Insitut National des Postes et Telecommunications

RAPPORT DE PROJET DE FIN D'ANNEE

Rapport

Mise en place d'un cloud privé en utilisant OpenStack

Realisé par :
NAJMI yassin
BABA salmane

Encadrant :
EL-NOUAARY abdeslame

Table des matières

Introduction

Dans un monde où la transformation numérique et l'explosion des données sont devenues des réalités incontournables, les organisations exigent des infrastructures informatiques à la fois flexibles, évolutives et économiques. Le cloud computing est la réponse dominante à ces impératifs, offrant la capacité de mobiliser des ressources informatiques à la demande.

Parmi les différents modèles de déploiement, le cloud privé occupe une position stratégique, car il allie les avantages de l'élasticité et de l'automatisation à un niveau de contrôle et de sécurité essentiel, notamment pour les institutions traitant des données sensibles. La mise en place d'une telle infrastructure représente un enjeu majeur pour optimiser la gestion des ressources informatiques et réduire les coûts d'exploitation.

Dans ce contexte, OpenStack s'est imposé comme une solution open source de référence mondiale pour la création de clouds privés et publics. Initiée en 2010 par la NASA et Rackspace, cette plateforme se distingue par sa modularité et son architecture distribuée, permettant de gérer de manière centralisée les services d'Infrastructure-as-a-Service (IaaS), tels que le calcul, le stockage et le réseau.

Le choix de ce projet de fin d'année a été motivé par la volonté d'acquérir une expertise pratique dans les technologies cloud et l'administration des systèmes distribués. En menant à bien ce travail, nous visons à démontrer non seulement la faisabilité technique de la mise en place d'un cloud privé OpenStack, mais aussi son potentiel comme plateforme pour des projets futurs au sein de l'Institut National des Postes et Télécommunications (INPT), contribuant ainsi à la modernisation de son infrastructure académique.

Ce rapport est structuré en plusieurs parties. Nous débuterons par une revue des concepts fondamentaux du cloud computing. Nous décrirons ensuite l'architecture d'OpenStack et ses composants principaux. La troisième partie sera consacrée à la conception et à la mise en œuvre pratique de notre solution, en détaillant les étapes d'installation et de configuration. Enfin, nous présenterons une analyse des résultats, des difficultés rencontrées et des perspectives pour l'avenir de ce projet.

1 Généralités sur le Cloud Computing

1.1 Définition et principes du Cloud Computing

Le **cloud computing** est un modèle d'accès à des ressources informatiques (serveurs, stockage, applications, services) via Internet, de manière flexible et à la demande.

Selon la définition du NIST, il repose sur cinq principes fondamentaux :

— **Self-service à la demande** : les utilisateurs peuvent provisionner des ressources sans intervention humaine.
— **Accès large au réseau** : les services sont accessibles depuis différents terminaux (PC, smartphone, tablette).
— **Mutualisation des ressources** : les ressources sont partagées entre plusieurs clients (multi-tenant).
— **Élasticité rapide** : les ressources peuvent être augmentées ou réduites dynamiquement.
— **Service mesuré** : la consommation est facturée en fonction de l'usage réel.

1.2 Les modèles de services (IaaS, PaaS, SaaS)

Le cloud computing propose trois modèles principaux de services :

— **IaaS (Infrastructure as a Service)** : fournit des ressources matérielles virtuelles (machines virtuelles, stockage, réseau).
Exemple : AWS EC2, OpenStack Nova.
— **PaaS (Platform as a Service)** : offre une plateforme complète pour développer, tester et déployer des applications sans gérer l'infrastructure sous-jacente.
Exemple : Google App Engine, Heroku.
— **SaaS (Software as a Service)** : met à disposition des applications accessibles via un navigateur, sans installation locale.
Exemple : Gmail, Office 365.

1.3 Les types de déploiement (Cloud public, privé, hybride, communautaire)

Il existe différents modèles de déploiement du cloud :

— **Cloud public** : services accessibles au grand public via Internet (AWS, Azure, GCP).
— **Cloud privé** : infrastructure dédiée à une seule organisation, gérée en interne ou par un fournisseur. Sécurité et contrôle accrus.
— **Cloud hybride** : combinaison du cloud privé et public, permettant de répartir les charges et de protéger les données sensibles.
— **Cloud communautaire** : infrastructure partagée par plusieurs organisations ayant des objectifs communs (exemple : recherche universitaire, secteur médical).

1.4 Avantages et limites du Cloud Computing

Le cloud computing présente plusieurs bénéfices, mais également certaines contraintes :

- **Avantages** :
  - Réduction des coûts liés à l'achat et à la maintenance de l'infrastructure.
  - Flexibilité et scalabilité des ressources.
  - Accessibilité mondiale et collaboration facilitée.
  - Maintenance et mises à jour assurées par le fournisseur.
- **Limites / défis** :
  - Dépendance vis-à-vis du fournisseur (*vendor lock-in*).
  - Risques liés à la sécurité et à la confidentialité des données.
  - Problèmes de performance dus à la latence ou aux interruptions de service.
  - Coût élevé à long terme en cas de mauvaise gestion de la consommation.

2 Présentation d'OpenStack

La façon dont les gens consomment du contenu a radicalement changé au cours de la dernière décennie. Aujourd'hui, on regarde la télévision, on lit les actualités, on écoute de la musique et on partage du contenu sur Internet. L'adoption d'Internet a ouvert la voie à de nouvelles opportunités commerciales. Des plateformes comme Netflix, Spotify, Twitter et Facebook sont des exemples de logiciels en tant que service (SaaS) émergents, accessibles sur le Web. Le cloud computing est l'un des principaux moteurs du SaaS, fournissant l'infrastructure et la puissance de calcul sur Internet, offrant aux entreprises les ressources et la puissance nécessaires pour exploiter cette opportunité tout en maîtrisant leur budget informatique. Le cloud computing a récemment transformé la façon dont les entreprises gèrent leurs activités, offrant de meilleures performances, une plus grande flexibilité et une plus grande élasticité à leur infrastructure informatique. Il existe deux principaux types de technologies cloud : le cloud public et le cloud privé. Le cloud public fournit de la puissance de calcul à un groupe d'utilisateurs partageant les mêmes ressources serveur, tandis que le cloud privé est un environnement propriétaire. OpenStack est actuellement la suite logicielle open source la plus populaire pour la création et le déploiement de solutions de cloud computing. Elle se compose de plusieurs projets/services permettant aux entreprises de contrôler leurs ressources informatiques, leur réseau et leur stockage. Avec OpenStack, vous pouvez gérer un vaste parc de serveurs via un tableau de bord web ou l'API OpenStack. Ce document vise à expliquer en termes simples la solution OpenStack : ses avantages, ses services et son infrastructure.

2.1 Historique et communauté OpenStack

L'initiative OpenStack a été lancée en 2010. Sa mission était, et est toujours, de «produire une plateforme de cloud computing open source universelle répondant aux besoins des clouds publics et privés, quelle que soit leur taille, grâce à sa simplicité de mise en œuvre et sa grande évolutivité». Depuis, OpenStack est utilisé par des centaines de leaders de tous les secteurs, tels que BMW, Volkswagen, BestBuy et bien d'autres, pour alimenter et gérer leur infrastructure d'hébergement.

Au fil des ans, OpenStack a rapidement évolué pour devenir la pile logicielle open source la plus populaire pour le cloud computing. Depuis le lancement de la Fondation OpenStack en 2012, OpenStack a continué de croître, de plus en plus de particuliers et d'entreprises investissant dans le projet, découvrant de nouvelles failles et comblant rapidement les lacunes. OpenStack publie des versions officielles tous les six mois, généralement en octobre et en avril. Voici comment OpenStack a évolué au fil des ans pour devenir ce qu'il est aujourd'hui.

Nom	Date	Nouveaux composants introduits
Austin	21 octobre 2010	Nova, Swift (bases du projet)
Bexar	3 février 2011	Glance (service d'images)
Essex	5 avril 2012	Horizon (tableau de bord), Keystone (authentification)
Folsom	27 septembre 2012	Quantum (devenu Neutron), Cinder (stockage en blocs)
Havana	22 octobre 2013	Heat (orchestration), Ceilometer (télémétrie)
Icehouse	17 avril 2014	Trove (base de données as a service)
Juno	16 octobre 2014	Sahara (traitement big data)
Kilo	30 avril 2015	Ironic (bare metal provisioning)
Liberty	26 octobre 2015	Manila (stockage partagé), Barbican (secrets), Zaqar (messaging)
Mitaka	7 avril 2016	Stabilisation et intégration des composants existants
Newton	6 octobre 2016	Améliorations des services existants
Ocata	22 février 2017	Améliorations et intégration container
Pike	30 août 2017	Améliorations et optimisation
Queens	28 février 2018	Zun (containers), Qinling (serverless), Octavia (load balancer)
Rocky	30 août 2018	Cyborg (accélérateurs matériels), Masakari (HA)
Stein	10 avril 2019	Placement (gestion des ressources séparée de Nova)
Train	16 octobre 2019	Consolidation des services existants
Ussuri	13 mai 2020	Adjutant (self-service tasks)
Victoria	14 octobre 2020	Skyline (nouveau tableau de bord expérimental)
Wallaby	14 avril 2021	Améliorations Skyline et services existants
Xena	6 octobre 2021	Stabilisation des services précédents
Yoga	30 mars 2022	Consolidation et améliorations des composants
Zed	5 octobre 2022	Venus (gestion de logs), Skyline stabilisé
2023.1 Antelope (SLURP)	22 mars 2023	Améliorations Skyline et services existants
2023.2 Bobcat	10 octobre 2023	Optimisation des services existants
2024.1 Caracal (SLURP)	3 avril 2024	Améliorations composants existants
2024.2 Dalmatian	2 octobre 2024	Consolidation et stabilisation des services

Nom	Date	Nouveaux composants introduits
2025.1 Epoxy (SLURP)	2 avril 2025	Freezer-api (sauvegarde/restauration) officialisé
2025.2 Flamingo	1 octobre 2025 (estimé)	En développement

2.2 Architecture générale d'OpenStack

OpenStack is a free open cloud computing platform, deployed as Infrastructure-as-a-Service (IaaS), where one can provide virtual services and resources as both public and private cloud.

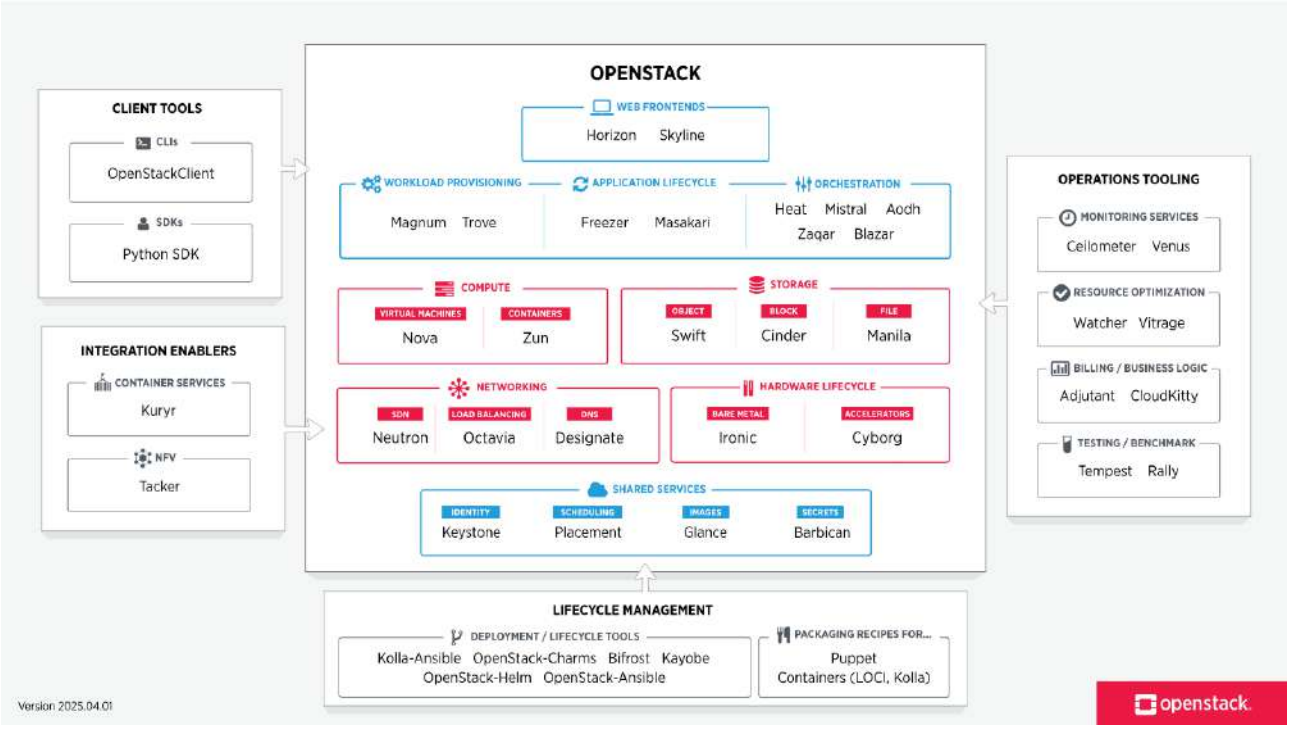


FIGURE 1 – Architecture générale d'OpenStack

2.3 Principaux composants

L'architecture d'OpenStack se compose de multiples projets Open Source. Ces projets sont utilisés pour paramétrer l'environnement undercloud pour les administrateurs système et l'environnement overcloud pour les utilisateurs du cloud. Les underclouds contiennent les composants essentiels dont les administrateurs système ont besoin pour paramétrer et gérer les overclouds, c'est-à-dire les environnements OpenStack des utilisateurs finaux.

Six services essentiels assurent la puissance de calcul, la mise en réseau, le stockage, la gestion des identités et la gestion des images. Une dizaine d'autres projets, disponibles en option, sont actuellement à un stade de développement plus ou moins avancé. Ces six services essentiels forment l'infrastructure qui permet aux projets restants de prendre en charge la création de tableaux de bord, l'orchestration, le provisionnement sur systèmes bare metal, la messagerie, les conteneurs et la gouvernance.

2.3.1 Keystone (Identity)

OpenStack Identity est un service partagé d'autorisation, d'authentification et d'audit. Il fournit également des informations d'identité aux utilisateurs finaux. Le nom du projet pour ce service d'identité est Keystone. D'autres services OpenStack communiquent avec Keystone pour l'autorisation des requêtes des utilisateurs. Le service d'identité se compose principalement de trois composants : le serveur, les pilotes et les modules. Grâce à une interface RESTful, un serveur centralisé fournit des services d'authentification et d'autorisation. Les pilotes permettent d'accéder aux informations d'identité. Les modules reçoivent les requêtes de service, extraient les identifiants et communiquent avec notre serveur centralisé pour l'autorisation.

2.3.2 Glance (Image)

L'image du système d'exploitation de la machine virtuelle est un modèle permettant de créer de nouvelles instances dans le cloud. Elle peut contenir uniquement le système d'exploitation ou également des logiciels et applications installés. Des images peuvent être créées à partir d'un système d'exploitation en cours d'exécution et servir au déploiement de nouvelles machines virtuelles de ce type. Le service Glance d'OpenStack permet d'interroger les métadonnées des images du système d'exploitation et de les récupérer à l'aide d'API RESTful. Le service Glance accepte les requêtes des utilisateurs finaux ou des composants de calcul OpenStack via des requêtes API. Glance comprend cinq composants principaux : glance-api : le rôle de glance-api est de découvrir, de récupérer et de stocker les images en acceptant les appels d'API. glance-registry : le registre Glance fournit des informations sur les métadonnées des images, telles que le type, la taille, etc. Il est également responsable du stockage et du traitement des informations sur les métadonnées. Base de données : la base de données est utilisée pour stocker les métadonnées des images. Dans notre cas, il s'agit de MySQL. Référentiel de stockage pour les fichiers images : OpenStack prend en charge différents types de stockage pour les fichiers images. Les images peuvent être stockées dans un système de fichiers standard, des périphériques de stockage en mode bloc RADOS, un stockage d'objets, HTTP ou Amazon S3. Service de définition de métadonnées : Ce service est une API permettant de définir des métadonnées personnalisées. Il peut être utilisé par différents services, tels que les images, les saveurs, les volumes, etc. Il inclut une clé décrivant les attributs tels que le type, la description et les contraintes.

2.3.3 Nova (Compute)

Nova est l'un des services essentiels d'OpenStack. Il interagit avec l'hyperviseur pour lancer et activer des machines virtuelles accessibles au sein d'une entreprise et prises en charge par OpenStack. Il agit comme une interface entre OpenStack et l'hyperviseur. Dans notre configuration, les services Nova s'exécutent sur deux machines virtuelles. Ainsi, lorsque de nouvelles machines virtuelles sont créées, elles se trouvent à l'intérieur de ces machines virtuelles (un environnement de machines virtuelles imbriquées : des machines virtuelles dans des machines virtuelles). C'est pourquoi nous avons activé la virtualisation imbriquée sur ostack01. Toutes les ressources de calcul, comme la RAM et les processeurs virtuels, sont allouées par Nova depuis son hyperviseur local. Les principaux composants du service Nova sont les suivants : Nova-consoleauth : ce service assure l'authentification des consoles Nova. Il s'exécute sur le nœud contrôleur. Nova-scheduler : Nova utilise nova-scheduler pour déterminer le déploiement des requêtes d'instance. Le service de planification détermine sur quel hôte la machine virtuelle demandée doit être lancée. Le service de planification, en fonction des ressources demandées pour

la VM, sélectionne les hôtes physiques pouvant fournir ces ressources. Dans le cas contraire, une erreur est générée. Ce service s'exécute sur l'hôte contrôleur. Nova-conductor : Ce service assure la coordination et la prise en charge des requêtes de base de données pour Nova. Ce service s'exécute également depuis les hôtes contrôleurs. Nova-compute : Ce service gère tous les processus liés à la VM. Il est responsable de la création de l'image disque, de son lancement via le pilote de virtualisation sous-jacent, etc. Il s'exécute depuis le nœud de calcul.

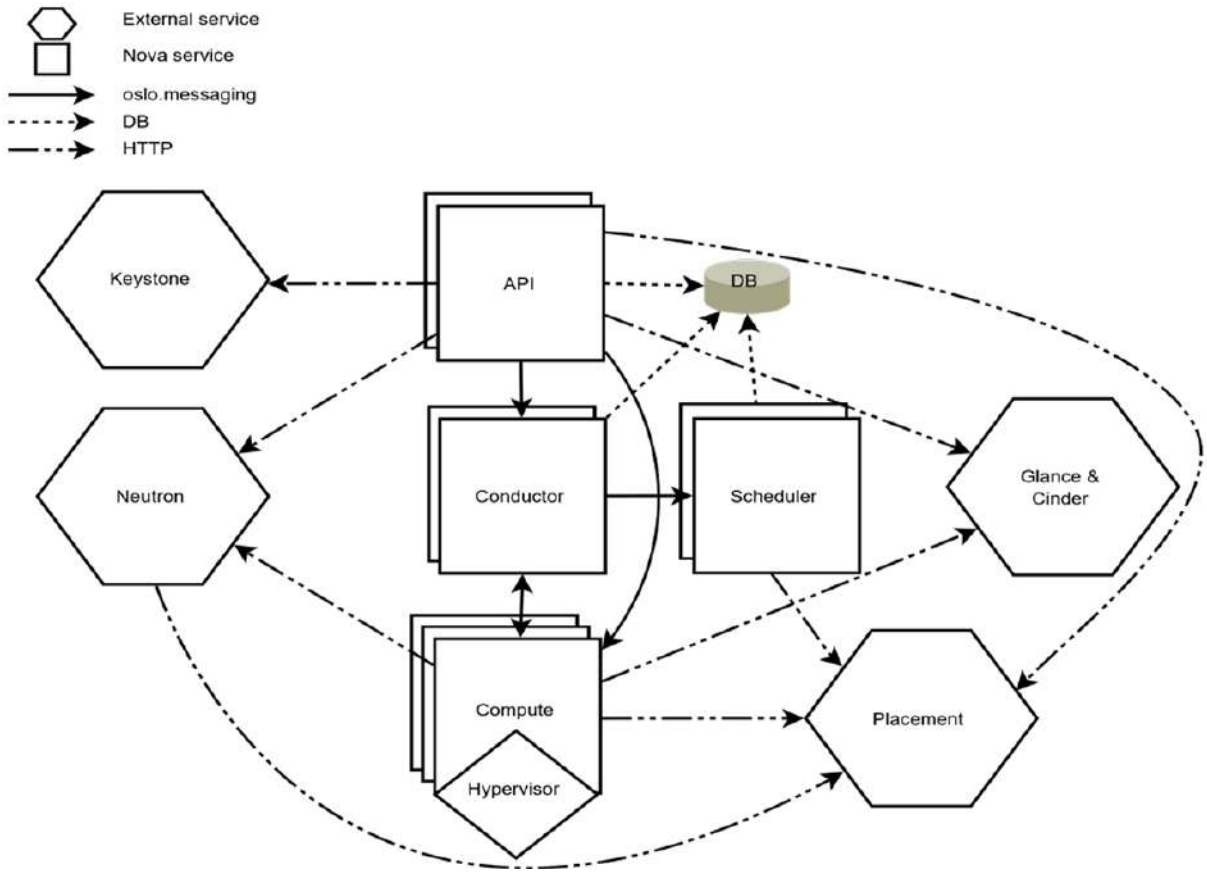


FIGURE 2 – Primary Nova Components and It's interaction with neutron, glance, keystone`https://docs.openstack.org/nova/latest/_images/architecture.svg`

2.3.4 Neutron (Networking)

Le service réseau d'OpenStack est appelé Neutron. Il fournit les composants réseau virtuels de l'environnement et crée une abstraction entre le réseau physique et le réseau virtuel. D'autres fonctionnalités réseau, telles que les services de pare-feu, le réseau privé virtuel (VPN), l'équilibrage de charge, etc., peuvent également être activées. Les machines virtuelles créées dans OpenStack peuvent accéder à des réseaux internes, externes ou aux deux. Un réseau externe est accessible depuis l'extérieur de l'installation OpenStack. Cependant, le réseau interne est accessible depuis l'installation OpenStack et se connecte directement aux machines virtuelles. Chaque routeur virtuel de l'environnement dispose d'une interface se connectant au réseau externe. Pour accéder aux machines virtuelles depuis Internet, des adresses IP externes doivent leur être attribuées. Pour contrôler l'accès réseau aux machines virtuelles, des groupes de sécurité leur sont attribués. Ces groupes de sécurité constituent un ensemble de règles de contrôle d'accès. Le réseau dans un environnement OpenStack peut être configuré de deux manières :

réseau fournisseur et réseau libre-service. Sur le réseau du fournisseur, les machines virtuelles sont uniquement connectées au réseau du fournisseur ou à un réseau externe. Dans une configuration réseau en libre-service avec réseau externe, les machines virtuelles peuvent disposer de réseaux privés et d'adresses IP flottantes qui assurent la connectivité aux machines virtuelles depuis le réseau externe. Les principaux composants de Neutron sont : Le serveur Neutron : il accepte les requêtes API et les achemine vers le plug-in réseau approprié pour effectuer des actions spécifiques. Les plug-ins et agents réseau OpenStack : l'agent de couche 3 (L3), l'agent DHCP et les agents de plug-in sont les agents courants de l'environnement OpenStack. La file d'attente de messagerie : elle assure le suivi des informations de routage entre le serveur Neutron et les différents plug-ins. Elle stocke également les informations d'état des plug-ins. Dans notre configuration OpenStack, nous avons dû configurer deux interfaces virtuelles pour la machine virtuelle contenant le service Neutron. Une interface était utilisée pour communiquer avec d'autres services OpenStack, et l'autre interface était reliée par NAT à l'interface KVM (hyperviseur). Cela permettait de fournir un accès Internet aux instances créées dans OpenStack. Neutron attribue une adresse IP interne aux instances, puis cette adresse est reliée par NAT à l'interface KVM, ce qui lui permet d'accéder à Internet.

2.3.5 Cinder (Block Storage)

Ce service fournit un stockage en mode bloc aux instances de machines virtuelles OpenStack. Le stockage est ajouté à la VM sous forme de disque séparé. Le service de stockage en mode bloc Cinder fournit des volumes pouvant être attachés à chaque instance de VM. Dans notre déploiement, nous avons ajouté un disque virtuel au groupe de volumes Cinder. La commande fdisk permet de transformer le disque virtuel en volume physique (PV). Ajoutez ensuite ce PV au groupe de volumes Cinder. Cinder permet également de créer des volumes amorçables. Les principaux composants de ce service sont les suivants : Cinder-scheduler : Comme nova-scheduler, ce service sélectionne l'hôte/nœud de stockage optimal pour créer le volume demandé. Ce service utilise plusieurs filtres pour sélectionner l'hôte. Il s'exécute depuis le nœud contrôleur. Cinder-volume : Ce service s'exécute sur le nœud de stockage et gère les volumes sur ce nœud.

2.3.6 Swift (Object Storage)

Swift est un stockage d'objets/blobs hautement disponible, distribué et cohérent. Les organisations peuvent utiliser Swift pour stocker de grandes quantités de données de manière efficace, sûre et économique. Conçu pour l'évolutivité, il est optimisé pour la durabilité, la disponibilité et la simultanéité sur l'ensemble des données. Swift est idéal pour stocker des données non structurées dont la croissance est illimitée.

2.3.7 Horizon (Dashboard)

Le service Horizon fournit un tableau de bord (interface web) permettant aux administrateurs cloud de gérer et de configurer les ressources cloud des utilisateurs. Ce service s'exécute sur le contrôleur.

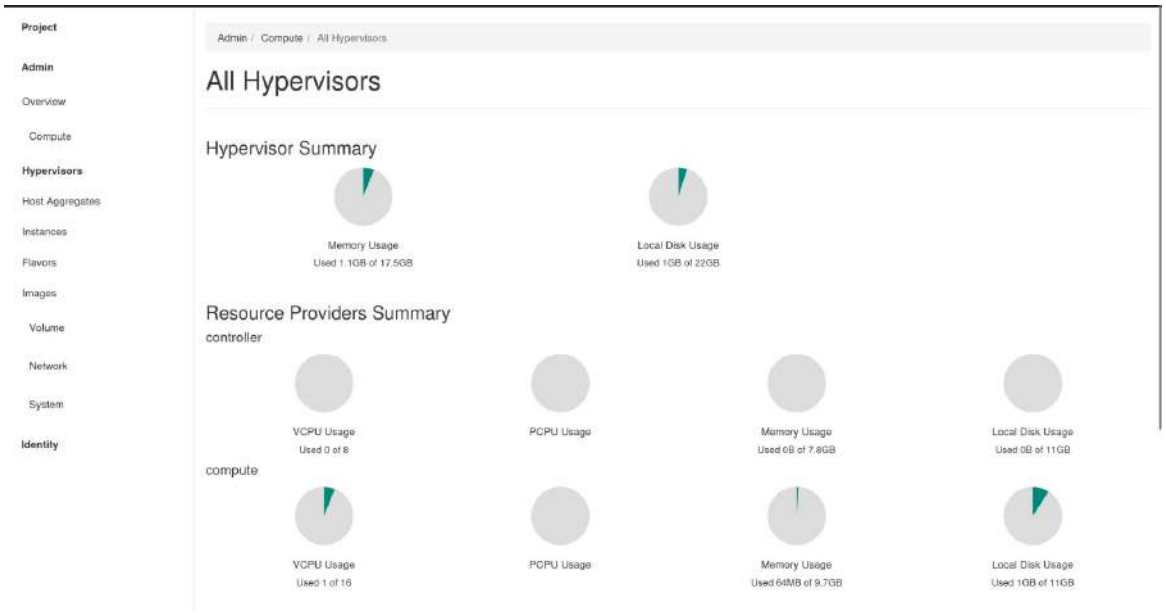


FIGURE 3 – OpenStack Dashboard showing resources summary

2.3.8 Heat (Orchestration)

Heat est le service d'orchestration d'OpenStack qui permet d'automatiser le déploiement et la gestion des ressources cloud. Au lieu de créer manuellement des machines virtuelles, des réseaux et des volumes de stockage, Heat utilise des modèles décrits en YAML ou JSON, appelés **HOT (Heat Orchestration Templates)**, pour définir l'infrastructure souhaitée.

Ces modèles sont exécutés par le moteur Heat, qui interagit avec les autres services OpenStack (Nova, Neutron, Cinder, Glance, Keystone) afin de provisionner les ressources de manière coordonnée. Un modèle déployé est appelé une **stack**, qui peut être mise à jour ou supprimée selon les besoins.

Heat offre plusieurs avantages :

— automatisation des déploiements,
— reproductibilité des environnements,
— scalabilité des ressources,
— réduction des erreurs humaines.

Par exemple, un seul template peut permettre de déployer un environnement complet composé de plusieurs machines virtuelles, réseaux et volumes en une seule opération. Cela rend Heat particulièrement utile dans un contexte académique comme celui de l'INPT, où des laboratoires cloud automatisés peuvent être rapidement mis en place pour soutenir l'apprentissage et la recherche.

3 Conception du Cloud Privé

3.1 Objectifs de la mise en place du cloud privé

L'objectif principal de ce projet est de concevoir et déployer une infrastructure de **cloud privé** basée sur OpenStack, afin de :

— Expérimenter les concepts du cloud computing dans un environnement pratique.

— Renforcer nos compétences techniques en administration système, virtualisation et orchestration.
— Fournir une plateforme flexible permettant de déployer des machines virtuelles et des services de manière centralisée.
— Contribuer au développement des ressources pédagogiques de l'INPT en offrant une solution qui pourrait être utilisée dans les travaux pratiques et projets de recherche.

3.2 Architecture proposée

3.2.1 Description des nœuds (Controller, Compute, Storage)

L'architecture proposée repose sur une répartition des rôles entre plusieurs nœuds :
— **contrôleur** : héberge les services centraux d'OpenStack tels que Keystone (authentification), Glance (images), Horizon (interface web), ainsi que les bases de données et la file de messages.

The controller node runs the Identity service, Image service, Placement service, management portions of Compute, management portion of Networking, various Networking agents, and the Dashboard. It also includes supporting services such as an SQL database, message queue, and NTP.

En option, le nœud contrôleur peut faire tourner des parties de services de Stockage par Blocs, de Stockage Objet, d'Orchestration et de Télémétrie.

Le nœud contrôleur nécessite au minimum deux interfaces réseau.
— **Compute** : Le nœud compute exécute la partie hyperviseur de Compute qui fait fonctionner les instances. par défaut, Compute utilise l'hyperviseur KVM. Le nœud compute héberge également un agent du service Réseau qui connecte les instances aux réseaux virtuels et fournit des services de firewall aux instances via les groupes de sécurité.

Vous pouvez déployer plus d'un nœud compute. Chaque nœud nécessite au minimum deux interfaces réseau.
— **stockage** :
  — **Stockage par blocs (Cinder)** : fournit des volumes persistants pouvant être attachés aux instances comme des disques durs virtuels. Utilisé notamment pour héberger des bases de données ou des systèmes de fichiers partagés. Dans un déploiement simplifié, le trafic de stockage transite par le réseau de management, mais en production un réseau dédié est recommandé pour améliorer la performance et la sécurité.
  — **Stockage objet (Swift)** : permet de stocker et gérer des données non structurées (comptes, conteneurs, objets) accessibles via une API REST. Ce service nécessite au minimum deux nœuds pour garantir la redondance et la fiabilité. Il est adapté à l'archivage et à la gestion de grandes quantités de données.

3.2.2 Réseau

Réseau Option 1 : Réseaux fournisseurs Cette option déploie le service Réseau d'OpenStack de la manière la plus simple possible avec essentiellement des services de couche-2 (bridging/switching) et une segmentation en VLAN. Elle établit un pont (*bridge*) entre les réseaux virtuels et physiques, en s'appuyant sur l'infrastructure réseau existante pour les services de couche-3 (routage). Un service DHCP fournit les adresses IP aux instances.

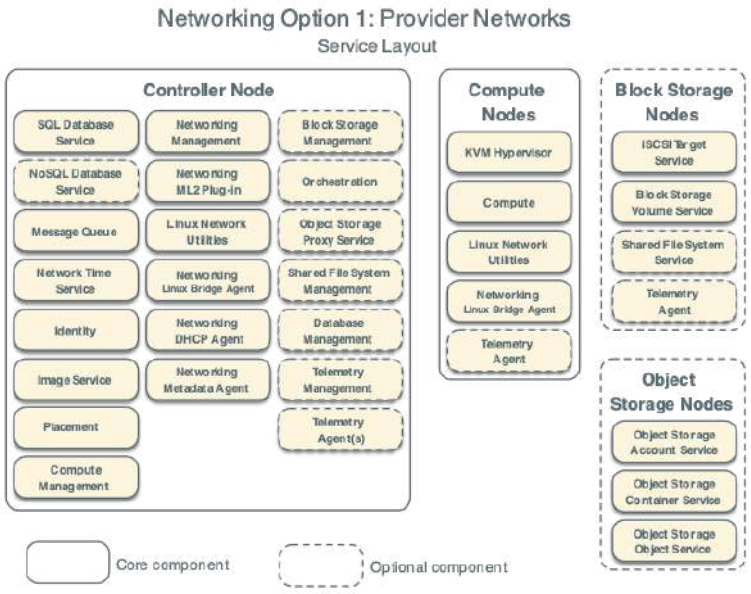


FIGURE 4 – Provider-network

Cependant, cette approche présente certaines limites : elle ne prend pas en charge les réseaux privés (*self-service*), les services de couche-3 natifs ni les services avancés comme LBaaS (Load Balancer as a Service) et FWaaS (Firewall as a Service).

Réseau Option 2 : Réseaux libre-service Cette option améliore l'approche précédente en ajoutant des services de couche-3 (routage) permettant la création de réseaux privés par les utilisateurs. Elle s'appuie sur des techniques de segmentation overlay telles que VXLAN et permet le routage vers les réseaux physiques via le NAT. Elle constitue également la base de services avancés comme LBaaS et FWaaS.

Contrairement à l'option fournisseurs, les utilisateurs peuvent créer leurs propres réseaux virtuels sans connaître les détails de l'infrastructure réseau sous-jacente.

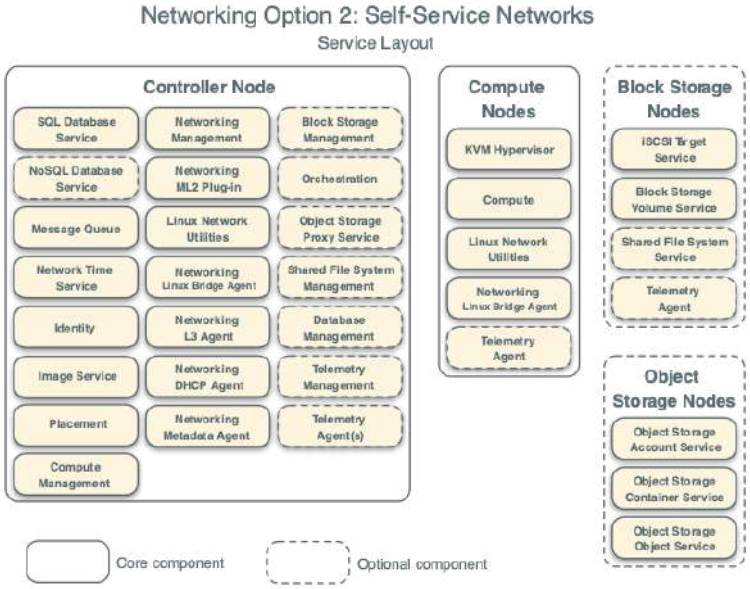


FIGURE 5 – self-service network

Conclusion Pour ce projet, nous avons choisi d'utiliser l'option **Réseaux fournisseurs (Provider Networks)**. Ce choix s'explique par sa simplicité de mise en œuvre et son adéquation avec les objectifs du projet, qui visent avant tout à mettre en place un cloud privé fonctionnel et pédagogique sans complexité excessive liée aux services avancés.

3.2.3 Outils et technologies utilisés

La mise en place d'un cloud privé avec OpenStack nécessite un ensemble d'outils et de technologies couvrant toutes les étapes du projet, de la virtualisation de l'environnement jusqu'à l'orchestration des ressources. Les principaux éléments utilisés sont détaillés ci-dessous :

- **Système d'exploitation** : L'infrastructure repose sur **Ubuntu Server (20.04 LTS)** en raison de sa stabilité, de sa compatibilité avec OpenStack et de sa large communauté de support. D'autres distributions compatibles (comme CentOS ou Debian) existent, mais Ubuntu est privilégié pour sa simplicité d'intégration.
- **Hyperviseur** : L'hyperviseur utilisé est **KVM (Kernel-based Virtual Machine)**, intégré au noyau Linux. Il permet la virtualisation matérielle et offre de bonnes performances pour l'exécution des instances. Libvirt est utilisé comme couche d'abstraction pour gérer les machines virtuelles.
- **Environnement de virtualisation** : Afin de simuler l'infrastructure matérielle nécessaire (contrôleur, compute nodes, stockage), des solutions comme **VirtualBox** ou **VMware Workstation/ESXi** sont employées. Elles permettent de créer des machines virtuelles hôtes représentant les différents nœuds OpenStack.
- **Services principaux d'OpenStack** : L'architecture repose sur une installation modulaire comprenant :
  - **Keystone** : gestion des identités et authentification.
  - **Glance** : gestion des images disques des instances.
  - **Nova** : service de calcul responsable du cycle de vie des machines virtuelles.
  - **Neutron** : service réseau permettant la gestion des adresses IP, VLAN, VXLAN et

routage.
- **Cinder** : stockage bloc persistant.
- **Swift** : stockage objet distribué.
- **Horizon** : tableau de bord web pour l'administration et l'utilisation d'OpenStack.

— **Bases de données et files de messages** : OpenStack s'appuie sur des services auxiliaires :
- **MariaDB/MySQL** : stockage des données de configuration.
- **RabbitMQ** : gestion de la communication entre les différents services.
- **Memcached** : optimisation des performances via la mise en cache.

— **Réseaux et adressage** : Le réseau est segmenté en trois parties principales :
- Réseau de gestion (*Management network*) : communication interne entre les services OpenStack.
- Réseau de données (*Tenant/VM network*) : connectivité des instances.
- Réseau externe (*External/Public network*) : accès Internet et attribution des IP flottantes.

3.3 Schéma global de l'infrastructure

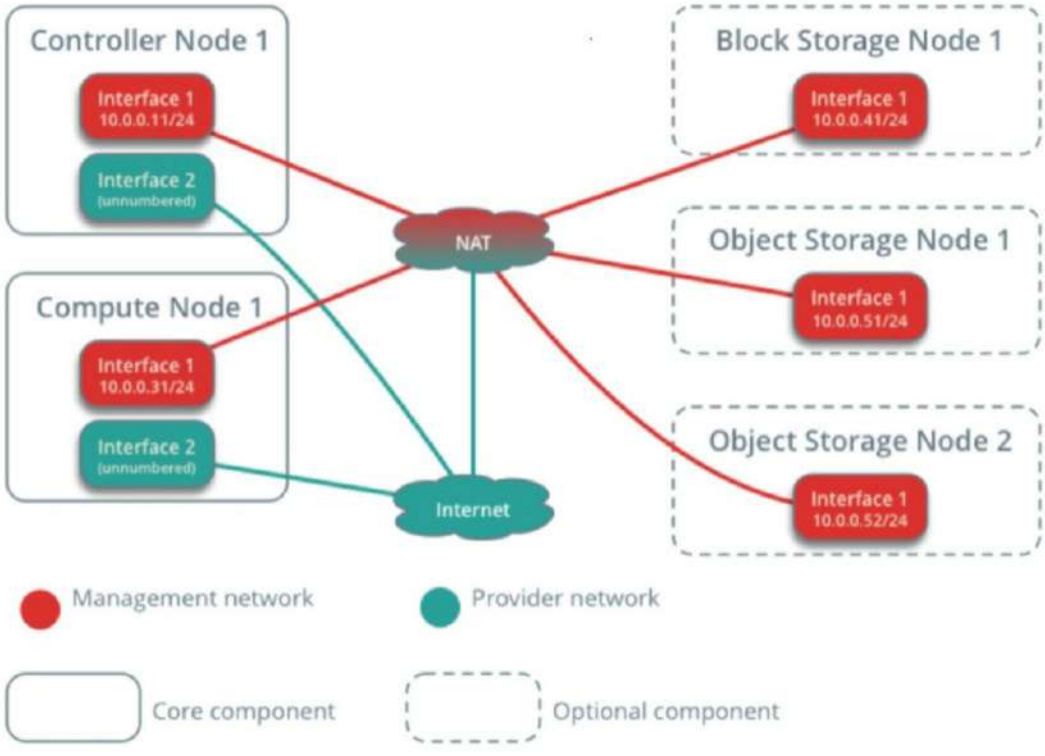


FIGURE 6 – Schéma global de l'infrastructure

4 Mise en Œuvre Pratique

4.1 Introduction

Nous nous intéressons dans ce chapitre, aux principales ´étapes d'installation d'OpenStack version Epoxy . Nous allons commencer par la préparation du système, sur lequel sera implémentée cette solution, et ceci par la configuration réseau, ainsi que l'installation des différents paquets requis pour le bon fonctionnement des différents services et composants. Ensuite on passe à l'installation et la configuration de Keystone, Glance, Nova etc. Nous finalisons par la mise en œuvre du Dashboard qui est l'interface graphique et l'outil de gestion d'OpenStack et une présentation d'un exemple de création d'une instance de machine virtuelle et différentes opérations possibles sur cette dernière.

4.2 La mise en place de la plateforme Openstack

4.2.1 Types d'installation d'Openstack

Selon la documentation officielle dOpenstack, il y a plusieurs architectures de déploiement de cette technologie, le choix se fait selon les besoins d'utilisation, les ressources disponibles et les exigences des entreprises et utilisateurs. Le schéma ci-dessous montre les différentes méthodes existantes :

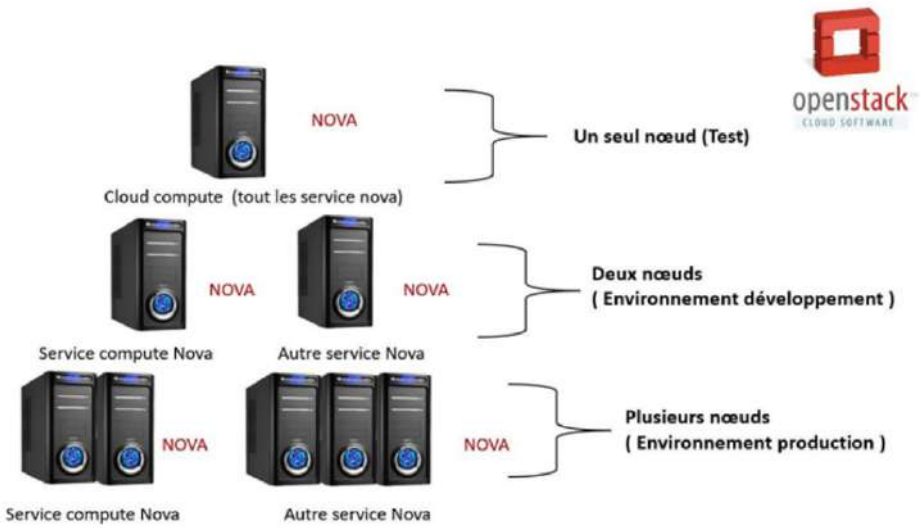


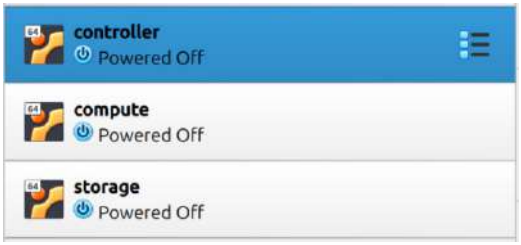
FIGURE 7 – Les différentes architectures possibles.

Comme on le constate dans la figure, openstack peut s'installer de deux manières différentes :

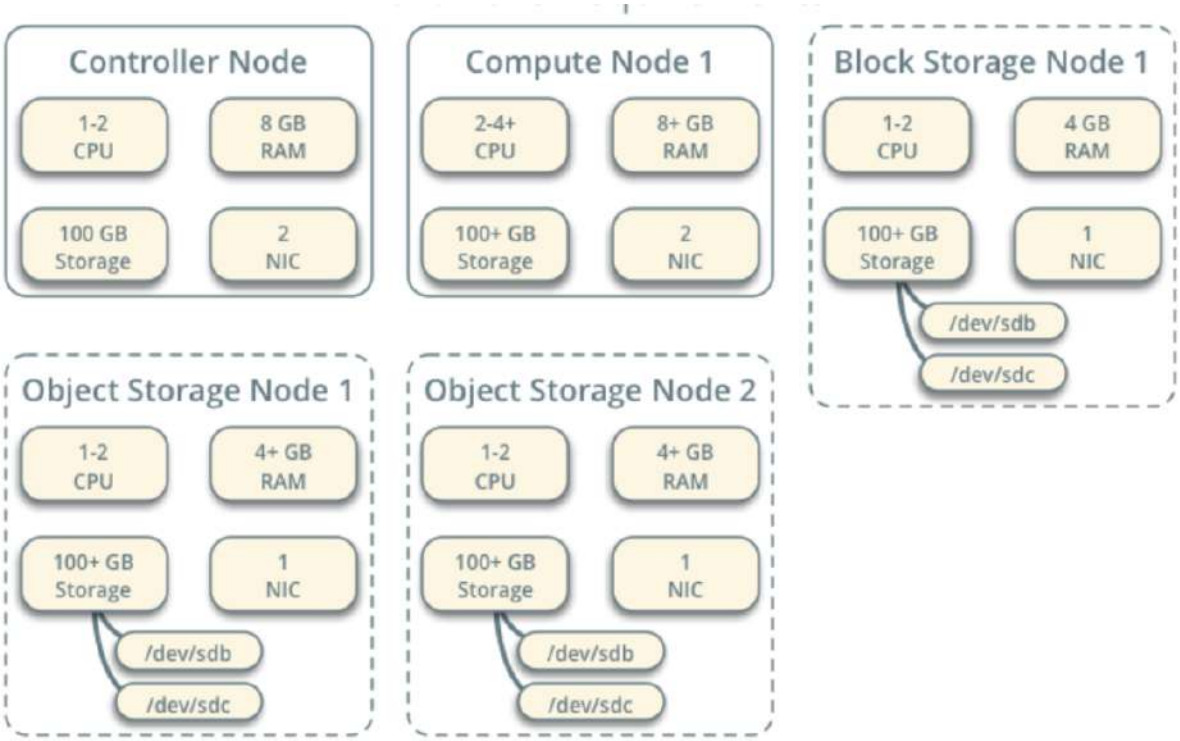
Single Node (un seul nœud) : dans cette configuration, tous les services sont installés et exécutés sur un système unique. Ce type de déploiement est adapté à des fins d'évaluation. Un tel déploiement n'est cependant pas adapté pour une utilisation dans un environnement de production.

Multi Node (plusieurs nœuds) : les composants d'Openstack seront installés sur des noeuds différents.

NB : Openstack propose aussi plusieurs types d'installation ; via des scripts ou bien via des packages etc. ∗ Dans notre cas, nous allons installer Openstack sur Multi Node (plusieurs nœuds) controller,compute, storage



4.2.2 Les prérequis



4.2.3 Outils

Hyperviseur VirtualBox 7.2.0 : Au lieu d'installer OpenStack directement sur la machine hôte, le logiciel de virtualisation VirtualBox sera utilisé pour créer une machine virtuelle, sur laquelle seront installés les composants d'OpenStack, et cela pour créer le nombre d'interfaces d´désirées, l'installation d'OpenStack sur une machine virtuelle a aussi pour objectif d'isoler la machine hôte et la machine virtuelle pour ne pas affecter le reste de notre environnement de travail.

FIGURE 8 – Logo du virtualBox.

Système d'exploitation Ubuntu 24.04.3 LTS Server : Nous avons choisi d'installer la version 24.04 LTS d'Ubuntu car cette dernière est la plus adéquate à utiliser avec OpenStack et ne souffre pas de problèmes de compatibilité par rapport aux autres versions.

FIGURE 9 – Logo du Ubuntu.

Version d'Openstack : Nous avons choisis de configurer la version 2025.1 'Epoxy' d'OpenStack. Notre choix s'est porté sur cette version car c'est la dernière version jusqu' à la réalisation de ce projet, qui est améliorée de fonctionnalités par rapport aux versions qui la précède.

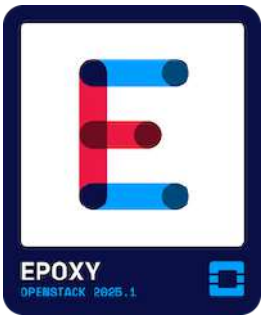


FIGURE 10 – Logo du OpenStack 2025.1 'Epoxy'

4.3 Installation des paquets de base et préparation du système

4.3.1 Configuration de la carte réseau

La version 24.04 d'Ubuntu utilise netplan pour la gestion des interfaces réseau donc remplace **/etc/network/** interfaces qui est utilisé précédemment.

— Pour pouvoir configurer les cartes réseau nous ouvrons le fichier YAML qui se trouve à l'emplacement **/etc/netplan** et le modifier selon la configuration souhaitée :

```
network:
  version: 2
  ethernets:
    enp0s3:   # Host-only (management)
      dhcp4: false
      addresses:
        - 192.168.56.10/24
          #gateway4: 192.168.56.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]

    enp0s8:   # Bridged (external/provider)
      dhcp4: false
      optional: true

    enp0s9:
      dhcp4: true
```

— Enfin, nous exécutons la commande ci-dessous pour prendre en considération la modification effectuée :

```
root@controller:/etc/netplan# netplan apply
```

Dans notre architecture, nous avons attribué l'adresse IP comme suit : 192.168.56.10/24 pour le nœud appelé "controller".et en repete le meme proccesus pour les deux autre noeds (storage,compute) pour obtenir :

```
127.0.0.1 localhost
#127.0.1.1 compute
192.168.56.10 controller
192.168.56.20 compute
192.168.56.30 storage
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

— Pour vérifier l'accès à internet, on ping Google.com avec 4 paquets comme suit :

```
root@controller:/home/openstack# ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=77.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=34.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=52.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=35.4 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3012ms
rtt min/avg/max/mdev = 34.803/49.952/77.099/17.199 ms
```

4.3.2 Mise à jour du système

— On va mettre à niveau les packages d'Ubuntu avec la commande suivante :

```
root@controller:~# apt update && dist-apt upgrade
```

4.3.3 Installation de l'outil de base de données MySql « mariadb »

— On commence par installer les paquets :

```
root@controller:~# apt install mariadb-server python-pymysql
```

— Ensuite on crée un fichier **/etc/mysql/mariadb.conf.d/99-openstack.cnf** et on ajoute par la suite les configurations suivantes :

```
[mysqld]
bind-address = 192.168.56.10

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

— Enfin, on redémarre le service MySql :

```
root@controller:~# service mysql restart
```

4.3.4 Gestion de fil d'attente, installation du service RabbitMQ

Utilisé pour la coordination des opérations et des informations d'état entre les services. Openstack prend en charge plusieurs services de file d'attente de messages, notamment RabbitMQ , Qpid et ZeroMQ . Dans cette installation, on a implémenté RabbitMQ.

— Installation de paquets :

```
root@controller:~# apt install rabbitmq-server
```

— Ajouter l'utilisateur openstack :

```
root@controller:~# rabbitmqctl add_user openstack RABBIT_PASS
```

— Autorisez les utilisateurs à configurer, lire et écrire sur openstack :

```
root@controller:~# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/"
```

4.3.5 Installer le service d'identité memcached

Memcached est utilisé par les services d'identité, c'est un système d'usage général servant à gérer la mémoire cache distribuée. Il est souvent utilisé pour augmenter la vitesse de réponse des sites web créés à partir de bases de données. Il gère les données et les objets en RAM de façon à réduire le nombre de fois qu'une même donnée est stockée dans un périphérique externe.

— Installer les packages du service :

```
root@controller:~# apt install memcached python-memcache
```

— Ouvrir le fichier **/etc/memcached.conf** pour configurer le service à utiliser l'adresse IP de notre système :

```
# This parameter is one of the only security measures that memcached has, so make s
ure
# it's listening on a firewalled interface.
-l 192.168.56.10
```

— Redémarrer le service avec la commande :

```
root@controller:~# service memcached restart
```

4.3.6 Installation du service ETCD

ETCD (Equipement Terminal de Circuit de Données) est un magasin de données clé- valeur Open Source cohérent et distribué, qui stocke la configuration de systèmes ou clusters de machines distribuées, coordonne leur planification et assure la découverte des services. ETCD facilite et sécurise les mises à jour automatiques, coordonne la planification des tâches affectées aux hôtes et aide à la mise en place d'un réseau pour les conteneurs.

— Installer les paquets :

```
root@controller:~# apt install etcd
```

— Ouvrir le fichier /etc/default/etcd et effectuer les modifications suivantes :

```
## etcd(1) daemon options
## See "/usr/share/doc/etcd-server/op-guide/configuration.md.gz"
## for available options.
##
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://192.168.56.10:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://192.168.56.10:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://192.168.56.10:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://192.168.56.10:2379"

## Use environment to override, for example: ETCD_NAME=default
```

— Enfin, on redémarre les services :

```
root@controller:~# service etcd restart
```

4.3.7 Installation du protocole NTP « chrony »

NTP (Network Time Protocol), utilisé pour synchroniser correctement via un réseau informatique, l'horloge locale d'ordinateurs ainsi que les services entre plusieurs machines si nécessaire, dans ce projet on a installé chrony une implémentation du NTP.

— Installer les paquets :

```
root@controller:~# apt install chrony
```

— Editer le fichier /etc/chrony/chrony.conf et ajouter la ligne allow 192.168.56.10/24 (la description du sous-réseau) :

```
# About using servers from the NTP Pool Project in general see (LP: #104525).
# Approved by Ubuntu Technical Board on 2011-02-08.
# See http://www.pool.ntp.org/join.html for more information.
#pool ntp.ubuntu.com        iburst maxsources 4
#pool 0.ubuntu.pool.ntp.org iburst maxsources 1
#pool 1.ubuntu.pool.ntp.org iburst maxsources 1
#pool 2.ubuntu.pool.ntp.org iburst maxsources 2

server controller iburst
allow 192.168.56.10/24

# Use time sources from DHCP.
sourcedir /run/chrony-dhcp
```

— Redémarrer le service NTP :

```
root@controller:~# service chrony restart
```

4.3.8 Installer et configurer les composants OpenStack

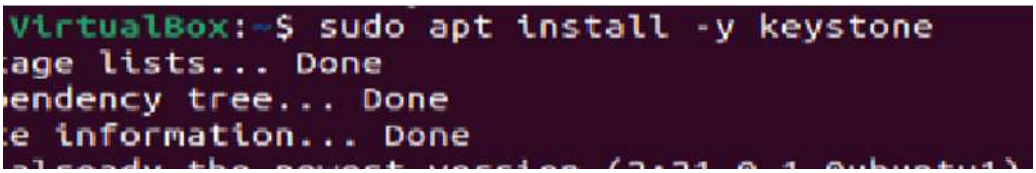
— **Keystone (Service d'identité)** Service d'authentification centralisé d'OpenStack qui gère les utilisateurs, projets, rôles et tokens d'accès aux autres services.

— **Prérequis :**

1. Se connecter à MySQL
2. Créer la base de données `keystone`
3. Créer les utilisateurs et attribuer les privilèges
4. Appliquer les changements

— **Installation et configuration des composants**

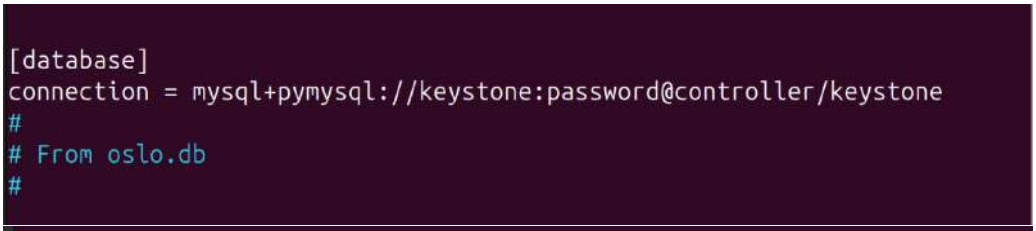
1. Installer Keystone



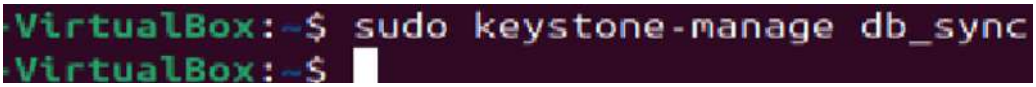
2. Configurer Keystone

   Modifier le fichier **/etc/keystone/keystone.conf** :

   Remplacez **KEYSTONE-BY-PASS** par le mot de passe que vous avez choisi pour la base de données.



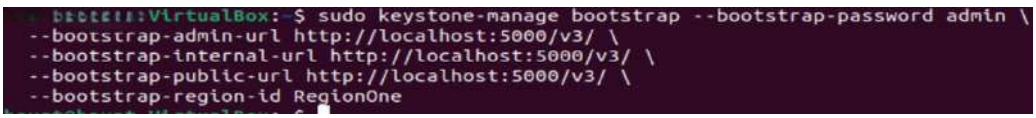
3. Peupler la base de données du service d'identité



4. Initialiser les dépôts de clés Fernet

5. Démarrer le service d'identité

   Remplacez **ADMIN-PASS** par un mot de passe approprié pour un utilisateur administratif.



6. Configurer le serveur Apache HTTP

   Modifier le fichier **/etc/apache2/apache2.conf :**

```
VirtualBox:~$ sudo nano /etc/apache2/apache2.conf
VirtualBox:~$
```

ServerName localhost

```
 GNU nano 6.2                  /etc/apache2/apache2.conf

# Global configuration
#
ServerName localhost
#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the Mutex documentation (available
# at <URL:http://httpd.apache.org/docs/2.4/mod/core.html#mutex>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
#ServerRoot "/etc/apache2"

#
# The accept serialization lock file MUST BE STORED ON A LOCAL DISK.
#

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Activer les services Apache et WSGI :

```
VirtualBox:~$ sudo service apache2 restart
VirtualBox:~$
```

7. Configurer le compte administrateur en définissant les variables d'environnement appropriées :

   Remplacez **ADMIN-PASS** par le mot de passe utilisé dans le **keystone-manage bootstrap.**

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

— Créer un domaine, des projets, des utilisateurs et des rôles

Le service d'identité fournit des services d'authentification pour chaque service OpenStack. Le service d'authentification utilise une combinaison de domaines, de projets, d'utilisateurs et de rôles.

(a) Bien que le domaine "default" existe déjà à partir de l'étape keystone-manage bootstrap dans ce guide, une méthode formelle pour créer un nouveau domaine serait la suivante :

```
root@controller:/home/openstack# openstack domain create --description "An Example
Domain" example
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | An Example Domain                |
| enabled     | True                             |
| id          | dd75af203d3e4182b6def6d6d9aa280a |
| name        | example                          |
| options     | {}                               |
| tags        | []                               |
+-------------+----------------------------------+
```

(b) Ce guide utilise un projet de service qui contient un utilisateur unique pour chaque service ajouté à l'environnement. Création du projet de service :

```
root@controller:/home/openstack# openstack project create --domain default \
  --description "Service Project" service
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Service Project                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | dc60b5b3280b461796598d6e9897fc65 |
| is_domain   | False                            |
| name        | service                          |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

(c) Les tâches régulières (non-administratives) doivent utiliser un projet et un utilisateur non privilégiés. À titre d'exemple, ce guide crée le projet myproject et l'utilisateur myuser.

```
root@controller:/home/openstack# openstack project create --domain default \
  --description "Demo Project" myproject
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | Demo Project                     |
| domain_id   | default                          |
| enabled     | True                             |
| id          | 04f979a3fa264957ac741710dd69a3dd |
| is_domain   | False                            |
| name        | myproject                        |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

```
root@controller:/home/openstack# openstack user create --domain default \
  --password-prompt myuser
User Password:
Repeat User Password:
No password was supplied, authentication will fail when a user does not have a pass
word.
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| default_project_id  | None                             |
| domain_id           | default                          |
| email               | None                             |
| enabled             | True                             |
| id                  | 72edb22ab4864e55948288452c551520 |
| name                | myuser                           |
| description         | None                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

```
root@controller:/home/openstack# openstack role create myrole
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| id          | 04cd1a1f1848444bb450b759f8e05b74 |
| name        | myrole                           |
| domain_id   | None                             |
| description | None                             |
+-------------+----------------------------------+
```

```
root@controller:/home/openstack# openstack role add --project myproject --user myus
er myrole
```

```
root@controller:/home/openstack# . admin-openrc
root@controller:/home/openstack# openstack token issue
+------------+-----------------------------------------------------------------------------------------+
| Field      | Value                                                                                   |
+------------+-----------------------------------------------------------------------------------------+
| expires    | 2025-09-29T13:12:00+0000                                                                |
| id         | gAAAAABo2neQtAT42UU7Fg1Ll5tWgWcnS282EzX2MPwEkT5DgbW1E93OBDm1FeRRkEC1mT4mLhf_IZB-        |
|            | oPVgU1AM8NGFFl3FmoDoxSv1vjfp6jhz4yN6lOpBS3cqlENSV2__09R-5ps5XDHXnd9j0dzwJ2Jrm--        |
|            | S6UmAwSIWmNiaZ1xSz08Hw8Q                                                                |
| project_id | 8dfb027d0c304b67a2f7313960cbc57b                                                        |
| user_id    | f39da39463ca463ba0386af26861d223                                                        |
+------------+-----------------------------------------------------------------------------------------+
```

— **Glance (Service d'image)** Service de gestion des images permettant de stocker, partager et récupérer des images de disques pour les machines virtuelles.

— **Prérequis**

1. Se connecter à MySQL
2. Créer le base de donnée glance
3. Créer les utilisateurs et attribuer les privilèges
4. Appliquer les changements

— **Installation et configuration des composants**

1. Installer Glance

```
root@controller:/home/openstack# apt install glance
Reading package lists... Done
Building dependency tree... Done
```

2. Configurer Glance

   Modifier le fichier **/etc/glance/glance-api.conf :**

```
-VirtualBox:~$ sudo nano /etc/glance/glance-api.conf
```

Remplacez **GLANCE-DBPASS** par le mot de passe que vous avez choisi pour la base de données du service Image.

```
 GNU nano 6.2                  /etc/glance/glance-api.conf

[database]
#connection = sqlite:////var/lib/glance/glance.sqlite
backend = sqlalchemy
connection = mysql+pymysql://glance:hayatips@localhost/glance
#
# From oslo.db
#

# If True, SQLite uses synchronous mode. (boolean value)
#sqlite_synchronous = true

# The back end to use for the database. (string value)
# Deprecated group/name - [DEFAULT]/db_backend
#backend = sqlalchemy

# The SQLAlchemy connection string to use to connect to the database. (string
# value)

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Remplacez **GLANCE-PASS** par le mot de passe que vous avez choisi pour l'utilisateur glance dans le service Identity.

- **www_authenticate_uri = http ://controller :5000**
  - URL que les services utilisent pour s'authentifier auprès de Keystone.
  - Généralement utilisé pour générer des réponses HTTP 401 avec une URL d'authentification.
- **auth_url = http ://controller :5000/v3**
  - URL pour communiquer avec le service d'identité Keystone (endpoint).
  - Les utilisateurs et services l'utilisent pour s'authentifier et obtenir des tokens.
- **memcached_servers = controller :11211**
  - Adresse du serveur `Memcached` utilisé pour le caching des tokens d'authentification.
  - Optimise les performances en réduisant les requêtes répétées vers Keystone.
- **auth_type = password**
  - Type de méthode d'authentification utilisée.
  - Ici, il s'agit d'une authentification par mot de passe.
- **project_domain_name = Default**
  - Domaine auquel appartient le projet spécifié.
  - Généralement, `Default` est le domaine par défaut dans Keystone.
- **user_domain_name = Default**
  - Domaine auquel appartient l'utilisateur.
  - Souvent défini sur `Default` pour des configurations de base.
- **project_name = service**
  - Nom du projet dans lequel les services s'exécutent.

— `service` est couramment utilisé pour les comptes de services dans OpenStack.

— **username = glance**

— Nom d'utilisateur du service OpenStack en question (ici `Glance`, le service d'images).

```
[keystone_authtoken]
www_authenticate_uri  = http://controller:5000
auth_url = http://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = password
```

```
 GNU nano 6.2                  /etc/glance/glance-api.conf
[paste_deploy]
flavor = keystone
#
# From glance.api
#


#
# Deployment flavor to use in the server application pipeline.
#
# Provide a string value representing the appropriate deployment
# flavor used in the server application pipeline. This is typically
# the partial name of a pipeline in the paste configuration file with
# the service name removed.
#
# For example, if your paste section name in the paste configuration
# file is [pipeline:glance-api-keystone], set ``flavor`` to
# ``keystone``.
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify
```

```
 GNU nano 6.2                  /etc/glance/glance-api.conf
[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
#
# From glance.multi_store
#


#
# The store identifier for the default backend in which data will be
# stored.
#
# The value must be defined as one of the keys in the dict defined
# by the ``enabled_backends`` configuration option in the DEFAULT
# configuration group.
#
# If a value is not defined for this option:
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify
```

3. Peupler la base de données du service d'image

```
VirtualBox:~$ sudo glance-manage db_sync
```

```
mysql> use glance;
Reading table information for completion of table and column name
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------------------+
| Tables_in_glance                 |
+----------------------------------+
| alembic_version                  |
| image_locations                  |
| image_members                    |
| image_properties                 |
| image_tags                       |
| images                           |
| metadef_namespace_resource_types |
| metadef_namespaces               |
| metadef_objects                  |
| metadef_properties               |
| metadef_resource_types           |
| metadef_tags                     |
| task_info                        |
| tasks                            |
+----------------------------------+
14 rows in set (0.00 sec)
```

4. Redémarrer le service Glance

```
VirtualBox:~$ sudo service glance-api restart
```

— **Placement (Service de gestion de l'allocation des ressources)** : est un service qui permet de gérer l'allocation des ressources dans un environnement OpenStack, en fournissant une API pour la gestion des ressources physiques disponibles (comme les CPU, la mémoire et le stockage) pour les instances de machines virtuelles. Il permet de planifier où déployer les instances en fonction des ressources disponibles et des contraintes spécifiques de placement.

— **Vérification de l'installation**

```
root@controller:/home/openstack# placement-status upgrade check
+----------------------------------------+
| Upgrade Check Results                  |
+----------------------------------------+
| Check: Missing Root Provider IDs       |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Incomplete Consumers            |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
| Check: Policy File JSON to YAML Migration |
| Result: Success                        |
| Details: None                          |
+----------------------------------------+
```

— **Nova (Service de calcul)** : Nova est le composant responsable du déploiement et de la gestion des machines virtuelles dans OpenStack. Il interagit avec l'hyperviseur (comme KVM) pour créer, exécuter et gérer les instances. Nova travaille en étroite collaboration avec Glance (images), Neutron (réseau) et Placement (allocation des ressources) pour assurer le cycle de vie complet des VM.

— **Vérification de l'installation**

```
root@controller:/home/openstack# . admin-openrc
root@controller:/home/openstack# openstack compute service list
+----------+----------+----------+----------+---------+-------+------------+
| ID       | Binary   | Host     | Zone     | Status  | State | Updated At |
+----------+----------+----------+----------+---------+-------+------------+
| 74082c4f | nova-sch | controll | internal | enabled | up    | 2025-09-   |
| -1897-   | eduler   | er       |          |         |       | 29T11:57:2 |
| 4d76-    |          |          |          |         |       | 0.000000   |
| 9b8b-    |          |          |          |         |       |            |
| f289351b |          |          |          |         |       |            |
| 731c     |          |          |          |         |       |            |
| a9caea40 | nova-con | controll | internal | enabled | up    | 2025-09-   |
| -b5a1-   | ductor   | er       |          |         |       | 29T11:57:2 |
| 44f7-    |          |          |          |         |       | 2.000000   |
| a9aa-    |          |          |          |         |       |            |
| 06d9aa0b |          |          |          |         |       |            |
| 1206     |          |          |          |         |       |            |
| ff266d07 | nova-    | controll | nova     | enabled | up    | 2025-09-   |
| -dbea-   | compute  | er       |          |         |       | 29T11:57:1 |
| 4f60-    |          |          |          |         |       | 7.000000   |
| a323-    |          |          |          |         |       |            |
| 9d13028f |          |          |          |         |       |            |
| 16ee     |          |          |          |         |       |            |
| e4e6c8af | nova-    | compute  | nova     | enabled | up    | 2025-09-   |
| -9f5b-   | compute  |          |          |         |       | 29T11:57:1 |
| 4b2e-    |          |          |          |         |       | 9.000000   |
| aa49-    |          |          |          |         |       |            |
| 40628ca3 |          |          |          |         |       |            |
| 34eb     |          |          |          |         |       |            |
+----------+----------+----------+----------+---------+-------+------------+
```

```
root@controller:/home/openstack# openstack image list
+--------------------------------------+---------------------+--------+
| ID                                   | Name                | Status |
+--------------------------------------+---------------------+--------+
| 8464d493-4b48-4d3b-99da-a9723986d3ab | Ubuntu-24.04-Noble | active |
| a1f1a7c1-959e-4a33-ac9e-7acf630f030d | cirros              | active |
+--------------------------------------+---------------------+--------+
```

```
root@controller:/home/openstack# nova-status upgrade check
+-----------------------------------------+
| Upgrade Check Results                   |
+-----------------------------------------+
| Check: Cells v2                         |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: Placement API                    |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: Cinder API                       |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: Policy File JSON to YAML Migration |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: Older than N-1 computes          |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: hw_machine_type unset            |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
| Check: Service User Token Configuration |
| Result: Success                         |
| Details: None                           |
+-----------------------------------------+
```

— **Neutron (Service de réseau)** : Neutron fournit les fonctionnalités réseau dans OpenStack. Il permet la création et la gestion des réseaux virtuels, sous-réseaux, routeurs et adresses IP. Grâce à Neutron, les instances peuvent communiquer entre elles et accéder à Internet. Il offre également des services avancés comme le DHCP, la sécurité via les groupes de sécurité et la gestion des IP flottantes.

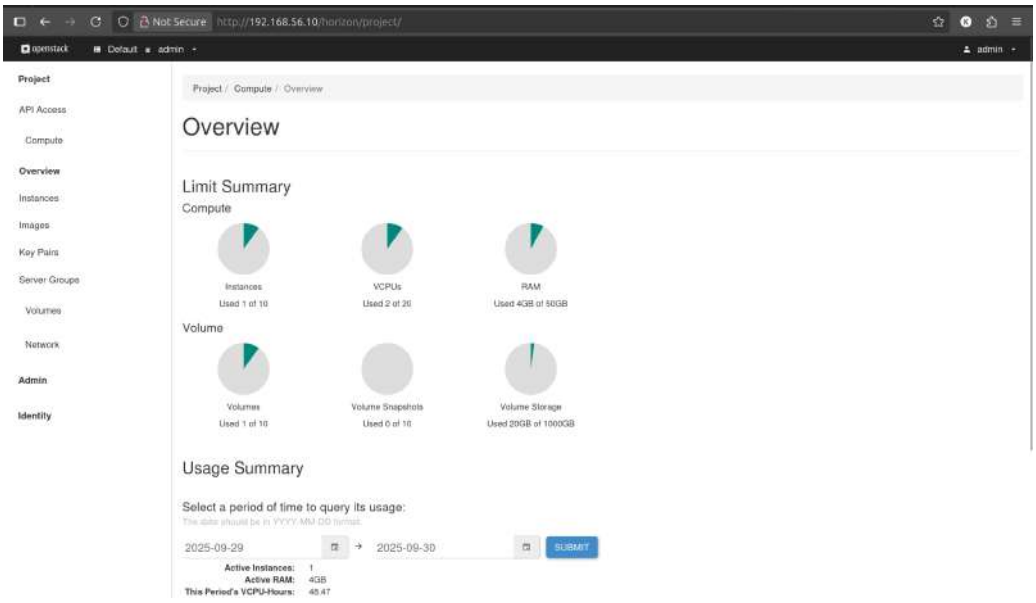
— **Vérification de l'installation**

```
root@controller:/home/openstack# openstack network agent list
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| ID                                   | Agent Type         | Host       | Availability Zone | Alive | State | Binary                    |
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
| 1934fcc1-7a77-4e55-b8a1-788f514c8316 | Open vSwitch agent | controller | None              | :-)   | UP    | neutron-openvswitch-agent |
| 574d774d-02ee-4697-aee2-7be7c0974a67 | Open vSwitch agent | compute    | None              | :-)   | UP    | neutron-openvswitch-agent |
| b550efe1-63ce-47bf-9984-f1636f72f49d | DHCP agent         | controller | nova              | :-)   | UP    | neutron-dhcp-agent        |
+--------------------------------------+--------------------+------------+-------------------+-------+-------+---------------------------+
```

— **Cinder (Service de stockage en bloc)** : Cinder offre un service de stockage persistant en mode bloc aux machines virtuelles. Les volumes créés avec Cinder peuvent être attachés ou détachés à chaud d'une instance, comme un disque dur virtuel. Cela permet par exemple d'héberger des bases de données ou de stocker des fichiers de manière fiable.
— **Vérification de l'installation**

```
root@controller:/home/openstack# openstack volume service list
+------------------+-------------+------+---------+-------+----------------------------+
| Binary           | Host        | Zone | Status  | State | Updated At                 |
+------------------+-------------+------+---------+-------+----------------------------+
| cinder-scheduler | controller  | nova | enabled | up    | 2025-09-29T12:05:22.000000 |
| cinder-volume    | storage@lvm | nova | enabled | up    | 2025-09-29T12:05:16.000000 |
+------------------+-------------+------+---------+-------+----------------------------+
```

— **Horizon (Tableau de bord)** : Horizon est l'interface graphique web d'OpenStack. Elle permet aux administrateurs et aux utilisateurs de gérer facilement les ressources cloud (machines virtuelles, réseaux, volumes, images, etc.) sans passer par la ligne de commande. Horizon simplifie ainsi l'administration et l'utilisation de la plateforme OpenStack.
— **Vérification de l'installation**



4.4 Vérification globale des services

Après l'installation et la configuration des composants principaux (Keystone, Glance, Placement, Nova, Neutron, Cinder, Horizon), nous avons exécuté la commande suivante afin de vérifier que chaque service est correctement enregistré dans le catalogue d'OpenStack :

```
openstack catalog list
```

Le résultat confirme la présence et la disponibilité de tous les services installés :

```
root@controller:/home/openstack# openstack catalog list
+-----------+-----------+-----------------------------------------------------------+
| Name      | Type      | Endpoints                                                 |
+-----------+-----------+-----------------------------------------------------------+
| cinderv3  | volumev3  | RegionOne                                                 |
|           |           |   internal:                                               |
|           |           | http://controller:8776/v3/8dfb027d0c304b67a2f7313960cbc57b |
|           |           | RegionOne                                                 |
|           |           |   public:                                                 |
|           |           | http://controller:8776/v3/8dfb027d0c304b67a2f7313960cbc57b |
|           |           | RegionOne                                                 |
|           |           |   admin:                                                  |
|           |           | http://controller:8776/v3/8dfb027d0c304b67a2f7313960cbc57b |
|           |           |                                                           |
| neutron   | network   | RegionOne                                                 |
|           |           |   public: http://controller:9696                          |
|           |           | RegionOne                                                 |
|           |           |   internal: http://controller:9696                        |
|           |           | RegionOne                                                 |
|           |           |   admin: http://controller:9696                           |
|           |           |                                                           |
| glance    | image     | RegionOne                                                 |
|           |           |   public: http://controller:9292                          |
|           |           | RegionOne                                                 |
|           |           |   admin: http://controller:9292                           |
|           |           | RegionOne                                                 |
|           |           |   internal: http://controller:9292                        |
|           |           |                                                           |
| nova      | compute   | RegionOne                                                 |
|           |           |   admin: http://controller:8774/v2.1                      |
|           |           | RegionOne                                                 |
|           |           |   internal: http://controller:8774/v2.1                   |
|           |           | RegionOne                                                 |
|           |           |   public: http://controller:8774/v2.1                     |
|           |           |                                                           |
| placement | placement | RegionOne                                                 |
|           |           |   admin: http://controller:8778                           |
|           |           | RegionOne                                                 |
|           |           |   internal: http://controller:8778                        |
|           |           | RegionOne                                                 |
|           |           |   public: http://controller:8778                          |
|           |           |                                                           |
| keystone  | identity  | RegionOne                                                 |
|           |           |   internal: http://controller:5000/v3/                    |
|           |           | RegionOne                                                 |
|           |           |   public: http://controller:5000/v3/                      |
|           |           | RegionOne                                                 |
|           |           |   admin: http://controller:5000/v3/                       |
|           |           |                                                           |
+-----------+-----------+-----------------------------------------------------------+
```

FIGURE 11 – Catalogue des services OpenStack (vérification globale)

5 Lancement d'une instance (Launch Instance)

Dans cette phase, nous utilisons l'option **network provider** (réseau fournisseur) pour lancer une instance OpenStack. La procédure officielle est décrite dans la documentation OpenStack, qui propose deux options réseau : soit uniquement le réseau fournisseur (provider), soit un réseau fournisseur + un réseau self-service (privé). Ici, nous avons choisi l'option réseau fournisseur.

5.1 Création du réseau fournisseur

Avant de lancer une instance, il faut créer le réseau fournisseur. Cela implique la création d'un réseau externe (provider) accessible, généralement vers l'extérieur (ex : Internet). Les commandes typiques sont :

```
openstack network create --external --share --provider-network-type <type>
    --provider-physical-network <physnet> provider
openstack subnet create --network provider --allocation-pool start=<ip>,end=<ip>
    --dns-nameserver <dns> --gateway <gateway> --subnet-range <CIDR> provider_subnet
```

```
root@controller:/home/openstack# openstack network create  --share --ex
ternal \
  --provider-physical-network provider \
  --provider-network-type flat provider
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2025-09-10T20:04:53Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 7eebaec3-c933-481a-90e5-52657ab0a49a |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_qinq              | None                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | provider                             |
| port_security_enabled     | True                                 |
| project_id                | 8dfb027d0c304b67a2f7313960cbc57b     |
| provider:network_type     | flat                                 |
| provider:physical_network | provider                             |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 1                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
```

FIGURE 12 – Vérification de la création du réseau fournisseur

— Create a subnet on the network :

```
root@controller:/home/openstack# openstack subnet create --network prov
ider \
  --allocation-pool start=192.168.1.200,end=192.168.1.250 \
  --dns-nameserver 8.8.8.8 \
  --gateway 192.168.1.1 \
  --subnet-range 192.168.1.0/24 provider
+----------------------+--------------------------------------+
| Field                | Value                                |
+----------------------+--------------------------------------+
| allocation_pools     | 192.168.1.200-192.168.1.250          |
| cidr                 | 192.168.1.0/24                       |
| created_at           | 2025-09-10T20:12:21Z                 |
| description          |                                      |
| dns_nameservers      | 8.8.8.8                              |
| dns_publish_fixed_ip | None                                 |
| enable_dhcp          | True                                 |
| gateway_ip           | 192.168.1.1                          |
| host_routes          |                                      |
| id                   | 3519a5ba-2aba-4770-85e5-92f0b27ee50d |
| ip_version           | 4                                    |
| ipv6_address_mode    | None                                 |
| ipv6_ra_mode         | None                                 |
| name                 | provider                             |
| network_id           | 7eebaec3-c933-481a-90e5-52657ab0a49a |
| project_id           | 8dfb027d0c304b67a2f7313960cbc57b     |
| revision_number      | 0                                    |
| router:external      | True                                 |
| segment_id           | None                                 |
| service_types        |                                      |
| subnetpool_id        | None                                 |
| tags                 |                                      |
| updated_at           | 2025-09-10T20:12:21Z                 |
+----------------------+--------------------------------------+
```

5.2 Création d'un flavor ("m1.nano")

Pour des environnements de test, on crée un petit flavor "m1.nano" afin de consommer peu de ressources. Exemple de commande :

```
root@controller:/home/openstack# openstack flavor create --id 0 --vcpus
 1 --ram 64 --disk 1 m1.nano
+----------------------------+---------+
| Field                      | Value   |
+----------------------------+---------+
| OS-FLV-DISABLED:disabled   | False   |
| OS-FLV-EXT-DATA:ephemeral  | 0       |
| description                | None    |
| disk                       | 1       |
| id                         | 0       |
| name                       | m1.nano |
| os-flavor-access:is_public | True    |
| properties                 |         |
| ram                        | 64      |
| rxtx_factor                | 1.0     |
| swap                       | 0       |
| vcpus                      | 1       |
+----------------------------+---------+
```

FIGURE 13 – Vérification du flavor m1.nano

5.3 Génération de la clé SSH publique

Pour accéder à l'instance via SSH, il faut ajouter une clé publique :

```
ssh-keygen -q -N ""
openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey
openstack keypair list
```

FIGURE 14 – Vérification de la paire de clés

5.4 Ajout de règles du groupe de sécurité

Par défaut, le groupe de sécurité « default » bloque les accès externes. Il faut autoriser au minimum le ping (ICMP) et le SSH (port 22) :

```
root@controller:/home/openstack# openstack security group rule create -
-proto tcp --dst-port 22 default
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| belongs_to_default_sg   | True                                 |
| created_at              | 2025-09-10T20:14:53Z                 |
| description             |                                      |
| direction               | ingress                              |
| ether_type              | IPv4                                 |
| id                      | db0acbda-eda4-4914-8546-7e874a593e2c |
| name                    | None                                 |
| normalized_cidr         | 0.0.0.0/0                            |
| port_range_max          | 22                                   |
| port_range_min          | 22                                   |
| project_id              | 8dfb027d0c304b67a2f7313960cbc57b     |
| protocol                | tcp                                  |
| remote_address_group_id | None                                 |
| remote_group_id         | None                                 |
| remote_ip_prefix        | 0.0.0.0/0                            |
| revision_number         | 0                                    |
| security_group_id       | a8bd52c4-3dfc-468a-97e7-44ecdf541a48 |
| tags                    | []                                   |
| updated_at              | 2025-09-10T20:14:53Z                 |
+-------------------------+--------------------------------------+
```

FIGURE 15 – Vérification des règles du groupe de sécurité

5.5 Lancement de l'instance

Enfin, on lance l'instance en spécifiant les options nécessaires, en particulier le réseau fournisseur :

```
openstack server create --flavor m1.nano --image <nom_image>
    --nic net-id=<ID_du_réseau_provider> --security-group default
    --key-name mykey <nom_instance>
```

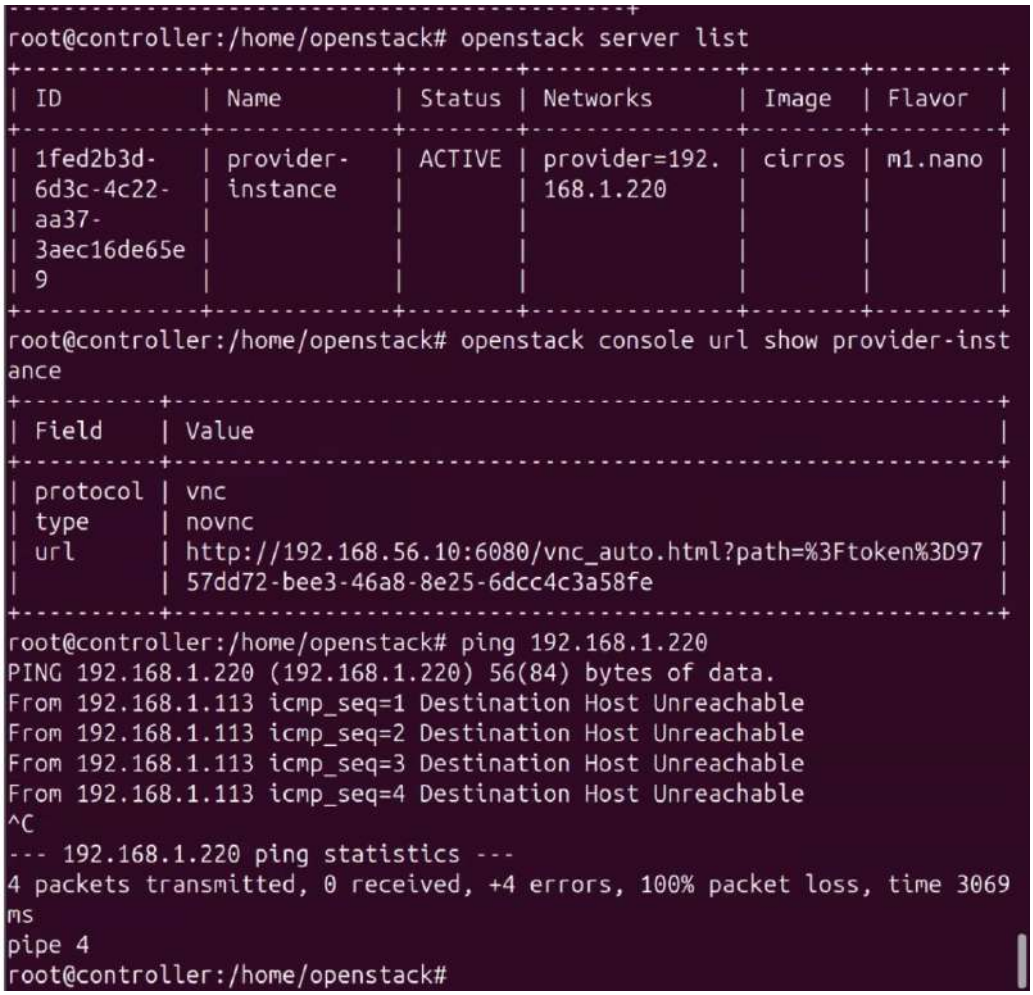


FIGURE 16 – Vérification du lancement de l'instance

FIGURE 17 – Connexion SSH et vérification de l'utilisateur et de l'adresse IP.

Conclusion

Le cloud computing est une technologie qui a révolutionné la manière dont les entreprises et les individus gèrent et accèdent aux ressources informatiques. En offrant une flexibilité, une évolutivité et une accessibilité accrues, le cloud computing permet aux organisations de se concentrer sur leur cœur de métier tout en bénéficiant d'une infrastructure informatique robuste et performante. Au cours de ce projet, nous avons exploré les concepts fondamentaux du cloud computing, y compris les différents modèles de service (IaaS, PaaS, SaaS) et les modèles de déploiement (public, privé, hybride). Nous avons également examiné les avantages et les défis associés à l'adoption du cloud computing, ainsi que les principales plateformes et technologies utilisées dans ce domaine. L'installation et la configuration d'OpenStack nous ont permis de comprendre les aspects pratiques de la mise en place d'une infrastructure cloud privée. Nous avons appris à gérer les ressources, à configurer les réseaux et à assurer la sécurité des données dans un environnement cloud. En conclusion, le cloud computing représente une avancée majeure dans le domaine de l'informatique, offrant des opportunités significatives pour les entreprises de toutes tailles. Cependant, il est essentiel de bien comprendre les implications techniques, économiques et de sécurité avant de migrer vers le cloud. Avec une planification et une gestion appropriées, le cloud computing peut transformer la manière dont les organisations opèrent et innovent dans un monde de plus en plus numérique.