République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Réseaux, Mobilités et système embarquées**

Présenté par
Guellal Lycia

Thème

Etude et mise en œuvre d'un portail web pour la gestion d'un Cloud privé de type IaaS cas Ooredoo

Mémoire soutenu publiquement le ……/……/ 20….. devant le jury composé de :

Président : Mr Djamah

Encadreur : Mme Djamah

Co-Encadreur : Mr Ali Zerourou

Examinateur : Mr daoui

Examinateur : Mr Ramdan

2015/2016

Remerciements

Nous remercions DIEU le tout puissant, maître des cieux et de la terre, qui nous a éclairé le chemin et permis de mener à bien ce travail.

Tout d'abord nous tenons surtout à adresser nos plus vifs remerciements à ***Mr Ali Zerourou et toutes l'équipes de Cloud et Solution****, qui nous a permis de réaliser ce travail sous sa direction. Nous ne saurons jamais oublier ses conseils judicieux.*

Aux membres du jury ***,*** *pour avoir accepté d'évaluer notre travail.*
Un grand merci à toutes les personnes qui nous ont soutenues de près ou de loin au cours de la réalisation de ce modeste travail.

Dédicaces

Je dédie ce travail

A mes parents Guellal Hamid et Hammache Nacera . et mon frères et mes sœurs un par un:

Vous vous êtes dépensés pour moi sans compter.

En reconnaissance de tous les sacrifices consentis par tous et chacun pour me permettre

d'atteindre cette étape de ma vie. Avec toute ma tendresse.

J' oublie pas mes collèges et mes amis et tous qui m'aime

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Sommaire

Liste des figures

Liste des figures

Introduction Générale

Le développement remarquable du Cloud Computing, ces dernières années, suscite de plus en plus l'intérêt des différents utilisateurs d'Internet et de l'informatique qui cherchent à profiter au mieux des services et des applications disponibles en ligne à travers le web en mode services à la demande et facturation à l'usage. C'est un nouveau modèle économique que ce modèle informatique promet pour les TIC. En effet, le modèle promet un changement dans le mode d'investissement et d'exploitation des ressources IT. Avec le Cloud, les organisations, institutions et entreprises n'ont plus besoin d'investir lourdement dans des ressources informatiques, nécessairement limitées, et nécessitant une gestion interne lourde et coûteuse. Aujourd'hui elles ont le choix de migrer vers un modèle Cloud Computing où elles peuvent acheter ou louer des ressources en ligne. Ce modèle leur épargne les coûts de gestion interne puisque les ressources informatiques sont administrées au niveau du fournisseur du Cloud. La disponibilité des services en ligne donne aussi la possibilité de ne plus s'approprier d'équipements informatiques mais de payer les frais en fonction de l'utilisation des ressources. Ce modèle attire déjà un grand nombre d'entreprises notamment les petites et moyennes entreprises « PME » et les très petites entreprises « TPE ». Ce modèle informatique offre également la modularité des ressources informatiques(hard et soft) et leur disponibilité, en terme de volume et dans le temps, selon les besoins du client et à sa demande. Dans un contexte économique où les entreprises cherchent à rentabiliser au maximum les investissements et à limiter les coûts d'exploitation, le Cloud Computing se présente comme étant la solution de demain.

Structure générale du rapport:

Ce rapport est constitué de cinq chapitres, deux chapitres sont consacrés pour définir les concepts importants qui interviennent dans notre travail et un chapitre sur OpenStack et qui représente le cœur de notre travail, Le quatrième chapitre sera consacrée pour la spécification et l'analyse des besoins permettant d'aboutir aux différents services et fonctionnalités offerts par le portail, dernier chapitre décrira la phase de la réalisation et détermine les différents modules qui interagissent au cours de l'implémentation ,on décortique notre structure générale du rapport comme suivant:

Introduction Générale

Chapitre 1 : Concept de Cloud Computing

Ce chapitre met en avant le concept du Cloud Computing avec les différents services offerts grâce à cette technologie, ainsi Cette partie est dédiée à la virtualisation, la base du Cloud Computing. Nous y présentons en particulier la technologie de l'hyperviseur et son apport au domaine de la virtualisation et le rôle important de la virtualisation dans le Cloud Computing.

Chapitre 2: Les solutions de Cloud privé de type infrastructure as a service

Ce chapitre donne une vision plus clair sur la couche infrastructure as a service de Cloud ainsi ces différentes solutions utiliser au sein des entreprises ,comme les solutions propriétaire et les solutions open source.

Chapitre 3: La solutions infrastructure as a service, opensource, openstack

Dans ce chapitre on a définir notre choix comme solution en présentant l'outil «OpenStack» et ses principaux composants et l'utilité d'utiliser ces APIs.

On schématise notre solution et par la suite on détaille les étapes de son installation dans le chapitre réalisation.

Chapitre 4: Analyse des besoins et conception

sera consacrée pour la spécification et l'analyse des besoins permettant d'aboutir aux différents services et fonctionnalités offerts par l'application ainsi la description de l'architecture de l'application et à sa conception .

Chapitre 5: Réalisation et implémentation

Le dernier chapitre a été consacré pour exposer le travail réalisé pendant le stage. on s'intéressera à la description de l'architecture de l'application et afin de déterminer les différents modules qui interagissent au cours de l'implémentation.

La clôture de ce rapport est une conclusion qui rappellera le contexte de ce présent travail

Chapitre I : Les concepts de Colud Computing

I.1Centre d'accueil et Problématique:

I.1.1Objectif du stage:

L'objectif de notre stage était initialement l'adaptation du Cloud Computing au sein de l'entreprise dans le service *Cloud* et solution qui travaille sur un point d'inflexion dans l'entreprise et qui va permettre d'être un véritable levier de croissance de l'infrastructure de télécommunications (*IT*)et la production des services du Cloud(destinés aux clients et aux utilisateurs internes). Après analyse du problème et discussion avec les responsables du service, il a été convenu de transformer l'infrastructure de télécommunication et de développer un nuage privé. Les objectifs du stage donc été redéfinis comme suit:

1. Implémentation et développement d'un mini Cloud Computing privé propre à L'IT.
2. Gestion de ce Cloud à travers un portail.

I.2Le*Cloud Computing*:

I.2.1Introduction:

Le concept du Cloud Computing est encore en évolution. Ce chapitre présentera les différentes techniques du Cloud Computing, ses modèles ainsi que la relation qui existe entre le Cloud et la virtualisation.

I.2.2Définition et Conception :

Le Cloud Computing, est un concept de l'informatique moderne. Il permet à une entreprise d'exploiter à distance :

- une puissance de calcul.
- un espace de stockage.
- une infrastructure de serveurs reliés par Internet.

Le Cloud Computing permet ainsi d'externaliser tout ou partie des composants d'un système d'information dans des salles informatiques (data center: centres de données) où sont partagés des serveurs. L'entreprise loue l'informatique dont elle a besoin et ne paye que ce qu'elle consomme.

- **Quelques définitions :**

Les définitions sont nombreuses, et chacune introduit une ou plusieurs caractéristiques qui selon l'auteur définissent ce qu'est le Cloud Computing:

Pour **Wikipédia**, Il consiste à déporter sur des serveurs distants des stockages et des traitements informatiques traditionnellement localisés sur des serveurs locaux ou sur le poste de l'utilisateur. Il consiste à proposer des services informatiques sous forme de service à la demande, accessible de n'importe où, n'importe quand et par n'importe qui. L'idée principale à retenir de cette

définition est que le Cloud n'est pas un ensemble de technologies, mais un modèle de fourniture, de gestion et de consommation de services et de ressources informatiques.

Les utilisateurs ou les entreprises ne sont plus gérants de leurs serveurs informatiques mais peuvent ainsi accéder de manière évolutive à de nombreux services en ligne sans avoir à gérer l'infrastructure sous-jacente, souvent complexe. Les applications et les données ne se trouvent plus sur l'ordinateur local, mais « dans le Cloud » composé d'un certain nombre de serveurs distants, interconnectés au moyen d'une excellente bande passante, indispensable à la fluidité du système. L'accès au service se fait par une application standard facilement disponible, la plupart du temps un navigateur web.

Le concept d'informatique dans le *Cloud* est comparable à celui de la distribution de l'énergie électrique. La puissance de calcul et de stockage de l'information est proposée à la consommation par des entreprises spécialisées et facturée d'après l'utilisation réelle. De ce fait, les entreprises n'ont plus besoin de serveurs dédiés, mais confient le travail à effectuer à une entreprise qui leur garantit une puissance de calcul et de stockage à la demande.

Pour le **Syntec**: le Cloud Computing consiste en « une interconnexion et une coopération de ressources informatiques, situées au sein d'une même entité ou dans diverses structures internes, externes ou mixtes, et dont le mode d'accès est basé sur les protocoles et standards Internet». «[il permet de] disposer d'applications, de puissance de calcul, de moyens de stockage[...]comme autant de «services».Ceux-ci seront mutualisés, dématérialisés (donc indépendants de toutes contingences matérielles, logicielles et de communication), contractualisés(en terme de performances, niveau de sécurité, coûts), évolutifs (en volume, fonction, caractéristique)et en libre-service.».

NIST (National Institute of Standards and Technology):

« Le Cloud Computing est un modèle qui permet un accès réseau à la demande et pratique à un pool partagé de ressources informatiques configurables (telles que calcul réseaux, serveurs, stockage, applications et services) qui peuvent être provisionnées rapidement et distribuées avec un minimum de gestion ou d'interaction avec le fournisseur de services. »

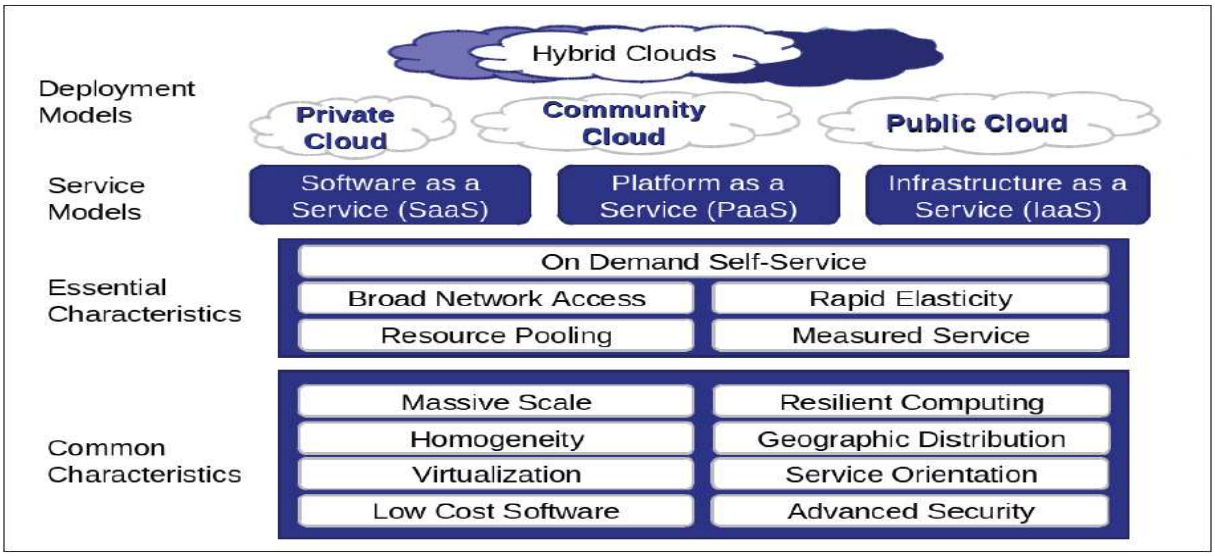


Figure II.2: Le plan de définition du *Cloud Computing* selon NIST [PT09a].

CISCO :

«Le Cloud Computing est une plateforme de mutualisation informatique fournissant aux entreprises des services à la demande avec l'illusion d'une infinité des ressources».

- Même si les experts ne sont pas d'accords sur sa définition exacte, la plupart s'accordent à dire qu'elle inclue la notion de services disponibles à la demande, sans forte interaction avec le fournisseur de service, extensibles à volonté. En contradiction avec les systèmes actuels, les services sont virtuels et illimités et les détails des infrastructures physiques sur lesquels les applications reposent ne sont plus du ressort de l'utilisateur.
- **Autres termes du Cloud:**

  Cloud provider

  Cloud builder

  Cloud application provider

  Market place

  Cloud broker

- Le *Cloud Computing* est une combinaison de deux concepts :
  - **Centre de données (*Data Center*)** : Le data center (centre de traitement des données en français) est l'un des éléments nécessaires au traitement et stockage des données numériques. Indispensable à Internet, il a connu un fort développement avec l'essor du Cloud Computing. Concrètement, il s'agit d'un lieu physique contenant les serveurs informatiques qui stockent les données numériques et dans lequel les entreprises peuvent

notamment louer un espace de stockage et ainsi éviter la présence de serveurs dans leurs locaux.

- **Virtualisation :**La virtualisation est un concept beaucoup plus ancien qui constitue le socle du *Cloud Computing*. Elle regroupe l'ensemble des techniques matérielles ou logicielles permettant de faire fonctionner, sur une seule machine physique, plusieurs configurations informatiques (systèmes d'exploitation, applications, mémoire vive,…) de manière à former plusieurs machines virtuelles qui reproduisent le comportement des machines physiques.

I.2.3Origines et historique du «*Cloud Computing*»:

1.Historique :

La naissance du *Cloud Computing* n'a pas une date précise, elle vient d'une évolution de certaines technologies telles que les web services, ou l'architecture *SOA* (*Service Oriented Architecture*). La notion de *Cloud* fait référence à un nuage utilisé dans les schémas techniques pour représenter Internet, d'où la confusion entre Internet et les services du *Cloud Computing*.

La notion même de consommation de services informatiques a été proposée pour la première fois en 1961 lors d'une conférence au MIT par John McCarthy [1] connu comme l'un des pionniers de l'Intelligence Artificielle. A sa naissance, il pouvait être comparé au cluster de calcul.

John McCarthy, est le principal pionner de l'intelligence artificielle. Il est également l'inventeur du langage Lisp dans lequel un groupe d'ordinateur se relient pour former un ordinateur virtuel unique permettant le calcul de haute performance.

Auparavant, seuls les supers ordinateurs permettaient de fournir une forte puissance de calcul et étaient principalement utilisés par les universités, les gouvernements, ou les universités pour des opérations complexes telles que prévoir les changements climatiques ou le comportement des avions en vol. Désormais, il est possible d'offrir cette puissance de calcul à tout le monde, à tout moment, n' importe où via Internet.

C'est le fait de formaliser une offre de services informatiques dématérialisés à la demande en direction des entreprises qui a été le moteur de développement du *Cloud Computing* en tant que tel.

Il est communément admis que le concept de *Cloud Computing* a été initié par le géantAmazonen2002.

2. Concepts du «*Cloud Computing*»:

On constate qu'il ya plusieurs concepts de *Cloud Computing*, étudiés et développés, associés à certaines technologies, nous allons détailler quatre concepts qui nous semblent être fondamentaux pour comprendre l'histoire du *Cloud Computing*.

2.1. Le concept d'«informatique utilitaire»:

L'*Utility Computing* ou Calcul *On-Demand* désigne le principe de location des ressources informatiques (historiquement la puissance de calcul des machines),que l'on facture à l'usage, à la manière de l'électricité ou du gaz aujourd'hui. Ce concept a l'avantage d'avoir un coût d'investissement initial très faible pour un accès immédiat aux ressources informatiques.

L'*Utility Computing* est un concept déjà ancien qui apparait souvent au cours de l'histoire de l'informatique, et dont l'une des premières références est un discours de John McCarthy lors d'une conférence au centenaire du MIT. Dès1961,celui-ci explique que «l'informatique pourrait un jour être organisée comme un service public tout comme le téléphone est un service public. L'informatique utilitaire pourrait devenir la base d'une industrie nouvelle et importante.». L'idée de l'informatique comme un service est née.

Une des entreprises pionnières dans la mise en place concrète de l'*Utility Computing* après l'éclatement de la bulle internet, fut IBM en 2002 avec son *e-business On-Demand*.Le constructeur jugea qu'il fallait répondre aux exigences croissantes des entreprises, que ce soit en termes de performances ou en termes d'adaptation à un environnement extrêmement changeant. Grâce au développement considérable des réseaux et avec l'avènement des standards informatiques et de la virtualisation, l'*Utility Computing*, avec la notion de *On-Demand*, devient possible et accessible aux *TPE/PME*.L'idée d'une informatique comme un service de base vital dans un monde connecté 24 heures sur 24, commence à se développer.

Cette même année, d'autres sociétés, dont *HP*9,investissent dans le concept d'Utility Computing et reposent leur propre solution. Mais dans son article «*Utility Computing*: peut-on tout externaliser ?» d'Août 2002, paru dans le Journal Du *Net*10, Nicolas Six montre bien l'avance d'*IBM* dans le domaine: «La méthode d'*IBM* est tout autre.[...]la facture [contient]un taux variable pour l'usage des mainframes, l'hébergement du web, la quantité de stockage utilisée, le temps de hotline comptabilisé, et surtout le nombre d'heures de calculs mobilisées sur les processeurs d'IBM. Le client paye réellement ce qu'il consomme. [...]Aucune entreprise n'a opté pour une externalisation complète, et la plupart des marchés remportés concernent seulement l'un des trois pôles majeurs d'une infrastructure: le stockage, la gestion du réseau et la gestion des ordinateurs. *IBM* est le seul à maîtriser les trois».

Peu après, d'autres acteurs, comme *Microsoft*11ou *Amazon*12 ont investi dans le concept et lancé leur propre solution. Ce concept informatique imaginé par John McCarthy est devenu le fondement de la transition vers l'informatique *On-Demand* et l'informatique *as a Service*. C'est un modèle économique qui est à la base des projets d'externalisation des solutions informatiques, des fermes de serveur et bien sur de *Cloud Computing*.

2.2Le concept de «ferme de serveurs »:

Les fermes de serveurs sont apparues au début des années 2000,lors de l'éclatement de la bulle internet pour répondre au besoin de «Haute Disponibilité» nécessaire à la plupart des services accessibles via le web.

Ces «fermes» hébergent jusqu'à plusieurs centaines de serveurs montés en *Cluster*14. Le *Clustering* est une technique qui consiste à regrouper plusieurs serveurs(ou nœuds) indépendant sa fin de dépasser les limitations d'une machine unique. Les différents nœuds mis en réseau ensemble vont apparaître comme une seule machine ayant plus de capacités (plus de puissance, de mémoire, de stockage).On trouve beaucoup d'avantages dans ces fermes de serveurs et dans ce système de *Clustering*. Cette technique permet de garantir la «Haute Disponibilité», d'augmenter la «scalabilité» et donc mieux gérer la montée en charge, et de garantir une bonne qualité de service. De plus ces fermes de serveurs sont très élastiques, grâce la possibilité d'augmenter la taille du Cluster en fonction de ses besoins en puissance de calcul, mémoire ou stockage, simplement en ajoutant des machines.

Ces capacités de *Clustering* sont réalisées grâce à ce système de nœuds multiples, et l'utilisateur a la garantie de ne jamais perdre complètement l'accès au service qu'il est en train de consommer grâce à certains mécanismes:

- La redondance qu'elle soit hardware ou software.
- La reprise sur panne (*Fail Over*).
- La répartition de charges entre nœud (*Load Balancing*).

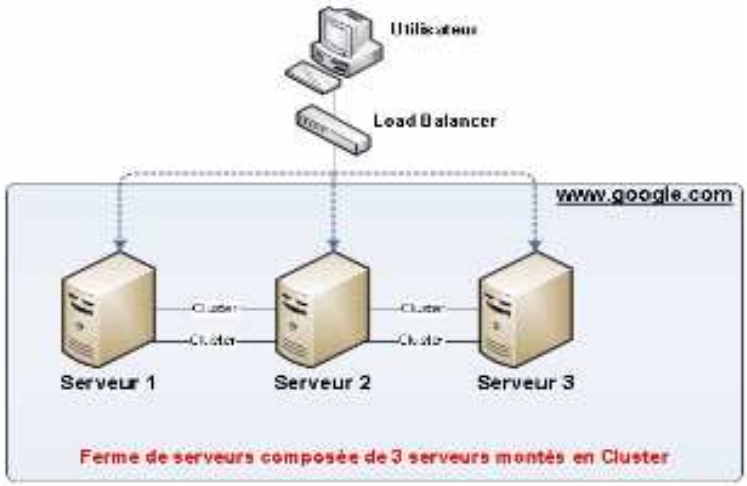


Figure I.3: Représentation schématique d'un cluster de serveurs hébergeant un site web

La répartition de charge est une technique qui permet de distribuer le travail entre les différents serveurs mis en Cluster. C'est cette technique qui va permettre de diminuer le temps de réponse et d'augmenter la disponibilité et la scalabilité du serveur.
Les informations relatives à la disponibilité des serveurs sont envoyées à un serveur maître chargé de distribuer les tâches entre les machines du Cluster. Celui-ci choisira, selon un algorithme prédéfini, vers quel nœud du Cluster il distribuera la tâche.

Nous retiendrons quatre algorithmes principaux:

- Le «*Round Robin*»: qui répartit la charge de manière séquentielle sur les serveurs
- Le«*WeightRoundRobin*»:qui répartit la charge en fonction de la puissance de la machine.
- Le «*Least Connections*»: qui répartit la charge vers le nœud du Cluster ayant le moins de connexions.
- Le «*FasterResponseTime*»:qui calcule le temps de réponse de chaque serveur et distribue le travail vers le serveur qui répond le plus rapidement.

L'éclatement de la bulle internet vers la fin des années1990,début des années2000, a permis pour la première fois de mettre des services à disposition du monde entier, et l'équilibrage de charge a permis à ces services de supporter de manière sûre, fiable et rapide des demandes provenant de plusieurs millions d'utilisateurs. Google utiliserait une ferme de plus de 10 000 serveurs rien que pour son moteur de recherche15.
Les fermes de serveurs permettent encore de nos jours, d'héberger de façon très fiable, des services nécessitant une très haute disponibilité(en général des services accessibles via internet).De plus, il est financièrement avantageux d'investir dans plusieurs petits serveurs que l'on a grège, plutôt que d'acheter un gros serveur multiprocesseur très coûteux. Le principal défaut du *Clustering* est qu'il est nécessaire que les applications soient développées spécifiquement pour pouvoir fonctionner en cluster.

2.3.Concepts *Cloud* liés à l'allocation des ressources:

L'allocation des ressources est la première opération réalisée dans le *Cloud*. Elle attribue au consommateur ou client sa portion de ressources exploitables. Par ressources, nous regroupons à la fois la mémoire, le processeur, l'espace de stockage, la bande passante et les équipements informatiques. Mutualisées entre tous les utilisateurs, les ressources représentent le point de rentabilité pour le fournisseur. Ainsi, la conception et l'implémentation des politiques d'allocation dans le *Cloud* dépendent de la stratégie commerciale du fournisseur.

Notons que l'allocation est provoquée entre autres par une réservation du consommateur. Le fournisseur peut dont proposer plusieurs manières de réservation **[9]**:

1. Réservation pour une durée indéterminée : dans ce mode, un contrat est établi avec le consommateur pour une durée infinie et continue (on peut également parler de forfait).

2. Réservation pour une utilisation à venir et pour une durée limitée : dans ce mode, la difficulté se situe dans la gestion des plages de réservation. On retrouve le problème très connu et complexe qui est celui de la planification.

3. Réservation pour une utilisation immédiate et limitée dans le temps : dans ce cas, les ressources requises doivent être disponibles dans l'immédiat.

2.4.La mutualisation:

C'est la pratique qui consiste à partager l'utilisation d'un ensemble de ressources par des consommateurs (ou entités quelconques) n'ayant aucun lien entre elles. Les ressources peuvent être de diverses natures : logicielles ou matérielles (machines, équipements réseau, énergie électrique). Cette pratique dépend du désir des entreprises de délocaliser leurs services informatiques vers des infrastructures de *Cloud*.

Le *Cloud* doit cependant faire face aux problèmes liés à son exploitation et utilisation. Il s'agit des problèmes classiques tels que la sécurité, la disponibilité, l'intégrité, la fiabilité et l'uniformité d'accès aux données.

I.2.4Le modèle *Cloud Computing* se différencie par les cinq caractéristiques essentielles suivantes:[8]

- **Libre-service à la demande: «**Accès aux services par l'utilisateur à la demande**».**

  Le libre-service à la demande» signifie qu'un client, ou consommateur, pourra commander des ressources informatiques(capacités de calcul, mémoire vive, bande passante, stockage) en fonction de ses besoins. Ce service sera effectué par le fournisseur d'accès de façon automatique sans qu'une intervention humaine soit nécessaire. On retrouve dans cette caractéristique le concept d'*UtilityComputing* et la notion d'informatique *On-Demand*.

- **Accès réseau large bande:**

  «L'accès étendu au réseau»,ou «l'ubiquité»,signifie que les capacités sont disponibles sur le réseau et via les protocoles et standards internet et autres (*TCP/IP, VPN*).

  Ces protocoles standardisés favorisent l'utilisation de plateformes hétérogènes de clients lourds ou légers(comme les téléphones mobiles, les tablettes tactiles, les ordinateurs portables et les postes de travail).

- **Mise en commun des ressources et multi-tenant:**

La «mise en commun des ressources» signifie que les ressources informatiques du fournisseurs ont partagées pour servir les clients multiples en utilisant un modèle multi-locataire.

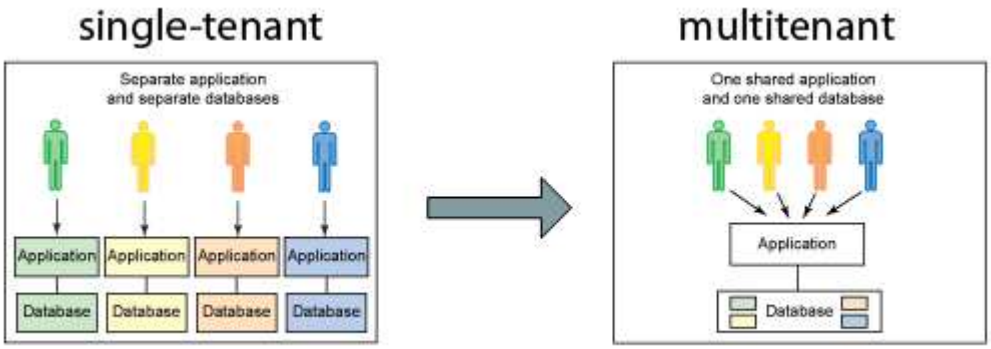


Figure I.4 : Architecture Multi-tenant "multi-locataire"

Les différentes ressources informatiques physiques et virtuelles se conçoivent dans leur ensemble et sont allouées dynamiquement en fonction de la demande des consommateurs.

Le client n'a généralement ni contrôle, ni connaissance sur l'emplacement des ressources, mais est en mesure de le spécifier à un plus haut niveau d'abstraction (pays, état ou centre de données). Ce partage des ressources est la caractéristique qui différencie le *Cloud Computing* de l'infogérance.

La mutualisation intervient au niveau des ressources physiques (réseau, serveur, stockage), mais également au niveau des ressources applicatives(base de données, serveurs web). Ce deuxième cas est le stade ultime de la mise en commun des ressources et se nomme le «multi-tenant».

Le but d'une architecture «*multi-tenant*» (ou «*multi-locataire*») est donc d'avoir une mutualisation maximale des ressources (matériel, système, jusqu'à l'application) tout en gardant la possibilité de personnaliser son architecture à souhait.

Le principe ressemble fortement à la programmation orientée objet. Un objet unique est créé pour un service et sert d'objet de référence. C'est une sorte de modèle (template). Si un client désire payer pour consommer un service, le système génère une instance de l'objet modèle. Le service présenté au client est donc une image de l'objet de référence, qui peut être personnalisée selon son en vie.

Pour résumer, chaque client bénéficie d'une même application, de manière personnalisée. L'avantage principal ici est que toute mise à jour ou modification effectuée sur l'objet modèle est répercutée automatiquement sur toutes les instances de cet objet. Le partage des ressources

(matérielles, applicatives), dont la mise en place a été simplifiée par la virtualisation, permet au fournisseur de maîtriser ses coûts. Le client quant à lui n'a plus à se soucier des problèmes de montée en charge ou de mise à jour d'application.

- **Scalabilité, élasticité rapide et paiement à la consommation:**

  La «scalabilité et l'élasticité rapide» signifient que les capacités informatiques mises à disposition du client peuvent être provisionnées et libérées rapidement (de façon automatique dans certains cas)en fonction des besoins et/ou de la charge. L'utilisateur a l'illusion d'avoir accès à des ressources illimitées. Cela permet d'exploiter de façon optimale les ressources informatiques et d'adapter rapidement l'architecture en fonction du problème à traiter(augmentation ou baisse de la charge, ou demande du client).En effet, avec une architecture faiblement élastique on pourrait facilement se retrouver en sous- utilisation des capacités informatiques disponibles, mais également être confrontés à un manque de capacité lors d'un pic de charge. L'élasticité, aujourd'hui rendue possible grâce à un usage élevé de la virtualisation, permet donc d'avoir une utilisation optimale des ressources informatiques qui suit de façon cohérente l'évolution de la charge réelle.

  Associée à ce concept technique, nous retrouvons la notion économique de «paiement à la consommation».Le gaspillage de capacité informatique aurait pour conséquence directe une perte de rentabilité, et le manque de capacité informatique pourrait faire perdre des clients qui ne seraient pas satisfaits de la qualité du service.

  Maintenant qu'il est possible techniquement d'optimiser l'utilisation des ressources informatiques en fonction de la charge et/ou de la demande, payer seulement ce que l'on consomme semblée vident. Les coûts sont rationnalisés et maîtrisés.

Afin de rendre une architecture plus souple et flexible, le *Cloud* offre différentes possibilités de déploiement :le *Cloud privé* et le *Cloud public*. Le *Cloud privé* nécessitera Le financement variable selon la maturité de virtualisation de la société. En cas de *Cloud public*, le financement se fera en fonction de la consommation et de ses besoins effectifs.

Pour conclure, le modèle économique du *Cloud Computing* permet une plus grande réactivité et une flexibilité qui fait cruellement défaut aux infrastructures classiques. Les concepts de scalabilité et d'élasticité sont les bases techniques du modèle «Payez ce que vous consommez». L'utilisateur a cette sensation de réactivité à chaque sollicitation (mémoire ou *CPU*). Les ressources s'ajustent au plus près du besoin et fluctuent en fonction de la demande et ce de manière totalement transparente.

- **Facturation à l'usage:**

Avoir un «service mesuré» signifie que les ressources consommées sont contrôlées et communiquées à l'utilisateur et au fournisseur de service de façon transparente. Cela garantit un niveau de performance et de disponibilité adapté aux besoins spécifiques des utilisateurs. Ceux-ci peuvent, grâce aux différentes mesures, savoir qu'ils ne sont facturés que pour ce qu'ils consomment. Ils gèrent leur consommation comme ils l'entendent et maîtrisent leurs coûts de façon optimale. La récupération des mesures et des informations de consommation des ressources permet l'application du paiement à la consommation.

Pour conclure ce point, nous pouvons dire que c'est l'addition de ces 5 caractéristiques qui aujourd'hui permet de dire si un service proposé par un fournisseur est vraiment de type *Cloud Computing* ou non. L'application de certaines de ces caractéristiques du *Cloud Computing* a été grandement facilitée par l'avènement de la virtualisation, permettant une élasticité rapide et une mutualisation des ressources maximale. Associé au monde hyper-connecté existant et à une logique de service, le *Cloud Computing* est né..

Maintenant que nous avons cerné ce qui fait qu'un service est réellement un service de type *Cloud Computing*, nous allons dans la section suivante étudier dans le détailles modèles services qui représente l'architecture du *Cloud* existants et leur mode de déploiement.

I.2.5Les Modèle de services du *Cloud Computing*:

L'architecture de *Cloud Computing* peut être décomposée en trois couches:

- Application(*SaaS, Software as a Service*)
- Plateforme (*PaaS, Platform as a Service*)
- Infrastructure (*IaaS, Infrastructure as a Service*)

La Figure ci-dessous représente les différentes couches du *Cloud Computing* de la couche la moins visible pour les utilisateurs finaux à la plus visible. L'infrastructure as a Service (Iaas) est plutôt gérée par les architectes réseaux, la couche PaaS est destinée aux développeurs d'applications et finalement le logicielle comme un service (SaaS) est le «produit final » pour les utilisateurs.

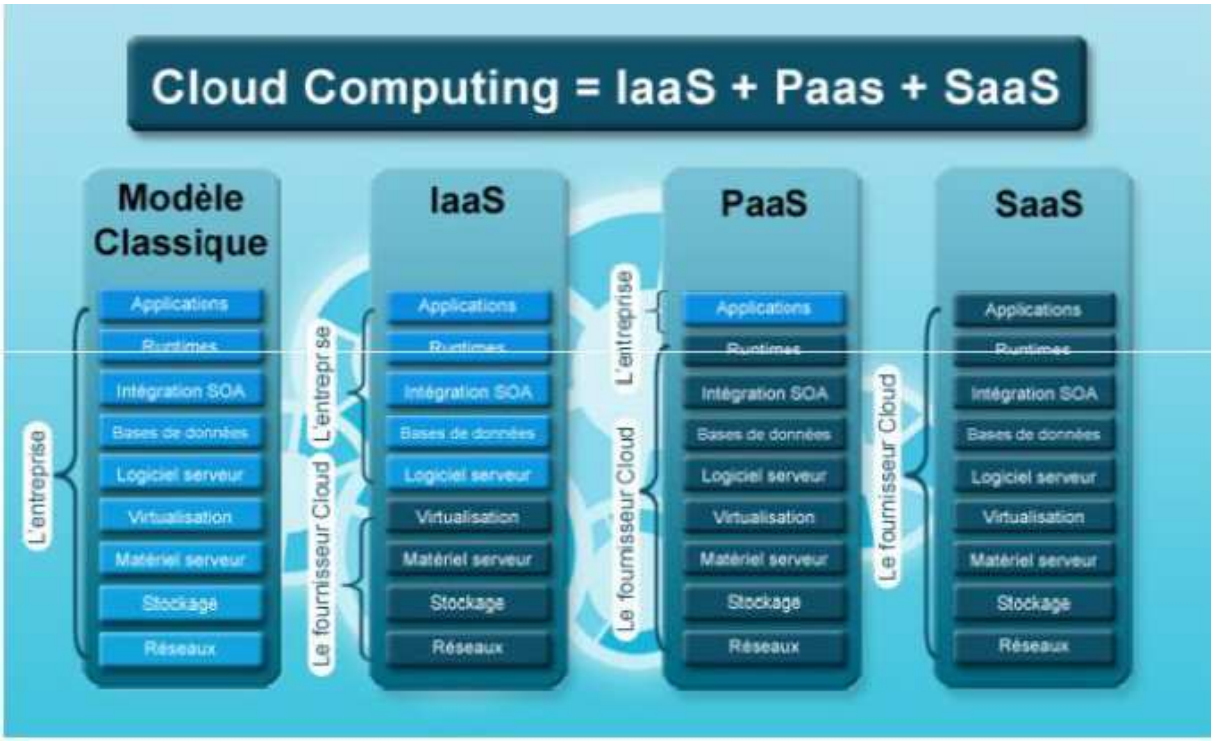


Figure I.5: Les couches du*Cloud Computing*[6]

1. Software as a Service (SaaS):

La dernière couche du *SaaS* est celle applicative mettant à la disposition du client des applications complètes fournies à la demande.

Le logiciel en tant que service, appelé en anglais *Software as a Service*(*SaaS*), est la couche visible du *Cloud Computing*-à l'utilisateur final. Elle repose sur la mise à disposition à travers internet d'applications telles que *CRM* (*customer relationship management: gestion des relations avec les clients*) à la gestion des ressources humaines, compatibilité, messagerie, etc...en contre partie d'un abonnement calculé en fonction du nombre d'utilisateurs. Le fournisseur de service, à sont tour, se charge de déploiement, de stockage, de la maintenance et de la sauvegarde de données. Bien que le mode du fonctionnement des applications et la configuration de certains services sont imposés par le fournisseur, le client de son coté bénéficie quand même d'une fourniture à la demande et quasi-instantanée des services souhaités sans aucun frais en immobilisation (serveurs et licences) et sans le moindre déploiement ou maintenance à assurer. Cette flexibilité du *SaaS* explique bien évidemment le fait d'avoir jusqu'à ce jour, selon Gartener, 90% des entreprises investissant ce mode de *Cloud* ou au moins désirant le faire.

Exemple : Prenons l'exemple de *Salesforces* qui propose des outils de *CRM* et de collaboration, ou de *Google App* qui offre un service de messagerie, d'agenda, de gestion documentaire ou encore d'échange et de partage d'informations. Nous trouvons aussi *Microsoft Online Services* qui met à

disposition leurs produits serveurs tel qu'*Exchange*, *SharePoint*, *Live meeting* ou encore *Office Communicator*, le tout sous forme de services.

2.Platform as a Service (*PaaS*):

La plateforme comme étant un service (*PaaS*), est la plateforme d'exécution, de déploiement et de développement des applications.

Le fournisseur se charge de fournir au client l'environnement d'exécution, l'intégration *SOA*, les bases de données, les serveurs, la virtualisation, le stockage et les réseaux. A son tour, le client n'a qu'à exploiter cette plateforme pour développer ses propres applications. Donc le mode *PaaS* diffère du mode *SaaS* par le fait que le client ne demande pas de son fournisseur une application déjà prête mais plutôt une plateforme hébergée sur le *Cloud* pour développer lui-même sa propre application en exploitant cette plateforme.

Exemple : Parmi les solutions, citons *Windows Azure* de *Microsoft*, *AppEngine* de *Google*, *Force.com* de *Salesforce*.

Chaque fournisseur propose des environnements de développement différents : *Google AppEngine* se limite aux langages *Java* et *Python*, tandis que *Windows Azure* permet de travailler avec les langages *NET, PHP, Pyhton, Ruby* et *Java*.

3. Infrastructure as a Service (*IaaS*):

Infrastructure as a service ou l'infrastructure en tant que service en français est une dernière couche du *Cloud Computing*, la plus complexe en terme de gestion. Elle fournit des capacités de calcul et de stockage ainsi qu'une connectivité réseau. Les serveurs, systèmes de stockage, commutateurs, routeurs et autres équipements, sont mis à disposition pour gérer une charge de travail demandée par les applications.

C'est un modèle où l'entreprise dispose d'une infrastructure informatique (serveurs, stockage, réseau) qui se trouve en fait chez le fournisseur. Cependant, elle ya accès sans restriction, comme si le matériel se trouvait dans ses locaux. Ceci permet à l'entreprise de s'affranchir complètement de l'achat et de la gestion du matériel.

La virtualisation répond de manière dynamique la où les serveurs physiques fournissent un ensemble de ressources allouées selon les besoin, et où la relation entre les applications et les ressources de calcul, de stockage et de réseau pourront s'adapter de manière automatique pour répondre à la charge de travail et aux exigences demandées.

L'entreprise exploite le matériel comme un service à distance. Cette couche permet à l'entreprise de se concentrer en premier sur ses processus métiers sans se préoccuper du matériel.

Exemple :*Microsoft Azure,Amazon AWS* propose, par exemple, le service EC2. Il s'agit d'une offre *IaaS* reposant sur un hyperviseur *Xen*, permettant à un client de demander des ressources informatiques pour y héberger ses applications.

La **table 1** présente les avantages et les inconvénients de chaque service *Cloud* :

	Avantage	**Inconvénient**
SaaS	• pas d'installation • plus de licence • migration	• logiciel limité • sécurité • dépendance des prestataires
PaaS	• pas d'infrastructure nécessaire • pas d'installation • environnement hétérogène	• limitation des langages • pas de personnalisation dans la configuration des machines virtuelles
IaaS	• administration • personnalisation • flexibilité d'utilisation	• sécurité • besoin d'un administrateur système

Table 1. Avantages et Inconvénients des services*Cloud***.**

Les deux modèles (*IaaS* et *PaaS*) sont généralement utilisés dans le cadre d'une relation entre entreprises (*Business To Business* (*B2B*)). Le modèle *SaaS* est au contraire utilisé généralement dans des relations entre une entreprise et le grand public. (*Business To Consumer* (*B2C*)) .

En fait, l'expansion du*Cloud* fait naître le modèle de *XaaS* "signifiant Tout comme un service" (*everything-as-a-service* en anglais). On voit par exemple apparaitre la notion de *Human as a Service* (*HuaaS*) qui caractérise une couche supérieure à *SaaS* correspondant à une ressource humaine élastique. Cette intelligence « artificielle » peut, par exemple, servir pour des décisions arbitraires comme choisir les vidéos les plus intéressantes à afficher (pour un système comme *Youtube*).

- **Services de *Cloud Computing***:

Mise à disposition par le prestataire de ressources à la demande impliquant l'ensemble des activités d'exploitation, de maintenance et de support associé et pouvant couvrir plusieurs niveaux :

- Infrastructure *IT* :solutions *IaaS* (exploitation et maintenance des serveurs)

- Middleware: solutions *PaaS* (exploitation et maintenance des ressources middleware (serveurs d'application, bases de données …)
- Application : solutions *SaaS* (exploitation et maintenance des applications métiers).

I.2.6Les modèles de déploiement du *Cloud Computing*:

Quel que soit le niveau de service du *Cloud Computing*, ces modèles reposent sur une infrastructure physique.

Quelles sont les différentes façons de déployer ces modèles de *Cloud*?

Le NIST distingue, dans sa définition du *Cloud Computing*, 4 modèles de déploiement:

- Le *Cloud privé*, qui peut se déployer sous forme interne ou externe.
- Le *Cloud* communautaire.
- Le *Cloud public*.
- Le *Cloud* hybride.

Que signifient ces 4 modèles?

1. Le *Cloud public*:

Un *Cloud public* est basé sur un modèle standard de *Cloud Computing*, dans lequel un prestataire de service met des ressources, telles que des applications, ou des espaces de stockage, à la disposition du grand public via internet.

Le *Cloud public* peut être gratuit ou fonctionner selon paiement à la consommation.

L'avantage de ce genre d'architecture est d'être facile à mettre en place, pour des coûts relativement raisonnables. La charge du matériel, des applicatifs, de la bande passante étant couverte par le fournisseur. De cette manière ce modèle permet de proposer une souplesse et une évolutivité accrue afin répondre rapidement au besoin. Il n'ya pas de gaspillage de ressources car le client ne paye que ce qu'il consomme.

Malheureusement les entreprises sont encore frileuses par rapport à ce genre de solution, et les acteurs du *Cloud Computing* public vont encore devoir user de quelques arguments afin de convaincre les décideurs *IT*. Les freins à l'adoption de ces technologies restent en effet inchangés. Toutes tailles confondues, les entreprises jugent la sécurité, la confidentialité et l'immaturité comme les faiblesses majeures des solutions d'*IaaS* de type public.

On retrouve dans le *Cloud public* tout type de solution *SaaS* (messagerie en ligne, stockage en ligne, etc) ou *PaaS* (*Microsoft Windows Azure*46).

2. Le *Cloud* privé:

a) Privé interne :

L'architecture est hébergée par l'entreprise. Le*Cloud privé* interne est à l'usage de plusieurs consommateurs appartenant à—une seule entreprise qui est propriétaire de l'infrastructure. Elle peut également être partagée ou mutualisée de façon privée avec ses filiales.

b) Privé externe :

Le *Cloud privé* externe suit la même logique que le *Cloud privé* interne. La différence est que l'architecture est hébergée chez un prestataire. Elle est entièrement dédiée à l'entreprise et accessible via des réseaux sécurisés (*VPN: Virtual Private Network*).

Dans la pratique, ces ressources virtualisés «privés» sont directement administrées par l'entreprise par le biais de son service *IT* interne. Elles peuvent être également mutualisées, et c'est un prestataire de confiance qui prendra en charge la plateforme. Ce modèle est censé apporter les avantages du *Cloud Computing* «public» (ex: baisse des coûts liés à la virtualisation des applications dans le cas d'une infrastructure mutualisée) sans en présenter les inconvénients: en mettant en avant l'aspect sécuritaire des données, le respect de la gouvernance d'entreprise et sur la fiabilité des services fournis.

Les applications infrastructures hébergées restent disponibles en «libre-service»,sont évolutives et modulables grâce à la proximité entre l'entreprise et son prestataire.

Contrairement à un *Cloud public*, et comme nous venons de le voir, l'entreprise reste propriétaire de l'infrastructure(matériel et logiciels), mais elle est aussi responsable de la gestion du taux de disponibilité. Ainsi, les Services Informatiques de l'entreprise passent d'un statut habituellement considéré comme un «Centre de Coût» vers un «Centre de Service» à destination des départements Métiers de l'entreprise. Il appartient donc au Service *IT* de gérer l'infrastructure et les besoins de montée en charge et de disponibilité de manière à respecter les dispositions d'engagement de service vis-à-vis des différents métiers de l'entreprise.

Une analyse approfondie du *Syntec*[1] a permis d'établir une liste d'avantage d'un point de vue opérationnel:

[1]**Syntec** représente près de 1250 groupes et sociétés françaises spécialisées dans les professions de l'Ingénierie, des Services Informatiques, des Études et du Conseil, de la Formation ProfessionnelleSource: 'Tout ce que vous devez savoir sur le Cloud Computing'

- La capacité d'ouverture de ce type d'infrastructure: il devient alors «communautaire», l'entreprise peut ouvrir ses applications aux partenaires de l'entreprise comme les fournisseurs, les bureaux d'études et les prestataires.
- Un meilleur respect des règles de la gouvernance d'entreprise, notamment en ce qui concerne la politique de confidentialité. Il découle de cet avantage que le lieu d'hébergement des données/applications de l'entreprise lui est nécessairement connu (à proximité) : une caractéristique qui permet d'éviter les problèmes juridiques qui se posent lorsque les serveurs du prestataire se trouvent dans un autre pays(quels droits s'appliquent en cas de problème ?).
- La flexibilité, l'informatique est maintenant considérée comme un service, et l'ajustement de la capacité se fait en fonction de la demande (scalabilité et élasticité).
- Une meilleure maîtrise des coûts (même si l'investissement initial est plus important pour l'entreprise dans le cas du *Cloud privé*).

Restent malgré tout certaines inconnues relatives à l'externalisation des données, qui sont soulignées par l'étude de *Syntec*: outre les problèmes souvent évoqués (sécurité ,disponibilité),l'un des points d'interrogation concerne la réversibilité: ce problème se pose quand les fournisseurs de services appliquent des normes standards et outils peu connus qui peuvent rendre les entreprises «captives» de la technologie utilisée. Un problème de transparence auquel les entreprises doivent être particulièrement attentives.

3.Le *Cloud* communautaire :

L'architecture est dédiée à une communauté professionnelle spécifique, pour permettre de travailler de manière collaborative sur un même projet.

Par exemple, dans le projet *Open Cirrus*, le *Cloud* communautaire est partagé par plusieurs universités dans le cadre d'un projet scientifique commun. Son infrastructure peut être gérée par les organisations de la communauté qui l'utilise ou par un tiers et peut être située, soit au sein des organisations, soit chez un prestataire de service.

4.Le *Cloud* hybride :

Comme le *Cloud public*, certains problèmes liés à la sécurité de l'information peuvent se poser. Il est alors possible de «mélanger» les deux approches du *Cloud*, privé et public, en débouchant sur une plate-forme hybride.

Ce terme n'évoque pas vraiment un *Cloud* en tant que tel, puisqu'ils' agit de faire cohabiter et communiquer un *Cloud privé* et un *Cloud public*. Dans le public, on déportera les éléments non sensibles et dans le privé, on gardera les données, applications sensibles liées au business de l'entreprise. Derrière ce terme de «communiquer», beaucoup de problèmes techniques peuvent être

rencontrés, comme l'interopérabilité des plates-formes(un *Cloud privé Microsoft* peut-il communiquer avec un *Cloud public VMware*, par exemple ?).Ce type de *Cloud* est souvent utilisé dans un but de montée en charge ponctuelle comme le permet le *Cloud public*, vu précédemment. La seule différence est que dans ce cas, il serait lié à un *Cloud privé* ou interne a fin de faire communiquer les deux infrastructures.

La figure I.6 illustre et résume parfaitement le concept de *Cloud* hybride.

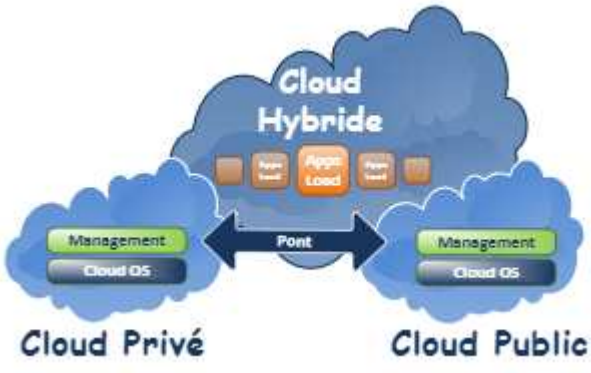


Figure I.6 : Représentation de *Cloud* Hybride

Concrètement, une entreprise pourrait fournir à ses «clients internes» et de façon privée, des plateformes de développement ou des applications spécifiques, en suivant les modèles *PaaS* ou *SaaS*, et dans le même temps utiliser d'autres services de type *PaaS* ou *SaaS* situés dans un*Cloud public*.

I.2.7Les technologies de*Cloud*:

1.La Virtualisation:

a)Introduction:

Depuis quelques années, les entreprises s'intéressent de plus en plus aux technologies de virtualisation. Bien que ce concept ne soit pas nouveau, de nombreuses solutions sont actuellement mises en œuvre autour de la virtualisation.

b) Définition de la virtualisation :

La virtualisation est un ensemble de techniques matérielles et/ou logicielles qui autorisent l'exécution de plusieurs applications indépendantes sur une même machine hôte.

Grâce à la virtualisation, il est possible d'exécuter plusieurs systèmes d'exploitation (*OS* invité) sur un même serveur. Ainsi, il n'est plus nécessaire d'utiliser un serveur par application. Pour bénéficier de cette technologie, il suffit d'équiper une machine d'un logiciel de virtualisation permettant d'ajouter une couche de virtualisation, appelée hyperviseur.

Cet **hyperviseur** masque les véritables ressources physiques de la machine afin de proposer des ressources différentes et spécifiques en fonction des applications qui tournent. Il y a donc une

totale indépendance entre le matériel et les applications. Le logiciel de virtualisation simule autant de machines virtuelles que de systèmes d'exploitation souhaité. Chaque *OS* croit alors qu'il est installé seul sur une machine alors qu'en réalité, plusieurs *OS* peuvent fonctionner en parallèle en partageant les mêmes ressources. Enfin, la virtualisation des serveurs permet un bien plus grande modularité dans la répartition des charges et la reconfiguration des serveurs en cas d'évolution ou de défaillance momentanée.

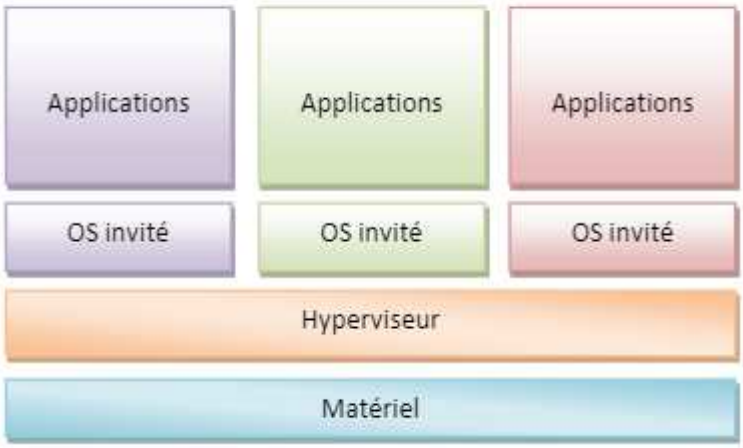
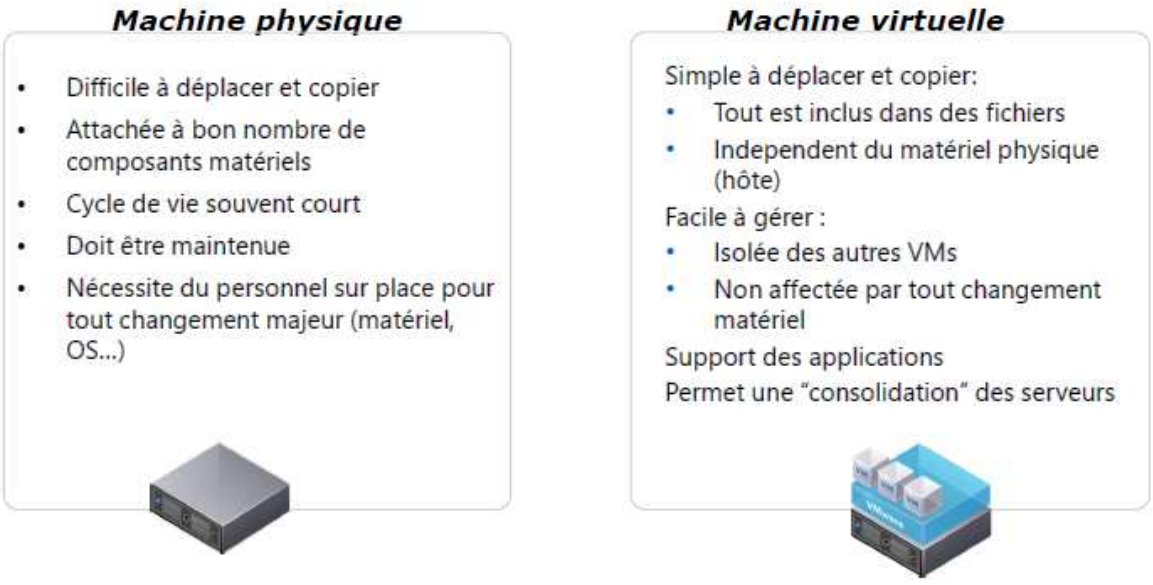


Figure I.7: les couches de virtualisation

c) Pourquoi la Virtualisation?



- Optimiser le développement des applications et les opérations de test:
  - Créer des environnements de développement et de test multiples sur une seule machine.

- Construire des applications sur Windows et/ou Linux.
- Archiver des environnements de test et les restaurer rapidement
- Tester de nouvelles mises à jour d'applications, de nouveaux patchs et service packs de systèmes d'exploitation.

➢ Accélérer le déploiement d'applications:

- Tester, configurer et mettre en place des serveurs d'Entreprise et les déployer en un seul serveur physique.
- Créer un réseau complet d'applications composé de plusieurs ordinateurs et connexions réseau dans un ensemble de machines virtuelles, et les tester sans affecter le réseau.
- Test physique de migrations virtuelles dans le cadre d'applications existantes.

➢ Assurer la compatibilité des applications et réussir la migration de système d'exploitation :

- Supporter les applications existantes tout en leur assurant une migration vers de nouveaux systèmes d'exploitation.
- Tester de nouveaux systèmes d'exploitation sur des machines propres et sécurisées avant tout déploiement.
- Eliminer le besoin de porter des applications existantes.

Le principe de la virtualisation permet une gestion optimisée des ressources matérielles en disposant de plusieurs machines virtuelles sur une machine physique, permet aussi une allocation flexible des ressources, par exemple si une machine virtuelle manque de ressources vu la non suffisance de la capacité du serveur physique sur lequel elle est crée, il suffit de la récréer sur un autre serveur à capacité suffisante.

La virtualisation permet aussi de diminuer le gaspillage des ressources.

On distingue plusieurs techniques de virtualisation à savoir: l'isolation, la para-virtualisation et la virtualisation complète. Ces deux derniers sont les plus utilisées dans le *Cloud*.

d) Techniques de Virtualisations :

1. Les hyperviseurs de type1"paravirtualisation":

C’est un système dont le noyau a été optimisé spécifiquement dans le but de gérer l'architecture à destination des systèmes d'exploitation invités et d’allouer directement les ressources matérielles au système invité. Si un système d'exploitation invité est installé sur un hyperviseur, il peut être optimisé et ainsi augmenter les performances. Il s'agit alors de para- virtualisation. C'est la méthode la plus performante actuellement. Ajout des pilotes pour Communication entre l’hyperviseur et le *guest* au travers de *backend*.

Exemples :

VMware ESXi (vSphere)

Hyper-V

XenServer

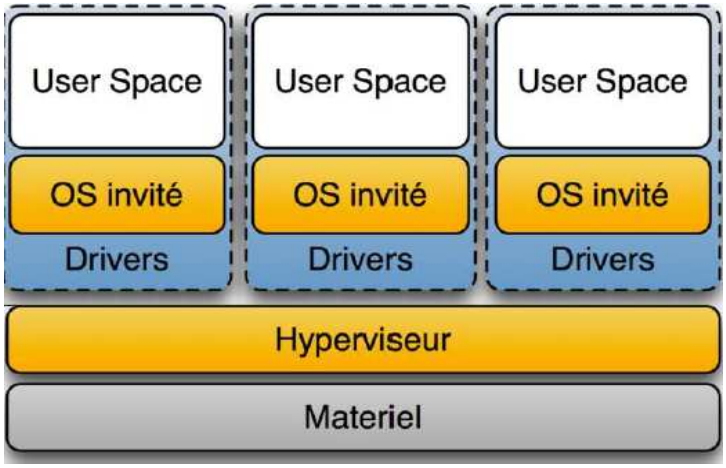


Figure I.8: Architecture d'une virtualisation type 1

2. Les hyperviseurs de type 2"la virtualisation complète":

Un hyperviseur de Type 2 est un logiciel qui s'exécute à l'intérieur d'un autre système d'exploitation qui permet de créer un environnement virtuel complet (en anglais *full-virtualization*).Logiciel qui s'installe sur un système d'exploitation *Linux, MS Windows, MacOS*. C'est un logiciel spécialisé chargé de faire fonctionner des systèmes d'exploitation invités (*guests*) sur un système hôte(*host*) en virtualisant ou émulant le matériel dédié aux systèmes invités. Les systèmes invités ne voient et ne dialoguent qu'avec ce matériel.

Exemples :

VMware Workstation.

XenClient.

VirtualBox.

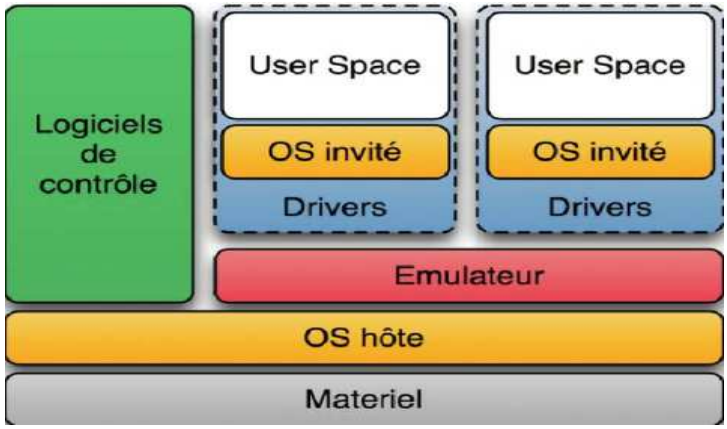


Figure I.9: Architecture d'une virtualisation type 2

3.Virtualisation matérielle:

Accès direct au *CPU* de l'hôte: le support de virtualisation est intégré avec le processeur(modification matérielle sur les *CPU*).

Technologies :

- *Intel-VT-x* (*VT-d, VT-c*).

- *AMD SVM*.

Le noyau de l'hôte est modifié pour ajouter les fonctions de virtualisation matérielles.

4.La notion de conteneurs - *containers* "isolation" :

L'isolation est uniquement liée aux systèmes *Linux*, *Guest OS*(système d'exploitation invité)sans noyau.

- Utilise le noyau de l'hôte, isolation du *guest OS*.
- Pas de virtualisation au sens émulation du matériel.
- Excellente performance, très léger, uniquement disponible sur *Unix-Linux*;

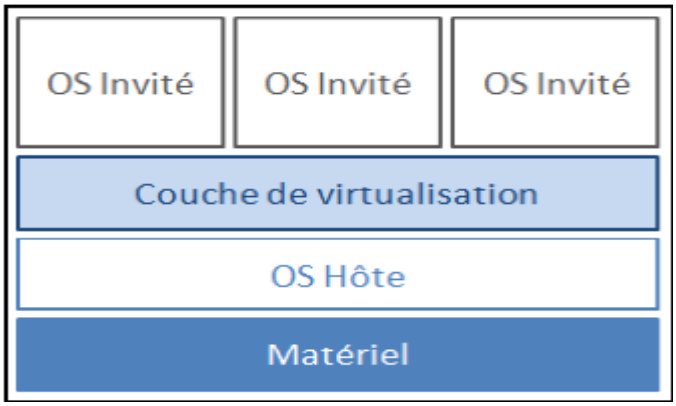


Figure I.10: Architecture de la technique d'isolation

Pour conclure cette comparaison :

- **L'isolation** est considérée comme la solution la plus performante. Cependant son inconvénient réside sur le fait qu'elle est incomplète et le *SE* doit être le même et de type Linux.
- **La paravirtualisation** suppose que le noyau du *SE* invité soit légèrement modifié. De ce fait, si le système ne dispose pas de fonctions dédiées à la paravirtualisation dans son noyau, cette technique devient inutilisable. L'objectif majeur de cette technique est d'offrir un accès quasi identique aux ressources matérielles entre systèmes hôte et système invité.
- **La virtualisation complète** permet d'exécuter le *SE* invité de manière native sans modification. En revanche, cette solution est considérée comme la moins performante puisque le système invité ne participe pas au processus de virtualisation et doit traverser la couche de virtualisation pour accéder aux ressources matérielles.

e)Domaines de la virtualisation :

La virtualisation est une étape majeure vers l'informatique à la demande. Cela permet aux utilisateurs d'avoir accès aux applications et aux ressources avec le minimum de contraintes techniques.

- **Virtualisation du poste de travail:**

  C'est la capacité de pouvoir utiliser le même environnement de travail indépendamment du matériel utilisé(*sur n'importe quel poste du moment qu'il soit connecté à Internet ou au réseau local dans le cas d'un Cloud privé*).

  L'infrastructure du Bureau virtuel est une architecture centralisée de fourniture de bureaux à distance qui permet de centraliser le stockage, l'exécution et la gestion des bureaux dans le centre de calcul. Il permet à Windows et à d'autres environnements de postes de travail de s'exécuter et d'être gérés dans des systèmes virtuels, consolidés sur des serveurs physiques du centre de calcul.

- **Virtualisation d'application:**

  Capacité de pouvoir utiliser à la demande des applications sur un poste de travail sans installation préalable.

- **Virtualisation de serveurs:**

  Capacité de mettre en commun et de rationaliser les ressources physiques des serveurs.

- **Virtualisation de réseaux:**

  De manière générale, la virtualisation des réseaux consiste à partager une même infrastructure physique (débit des liens, ressources CPU des routeurs,...) au profit de plusieurs réseaux virtuels isolés. Un VLAN (Virtual Local Area Network) est un réseau local regroupant un ensemble de machines de façon logique et non physique. Puisqu'un VLAN est une entité logique, sa création et sa configuration sont réalisées de manière logicielle et non matérielle.

- **Virtualisation de stockage:**

  Capacité de mutualiser les espaces de stockage des *Data-centers*.

2.Les services de web:

- **Définition:** D'après la *W3C*[2]: «Un service web est un système logiciel conçu pour supporter les interactions entre applications à travers le réseau».

Les services web offrent un moyen standard d'interopérabilité entre différentes applications qui s'exécutent sur une variété de machines/plateformes. Ils sont caractérisés par leur grande interopérabilité et extensibilité, ainsi qu'une description interprétable/compréhensible automatiquement par la machine. Ils peuvent être combinés d'une façon faiblement couplée afin de réaliser des opérations complexes. Les programmes offrant des services simples peuvent interagir ensemble afin de mettre en place des services sophistiqués avec des valeurs ajoutées.

[2]World Wide Web Consortium, http ://www.w3c.org

De ce fait, les services web enrichissent les programmes et les applications existantes en permettant la communication entre ces dernières [*GM03*]. Les services web permettent d'interfacer des systèmes d'information hétérogènes, qui ont les avantages suivants:

- Un faible couplage avec des technologies propres à chaque programme ou application.
- Une grande flexibilité de mise à jour des systèmes employés.
- L'utilisation de protocoles réseaux standards, simples à utiliser, très répandus et qui sont implémentés dans la plupart des systèmes et technologies utilisés.

L'infrastructure des services web couvre essentiellement deux aspects de base [*FWAS01*] :

- Un service de communication permettant l'échange de données entre les services web.
- Un ensemble de services techniques destinés à automatiser le processus de localisation et d'invocation des composants.

Les services web se basent exclusivement sur les protocoles les plus répandus d'internet comme le *HTTP* (*Hyper Text Transfer Protocol*) avec d'autres protocoles.

L'interopérabilité : C'est l'un des avantages du service web qui est la capacité d'interaction des services web avec d'autres composantes logiciels via *XML* et par l'utilisation d'Internet.

La simplicité : Une fonctionnalité est crée une seule fois, et n'oblige pas les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté. Ainsi, les services web réduisent la complexité des branchements entre les participants.

Une composante logicielle légèrement couplée : Les services web permettent leur utilisation et réutilisation et peuvent être combinés facilement dans d'autres applications.

Ceci est dû à l'architecture modulaire des services web, combinée au faible couplage des interfaces associées.

L'hétérogénéité : Les services web n'imposent aucune contrainte dans la façon dont ils sont implémentés. Ainsi, Ils permettent d'ignorer l'hétérogénéité entre les différentes composantes applicatives.

L'auto-descriptivité: les services web sont capables de s'auto-décrire en fournissant des informations permettant de comprendre la manière de les manipuler. Ceci permet notamment l'automatisation de l'intégration des services.

❖ **Portail web:**

Un portail web est un site web qui offre une porte d'entrée commune à un large éventail de ressources et de services accessibles sur l'Internet et centrés sur un domaine ou une communauté particulière.

❖ **Le *Web 2.0*:**

L'expression fondamentale *Web 2.0* désigne l'ensemble des techniques, des fonctionnalités et des usages qui ont suivi la forme originelle du web, *WWW* ou *World Wide Web*, caractérisée par plus de simplicité et d'interactivité. Elle concerne en particulier les interfaces et les échanges permettant aux internautes ayant peu de connaissances techniques de s'approprier de nouvelles fonctionnalités du web. Les internautes peuvent d'une part contribuer à l'échange d'informations et interagir (partager, échanger, etc.) de façon simple, à la fois au niveau du contenu et de la structure des pages, et d'autre part entre eux, créant notamment le web social L'internaute devient, grâce aux outils mis à sa disposition, une personne active sur la toile.

Le *Web 2.0* est donc l'évolution du Web vers l'interactivité à travers une complexification interne de la technologie mais permettant plus de simplicité d'utilisation, les connaissances techniques et informatiques n'étant pas indispensables pour les utilisateurs.

- **Un intranet:**

L'intranet est un réseau informatique utilisé à l'intérieur d'une entreprise ou de toute autre entité organisationnelle utilisant les techniques de communication d'Internet (IP, serveurs HTTP). Dans les grandes entreprises, l'intranet fait l'objet d'une gouvernance particulière en raison de sa pénétration dans l'ensemble des rouages des organisations.

3. *Datacenter*:

Un centre de traitement de données (*Datacenter* en anglais) est un site physique sur lequel se trouvent regroupés des équipements constituants du système d'information de l'entreprise (serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne et/ou externe à l'entreprise, exploité ou non avec le soutien de prestataires. Il comprend en général un contrôle sur l'environnement (climatisation, système de prévention contre l'incendie, etc.), une alimentation d'urgence et redondante, ainsi qu'une sécurité physique élevée.

Cette infrastructure peut être propre à une entreprise et utilisée par elle seule ou à des fins commerciales. Ainsi, des particuliers ou des entreprises peuvent venir y stocker leurs données suivant des modalités bien définies.

3.1 Caractéristiques:

Selon la quantité de données à stocker, les caractéristiques d'un Datacenter seront différentes :

- Le type de serveur employé et sa capacité maximale
- Le système de climatisation et de ventilation
- La consommation énergétique globale

a/ Serveur employé : Il existe différents types de serveurs, qui varient dans leur performance de transmission et leur capacité de stockage. Les derniers en place sont les serveurs lames, possédant

une capacité de stockage supérieure aux modèles précédents grâce à la mutualisation des composants logiciel dans un châssis, permettant au serveur d'être plus compact.

b/ Température ambiante : La température d'un data center doit être constante et de plus ou moins 20 °C. Elle est assurée par une climatisation classique ainsi que des systèmes de refroidissement par eau dans les villes plus chaudes. Lorsque le local est situé dans une ville où la température moyenne est proche ou en dessous des 20 °C souhaités, un système de récupération d'air froid extérieur pourra être utilisé. Cela permet des économies d'énergie, devenu un nouvel objectif d'évolution (Green Computing). La disposition des serveurs en allées a également été pensée de manière à répartir la chaleur qu'ils dégagent.

c/ Alimentation électrique : L'alimentation électrique peut être assurée par plusieurs circuits différents et complétée par des systèmes de batteries de secours (UPS) ou de générateurs. L'objectif est d'alimenter les systèmes selon le principe du No-Break (zéro coupure). Une défaillance du système électrique aurait pour effet immédiat de rendre impossible l'accès aux données et pourrait même endommager le matériel**.[S4]**

I.2.8 Les avantages et inconvénients de *Cloud Computing*:

En fonction du type de *Cloud Computing* choisi (public, privé ou hybride, communautaire), il offre de nombreux avantages, mais aussi des inconvénients.

Cloud public:

Pour les consommateurs du service le *Cloud public* offre des avantages suivants:

- Aucun investissement préalable:

  Pour une entreprise, le *Cloud Computing* offre des possibilités d'être immédiatement opérationnel.

  Un jeune entreprise qui vient d'être mise sur pied et n'ayant pas d'énormes ressources pour l'achat de serveurs puissants et des logiciels peut être très vite compétitive en utilisant le *Cloud Computing* public.

- Aucun pré requis demandé: L'utilisation d'un service *Cloud* ne nécessite pas des connaissances particulières, mis à part pour le service demandé. Ceci grâce à l'accès simplifié utilisant des navigateurs web.
- Un service d'une grande flexibilité: L'utilisateur n'a aucun effort à fournir, il suffit qu'il demande les services dont il a besoin.
- Un service d'une grande disponibilité: Dans la plupart des cas, le contrat signé entre le fournisseur et le consommateur de service stipule une disponibilité du service à 99%.

- Un payement sur mesure: Le consommateur est facturé à la consommation suivant une grille tarifaire prédéfinie. Ainsi, il paye uniquement ce qu'il consomme.

Par contre, les inconvénients du *Cloud public* sont tout aussi nombreux. On peut citer:

- Le budget:
  - ✓ Etant donné que la consommation des services proposés par le *Cloud public* nécessite une connexion Internet stable, les besoins en bande passante peuvent faire exploser le budget. Par exemple, pour une grande entreprise et/ou ayant un besoin fort en ressources, il sera peut être préférable de trouver une autre solution.
  - ✓ De plus, pour les entreprises situées dans une zone non desservie par une excellente connexion Internet, l'investissement pour avoir une bonne connexion Internet peut ne pas être très raisonnable.
- Le cadre légal:

  Il n'y a aucun accès physique aux données transférées dans le *Cloud public*. Elles ne sont pas forcément présentes sur le territoire national. Il est ainsi difficile de connaitre précisément à quel endroit elles sont stockées. De plus, selon le type d'activité d'une entreprise, la loi peut imposer de pouvoir localiser précisément et rapidement les données, tout en ayant la possibilité d'avoir un accès physique sur celles-ci. Ceci est donc problématique pour un bon nombre d'entreprises.
- La pérennité du service:

  Toutes les entreprises utilisant le *Cloud public* sont dépendantes de leur hébergeur. L'arrêt de son activité, par exemple, pourrait être très problématique. Des études sont en cours pour palier à ces désagréments. En effet, un changement d'hébergeur prend du temps, et peut nécessiter un recodage des applications.
- Confidentialité et sécurité des données:

  Les données sont hébergées en dehors de l'entreprise. Ceci peut donc poser un risque potentiel pour l'entreprise de voir ses données mal utilisées ou volées. C'est actuellement le problème majeur du *Cloud Computing* public.

A cause des problèmes et inconvénients cités plus haut, de nombreuses entreprises se tournent vers le *Cloud Computing* privé. Si le *Cloud Computing* public propose des ressources informatiques hébergées (serveurs, stockage, puissance de calcul, applications...) distantes et mutualisées, les offres de *Cloud Computing* privées se distinguent par leur aspect dédié. Leur usage est réservé pour une seule entreprise, ceci dans le but de répondre à un besoin personnalisé de ressources informatiques.

***Cloud privée*:**

Les avantages du *Cloud privé* sont les suivantes :

- Sécurité et confidentialité: Contrairement au *Cloud public*, l'entreprise est propriétaire de ses données et on peut y avoir accès physiquement.
- Une architecture sur mesure : Le *Cloud privé* est développé en fonction de l'entreprise et elle seule. Ainsi, elle est facilement gérable et adaptable.

Malgré cela, le *Cloud privé* a quelques inconvénients:

- Budget : Toute la charge financière et technique (serveur, personnels qualifiés, logiciel), pour la mise en place du service sont supportés par l'entreprise.

I.2.9 Choix du type de *Cloud* utilisé:

La solution optimale adoptée est le *Cloud privé* (couche *IaaS*) comme une première étape pour l'adaptation du *Cloud Computing* dans l'*IT* vu que les données sont très critiques et le *Cloud privé* assure le contrôle total de données et des applications.

Il dissipe ainsi les préoccupations en matière de sécurité et du contrôle de données. De plus, le *Cloud privé* consolide les ressources informatiques distribuées et les virtualisent dans le *Datacenter*.

La Société peut ainsi gérer ses ressources de manière plus rentable tout en fournissant des services plus rapidement.

Conclusion:

Ce chapitre, a clarifié en premier lieu le Concept de *Cloud Computing*, ses modèles, ses modes de fonctionnement et la technologie de virtualisation.

En conclure que le *Cloud Computing* est un moyen efficace pour faire des économies, d'une part, pour le fournisseur, en proposant le même service à des tarifs compétitifs tout en assurant la maintenance et l'évolution. Et d'autre part, pour le client qui est de plus en plus efficace et performant en exécutant des tâches plus rapidement, et ce en externalisant ses données.

Néanmoins, la sécurité et la confidentialité restent un obstacle non négligeable pour les utilisateurs. Ce qui les rend méfiants envers cette nouvelle technologie. Le-*Cloud Computing* reste un terrain à déchiffrer, plein d'opportunités à découvrir, son usage doit être affiné et ses utilisateurs doivent perfectionner le concept.

Chapitre II : les solutions de Cloud privé de type infrastructure as a service

I. Introduction:

Depuis ces dernières années, plusieurs projets autour du *Cloud Computing* ont vu le jour et donné naissance à autant de plateforme d'administration dans le *Cloud*, Nous allons développer notre raisonnement sur une solution de *Cloud* privé interne de type infrastructure as a service (IaaS). Dans cette section, un extrait des solutions Infrastructure as a service propriétaires et des solutions open sources est étudié .

II. Qu'est-ce que l'*IaaS* ?

L'Infrastructure as a Service (*IaaS*) est l'une des trois principales catégories de services de *Cloud Computing*, avec *Platform as a Service* (*PaaS*) et *Software as a Service*(*SaaS*). Comme tous les services de *Cloud Computing*, ce service donne accès aux ressources informatiques dans un environnement virtualisé, le "*Cloud*", à travers une connexion publique, généralement Internet.

Dans le cas de l'*IaaS*, la ressource fournie est du matériel informatique virtualisé ou, en d'autres mots, une infrastructure informatique. La définition du service comprend des offres telles que l'espace serveur, des connections réseau, de la bande passante, des adresses *IP* et des *load balancers*. Physiquement, les ressources hardware proviennent d'une multitude de serveurs et de réseaux généralement distribués à travers de nombreux centre de donnée (Data center), que le fournisseur de services *Cloud* a la responsabilité d'entretenir. Parallèlement, l'accès aux composants virtualisés est donné à l'entreprise cliente afin que celle-ci puisse construire ses propres plateformes infrastructures de télécommunications(*IT*).

A l'instar des deux autres formes d'hébergement *Cloud*, l'*IaaS* peut être utilisée par les entreprises clientes pour créer des solutions informatiques à des coûts avantageux et facilement extensibles, dans la mesure où la complexité et les coûts inhérents à la gestion du matériel informatique sous-jacent sont externalisés au fournisseur *Cloud*. Ainsi, les entreprises ont tout loisir d'utiliser les ressources *Cloud IaaS* lorsqu'elles en ont besoin, au gré de l'évolution de leurs opérations et de leur développement, plutôt que d'acheter, d'installer et d'intégrer elles-mêmes le matériel.

Exemples d'utilisations de services *IaaS* par une entreprise :

- **Infrastructure de l'entreprise:**

  Les réseaux de *Cloud* privés et les réseaux locaux virtuels, qui utilisent des pools de serveurs et des ressources réseau sont utilisés par des entreprises pour stocker leurs données et faire fonctionner les applications dont elle ont besoin pour leurs activités quotidiennes. Les entreprises en expansion peuvent augmenter leur infrastructure en fonction de leur croissance,

et leurs *Clouds* privés (auxquels seule l'entreprise a accès) peuvent protéger le stockage et le transfert de leurs données sensibles.

- **Hébergement *Cloud*:**
  C'est l'hébergement de sites web sur des serveurs virtuels fondés sur un pool de ressources tirées de serveurs physiques sous-jacents. Par exemple, un site web hébergé dans le *Cloud* peut bénéficier de la redondance fournie par un vaste réseau de serveurs physiques et d'une extensibilité sur demande pour affronter les demandes inattendues dont il fait l'objet.
- ***Virtual Data Centers* (*VDC*):**
  C'est un réseau virtualisé de serveurs virtuels interconnectés pouvant être utilisé pour offrir davantage de capacités d'hébergement *Cloud* ou d'infrastructure informatique ou pour intégrer l'ensemble de ces opérations dans une implémentation de *Cloud* privé ou de *Cloud* public.

Une offre typique d'*Infrastructure as a Service* (*IaaS*) peut fournir les fonctionnalités et avantages suivants :

- **Extensibilité:**
  - ✓ les ressources étant disponibles dès que le client en a besoin, il n'y a donc pas de délais pour étendre ses capacités ni gaspillage de capacités non utilisées.
  - ✓ Pas d'investissement en matériel : le matériel physique sous-jacent supportant un service *IaaS* est installé et maintenu par le fournisseur *Cloud*, économisant ainsi du temps et de l'argent pour le client qui n'a pas besoin d'y veiller lui-même.

- **Modèle de prix basé sur l'utilisation** :
  Le service est accessible à la demande et le client ne paie que pour les ressources qu'il a effectivement utilisé.
- **Indépendance de l'emplacement** :
  Généralement, il est possible d'accéder au service depuis n'importe quel emplacement tant qu'il dispose d'une connexion Internet et que la sécurité du *Cloud* le permet.
- **Sécurité physique des sites de *data centers*** :
  Les services accessibles par le *Cloud* public, ou *Cloud* privés hébergés à l'externe auprès du fournisseur *Cloud*, bénéficient de la sécurité physique des serveurs hébergés dans un *data center*.
- **Aucun point de défaillance unique** :

En cas de panne d'un serveur ou d'un commutateur réseau, le service n'est pas affecté grâce à la multitude de ressources de matériel et de configurations de redondance. Pour de nombreux services, si un data center entier devait se retrouver hors ligne, le service IaaS continuerait à fonctionner sans problèmes.

III. Les différentes solutions *IaaS* disponibles:

Depuis quelques années, les éditeurs de logiciel se sont lancés dans la technologie *Cloud* en proposant à de nombreuses sociétés et PME des solutions de *Cloud Computing* en *IaaS* pour installer leur propre *Cloud* privé ou d'autre type, en se basant sur un *Cloud* privé.

III.1Solutions propriétaires:

III.1.1 *Microsoft* et son offre au Cloud:

Exchange, *SharePoint*, *SQL Server*, *Dynamics CRM*, *Windows Server*, sont des solutions connues qui ont fait leurs preuves sur lesquelles se basent les solutions *Cloud Microsoft*.

Les solutions les plus en avance, une informatique toujours adaptée aux besoins et à jour, pour tout ou partie de l'entreprise, sont accessibles très facilement et le paiement se fait uniquement sur ce qui a été réellement utilisé.

En Novembre 2008, *Microsoft* a crée *Windows Azure*, Solution de *Cloud Computing*. Mise sur le marché en janvier 2010, elle est toujours en pleine évolution aujourd'hui.

Les services du *Cloud* de *Microsoft* sont:

- Une infrastructure *IaaS* avec la notion de «VMRole».
- L'hébergement d'applications *.Net*, le stockage, les *BDD* sont rendus possibles grâce à la plateforme *PaaS* proposée par *Windows Azure*.
- Les applications *SaaS* de la gamme *Live* et *Online Service*.

Ils sont proposés sous deux modèles de déploiement différents:

- Le *Cloud* public.
- Le *Cloud* privé.

Quel que soit le niveau de service désiré, le client peut choisir l'un ou l'autre des modèles en fonction de ses besoins. Il a même la possibilité de combiner ces deux modèles et de créer un Cloud hybride si des besoins spécifiques le nécessitent. La solution *Cloud* de *Microsoft* allie souplesse et facilité de gestion.

La gamme de composants basés sur les services proposée par le *Cloud* de *Microsoft* est étendue.

La société fournit donc une solution complète de :

- services d'infrastructures.
- services de plateforme.

- de logiciels.

Lessolutionsde*CloudComputingMicrosoft*,qu'ellessoientprivéesoupubliques comprennent :

- la virtualisation et l'automatisation.
- des ressources diverses: serveurs, capacités de stockage et réseaux.
- la gestion centralisée de ces ressources.
- une élasticité au niveau de la puissance, en fonction des exigences de l'entreprise.
- des applications et outils de développement évolutifs.

➢ **Les Services (*SaaS*):**

L'arrivée de la génération d'applications *SaaS* remplacera *BPOS*(*Microsoft Business Productivity Online Standard*).

Office365[3]proposera, pour chaque licence utilisateur lesversions2010de :

- *SharePoint Online*: plateforme de collaboration.
- *Exchange Online*: solution de messagerie à la demande.
- *Lync Online*: plateforme de communication.
- *Office Web Apps* (bureautique en mode Web).
- *Office Professionnel Plus2010*.

Office365@Edu remplacera bientôt la gamme utilisée par le grand public et le monde éducatif. Jusqu'ici *Microsoft* proposait un service *Live* composé de:

- Une messagerie: *Windows Live Hotmail*.
- Un service de partage et d'hébergement: *Windows Live SkyDrive*.
- Unservicespécifiqueàl'éducation,*Live@Edu*,avecdessolutionsadaptées à l'école.

Microsoft propose également d'autres logiciels de type *SaaS*:

- *Microsoft Dynamics CRM Online*[4] propose des solutions via *Microsoft Office Outlook* (ou un navigateur Internet) dans le domaine du service à la clientèle, des ventes et du marketing.
- *Exchange Hosted Services*(*EHS*)[5] fournit une protection contre les courriers indésirables et les programmes malveillants et une aide organisationnelle.

➢ **La Plateforme *Microsoft* (*PaaS*):**

La plateforme *PaaS Windows Azure* propose trois briques:

[3]**Source**:Office 365
[4]**Source:**Microsoft Dynamics CRM Online
[5]**Source:**Exchange Online

Windows Azure, un système d'exploitation. *Windows Azure*™ est hébergé sur un *Data Center Microsoft*. Les entreprises qui utilisent ce service ont a à leur disposition une infrastructure souple et évolutive permettant de travailler sur des applications.*NET*. La tarification se fait à l'utilisation. Cette plateforme permet l'exécution et l'hébergement de n'importe quel type d'application. On retrouve la notion de *Worker rôle*. La charge est automatiquement répartie par *Windows Azure*.

Azure Blobs, un service de stockage de fichiers, *Azure Tables*, un service de stockage de données semi structurées et *Azure Queue* qui est un service de files d'attente, sont mis à disposition au travers d'*API* standards. Ce qui permet de les utiliser depuis n'importe quelle application même extérieure au *Cloud*.

- *SQLAzure*, une base de données relationnelle qui permet de créer instantanément une base de données *SQL Server* sur une infrastructure fiable et garantie disponible à 99,95%.
- *AppFabric*, ensemble de services basés sur le Web qui garantit une connectivité sécurisée et un contrôle d'accès.

La plateforme *Windows Azure* offre tout un ensemble de services adaptés et souples.

L'expérience de développement, l'évolution à la demande, la réduction des délais sont de véritables atouts.

- **L'*IaaS* et *Cloud* privé:**

La notion de *VMrole* fournit une machine virtuelle qui permet aux utilisateurs d'héberger, de faire évoluer et de gérer les applications et services comme ils le désirent.

La solution *Microsoft* pour un *Cloud* Privé repose sur *Windows Server2008R2* ou bien *server 2012*,*Hyper-V*et *System Center*.

Hyper-V permet aux entreprises d'utiliser un *Cloud* privé réactif et efficace. *System Center* leur permet de créer des services *Cloud* au sein de leur propre *Data Center*.

Elles ont aussi la possibilité d'exploiter les services d'un partenaire d'hébergement *Microsoft*. Elles utilisent alors les services *Cloud* en se basant sur des machines virtuelles pour pouvoir augmenter la capacité de leur *Data Center*, et externaliser l'acquisition matérielle et les dépenses de gestion.

La surveillance et la gestion complète du matériel, des systèmes d'exploitation, des machines virtuelles, des applications physique sou virtuelles sont assurées par *System Center*.

System Center permet ainsi une vision globale des opérations du *Data Center*, simplifiant le dépannage et la maintenance, et optimisant l'efficacité du *Data Center*.

A noter que début 2011, l'Appliance *Microsoft Windows Azure*, a été lancée. Elle permet aux entreprises la création d'un *Cloud* privé, type *Windows Azure*, sur leur propre infrastructure.

III.1.2 Le *Cloud* de *VMware*:

La société *VMware* existe depuis maintenant 15 ans. Cette filiale d'EMC conçoit et vend des produits liés à la virtualisation d'environnement informatique. En 1999 elle propose un des ses produits : *VMware* Workstation, la version station de travail de son offre. Depuis *VMware* n'a cessé d'enrichir son offre et d'investir dans ce marché naissant:

- *VMware Fusion* : logiciel pour stations de travail Macintosh.
- *VMware* GSX Server, *VMware* Server et *VMwareESX/ESX$_i$ Server*: logiciel pour serveurs.
- *VMwareVirtual Center* et *VMware Converter*: logiciels de gestion et outils.
- Rachat de *Zimbra*, la plateforme collaborative qui est maintenant la base de sa solution *SaaS*.
- Rachat de *Spring Source*, renommé *vFabric* et complétant son offre *Paas*.

➢ **L'infrastructure selon *VMware* :**

Affichant fièrement son savoir-faire sur les plates-formes virtualisées, *VMware* complète sa gamme d'hyperviseurs *ESX/ESXi* grâce à sa suite d'outil *vSphere* Cette plate forme de virtualisation comprend un certain nombre de fonctions et de mécanismes permettant la gestion complète d'une infrastructure partagée(un cluster d'hyperviseurs par exemple).

Afin de répondre à des problématiques de répartition de charge, des fonctionnalités évoluées permettent de migrer à chaud des ressources d'un nœud vers un autre. Ainsi il sera possible de jongler entre plusieurs *ESXi* pour héberger un host grâce au *VMotion*, ou de migrer des volumes de données entre différents *SAN* grâce au *vStorage*.

La disponibilité et la garantie de service pourront être assurées avec les fonctions de *High Availibilty* (*HA*) et *Fault Tolerance* (*FA*), toujours incluses dans la plateforme*VSphere*. Ainsi, en cas défaillance d'un nœud, les *VMs* pourront basculer sur un *ESX* sain. Contrairement à *VMotion*, la migration se fera froid. *FA* quant à lui autorisera la création d'une instance active et d'autres passives sur plusieurs ESXi. De cette manière en cas de problème, la migration vers un nouvel hyperviseur se fera instantanément .

En parallèle, et pour une souplesse maximale, les mécanismes de *Distributed Resource Scheduler*(*DRS*)et de *Distributed Power Management*(*DPM*)vont autoriser une gestion plus poussée du parc de machines. Ainsi, le premier utilise le *VMotion* pour repartir les*VMs* sur les différents noeuds disponibles en fonction de la charge de chacun. De la même manière, le *DPM* arrête les machines les moins critiques afin de prioriser celles qui le sont le plus.

A l'aide de tous ces composants, *vSphere* garantit une très grande réactivité et permet de se prémunir contre des défaillances inopportunes.

Annoncé lors de la conférence *VMworld 2010, VMwarev Cloud Director* est une surcouche majeure dans l'approche du *Cloud Computing* de *VMware*. Elle est conçue pour regrouper les ressources d'une infrastructure virtuelle en *Datacenters* virtuels afin de les mettre à la disposition des utilisateurs sous la forme de services basés sur catalogues. v*Cloud* introduit la notion de*vApp*. Elle peut être vue comme un environnement applicatif complet comprenant machines et réseau dédié. Concrètement, une plateforme de supervision composée de trois nœuds et d'un autre dédié à la base de donnée sera vendue sous la forme d'un package, la *vApp*. Ce package permet dans un premier temps de concevoir des *Templates* autorisant la reconstruction complète d'une infrastructure. v*Cloud Director* offre ensuite la possibilité de consulter un catalogue en ligne de*vApp* prêt à l'emploi. De la même façon qu'un *Apple Store*, l'utilisateur pourra choisir son application au sens large du terme(*vApp*)a fin de l'implémenter. Les *vApps* se présentent sous le format *OVF* (*Open Virtualization Format*). Il conserve les propriétés de l'application, la configuration réseau ainsi que d'autres paramètres. L'interopérabilité entre *Cloud* est ainsi garantie.

Bien entendu *VMware* n'est pas à proprement parlé un fournisseur de services, il se limite à éditer des solutions *Cloud Computing*.

Il sera alors possible d'utiliser les technologies de *Cloud Computing* de *VMware* sur ses propres installations physiques, créant ainsi son propre *Cloud* privé.

On retrouvera aussi les technologies de *VMware* chez des fournisseurs de services tels que *Verizon*, *Terremark*, *Colt* ou prochainement *Orange* Entreprises qui proposera cette technologie sur les infrastructures physiques (*Cloud* public).

- **Les offres de *PaaS*:**

C'est à l'occasion du *VMworld 2011* que *VMware* annonce sa gamme *vFabric*. Elle est conçue pour porter des applications *Java* basées sur le*framework Spring* sur des machines virtuelles *vSphere*. On retrouve ainsi[6]:

- *TcServer*: une plateforme offrant l'hébergement d'applications *Spring*, *Grail* et *Java* basée sur *Tomcat*. Cela permet aux développeurs de déployer très facilement leurs applications en laissant la plateforme répartir la charge.
- *Hyperic*: permet de surveiller les applications Web personnalisées quelque soit leur emplacement, sur des machines physiques ou au sein d'un environnement d'infrastructure virtuelle, d'un *Cloud* public, privé ou hybride.
- *Gemfire* : est une plateforme de gestion de données distribuées qui offre une évolutivité dynamique, et une persistance rappelant celle des bases de données. Elle combine des

[6]Définition issue du site VMware

techniques avancées, telles que la réplication, le partitionnement, le routage orienté des données et l'interrogation continue.

- *RabbitMQ* : est un service de messagerie. Plus spécifiquement un logiciel de mise en file d'attente et évolutif qui simplifie la gestion du trafic des messages.

La plateforme *PaaSvFabric* peut être utilisée sur une infrastructure *VMware* (*vSphere*ou *vCloud*), ou sur une infrastructure partenaire comme celle de *Salesforces.com* (qui a donné naissance par la suite à *VMForce*),ou prochainement celle de *Google* qui a annoncé proposer *vFabric* au sein de *Google App Engine*.

- **Les Services à la demande:**

  Le *SaaS* de *VMware* est une informatique moderne qui prend en compte l'utilisateur. Elleoffreunegarantiedesécuritédesapplicationsetdesdonnées,quelsquesoientla source, le lieu, ou le moment.

En janvier 2010, *VMware*r achète *Zimbra*[7](*Yahoo*).Il s'agit d'une application Web collaborative qui inclut de l'organisation et du partage de documents, du stockage, des liens, une boite aux lettres, des contacts, des agendas. *Zimbra* revendique près de 55 millions d'utilisateurs dans 90 pays avec une croissance de 86 % en 2009. *VMware* a annoncé vouloir intégrer *Zimbra* dans son catalogue de *vApp* dans sa stratégie v*Cloud*.

III.2 Les solutions Open Source pour la création d'un *Cloud* privé:

Les solutions open source ne fournissent que le support logiciel (pas matériels) de la mise en place d'une véritable plateforme de *Cloud*. Il existe désormais un certain nombre de solutions open source, telles que:

- ***Eucalyptus*:**

  Issue d'un projet de recherche de l'université de Californie, cette plateforme *Cloud* open source est certainement la plus connue, car intégrée dans *Ubuntu Server* et *Debian*. Ecrite en *C*, *Java* et *Python*, elle permet de créer des *Cloud IaaS* de type privé ou hybride et supporte des machines virtuelles *Linux* ainsi que quelques hyperviseurs *Xen* et *KVM*. Par ailleurs, elle est compatible avec *EC2* d'*Amazon*. Il existe également une version propriétaire commercialisée par la société *Eucalyptus Systems*. Elle apporte des fonctionnalités supplémentaires comme le support de *VMware* des machines virtuelles *Windows* et l'intégration *SAN*.

- ***OpenNebula*:**

[7]Extrait de presse, JDN, 'VMware / Zimbra 7 : suite bureautique collaborative en mode SaaS

Cette plate forme purement opensource permet de déployer des *Cloud* privés, hybrides et publics. Ecrite en *C++*,*Ruby* et *Shell*, elle supporte les hyperviseurs *Xen*, *KVM* et *VMware*.

Le support de *Virtualbox* est prévu. Comme *Eucalyptus*, elle permet de s'interfacer avec le *Cloud* d'*Amazon*, *EC2*. Le projet est publié sous licence *Apache2.0*. Par ailleurs, *OpenNebula* est soutenu par le projet européen *Réservoir*, qui propose une architecture complète pour la gestion de *Datacenters* et la création de services *Cloud*.

- ***Niftyname*:**

Le seul projet d'origine française, la plateforme *Niftyname* a été créée par l'hébergeur *Ielo*. Elle est diffusées au licence*GPLv3*.Articulée autour d'un système de gestion écrit en *Python*, elle supporte l'hyperviseur *KVM* et permet de créer des machines virtuelles *Windows*, *Linux*, *BSD* et *Solaris*. Elle sait également gérer les fonctionnalités de stockage et de réseaux associés à ces machines.

- ***Nimbus*:**

Issue du monde de la recherche, *Nimbus* permet de déployer un *Cloud* de type *IaaS*. Diffusée sous licence *Apache2.0*,cette plateforme supporte les hyperviseurs *Xen* et *KVM*, et peut s'interfacer avec le *Cloud* d'*Amazon*, *EC2*.Elle est associée à un autre projet, baptisé *Cumulus*, qui permet de déployer des services de stockage en *Cloud*, compatible avec le service *Amazon S3*.

Nimbus a été déployé, entre autres, par un réseau d'universités américaines qui proposent des *Cloud* en libre accès pour des projets de recherche.

- ***OpenStack*:**

OpenStack est un ensemble de projets de logiciels opensource que les entreprises /fournisseurs de services peuvent utiliser pour configurer et gérer leur infrastructure de stockage *Cloud* .

Rackspace et la *NASA* sont les principaux contributeurs initiaux de la pile. *Rackspace* a contribué par leur plateforme *CloudFiles* (code) pour alimenter la partie du stockage d'objets de l'*OpenStack*, tandis que la *NASA* a apporté leur plateforme *nébuleuse* (code) pour alimenter la partie *Compute*.

OpenStack supporte les hyperviseurs *Xen*, *XenServer/XCP*, *KVM*, *UML*(*User-Mode Linux*), *VMware ESXi/ESX* et *Hyper-V*.

Synthèse:

	Eucalyptus	OpenNebula	Nimbus	OpenStack
Produit par	-Apparu au début par l'université Santa Barbara de l'université de Californie -Eucalyptus System Company	L'union Européenne	Chercheurs de l'université chicago	Rackspace, NASA, Dell, Citrix, Cisco, Canonical et plus que 50 autres organisations
But	Une réponse open source pour le *Cloud* commerciale EC2	Un *Cloud* privé pure	Solution scientifique du *Cloud Computing*	Créer et offrir des fonctionnalités de *Cloud Computing* en utilisant un logiciel open source fonctionnant sur du matériel standard
Systèmes d'exploitation supportés	Linux (Ubuntu, Fedora, CentOS, OpenSUSE et Debian)	Linux (Ubuntu,RedHat Entreprise Linux, Fedora et SUSE Linux Entreprise Server)	La plupart des distributions Linux	-Linux et récemment Windows
Langage de programmation	Java, C et Python	Java, Ruby et C++	Python et Java	Python
Stockage	Walrus	-GridFTP, Comulus (version récente de GridFTP -XCP	-SCP -SQLite3	OpenStack Storage
Réseau	Serveur DHCP installé sur le cluster controller	Configuration manuelle par l'administrateur	Serveur DHCP installé sur chaque nœud	OpenStackCompute
Interface utilisateur	-EC2 WS API -Outils tel que: HybridFox, ElasticFox	-EC2 WS API -OCCI API	-EC2 WS API -Nimbus WSRF	Interface Web

	-Fichier zip			
Sécurité niveau utilisateur	téléchargeable à travers l'interface Web qui contient certificats -Connexion HTTPS	Authentification	-Certificat X509	Authentification
Sécurité niveau administrateur	Le *Cloud* Controller	Nginx	Le context broker	Le *Cloud* Controller
Tolérance aux pannes	Séparation des clusters controllers	Database backend (enregistre les informations des machines virtuelles)	Vérification périodique des nœuds du *Cloud*	Réplication

Tableau II.1 : Solution *IaaS* open source de *Cloud Computing*

IV. Choix de la solution *IaaS open source* de *Cloud Computing* type *Cloud* privé:

Pour notre projet l'*OpenStack* est la bonne solution puisque:

OpenStack est l'une des solutions les plus importantes pour construire l'environnement *Cloud* et fournir une solution complète pour créer un multi-locataires et un centre de données redondant . L'utilisation de *OpenStack* offre des fonctionnalités aux contrôles de grands bassins de calcul, de stockage , et la mise en réseau des ressources à travers un centre de données, tous géré à travers un tableau de bord qui permet aux administrateurs de contrôler tout en permettant à leurs utilisateur l'accès à des ressources d'approvisionnement par le biais d'une interface web où bien portail web.

Conclusion:

Ce chapitre, a clarifié la couche la plus complexe de *Cloud Computing* qu'est l'Infrastructure as a service. Les différentes solutions existantes en montrant celle adoptée dans ce projet y ont été présentées. le chapitre suivant donne les détails de la solution choisie.

Chapitre III : la solution infrastructure as a service, opensource, openstack

III.1.Introduction:

Un logiciel *OpenSource* est un logiciel dont l'accès au code source est autorisé par son auteur. Il peut être modifié et donner lieu à des progiciels dérivés. Il repose sur une communauté de développeurs qui travaillent sur un même code source et procèdent à un échange de «bons procédés»: les modifications apportées sur les sources des logiciels sont mis à la disposition de la communauté de développeurs.

Au contraire, un logiciel Propriétaire est un logiciel développé par une personne ou une entreprise, qui en est le propriétaire. Les sources sont «encapsulées» et ne sont pas accessibles sans autorisation du concepteur. Les évolutions sont pilotées par ce dernier qui peut s'appuyer sur sa communauté d'utilisateurs pour faire évoluer le logiciel.

la solution Propriétaire est pilotée par un concepteur qui se charge de centraliser les développements et les évolutions.

Pour notre projet on a préféré utiliser une solution *OpenSource* qui est modifiable par un développeur, parmi celles citées dans le chapitre précédent, on a choisit *OpenStack.*

III.2.Définition d'*OpenStack*:

OpenStack est une pile logicielle libre, développée en *Python* et permettant la mise en place du *Cloud* de type ***IaaS***. L'outil *OpenStack,* un système d'exploitation *Cloud,* est beaucoup utilisé. En effet, *OpenStack* contrôle de larges bassins de ressources de calcul, de stockage et des réseaux à travers un centre de données, le tout géré par un tableau de bord qui donne le contrôle aux administrateurs tout en offrant à leurs utilisateurs des ressources d'approvisionnement à travers une interface web. Sa première version *Austin*a vu le jour en Octobre 2010, en suite beaucoup d'autres versions sont apparues (*Bexar*, *Cactus*, *Diablo*, *Essax* et *Folsom*). Les principales caractéristiques de cet outil, sont:

1. C'est un outil opensource conçu par la *NAZA* et l'entreprise *Rackspace*[*RAC*13].

2. Il est supporté par beaucoup de prestataires de *Cloud Computing* et d'entreprises tels que: *Yahoo!*, *VMware*, *Cisco*, *HP*, *IBM* et bien d'autres[*RAC*13].

3. Il s'installe sur différents types de machines tels que les serveurs, les ordinateurs portables, les tablettes, etc.[*OPE*12b].

4. Plusieurs distributions de *Linux* peuvent jouer le rôle d'hôte pour l'installation d'*OpenStack.* Ses paquets sont disponibles sur les distributions suivantes: *CentOS, Debian, Fedora*, *RHEL*, *OpenSuse* et *Ubuntu*[*OPE*12b].

5. Il existe une communauté *OpenStack* active.

Les différents outils d'*OpenStack* ainsi que les technologies intervenants dans son fonctionnement sont expliqués ci dessous.

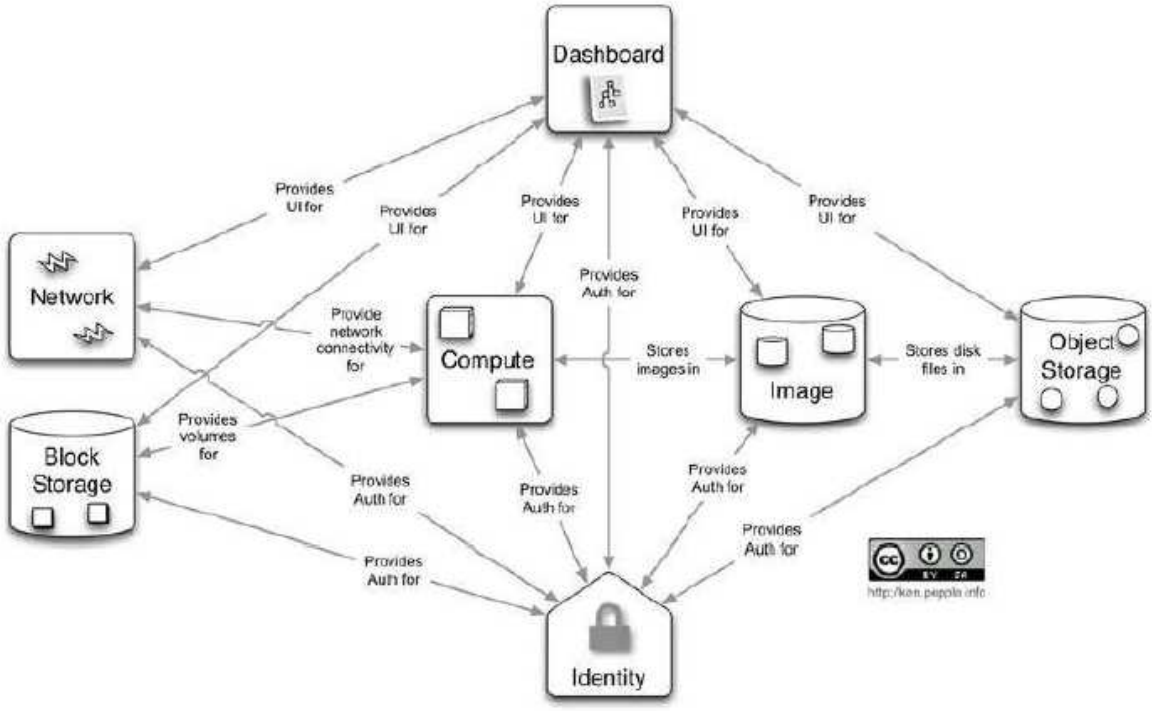


Figure3.1: Architecture interne d'*OpenStack*

la figure3.1 montre l'architecture interne d'*OpenStack* qui est composé de différents outils qui s'installent séparément mais qui travaillent ensemble selon les besoins du Cloud. Des objets ont été développé sa fin d'atteindre ce but.

Un service pour gérer les authentifications des utilisateurs et leurs droits d'utilisation (*Identity*), un outil pour le stockage de donnée (*ObjectStorage*), un autre pour le stockage des images des systèmes (des machines virtuelles) (*Image*), un autre pour la gestion du calcul (*Compute*) et un tableau de bord pour l'interface (*Dashboard*).

III.3 Description des différentes composantes d'*OpenStack*:

Il existe actuellement cinq composantes essentielles d'*OpenStack*.

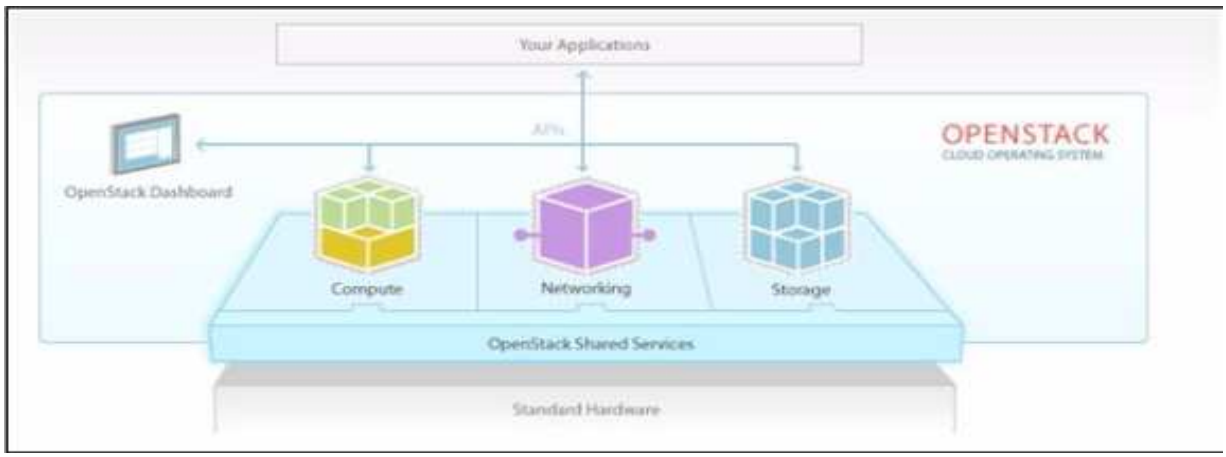


Figure3.2: Composants d'*OpenStack*

- **Le service Identité**(*Identity service*) *Keystone*:

  Keystone joue le rôle de la pierre pionnière d'*OpenStack*, tous les autres composants reposent sur ce système d'authentification. Plusieurs concepts sont rattachés au service d'identité et sont importants à connaitre. Ils sont expliqués en détails dans le tableau suivant:

Concept	Définition
Utilisateur (User)	Une représentation numérique d'une personne, système ou service qui utilise *OpenStack*. Le Services d'authentification d'identité valide la demande entrante faite par l'utilisateur qui prétend a voir fait la demande.
Information d'identification (Credentials)	C'est des données qui prouvent qu'un utilisateur est ce qu'il prétend. Exemple: -La correspondance nom utilisateur et mot de passe. -La correspondance nom utilisateur et une clé API. -Un jeton délivré que personne d'autre ne connait.
Authentification	Le service d'identité confirmera que les demandes entrantes sont faites par l'utilisateur qui prétend a voir fait la demande. Après la confirmation initiale, le service de l'identité de l'utilisateur émettra un jeton avec lequel l'utilisateur peut prouver son identité authentifiée lors des demandes subséquentes.
Jeton (Token)	Un jeton est un assemblage arbitraire de texte qui est utilisé pour accéder à des ressources. Chaque jeton a une portée qui décrit quelles ressources sont accessibles avec elle. Un jeton peut être révoqué à tout moment et est valable pour une durée limitée.
Locataire (Tenant)	Récipient utilisé pour isoler ou regrouper des ressources et/ou des objets d'identité. Selon l'opérateur de service, un locataire peut correspondre à un client, compte, une organisation ou un projet.
Service	Un service *OpenStack*, comme *Compute* (*Nova*), objet de stockage(*Swift*) ou le Service Image (*Glance*). Un service fournit un ou plusieurs effets grâce au quelles utilisateurs peuvent accéder à des ressources et exécuter des opérations.
Endpoint (Pointd'accès)	Une adresse réseau accessible, le plus souvent décrit par URL, où un service peut être consulté.
Le rôle	C'est la définition du rôle de l'utilisateur sur un locataire. Un utilisateur peut avoir un ou plusieurs rôles sur différents Tenants. Dans le service d'identité, un jeton qui est délivré à un utilisateur comprend la liste des rôles que l'utilisateur peut assumer.

Tableau3-1: Concepts du service *Keystone*

❖ **OpenStack Compute « NOVA » :**

Développé à l'origine par la NASA, c'est le cœur d'OpenStack. Il s'occupe principalement de la gestion des hyperviseurs (ordonnanceur et gestion des machines virtuelles) par l'intermédiaire de la libvirt ou directement par des API . Aujourd'hui l'hyperviseur le mieux supporté reste KVM, mais nova fonctionne aussi avec Xen, ESX, et Hyper-V voire avec des gestionnaires de conteneur comme Docker. Il contrôle aussi les ressources (CPU, RAM, réseaux et stockages) L'architecture de la brique de Nova est conçue pour évoluer horizontalement en rajoutant du matériel. D'ailleurs Nova fonctionne avec du matériel non spécialisé ce qui permet de réutiliser des serveurs existants par exemple.

Le tableau ci-dessous va nous permettre de comprendre l'architecture Nova Compute et les rôles de chaque composant :

Le composant Nova	**Le rôle**
API	✓ Cœur de Nova ✓ Fonction Principale : Cloud Controller avec le servicenova-api. ✓ Compatible avec l'API AmazonEC2 ✓ Ecoute sur le port 8773 pour EC2 API et 8774 pour OpenStack API ✓ Initialise la plupart des activités ✓ Renforce certaines fonctionnalités (ex :quotas)
Scheduler	✓ Principe simple : il prend une demande d'instance de machine virtuelle et détermine où (quel « Compute server ») doit-elle être exécutée. ✓ Fonctionnement par algorithmes pour assurer un fonctionnement optimal. ✓ 3 choix d'ordonnancement: - Simple : tente de trouver l'hôte le moins « chargé» - Chance (celui par défaut) : choisit un hôte disponible au hasard depuis sa « Service Table» - Zone : Prend un hôte au hasard depuis une zone « disponible»
Nova Compute	✓ Créé et termine les instances de machines virtuelles ✓ Reçoit et exécute des actions visant à mettre à jour les états des VM dans la base de données ✓ Supporte plusieurs API : KVM, Xen, Citrix, VMware,Hyper-V,

Nova Volume	✓ Gère la création, l'attachement et le détachement de volumes persistants. ✓ Compatible avec AoE, iSCSI (dont Solaris ZFS), Sheep dog, RBD, Left Hand(HP).
Nova Network	✓ Configure les interfaces bridge ✓ Adapte les règles de pare-feu(Iptables) ✓ 2 types d'adresse IP pour une instance: - Adresse fixe :privée
Queue	✓ Point de passage obligé pour les instructions échangées entre les services ✓ Différents types de files d'attente de messages pour faciliter la communication : Topics, Fanout,Host…
Data Base	✓ Enregistre la configuration et les états en temps réels pour une infrastructure Cloud : types d'instances disponibles, instances en cours d'utilisation, réseaux disponibles, projets, ✓ Supporte la plupart des SGBD : MySQL, PostgreSQL

Tableau : Les composants de Nova

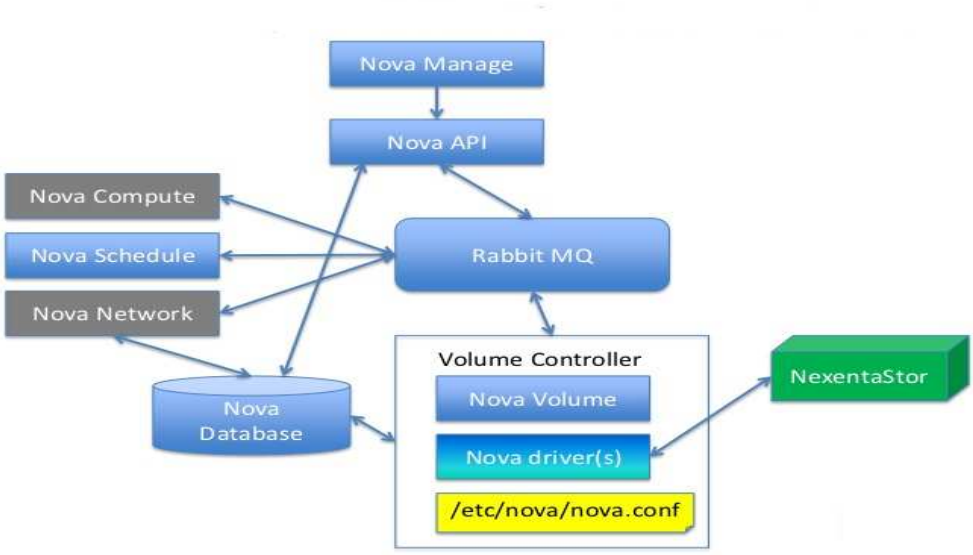


Figure 3.3: Architecture de Nova

❖ **Le service Object Storage « SWIFT » :**

Le stockage objet d'OpenStack s'appelle Swift. C'est un système de stockage de données redondant et évolutif. Les fichiers sont écrits sur de multiples disques durs répartis sur plusieurs serveurs dans un Datacenter. Il s'assure de la réplication et de l'intégrité des données au sein du cluster. Le Cluster Swift évolue horizontalement en rajoutant simplement de nouveaux serveurs. Si un serveur ou un disque dur tombe en panne, Swift réplique son contenu depuis des nœuds actifs du cluster dans des emplacements nouveaux. Puisque toute la logique de Swift est applicative, elle permet l'utilisation de matériel peu couteux et non spécialisé.

En aout 2009, c'est Rackspace qui a commencé le développement de Swift, en remplacement de leur ancien produit nommé Cloud Files. Aujourd'hui c'est la société Swift Stack qui mène le développement de Swift avec la communauté.

Object Storage est idéal pour le stockage rentable. Il fournit une plateforme distribuée accessible via l'API qui peut être intégrée directement dans les applications ou utilisée pour la sauvegarde, l'archivage et la conservation des données.

L'Architecture de Swift est composée par les éléments suivants:

- **Proxy Server**
  - ✓ Partie visible (public) de Swift,
  - ✓ Détermine le nœud de stockage approprié,
  - ✓ Coordonne les réponses.
- **L'anneau (Ring)**
  - ✓ Lie les requêtes au nœud de stockage,
  - ✓ Gère les zones de disponibilité,
  - ✓ Extensible sans affecter les autres entités.
- **Serveurs de stockage (Comptes et Containers):**
  - ✓ Base de données SQLite,
  - ✓ Les groupes de containers et les objets sont contenus dans les comptes,
  - ✓ Schéma simple : table pour les listes et table pour les métadatas.
- **Serveurs de stockage (Objet):**
  - ✓ stocker les fichiers,
  - ✓ Fichiers nommés avec un marqueur de temps.
- **Serveurs de consistance**
  - ✓ Réplicats.
  - ✓ Gestion des mises à jour.

- ✓ Gestion des audits.

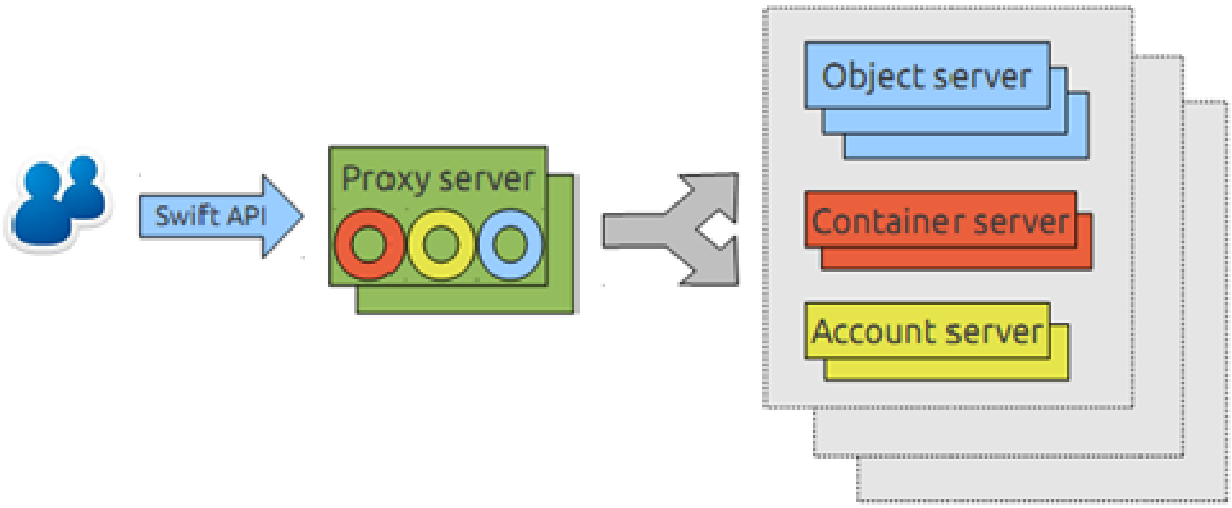


Figure 3.4 : Architecture de Swift

❖ **Le Service Image « GLANCE »:**

Glance a été extrait rapidement de Nova pour en faire un composant à entière. Imaging Service fournit les services de stockages, de découvertes, d'enregistrements et de distributions pour les images disques de machines virtuelles. Il fournit également une API compatible REST permettant d'effectuer des requêtes pour obtenir des informations sur les images hébergées par les différents magasins de stockages .

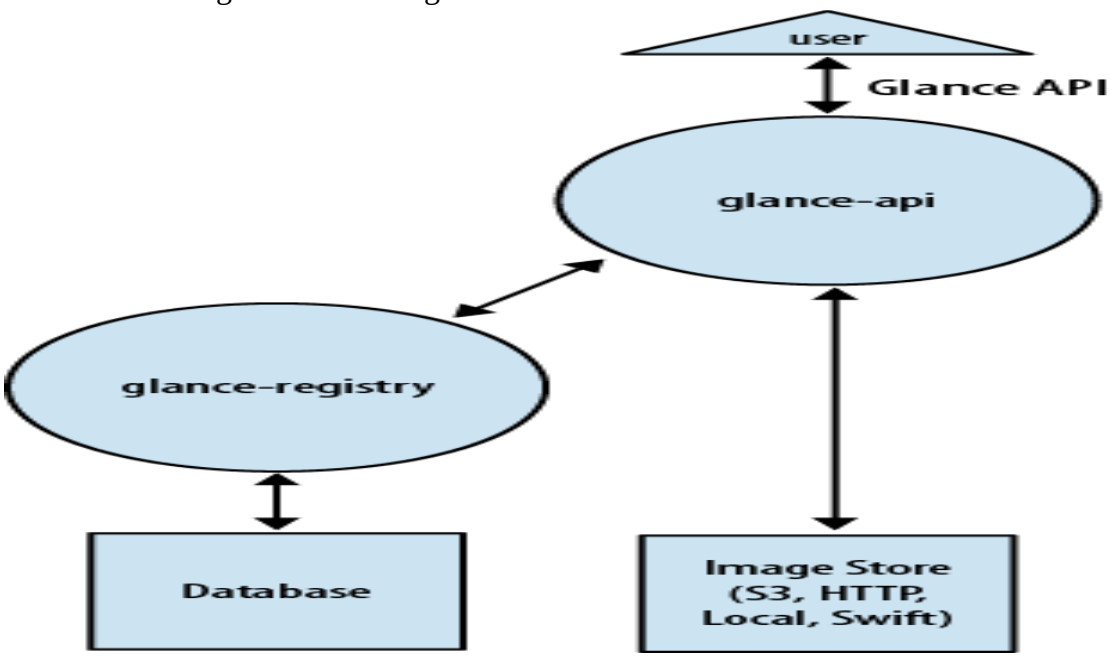


Figure3.5: Architecture de Glance**.**

- **Le tableau de bord *OpenStack* (*Dashboard HORIZON*):**

  Le tableau de bord *OpenStack* fournit aux administrateurs et aux utilisateurs une interface graphique pour l'accès, la fourniture et l'automatisation des ressources basées sur le Cloud. Sa conception extensible rend facile l'interaction avec l'utilisateur et l'exposition des produits et services, tels que la facturation, le suivi et les outils de gestion supplémentaires. Le tableau de bord est aussi attractif pour les fournisseurs de services et autres fournisseurs commerciaux qui veulent en faire usage[*OPE*12b].

  Le tableau de bord aide à l'interaction de l'administrateur avec les ressources *OpenStack*.

  Les développeurs peuvent automatiser l'accès ou construire des outils pour gérer leurs ressources en utilisant l'*API OpenStack* natif ou l'*API* de compatibilité *EC2* [*OPE*13c].

- **Quantum :**

  permet d'offrir une gestion des réseaux à la demande à l'intérieur de son Cloud. Le service permet aux utilisateurs de créer des réseaux à la demande et d'y attacher des machines virtuelles. Quantum a une architecture ouverte grâce à des plugins permettant de supporter différents fournisseurs de réseau ou des technologies réseaux différentes.

- **Cinder :**

  permet d'offrir des disques persistants pour les machines virtuelles. Ce service était inclus dans Nova à l'origine (sous le nom nova-volume) dans les versions précédente de OpenStack.

III.4 L'hyperviseur utilisé:

Les hyperviseurs les plus utilisés sont *KVM* et *Xen*. *KVM* travaille sous *libvirt13*, *Xen* fonctionne mieux par les appels de *Xen API*. *KVM* est utilisé par défaut malgré le fait qu'il nécessite une configuration supplémentaire car il supporte plusieurs formats d'images de machine virtuelles: *RAW, QEMU* (*Copy-on-Write*).

- **Définitionde *KVM*:**

L'hyperviseur *KVM* (*Kernel-based Virtual Machine*) est une solution de virtualisation basée sur le système d'exploitation *Linux*. Il est implémenté comme un module du noyau qui convertit le noyau *Linux* en un hyperviseur.

L'hyperviseur *KVM* est vraiment un système d'exploitation spécialisé, il permet à des machines virtuelles de s'exécuter contrairement aux systèmes d'exploitation ordinaire qui permettent à des applications de s'exécuter. L'hyperviseur *KVM* fournit plusieurs avantages tels que[*WIL*11]:

1-*KVM* permet un grand isolement entre les hôtes en ajoutant une couche supplémentaire de protection.
2- *KVM* est testé régulièrement par les développeurs pour détecter toute anomalie et régler le problème dans des délais record.
3-*KVM* réduit le coût total de possession et, est plus souple que les autres hyperviseurs.

Figure 3.6: Emplacement de l'hyperviseur *KVM* dans notre architecture.

Dans l'architecture *KVM*, les machines virtuelles sont implémentées comme des processus *Linux* classiques. En effet, chaque processeur virtuel joue le rôle d'un processus *Linux*. Ceci permet à *KVM* de bénéficier des fonctionnalités du noyau *Linux*.
Cependant, pour atteindre ces objectifs, l'hyperviseur *KVM* a besoin de la technologie de virtualisation du processeur (connu aussi sous le nom de virtualisation matérielle), donc ceci doit être pris en compte lors du choix du matériel à utiliser.

A ce niveau, une fois *KVM* et *OpenStack* installés, il est possible de déployer un Cloud *IaaS* car *KVM*, étant un outil de virtualisation, permet de créer des machines virtuelles et *OpenStack* gère ces machines et offre une interface à l'utilisateur a fin d'accéder à ces dernières via un navigateur web.

Pour le déploiement du *Cloud* autour d'un *DataCenter* comportant plusieurs serveurs, il suffit d'installer l'hyperviseur *KVM* sur chacun des serveurs du *DataCenter*, et de configurer certains fichiers de configuration pour les relier entre eux, chose qui est simplifiée grâce à la plateforme *Cloud OpenStack*.

III.5 Architecture Globale de la plateforme:

Lafigure3-7 donne le schéma global de notre système et de son fonctionnement et montre ses différents composants:

Le *Cloud* sera déployé sur le matériel (serveurs) grâce à l'hyperviseur qui est situé au plus bas niveau de l'environnement matériel et qui va jouer le rôle de mécanisme de transport dans le *Cloud Computing* pour l'affichage des applications sur de nombreux environnements d'exploitations sans avoir à les copier. L'hyperviseur est un moyen très pratique de virtualiser des *OS* rapidement et efficacement. Il peut aussi être configuré pour partager les ressources de l'ordinateur physique (ou ensemble de serveurs sur le schéma).

Par exemple, les ressources peuvent être divisées 50%-50% ou 80%-20% entre les deux systèmes d'exploitation invités sans que ces derniers ne se soucient du fait qu'ils s'exécutent sur une partition virtuelle (ils auront l'impression de disposer d'un ordinateur à part entière).

Ensuite, plusieurs machines virtuelles sont crées dans ce *Cloud*. Elles–joueront le rôle de serveurs virtuels sur les quels vont être installés les différents systèmes d'exploitation serveurs et les applications que les clients vont utiliser.

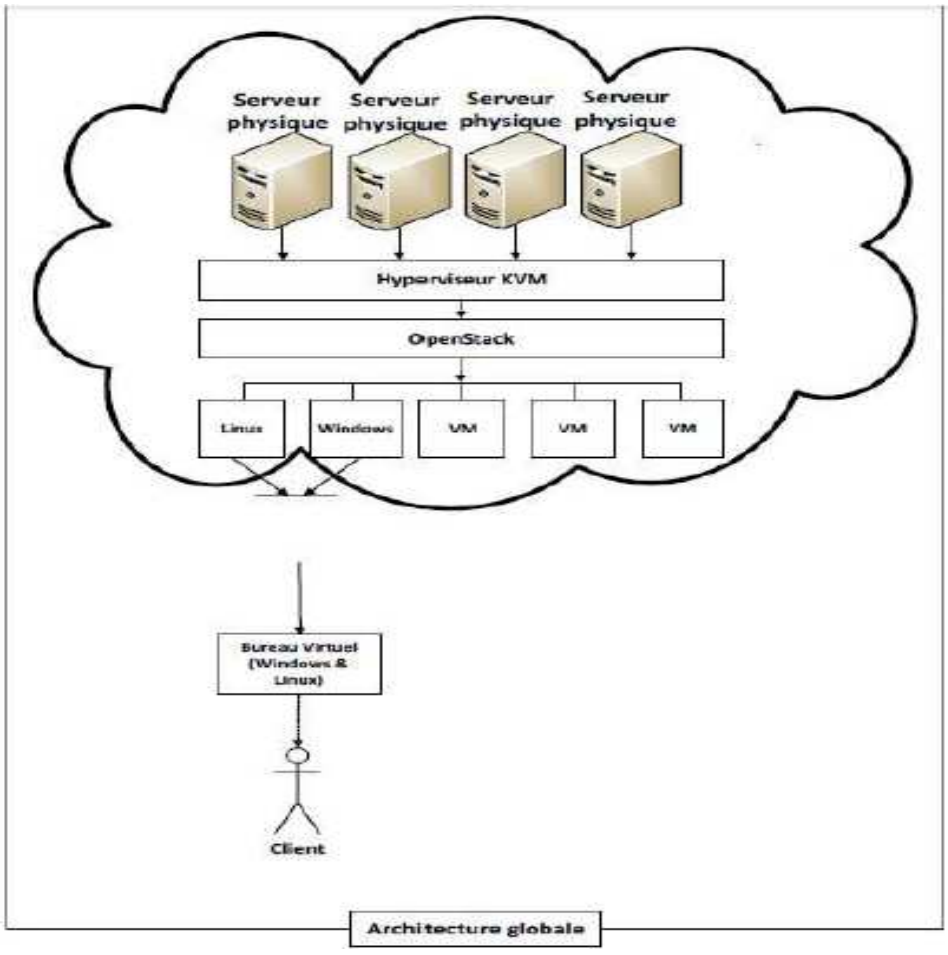


Figure3-7: Schéma global de la solution

III.6 Technologies utilisées:

OpenStack s'installe sous toutes sortes de machines. Néanmoins, il requiert certaines ressources matérielles pour son bon fonctionnement. Il a besoin essentiellement d'un processeur et de deux cartes réseaux:

a) **Un processeur**: supportant la virtualisation matérielle ou virtualisation du processeur.

Cette approche de virtualisation améliore les performances des environnements virtuels en faisant appel aux capacités des machines physiques, spécialement le processeur.

A l'aide de la virtualisation matérielle, l'hyperviseur est en mesure de virtualiser correctement l'ensemble des instructions du *CPU*[*SAN*10].

Les deux plus grands fabricants de processeurs sur le marché (*Intel* et *AMD*) se sont intéressés à l'intégration de la virtualisation matérielle dans leurs produits. Ainsi chacun a mis sur le marché une nouvelle gamme de processeurs supportant cette technologie: *VT-x* pour Intel et *AMD-V* pour AMD . Les deux technologies sont très semblables [*SAN*10]. Aujourd'hui presque tous les processeurs AMD disposent de cette technologie (excepté les *Sempron* et certains *Turion*) alors que chez Intel cela diffère d'un produit à un autre. Cette dernière propose une liste de ces produits qui prennent ou qui ne prennent pas en compte la virtualisation, mise à jour régulièrement.

b)Deux cartes réseaux: Comme le montre la figure3-1, les services *OpenStack* sont répartis sur 2 nœuds:

- Le nœud de calcul(*Compute node*): il héberge *Nova Network*, *Nova compute* et *KVM*.
- Le nœud de contrôle (*Control node*): Héberge les services *Keystone*, *Glance*, *Nova service*, *Horizon* et *Cinder*.

Ces deux nœuds sont généralement installés sur deux machines distinctes (deux serveurs) ayant chacune sa propre adresse *IP* leurs permettant de communiquer entre eux, mais il est possible de les rassembler sur une seule machine à condition que cette dernière dispose de deux cartes réseaux pour bénéficier d'une adresse *IP* privée pour le réseau local de configuration d'*OpenStack* et l'autre pour le réseau des machines virtuelles. C'est cette méthode qui est utilisée pour notre étude en travaillant sur *Vmware workstation* pour simuler deux cartes réseau du moment que notre machine physique n'a qu'une seul carte réseau.

III.7.Matériel utilisé:

Dans le cadre de la réalisation de notre solution le système Linux a été préparé pour *RDO Mitaka*,a fin d'installer *OpenStack*.

Le matériel utilisé a les caractéristiques suivantes:

- **Processeur:** Intel(R) Core (TM)I5-2430MCPU@2.40GHz2.40GHz.

- **Disque dur:** 500GO.
- **Mémoire:**8GodeRAM.
- **2cartes réseaux virtuelles:**

  Une carte réseau *Etherneteth0*,@locale .

  Une carte réseau *Wifiwlan0*,@publique.

III.8.Le système d'exploitation utilisé:

CentOS(***C****ommunity* ***ent****erprise* ***O****perating* ***S****ystem*) est une distribution *GNU/Linux* principalement destinée aux serveurs. Tous ses paquets, à l'exception du logo, sont des paquets compilés à partir des sources de la distribution *RHEL*(*RedHat Enterprise Linux*), éditée par la société *RedHat*. Elle est donc quasiment identique à celle-ci et–est à 100% compatible d'un point de vue binaire.

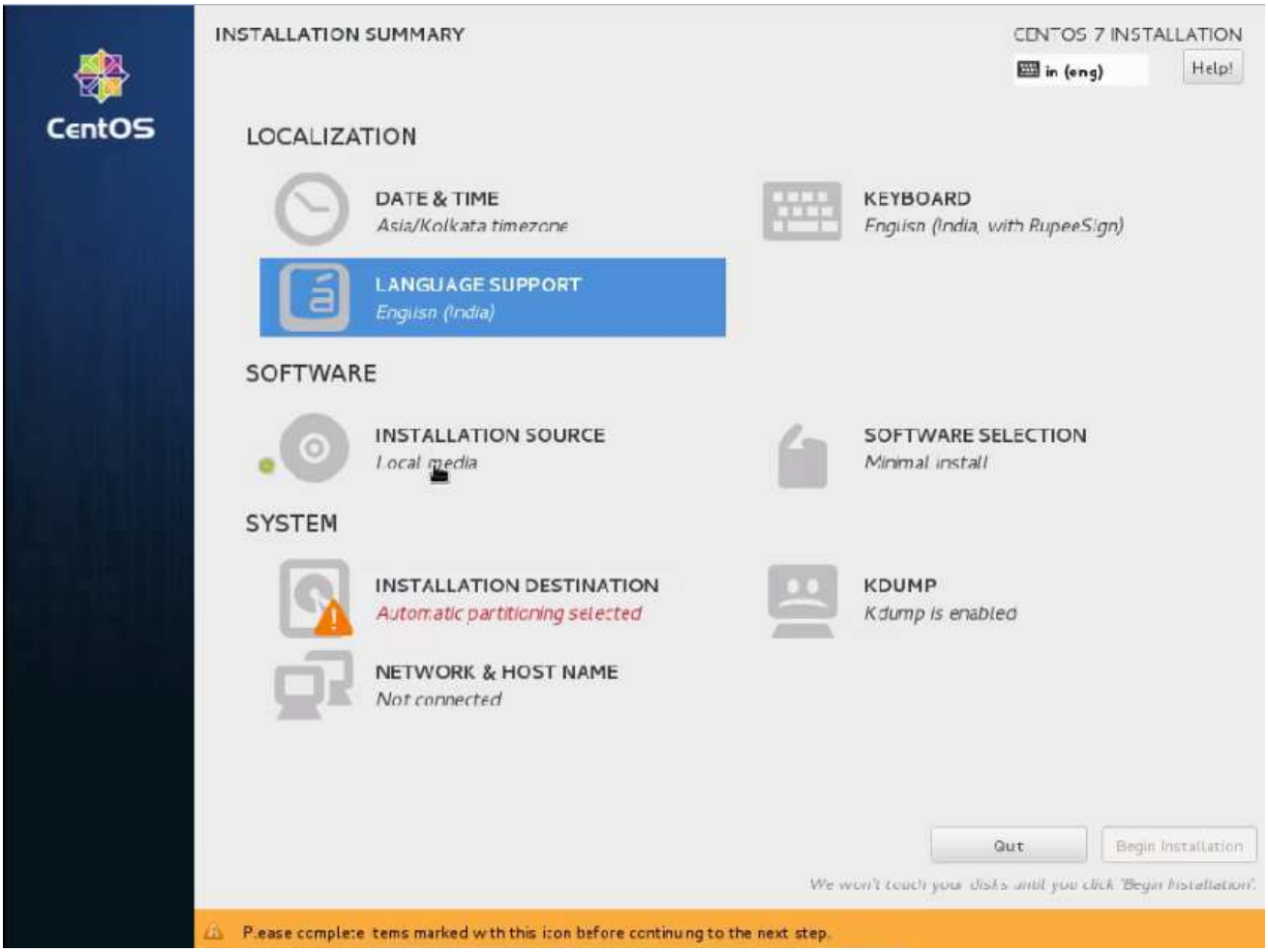


Figure3.8:Sélection des logiciels à installer manuellement

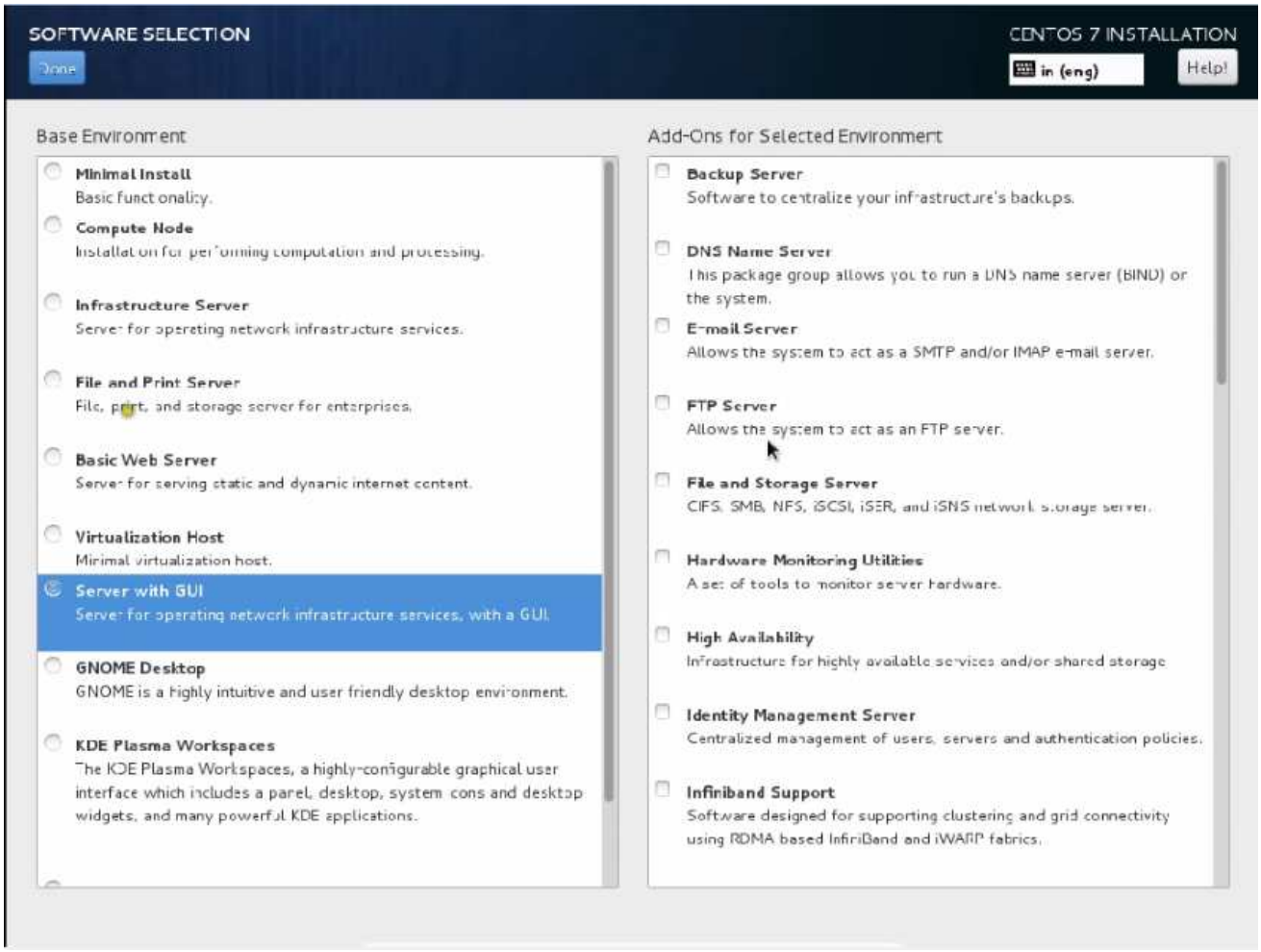


Figure3.9: liste des logiciels à installer

III.9 Déploiement du *Cloud*:

Dans cette section, tous les composants qui interviennent dans l'installation d'*OpenStack* pour la mise en place du Cloud, seront abordés. Ensuite, les étapes suivies pour l'installation des différents services seront détaillées et enfin les fonctionnalités seront présentées .

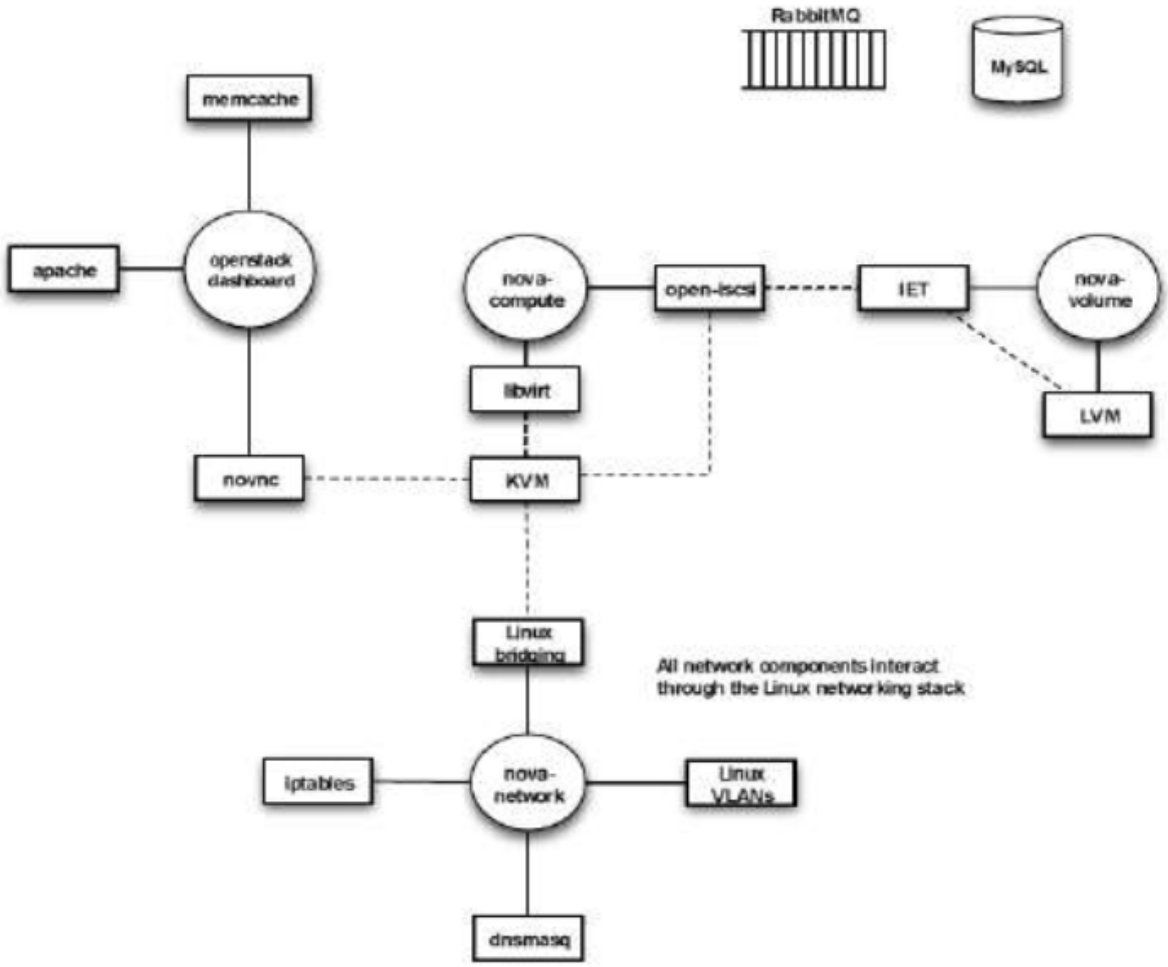


Figure3-10: Outils utilisés pour le déploiement d'un Cloud avec *OpenStack*

Comme le montre **la figure3-10**, *OpenStack* est une boite à outils qui sert à construire un environnement *Cloud Computing* en reliant un ensemble de technologies *linux*:

(Rappelons que *Dashboard* est le tableau de bord qu'utilise l'administrateur pour la gestion du Cloud, *KVM* est l'hyperviseur utilisé par *OpenStack* et *Libvirt* est l'*API* utilisé par *KVM*).

a) **Services d'*OpenStack*:** représentés par le symbole

- *Nova-Compute*, *Nova-Network*, et *Nova-Volume*: sont les trois composantes du service *Nova*. Dans la nouvelle version d'*OpenStack* le service *Nova-volume*, qui sert à la gestion des volumes crées, a été éliminé et remplacé par le service *CINDER*.

- Les services *Keystone* et *Glance* ne sont pas inclus dans le schéma, ceci ne diminue pas de leur importance pour le fonctionnement d'*OpenStack*. Nous verrons dans ce qui va suivre l'installation et la configuration de ces différents services.

b)Services *Linux* externes: non pris en charge par *OpenStack*, représenté par le symbole

- ***LVM* (***Logical Volume Manager***):***LVM*** crée une couche d'abstraction sur le stockage physique permettant, ainsi, de créer des volumes de stockage logiques. Un volume logique fournit la virtualisation du stockage. Avec un volume logique, l'utilisateur n'est pas limité à la taille des disques physiques[*RED*07].

- ***NoVNC*:** c'est le nom d'un client *VNC* qui est utilisé pour fournir l'accès à la console *VNC* associé aux instances de *KVM* (machines virtuelles)[*OPE*12b].
- Interactions entre les composants *Open Stack* et les composants externes (représenté par le symbole_____).
- Interactions entre les composants externes (représenté par le symbole----------).

Les composants *OpenStack* interagissent avec un outil de transfert de messages *RabbitMQ* et un outil pour les bases de données *MySQL*:

- **SGBD utilisé**: *MariaDB*

  C'est un système de gestion de base de données, il s'agit d'un *forck* communautaire de MySQL qui est un système opensource qui utilise *SQL*, le langage le plus connu pour l'ajout, l'accès et le traitement des données dans une base de données. MySQL est utilisé a fin de créer les bases de données pour les services *OpenStack*.

- **Outil de transfert de messages:** *RabbitMQ*

  RabbitMQ est une solution de messagerie d'entreprise complète et fiable qui se base sur le standard *AMQP*(*Advanced Message Queuing Protocol*). Elle est écrite en *Erlang*, publiée sous licence *MPL*[*RAB*13]. C'est l'outil qui permet la communication entre les différents composants d'*OpenStack*.

III.10 Déploiement(installation des outils):

Les différentes étapes d'installation d'*OpenStack* sont représentées dans le diagramme de la Figure3-8, il schématise les étapes à suivre, depuis l'installation du système d'exploitation *Centos7* jusqu'à arriver aux différents services du *Cloud* (*Keystone*,*Glance*,etc.), en passant par l'installation des différents outils: *MySQL*, *RabbitMQ*, *KVM*, etc.,et la configuration du réseau.

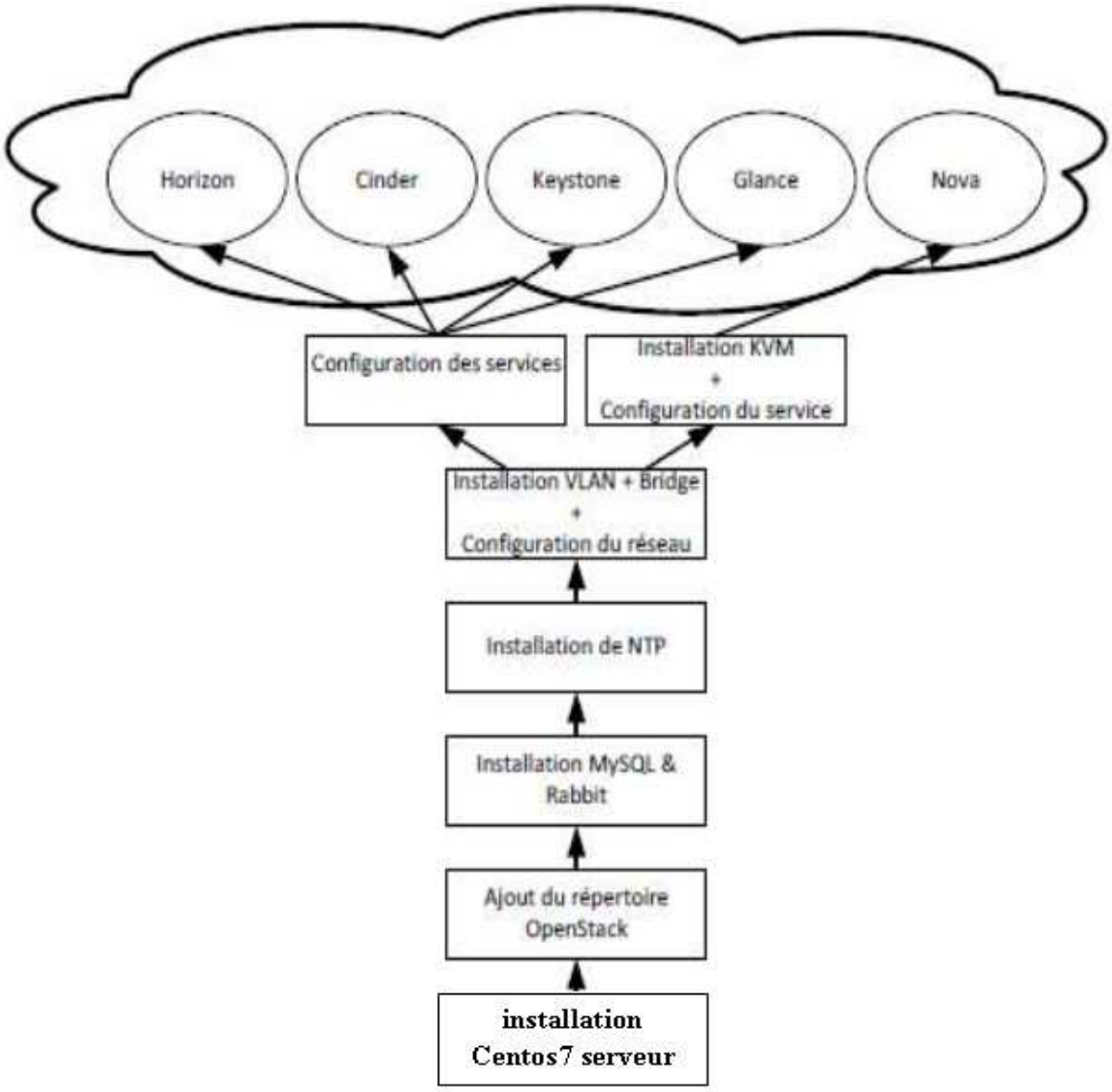


Figure3.11: Etapes d'installation d'*OpenStack*.

Il existe plusieurs méthodes pour installer *OpenStack*, comme *Devstack*, *Mitaka RDO*, etc.

La méthode *Mitaka RDO* a été utilisée pour Installer et configurer *OpenStack* sur *CentOS7*. En effet, *Mitaka* est la version la plus récente de *OpenStack*

III.11 Préparation du système Linux pour RDO Mitaka:

La documentation n'est pas très claire au sujet du choix du bon système, c'est pour ce la que nous avons dû procéder à de nombreuses tentatives, qui nous ont finalement guidées vers Centos7 serveur 64 bits que nous avons adoptée.

La configurer de cette dernière version Mitaka sur CentOS 7 système en utilisant RDO et packstack installateur.

- **Qu'est-ce que Mitaka?:**

OpenStack Mitaka facilite la gestion de notre centre de données et ses ressources connexes. On peut redimensionner vos nuages composants d'infrastructure comme Storage, Compute, la mise en

réseau des ressources, etc. pour répondre aux besoins informatiques sans cesse croissants de vos clients. OpenStack gagne en popularité au rythme de la fusée, de sorte que son tout à fait évident que cela va être des leaders dans les technologies de Cloud Computing. cycle de sortie OpenStack est de 6 mois, ce qui signifie que, après tous les six mois une nouvelle version et un nouvel ensemble de fonctionnalités sont ajoutées à cela. Mitaka est la continuation de cette entreprise.

- **Qu'est-ce que RDO?:**

En mots plus simples, son de la communauté des gens prospères faire OpenStack fonctionner parfaitement sur RHEL et ses dérivés (CentOS, Fedora, Scientific Linux). Donc, cette communauté est l'un de la communauté la plus active pour les utilisateurs CentOS / RHEL en ce qui concerne OpenStack. Il est synonyme de "RPM Répartition des OpenStack» et tout le monde est invité à poser des questions ou de contribuer à cette communauté en utilisant son canal de communication officiel.

- **Qu'est-ce que Packstack?:**

Packstack est l'utilitaire d'installation qui utilise l'application de marionnettes pour déployer OpenStack. Pour preuve de concept (POF) fins, on utilise Packstack pour installer et configurer OpenStack sur une seule machine. Il est le moyen le plus simple et automatisé pour commencer avec OpenStack. Si vous cherchez à déployer RDO à des fins de production sur plusieurs hôtes .

III.12 Les APIs (Application Programming Interface) OpenStack:

1. Introduction :

les services OpenStack fournissent une API à travers laquelle notre logiciel peut communiquer avec le service, qu'on peut utiliser pour le logiciel d'écriture qui gère un Cloud . par exemple, on peut écrire des scripts qui exécutent les emplois d'allouer automatiquement les serveurs, ou d'écrire des applications qui permettent à un utilisateur de gérer leurs propres serveurs Cloud .Les possibilités sont infinies. Mais pour accomplir ces tâches ,votre logiciel doit communiquer avec l'installation de OpenStack en utilisant l'API .

Il y a deux façons que vous pouvez utiliser OpenStack - API et SDK pour interagir avec un portail développer par un utilisateur .

2. Le rôle de l'API(Application Programming Interface):

Une interface de programmation permet de gagner du temps par exemple à un programme d'accéder aux services offerts par le système d'exploitation qui héberge le programme. L'interface *sockets* est un exemple classique d'interface de programmation qui permet à un programme d'exploiter les possibilités de la couche réseau du système d'exploitation.

Une des interfaces les plus connues est la *Windows API*. C'est une collection de fonctions, de types de données et de constantes, en langage de programmation C, qui permet à des programmeurs de

créer des applications pour les systèmes d'exploitation Windows. Elle offre la possibilité de manipuler des fichiers, des processus, communiquer par les réseaux et manipuler des interfaces graphiques.

3. Définition API OpenStack:

L'API (Application Programming Interface) est une interface RESTful, ce qui signifie que le format du chemin dans l'URL détermine la commande que vous envoyez, on envoie une demande en utilisant un HTTP GET ou HTTP POST. (Il existe d'autres possibilités en dehors de GET et POST, mais ceux-ci sont les plus couramment utilisés.) En plus de l'URL elle-même, pour une requête GET vous envoyez parfois des données supplémentaires sous la forme de demandes d'en-tête, et pour un POST vous pouvez spécifier les données qui est envoyé en même temps que la demande sous la forme de données POST.

Ces URL utilisent le même protocole HTTP que les navigateurs Web font .Cela signifie que certaines interfaces REST peuvent faire quelques-unes des demandes directement à partir de la barre d'adresse d'un navigateur, mais cela ne fonctionnera que s'il n'ya pas des informations personnalisées à inclure avec la demande. (Et ce n'est pas vraiment le but du navigateur, de toute façon.). Au lieu de cela, nous allons faire les demandes à partir de notre code de programme, ou en utilisant un utilitaire de ligne de commande, comme ce qu'on appelle cURL. cURL est l'un des moyens les plus faciles à envoyer des demandes aux services, et ca ne vaut pas seulement juste pour *OpenStack***.**

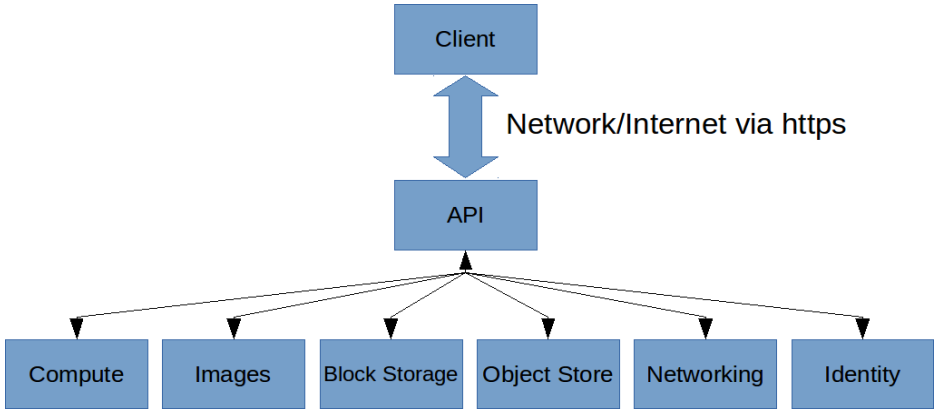


Figure3.12: la manière d'interagir avec les services OpenStack en utilisant API Openstack

4. La raison derrière OpenStack SDKs (Kit de développement logiciel):

L'API est le pipeline de base de la communication au service, en utilisant l'API peut être un peu lourd. Sauf si on veut juste faire quelques demandes de temps en temps, en utilisant cURL pour appeler l'API pourrait être très bien. Mais si vous écrivez des applications qui ont besoin de faire beaucoup d'appels, on se trouve avoir à répéter une grande partie du même code à plusieurs reprises. En fonction de langage qu'on utilise, on aura à mettre ensemble les données qu'on veut envoyer à la demande, de construire l'URL, et d'appeler dans tout ce que

les bibliothèques de la langue dispose pour effectuer les requêtes HTTP. Ensuite, on récupère un code d'état HTTP et une réponse, qu'on pourrait avoir à décoder dans un objet de notre langage de programmation peut comprendre. Voilà où SDKs viennent et rendent beaucoup plus facile.

Au lieu de mettre sur pied des URL et des requêtes HTTP, il suffit de faire des appels de fonction. Dans les coulisses, ces fonctions mettent ensemble les URL et les données et faire les appels d'API . Mais d'un autre côté, il fonctionne exactement comme tout autre code . Généralement, on utilise le SDK qui est disponible pour notre langage de programmation, et s'il n'y a pas un SDK disponible on utilise des API.

Dans le cas d' OpenStack, il y a des SDKs pour les langages les plus populaires. Cependant, avant de plonger dans le SDK, on trouve qu'il est préférable de commencer par étudier la documentation de l'API lui-même afin qu'on sache exactement ce que le SDK est en train de faire. Et de cette façon si le SDK manque quoi que ce soit. En outre, puisque nous parlons ici open source, si le SDK ne répond pas à vos attentes, vous pouvez créer un nouveau SDK complètement.

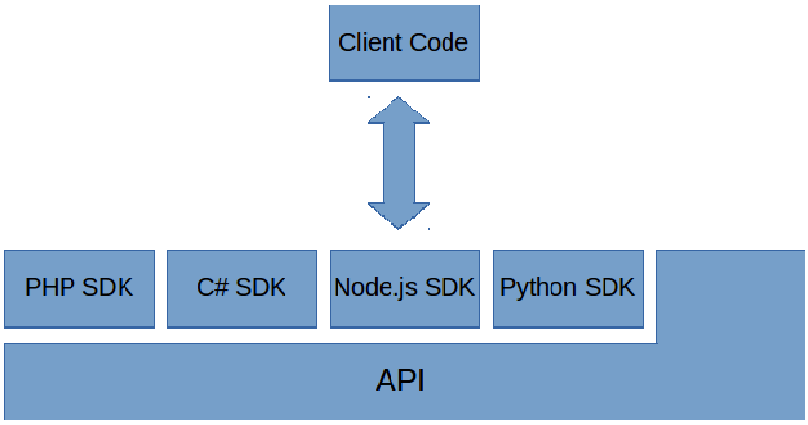


Figure3.13: la manière d'interagir avec les services OpenStack en utilisant SDKs

- **La liste des SDKs pour OpenStack avec tout les langages:**

En langage C: on a deux méthodes: Keystone and Swift libraries, c-keystoneclient .

En langage C++: on a SWIFT_CPP_SDK .

En langage Clojure: clj-openstack.

En langage Erlang: lfe-openstack.

En langage Go: on a les méthodes suivantes: Goose, Gophercloud, golang-client ,swift

En langage Java: on a les méthodes suivante: Apache jclouds, OpenStack4j ,OpenStack Java SDK , User Registration Service, Java OpenStack Storage aka JOSS

En langage Android: OpenStackIntegration

En langage JavaScript: jstack, js-openclient **,**node-openstack-wrapper

En langage .NET: OpenStack-SDK-DotNet ,OpenStack.NET, MarconiClient.NET

En langage Node.js: pkgcloud

En langage Perl: Net::OpenStack::Compute

En langage PHP: php-opencloud/openstack, rackspace/phpopencloud, ZendService_OpenStack , OpenStack-SDK-PHP

En langage Python:

- Les **OpenStackClients** sont les liaisons Python natives pour les API OpenStack . Ils sont utilisés pour mettre en œuvre les interfaces de ligne de commande (qui sont fournis avec la bibliothèque).
- Le **SDK-Development/PythonOpenStackSDK** est une solution proposée à offrir un SDK qui fournit un point d'entrée pour les consommateurs unique et une base à partir de laquelle d'autres outils peuvent être construits sur , comme les interfaces de ligne de commande .
- **pyrax** devrait fonctionner avec la plupart des déploiements de cloud basés OpenStack , si elle vise spécifiquement le Cloud public Rackspace .
- **Apache Libcloud** est une bibliothèque Python standard qui fait abstraction des différences entre plusieurs API de fournisseur de Cloud .
- **OpenStack Shade** ombre est une bibliothèque de client simple pour faire fonctionner OpenStack nuages.

En langage Ruby: on a les méthodes : Aviator, fog**,** swift_client, nightcrawler_swift, swift-storage.

En langage Yaml :Warm

Conclusion:

Ce chapitre, a clarifié la solution Infrastructure as a service celle adapté a ce projet en présentant OpenStack, son architecture d'installation ainsi que ses différents composants en présentant leurs architectures ainsi l'objectif d'utiliser ces APIs .

Chapitre IV : Analyse des besoins et Conceptions

IV.1 Introduction:

Ce chapitre déclenchera la phase d'analyse les besoins et de conception du projet. En effet, une première section du chapitre sera consacrée pour la spécification et l'élaboration des différents cas d'utilisation. Donc, elle sera orientée à énoncer les différents besoins fonctionnels auxquels devrait répondre le portail web à réaliser, ainsi que les besoins non fonctionnels qu'il devrait respecter.

La deuxième section de ce chapitre sera pour la partie conception. Dans la quelle le projet entame la phase de conception qui permettra de décrire de manière détaillée l'architecture du portail afin de faciliter sa réalisation.

Analyse des besoins

IV.2 Identification de besoins:

Notre portail s'adresse essentiellement à deux types d'utilisateurs : l'administrateur et les membres des projets. Cette première partie, sera pour énoncer et analyser les différents besoins fonctionnels et non fonctionnels du nuage.

IV.2.1 Besoins fonctionnels:

Cette partie, est pour détailler l'ensemble des fonctionnalités que le mini nuage, à travers un portail, doit offrir aux utilisateurs. En effet, le système à réaliser doit répondre aux besoins fonctionnels suivants.

- **Gestion d'images:**

  On parle d'images disques stockées par le service *Glance*.

  L'utilisateur pourrait consulter la liste des images autorisées pour les projets, les éditer. Aussi il sera possible de lancer de nouvelles instances de cette image, créer une nouvelle ou supprimer une existante.

- **Gestion d'instances:**

  Une instance est une machine virtuelle en cours d'exécution ou dans un état connu comme «suspendue» qui peut être utilisé comme un serveur matériel. L'utilisateur pourrait consulter la liste d'instances de machines virtuelles actuelles plus quelques informations globales comme le projet auquel elles appartiennent, le serveur hôte, l'adresse IP, la taille, le statut et les actions en cours. Il aurait aussi les possibilités d'éditer, mettre fin, mettre en pause, redémarrer ou supprimer une instance. Aussi Il pourrait se connecter à la console *VNC* de l'instance ou créer une nouvelle.

- **Gestion de volumes:**

  Le portail permettrait à l'utilisateur de consulter la liste des volumes disques virtuels existants, la création d'un nouveau volume et la modification d'un ancien.

- **Gestion de *Flavors*:**

  Un *Flavor* est une configuration de matériel disponible dans un serveur. Chaque *Flavor* possède une combinaison unique d'espace disque et la capacité de mémoire. L'utilisateur pourrait consulter la liste des types d'instances disponibles, leurs spécifications en nombre de *CPUs*, mémoire, espace disque et créer de nouvelles définitions d'instance.

- **Gestion de projets:**

  Un projet est un groupement logique des utilisateurs au sein de Nova, utilisé pour définir les limites des ressources pour ce projet et l'accès aux images des machines

  virtuelles .Il seraient possible de consulter les projets existants et leur statut et de créer de nouveaux projets.

- **Gestion d'utilisateurs:**

  L'utilisateur aurait la possibilité de consulter la liste des utilisateurs enregistrés, avec la possibilité d'ajouter ou d'éditer les détails mais pas d'ajouter un utilisateur à plusieurs projets.

- **Gestion de la sécurité et de l'accès:**

  L'utilisateur pourrait consulter les adresses IP disponibles pour connecter les instances au réseau public avec la possibilité de création, les groupes de règles de pare-feu et leur interface d'édition et enfin la liste des clés SSH avec l'import ou la création de certificat.

IV.2.2 Besoins non fonctionnels:

- **Simplicité d'un service à la demande:**

  Un utilisateur peut de manière unilatérale, immédiatement et généralement sans intervention humaine, avoir à sa disposition les ressources informatiques dont il a besoin(temps de calcul de serveurs, capacité de stockage, etc.)

- **Extrême flexibilité:**

  Les ressources mises à disposition ont une capacité d'adaptation forte et rapide à une demande d'évolution, généralement de manière transparente pour l'utilisateur.

- **Accès « léger »:**

  L'accès aux ressources ne nécessite pas d'équipement ou de logiciel propriétaire. Ilse fait au travers d'applications facilement disponibles (parfois libres), généralement depuis un simple navigateur Internet.

- **Sûreté:**

  Un évènement indésirable ne devrait pas se produire pendant l'accès d'une machine virtuelle aux ressources informatiques.

- **Vivacité:**

  Une action souhaitée par une machine virtuelle arrivera nécessairement à être réalisée pour garantir la progression du programme.

IV.3Les diagrammes de cas d'utilisation:

Cette phase a pour objectif de décrire le comportement attendu de l'application. Pour cela l'utilisation du diagramme des cas d'utilisation qui représente un élément essentiel de la modélisation orientée objet assure des bonnes résultats . Elle permet de modéliser les fonctionnalités de l'application de point de vue besoins utilisateur. Elle sert aussi à définir et à modéliser le produit à développer.

IV.3.1 Les acteurs du système:

Les acteurs manipuleront notre application sont :

- L'administrateur : Il possèderait les droits administratifs qui lui permettrait de contrôler tout le nuage et lui permettrait d'accéder à l'interface d'administration sur le portail « Dashboard » ainsi qu'à tous les autres projets.
- Membre du projet du IIT « Membre »: C'est un membre d'un ou de plusieurs projets qui sont propres au IIT. Il n'aurait accès qu'à son (ses) projet(s). Par la suite les principaux cas d'utilisations qui assurent toutes les tâches exécutées par le système seront mis en place.

IV.3.2 Authentification:

Avant l'accès au nuage, l'utilisateur (administrateur, membre) doit s'authentifier avec la saisie de son login, et son mot de passe. Après vérification, si l'utilisateur est accepté, il aura accès au nuage et selon son rôle des projets et des fonctionnalités s'activeront sinon on aura un message d'erreur.

IV.3.3 Le cas d'utilisation de l'administrateur:

La figure 4.1 présente le diagramme de cas d'utilisation de l'administrateur du nuage.

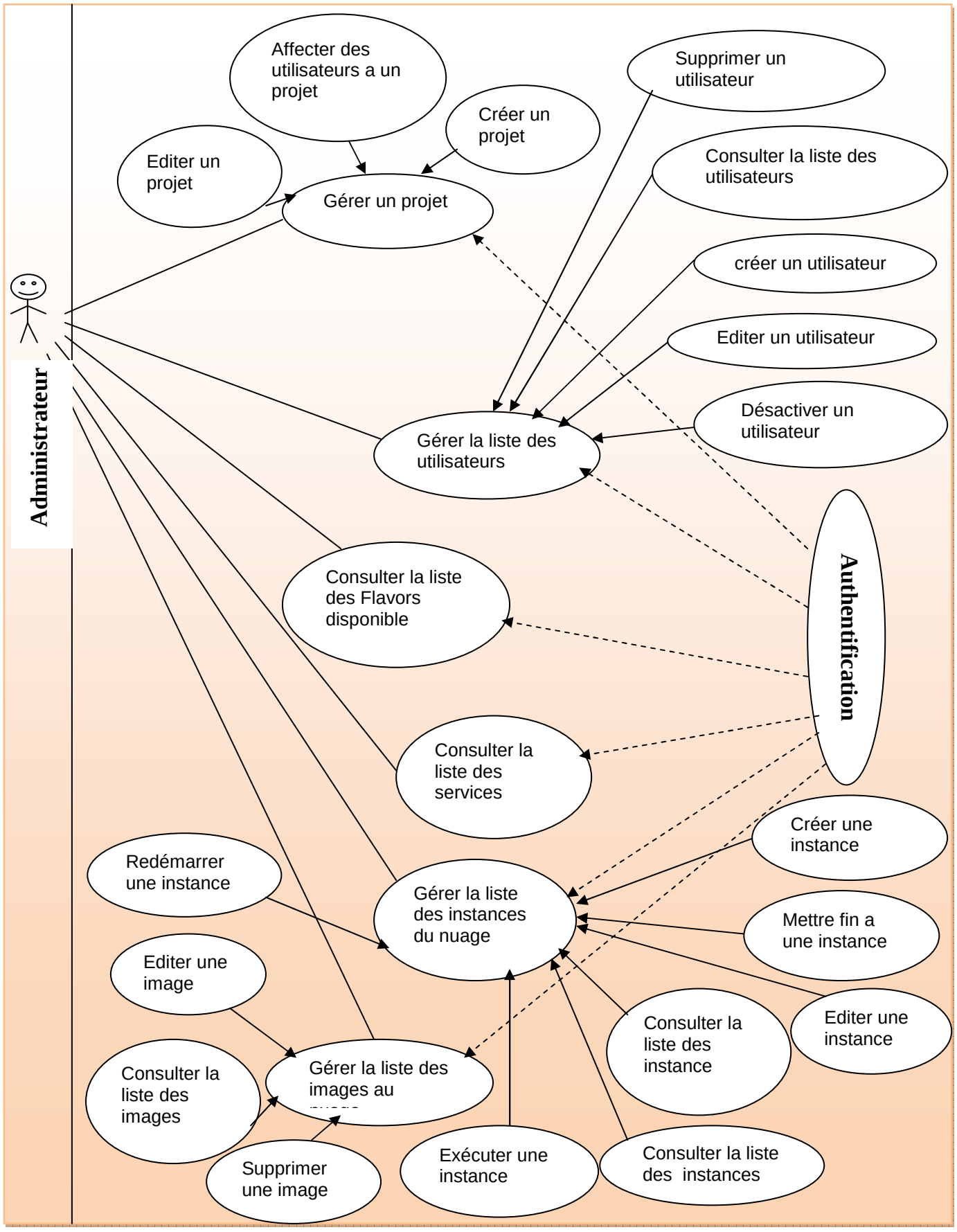


Figure4.1: Diagramme des cas d'utilisation pour un administrateur

Etant donné que l'administrateur a le droit de consulter et de gérer la totalité du nuage, il serait possible pour lui de :

Consulter l'état du nuage:

L'administrateur pourrait consulter l'usage des serveurs par projet, utilisation actuelle en nombre de CPU virtuels, RAM et Disques puis compteur en CPU et espace disque(GB) par heures.

Gérer les instances:

L'administrateur pourrait gérer les instances existantes sur le nuage, il aurait la possibilité de :

- Consulter la liste des instances existantes et leurs détails.
- Editer les détails d'une instance.
- Mettre fin à une instance.
- La suspendre.
- La redémarrer.
- La supprimer.

L'administrateur ne pourrait pas créer une nouvelle instance.

Consulter la liste des services et leurs détails:

Liste des services (Volume, Glance, Nova, Keystone..) activés, le serveur hôte et leurs statut (activé/désactivé).

Gérer les Flavors:

L'administrateur pourrait:

- Consulter la liste de Flavors actuellement disponibles qui pourraient être utilisés pour lancer une instance.
- Créer des Flavors personnalisées.
- Editer des Flavors.
- Supprimer des Flavors existants.

Gérer les images:

L'administrateur aurait la possibilité de:

- Consulter la liste des images disponibles.
- Editer les détails d'une image (nom, noyau ID, Ramdisk ID, architecture, format, public ou privé).
- Supprimer des images si elles ne sont plus nécessaires.

Gérer les projets:

L'administrateur aurait la possibilité de gérer les projets existants sur le nuage, ainsi il pourrait:

- Consulter la liste des projets disponibles (locataires) qui ont été créés, leurs détails et leurs utilisations.
- Créer de nouveaux projets.
- Affecter des utilisateurs à un projet.
- Modifier les détails d'un projet.
- Supprimer un projet.

Gérer les utilisateurs:

L'administrateur pourrait gérer les comptes utilisateurs existants sur le nuage, ainsi il pourrait:

- Consulter la liste des utilisateurs qui ont été créés et leurs détails.
- Créer de nouveaux utilisateurs.
- Désactiver / supprimer des utilisateurs existants.

Consulter les Quotas:

L'administrateur pourrait consulter les ressources des serveurs du nuage, mais il ne pourrait rien changer.

IV.3.4 Le cas d'utilisation d'un membre:

La figure 4.2 présente le diagramme de cas d'utilisation d'un membre d'un projet existant dans le nuage.

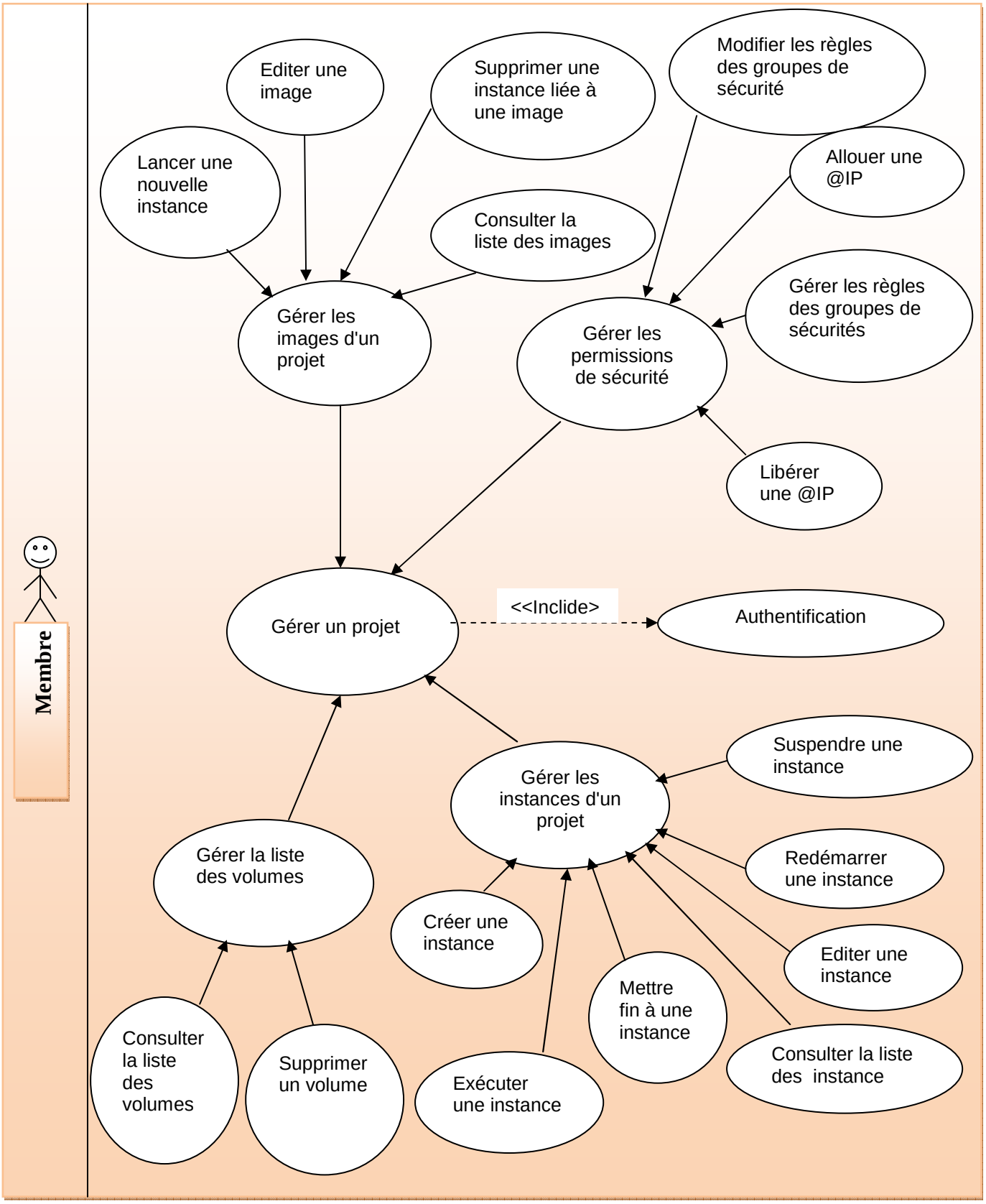


Figure4.2 : Diagramme des cas d'utilisation pour un membre d'un projet

Un membre ne pourrait consulter et gérer que les ressources des projets auxquels il appartient, ainsi il serait possible pour lui de:

- **Consulter les états de ces projets:**

  Un membre pourrait consulter l'usage des serveurs d'un projet sélectionné, utilisation actuelle en nombre de CPU virtuels, RAM et Disques puis compteur en CPU etespace disque(GB) par heures.

- **Gérer les instances et les volumes:**

  Un membre pourrait consulter la liste des instances existantes et aurait la possibilité de les éditer, de créer ou de modifier des volumes disques virtuels.

- **Gérer les images et leurs instances:**

  Un membre aurait la possibilité de consulter la liste des images autorisées pour le projet et lancer de nouvelles instances.

- **Gérer la sécurité et l'accès:**

  Un membre pourrait consulter la liste des adresses IP disponibles pour connecter les instances au réseau public avec la possibilité de création des groupes de règles et de Pare-feu. Il aurait aussi la possibilité de consulter la liste des clés SSH et de créer de certificat.

Conception

IV.4. Diagrammes de séquences système:

Avec les diagrammes de séquences système, l'UML fournit un moyen graphique pour représenter les interactions entre un acteur et le système au cours de l'exécution du cas d'utilisation. Ce paragraphe, sera consacré pour présenter quelques diagrammes de séquences les plus significatifs.

4.1Diagramme de séquences pour le scénario d'authentification:

Acteur : Un membre.

Pré conditions: Le membre, doit avoir un compte valide dans le système. Déclencheur: Un membre veut consulter l'état du projet auquel il appartient.

Description: Ce cas d'utilisation permet à un membre du projet de s'identifier pouraccéder au nuage à travers le portail.

Scénario principal:

- Un membre accède au portail.
- Une interface d'authentification s'affiche.
- Le membre entre ses données (login, mot de passe) et tape le bouton « valider ».
- Les différents services propres au nuage s'affichent Scénario alternatif:

- ✓ Les données saisies sont erronées.
- ✓ Un message d'erreur s'affiche.

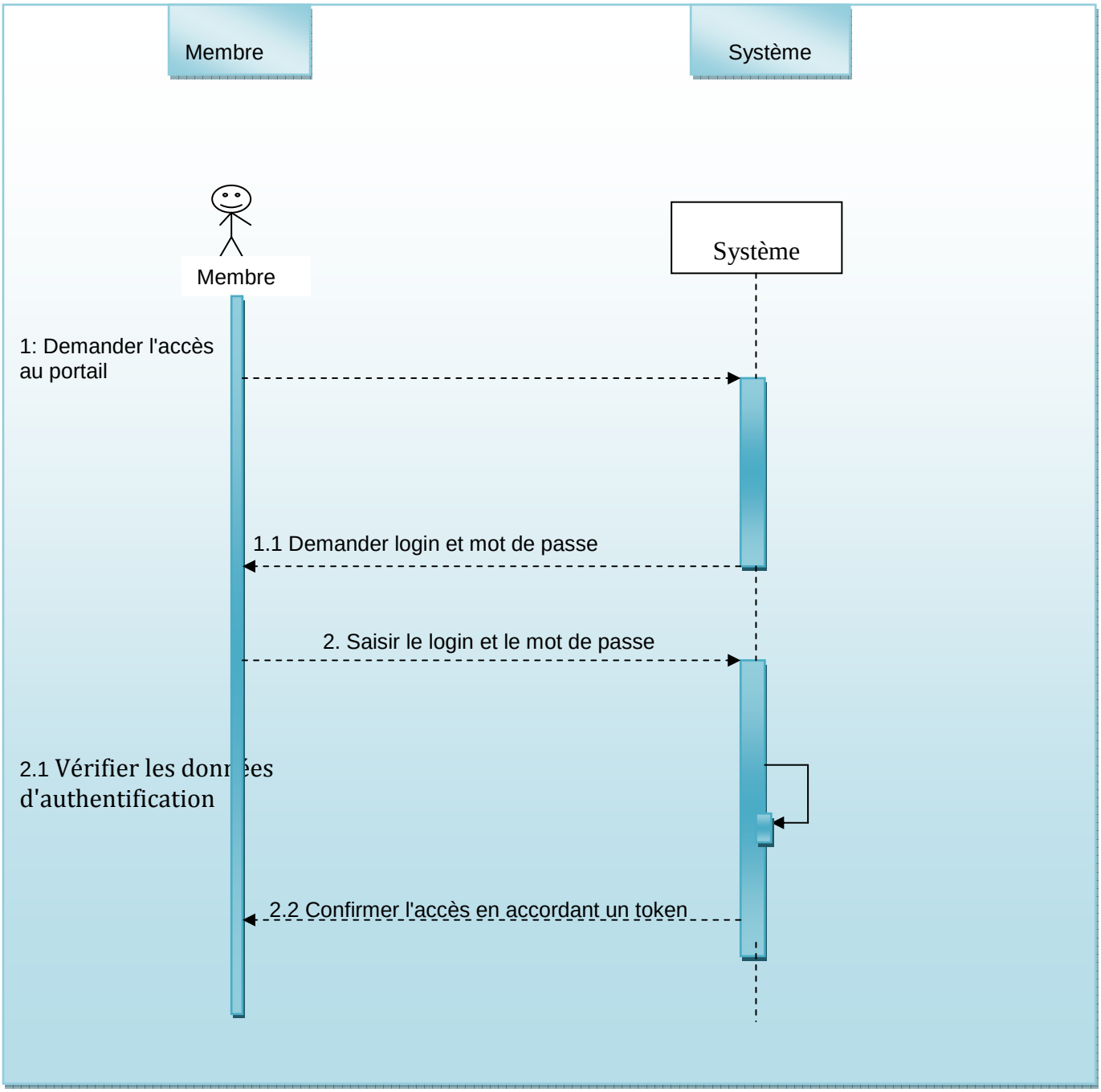


Figure 4.3: Diagramme de séquences pour le scenario d'authentification

4.2 Diagramme de séquences pour le scenario de création d'un nouveau projet par l'administrateur:

Acteur: Administrateur.

Pré conditions: L'administrateur a passé l'étape d'authentification avec succès.

Déclencheur: la IIT veut créer un nouveau projet.

Description: Ce cas d'utilisation permet à l'administrateur de créer un nouveau projet dans le nuage et l'accorder aux utilisateurs.

Scénario principal :

- Connecter au nuage en tant qu'administrateur.
- Sélectionner le lien Projets dans le menu.
- Choisir de créer un nouveau projet en cliquant sur un bouton « créer ».
- Remplir tous les champs et affecter des membres au projet.
- Valider l'opération.
- Le nouveau projet est ajouté à la liste des projets

Scénario alternatif:

- Des champs obligatoires ne sont pas été remplis.
- Un message d'erreur s'affiche.

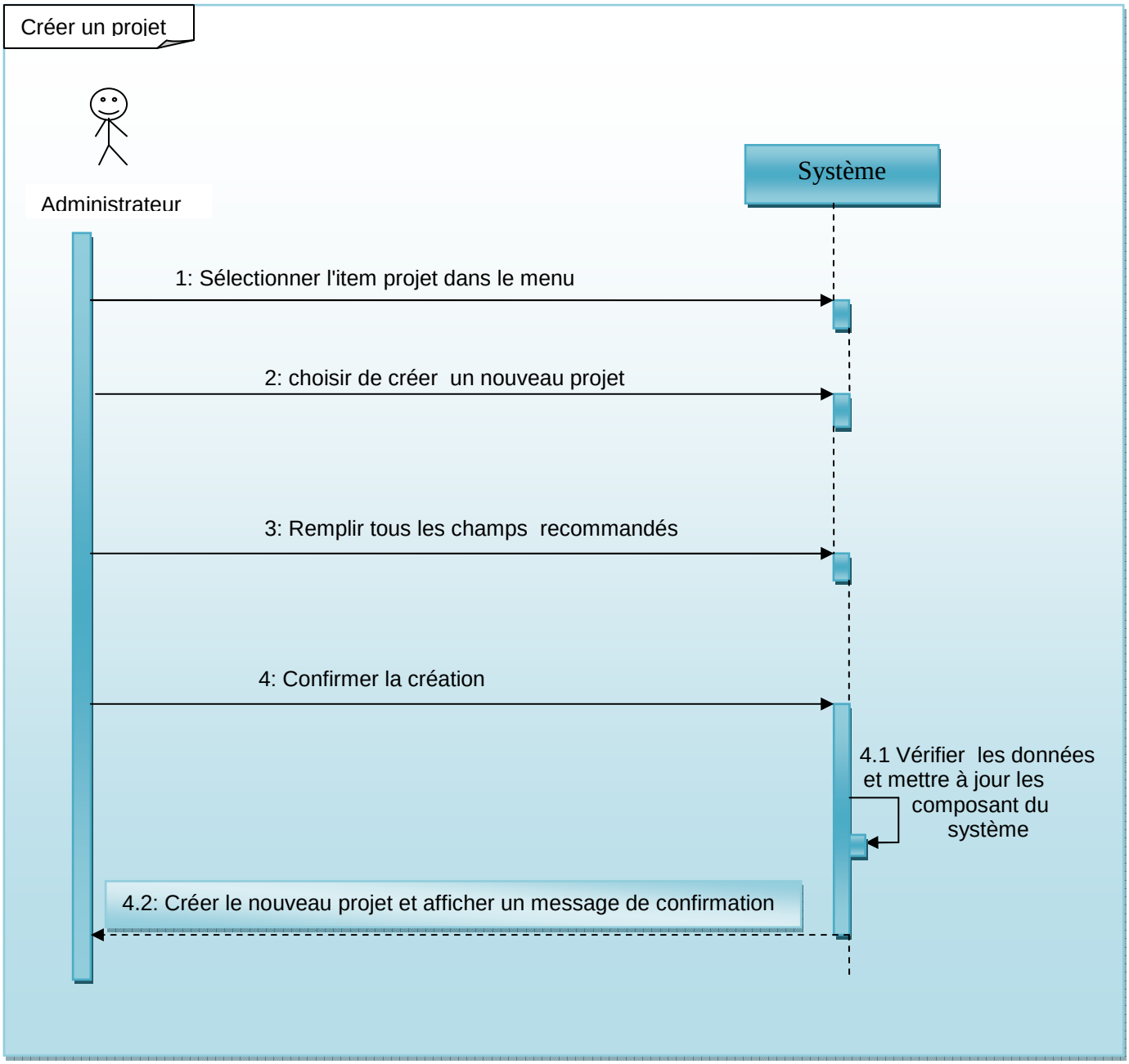


Figure 4.4 : Diagramme de séquences pour le scenario de création un nouveau projetpar l'administrateur

4.3Diagramme de séquences pour le scenario de création d'une instance d'une machine virtuelle:

Acteur : Membre.

Pré conditions: Le membre a passé l'étape d'authentification avec succès.

Déclencheur: Un membre d'un projet veut créer une nouvelle instance d'une machine virtuelle déjà existante dans le nuage.

Description: Ce cas d'utilisation permet à un membre d'un projet de créer une nouvelle instance d'une machine virtuelle du nuage.

Scénario principal:

- Connecter au portail en tant que membre.
- Choisir le projet dans lequel la machine existe.
- Sélectionner le lien « Images » dans le menu.
- Choisir la machine cible et choisir de créer une nouvelle instance.
- Remplir tous les champs.
- Valider l'opération.
- Une nouvelle instance est ajoutée à la liste des instances.

Scénario alternatif:

- Des champs obligatoires ne sont pas été remplis.
- Un message d'erreur s'affiche.

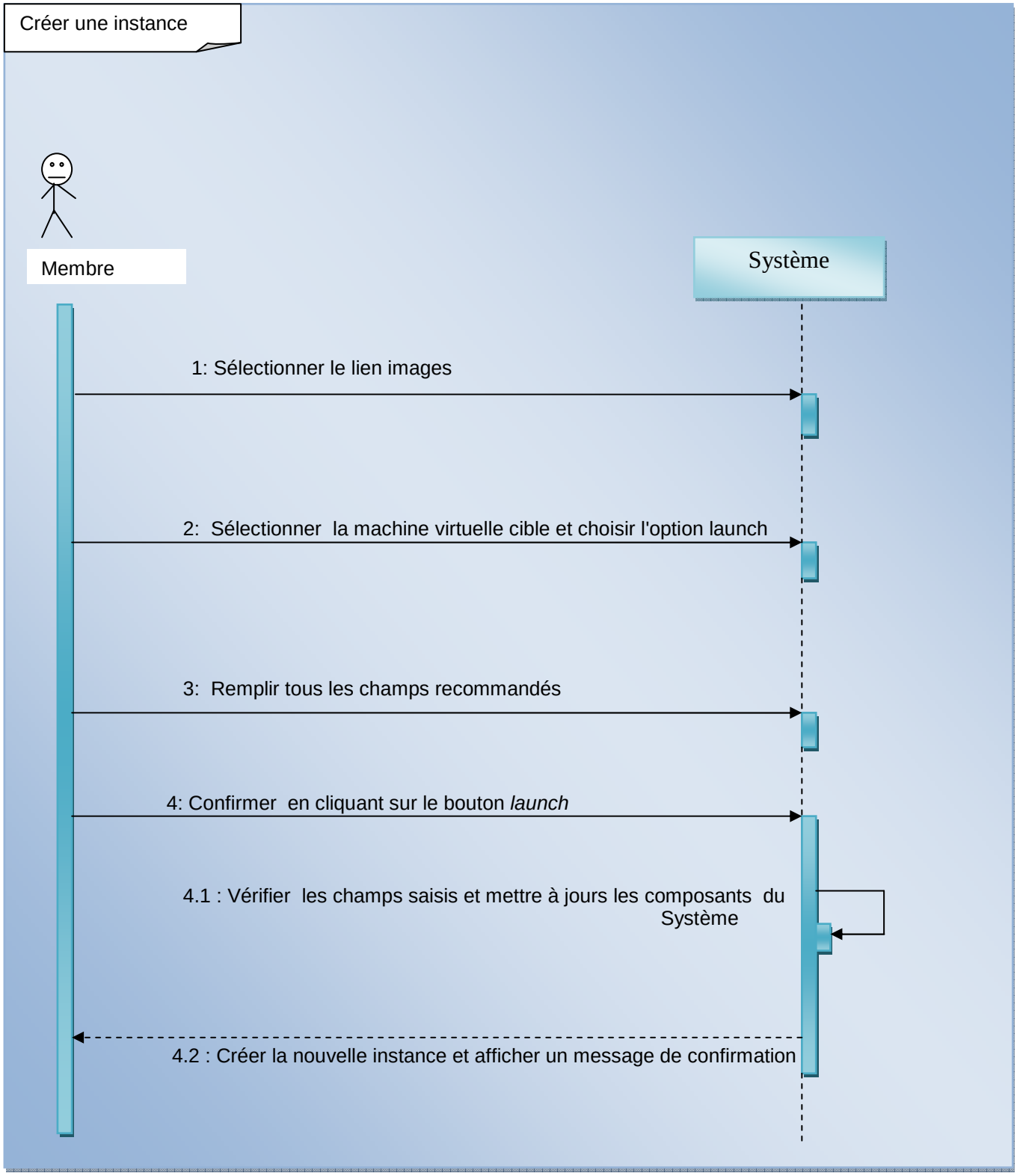


Figure 4.5 : Diagramme de séquences pour le scenario de création d'une instance d'une machine virtuelle.

Conclusion:

A travers ce chapitre, l'exposition des besoins tant fonctionnels que non fonctionnels du projet est faite. De plus, la présentation des diagrammes de cas d'utilisation et des diagrammes des séquences en définissant les différents acteurs et les détailles ses différents cas d'utilisation sont mises en place. Ainsi, la présentation de l'architecture générale de notre système ainsi que quelques les diagrammes de séquences détaillées.

Chapitre V : Réalisation et implémentation

V.1 Introduction :

Ce chapitre Présente en premier lieu, l'environnement matériel et logiciel du travail, le travail effectué en se basant sur les vraies interfaces configurées et implémentées pour ce Portail.

V.2 Etape de réalisation et d'implémentation:

Cette partie expose les différentes phases de réalisation illustrées par la figure V.1 En effet, notre travail est composé de trois étapes:

- Planification: Ecrire le scénario de déploiement, finaliser les choix d'architectures, et s'assurer que le matériel requis soit disponible.
- Déploiement: Installer les composants d'*OpenStack*, et enfin les configurer.
- Utilisation et test: Utiliser *OpenStack* afin d'accueillir les utilisateurs finaux. Ces étapes seront détaillées dans la suite.

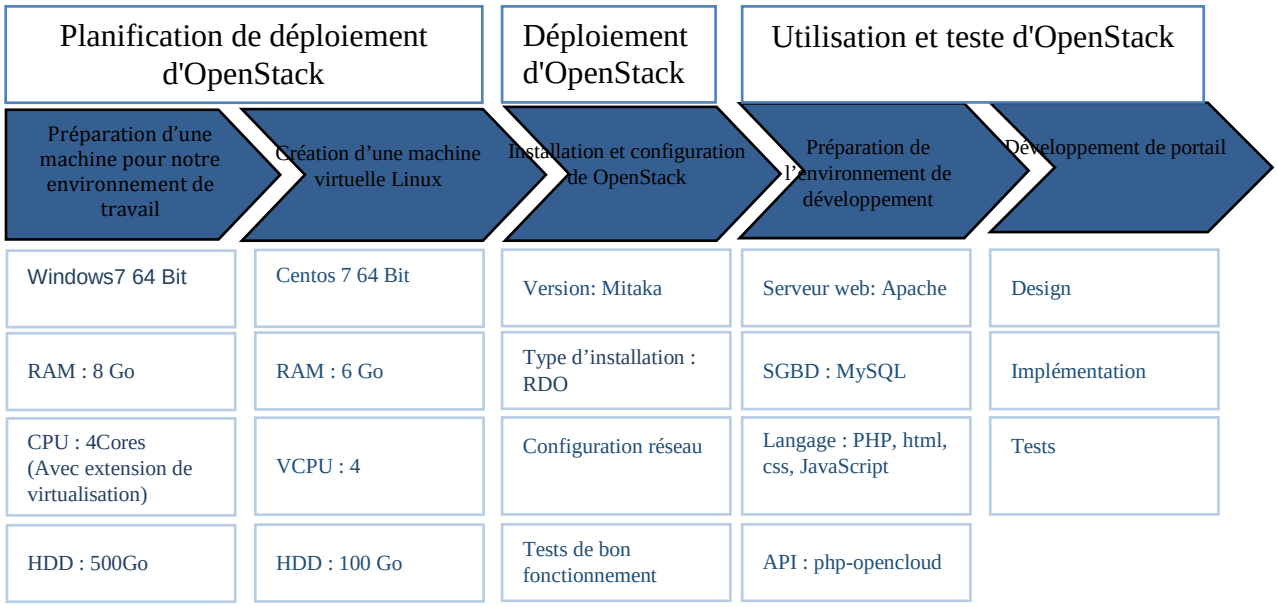


Figure 5.1: Etape de réalisation et d'implémentation

V.2.1 Planification du déploiement d'*OpenStack*:

Il existe de nombreuses méthodes pour le déploiement d'*OpenStack*.

• Nœud unique: un seul serveur exécute tous les services nova et également conduit toutes les instances virtuelles.

• Deux nœuds: Un nœud de contrôleur nuage exécute les services nova à l'exception de *Nova-Compute*, et un nœud de calcul où fonctionne *nova-compute* .

• Plusieurs nœuds: Un minimum de quatre nœuds est le meilleur pour l'exécution de plusieurs instances virtuelles qui nécessitent beaucoup de puissance de traitement.

Notre choix se fait avec un Nœud unique,-à cause du manque de ressources matérielles, mais reste un bon choix car notre but est de comprendre le coté développement de notre étude.

V.2.2 Environnement du travail:

Cette section présentera les plateformes matérielle et logicielle qui ont servi pour la réalisation de ce travail.

1. Environnement matériel:

Les caractéristiques de l'ordinateur portable (**Architecture matérielle, Matériels utilisés**) utilisé durant toute la période du stage de fin d'étude, et pendant toutes les phases de configuration et de développement, sont dans le tableau ci-dessous :

Pc Portable	HP
Mémoire RAM	8 Go (7 Go utilisable)
Disque dur	500 Go
Processeur	Intel R Core ™ i5-2430M CPU @2.40 GHz *2
Système d'exploitation	Centos7 Server 64 bit

Table 5.1: Configuration matérielle

2. Technologies utilisées:

2.1 Shell : sont des commandes sont rédigées en terminal.

2.2 KVM : (*Kernel-based Virtual Machine*) module de noyau linux pour la prise en charge de la virtualisation.

2.3 libviert : Une API pour interagir avec KVM pour la gestion des machines virtuelles.

2.4 php-opencloud: Framework qui permet le développement d'un Cloud privé (IaaS).

2.5 Composer : Composer est un outil pour gérer les dépendances en PHP. Les dépendances dans un projet , ce sont toutes les bibliothèques dont votre projet dépend pour fonctionner. dans notre cas on la utiliser pour le téléchargement et la compilation de php-opencloud.

2.6 SGBD MariaDb : MariaDB est un Système de Gestion de Base de Données (SGBD) disponible sous licence GPL. Ce système est un fork de MySQL, ce qui signifie que c'est un nouveau logiciel créé à partir du code source de MySQL.

2.7 Serveur Web Apache : Le logiciel libre **Apache HTTP Server** (**Apache**) est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du World Wide Web.

2.8 Langage de programmation php [8]:

PHP: HypertextPreprocessor, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

2.9 Présentation de données (interfaces graphiques):

- **Html :** L'HyperText Markup Langage, généralement abrégé **HTML**, est le format_de données conçu pour représenter les pages web. C'est un langage de balisage permettant d'écrire de l'hypertexte. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques. Il est souvent utilisé conjointement avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade). HTML est initialement dérivé du Standard Generalized Markup Language (SGML).
- **Javascript :** est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés.
- **Css :** (Cascading Style Sheets), C'est lui qui vous permet de choisir la couleur de votre texte. et qui vous permet de sélectionner la police utilisée sur votre site. et encore qui permet de définir la taille du texte, les bordures, le fond…

  Et aussi, c'est lui qui permet de faire la mise en page de votre site.

2.10 Format d'échange de données

- **JSON :** (JavaScript Object Notation - Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est facile à lire ou à écrire . Il est aisément analysable ou générable par des machines. Il est basé sur un sous-ensemble du langage de

[8] **PHP: https://fr.wikipedia.org/wiki/PHP**

programmation JavaScript (JavaScript Programming Langage, Standard ECMA-262 3rd Edition-Décembre 1999).

JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C, Ces propriétés font de JSON un langage d'échange de données idéal.

Un document JSON ne comprend que deux types d'éléments structurels :

- des ensembles de paires nom / valeur
- des listes ordonnées de valeurs.

- **Comment JSON va être utiliser dans notre application ?:**

Lorsque l'application s'exécute, elle se connectera au script PHP. Le script PHP va récupérer les données depuis la base de données MySQL. Ensuite les données seront encodées au format JSON et envoyées au système Openstack. Ensuite, l'application va obtenir ces données codées. Elle les analysera et les affichera sur serveur wamp où on a créer notre application . Le schéma de la figure5.3 illustre bien la façon d'échanger les données entre le serveur wamp de l'application et la partie de centos7 serveur où on a installé Openstack .

2.11 Linux :

Linux est un système d'exploitation entièrement libre, qui veut dire Le code source du noyau de Linux est accessible gratuitement, ce qui fait que ce système peut être compilé sur d'autres plates-formes que le PC.

2.12 VMWar Workstation *:*

VMWare Workstation est un logiciel à destination des professionnels. tout comme VMWare Player, il permet de créer des machines virtuelles et de les exécuter en même temps au-dessus d'un système d'exploitation hôte. Il propose toutefois quelques fonctionnalités plus poussées dont l'utilisateur final a rarement besoin, mais fort utiles aux professionnels de gestion de réseaux informatiques, on la utiliser pour l'installation et configuration d'*OpenStack*.

3. Environnement logiciel:

3.1. La plateforme IaaS *OpenStack*:

OpenStack joue le rôle d'une couche de management de Cloud qui assure la communication entre la couche physique ou se trouve des serveurs physiques occupés par des hyperviseurs différents (Vmware ESX, Citrix Xen, KVM, qemu...) et la couche applicative (Applications, utilisateurs, administrateurs…).

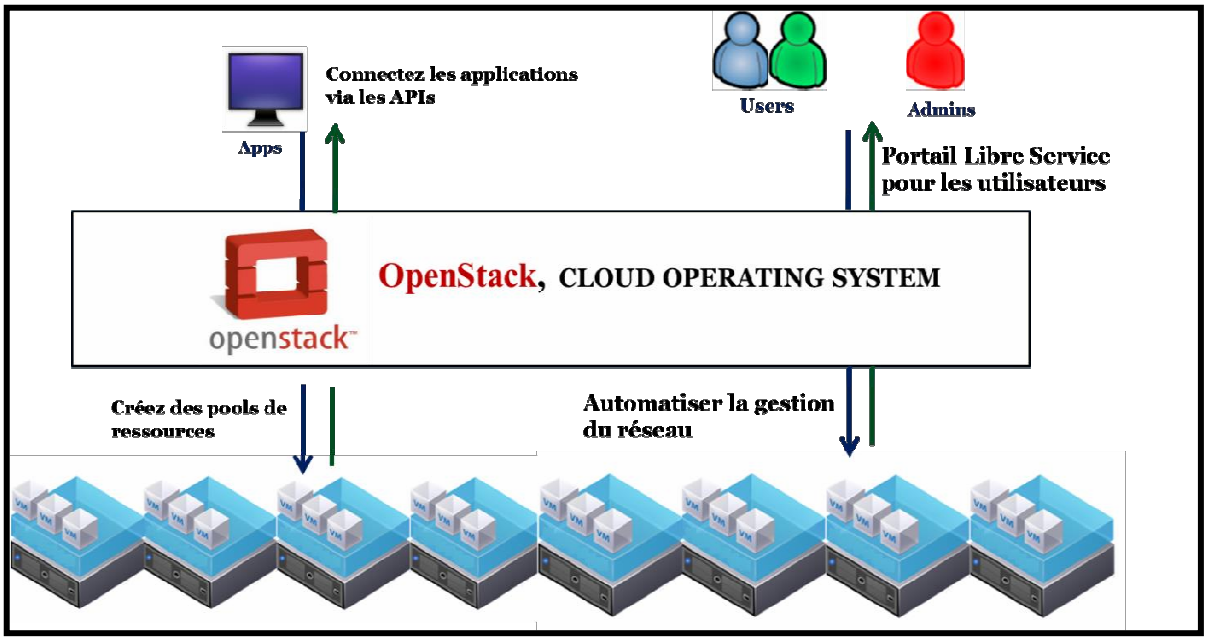


Figure5.2 : Le rôle d'*OpenStack*

3.2. Présentation de OpenStack (rappel):

OpenStack est un ensemble de logiciels libre (open source) permettant de déployer des infrastructures de Cloud Computing (infrastructure en tant que service) développées en Python et permettant de déployer un Cloud de type **IaaS** avec des fonctionnalités similaires aux offres bien connues du marché comme celle de Amazon .

Il possède une architecture modulaire composée de plusieurs projets adressant des fonctionnalités complémentaires permettant la construction de Cloud privé et public de type IaaS et de contrôler de grands bassins de calcul, de stockage, et les ressources de réseautage à travers un centre de données , tout géré par un tableau de bord qui donne aux administrateurs de contrôler tout en permettant à leurs utilisateurs à des ressources d'approvisionnement par le biais d'une interface web .

3.2 Wamp (apache/MySql/php):

WampServer est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHP MyAdmin pour gérer plus facilement vos bases de données.

V.2.3.Préparation de l'environnement de travail pour utiliser l'API *OpenStack*:

V.2.3.1. Architecture matériel et applicative :

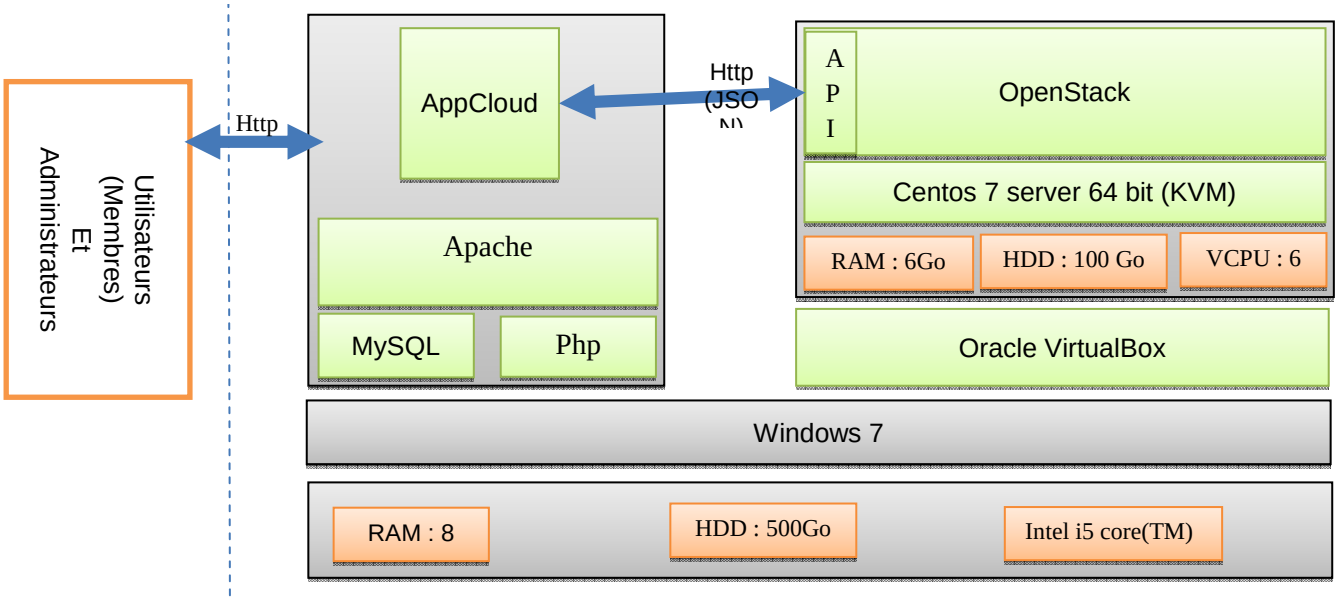


Figure 5.3: L'architecture matériel et applicative utilisé pour notre projet

V.2.3.2 Installation et configuration de *OpenStack*:

❖ **Préparation du système Linux pour RDO Mitaka:**

C'est l'étape la plus importante, et si elle est suivie correctement, vous ne devriez pas courir dans tous les problèmes / erreurs lors de l'installation de Mitaka. Les exigences matérielles de base pour le système linux pour exécuter RDO sont:

- 4 Go de mémoire.
- Processeurs avec la virtualisation matérielle activée.
- Un adaptateur réseau.

Une fois que nous avons rempli au-dessus des exigences matérielles mentionnées et fraîche CentOS7 Serveur a été installé sur le système (nous pouvons utiliser la machine virtuelle à cet effet aussi), aller de l'avant et faire en sorte que IP statique est configurée sur notre système. autrement désactiver DHCP et configurer votre système pour utiliser l'adresse IP statique au lieu. Une fois IP statique a été configuré, désactiver le service *NetworkManager* pour ne pas avoir des conflit avec service neutron de *OpenStack*.

```
[root@dhcppc1 ~]# systemctl stop NetworkManager.service
[root@dhcppc1 ~]# systemctl disable NetworkManager.service
[root@dhcppc1 ~]# systemctl restart network
```

on désactive temporairement Selinux pour éviter les erreurs lors de l'installation et arrêter firewall (ou bien le configurer pour permettre à tous les ports requis, etc.).

```
[root@dhcppc1 ~]# setenforce 0
[root@dhcppc1 ~]# systemctl stop firewalld
[root@dhcppc1 ~]# systemctl disable firewalld
```

maintenant, on va attribuer une adresse IP fixe à la machine host après que notre système à a une adresse local puisque c'est un serveur, en utilisant cette commande :

```
[root@dhcppc1 ~]# vi /etc/hosts
```

voila le fichier dans lequel a effectué les modifications:

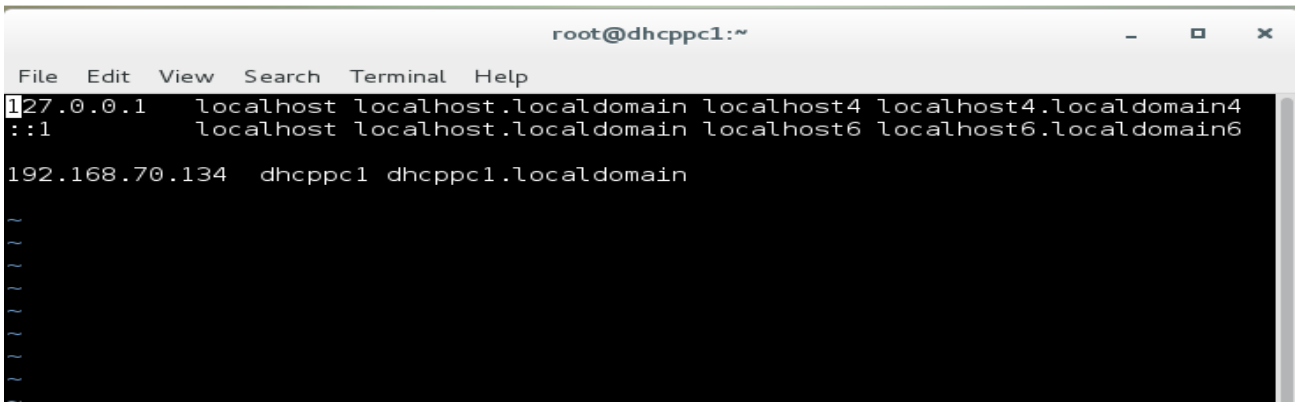


Figure 5.4:configuration manuelle de l'adresse host

- puis, on a besoin de modifier notre fichier configuration de réseau qui se trouve dans ce répertoire /etc/sysconfig/network-scripts/.

```
[root@dhcppc1 ~]# cd /etc/sysconfig/network-scripts/
[root@dhcppc1 network-scripts]# ls
ifcfg-br-ex         ifdown-post     ifup-ib      ifup-Team
ifcfg-eno16777736   ifdown-ppp      ifup-ippp    ifup-TeamPort
ifcfg-lo            ifdown-routes   ifup-ipv6    ifup-tunnel
```

- la carte réseau utilisée est:ifcfg-eno1677773

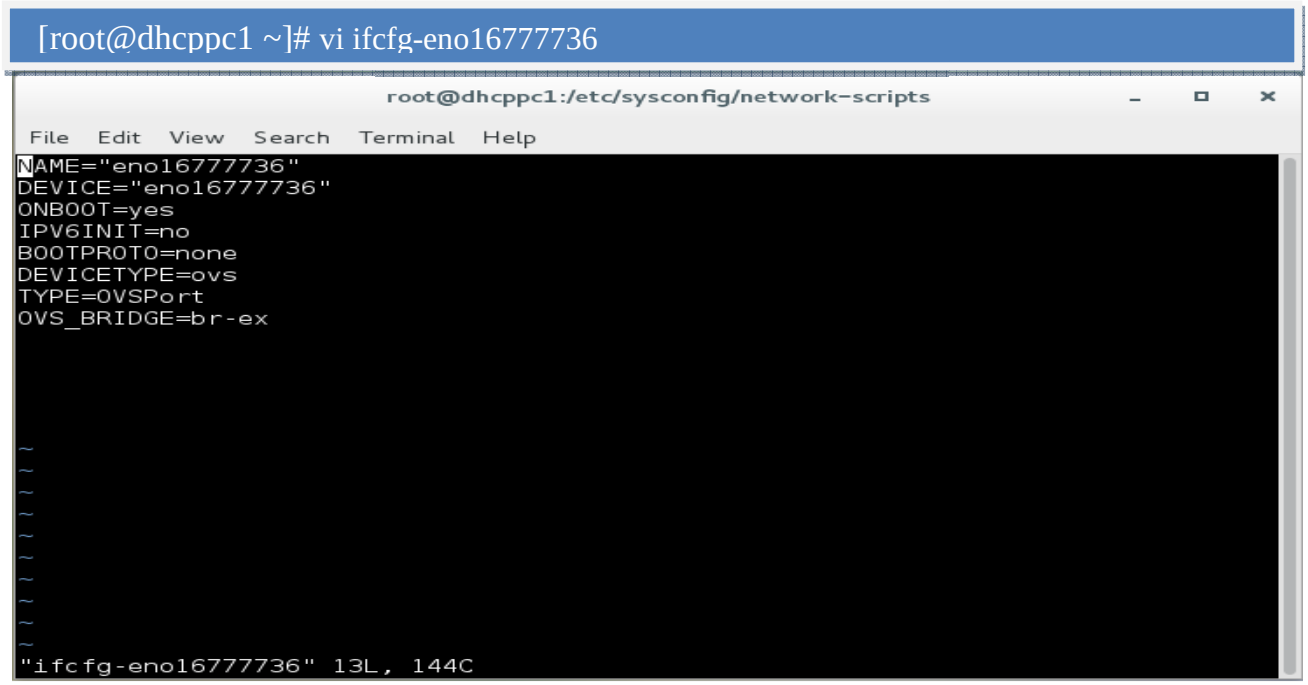


Figure 5.5:fichier de configuration de la carte réseau ifcfg-eno1677773

Ensuite, on a besoin de modifier l'interface du pont que l'installateur RDO mis en place pour nous. Le fichier, qui devrait être appelé / etc / sysconfig / network-scripts / ifcfg-br-ex, devrait être modifié pour correspondre à l'exemple ci-dessous. Assurez-vous de remplacer vos propres valeurs pour IPADDR, GATEWAY, DNS1 et NETMASK.
la carte réseau qu'on va travailler au dessue c'est:

Figure 5.6:fichier de configuration de la carte ifcfg-br-ex

Tous les pré-requis sont remplies maintenant. Nous allons aller de l'avant avec le processus d'installation réelle de Mitaka tout de suite.

Installation Mitaka:

on assure que notre / etc / environnement est peuplée en mettent c'est deux commande dans ce dernier fichier:

LANG=en_US.utf-8

LC_ALL=en_US.utf-8

Sur la toute première étape, installer RPM(Red Hat Package Manager) pour RDO référentiel en utilisant la commande YUM suivante.

[root@dhcppc1 ~]# yum install https://rdoproject.org/repos/rdo-release.rpm

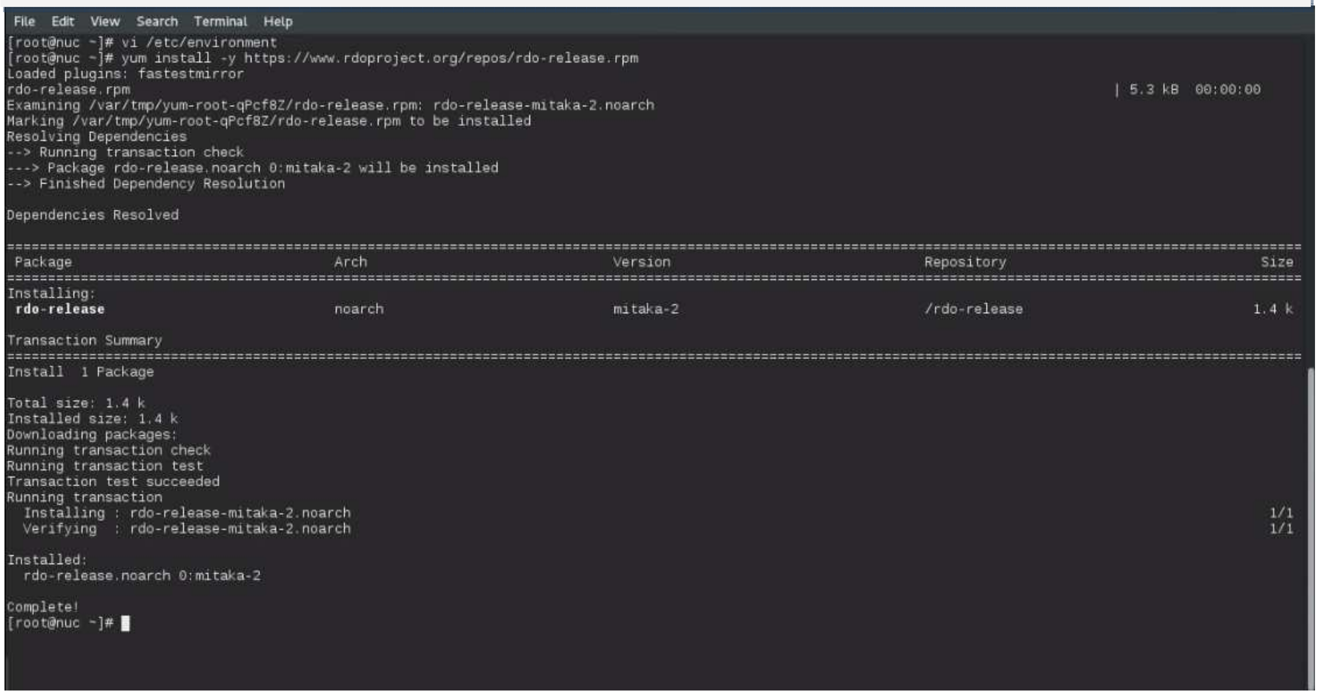


Figure 5.7:installation de système de gestion de paquets de logiciels pour RDO référentiel

Exécutez cette commande pour installer RPM référentiel de CentOS suivant.

[root@dhcppc1 ~]# yum install centos-release-OpenStack-mitaka

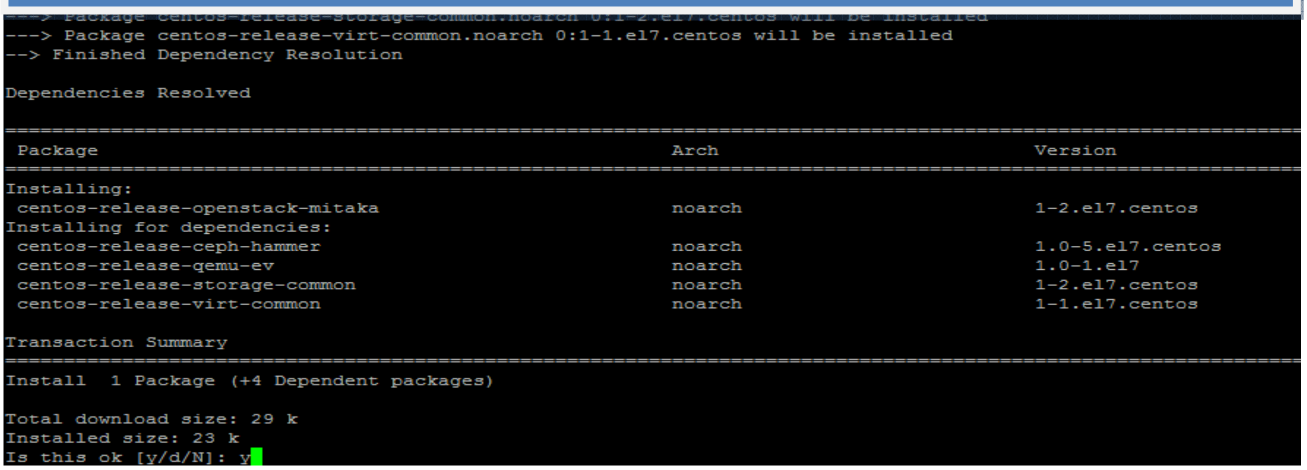


Figure 5.8:installation de système de gestion de paquets de logiciels utilisé sur Centos7

on exécute la commande suivante pour mettre à jour le gestionnaire de packager système pour tenir compte de ces derniers ajouts.

```
[root@dhcppc1 ~]#  yum  update
```

```
root@nuc:~
File  Edit  View  Search  Terminal  Help
================================================================================
Install           ( 2 Dependent packages)
Upgrade  1 Package

Total download size: 717 k
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/cache/yum/x86_64/7/openstack-mitaka/packages/mariadb-config-10.1.12-4.el7.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 764429e6: NOKEY
Public key for mariadb-config-10.1.12-4.el7.x86_64.rpm is not installed
(1/3): mariadb-config-10.1.12-4.el7.x86_64.rpm                  |  23 kB  00:00:00
(2/3): mariadb-common-10.1.12-4.el7.x86_64.rpm                  |  60 kB  00:00:00
(3/3): mariadb-libs-10.1.12-4.el7.x86_64.rpm                    | 634 kB  00:00:00
--------------------------------------------------------------------------------
Total                                           1.4 MB/s | 717 kB  00:00:00
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Cloud
Importing GPG key 0x764429E6:
 Userid     : "CentOS Cloud SIG (http://wiki.centos.org/SpecialInterestGroup/Cloud) <security@centos.org>"
 Fingerprint: 736a f511 6d9c 40e2 af6b 074b f9b9 fee7 7644 29e6
 Package    : rdo-release-mitaka-2.noarch (@/rdo-release)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-SIG-Cloud
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 1:mariadb-config-10.1.12-4.el7.x86_64                       1/4
  Installing : 1:mariadb-common-10.1.12-4.el7.x86_64                       2/4
  Updating   : 1:mariadb-libs-10.1.12-4.el7.x86_64                         3/4
  Cleanup    : 1:mariadb-libs-5.5.47-1.el7_2.x86_64                        4/4
  Verifying  : 1:mariadb-config-10.1.12-4.el7.x86_64                       1/4
  Verifying  : 1:mariadb-common-10.1.12-4.el7.x86_64                       2/4
  Verifying  : 1:mariadb-libs-10.1.12-4.el7.x86_64                         3/4
  Verifying  : 1:mariadb-libs-5.5.47-1.el7_2.x86_64                        4/4

Dependency Installed:
  mariadb-common.x86_64 1:10.1.12-4.el7                 mariadb-config.x86_64 1:10.1.12-4.el7

Updated:
  mariadb-libs.x86_64 1:10.1.12-4.el7

Complete!
```

Figure 5.9: Mettre à jour le gestionnaire de package de système

on est sur la scène la plus importante de l'installation maintenant, exécuter la commande pour installer l'utilitaire Packstack suivant.

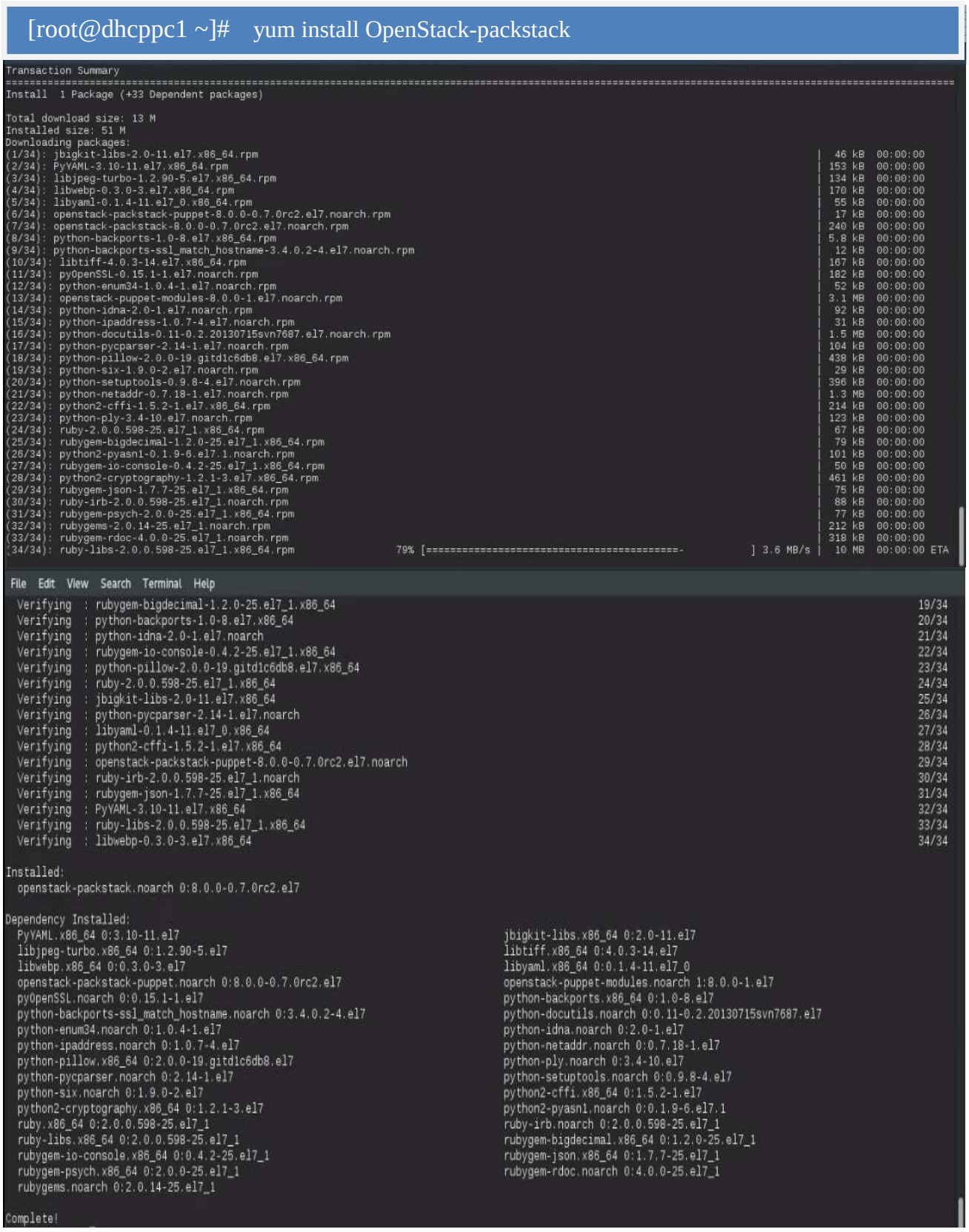


Figure 5.10:installation de l'utilitaire Packstack

Dès que l'installation de packstack est terminée, on exécute la commande suivante pour installer *OpenStack* Mitaka.

avant l'installation d'*OpenStack* Mitaka on exécute cette commande ,pour obtenir un fichier ou on peut choisir l'installation des services dont on a besoin.

```
[root@dhcppc1 ~]#  packstack--gen-answer-file=answer.txt
[root@dhcppc1 ~]#  vi answer.txt
```

```
root@dhcppc1:~
File  Edit  View  Search  Terminal  Help

# Specify 'y' to install MariaDB. ['y', 'n']
CONFIG_MARIADB_INSTALL=y

# Specify 'y' to install OpenStack Image Service (glance). ['y', 'n']
CONFIG_GLANCE_INSTALL=y

# Specify 'y' to install OpenStack Block Storage (cinder). ['y', 'n']
CONFIG_CINDER_INSTALL=n

# Specify 'y' to install OpenStack Shared File System (manila). ['y',
# 'n']
CONFIG_MANILA_INSTALL=n

# Specify 'y' to install OpenStack Compute (nova). ['y', 'n']
CONFIG_NOVA_INSTALL=y

# Specify 'y' to install OpenStack Networking (neutron); otherwise,
# Compute Networking (nova) will be used. ['y', 'n']
CONFIG_NEUTRON_INSTALL=y

# Specify 'y' to install OpenStack Dashboard (horizon). ['y', 'n']
CONFIG_HORIZON_INSTALL=y

# Specify 'y' to install OpenStack Object Storage (swift). ['y', 'n']
CONFIG_SWIFT_INSTALL=n

# Specify 'y' to install OpenStack Metering (ceilometer). ['y', 'n']
CONFIG_CEILOMETER_INSTALL=n
```

Figure 5.11:fichier des services d'*OpenStack*

Ensuite on exécute cette commande pour lancer l'installation :

```
[root@dhcppc1 ~]#  packstack --answer-file=answer.txt
```

```
Adding Provisioning Demo bridge manifest entries      [ DONE ]
Copying Puppet modules and manifests                  [ DONE ]
Applying 10.42.10.34_prescript.pp
10.42.10.34_prescript.pp:                             [ DONE ]
Applying 10.42.10.34_amqp.pp
Applying 10.42.10.34_mariadb.pp
10.42.10.34_amqp.pp:                                  [ DONE ]
10.42.10.34_mariadb.pp:                               [ DONE ]
Applying 10.42.10.34_keystone.pp
Applying 10.42.10.34_glance.pp
10.42.10.34_keystone.pp:                              [ DONE ]
10.42.10.34_glance.pp:                                [ DONE ]
Applying 10.42.10.34_api_nova.pp
10.42.10.34_api_nova.pp:                              [ DONE ]
Applying 10.42.10.34_nova.pp
10.42.10.34_nova.pp:                                  [ DONE ]
Applying 10.42.10.34_neutron.pp
10.42.10.34_neutron.pp:                               [ DONE ]
Applying 10.42.10.34_osclient.pp
Applying 10.42.10.34_horizon.pp
10.42.10.34_osclient.pp:                              [ DONE ]
10.42.10.34_horizon.pp:                               [ DONE ]
Applying 10.42.10.34_provision.pp
Applying 10.42.10.34_provision_glance
10.42.10.34_provision.pp:                             [ DONE ]
10.42.10.34_provision_glance:                         [ DONE ]
Applying 10.42.10.34_provision_bridge.pp
10.42.10.34_provision_bridge.pp:                      [ DONE ]
Applying Puppet manifests                             [ DONE ]
Finalizing                                            [ DONE ]

 **** Installation completed successfully ******

Additional information:
 * Time synchronization installation was skipped. Please note that unsynchronized time on server instances might be problem for some OpenStack components.
 * File /root/keystonerc_admin has been created on OpenStack client host 10.42.10.34. To use the command line tools you need to source the file.
 * To access the OpenStack Dashboard browse to http://10.42.10.34/dashboard .
Please, find your login credentials stored in the keystonerc_admin in your home directory.
 * The installation log file is available at: /var/tmp/packstack/20160412-213615-b68Zlu/openstack-setup.log
 * The generated manifests are available at: /var/tmp/packstack/20160412-213615-b68Zlu/manifests
[root@nuc ~]#
```

Figure 5.12: Exemple d'installation de quelque service d'*OpenStack* après l'exécution de la commande au-dessus

Voici ce qu'on devrait voir dans le cas d'une installation réussie comme est montré dans la figure précédente , sinon durant l'installation on corrige les erreurs apparait au fur et à mesure .

on est en mesure de localiser *OpenStack* login de tableau de bord à l'intérieur "/ root / keystonerc_admin" fichier comme indiqué dans la capture d'écran ci-dessus.
on exécute la commande ci-dessous pour voir fichier et administrateur login complets.

```
[root@dhcppc1 ~]#  vi / root / keystonerc_admin.
```

```
File  Edit  View  Search  Terminal  Help
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD=7073f21f801f4ca0
export OS_AUTH_URL=http://192.168.70.134:5000/v2.0
export PS1='[\u@\h \W(keystone_admin)]\$ '

export OS_TENANT_NAME=admin
export OS_REGION_NAME=RegionOne
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"~/keystonerc_admin" 8L, 239C
```

Finalement on parcoure URL *OpenStack* d'installation (http: //192.168.70.134/ tableau de bord) et après écran de connexion devrait vous accueillir.

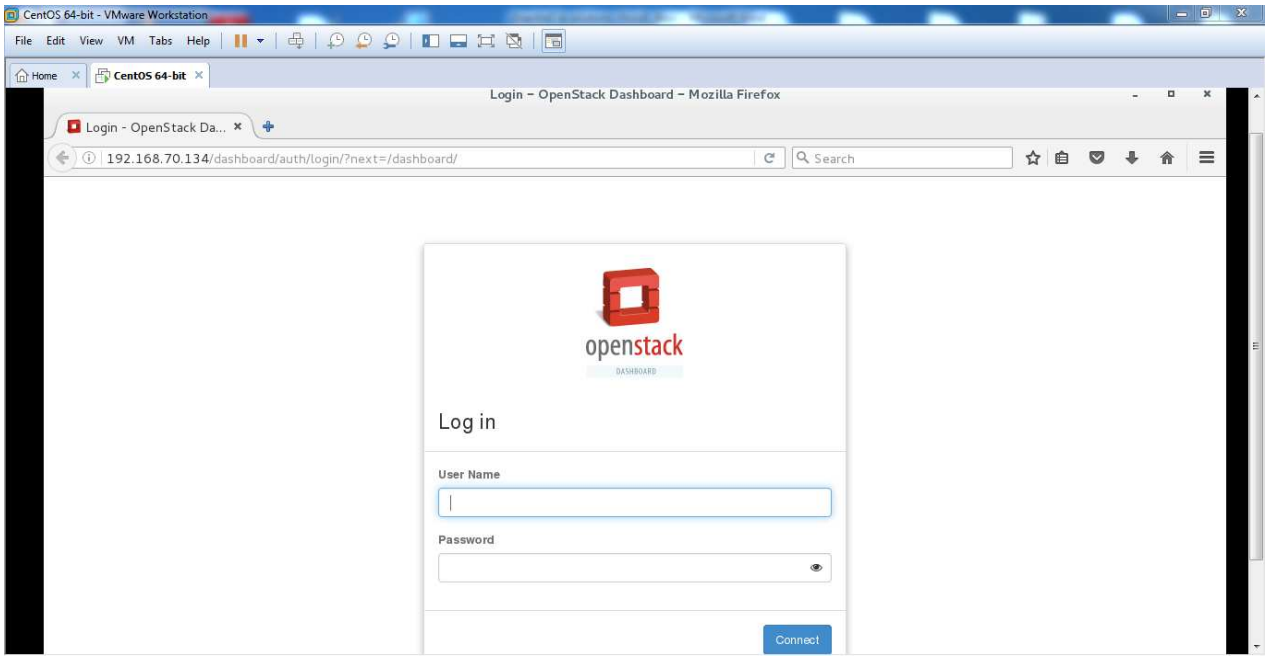


Figure 5.13:la page d'authentifiions pour *OpenStack*

V.2.3.3 Installation et configuration de développement : Wamp, php-opencloud via composer

Nous allons expliquée la méthode qu'on a choisi pour utiliser l'API d'*OpenStack* comme un moyen pour interagie avec notre portail web .

- **Utilisation du SDK php-opencloud avec *OpenStack* :**

La méthode choisit pour utiliser *OpenStack*-API c'est bien un SDK .

D'après Wikipédia, un SDK c'est :*Un kit de développement ou trousse de développement logiciel est un ensemble d'outils permettant aux développeurs de créer des applications de type défini. Les kits de développement logiciels sont souvent désignés par le sigle anglais SDK (Software Development Kit) ou devkit.*

Concrètement, c'est un ensemble de librairies pouvant être utilisées dans une application.

Dans le cas présent, php-opencloud permet de gérer une plate-forme OpenStack via les API de ce dernier. Ce SDK est maintenu par la société Rackspace, il est compatible avec *OpenStack* et donc avec la quasi totalité de ses composants*(excepté* Ceilometer*)*.

Pour résumer, ce SDK vous permettra d'intégrer la gestion de votre plate-forme *OpenStack* à votre site internet développé en PHP, au lieu de passer par l'interface Horizon.

php-opencloud aurait pu être fourni sous la forme de paquet, peut-être qu'un jour ce sera la cas mais pour le moment git sera notre ami. php-opencloud utilise curl pour communiquer avec les API *OpenStack*, il faut s'assurer que l'extension curl pour PHP soit installée ou compilée.

L'installation est assez simple, soit sur linux ou bien sur Windows, elle se déroule en quatre étapes :

1. Installation des paquets nécessaires PHP, Curl et Git.
2. Récupération des sources de php-opencloud.
3. Installation de Composer.
4. Installation des dépendances de php-opencloud via Composer.

Informations supplémentaires au sujet de Composer :

Composer est un outil pour gérer les dépendances en PHP. Les dépendances, dans un projet, ce sont toutes les bibliothèques dont votre projet dépend pour fonctionner. Par exemple, votre projet utilise la bibliothèque SwiftMailer pour envoyer des e-mails, il « dépend » donc de SwiftMailer.

Pour l'installation du SDK et de ses dépendances via Composer il existe deux commande :

1.composer require php-opencloud/*OpenStack*.

2. composer require rackspace/php-opencloud.

Nous, on a choisi la deuxième commande, ensuite on a besoin de charger le composer automatiquement(qui enregistre tous les espaces de nom requis), pour ce faire, on place la ligne de code PHP suivant**: require** 'vendor/autoload.php',en haut des fichiers PHP de notre application.

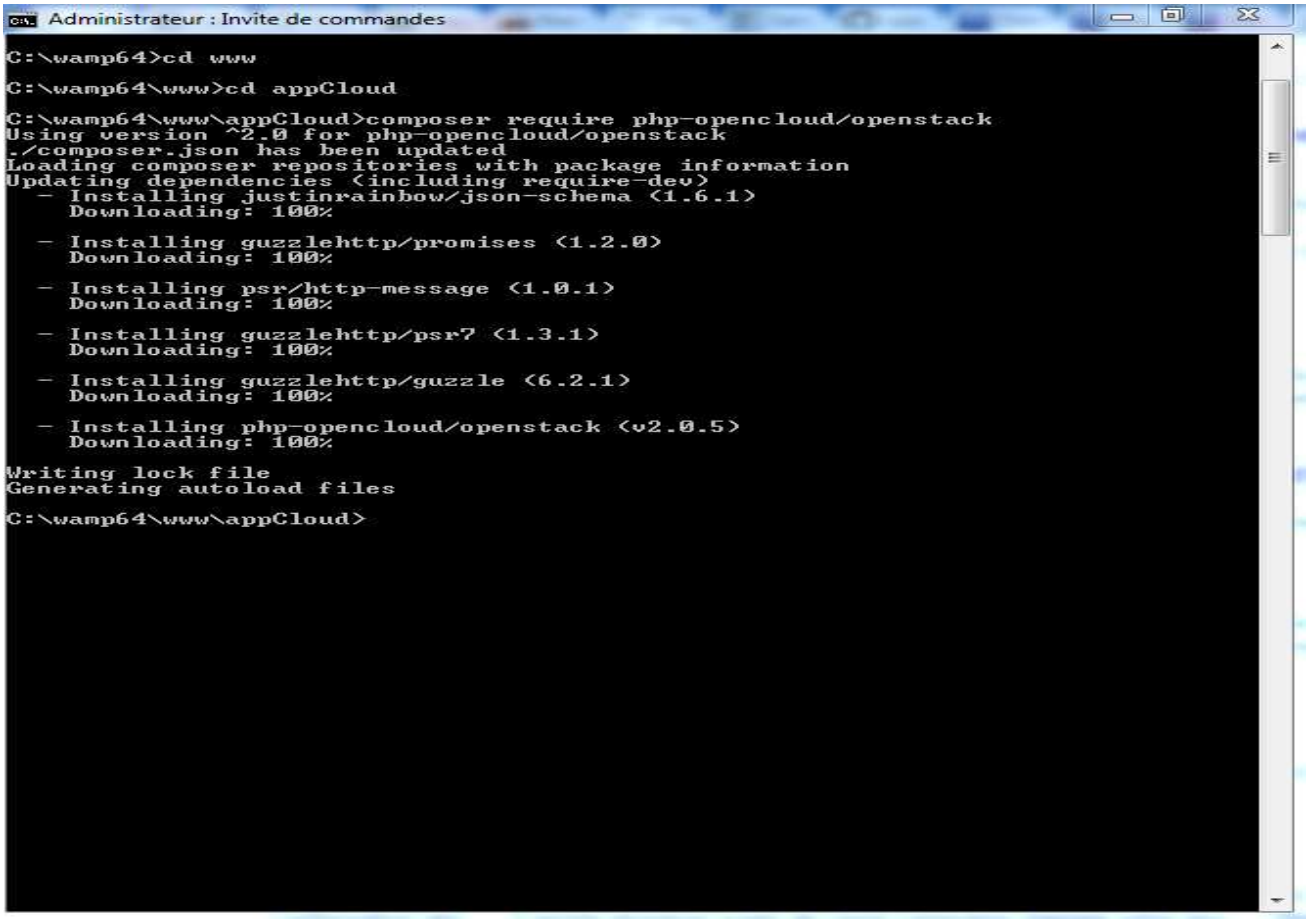
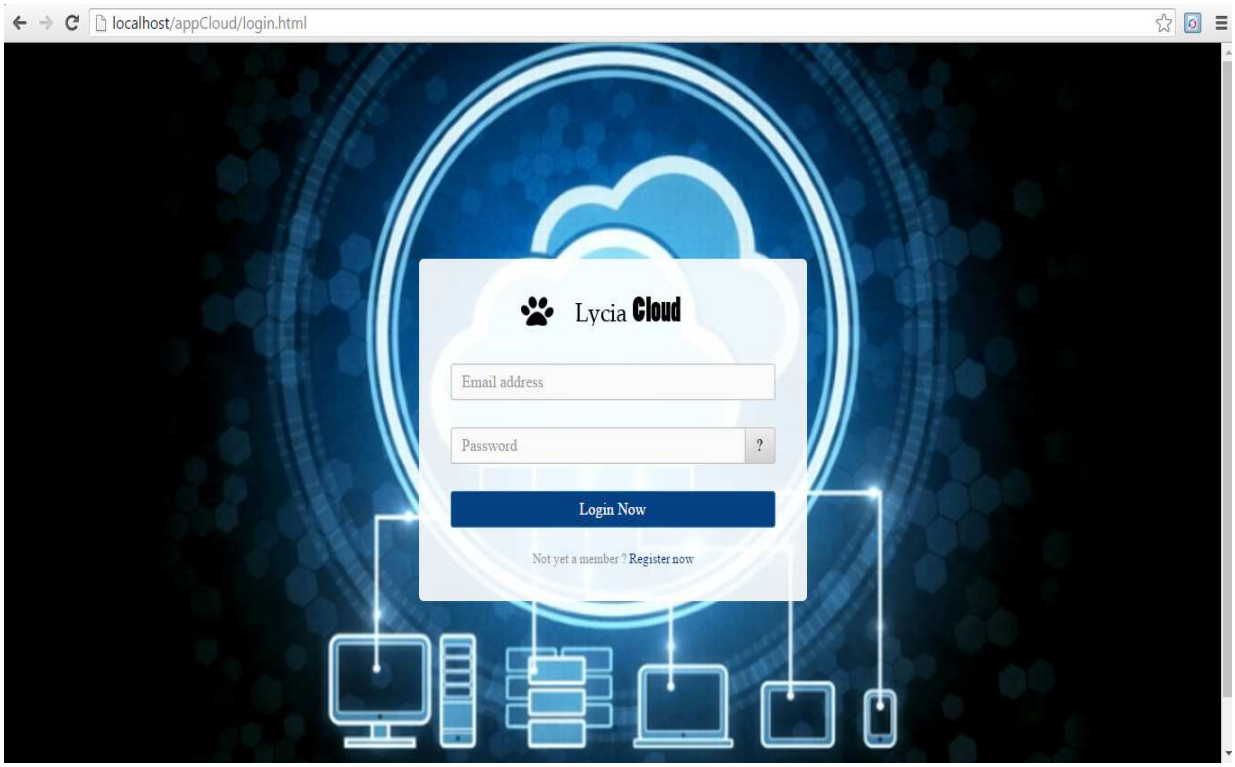


Figure 5.14:lancement de la commande choisit sur l'invite de commande de Windows en tant qu' administrateur

V.3. Présentation des interfaces graphiques:



Conclusion:

Dans ce chapitre, nous avons mis en place la description des détails de l'environnement matériel et logiciel dans lequel le travail a été effectué pour utiliser l'API d'*OpenStack* dans le but d'interagir la gestion de notre plate-forme *OpenStack* à notre portail web interne développé en PHP au lieu de passer par l'interface Horizon, après la planification et déploiement de *OpenStack* .Ensuite, illustrer les différentes étapes de réalisation et implémentation. Après avoir compris et métriser le fonctionnement du SDKs *OpenCloud*, nous avons montrer comment on peut développer un portail web en utilisant ce dernier.

Conclusion Générale

Le Cloud Computing est un concept qui prend de plus en plus d'ampleur dans le domaine de l'IT, les entreprises adoptent ce modèle afin de bénéficier de ces multiples avantages.

Notre objectif dans ce travail est de présenter le déploiement d'une plateforme de Cloud Computing et de mettre en place une application de virtualisation de bureau pour permettre à l'utilisateur d'accéder à des ressources, des capacités de calcul et de stockage et des applications à la demande.

Au départ, après qu' on a compris le concept de Cloud et la virtualisation on a choisit Vmware comme solution infrastructure as a service mais lorsque on a passé à l'étape développement pour notre portail web ,non seulement le matériel nécessaire pour remplir les besoins exigés n'était pas entièrement disponible, Vmware c'est une solution propriétaire donc la documentation sur ces API sont privé car est une solution payante, Pour cela, nous avons fait appel à l'outil « OpenStack » sur linux (distribution Centos7) puisque est une solution opensource.

OpenStack est un ensemble de logiciels open source permettant de déployer des infrastructures de Cloud Computing. La technologie possède une architecture modulaire composée de plusieurs projets corrélés (Nova, Swift, Glance...) qui permettent de contrôler les différentes ressources des machines virtuelles telles que la puissance de calcul, le stockage ou encore le réseau inhérents au centre de données sollicité. Le projet est porté par la Fondation OpenStack, une organisation non-commerciale qui a pour but de promouvoir le projet OpenStack ainsi que de protéger et d'aider les développeurs et toute la communauté OpenStack.

Après l'étude et déploiement d'OpenStack, on l'a utiliser et tester en préparant l'environnement de développement pour interagir la gestion de notre plate-forme OpenStack à notre portail web interne développé en PHP.

Nous avons acquis de nombreuses connaissances techniques et technologiques et la capacité de prendre des décisions quant au choix du matériel à utiliser et des outils adéquats pour un besoin exprimé par une entreprise.

Enfin, la réalisation de ce projet nous a permis, non seulement d'appliquer les connaissances acquises durant nos études, mais aussi nous a donné l'occasion d'apprendre et maitriser de nouveaux outils et nouvelles technologies.

Cela s'est concrétisé par l'amélioration de nos compétences, que ce soit sur le plan technique ou le plan social et la découverte du milieu professionnel.

Annexe A

Bibliographie:

[SAN 10]: Santy François. La Virtualisation. Thèse. Projet de recherche et communication. Faculté des Sciences - Département d'Informatique : Université Libre de Bruxelles,2010. 22p.

[PT09a]: Thibaud Chardonnens, Les enjeux du Cloud Computing en entreprise, Université de Fribourg, Suisse, 91pages.

[MIT5]: CISCO, Les bases du Cloud Computing : revaloriser les technologies de l‟‟information, Mai 2011, 7 pages.

[9]: Mémoire MASTER ACADEMIQUE, Etude et mise en place d'une solution Cloud Computing privé au sein de l'université de ouargla, Présenté par :Akbi khalil, Zehri mohammed, Année Universitaire 2012 /2013.

[6]: Rapport de stage pour le projet, Installation et configuration d'un outil de Cloud Computing en ligne, Réalisé par **Heni BELLAAJ,** Encadreur par M. **Mahdi Khemakhem ,** Mr.**Mahmoud Ghorbel** ,2012/2013.

[5]: Rapport mini projet, Master 2 Systèmes embarqués et mobiles, OpenStack, Réalisé par Yasmina BENTCHAKAL, Yakoub BOUZIDI, Encadré par Mme **Samia BOUZAFRANE**, Mr **Jean-Marc FARINONE,**2015.

[10]: Introduction to Openstack, Running a Cloud Computing Infrastructure with OpenStack, 6th International Conference on Autonomous Infrastructure, Management and Security04 June 2012, University of Luxembourg.

[11]: Maxime Besson,Virtualisation et cloud open source, décembre 2012, Smile,50 pages.

Webographie:

[S1] https://fr.wikipedia.org/wiki/Cloud_computing

[S2] http://www.histoire-cigref.org/blog/cloud-computing-histoire-de-linformatique-en-nuages/

[S3] http://www.figer.com/Publications/nuage.htm#.V3SyfvmLTIU

[S4]http://www.vinci-energies.com/cest-deja-demain/pour-un-monde-connecte/data-center-bienvenue-dans-les-usines-a-donnees/

[S5] http://www.channelnews.fr/qui-sont-les-principaux-vendeurs-de-services-de-cloud-l-computing-r-1998

[S6] http://www.lesechos.fr/idees-debats/cercle/cercle-118688-cloud-computing-la-securite-en-question-1065146.php

[S7]http://linuxfr.org/news/presentation-d-openstack

[S9]http://docs.openstack.org/liberty/fr/install-guide-ubuntu/

[S10]:http://docplayer.fr/562935-Le-cloud-computing-est-il-une-solution-d-avenir-pour-lentreprise.html.

[S11]: http://docplayer.fr/561407-Le-cloud-on-parle-de-quoi.html.

Annexe A

Les interfaces de portail veulent atteindre:

La partie visible du nuage:

Il existe deux types d'utilisateur du nuage, administrateur ou membre d'un projet. Selon le type d'utilisateur des interfaces ou d'autres s'affichent après l'authentification:

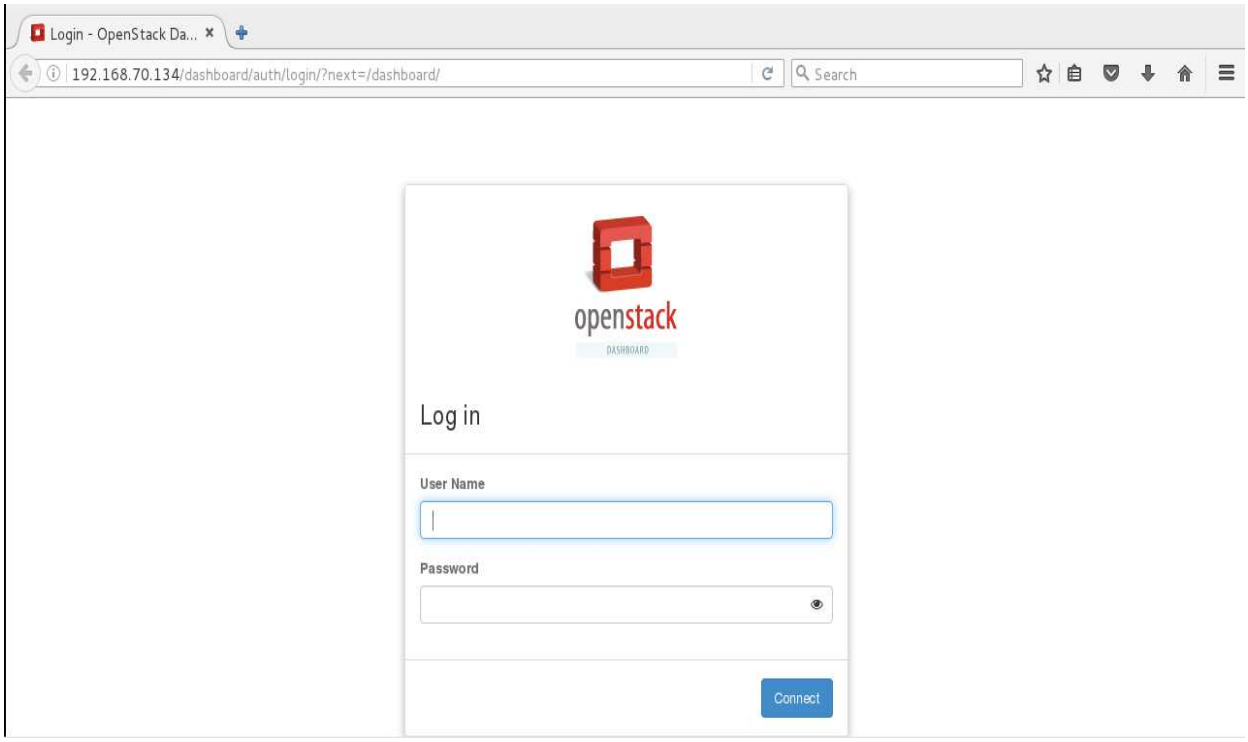


Figure 1: Se connectez sur Horizon

1. Administrateur:

1.1 Se connecter sur Horizon :

La première étape qui devrait être effectuée par l'administrateur pour qu'il puisse se connecter à l'horizon est l'authentification comme le montre Figure 1.

1.2 Overview :

Une fois connecté, en fonction des privilèges d'accès, l'utilisateur est autorisé à accéder à des projets spécifiques. Ce qui suit (figure) est une page d'aperçu pour un projet appartenant à l'utilisateur "admin". On peut visualiser et télécharger des rapports métriques d'utilisation de base ici.

Annexe A

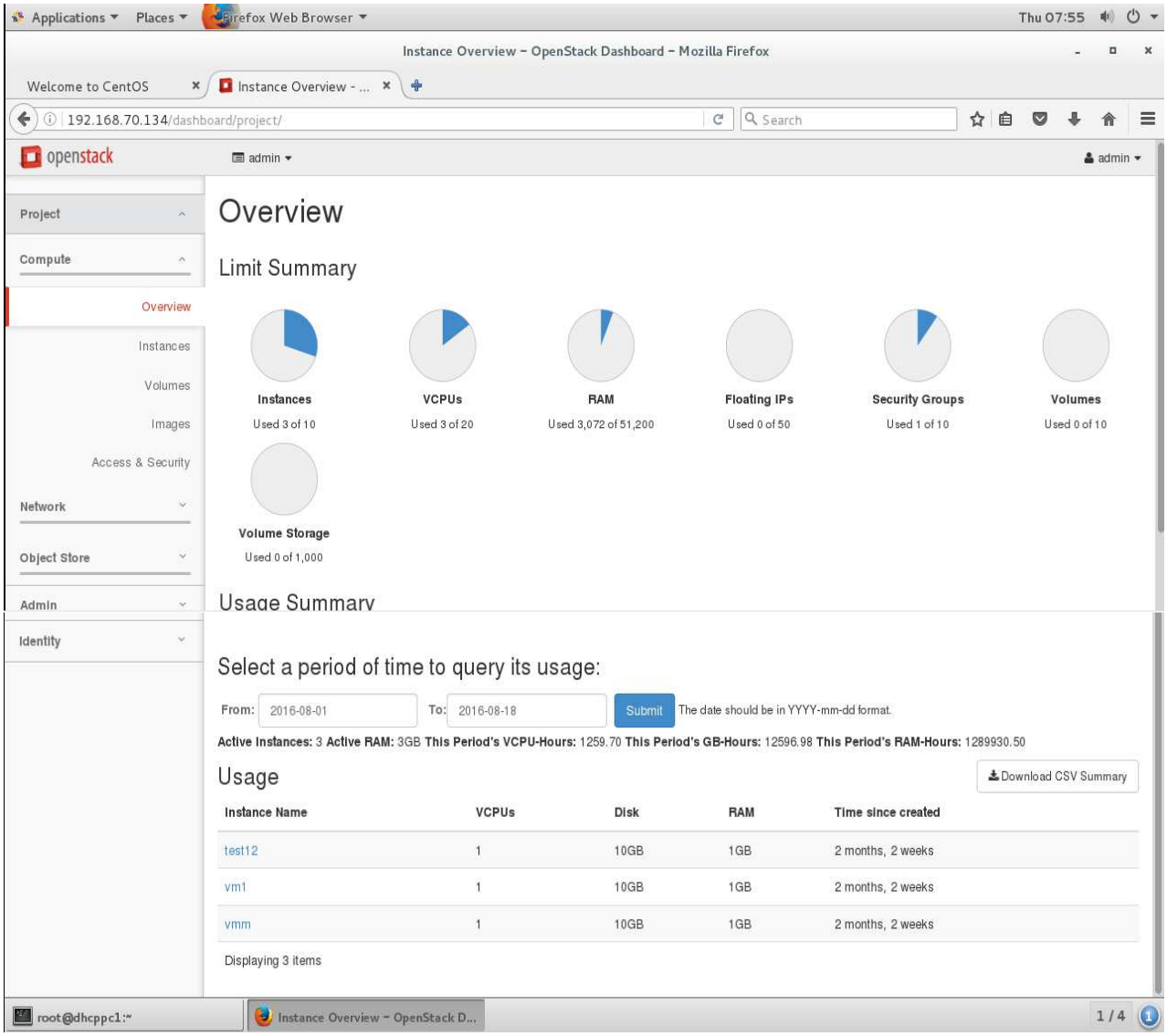


Figure 2: Administrateur :Overview

1.3 Gestion des Instances:

La figure suivante contient une présentation des instances en cours d'exécution. Depuis cette page l'administrateur peut mettre fin, pause, redémarrage des instances, connexion à la console vnc de l'instance.

Annexe A

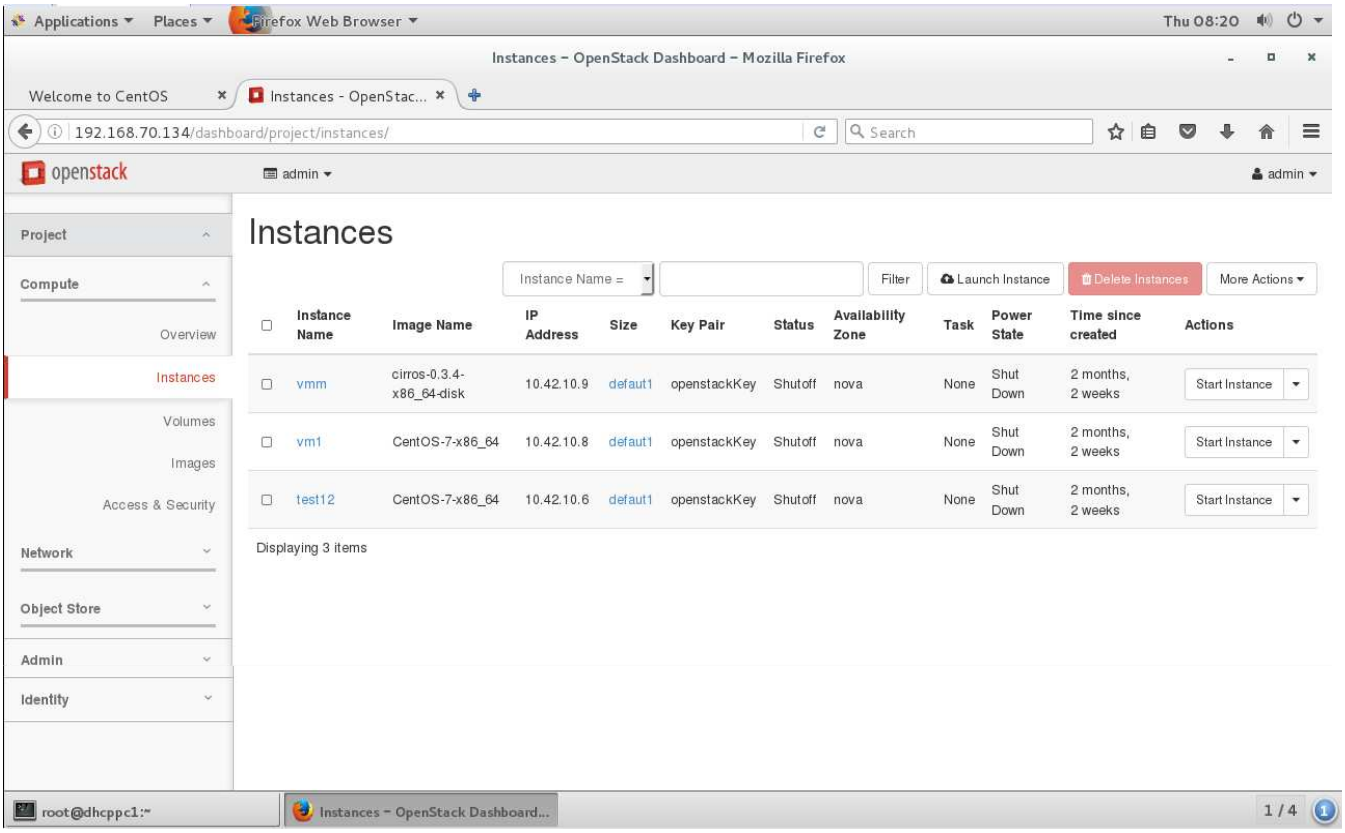
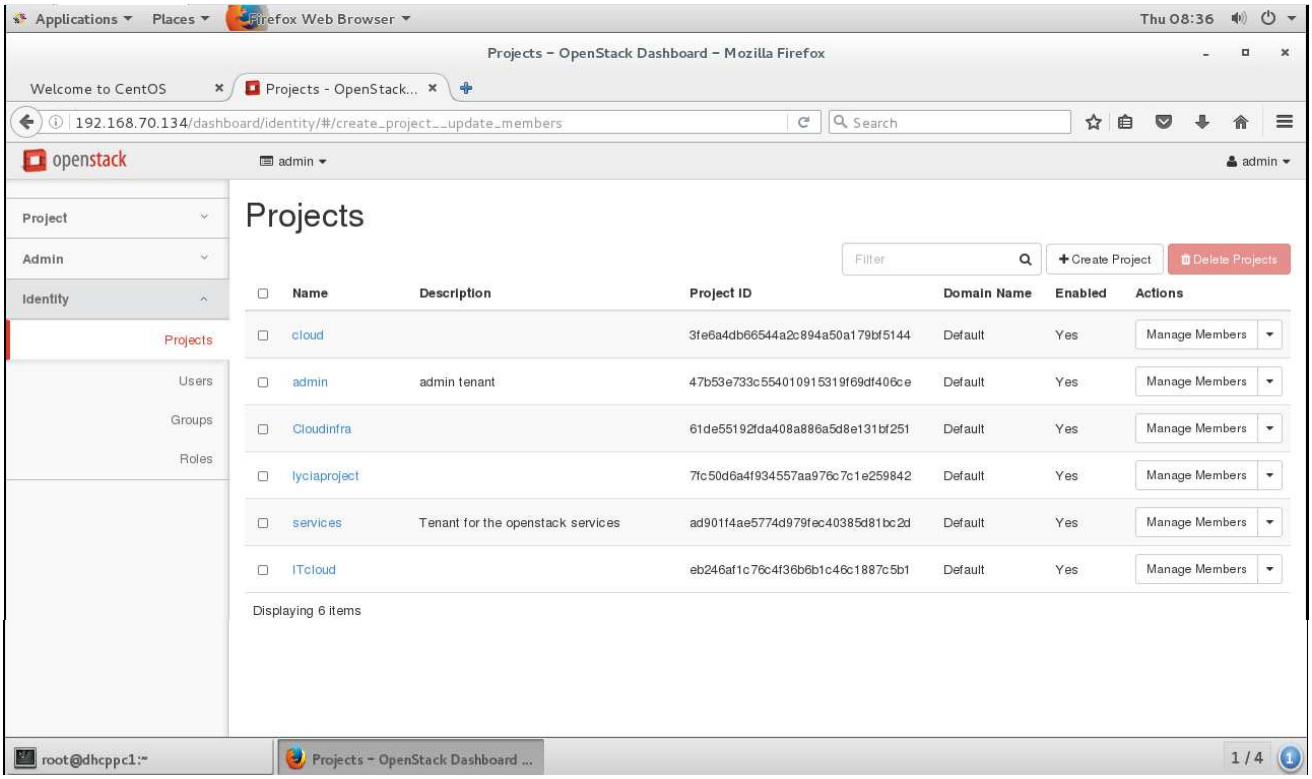


Figure 3: Administrateur: Instances

1.4 Gestion des Projets :

La figure suivante liste les projets disponibles (Tenants) qui ont été créés. On peut aussi créer de nouveaux projets, affecter des utilisateurs aux projets, etc.



Annexe A

1.5 Gestion des Utilisateurs:

La figure montre la page liste les utilisateurs qui ont été créés. On peut aussi créer de nouveaux utilisateurs et/ou désactiver / supprimer des utilisateurs existants.

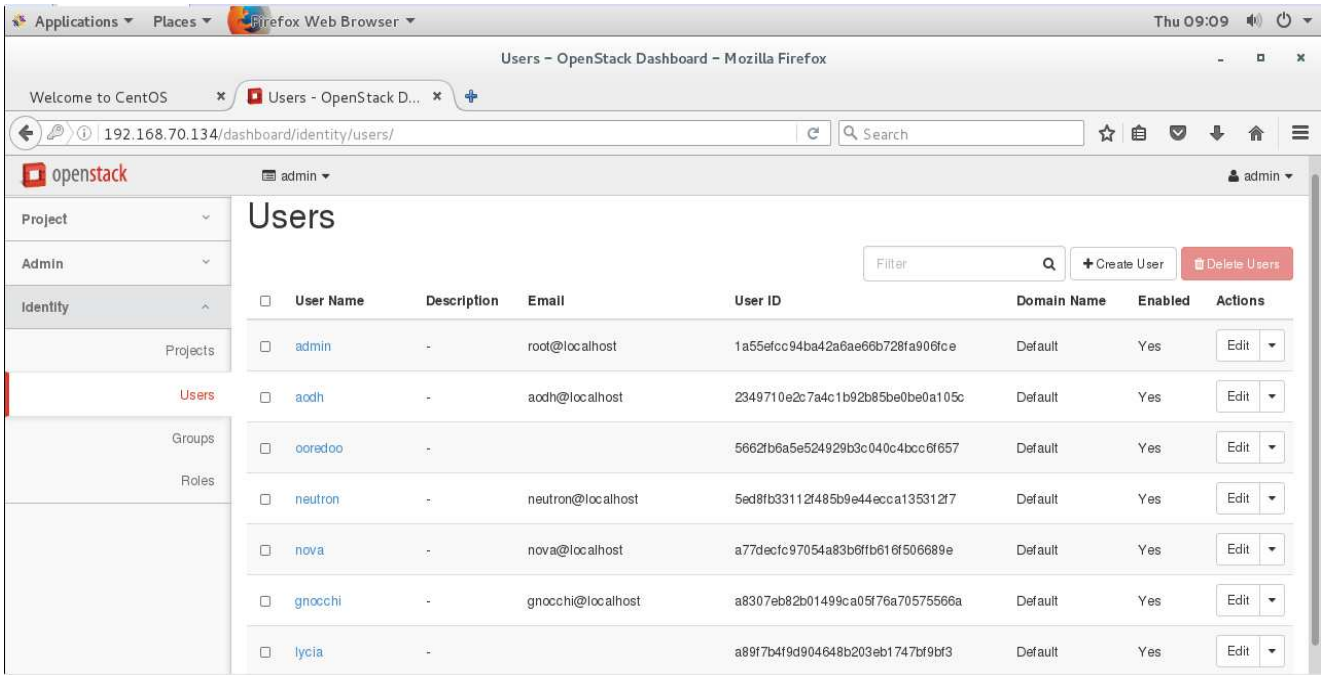


Figure 4 : Administrateur :Utilisateurs

1.6 Gestion des Flavors:

La page suivante (figure) liste les Flavors actuellement disponibles qui peuvent être utilisés pour lancer une instance. On peut également créer des serveurs personnalisées sur cette page.

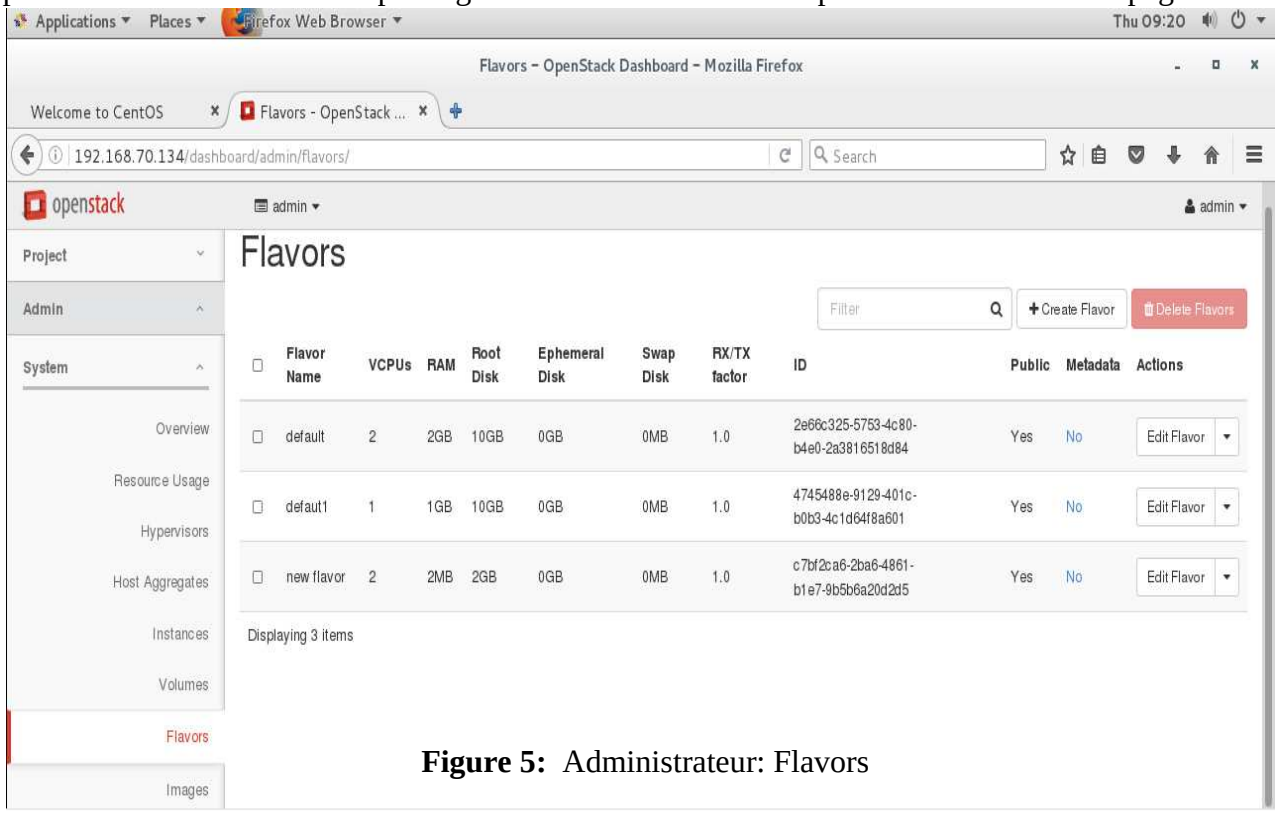


Figure 5: Administrateur: Flavors

Annexe A

1.7 Gestion des Quotas:

La page suivante (figure)liste les quotas des ressources allouées à un utilisateur, le nombre de processeurs, quantité de RAM, espace disque, max, nombre d'instances qui peuvent être soulevées . . .

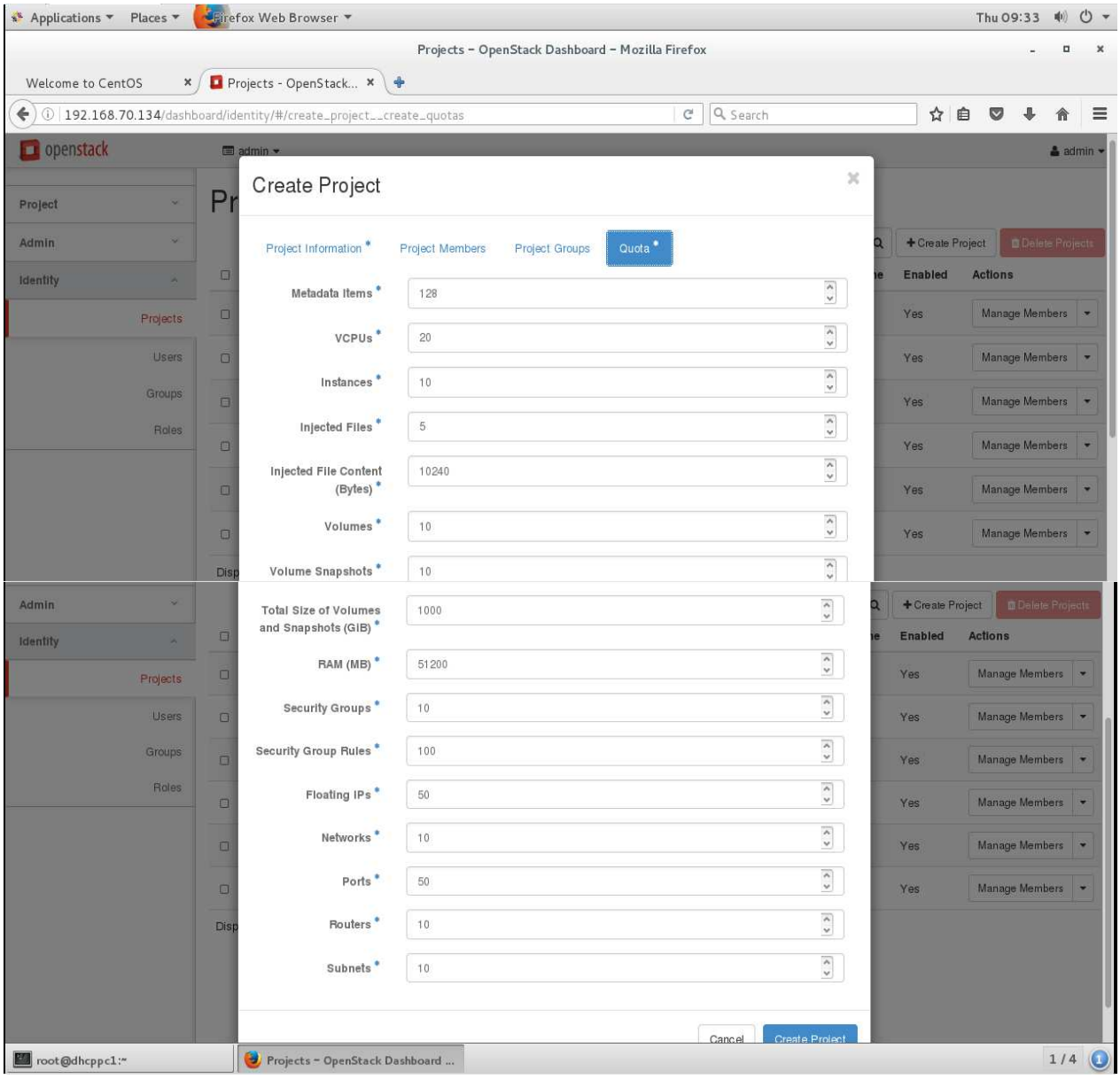


Figure 6: Administrateur: Quotas

Annexe A

1.8 Gestion de Network:

la page suivante (figure) liste les réseaux créer pour le déploiement des machine virtuelle (instance) ,comme montre la figure on a créer un réseau publique et un réseau privé.

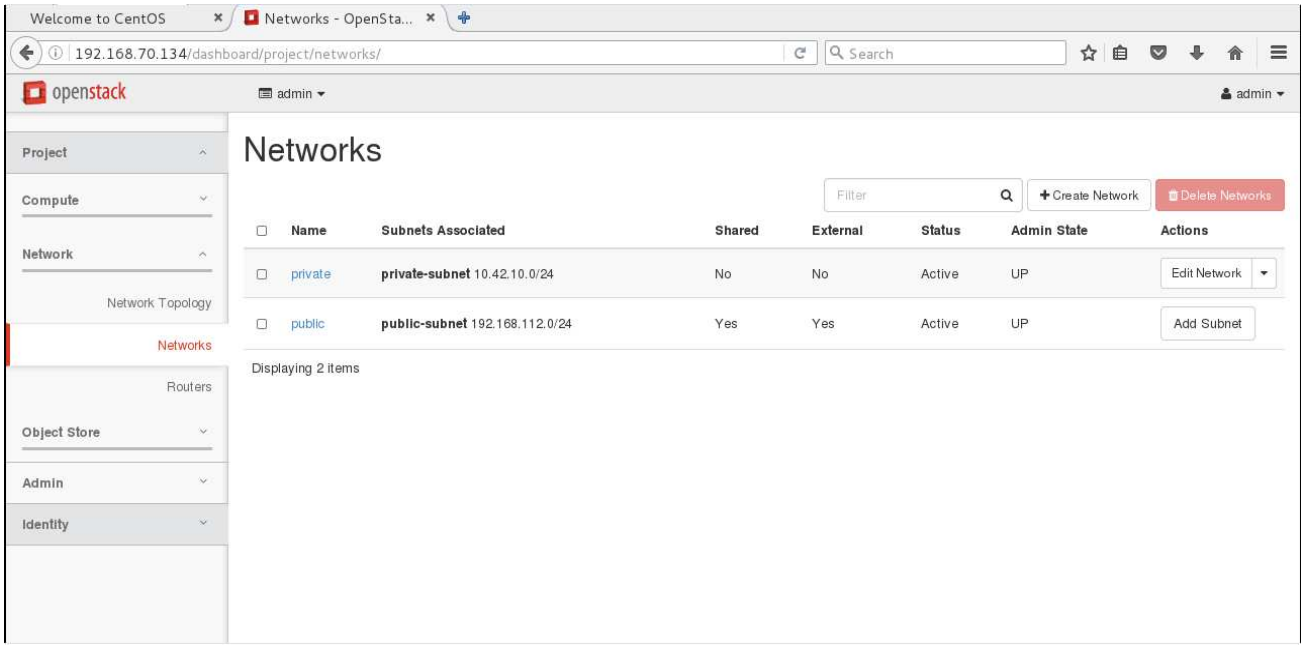


Figure7: Administrateur: Network

Network Topology:

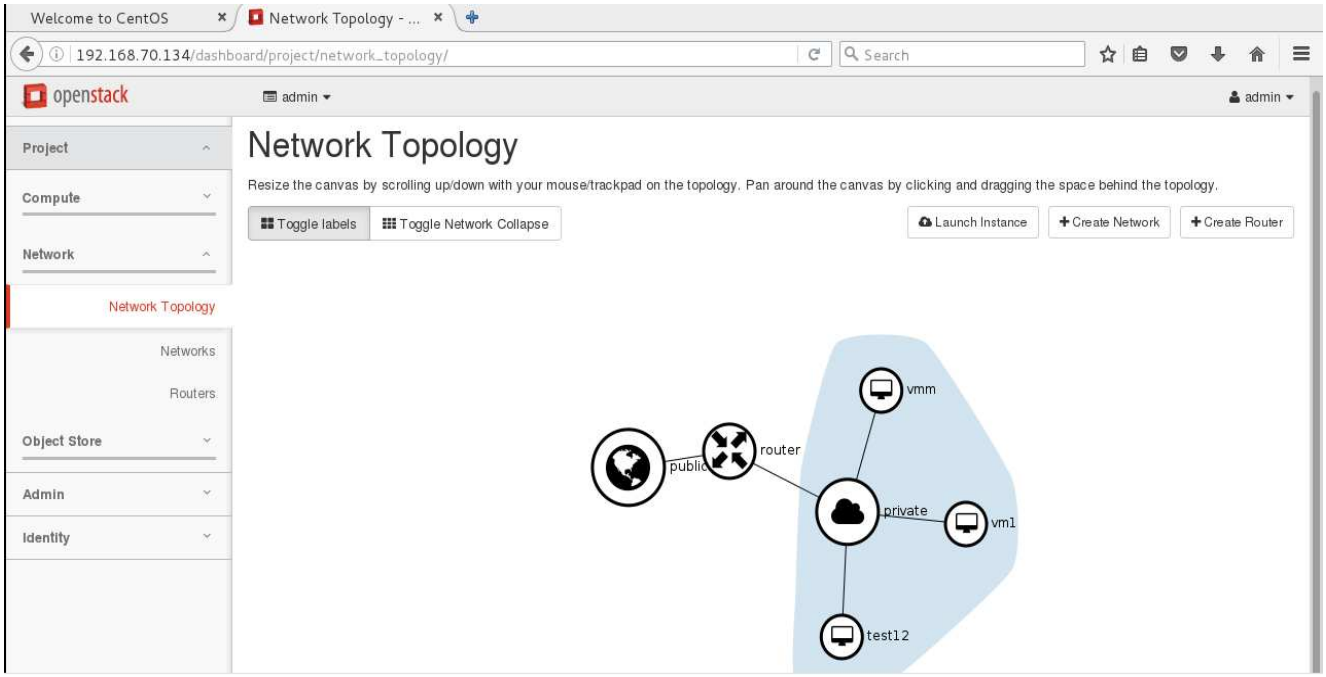


Figure 8: une aperçue sur la topologie de notre réseau

Annexe A

2. Membre d'un projet:

La première étape qui devrait être effectuée par un membre pour qu'il puisse se connecter à l'horizon est l'authentification.

2.1 Overview:

La figure montre un aperçu du projet "IIT ou on peut visualiser et télécharger des rapports métriques d'utilisation de base.

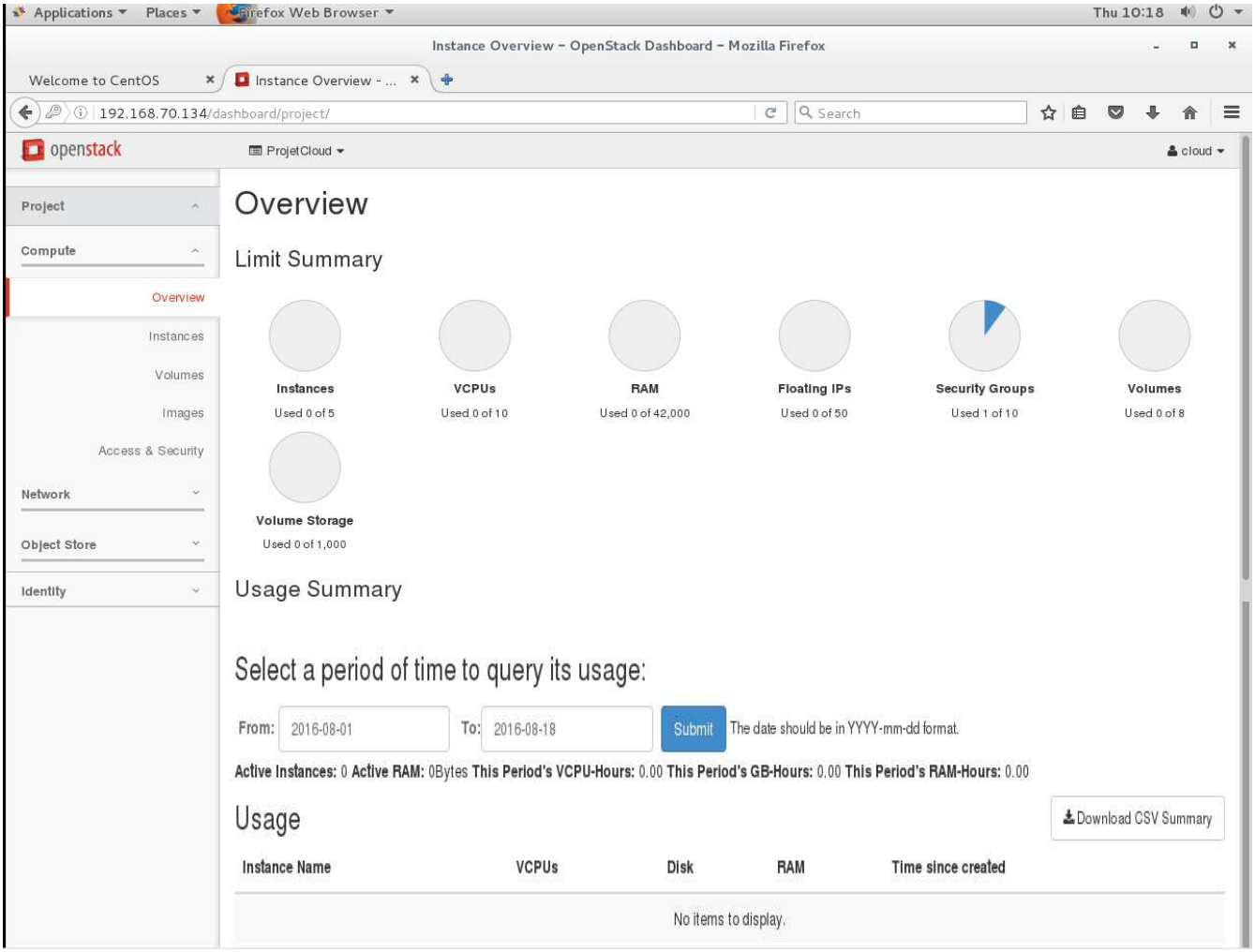


Figure 9: Membre: Overview

Annexe A

2.3 Gestion des Images :

La page suivante (figure) liste les images personnalisées qui ont été téléchargées. On peut modifier les propriétés de l'image, supprimer et lancer de nouvelles instances d'images. Cette page répertorie également les photos prises à partir des instances et des volumes.

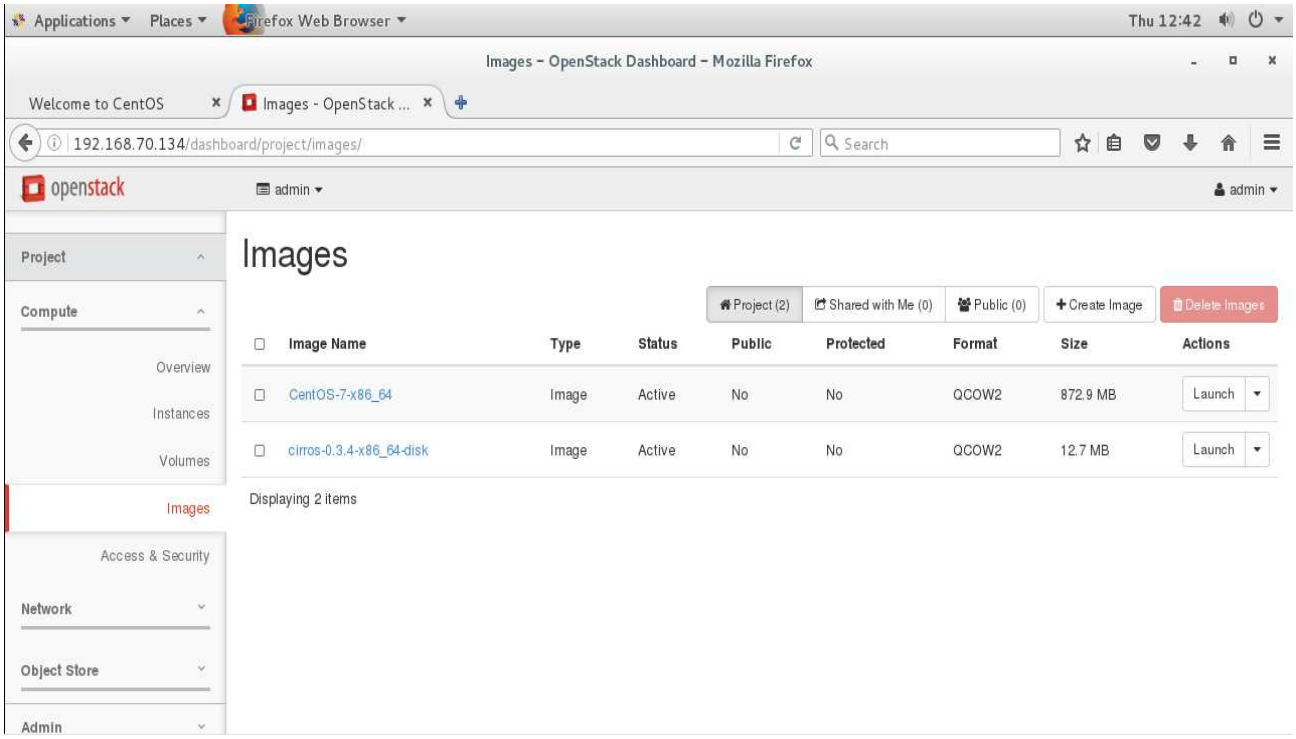


Figure 10: Membre: image

Annexe B

B.1 Un Hôte, hyperviseur, ESXi, barre métal,…:

Serveur physique, le matériel (HP Proliand, IBM X Series,…)

B.2 Un Serveur :

OS ou environnement faisant tourner un ou X services

B.3 Un Service :

Fonctionnalité, logiciel (Base de données, Site Web, application,…)

B.4 Mainframe

Est un ordinateur de grande puissance de traitement

B.5 Système d'information

Un système d'information (SI) est un ensemble organisé de ressources (matériels, logiciels, personnel, données et procédures) qui permet de collecter, regrouper, classifier, traiter et diffuser de l'information sur un environnement donné.

B.6 Datacenter

Un centre de traitement de données est un site physique sur lequel se trouvent regroupés des équipements constituants du système d'information de l'entreprise (ordinateurs centraux, serveurs, baies de stockage, équipements réseaux et de télécommunications, etc.). Il peut être interne et/ou externe à l'entreprise, exploité ou non avec le soutien de prestataires.

B.7 Baies de stockage

Une baie de stockage est un équipement de sauvegarde de données informatique.

Bien que son apparence varie souvent, suivant sa taille et son utilisation, sa structure et toujours la même. on retrouve plusieurs éléments, indispensable à son fonctionnement.

Les disques: La baie comporte une série de disques de stockage, qui sont chargés d'emmagasiner les données. La taille de la baie de disque dépend bien évidemment, de la quantité à stocker.

Un bus: Le bus est l'élément, par lequel les différents éléments de la baie vont Communiquer.

C'est donc par son intermédiaire que toute les données vont circuler. Il est souvent de normes PCI, Infini Band ou Rapid IO.

Le processeur: C'est l'unité de calcul qui va traiter toutes les informations. Il est le centre névralgique de la baie de stockage, et fonctionne la même manière que dans un ordinateur.

B.8 Switch:

Le Switch est un matériel d'interconnexion de type concentrateur réseau mais il fractionne le réseau en domaines de collision indépendants.

B.9 VLAN:

Annexe B

Un réseau local virtuel, communément appelé VLAN(pour virtuelle LAN), est un réseau informatique logique indépendant. De nombreux VLAN peuvent coexister sur un même commutateur réseau.

B.10 LAN:

Un réseau local, souvent désigné par l'acronyme anglais LAN de Local Area Network, est un réseau informatique tel que les terminaux qui y participent (ordinateurs, etc.) s'envoient des trames au niveau de la couche de liaison sans utiliser d'accès à internet.

B.11 Rackspace:

C’est une société d’hébergement informatique basée à San Antonio, Texas, USA.

La société possède également des bureaux en Australie, le Royaume-Uni, Suisse, Pays Bas et Hong Kong, et des centres de données fonctionnant dans le Texas, l’Illinois, la Virginie, le Royaume-Uni, l’Australie et Hong Kong . L’e-mail de l’entreprise et de la division applications fonctionne de Blacksburg, VA, les autres bureaux sont situés à Austin, TX et San Francisco, CA.

B.12 Nasa

La National Aéronautique and Space Administration («Administration nationale de l’aéronautique et de l’espace »), plus connue sous son abréviation NASA, est l’agence gouvernementale qui a en charge la majeure partie du programme spatial civil des États-Unis.

B.13 iSCSI

C’est une abréviation d’Internet Small Computer System Interface. C’est un protocole de stockage en réseau basé sur le protocole IP destiné à relier les installations de stockage de données.