un cloud public pour gérer les pics de demande informatique.

nuage à la disposition des consommateurs en nuage, conformément aux garanties de contrats convenues. Le fournisseur du cloud est en outre chargé de toute tâche administrative et de gestion requise pour assurer l'exploitation en cours de l'infrastructure globale du cloud.

1.5.2 Consommateur de Cloud (Consumer)

Un consommateur en nuage est une organisation (ou un être humain) qui a un contrat ou un accord formel avec un fournisseur de cloud pour utiliser les ressources informatiques mises à disposition par le fournisseur du cloud. Plus précisément, le consommateur en nuage utilise un service cloud de consommateur pour accéder à un service cloud.

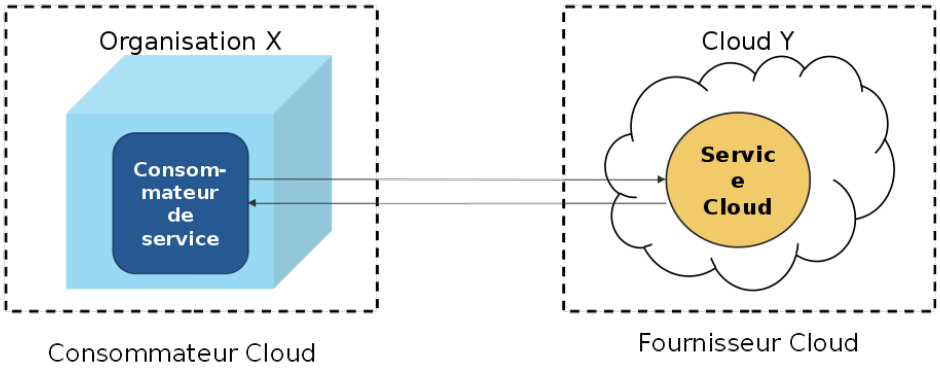


FIGURE 1.2 : Fournisseur et consommateur du Cloud

1.5.3 Propriétaire de service Cloud

La personne ou l'organisation qui possède légalement un service en nuage s'appelle un propriétaire de service cloud. Le propriétaire du service cloud peut être le consommateur en nuage ou le fournisseur de cloud possédant le nuage dans lequel réside le service cloud. Par exemple, le consommateur en nuage de Cloud X ou le fournisseur du Cloud X pourrait posséder le service Cloud A.

de cloud.png

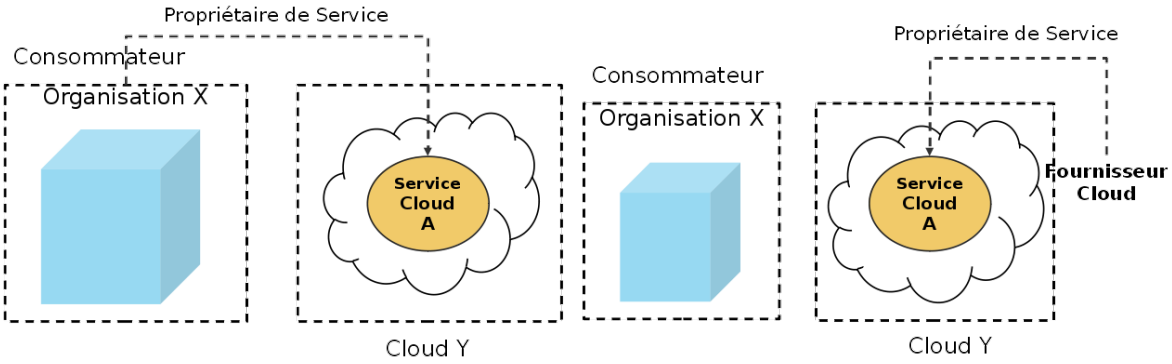


FIGURE 1.3 : Propriétaire de service Cloud

1.5.4 Administrateur de ressources

Un administrateur de ressources en nuage est la personne ou l'organisation responsable de l'administration d'une ressource informatique basée sur le cloud (y compris les services en nuage). L'administrateur de ressources en nuage peut être (ou appartenir) au consommateur en nuage ou au fournisseur du nuage dans lequel réside le service cloud. Alternativement, il peut être (ou appartenir) à un organisme tiers contracté pour administrer la ressource informatique basée sur le cloud.

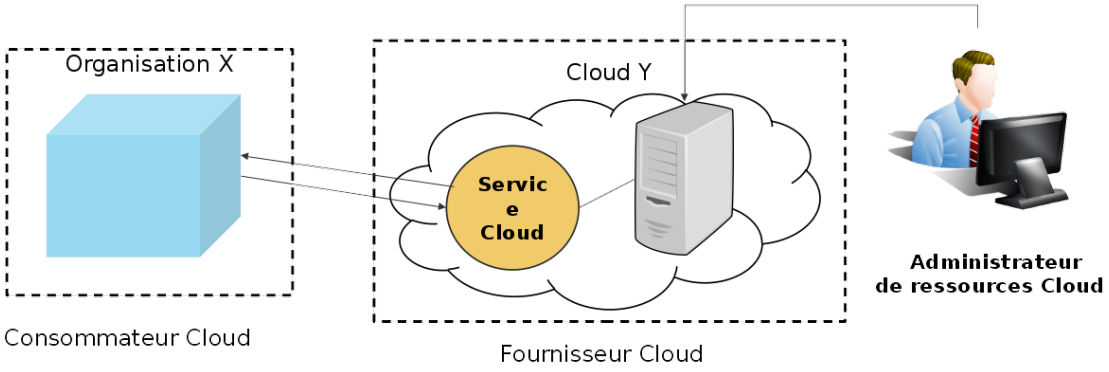


FIGURE 1.4 : Administrateur de Ressources

1.5.5 Autres

L'architecture de référence Cloud Computing du NIST [1] définit les acteurs supplémentaires suivants :

- Auditeur Cloud (Cloud Auditor) : un tiers (souvent accrédité) qui effectue des évaluations indépendantes des environnements en nuage assume le rôle de l'auditeur du cloud. Les responsabilités typiques associées à ce rôle comprennent l'évaluation des contrôles de sécurité, des impacts sur la vie privée et des performances. L'objectif principal du rôle de l'auditeur en nuage est de fournir une évaluation impartiale (et une approbation possible) d'un environnement de cloud computing afin de renforcer la relation de confiance entre les consommateurs de cloud et les fournisseurs de cloud [15].

- Cloud Broker : ce rôle est assumé par une partie qui assume la responsabilité de gérer et de négocier l'utilisation de services en nuage entre les consommateurs en nuage et les fournisseurs de cloud. Les services de médiation fournis par les courtiers en nuage comprennent l'intermédiation, l'agrégation et l'arbitrage des services [15].

- Cloud Carrier : La partie responsable de la connectivité au niveau du fil entre les consommateurs en nuage et les fournisseurs de cloud assume le rôle de fournisseur de cloud. Ce rôle est souvent assumé par les fournisseurs de réseaux et de télécommunications [15].

1.6 Caractéristiques du cloud computing

Un environnement informatique nécessite un ensemble spécifique de caractéristiques pour permettre l'approvisionnement à distance de ressources informatiques évolutives et mesurées de manière efficace. Ses caractéristiques doivent exister dans une mesure significative pour que l'environnement informatique soit considéré comme un nuage efficace. Les six caractéristiques spécifiques suivantes sont communes à la majorité des environnements en nuage :

- Utilisation à la demande (On-Demand Usage).
- Accès omniprésent (Ubiquitous Access).
- Mutualisation et regroupement de ressources (Multitenancy).
- Élasticité (Elasticity).
- Utilisation mesurée (Measured Usage).
- Résilience (Resiliency)

1.7 Conclusion

Dans ce chapitre, nous avons défini les concepts de l'informatique en nuage, ses différents modèles de service et de déploiements ainsi que ses caractéristiques principales.

Chapitre 2

Virtualisation

2.1 Introduction

La virtualisation consiste à faire fonctionner un à plusieurs systèmes d'exploitation sur un système hôte et de pouvoir utiliser ses ressources. Avant que le cloud computing puisse émerger, certaines technologies devaient évoluer dans les pratiques commerciales et, en outre, Internet devait jouer le rôle de médiateur auprès de la population via un accès Internet à bas prix et à haute performance. Sans la virtualisation et l'accès Internet a large bande ainsi que l'évolution des systèmes distribués, l'informatique en nuage n'aurait jamais été possible sous sa forme actuelle.

La virtualisation est au cœur du cloud computing, c'est-à-dire de l'abstraction des systèmes logiques de la mise en œuvre physique et de l'infrastructure. Les logiciels et en particulier le système d'exploitation et les données sont découplés d'un matériel physique spécifique, peuvent être affectés de manière dynamique et utilisés en tant que service.

2.2 Avantages de la virtualisation

Les avantages de la virtualisation sont la reconfigurabilité dynamique, la prise en charge de différentes architectures de réseau, la réduction des frais généraux et une efficacité énergétique accrue. La reconfigurabilité dynamique est le principal avantage de la virtualisation. Les ressources virtuelles peuvent être créées, déplacées, supprimées et même dynamiquement modifiées[16].

La prise en charge de différentes architectures de réseau signifie que plusieurs réseaux virtuels avec différentes couches de protocoles peuvent être exploités en parallèle sur le réseau physique. Par unifier, les interfaces et l'utilisation de logiciels réduisent également la charge administrative aux ressources physiques et matérielles.

La virtualisation peut également être utilisée pour augmenter l'efficacité énergétique. Le matériel

existant peut être utilisé simultanément pour plusieurs tâches, ce qui augmente l'utilisation et exploite de manière optimale les ressources matérielles. L'agrégation et la virtualisation de services distribués physiquement sur une seule instance de matériel physique peuvent même rendre certains matériels obsolètes et donc être arrêtée et supprimés progressivement. L'objectif d'utilisation de la capacité par la virtualisation est comparable à l'âge du mainframe [1], lorsque les ordinateurs mainframe de cette époque étaient aussi saturés que possibles, mais d'autre part, la flexibilité était encore plus grande qu'à l'époque des ordinateurs. La virtualisation offre le meilleur des deux mondes [17].

En conséquence, la virtualisation conduit à une nouvelle adaptabilité qui n'est pas connue dans le contexte de l'utilisation exclusive de matériel physique et de son utilisation. Si la virtualisation masque la complexité structurelle des systèmes physiques, elle augmente également la complexité technique d'un tel système à travers la couche d'abstraction supplémentaire.

2.3 Virtualisation et cloud computing

Sans virtualisation, l'informatique en nuage peut être limitée, voire impensable. La plupart des avantages du cloud computing proviennent de l'utilisation de la virtualisation. Les avantages de la virtualisation se reflètent directement dans le concept de cloud computing. La virtualisation est une technique qui a fait ses preuves depuis longtemps, de sorte que la mise en œuvre par un fournisseur de cloud ne devrait poser aucun problème. Même sans la mise en œuvre du concept de cloud, la virtualisation permet aux entreprises de réaliser d'importantes économies [16]. Il y a également beaucoup de marge de manœuvre, car un cinquième seulement de toutes les petites et moyennes entreprises (PME) utilise la virtualisation. De plus, une transition complète ou incrémentielle vers le cloud computing est plus facile si la virtualisation a déjà été utilisée.

Plus spécifiquement, les environnements en nuage sont appelés «virtualisation de serveur». En général, trois classes de virtualisation peuvent être dérivées, à savoir la division de systèmes physiques individuelles en plusieurs systèmes logiques (partitionnement), la connexion de plusieurs systèmes physiques à des systèmes logiques plus grands (agrégation) ou la mise en correspondance d'architecture de systèmes différentes (émulation). Seules les deux premières classes sont décisives pour le cloud computing. L'émulation est plus susceptible d'être utilisée en privé ou d'utiliser des systèmes plus anciens [18].

2.4 Formes de virtualisation

La virtualisation est possible sous différentes manières. En règle générale, une distinction est établie entre le partitionnement physique, la virtualisation basée sur un hyperviseur, la virtualisation dans

1. Un ordinateur central ou un macroordinateur (mainframe computer), est un ordinateur de grande puissance de traitement et qui sert d'unité centrale à un réseau de terminaux.

un système d'exploitation opérateur, la virtualisation de système d'exploitation et la virtualisation d'application. En outre, il existe des formes spéciales telles que la consolidation et l'agrégation de ressources ou la virtualisation du stockage et du réseau, qui visent la virtualisation des périphériques.

2.4.1 Virtualisation de réseau

La virtualisation des réseaux au moyen de réseaux locaux virtuels (VLAN) et des réseaux privés virtuels (VPN) revêt également une importance particulière pour l'informatique en nuage, notamment grâce aux nouveaux nuages privés virtuels. Les VLAN permettent l'organisation et la distribution flexibles de réseaux virtuels à l'aide d'un réseau physique sous-jacent. Le câblage physique, l'achat supplémentaire de routeurs et de commutateurs supplémentaires et leur configuration éliminent ou sont facilités par l'utilisation de VLAN [19]. À cela s'ajoute la souplesse d'attribution des terminaux aux segments de réseau, quel que soit l'emplacement de la station. Outre l'utilisation efficace d'un réseau local physique, les VLAN permettent également une meilleure protection des sous-réseaux que les sous-réseaux commutés, de sorte que la sécurité est augmentée par leur utilisation. Enfin, les performances peuvent être améliorées en donnant la priorité à certains VLAN [20]. Cependant, un inconvénient est que les VLAN entraînent des efforts accrus dans l'administration du réseau et dans la programmation des composants actifs du réseau.

Grâce à l'utilisation de tunnels cryptés, les VPN permettent la connexion sécurisée de plusieurs ordinateurs ou réseaux via Internet public et sécurisent ainsi l'accès aux réseaux d'organisations externes. Une application typique est l'accès des employés actifs en externe à l'intranet interne de l'entreprise ou même l'accès à des ordinateurs spécifiques, par exemple l'accès via un smartphone ou un ordinateur portable à son propre ordinateur de bureau [19].

2.4.2 Virtualisation du stockage

La virtualisation du stockage, permet une utilisation plus efficace des magasins de données existantes, tels que les disques durs de serveur.

L'idée de base de la virtualisation du stockage est de séparer le magasin de données des serveurs de fichiers classiques et de regrouper le stockage physique. L'espace de stockage peut bien être divisé en systèmes de stockage ou en disques durs, mais ces supports ne doivent pas obligatoirement être physiquement présents. Un logiciel garantit que la division de la mémoire virtuelle correspond à l'espace mémoire physiquement disponible. Les utilisateurs tirent parti de la virtualisation du stockage car ils ne sont pas liés par des limites physiques. Pour les administrateurs systèmes, l'avantage est que les offres de stockage physique existant peuvent être partagées plus efficacement entre les utilisateurs existants, améliorant ainsi les niveaux d'utilisation [21].

2.4.3 Virtualisation basée sur l'hyperviseur

Cette forme de virtualisation implémente une couche de virtualisation proche de la virtualisation sur laquelle les systèmes d'exploitation invités peuvent s'exécuter. En fonction du produit utilisé, certaines modifications du système d'exploitation invité peuvent être nécessaires (paravirtualisation). Un hyperviseur est donc un mécanisme logiciel pour la fourniture de cette couche de virtualisation supplémentaire. Cela inclut le matériel virtualisé, c'est-à-dire les machines virtuelles (VM) et les systèmes d'exploitation invités virtuels qui les exécutent. Les hyperviseurs sont également appelés moniteurs de machines virtuelle, ce qui décrit leur fonction plus clairement.

Dans l'architecture des machines virtuelles, l'élément essentiel est l'hyperviseur. C'est une plateforme de virtualisation qui permet de gérer l'exécution des machines virtuelles. Le concept de ce type de virtualisation est d'émuler le matériel physique pour que chaque machine virtuelle ait l'impression de fonctionner avec son propre matériel. Les deux types d'hyperviseurs qui existent sont représentés ci-dessous. L'hyperviseur du type 1, appelé aussi « barre métal » (« métal nu » en français) est exécuté directement sur le matériel de la machine hôte. Il permet de gérer l'exécution des machines virtuelles et des systèmes d'exploitation invités. Contrairement à l'hyperviseur du type 1, l'hyperviseur du type 2 ne s'exécute pas directement sur le matériel mais à l'intérieur d'un autre système d'exploitation. Dans ce cas, la machine virtuelle et le système d'exploitation invité se situent à un niveau plus élevé par rapport au matériel de la machine physique. La majorité des logiciels de virtualisation fonctionne avec un hyperviseur du type 2. Son avantage est de pouvoir exécuter plusieurs hyperviseurs simultanément étant donné qu'ils ne sont pas directement liés à la couche matérielle.

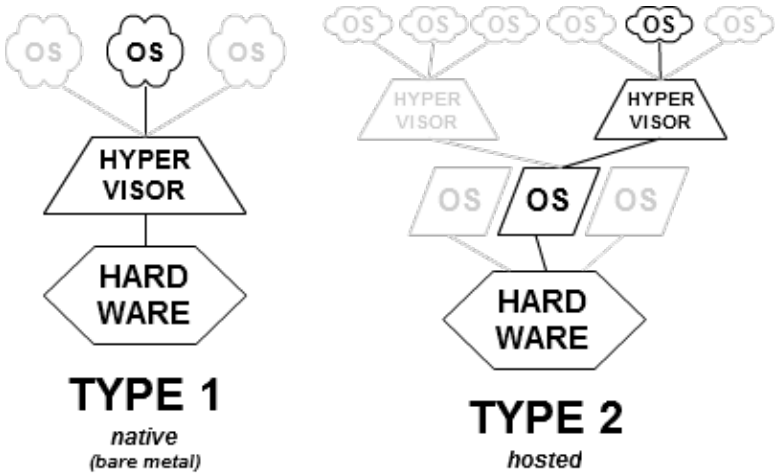


Figure 2.1 : Types d'hyperviseurs [2]

La virtualisation basée sur un hyperviseur est la forme de virtualisation la plus simple et la plus largement utilisée pour les systèmes en nuage. Dans cette forme on trouve les techniques de virtualisation :

2.4.3.1 La virtualisation complète

La virtualisation est dite complète lorsque le système d'exploitation invité n'a pas conscience d'être virtualisé. L'OS qui est virtualisé n'a aucun moyen de savoir qu'il partage le matériel avec d'autres OS. Ainsi, l'ensemble des systèmes d'exploitation virtualisés s'exécutant sur un unique ordinateur, peuvent fonctionner de manière totalement indépendante les uns des autres et être vu comme des ordinateurs à part entière sur un réseau.

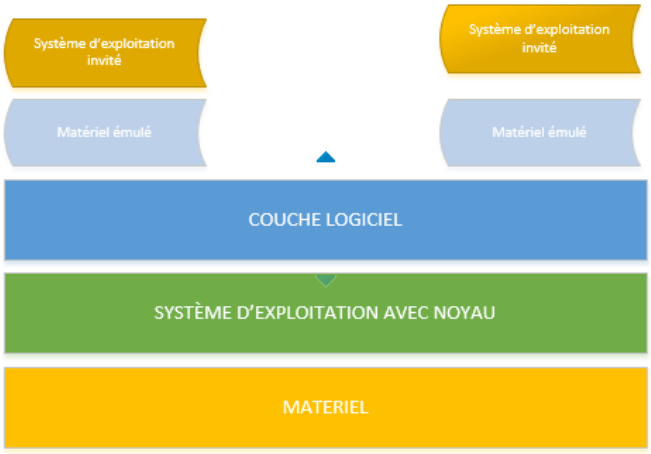


FIGURE 2.2 : Virtualisation Complète

2.4.3.2 La Paravirtualisation

Par opposition à la virtualisation, on parle de paravirtualisation lorsque les systèmes d'exploitation doivent être modifiés pour fonctionner sur un hyperviseur de paravirtualisation. Les modifications sont en fait des insertions de drivers permettant de rediriger les appels système au lieu de les traduire.

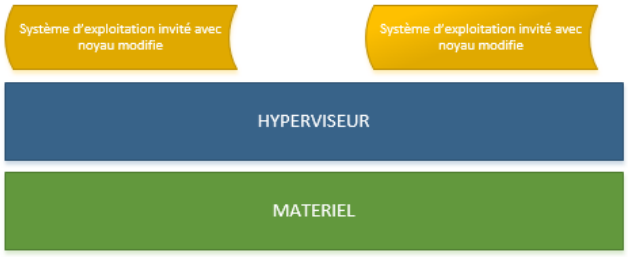


FIGURE 2.3 : Paravirtualisation

2.4.3.3 La virtualisation assisté par le matériel

La technique de virtualisation assistée par matériel améliore la flexibilité et la fiabilité des solutions logicielles traditionnelles en accélérant les fonctions clés de la plate-forme virtualisée.Cette efficacité offre notamment des avantages dont : Accélération du transfert du contrôle de la plate-forme entre les

systèmes d'exploitation invités et le gestionnaire.

Des exemples de technologies de virtualisation assistée par le matériel sont :Intel Tm (VT : intel Virtualisation Technologie) et AMD (AMD-V : Industry Leading virtualisation Platform Efficiency) dont les processeurs récents en disposent.

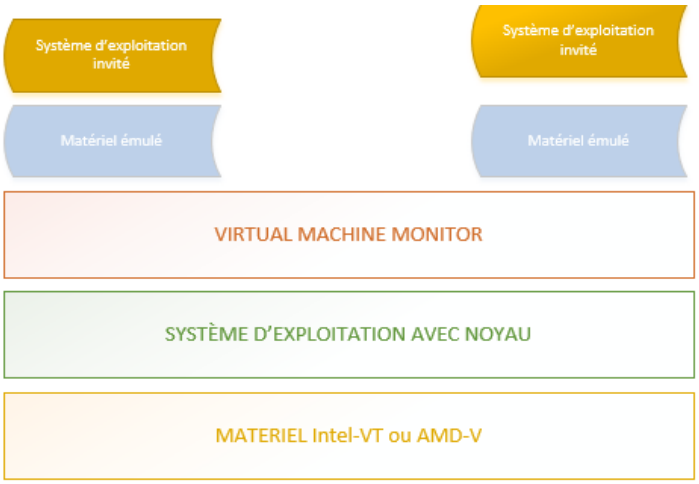


FIGURE 2.4 : Virtualisation assistée par le matériel

2.4.4 Conteneurs

On peut définir les conteneurs comme un ensemble de processus visant à offrir des outils de gestion de ressources tout en assurant leur isolement au niveau du noyau et empêchant l'interférence entre eux.

Un conteneur contient une image et les fichiers qui composent l'application qui va être exécutée à l'intérieur. Son principe de base est l'empaquetage qui rend le déploiement des applications plus facile. De plus, il consomme une quantité limitée de ressources afin de satisfaire les demandes des clients. Souvent, il est utilisé dans les micro-services et il peut être connecté à d'autres conteneurs ou services via le réseau.

Parmi les exemples de conteneurs qu'on peut citer : le conteneur Linux LXC. Grâce aux conteneurs Linux ou bien la technique Docker, le code d'application conteneurisée est totalement isolé du SE (Système d'Exploitation) de la machine hôte et des autres conteneurs, ce qui donne au développeur plus de souplesse de développement et déploiement au niveau de l'application.

2.4.4.1 Conteneurs et machines virtuelles

Pour savoir quels sont les cas d'utilisation adaptés à la virtualisation basée sur des conteneurs ou à la virtualisation avec des machines virtuelles, nous allons analyser les forces et les faiblesses de ces deux

façons de mettre en œuvre la virtualisation.

La virtualisation avec des conteneurs se différencie de la virtualisation «classique » avec des machines virtuelles . Dans la virtualisation, l'hyperviseur gère les ressources de la machine hôte. Il est donc possible de partager les ressources mémoires car les machines virtuelles ne consomment pas la totalité de leur mémoire allouée. Les machines virtuelles sont composées d'un système d'exploitation complet avec leur propre gestion de mémoire, pilotes de périphériques et bibliothèques. Par contre, les conteneurs partagent le même système d'exploitation, celui de la machine hôte et également des bibliothèques si elles sont communes à plusieurs conteneurs. Par conséquent, les conteneurs représentent une solution de virtualisation plus légère et portable car ils utilisent moins de ressources que les machines virtuelles. D'ailleurs, les conteneurs sont souvent appelés « machines virtuelles légères » car les deux concepts ont un même objectif qui est de virtualiser des systèmes. La principale différence entre les machines virtuelles et les conteneurs est leur architecture.

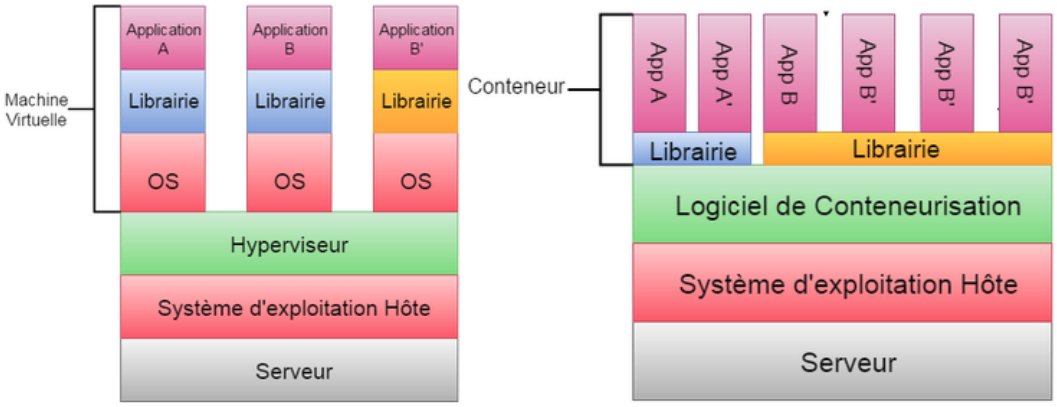


FIGURE 2.5 : Machines virtuelles vs Conteneur [3]

Le tableau suivant résumé les principales différences de ces deux approches :

TABLE 2.1 : Machine virtuelle vs Conteneur

Caractéristiques	Machines virtuelle	Conteneur
Système d'exploitation invité	Chaque machines virtuelle est exécutée directement sur le matériel	Tous les systèmes d'exploitation invités partagent le même système d'exploitation hôte et le même noyau linux
Sécurité	Plus sécurisée car chaque machine virtuelle a son propre noyau de systèmes d'exploitation	Vulnérable à cause du partage du noyau linux par les différents conteneurs
Performance	Performances limitées à cause du temps de latence entre le système d'exploitation invité et hôte	Performances proches du système d'exploitation hôte
Isolation	Partage de bibliothèques et de fichiers entre les invités et l'hôte impossible	Les conteneurs sont isolés mais peuvent partager un système d'exploitation, des bibliothèques ou des fichiers
Temps de démarrage	Quelques minutes	Quelques secondes
Stockage	Beaucoup de mémoires car tout le système d'exploitation et ses programmes sont installés et exécutés	Les conteneurs prennent peu de mémoire car le système d'exploitation hôte est partagé
Déploiement	Lourd et difficile	Léger et facile

2.4.4.2 Types de conteneurs

Conteneurs machines Les conteneurs machines sont des environnements virtuels qui partagent le noyau du système d'exploitation hôte mais fournissent une isolation de l'espace utilisateur. Ils ressemblent beaucoup plus aux machines virtuelles. Ils ont leur propre processus init et peuvent exécuter un nombre limité de démons. Les programmes peuvent être installés, configurés et exécutés exactement comme sur n'importe quel système d'exploitation invité. Semblable à une machine virtuelle, tout ce qui est exécuté dans un conteneur ne peut voir que les ressources qui ont été attribuées à ce conteneur. Les conteneurs machines sont utiles lorsque le cas d'utilisation consiste à gérer un parc de distributions de flavors identiques ou différents [4].

Les conteneurs machines dotés de leur propre système d'exploitation ne signifient pas qu'ils exécutent une copie intégrale de leur propre noyau. Au lieu de cela, ils exécutent quelques démons légers et ont un certain nombre de fichiers nécessaires pour fournir un système d'exploitation distinct dans un

autre système d'exploitation. Les technologies de conteneur telles que LXC, OpenVZ, Linux vServer, BSD Jails et les zones Solaris conviennent toutes à la création de conteneurs de machines[4].

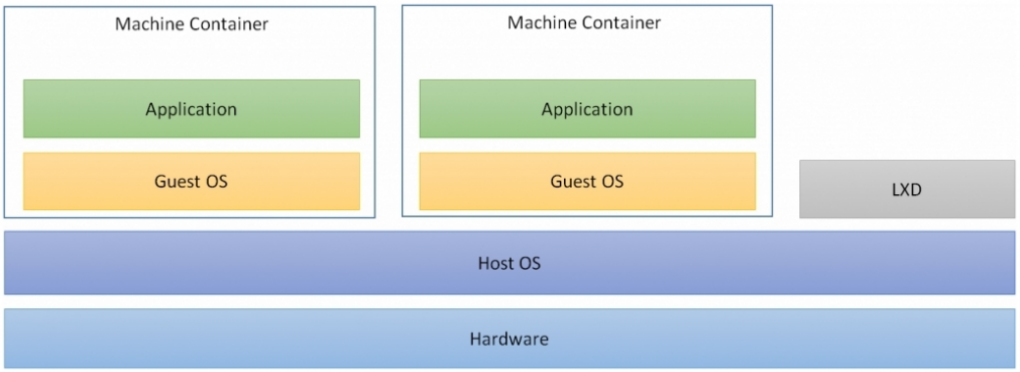


FIGURE 2.6 : Conteneurs machines [4]

Conteneurs d'applications Alors que les conteneurs machines sont conçus pour exécuter plusieurs processus et applications, les conteneurs d'applications sont conçus pour empaqueter et exécuter une seule application. Ils sont conçus pour être très petits. Ils n'ont pas besoin de contenir un shell ou un processus init. L'espace disque requis pour un conteneur d'applications est très petit. Les technologies de conteneur telles que Docker et Rocket sont des exemples de conteneurs d'application[4].

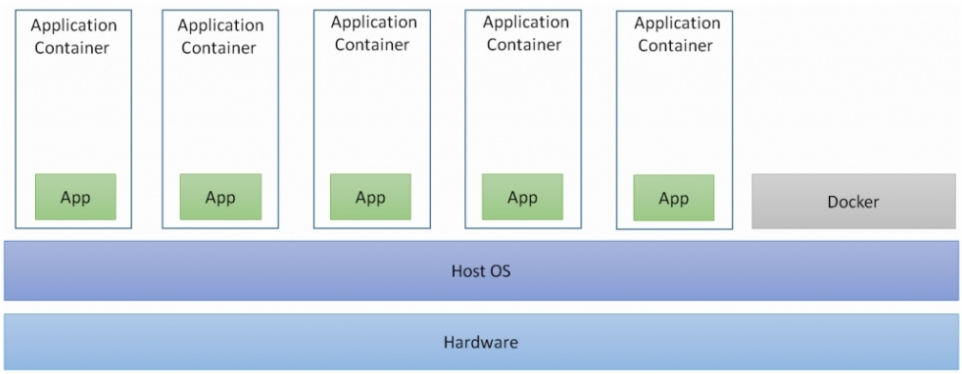


FIGURE 2.7 : Conteneurs d'application [4]

2.5 Conclusion

Dans ce chapitre nous avons présenté l'un des principaux techniques de Cloud Computing qui est la virtualisation, ces avantages, ces formes et les principales différences entre machine virtuelle et conteneur.

Chapitre 3

Openstack et Docker

3.1 Openstack

3.1.1 Introduction

OpenStack est un logiciel libre et open source permettant de créer des clouds privés et publics. Il fournit des ensembles interdépendants de composants permettant de gérer et d'accéder à de vastes pools de ressources de calcul, de réseau et de stockage répartis dans un centre de données. Les utilisateurs peuvent le gérer en utilisant des interfaces utilisateur et des lignes de commande basées sur le Web ou des REST APIs [1]. OpenStack a été open sourcer en 2010 par Rackspace [2] et la NASA [3]. Actuellement, il est géré par le OpenStack Fondation : une entité à but non lucratif[22].

3.1.2 Composants principales d'OpenStack

Les projets principaux sont des composants nécessaires d'OpenStack pour la fourniture d'infrastructure en tant que service. En pratique, sans l'un des composants principaux énumérés ci-dessus, Openstack ne peut pas approvisionner la machine virtuelle pour l'utilisateur.

3.1.2.1 Keystone (identity service)

Keystone agit tel un annuaire qui centralise toutes les authentifications et autorisations nécessaires aux multiples services d'OpenStack. Il peut aussi servir de catalogue de service[23].

1. Une API REST est une interface de programme d'application (API) qui utilise des requêtes HTTP pour les données GET, PUT, POST et DELETE.

2. Rackspace Inc. est une société américaine de cloud computing gérée basée à Windcrest, au Texas, dans la banlieue de San Antonio.

3. La National Aeronautics and Space Administration, Est l'agence gouvernementale qui est responsable de la majeure partie du programme spatial civil des États-Unis.

3.1.2.2 Nova (computing service)

Nova est le composant permettant la gestion de machines virtuelles. Nova ne fait pas office d'hyperviseur et ne peut donc pas créer d'instances par lui-même, au lieu de cela, il s'appuie sur un ou plusieurs hyperviseurs configurés pour fonctionner avec lui[23].

3.1.2.3 Glance (image service)

Glance permet la découverte, la sauvegarde et la récupération des images de disques durs utilisées par les machines virtuelles[23].

3.1.2.4 Neutron (networking service)

Neutron est le composant OpenStack offrant le "réseau en tant que service". Il permet entre autres de créer des réseaux, d'assigner des adresses IPs aux machines virtuelles dans ces réseaux, mais aussi de créer des accès VPN, des balancements de charges, des pare-feux,etc[23].

3.1.3 Composants optionnels

Sont les projets qui ajouteront une fonctionnalité étendue à la plate-forme IaaS dans OpenStack. Ce composant est classé dans des projets supplémentaires d'OpenStack, principalement parce que le projet n'est pas nécessairement requis dans OpenStack pour fournir une plate-forme IAAS de base. parmi ses composants :

3.1.3.1 Cinder (block storage service)

Cinder fournit le support de volumes (disques) persistants pour les machines virtuelles[23].

3.1.3.2 Swift (object storage service)

OpenStack Object Storage (Swift) est un système de stockage redondant évolutif. Les objets et les fichiers sont écrits sur plusieurs lecteurs de disque répartis sur les serveurs du centre de données, le logiciel OpenStack étant chargé de garantir la réplication et l'intégrité des données sur l'ensemble du cluster[23].

En août 2009, Rackspace a lancé le développement du précurseur d'OpenStack Object Storage, en remplacement complet du produit Cloud Files. L'équipe de développement initiale était composée de neuf développeurs. SwiftStack, une société de logiciels de stockage d'objets, est actuellement le principal développeur de Swift, avec des contributions importantes de HP, Red Hat, NTT, NEC, IBM, etc[24, 23].

3.1.3.3 Horizon

Horizon offre une interface utilisateur à travers un navigateur, permettant de gérer les services

3.1.4 Architecture logique

La figure suivante représente l'architecture logique d'OpenStack et explique comment les utilisateurs peuvent se connecter à divers services. OpenStack comporte plusieurs composants à des fins différentes, tels que Nova pour la gestion des ressources de calcul, Glance pour la gestion des images de système d'exploitation, etc. Nous étudierons chaque composant en détail dans les prochaines sections.

En termes très simples, si un utilisateur demande de provisionner une machine virtuelle à l'aide de la CLI ou des APIs [1], la demande est traitée par Nova. Nova parle ensuite à KeyStone pour authentifier la demande, Glance pour l'image du système d'exploitation et Neutron pour la configuration des ressources réseau. Ensuite, après avoir reçu les réponses de chaque composant, il démarre la machine virtuelle et renvoie une réponse à l'utilisateur[6]

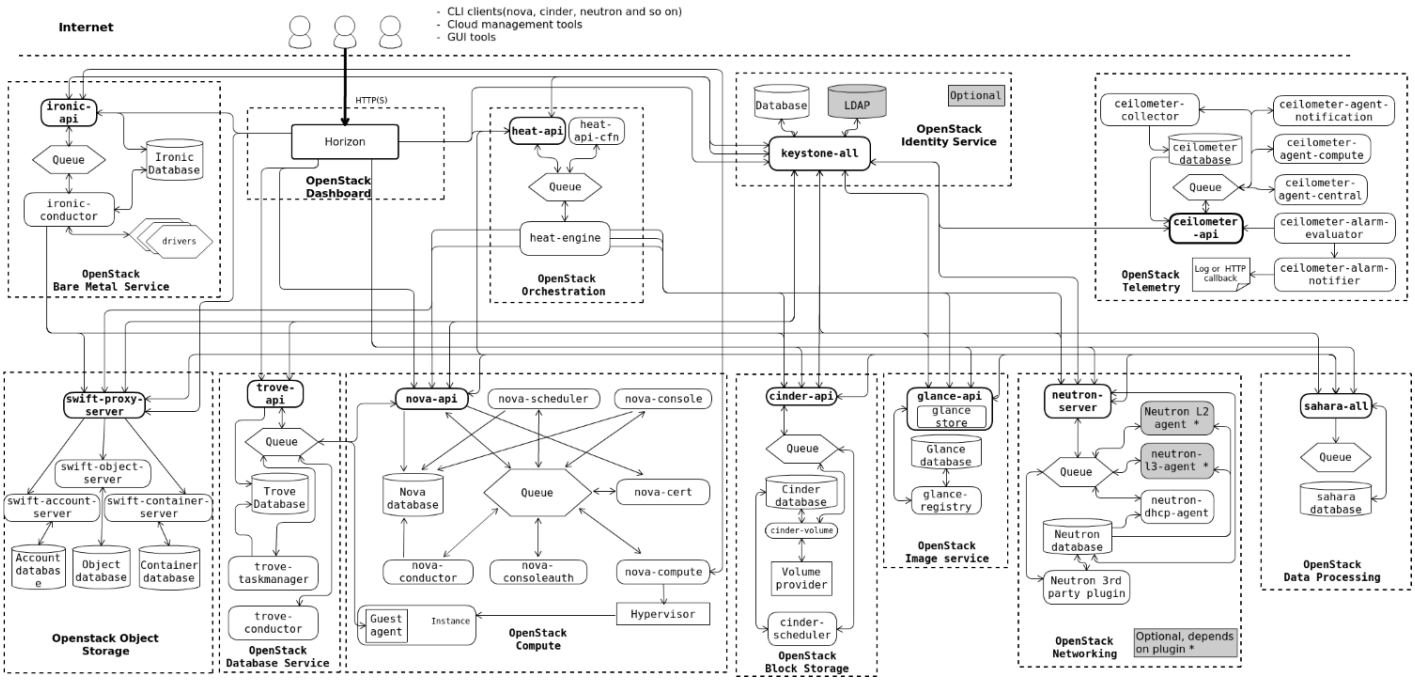


FIGURE 3.1 : Architecture logique OpenStack [5]

3.1.4.1 Compute service - Nova

Le service de calcul se compose de divers services et démons, notamment le service nova-api, qui accepte les demandes d'API [1] et les transmet aux autres composants du service.Le diagramme suivant

montre comment différents processus et démons travaillent ensemble pour former le service de calcul (Nova) et les interconnecter entre eux :

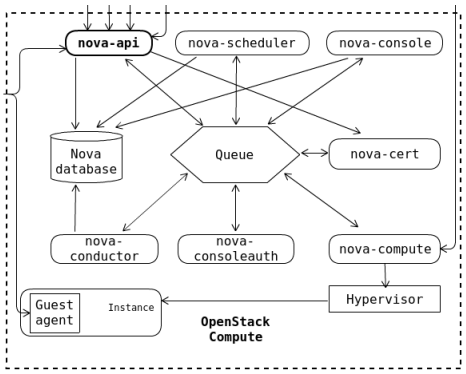


FIGURE 3.2 : compute Service Nova [6]

Le calcul OpenStack comprend les services et les démons suivants.

nova-api service Tous les services d'OpenStack ont au moins un processus d'API [1] qui accepte et répond aux appels d'API de l'utilisateur final, les traite en préprocessus et transmet la demande au processus approprié du service. Pour le service de calcul OpenStack, nous avons le service nova-api, qui écoute et répond aux appels d'API de calcul de l'utilisateur final. Le service nova-api prend en charge le démarrage de la plupart des activités d'orchestration, telles que le provisionnement de nouvelles machines virtuelles[6].

nova-api-metadata service Le service de métadonnées fournit les données spécifiques à l'instance aux instances de la machine virtuelle. Les données spécifiques à l'instance comprennent le nom d'hôte, l'identificateur d'instance, les clés ssh, etc[6].

nova-compute service L'intégralité du cycle de vie de la machine virtuelle est gérée par les hyperviseurs. Chaque fois que l'utilisateur final soumet l'appel d'API [1] de création d'instance au service nova-api, le service nova-api le traite et transmet la demande au service nova-compute. Le service nova-compute traite l'appel nova-api pour la création d'une nouvelle instance et déclenche la demande d'API appropriée pour la création d'une machine virtuelle d'une manière compréhensible pour l'hyperviseur sous-jacent[6].

nova-scheduler service Lorsqu'on a plusieurs nœuds de calcul dans l'environnement OpenStack, le service nova-scheduler se chargera de déterminer où la nouvelle machine virtuelle sera provisionnée. Selon les différents filtres de ressources, tels que RAM / CPU / Disque / disponibilité, la nova-scheduler filtrera l'hôte de calcul approprié pour la nouvelle instance :

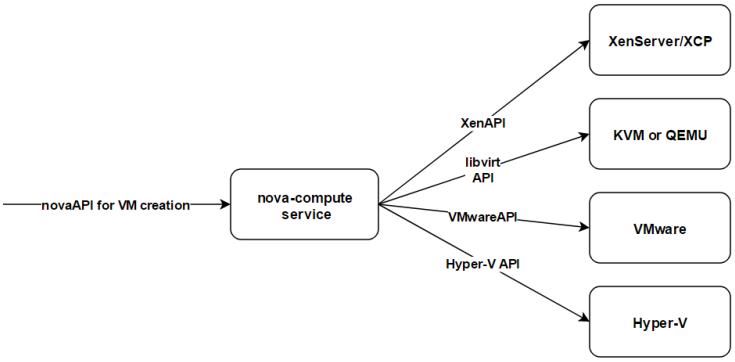


FIGURE 3.3 : nova scheduler service [6]

nova-conductor module Le service nova-conductor qui s'exécute sur l'hôte de calcul n'a pas d'accès direct à la base de données, car si l'un des nœuds de calcul est compromis, l'attaquant dispose d'un accès (presque) total à la base de données. Avec le démon nova-conductor, le nœud compromis ne peut pas accéder directement à la base de données et toutes les communications ne peuvent passer que par le démon nova-conductor. Ainsi, le nœud compromis est maintenant limité dans la mesure où les API de conducteur le permettent.

Le module nova-conductor ne doit pas être déployé sur des nœuds de calcul, sinon l'objectif de suppression de l'accès direct à la base de données pour le calcul nova deviendra invalide[6].

nova-consoleauth daemon Le démon nova-consoleauth se charge d'autoriser les jetons des utilisateurs finaux à accéder à une console distante des machines virtuelles invitées fournies par les mandataires de contrôle suivants : le démon nova-novncproxy fournit un proxy pour accéder aux instances en cours d'exécution via une connexion VNC [4]. Le démon nova-spicehtml5proxy fournit un proxy via une connexion SPICE [5]

nova-cert module Utilisé pour générer des certificats X509 [6] pour euca-bundle-image [7], nécessaire uniquement pour l'API EC2[6].

4. VNC (Virtual Network Computing, littéralement « informatique virtuelle en réseau ») est un ... installé sur n'importe quel système d'exploitation peut se connecter à un serveur VNC installé sur un système d'exploitation différent ou identique.

5. SPICE (acronyme anglais de « Simple Protocol for Independent Computing Environments », signifiant « Protocole simple pour des environnements informatiques indépendants ») est un protocole de communication, utilisé en général en virtualisation, pour visualiser et gérer à distance un système d'exploitation.

6. X.509 est une norme spécifiant les formats pour les certificats à clé publique, les listes de révocation de certificat, les attributs de certificat, et un algorithme de validation du chemin de certification, définie par l'Union internationale des télécommunications (UIT)

7. http ://docs.eucalyptus.cloud/eucalyptus/4.4.2/euca2ools-guide/euca-bundle-image.html

The queue (AMQP message broker) En règle générale, la file d'attente de messages AMQP [8] est implémentée avec RabbitMQ où Zero Mq. Dans OpenStack, un courtier de messages AMQP [8] est utilisé pour toutes les communications entre les processus et les démons d'un service. Cependant, la communication entre les deux services différents dans OpenStack utilise des points de terminaison de service(endpoint)[6].

Par exemple, Nova-api et Nova-scheduler communiquent via le courtier de messages AMQP [8]. Cependant, la communication entre le service nova-api et le service cinder-api se fera par les points d'extrémité du service.

Database La plupart des services OpenStack utilisent une base de données SQL pour stocker les états de construction et d'exécution d'une infrastructure de cloud, tels que l'état de l'instance, les réseaux, les projets, etc[6].

3.1.4.2 Image service - Glance

Le diagramme d'architecture logique suivant montre comment différents processus et démons fonctionnent ensemble pour effectuer le service d'imagerie (Glance) dans OpenStack, ainsi que leur interconnexion :

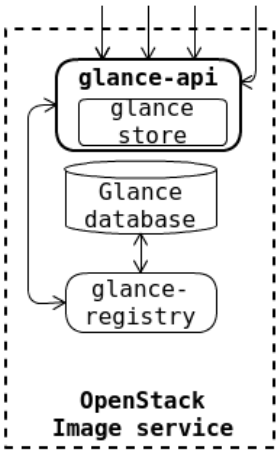


FIGURE 3.4 : Glance image service [6]

glance-api service Pour le service d'imagerie OpenStack, le service glance-api traite les appels à l'API [1] de l'image pour la découverte, la récupération et le stockage des images.

glance-registry service Au fur et à mesure que son nom se lit, le service de registre se charge de stocker, de traiter et de récupérer les métadonnées relatives aux images. Notamment, le service de

8. AMQP (pour Advanced Message Queuing Protocol) est un protocole ouvert pour les systèmes de messagerie orientés intergiciel, son objectif est de standardiser les échanges entre serveurs de messages en se basant sur les principes suivants : orienté message, utilisation de files d'attente, routage (point à point et publish-subscribe), fiabilité et sécurité.

registre ne traite que les métadonnées de l'image, pas l'image elle-même. Les métadonnées incluent des informations sur l'image telle que la taille et le type[6].

3.1.4.3 Identity service - KeyStone

Le diagramme d'architecture logique suivante montre comment le service d'identité dans OpenStack est conçu pour traiter l'authentification et l'autorisation :

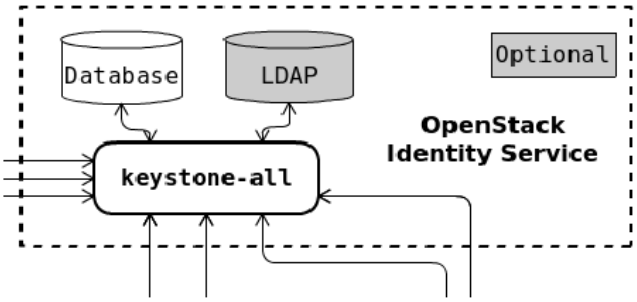


FIGURE 3.5 : Keystone identity service [6]

À l'exception de tous les autres services OpenStack, le service d'identité ne dispose d'aucun service d'API [1] dédié pour traiter et écouter les appels d'API. Cependant, le travail réel est effectué par des processus distincts[6].

Server (keystone-all) Un serveur centralisé traite les demandes d'authentification et d'autorisation à l'aide d'une interface RESTful [1].

Drivers (optional) À l'aide des pilotes d'arrière-plan, le serveur Keystone centralisé peut être intégré au système de service sélectionné, tel que les serveurs LDAP [9], Active directory [10] et les bases de données SQL. Ils peuvent être utilisés pour accéder aux informations d'identité à partir des référentiels communs externes à Open stack[6].

3.1.4.4 Neutron server

Le serveur neutron accepte et achemine les requêtes API [1] vers le plug-in neutron OpenStack approprié pour traiter la requête. Le service du serveur à neutrons s'exécute sur le nœud du réseau (dans la plupart des cas, le nœud du réseau est combiné au nœud du contrôleur).

9. LDAP (Lightweight Directory Access Protocol) est un protocole standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur les utilisateurs d'un réseau par l'intermédiaire de protocoles TCP/IP.
10. AD sont des services d'annuaire LDAP pour les systèmes d'exploitation Windows.

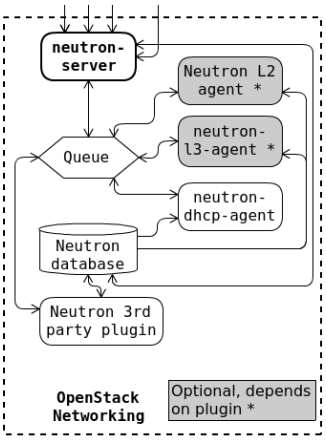


FIGURE 3.6 : Neutron server [6]

plugin agent (neutron-*-agent) Alors que le serveur neutron agit en tant que contrôleur centralisé s'exécutant sur le nœud du réseau, l'agent neutron s'exécute sur le nœud de calcul pour gérer la configuration du commutateur virtuel local (vswitch). De plus, les agents reçoivent des messages et des instructions du serveur Neutron (via des plugins ou directement) sur le bus de messages AMQP, puis les commandes et la configuration liées à la mise en réseau sont exécutées par l'agent neutron[6].

DHCP agent (neutron-dhcp-agent) La mise en réseau OpenStack est très similaire à la mise en réseau dans le monde réel. Les machines virtuelles nécessitent une connectivité réseau de couche 2 (L2) de manière minimale. Neutron-dhcp-agent agit en tant que serveur DHCP sur le réseau client et s'occupe de fournir une adresse IP DHCP à l'instance[6].

L3 agent (neutron-l3-agent) Comme, on l'a mentionné précédemment,dans le monde réel,un poste de travail a besoin d'un accées internet/réseau externe. Dans ce cas, routeur (fonctionnalité L3) est utilisé. De méme, Dans la mise en réseau OpenStack, l'agent neutron-l3 pour fournir une fonctionnalité de transfert L3 / NAT aux machines virtuelles[6].

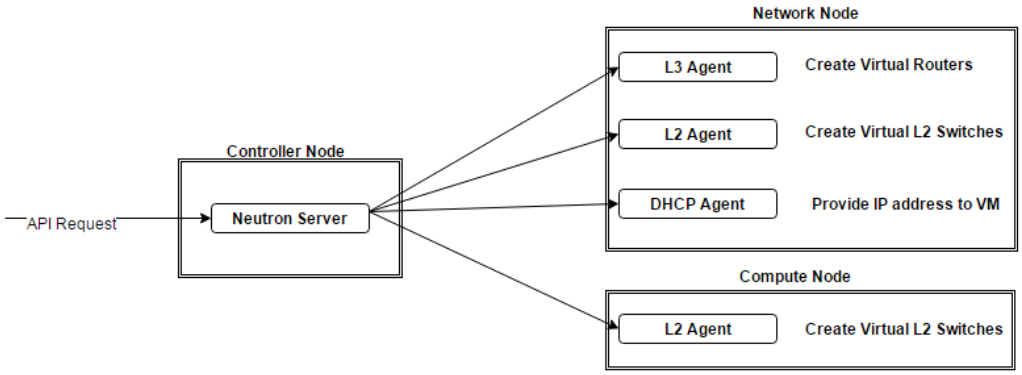


FIGURE 3.7 : neutron-l3-agent [6]

3.1.4.5 Block storage service - Cinder

Le diagramme d'architecture logique montre comment différents processus et démons travaillent ensemble pour exécuter le service de stockage de blocs (Cinder) dans Open Stack, ainsi que leur interconnexion :

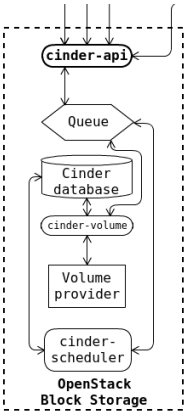


FIGURE 3.8 : Block storage service - Cinder [6]

cinder-api service Pour le service de stockage en mode bloc, le service cinder-api accepte les demandes d'API [1] et les achemine vers le volume cinder pour qu'elles soient traitées[6].

cinder-scheduler daemon Semblable à nova-scheduler pour le service de calcul, cinder-scheduler joue le même rôle que le service de stockage en mode bloc. Le démon cinder-scheduler planifie et achemine les demandes vers le service de volume approprié en fonction des filtres dans les paramètres de configuration, tels que la capacité de stockage, les zones de disponibilité, les types de volume et les fonctionnalités[6].

cinder-volume Service Après avoir filtré et choisi le service de volume approprié, le planificateur cinder acheminera la demande de stockage aux services cinder-volume pour les traiter. Ensuite, le service cinder-volume interagit directement avec le service de stockage en mode bloc pour fournir un espace de stockage requis. Notamment, le service cinder-volume pourrait interagir avec plusieurs fournisseurs de stockage en même temps via les différents pilotes de périphérique disponibles[6].

cinder-backup daemon Chaque fois qu'une demande est reçue pour des opérations de sauvegarde / restauration de volume, le démon cinder-backup interagit avec les cibles de sauvegarde, telles que le stockage d'objets Swift ou le magasin de fichiers NFS [11], pour la sauvegarde ou la restauration de volumes de tous types ;

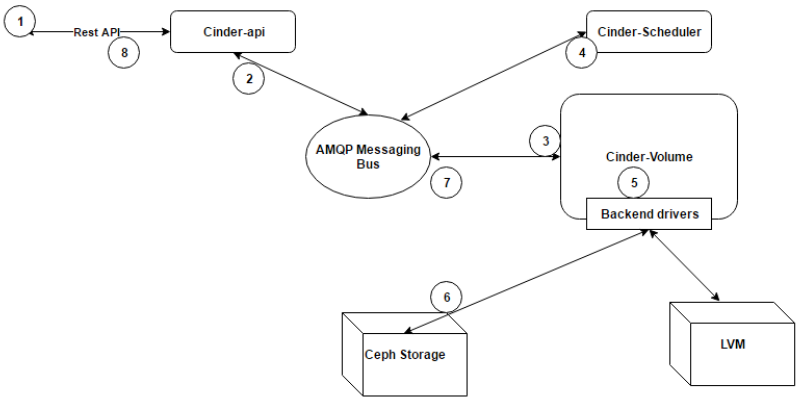


FIGURE 3.9 : cinder-backup daemon [6]

3.1.4.6 Object storage - Swift

Comme mentionné précédemment, le stockage d'objet est très similaire au stockage en nuage, tel que Google Drive ou Dropbox. En tant qu'utilisateur final, nous ne pouvons utiliser que la fenêtre de l'utilisateur final de Google Drive ou Dropbox pour stocker et récupérer les fichiers. Cependant, les actions effectuées par le groupe de processus et de démons derrière l'écran de l'utilisateur final pour enregistrer et récupérer le fichier ont une architecture de stockage d'objet très complexe.

Le service Swift est assez différent des autres services OpenStack car nous pouvons configurer les services Swift en tant que services autonomes pour fournir uniquement les services de stockage d'objets aux utilisateurs finaux, sans configurer les fonctionnalités IaaS.

Pour OpenStack, le service Swift est classé dans les services supplémentaires, et non dans le service principal, car OpenStack a pour objectif principal de fournir aux utilisateurs finaux une infrastructure en tant que service (IaaS). Toutefois, Swift n'est pas un service obligatoire pour activer la fonctionnalité IAAS[6].

11. NFS est un système de fichiers en réseau

Le diagramme d'architecture logique suivante montre comment différents processus et démons travaillent ensemble pour faire apparaître le service de stockage d'objets (Swift) dans OpenStack et l'interconnexion entre les services :

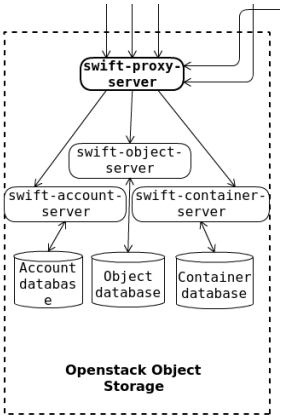


FIGURE 3.10 : Object storage - Swift [6]

Proxy servers (swift-proxy-server) Pour le stockage d'objets Swift, on dispose d'un service de serveur proxy qui accepte les API [1] de stockage d'objets OpenStack et les requêtes HTTP brutes. Les requêtes API et HTTP incluent le téléchargement de fichiers, la création de dossiers (conteneurs) et la modification des métadonnées[6].

Account servers (swift-account-server) Le service de serveur de comptes gère la demande de traitement des informations de métadonnées des comptes individuels et de la liste des conteneurs mappés pour chaque compte[6].

Container servers (swift-container-server) Le serveur de conteneur gère les requêtes concernant les métadonnées de conteneurs et la liste des objets contenus dans chaque conteneur. Les objets stockés dans le conteneur ne contiennent aucune information sur l'emplacement de stockage réel, mais sur le conteneur particulier dans lequel les objets sont stockés[6].

Object servers (swift-object-server) Le serveur d'objets est responsable de la gestion des objets réels, tels que les fichiers, sur les nœuds de stockage[6].

Telemetry service - ceilometer Le service secondaire OpenStack qui gère le comptage de toutes les autres ressources utilisées par le service OpenStack est le service ceilometer. Avec l'aide d'agents, le service Ceilometer collectera des charges et des données de mesure relatives à l'utilisation du service Openstack. Les métriques collectées peuvent être utilisées pour facturer la ressource, mais également pour déclencher les alarmes lorsque les métriques obtenues ou les données d'événement dépassent le seuil défini. Pour de meilleures performances, le service ceilometer est généralement configuré avec

une base de données NoSQL dédiée, telle que MongoDb, où, comme les autres services OpenStack, utilisent des bases de données SQL telles que MySql :

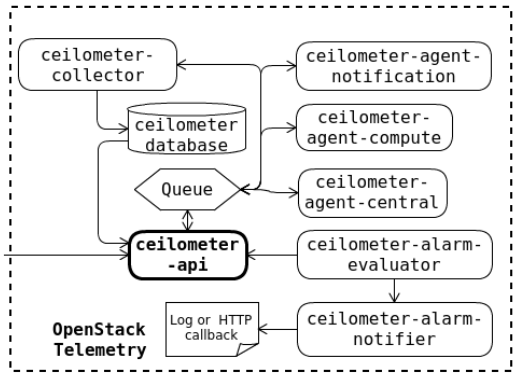


FIGURE 3.11 : Telemetry service - ceilometer [6]

Ceilometer peut être utilisé pour les alarmes et la facturation. À partir de la version OpenStack de Newton, la fonctionnalité d'alarme du service de céilomètre est découplée et ajoutée dans un nouveau projet OpenStack appelé adh (service d'alarme de télémétrie)[6].

ceilometer-agent-compute L'agent ceilometer s'exécute sur le nœud de calcul pour collecter les statistiques d'utilisation des ressources de l'hôte et des machines virtuelles à des intervalles d'interrogation réguliers et envoyer les statistiques collectées au collecteur de céilomètre pour les traiter[6].

ceilometer-agent-central Outre les métriques collectées à partir du nœud de traitement, ceilometer-agent-central se charge de collecter les statistiques d'utilisation des ressources à des intervalles d'interrogation réguliers à partir des autres services OpenStack tels que Glance, Cinder et neutron[6].

ceilometer-agent-notification Contrairement aux deux autres agents de céilomètre, l'agent de notification ne fonctionne pas avec la méthode d'interrogation. Cependant, lorsqu'une nouvelle action est effectuée par un service Open Stack, l'incident sera communiqué via le bus AMQP [8]. L'agent Ceilometer-notification surveille les files de messages en attente de notifications et consomme les messages générés sur le bus de notification, puis les transforme en données ou événements de mesure Ceilometer[6].

ceilometer-collector Comme son nom l'indique, le collecteur de céilomètre s'exécute sur le serveur de gestion central et collecte les données de mesure de tous les agents de céilomètre mentionnés précédemment. Les données de mesure collectées sont stockées dans un magasin de données ou peuvent être configurées pour être envoyées au service de surveillance externe, tel que le service de surveillance Nagios[6].

3.1.4.7 Orchestration service - Heat

Le service heat s'occupe de l'organisation et de la coordination des tâches automatisées, ce qui aboutit au flux de travail de processus permettant de gérer le cycle de vie complet de l'infrastructure et des applications dans les clouds OpenStack. Les codes de format lisible par un humain dans un fichier texte, à savoir des modèles actifs, sont utilisés par le service heat pour gérer le cycle de vie des ressources dans le cloud[6].

En termes simples, toute séquence d'activités exécutées par l'utilisateur final à l'aide d'OpenStack Horizon peut être codée dans un modèle, puis les mêmes opérations peuvent être effectuées par le service heat en traitant ce modèle dans un moteur heat-engine

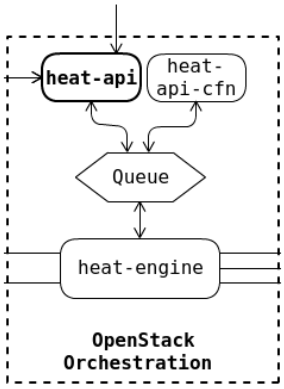


FIGURE 3.12 : Orchestration service - Heat [6]

heat-api component Le service heat-api accepte l'API REST [1] native d'OpenStack et y répond, puis transmet la demande au moteur thermique via un appel de procédure distante[6].

heat-api-cfn component heat-engine est également compatible avec les modèles AWS Cloud Formation. L'API-cfn thermique accepte la demande d'API [1] et l'envoie au moteur thermique via RPC [12] pour la traiter[6].

heat-engine heat-engine gère l'orchestration des modèles et en rend compte au consommateur d'API[6].

12. En informatique et en télécommunication, RPC (remote procedure call) est un protocole réseau permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications.

3.2 Docker

3.2.1 Historique

Docker a été développé par Solomon Hykes [13] pour un projet interne de dot Cloud, une société proposant une plate-forme en tant que service (Paas), avec les contributions d'Andrea Luzzardi [14] et Francois-Xavier Bourlet, également employés de dot Cloud. Docker est une évolution basée sur les techniques propriétaires de dot Cloud, elles-mêmes construites sur des projets open source. Docker a été distribué en tant que projet open source à partir de mars 2013[25].

3.2.2 Présentation de Docker

Docker est une technique de virtualisation de conteneur. Donc, c'est comme une machine virtuelle très légère[26]

Techniquement, Docker étend le format de conteneur Linux standard, LXC, avec une API [1] de haut niveau fournissant une solution pratique de virtualisation qui exécute les processus de façon isolée[27]. Pour arriver à ses fins, Docker utilise entre autres LXC, cgroups et le noyau Linux lui-même[28]. Contrairement aux machines virtuelles traditionnelles, un conteneur Docker n'inclut pas de système d'exploitation, mais s'appuie au contraire sur les fonctionnalités du système d'exploitation fournies par la machine hôte[29].

La technique de conteneur de Docker peut être utilisée pour étendre des systèmes distribués de façon qu'ils s'exécutent de manière autonome depuis une seule machine physique ou une seule instance par nœud. Cela permet aux nœuds d'être déployés au fur et à mesure que les ressources sont disponibles, offrant un déploiement transparent et similaire aux Paas pour des systèmes comme Apache Cassandra [15], Riak [16] ou d'autres systèmes distribués[30].

3.2.3 Intérêt de Docker

Docker fournit ce qu'on appelle une abstraction. Les abstractions permettent de travailler avec des choses compliquées en termes simplifiés. Ainsi, dans le cas de Docker, au lieu de se concentrer sur

13. Solomon Hykes, né à New York en 1983, est un créateur d'entreprises franco-américain, vivant dans la Silicon Valley, et le créateur d'un concept à l'origine d'un ensemble de logiciels, Docker, qui suscite l'intérêt de nombreuses sociétés informatiques.

14. Andrea Luzzardi, responsable technique des systèmes distribués, présente Docker et présente les fonctionnalités d'orchestration, notamment un aperçu de Docker Swarm.

15. http ://cassandra.apache.org/

16. Riak est un système de gestion de base de données distribué, scalable de manière linéaire, hautes performances, sans schéma et orienté clé-valeur. Riak est écrit avec les langages de programmation Erlang, C et JavaScript, distribué sous licence Apache et inspiré de Dynamo.

toutes les complexités et les spécificités associées à l'installation d'une application, on ne prends en compte que les logiciels qu'on souhaite installer. Comme une grue chargeant un conteneur d'expédition sur un navire, l'installation de tout logiciel avec Docker est identique à celle de n'importe quel autre. La forme ou la taille de la chose à l'intérieur du conteneur d'expédition peut varier, mais la façon dont la grue saisit le conteneur sera toujours la même. Tout l'outillage est réutilisable pour n'importe quel conteneur d'expédition[31].

C'est également le cas pour la suppression d'application. Lorsqu'on souhaite supprimer un logiciel, on indique simplement à Docker le logiciel à supprimer. Aucun artefact persistant ne restera car ils ont tous étés soigneusement contenus et comptabilisé[31].

L'abstraction des conteneurs et les outils fournis par Docker pour travailler avec les conteneurs vont changer le paysage de l'administration système et du développement logiciel. Docker est important car il rend les conteneurs accessibles à tous. Son utilisation permet d'économiser du temps, de l'argent et de l'énergie[31].

une autre raison pour laquelle Docker est important est que la communauté des logiciels est fortement incitée à adopter les conteneurs et Docker. Cet effort est si fort que des entreprises telles qu'Amazon, Microsoft et Google ont toutes collaboré pour contribuer à son développement et l'adopter dans leurs propres offres de cloud. Ces sociétés, qui sont généralement en désaccord, se sont associées pour soutenir un projet open source au lieu de développer et de publier leurs propres solutions[31].

3.2.4 Composants Docker

3.2.4.1 Docker Engine (Client et serveur Docker)

Docker est une application client-serveur. Le client Docker parle au serveur ou au démon Docker, qui effectue tout le travail à son tour. Docker est livré avec un binaire client en ligne de commande, un menu fixe, ainsi qu'une API RESTful [1] complète. On peut exécuter le démon Docker et le client sur le même hôte ou connecter le client Docker local à un démon distant exécuté sur un autre hôte[7]. L'architecture de Docker est illustrée dans la figure suivante :

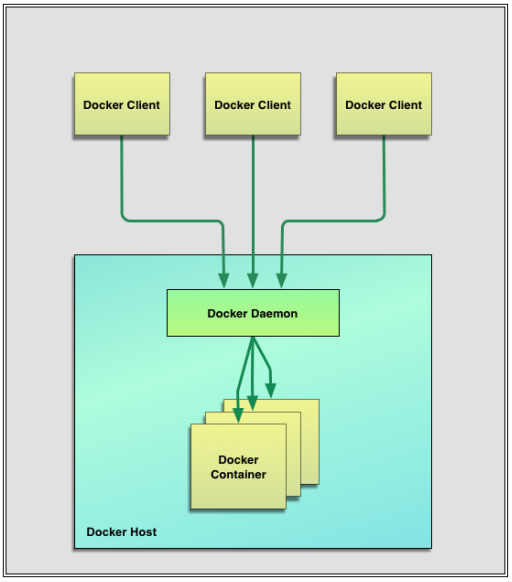


FIGURE 3.13 : Architecture Docker [7]

3.2.4.2 Docker images

Les images sont les éléments constitutifs de l'architecture de Docker. Les conteneurs sont lancés à partir d'images. Les images font partie intégrante du cycle de vie de Docker. Il s'agit d'un format en couches, utilisant les systèmes de fichiers Union [17], qui est construit pas à l'aide d'une série d'instructions[7]. Par exemple :

- Ajouter un fichier.
- Exécuter une commande.
- Ouvrir un port.

On peut considérer les images comme le "code source" des conteneurs. Ils sont hautement portables et peuvent être partagés, stockés et mis à jour[7].

3.2.4.3 Registres

Docker stocke les images construites dans des registres. Il existe deux types de registres : public et privé. Docker Inc exploite le registre public d'images, appelé Docker Hub. On peut créer un compte sur l'hub Docker et l'utiliser pour partager et stocker des images.

Docker Hub contient également, selon le dernier décompte, plus de 10 000 images créées et partagées par d'autres personnes telles que les images de serveur Web Nginx, de système PABX [18] open

17. UnionFS est un service du système de fichiers de Linux qui permet de fusionner plusieurs points de montage appelés "branches" : c'est un union mount.

18. Un PABX ou Private Automatic Branch eXchange est un système électronique indépendant, il s'agit d'un autocommutateur téléphonique privé.

source Asterisk [19] ou une base de données MySql.

On peut stocker des images de façon confidentielle sur l'hub Docker. Ces images peuvent inclure du code source ou d'autres informations exclusives qu'on ne souhaite pas partager qu'avec d'autres membres. On peut également exécuter nos propres registres privés[7].

3.2.4.4 Les conteneurs

Docker aide à créer et à déployer des conteneurs dans lesquels on peut conditionner nos applications et nos services. Les conteneurs sont lancés à partir d'images et peuvent contenir un ou plusieurs processus en cours d'exécution. On peut considérer les images comme l'aspect de la construction ou de l'emballage de Docker et les conteneurs comme l'aspect de l'exécution ou de l'exécution de docker.

Un conteneur Docker est :

- format d'image.
- ensemble d'opérations standard.
- environnement d'exécution.

Docker emprunte le concept du conteneur d'expédition standard, utilisé pour transporter des marchandises dans le monde entier, comme modèle pour ses conteneurs. Mais au lieu d'expédier des marchandises, les conteneurs Docker sont fournis avec un logiciel. Chaque conteneur contient une image logicielle, tout comme son équivalent physique, permet d'effectuer un ensemble d'opérations. Par exemple, il peut être créé, démarré, arrêté, redémarré et détruit. Comme un conteneur d'expédition, Docker ne se soucie pas du contenu du conteneur lors de l'exécution de ses actions. Par exemple, si un conteneur est un serveur Web, une base de données ou un serveur d'applications. Chaque conteneur est chargé de la même manière que tout autre conteneur.

Docker ne se soucie pas non plus du lieu d'expédition du conteneur : il peut être créé sur un ordinateur portable, charger dans un registre, puis télécharger sur un serveur physique ou virtuel, tester, déployer sur un cluster d'une douzaine d'hôtes Amazon EC2 et exécuter. Comme un conteneur d'expédition normale, il est interchangeable, empilable, portable et aussi générique que possible.

Avec Docker, nous pouvons rapidement créer un serveur d'applications, un bus de messages, un utilitaire, un banc d'essai de CI pour une application ou l'un des milliers d'applications, de services et d'outils possibles. Il peut créer des environnements de test autonomes locaux ou répliquer des piles d'applications complexes à des fins de production ou de développement.Les cas d'utilisation possibles sont infinis[7].

19. https ://www.asterisk.org/

3.3 Conclusion

Dans ce chapitre, nous avons présenté la plateforme OpenStack avec laquelle, nous allons travailler, Nous avons aussi, présenté ces composants et son architecture. Enfin, nous avons étudié l'outil docker et ses principales fonctionnalités.

Chapitre 4

Intégration Docker dans OpenStack

4.1 Introduction

Les conteneurs sont un sujet d'actualité puisqu'ils permettent aux utilisateur d'exécuter leurs charges de travail facilement et rapidement en comparaison avec les machines virtuelles. Dans ce chapitre, nous allons commencé par expliquer le besoin de conteneurs dans OpenStack. Ensuite, nous allons abordé les différents processus permettent de prendre en charge les conteneurs Docker.

l'utilisation des conteneurs est de plus en plus populaires ces dernières années et ceci pour les raisons suivantes :

- Les conteneurs fournissent des modèles d'infrastructure immuables utilisant le concept d'emballage.
- Il est facile de développer et d'exécuter des microservices [1] à l'aide de conteneurs.
- Ils permettent un développement et des tests plus rapides des applications.

Le noyau Linux supporte les conteneurs depuis plusieurs années. Microsoft a également récemment commencé à prendre en charge les conteneurs sous forme de conteneurs Windows serveur et des conteneurs Hyper-V. Les conteneurs ont évolué au fil du temps, tout comme le support OpenStack pour les conteneurs. OpenStack fournit des API pour gérer les conteneurs et leurs moteurs d'orchestration au sein des centres de données. Dans ce chapitre, nous verrons comment OpenStack et les conteneurs Docker s'assemblent.

4.2 Le besoin de conteneurs dans OpenStack

OpenStack est utilisé par un grand nombre d'organisations. Les fournisseurs d'infrastructure en nuage ont appelé OpenStack une alternative open source aux services Web Amazon pour les organisations

1. Les microservices sont un style d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés.

visant à maintenir un nuage privé, mais avec l'évolutivité et l'agilité du nuage public. OpenStack est populaire pour les offres d'infrastructure Linux en tant que service (IaaS). Comme les conteneurs gagnent en popularité, il est devenu nécessaire pour OpenStack de fournir diverses ressources d'infrastructure telles que le calcul, le réseau, et le stockage aux conteneurs. Plutôt que de créer de nouveaux silos verticaux pour gérer les conteneurs dans leurs centres de données, les développeurs et les opérateurs peuvent trouver de la valeur dans la fourniture d'une API [1] à plaques croisées pour gérer les machines virtuelles, les conteneurs et les bare-metals [32].

4.3 Les solution possibles d'intégration de Docker dans OpenStack

Il existe plusieurs projets qui travaillent sur l'intégration de docker dans OpenStack, son but et soit l'intégration des conteneurs docker au-dessus des Vms ou bare-metals ou/et l'intégration directe au-dessus de serveur OpenStack. Dans notre projet on s'intéresse au deuxième cas. Parmi ces projets on trouve Nova-Docker et Zun.

4.3.1 Nova-docker

Dans le cycle Havana [2], OpenStack a introduit un pilote Docker, qui est un pilote hyperviseur dans le projet Nova Compute ; appelé hyperviseur Nova-Docker. L'API interne de Docker interagit avec le client HTTP de Nova en utilisant le socket Unix. Docker est géré et contrôlé à l'aide d' API [1] HTTP. Toutes les images Docker sont stockées dans le service d'image d'OpenStack Glance. Les conteneurs exécutaient à l'aide du pilote Docker-virt dans Nova à l'aide de virtAPI [33].

Nova-Docker a commencé comme projet visant à intégrer la fonctionnalité Docker au projet de Nova compute. Il s'agissait d'un projet non propre car le cycle de vie des machines virtuelles et des conteneurs était très différent. Le développement de nombreux autres projets basés sur des conteneurs a conduit à la fermeture du projet Nova-Docker, mais le soutien à Nova reste pour l'approvisionnement en conteneurs. Nova pour conteneurs ont été le début du soutien à part entière de Docker. L'alternative à Nova-Docker est Zun, qui sera traitée en détail dans ce chapitre [34].

2. Havana est une verion d'OpenStack lancée le 2013-10-17

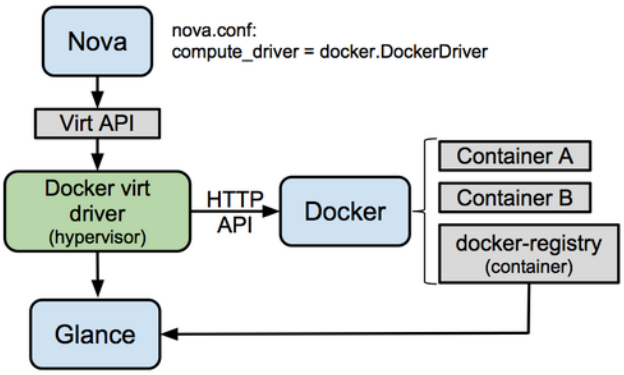


FIGURE 4.1 : Architecture Nova-Docker [8]

La principale différence entre zun et Nova et que Nova-docker permet d'accéder aux conteneurs via l'API [1] de Nova, alors que Zun n'est pas limité par l'API de Nova ; Zun vise à résoudre le problème de la solution de pilote Nova Docker. Il implémente la structure de planification de déploiement de Docker indépendamment de Nova et s'intègre à Glance, Neutron, Cinder et à d'autres composants, mais n'implémente pas la planification de déploiement des moteurs d'orchestration de conteneur (COE). Nova-docker accède aux conteneurs via l'API Nova et Zun n'est pas restreint par l'API Nova [35].

4.3.2 ZUN

Zun est un service OpenStack lancé dans le cycle Mitaka (07/04/2016), développé par les membres de l'équipe Magnum [3]. Une décision a été prise lors de l'OpenStack Austin summit en 2016 pour créer un nouveau projet pour permettre la gestion des conteneurs et laisser le service Magnum Container Infrastructure Management gérer seulement l'infrastructure pour l'exploitation des conteneurs. Le résultat a été le projet zun [9].

Zun est un service de gestion de conteneurs pour OpenStack qui fournit des API [1] pour gérer les conteneurs abstraits par différentes technologies en arrière-plan. Zun supporte Docker comme outil d'exécution de conteneur. Aujourd'hui, zun s'intègre avec de nombreux services OpenStack tels que Neutron pour le réseautage, Glance pour la gestion des images de conteneurs et cinder pour fournir le volume aux conteneurs [32].

Zun a différents ajouts sur Docker, ce qui en fait une solution puissante pour la gestion des conteneurs. Voici une liste de certaines des caractéristiques de zun :

- Fournis une API [1] standard pour la gestion complète du cycle de vie des conteneurs.
- Assure la sécurité multi-locataires et la gestion des autorisations en fonction de Keystone.

3. Ne confondez pas entre service de conteneur et service de gestion d'infrastructure de conteneur dans OpenStack. Ici, le service de conteneur (Zun) permet à OpenStack de lancer et de gérer les conteneurs directement, le projet Magnum permettant de prendre en charge le moteur d'orchestration de conteneur dans OpenStack.

- Prise en charge de Docker avec les conteneurs runc [4] et clear [5] pour la gestion des conteneurs.
- La prise en charge de clear container [5] offre une sécurité accrue en emballant un conteneur individuel dans une VM ayant un faible encombrement.
- Prise en charge de Cinder pour fournir du volume aux conteneurs.
- Mise en réseau basée sur Kuryr [6] pour l'isolation au niveau du conteneur.
- Prise en charge de l'orchestration des conteneurs via Heat.
- La composition des conteneurs, appelée capsules, permet à l'utilisateur de gérer plusieurs conteneurs avec les ressources associées en une seule unité.
- Prends en charge la fonctionnalité SR-IOV [7] qui permet le partage d'un périphérique PCIe [8] physique entre des ordinateurs virtuels et des conteneurs.
- Prends en charge les sessions interactives avec des conteneurs.
- Zun permet aux utilisateurs d'exécuter de lourdes charges de travail avec des ressources dédiées en exposant des ensembles de CPU.

Zun comprend deux composants principaux : l'API [1] zun est utilisé pour communiquer et interagir avec l'utilisateur, et en arrière-plan, zun Compute interagit avec les composants OpenStack via les pilotes et gère les ressources pour les conteneurs, comme communiquer avec Glance pour fournir les images nécessaires ou avec Neutron pour le réseau. Un autre projet joue un rôle important : Kuryr [6] [10] ; forme le pont entre les réseaux d'OpenStack d'un côté et les conteneurs de l'autre. L'architecture d'integration de docker avec zun dans OpenStack est représentée dans la figure suivante.

4. runC, un environnement de conteneur universel léger, est un outil de ligne de commande permettant de générer et d'exécuter des conteneurs conformément à la spécification Open Container Initiative (OCI).

5. https ://clearlinux.org/downloads/containers

6. Kuryr est un plugin réseau Docker qui utilise Neutron pour fournir des services réseau aux conteneurs Docker. Il fournit des images conteneurisées pour les plugins Neutron courants.

7. SR-IOV est une spécification qui permet de partager les ressources d'un adaptateur d'entrées/sorties entre plusieurs machines virtuelles au sein d'un même serveur

8. Le PCI Express, abrégé PCI-E ou PCIe (anciennement 3GIO, 3rd Generation Input/Output) est un standard développé par Intel et introduit en 2004. Il spécifie un bus local série (« bus PCI express ») et un connecteur qui sert à connecter des cartes d'extension sur la carte mère d'un ordinateur.

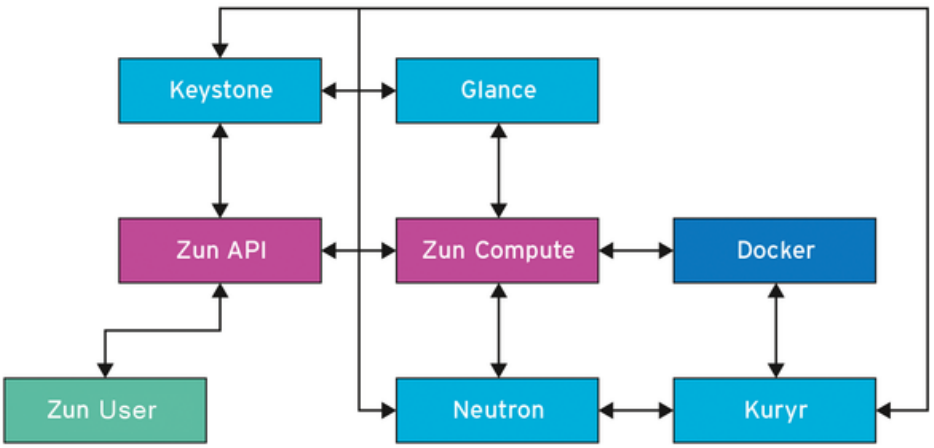


FIGURE 4.2 : Architecture basic de zun et l'integration avec OpenStack [9]

4.3.2.1 Concepts de base

Containers Un conteneur en zun représente tout conteneur d'application exécuté par les utilisateurs. Un objet conteneur stocke des informations telles que l'image, la commande, workdir, l'hôte, etc. Il a une implémentation basée sur le driver pour chaque outil. Le driver Docker dans zun gère les conteneurs via Docker. Les conteneurs zun supportent de nombreuses opérations avancées, y compris les opérations CRUD telles que créer, démarrer, arrêter, pause, supprimer, mettre à jour, tuer, et ainsi de suite[9].

Images Les images en zun sont des images de conteneurs. Ces images sont gérées par Docker Hub ou Glance. Les utilisateurs peuvent télécharger l'image et l'enregistrer dans Glance avant la création du conteneur pour gagner du temps. Un objet image stocke des informations telles que le nom, la balise, la taille, etc. Les opérations prises en charge pour les images sont les suivantes : télécharger, mettre à jour et rechercher des images[9].

Services Un service en zun représente le service zun-compute. Zun peut avoir de multiples instances de zun-compute services en cours d'exécution pour soutenir l'évolutivité. Cet objet est utilisé pour établir l'état des services de calcul fonctionnant dans le cluster zun. Un service stocke des informations telles que l'état, activé ou désactivé, la dernière heure connue, et ainsi de suite[9].

Les hôtes Un hôte en zun représente les ressources du nœud de calcul. Le nœud de calcul est la machine physique où les conteneurs s'exécutent. Ceci est utilisé pour établir la liste des ressources disponibles et utilisées en zun. Un objet hôte dans zun stocke des informations utiles sur un nœud de calcul tel que la mémoire totale, la mémoire libre, le nombre total de conteneurs en cours d'exécution, arrêtés ou en attente, le total des processeurs, CPU libres, et ainsi de suite[9].

Capsules Une capsule en zun représente une unité de composition qui contient plusieurs conteneurs et d'autres ressources connexes. Les conteneurs dans une capsule partagent les ressources entre eux et sont étroitement couplés pour travailler ensemble comme une seule unité. Un objet capsule stocke des informations telles que la liste des conteneurs, CPU, mémoire, et ainsi de suite[9].

Container drivers Zun est conçu pour être une solution extensible pour la gestion des conteneurs en dessus d' OpenStack. Zun supporte Docker pour gérer les conteneurs. Il vise également à soutenir de nombreux autres outils à l'avenir, comme Rocket. Pour soutenir ceci, zun a une collection de pilotes de conteneurs, qui peut être mis en œuvre avec de nombreux autres outils d'exécution et mis à disposition comme solutions avec zun. Les utilisateurs peuvent choisir de gérer leurs conteneurs avec leur choix d'outil[9].

Image drivers Nous avons appris que Zun peut prendre en charge plusieurs outils d'exécution de conteneur pour gérer les conteneurs. De même, il prend en charge plusieurs drivers d'image pour la gestion des images de conteneurs, tels que le driver Glance et le driver Docker. Le driver d'image est également configurable. les utilisateurs peuvent choisir l'une des solutions disponibles pour leur cas d'utilisation[9].

Network drivers La possibilité de communiquer entre deux conteneurs et entre un conteneur et une VM est fournie par le driver de réseau dans Zun. Zun dispose d'un driver Kuryr [6] pour gérer toutes les ressources réseau des conteneurs. Il prend en charge des opérations telles que la création et la suppression d'un réseau, la connexion et la déconnexion d'un conteneur d'un réseau, etc[9].

4.3.2.2 Fonctionalité

Zun a de nombreuses fonctionnalités avancées en plus de la gestion de base des conteneurs. Dans cette section, nous allons parler de certaines des fonctionnalités avancées présentes dans zun. Il existe beaucoup d'autres caractéristiques tels que la mise en réseau SRIOV [7], PCIe [8] dispositifs, et ainsi de suite, qui sont mentionnés dans la documentation zun[9].

integration Zun soutient la fixation du stockage persistant aux conteneurs qui existent même après la sortie des conteneurs. Ce stockage peut être utilisé pour stocker de grandes quantités de données en dehors de l'hôte, ce qui est plus fiable si l'hôte tombe en panne. Ce support est activé dans zun via cinder. Les utilisateurs peuvent monter et démonter des volumes de cendres sur leurs conteneurs. Les utilisateurs doivent d'abord créer le volume en cendre et ensuite fournir le volume tout en créant le conteneur[9].

Container composition Zun soutient la création de plusieurs conteneurs comme une seule unité. Cette unité est connue comme une capsule en zun. Ce concept est très similaire aux pods [9] de Kubernetes [10]. Une capsule contient plusieurs conteneurs et toutes les ressources connexes telles que le réseau et le stockage, étroitement couplé. Tous les conteneurs d'une capsule sont programmés sur le même hôte et partagent des ressources telles que les namespaces Linux, cgroups, etc[9].

Kuryr networking Un conteneur créé par zun peut interagir avec les VM créées par Nova. Cette fonctionnalité est fournie par Kuryr-libnetwork. Il interagit avec Neutron pour créer les ressources réseau nécessaire pour le conteneur et fournit un chemin de communication pour d'autres ressources OpenStack[9].

Container sandbox sandbox est un conteneur auquel sont associées toutes les ressources IaaS, comme les ports, les adresses IP, les volumes, etc. Le but sandbox est de découpler les frais généraux de gestion de ces ressources IaaS des conteneurs d'application. Un sandbox peut gérer des conteneurs simples ou multiples et fournir toutes les ressources nécessaires[9].

CPU sets Zun permet à ses utilisateurs d'exécuter un conteneur haute performance avec des ressources dédiées. Zun expose ses capacités d'hôte aux utilisateurs, et les utilisateurs peuvent spécifier le processeur requis lors de la création d'un conteneur[9].

L'ordonnanceur filtre un nœud avec la ressource disponible et fournit le conteneur sur ce nœud. L'information sur l'hôte est mise à jour dans la base de données pour refléter les ressources mises à jour[9].

4.3.2.3 Les composantes de Zun

zun-api zun-api est un serveur WSGI [11] qui sert les requêtes API des utilisateurs. Pour chaque ressource en zun, il y a des gestionnaires distincts[9] :

- Container
- Host
- Images
- Zun services

9. Pods de Kubernetes est un groupe d'un ou plusieurs conteneurs (comme des conteneurs Docker), ayant du stockage/réseau partagé, et une spécification sur la manière d'exécuter ces conteneurs.

10. Kubernetes (communément appelé « K8s2 ») est un système open source qui vise à fournir une « plate-forme permettant d'automatiser le déploiement, la montée en charge et la mise en œuvre de conteneur d'application sur des clusters de serveurs

11. La Web Server Gateway Interface (WSGI) est une spécification qui définit une interface entre des serveurs et des applications web pour le langage Python.

Chacun des contrôleurs traite une demande de ressources spécifiques. Ils valident la demande de permission, valident les ressources Open Stack, y compris la validation si l'image est présente dans Docker Hub ou Glance, et créent un objet DB pour la ressource avec les données d'entrée. La demande est transmise au gestionnaire de calcul. Après avoir reçu une réponse du service zun-computing, le service zun-api retourne la réponse à l'utilisateur[9].

Zun scheduler L'ordonnanceur en zun n'est pas un service RPC [12] . Il s'agit d'une classe Python simple qui applique un filtre sur les nœuds de calcul et ramasse le nœud approprié pour servir la requête. Le gestionnaire de calcul transmet ensuite la demande au zun-compute sélectionné via un appel RPC [12]. L'appel à zun-compute peut être synchrone ou asynchrone selon le temps de traitement pris par chacune des opérations. Par exemple, les appels de liste peuvent être synchrones car ils ne sont pas chronophages, tandis que les demandes de création peuvent être asynchrones [9].

zun-compute Le service zun-compute est le composant principal du système zun. Il effectue la plupart des opérations backend, cachant toutes les complexités. zun-compute sélectionne un driver approprié pour répondre à chaque demande et crée les ressources connexes pour les conteneurs, tels que les ressources réseau. Il transmet ensuite la demande au driver avec toutes les informations requises. zun-compute parle à de multiples projets pour diverses ressources telles que Glance pour les images de conteneurs et Neutron pour les ressources du réseau [9].

Zun WebSocket proxy Zun dispose d'un service de proxy WebSocket pour l'exécution de conteneurs en mode interactif. Ce service établit une connexion sécurisée avec le conteneur pour exécuter toutes les commandes à l'intérieur [9] :

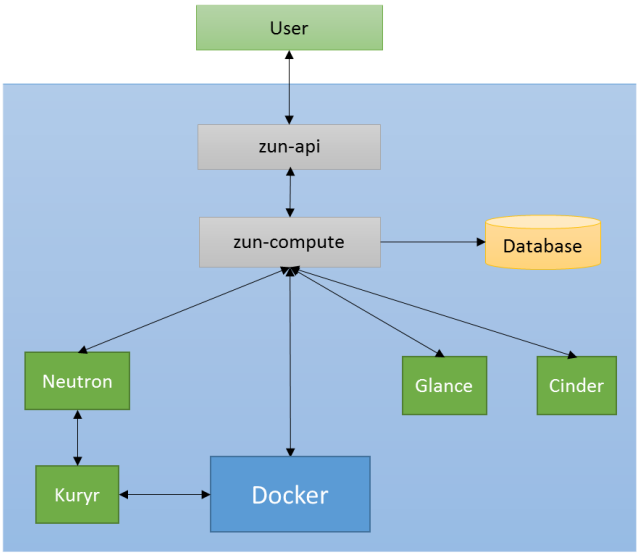


FIGURE 4.3 : Architecture zun et l'integration avec OpenStack [9]

4.4 Expérimentation

4.4.1 Environnement de travail

Pour la réalisation de notre expérimentation nous avons utilisé la plateforme cloudlabs [12] ; qui offre toute une infrastructure virtuelle (serveurs, réseaux, ...) avec des performances très satisfaisantes pour le test.

Nous avons utilisé un profil OpenStack version Rocky qui contient deux nœuds (compute et Controller). Cette plateforme met à notre disposition des serveurs OpenStack pré-installés, que les étudiants, enseignants ou chercheurs peuvent les utilisés afin d'expérimenter le Cloud Computing. Les informations d'identification et d'accès à notre expérimentation sont les suivantes :

- URL : http://controller.zunrocky0.labs1-pg0.utah.cloudlab.us/horizon/auth/login/?next=/horizon/project/instances/
- Domaine : default
- Utilisateur : admin
- Mot de passe : c34c01f73248

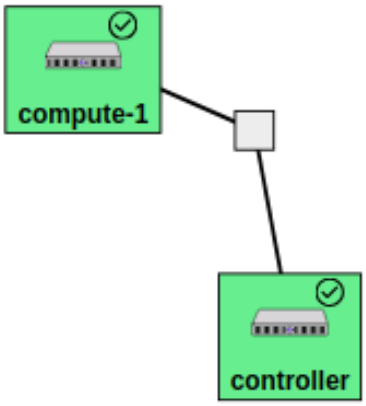


FIGURE 4.4 : Structure de notre environnement de travail

12. http ://docs.cloudlab.us/

4.4.2 Installation et Configuration

Le déploiement de Zun dans un environnement OpenStack nécessite les services OpenStack suivants pour la fonction de base : keystone, Neutron et kuryr-libnetwork. Il peut également s'intégrer à d'autres services pour inclure : Cinder, Heat et Glance.

4.4.2.1 Installation et configuration sur le nœud controlleur

Dans cette partie, nous allons présenter les différentes étapes de configuration et installation de Zun afin d'intégrer Docker à la plateforme OpenStack

prérequis

1. Créer la base de données zun pour stocker des informations de service zun

```
# mysql
MariaDB [(none)] CREATE DATABASE zun;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON zun.* TO 'zun'@'localhost' IDENTIFIED BY 'zun';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON zun.* TO 'zun'@'%' IDENTIFIED BY 'zun';
```

2. Sourcer les informations d'identification de l'administrateur

```
#source admin-openrc.sh
```

3. Créer les informations d'identification du service.

   - Créer l'utilisateur zun

```
# openstack user create --domain default --password-prompt zun
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | ca2e175b851943349be29a328cc5e360 |
| name      | zun                              |
+-----------+----------------------------------+
```

   - Ajouter le rôle admin à l'utilisateur zun :

```
# openstack role add --project service --user zun admin
```

   - Créer les entités de service zun :

```
# openstack service create --name zun --description "Container Service" container
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Container Service                |
| enabled     | True                             |
| id          | 727841c6f5df4773baa4e8a5ae7d72eb |
| name        | zun                              |
| type        | container                        |
+-------------+----------------------------------+
```

4. Créer les API endpoints de service de conteneur pour qu'il communique à d'autre service.

```
# openstack endpoint create --region RegionOne container public http://controller:9517/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 3f4dab34624e4be7b000265f25049609 |
| interface    | public                           |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb |
| service_name | zun                              |
| service_type | container                        |
| url          | http://controller:9517/v1        |
+--------------+----------------------------------+

# openstack endpoint create --region RegionOne container internal http://controller:9517/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 9489f78e958e45cc85570fec7e836d98 |
| interface    | internal                         |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb |
| service_name | zun                              |
| service_type | container                        |
| url          | http://controller:9517/v1        |
+--------------+----------------------------------+

# openstack endpoint create --region RegionOne container admin http://controller:9517/v1
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| enabled      | True                             |
| id           | 76091559514b40c6b7b38dde790efe99 |
| interface    | admin                            |
| region       | RegionOne                        |
| region_id    | RegionOne                        |
| service_id   | 727841c6f5df4773baa4e8a5ae7d72eb |
| service_name | zun                              |
| service_type | container                        |
| url          | http://controller:9517/v1        |
+--------------+----------------------------------+
```

Installer et configurer les composants

1. Créer un utilisateur zun et les répertoires nécessaires :

   - créer l'utilisateur :

```
# groupadd --system zun
# useradd --home-dir "/var/lib/zun" \
      --create-home \
```

```
    --system \
    --shell /bin/false \
    -g zun \
    zun
```

- Créer les répertoires :

```
# mkdir -p /etc/zun
# chown zun:zun /etc/zun
```

2. Cloner et installer zun :

```
# apt-get install python-pip
# cd /var/lib/zun
# git clone -b stable/rocky https://git.openstack.org/openstack/zun.git
# chown -R zun:zun zun
# cd zun
# pip install -r requirements.txt
# python setup.py install
```

3. Générer le fichier de configuration :

```
# su -s /bin/sh -c "oslo-config-generator --config-file etc/zun/zun-config-generator.conf" zun
# su -s /bin/sh -c "cp etc/zun/zun.conf.sample /etc/zun/zun.conf" zun
```

4. Copier api-paste.ini :

```
# su -s /bin/sh -c "cp etc/zun/api-paste.ini /etc/zun" zun
```

5. Éditer /etc/zun/zun.conf

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller

[api]
...
host_ip = 192.168.0.1
port = 9517

[database]
...
connection = mysql+pymysql://zun:zun@controller/zun

[keystone_auth]
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
project_domain_name = default
project_name = service
user_domain_name = default
password = ZUN_PASSWORD
username = zun
auth_url = http://controller:5000
auth_type = password
auth_version = v3
auth_protocol = http
```

```
service_token_roles_required = True
endpoint_type = internalURL

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
project_domain_name = default
project_name = service
user_domain_name = default
password = ZUN_PASSWORD
username = zun
auth_url = http://controller:5000
auth_type = password
auth_version = v3
auth_protocol = http
service_token_roles_required = True
endpoint_type = internalURL

[oslo_concurrency]
...
lock_path = /var/lib/zun/tmp

[oslo_messaging_notifications]
...
driver = messaging

[websocket_proxy]
...
wsproxy_host = 0.0.0.0 #host running horizon
wsproxy_port = 6784
allows origins = 192.168.0.1,10.11.10.1,128.110.153.158 #all  @IP  of the controller node
base_url = ws://128.110.153.158:6784/
```

6. Remplir la base de données zun

```
# su -s /bin/sh -c "zun-db-manage upgrade" zun
```

Finaliser l'installation

1. Créer une configuration upstart pour zun-api service /etc/systemd/system/zun-api.service :

```
[Unit]
Description = OpenStack Container Service API

[Service]
ExecStart = /usr/local/bin/zun-api
User = zun

[Install]
WantedBy = multi-user.target
```

2. Créer une configuration upstart pour zun-wsproxy service. /etc/systemd/system/zun-wsproxy.service :

```
[Unit]
Description = OpenStack Container Service Websocket Proxy

[Service]
ExecStart = /usr/local/bin/zun-wsproxy
User = zun

[Install]
WantedBy = multi-user.target
```

3. Activer et démarrer zun-api et zun-wsproxy :

```
# systemctl enable zun-api
# systemctl enable zun-wsproxy

# systemctl start zun-api
# systemctl start zun-wsproxy
```

4. Vérifier que les services zun-api et zun-wsproxy sont en cours d'exécution :

```
# systemctl status zun-api
# systemctl status zun-wsproxy
```

4.4.2.2 Installation et configurer sur le nœud compute

Prérequis Sur le nœud contrôleur On doit assuré que Etcd [13]est installé correctement dans le nœud contrôleur.

- Sourcer les informations d'identification de l'administrateur :

```
# source admin-openrc.sh
```

- Créer les informations d'identification du service kuryr :

```
# openstack user create --domain default --password-prompt kuryr
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | e0353a670a9e496da891347c589539e9 |
| enabled   | True                             |
| id        | ca2e175b851943349be29a328cc5e360 |
| name      | kuryr                            |
+-----------+----------------------------------+
```

- Ajoutez le rôle admin à l'utilisateur kuryr :

```
# openstack role add --project service --user kuryr admin
```

13. ETCD est Un équipement terminal de circuit de données (ETCD ; en anglais, Data Communication Equipment ou Data circuit-terminating equipment [DCE]) est un élément permettant la connexion des terminaux ETTD (équipement terminal de traitement de données) au canal de transmission de données.

Prérequis Sur le nœud compute

1. Installer de docker

```
# sudo apt-get remove docker docker-engine docker.io containerd runc
# sudo apt-get update

# sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
# sudo apt-key fingerprint 0EBFCD88

# sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
# sudo apt-get update
# sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Créer un utilisateur kuryr et les répertoires nécessaires :

```
# groupadd --system kuryr
# useradd --home-dir "/var/lib/kuryr" \
      --create-home \
      --system \
      --shell /bin/false \
      -g kuryr \
      kuryr

# mkdir -p /etc/kuryr
# chown kuryr:kuryr /etc/kuryr
```

3. Cloner et installer kuryr-libnetwork :

```
# apt-get install python-pip
# cd /var/lib/kuryr
# git clone https://git.openstack.org/openstack/kuryr-libnetwork.git
# chown -R kuryr:kuryr kuryr-libnetwork
# cd kuryr-libnetwork
# pip install -r requirements.txt
# python setup.py install
```

4. Générer le fichier de configuration

```
# su -s /bin/sh -c "./tools/generate_config_file_samples.sh" kuryr
# su -s /bin/sh -c "cp etc/kuryr.conf.sample /etc/kuryr/kuryr.conf" kuryr
```

5. Éditer /etc/kuryr/kuryr.conf :

```
[DEFAULT]
...
bindir = /usr/local/libexec/kuryr

[neutron]
...
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
username = kuryr
user_domain_name = default
password = KURYR_PASSWORD
project_name = service
```

```
project_domain_name = default
auth_type = password
```

6. Créer une configuration upstart /etc/systemd/system/kuryr-libnetwork.service :

```
[Unit]
Description = Kuryr-libnetwork - Docker network plugin for Neutron

[Service]
ExecStart = /usr/local/bin/kuryr-server --config-file /etc/kuryr/kuryr.conf
CapabilityBoundingSet = CAP_NET_ADMIN

[Install]
WantedBy = multi-user.target
```

7. Activer et démarrer le service kuryr-libnetwork :

```
# systemctl enable kuryr-libnetwork
# systemctl start kuryr-libnetwork
```

8. Après le démarrage de Kuryr, redémarrer le service Docker :

```
# systemctl restart docker
```

Installer et configurer les composants

1. Créer un utilisateur zun et les répertoires nécessaires :

   - Créer un utilisateur :

```
# groupadd --system zun
# useradd --home-dir "/var/lib/zun" \
      --create-home \
      --system \
      --shell /bin/false \
      -g zun \
      zun
```

   - Créer les répertoires :

```
# mkdir -p /etc/zun
# chown zun:zun /etc/zun
```

2. Cloner et installer zun

```
# apt-get install python-pip
# cd /var/lib/zun
# git clone -b stable/rocky https://git.openstack.org/openstack/zun.git
# chown -R zun:zun zun
# cd zun
# pip install -r requirements.txt
# python setup.py install
```

3. Générer le fichier de configuration :

```
# su -s /bin/sh -c "oslo-config-generator config-file etc/zun/zun-config-generator.conf" zun
# su -s /bin/sh -c "cp etc/zun/zun.conf.sample /etc/zun/zun.conf" zun
# su -s /bin/sh -c "cp etc/zun/rootwrap.conf /etc/zun/rootwrap.conf" zun
# su -s /bin/sh -c "mkdir -p /etc/zun/rootwrap.d" zun
# su -s /bin/sh -c "cp etc/zun/rootwrap.d/* /etc/zun/rootwrap.d/" zun
```

4. Configurer sudoers pour l'utilisateur zun :

```
# echo "zun ALL=(root) NOPASSWD: /usr/local/bin/zun-rootwrap \
    /etc/zun/rootwrap.conf *" | sudo tee /etc/sudoers.d/zun-rootwrap
```

5. Éditer le fichier de configuration/etc/zun/zun.conf :

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
state_path = /var/lib/zun

[keystone_auth]
memcached_servers = controller:11211
www_authenticate_uri = http://controller:5000
project_domain_name = default
project_name = service
user_domain_name = default
password = ZUN_PASS
username = zun
auth_url = http://controller:5000
auth_type = password
auth_version = v3
auth_protocol = http
service_token_roles_required = True
endpoint_type = internalURL

[keystone_authtoken]
...
memcached_servers = controller:11211
www_authenticate_uri= http://controller:5000
project_domain_name = default
project_name = service
user_domain_name = default
password = ZUN_PASS
username = zun
auth_url = http://controller:5000
auth_type = password

[oslo_concurrency]
lock_path = /var/lib/zun/tmp

[docker]
docker_remote_api_url = tcp://192.168.0.5:2375
docker_remote_api_host = 192.168.0.5

[etcd]
...
etcd_host = 192.168.0.1
```

```
etcd_port = 2379
```

6. Configurer docker et kuryr

   - créer le répertoire /etc/systemd/system/docker.service.d :

```
# mkdir -p /etc/systemd/system/docker.service.d
```

   - Créez le fichier /etc/systemd/system/docker.service.d/docker.conf. configurer docker pour écoute sur le port 2375 ainsi que le socket Unix par defaut, En outre, configurer docker pour qu'il utilise etcd3 comme backend de stockage :

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd --group zun -H tcp://compute-1:2375 -H unix:///var/run/docker.sock --cluster-store etcd://
      controller:2379
```

   - Redémarrer docker :

```
# systemctl daemon-reload
# systemctl restart docker
```

   - Éditer le fichier de configuration Kuryr /etc/kuryr/kuryr.conf :

```
[DEFAULT]
...
capability_scope = global
process_external_connectivity = False
```

   - Redémarrer kuryr-libnetwork :

```
# systemctl restart kuryr-libnetwork
```

Finaliser l'installation

1. Créer une configuration upstart pour le service zun-compute /etc/systemd/system/zun-compute.service :

```
[Unit]
Description = OpenStack Container Service Compute Agent

[Service]
ExecStart = /usr/local/bin/zun-compute
User = zun

[Install]
WantedBy = multi-user.target
```

2. Activer et démarrer zun-compute :

```
# systemctl enable zun-compute
# systemctl start zun-compute
```

3. Créer une configuration upstart, elle pourrait être nommée /etc/systemd/system/zun-compute.service :

```
[Unit]
Description = OpenStack Container Service Compute Agent

[Service]
ExecStart = /usr/local/bin/zun-compute
User = zun

[Install]
WantedBy = multi-user.target
```

4. Vérifier que les services zun-compute sont en cours d'exécution :

```
# systemctl status zun-compute
```

Vérifier le fonctionnement sur le nœud contrôleur

1. Installer python-zunclient :

```
#  pip install python-zunclient==2.1.0
```

2. Sourcer les informations d'identification d'administrateur :

```
# source admin-openrc.sh
```

3. Répertorier les composants de service pour vérifier le lancement et l'enregistrement de chaque processus :

```
# openstack appcontainer service list
+----+-------------------------+-------------+-------+----------+-----------------+----------------------------+-------------------+
| Id | Host                    | Binary      | State | Disabled | Disabled Reason | Updated At                 | Availability Zone |
+----+-------------------------+-------------+-------+----------+-----------------+----------------------------+-------------------+
|  1 | localhost.localdomain | zun-compute | up    | False    | None            | 2018-03-13 14:15:40+00:00 | nova              |
+----+-------------------------+-------------+-------+----------+-----------------+----------------------------+-------------------+
```

4.4.2.3 zun-ui pour horizon

```
# git clone -b stable/rocky https://github.com/openstack/zun-ui
# cd zun-ui
# pip install -r requirements.txt
# pip install .
# cp zun_ui/enabled/* /usr/share/openstack-dashboard/openstack_dashboard/local/enabled/
# python /usr/share/openstack-dashboard/manage.py collectstatic
# python /usr/share/openstack-dashboard/manage.py compress
# systemctl restart apache2
```

4.4.3 Démonstration

Pour tester l'integraion de docker dans OpenStack, nous allons utiliser le CMS WordPress [14]. WordPress repose sur l'utilisation d'une base de données. A cet effet,nous avons besoin de deux conteneurs : un conteneur MySQL pour le stockage de données et un conteneur WordPress pour le traitement "frontend" et "backend".

La base de données MySQL écoute sur le port 3306 ou 33060 et WordPress sur le port 80, donc nous ouvrons ces ports dans le groupe de sécurité qui va être affecté aux conteneurs.

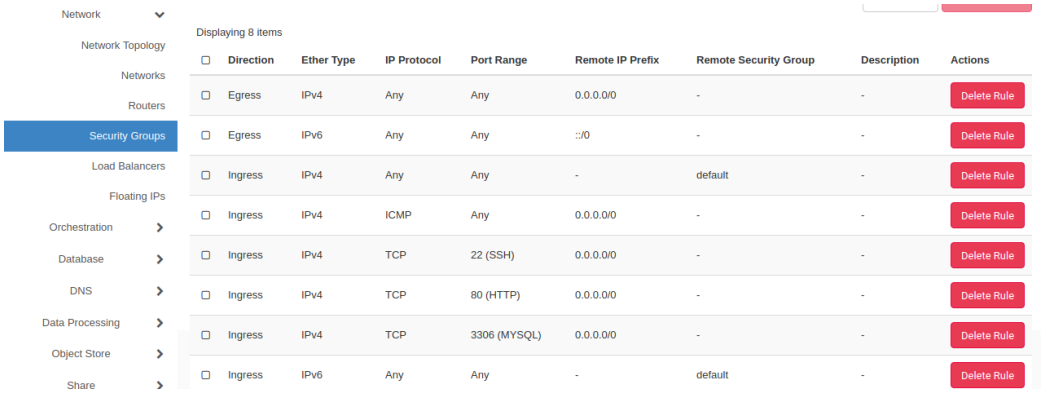


FIGURE 4.5 : Groupe de sécurité

4.4.3.1 Créer le conteneur de base de données Mysql

Nous affectons le réseau de locataire (tenant network) au conteneur database qui est dans notre cas le réseau "flat-lan-1-net", nous affectons le groupe de sécurité defaut précédemment, nous devons aussi ajouter les variables d'environnements.

14. WordPress est un système de gestion de contenu (SGC ou content management system (CMS) en anglais) gratuit, libre et open-source. Ce logiciel écrit en PHP repose sur une base de données MySQL et est distribué par l'entreprise américaine Automatic.

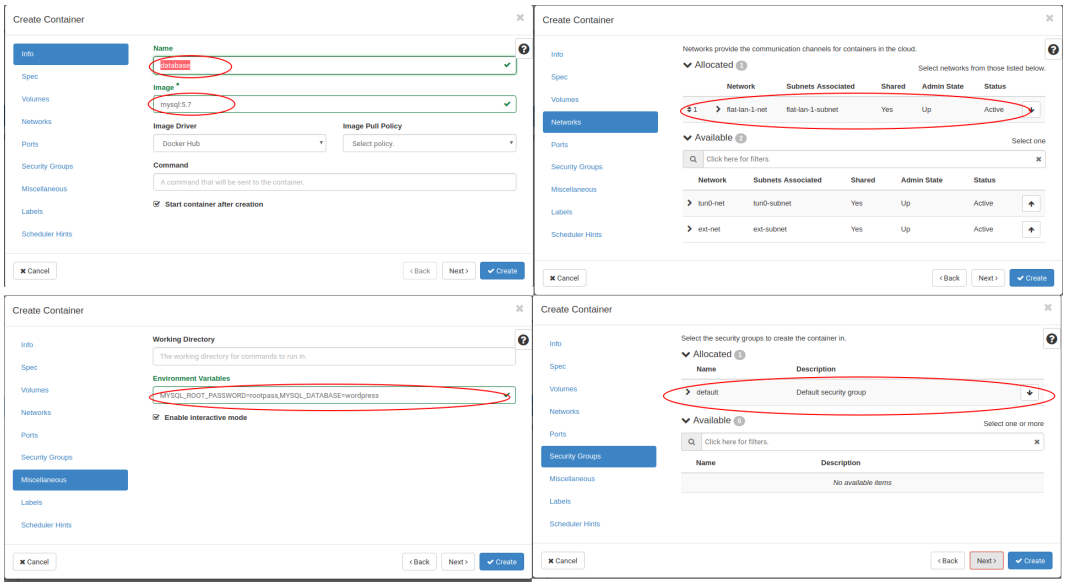


FIGURE 4.6 : Conteneur Mysql

Le conteneur est en état "running" et il est accessible depuis le nœud contrôleur

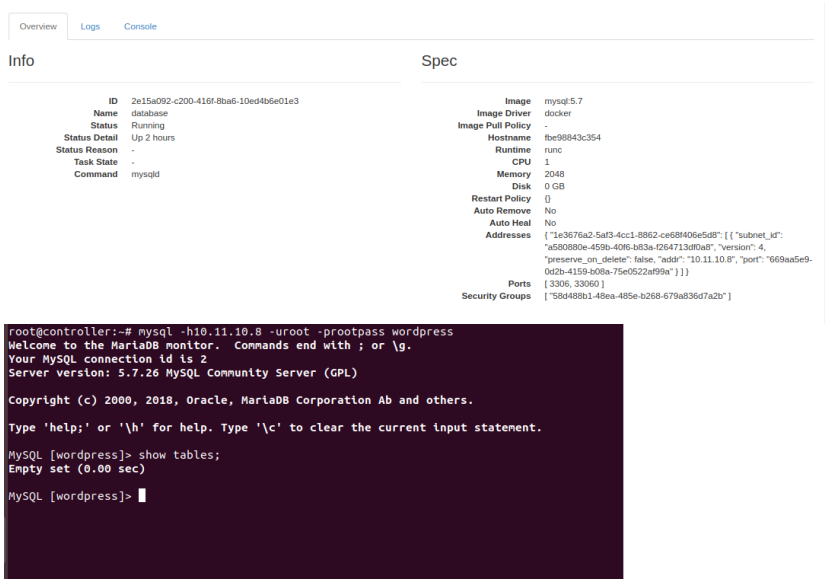


FIGURE 4.7 : Conteneur Mysql "running"

4.4.3.2 Créer le conteneur WordPress

Nous créons le conteneur wordpress de la même façon que le précédent ; nous ajoutons les variables d'environnements.

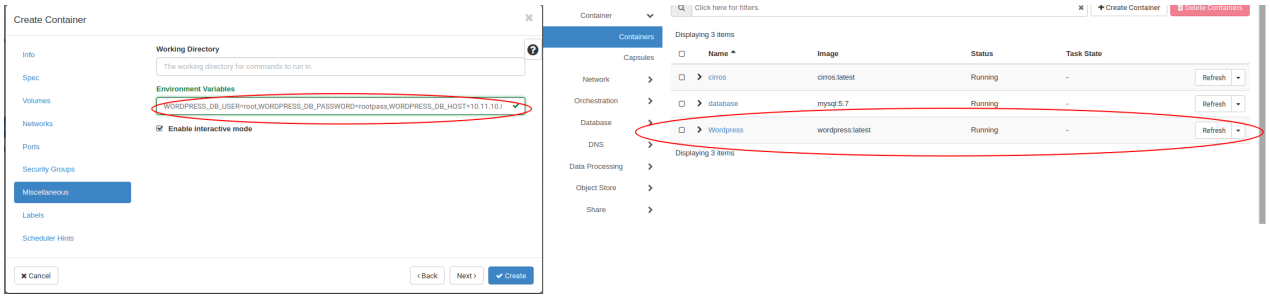


FIGURE 4.8 : Créer le conteneur Wordpress

Nous associons le port du conteneur wordpress à une adresse publique

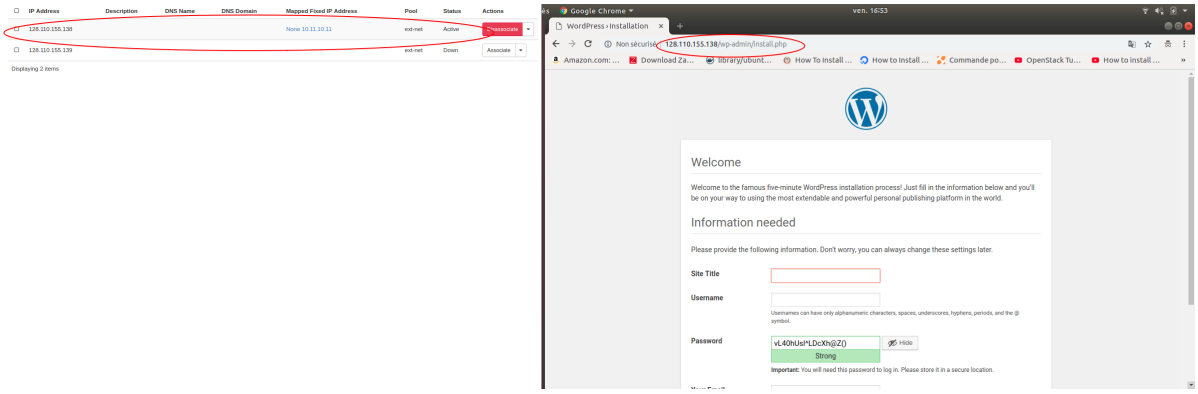


FIGURE 4.9 : Accéder au conteneur via le navigateur

4.4.3.3 Création des conteneurs avec Heat

Zun peut utiliser Heat pour l'automatisation de création des conteneurs composites et pour l'orchestration externe. Pour générer l'exemple précédent avec Heat nous avons créé le fichier ".yaml" du template, ce fichier contient tous les informations (image, réseau, groupe de sécurité, etc.) sur la topologie souhaitée.

```
heat_template_version: rocky

resources:
  secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      name: sg_wordpress
      description: wordpress security group
      rules:
      - protocol: icmp
      - protocol: tcp
        port_range_min: 80
        port_range_max: 80
      - protocol: tcp
        port_range_min: 3306
        port_range_max: 3306
  db:
    type: OS::Zun::Container
    properties:
      image: mysql:5.7
```

```
      environment:
        MYSQL_ROOT_PASSWORD: rootpass
        MYSQL_DATABASE: wordpress
      security_groups:
      - {get_resource: secgroup}
  wordpress:
    type: OS::Zun::Container
    properties:
      image: "wordpress:latest"
      environment:
        WORDPRESS_DB_HOST: {get_attr: [db, addresses, ext-net, 0, addr]}
        WORDPRESS_DB_USER: root
        WORDPRESS_DB_PASSWORD: rootpass
      security_groups:
      - {get_resource: secgroup}

outputs:
  url:
    value: {get_attr: [wordpress, addresses, ext-net, 0, addr]}
    description: The web server url
```

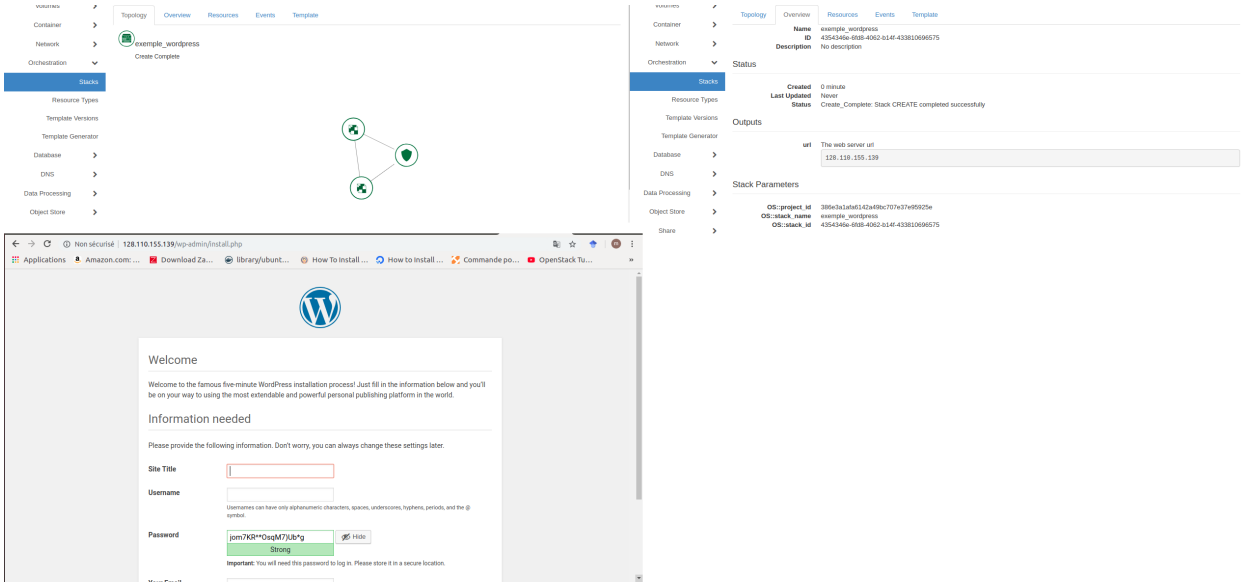


FIGURE 4.10 : Création de conteneurs avec Heat

4.5 Conclusion

Dans ce chapitre, nous sommes concentrés sur le déploiement du docker dans OpenStack. Il existe plusieurs projets dans cette thématique ; parmi ces projets on trouve Nova-Docker et Zun. Grâce à la structure souple de Zun et ses interactions avec différents services d'OpenStack nous avons conclu qu'il est la meilleure solution.

Nous soulignons que le projet Zun est en cycle de développement et il n'est pas encore prêt à cent-pourcents pour être utilisé dans un environnement de production ; nous avons rencontré pas mal des "bugs" et nous avons les corrigés lors de la configuration.

Conclusion

Le cloud computing devient rapidement un modèle dominant permettant aux utilisateurs finaux d'accéder à des ressources informatiques gérées de manière centralisée. OpenStack, en raison de sa flexibilité exceptionnelle, reste inégalé dans sa capacité à englober les techniques émergentes.

Au cours de notre étude, nous avons essayé d'exploiter l'intégration des conteneurs (et plus précisément les conteneurs Docker) dans la plateforme OpenStack afin d'avoir un environnement encore plus souple et configurable. Les projets OpenStack en cours de maturation qui permettant de réalisé ce travail sont Nova-Docker et Zun. Ces projets fournissent des services de pointe pour l'exécution des conteneurs d'applications.

Aprés réflexion de test des différentes solutions ; nous avons pu conclure que l'utilisation de Docker en tant qu'hyperviseur avec Nova-docker au-dessus d'OpenStack est devenu une solution presque obsolète en raison des différences entre Docker et les machines virtuelles.

Contrairement à Zun, qui permet de résoudre les problèmes rencontrés dans la solution Nova-Docker car il implémente la structure de planification de déploiement de Docker indépendamment de Nova et s'intègre à Glance, Neutron, Cinder et à d'autres composants.

Actuellement, le projet virtual-kubelet est en cours du développement, il se focalise sur l' accouplement de Kubernetes et Zun pour l'exécution d'une composition des conteneurs ou ce qu'on appelle une capsule.

Références

[1] Inc. Turnkey Technologies. Hosting microsoft dynamics in the cloud – evaluating iaas, paas and saas. https://www.erpsoftwareblog.com/2016/09/hosting-microsoft-dynamics-cloud-evaluating-iaas-paas-saas/, 2016. Online ; accés le 30/01/2019.

[2] Architecture des hyperviseurs. https://fr.wikipedia.org/wiki/Architecture_des_hyperviseurs. Online ; accés le 2/02/2019.

[3] phelepjeremy. La conteneurisation. https://phelepjeremy.wordpress.com/2017/06/21/la-conteneurisation/, 2017. Online ; accés le 18/02/2019.

[4] Michael Solberg and Ben Silverman. *Containers in OpenStack*, chapter Working with Containers. Packt Publishings, Dec 2017.

[5] phelepjeremy. Logical architecture. https://docs.openstack.org/install-guide/get-started-logical-architecture.html, 2019. Online ; accés le 03/06/2019.

[6] Vinoth Kumar Selvaraj. *OpenStack Bootcamp*, chapter Day 3 - Field Sketch. Packt Publishing, nov 2017.

[7] James Turnbull. *The Docker Book (Version : v1.0.7 (8f1618c))*, chapter Introduction. Manning Publications, Aug 2014.

[8] openstack. Docker. https://wiki.openstack.org/wiki/Docker. Online ; accés le 19/05/2019.

[9] Michael Solberg and Ben Silverman. *Containers in OpenStack*, chapter Zun – Container Management in OpenStack. Packt Publishings, Dec 2017.

[10] Anthony D JoSEP, RAnDy KAtz, AnDy KonWinSKi, LEE Gunho, DAViD PAttERSon, and ARiEL RABKin. A view of cloud computing. *Communications of the ACM*, 53(4), 2010.

[11] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[12] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud computing : A study of infrastructure as a service (iaas). *International Journal of engineering and information Technology*, 2(1) :60–63, 2010.

[13] Lamia Youseff, Maria Butrico, and Dilma Da Silva. Toward a unified ontology of cloud computing. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, 2008.

[14] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3) :583–592, 2012.

[15] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture. *NIST special publication*, 500(2011) :1–28, 2011.

[16] Marisol a Valls, Tommaso Cucinotta, and Chenyang Lu. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, 60(9) :726–740, 2014.

[17] Ashiq Khan, Alf Zugenmaier, Dan Jurca, and Wolfgang Kellerer. Network virtualization : a hypervisor for the internet ? *IEEE Communications Magazine*, 50(1) :136–143, 2012.

[18] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18, 2010.

[19] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization : state of the art and research challenges. *IEEE Communications magazine*, 47(7) :20–26, 2009.

[20] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5) :862–876, 2010.

[21] Pankaj Sareen. Cloud computing : types, architecture, applications, concerns, virtualization and role of it governance in cloud. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(3), 2013.

[22] Sonali Yadav. Comparative study on open source software for cloud computing platform : Eucalyptus, openstack and opennebula. *International Journal Of Engineering And Science*, 3(10) :51–54, 2013.

[23] Vinoth Kumar Selvaraj. *OpenStack Bootcamp*, chapter Day 2 - Know Your Battalion. Packt Publishing, nov 2017.

[24] Antonio Alonso Gil. Cloud computing enhancements and private cloud management. B.S. thesis, Universitat Politècnica de Catalunya, 2018.

[25] Afef Mabrouk. *Outil de gestion de déploiement des conteneurs*. PhD thesis, Université Virtuelle de Tunis, 2017.

[26] Charles Anderson. Docker [software engineering]. *IEEE Software*, 32(3) :102–c3, 2015.

[27] Abel Avram. Docker : Automated and consistent software deployments. https://www.infoq.com/news/2013/03/Docker/, 2013. Online ; accés le 14/02/2019.

[28] KATHERINE NOYES. Docker : A 'shipping container' for linux code. https://www.linux.com/news/docker-shipping-container-linux-code, August 2013. Online ; accés le 14/02/2019.

[29] Alex Williams. The matrix of hell and two open-source projects for the emerging agnostic cloud. https://techcrunch.com/2013/07/28/the-matrix-of-hell-and-two-open-source-projects-for-the-emerging-agnostic-cloud/, 2013. Online ; accés le 14/02/2019.

[30] Adron Hall. Oscon : Conversations, deployments, architecture, docker and the future ? https://www.cloudave.com/30655/oscon-conversations-deployments-architecture-docker-and-the-future/, 2013. Online ; accés le 14/02/2019.

[31] Jeff Nickoloff. *Docker IN ACTION*, chapter Keeping a Tidy Computer. Manning Publications, 2016.

[32] Michael Solberg and Ben Silverman. *Containers in OpenStack*, chapter Containerization in OpenStack. Packt Publishings, Dec 2017.

[33] Ashish Lingayat, Ranjana R Badre, and Anil Kumar Gupta. Integration of linux containers in openstack : An introspection. *Indonesian Journal of Electrical Engineering and Computer Science*, 12(3) :1094–1105, 2018.

[34] Madhuri Kumari and Pradeep Kumar Singh. *OpenStack for Architects - Second Edition*, chapter containers. Packt Publishings, May 2018.

[35] Ji Wei. What is zun : A typical component of openstack to provide container management service. http://telecoms.com/intelligence/what-is-zun-a-typical-component-of-openstack-to-provide-container-management-service/, November 2018. Online ; accés le 30/05/2019.