REPUBLIQUE DU SENEGAL

Un peuple-Un but-Une foi

Ministère de l'Enseignement Supérieur et de la Recherche

Direction Générale de l'Enseignement Supérieur Privé

Institut Supérieur d'Informatique

RAPPORT DE FIN DE CYCLE

Présenté et soutenu par :

M. Ahmad Thiongane

M^{lle} Dieynaba Senghor

Pour l'obtention du diplôme de :

Licence Professionnelle : Réseaux Informatiques

Parcours : Informatique

Déploiement d'une infrastructure de cloud privé sécurisé avec Openstack

Soutenu à Dakar, le 20/09/2025

Membres du Jury

Statut	NOM et Prénom	Grade
Président	Dr Latyr NDIAYE	Docteur
Directeur de mémoire	M. Baye Sabarane LAM	Ingénieur Cloud
Examinateur 1 :	M. Moustapha SENE	Ingénieur
Examinateur 2 :	M. Souleymane TOURE	Ingénieur

Année académique : 2024-2025

REPUBLIQUE DU SENEGAL

Un peuple-Un but-Une foi

Ministère de l'Enseignement Supérieur et de la Recherche

Direction Générale de l'Enseignement Supérieur Privé

Institut Supérieur d'Informatique

RAPPORT DE FIN DE CYCLE

Présenté et soutenu par :

M. Ahmad Thiongane

M^{lle} Dieynaba Senghor

Pour l'obtention du diplôme de :

Licence Professionnelle : Réseaux Informatiques

Parcours : Informatique

Déploiement d'une infrastructure de cloud privé sécurisé avec Openstack

Soutenu à Dakar, le 20/09/2025

Membres du Jury

Statut	NOM et Prénom	Grade
Président	Dr Latyr NDIAYE	Docteur
Directeur de mémoire	M. Baye Sabarane Lam	Ingénieur Cloud
Examinateur 1 :	M. Moustapha SENE	Ingénieur
Examinateur 2 :	M. Souleymane TOURE	Ingénieur

Année académique : 2024-2025

DÉDICACE

Ce rapport est dédié avec une profonde gratitude à nos parents, véritables piliers de notre vie. Leur amour, leurs encouragements et leur soutien constant ont été la force qui nous a portés jusqu'au bout de ce parcours.

Nos remerciements s'adressent aussi à nos familles et à toutes celles et ceux qui, de près ou de loin, nous ont accompagnés par leurs conseils, leur patience et leur bienveillance.

Nous soumettons ce travail à l'appréciation du jury, en espérant qu'il reflète non seulement l'effort scientifique qui l'a guidé, mais aussi les valeurs de persévérance, de solidarité et de détermination qui nous animent.

Puisse cette dédicace témoigner de notre reconnaissance sincère et de notre volonté d'honorer tous ceux qui ont cru en nous et qui continueront d'inspirer notre chemin vers l'avenir.

REMERCIEMENTS

Avant toute chose, nous rendons grâce à Dieu, le Tout-Puissant, pour la force, la santé et la patience qu'il nous a accordées tout au long de notre parcours. C'est grâce à Sa volonté que nous avons pu franchir avec succès cette étape importante.

Nos remerciements les plus sincères vont à notre encadreur, Monsieur LAM, Professeur de l'enseignement supérieur en Cloud Computing en Licence 3 de Réseau Informatique. Nous avons eu le privilège de bénéficier de votre accompagnement malgré vos nombreuses responsabilités. Votre disponibilité, vos conseils éclairés et votre bienveillance ont été des repères solides pour mener ce travail à son terme. Nous vous exprimons ici toute notre gratitude et nos profonds respects.

Nous souhaitons également exprimer notre reconnaissance à l'ensemble de nos enseignants, qui ont contribué, chacun à sa manière, à enrichir nos connaissances et à forger notre esprit critique.

Nos pensées reconnaissantes vont également à nos amis, pour leur soutien, leurs encouragements et leur présence amicale à chaque étape de ce parcours. Leur fraternité et leur solidarité ont souvent été une source de motivation dans les moments difficiles.

Nous adressons enfin des remerciements particuliers à nos frères et sœurs, Ibrahima Thiongane et Aissatou Konate, dont le soutien moral et les encouragements constants nous ont donné la force d'aller toujours de l'avant.

À toutes celles et ceux qui, de près ou de loin, ont contribué à la réalisation de ce mémoire, nous exprimons ici notre profonde gratitude.

AVANT-PROPOS

L'Institut Supérieur d'Informatique (ISI), fondé en 1994, est un établissement dont la mission principale est de former des professionnels compétents dans les domaines de l'informatique et de la gestion. L'ISI propose plusieurs filières, notamment le Génie Logiciel, les Réseaux et Télécommunications, l'Informatique de Gestion, les Réseaux Informatiques, la Sécurité des systèmes, les Systèmes embarqués, l'Internet des Objets (IoT) et le Multimédia. Les diplômes délivrés vont du DTS et du BTS jusqu'à la Licence, le Master et le Doctorat.

Dans le cadre de l'obtention de la Licence professionnelle en Réseaux Informatiques, l'Institut exige la rédaction d'un rapport de fin d'études. C'est dans ce contexte que nous avons élaboré le présent document portant sur le sujet : **« Déploiement d'une infrastructure de cloud privé sécurisé avec Openstack »**.

Ce travail illustre la mise en pratique des connaissances acquises durant la formation et propose une solution intégrant les principes de sécurité, de fiabilité et de performance nécessaires à la mise en place d'une infrastructure cloud moderne.

Enfin, nous sollicitons l'indulgence du jury lors de l'évaluation de ce travail et espérons que ce rapport pourra constituer une contribution utile dans le domaine du déploiement et de la gestion sécurisée des environnements cloud.

SOMMAIRE

GLOSSAIRE

API (Application Programming Interface) : interface permettant à des applications ou services de communiquer entre eux et d'échanger des données.

Br-ex (External Bridge) : interface réseau d'Openstack permettant la connexion des instances au réseau externe.

CPU (Central Processing Unit) : unité centrale de traitement d'un ordinateur, responsable de l'exécution des instructions.

Go (Gigaoctet) : unité de mesure de capacité de stockage ou de mémoire, équivalente à 1024 Mo.

HTTPS (Hypertext Transfer Protocol Secure) : protocole de communication sécurisé sur Internet utilisant le chiffrement TLS/SSL pour protéger les données.

ICMP (Internet Control Message Protocol) : protocole utilisé pour échanger des messages de contrôle et de diagnostic sur les réseaux IP, notamment pour le ping.

IT (Information Technology) : ensemble des technologies liées à l'informatique et au traitement de l'information.

Kolla internal VIP : adresse IP virtuelle utilisée par Kolla-Ansible pour les communications internes entre services Openstack.

Kolla external VIP : adresse IP virtuelle utilisée par Kolla-Ansible pour les communications externes vers les services Openstack.

KVM (Kernel-based Virtual Machine) : technologie de virtualisation intégrée au noyau Linux permettant d'exécuter des machines virtuelles.

LDAP (Lightweight Directory Access Protocol) : protocole permettant la gestion centralisée des utilisateurs et des groupes.

Node exporter : outil de collecte de métriques système sur un serveur, utilisé avec Prometheus pour la supervision.

Qemu (probablement QEMU) : logiciel de virtualisation permettant de créer et exécuter des machines virtuelles.

RAM (Random Access Memory) : mémoire vive d'un ordinateur, utilisée pour stocker temporairement les données et instructions en cours d'utilisation.

SSH (Secure Shell) : protocole permettant d'établir une connexion sécurisée à distance sur un serveur ou un équipement réseau.

TCP (Transmission Control Protocol) : protocole réseau garantissant la transmission fiable des données entre machines.

UDP (User Datagram Protocol) : protocole réseau léger pour la transmission rapide des données, sans garantie de réception.

URL (Uniform Resource Locator) : adresse web permettant de localiser une ressource sur Internet.

Vcpu (Virtual CPU) : unité centrale virtuelle attribuée à une machine virtuelle pour exécuter des tâches.

VM (Virtual Machine / Machine Virtuelle) : environnement informatique simulé permettant d'exécuter un système d'exploitation et des applications comme sur un ordinateur physique.

SSL/TLS (Secure Sockets Layer / Transport Layer Security) : protocoles de sécurité permettant de chiffrer les communications sur un réseau.

Hyperviseur : Logiciel permettant de créer et de gérer des machines

Instance : Machine virtuelle déployée sur l'infrastructure Openstack.

Flavor : Modèle de ressources prédéfini dans Openstack spécifiant la puissance de calcul, la mémoire et l'espace disque attribué à une instance.

Gabarit : Modèle standard servant de référence pour créer rapidement une ressource ou une configuration dans un environnement cloud.

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Ce rapport présente l'étude et le déploiement d'une infrastructure de cloud privé sécurisé en s'appuyant sur la plateforme OpenStack. L'objectif est de concevoir une solution flexible, évolutive et fiable qui permette aux organisations de bénéficier des avantages du cloud computing tout en garantissant la sécurité et la souveraineté des données. La démarche adoptée combine une analyse théorique de la virtualisation et des modèles de services cloud avec une mise en œuvre pratique orientée production.

L'infrastructure a été déployée en mode multi-nœuds à l'aide de Kolla-Ansible, ce qui assure modularité et scalabilité. Plusieurs mécanismes de sécurisation ont été intégrés : une gestion centralisée des identités via LDAP avec Keystone, le chiffrement des communications par TLS/HTTPS, ainsi que la mise en place de politiques de contrôle d'accès. Enfin, la supervision de l'environnement a été assurée grâce à Prometheus et Grafana, permettant un suivi en temps réel des performances et une meilleure résilience de la plateforme.

ABSTRACT

This report presents the study and deployment of a secure private cloud infrastructure based on the OpenStack platform. The objective is to design a flexible, scalable, and reliable solution that enables organizations to benefit from the advantages of cloud computing while ensuring security and data sovereignty. The adopted approach combines a theoretical analysis of virtualization and cloud service models with a practical, production-oriented implementation.

The infrastructure was deployed in a multi-node architecture using Kolla-Ansible, ensuring modularity and scalability. Several security mechanisms were integrated: centralized identity management with LDAP through Keystone, encrypted communications with TLS/HTTPS, and the implementation of access control policies. Finally, the environment was monitored using Prometheus and Grafana, providing real-time performance tracking and improving the resilience of the platform.

Chapitre 1 : Introduction générale

Le développement des technologies numériques a profondément transformé les infrastructures informatiques traditionnelles. Aujourd'hui, les organisations recherchent des solutions plus flexibles, scalables et sécurisées pour répondre à leurs besoins en matière de traitement, de stockage et de gestion des données. Dans ce contexte, le cloud computing s'est progressivement imposé comme une solution incontournable pour héberger, gérer et faire évoluer des services informatiques. Parmi les différentes formes que peut prendre le cloud, le cloud privé se distingue par sa capacité à offrir un environnement contrôlé et sécurisé.

Aujourd'hui, de nombreuses entreprises, organisations publiques et institutions de recherche choisissent d'implémenter des solutions de cloud privé open source, à la fois pour des raisons économiques, techniques et stratégiques. C'est dans cette optique que s'inscrit notre travail, qui consiste à concevoir, déployer et de documenter une infrastructure de cloud privé intégrant des mécanismes de sécurité avancées en s'appuyant sur la plateforme Openstack.

Afin de mieux cerner la portée de ce travail, nous allons d'abord exposer le cadre théorique et méthodologique dans lequel s'inscrit notre projet. Cette première étape est essentielle pour comprendre les fondements du cloud privé et les principes qui ont guidé notre approche technique.

1.1 Cadre théorique

1.1.1 Contexte

Aujourd'hui, face à l'évolution constante des technologies, les entreprises et institutions doivent moderniser leurs systèmes informatiques pour gagner en efficacité, en flexibilité et en sécurité. Le cloud computing s'impose comme une solution idéale, offrant des ressources partagées et une réduction des coûts liés aux infrastructures physiques. Il existe différentes approches de déploiement telles que le cloud public, privé et hybride ainsi que plusieurs solutions comme AWS, Microsoft Azure, Google Cloud pour le cloud public, ou encore VMware, CloudStack et Openstack pour le cloud privé ou hybride.

Parmi les solutions open source les plus populaires pour déployer des cloud privés ou hybrides, Openstack se démarque grâce à sa modularité, ses

fonctionnalités complètes et sa communauté dynamique. Ainsi, Openstack apparaît comme un choix stratégique pour les organisations qui veulent adopter le cloud tout en gardant le contrôle de leurs infrastructures. Néanmoins, cette transition vers le cloud avec Openstack présente toutefois des défis techniques, organisationnels et économiques que les entreprises doivent anticiper pour garantir une migration réussie.

1.1.2 Problématique

Face à l'adoption massive du cloud, les organisations cherchent à garantir sécurité, robustesse et souveraineté tout en préservant leur indépendance technologique. Contrairement au cloud public, le cloud privé permet de conserver la maîtrise des données et d'appliquer des politiques de traçabilité adaptées aux besoins spécifiques. Toutefois, son déploiement représente un défi technique et stratégique qui nécessite une approche méthodique.

Dès lors, une question centrale se pose : comment déployer une infrastructure de cloud privé à la fois robuste, sécurisée et souveraine en s'appuyant sur la solution open source Openstack ?

Pour y répondre, nous définirons des objectifs clairs orientant le déploiement et la sécurisation de l'infrastructure.

1.1.3 Objectifs du rapport

Notre objectif général est de mener une étude approfondie et de mettre en œuvre une solution de cloud computing privé sécurisé en utilisant Openstack. Pour atteindre cet objectif, nous suivrons plusieurs étapes clés:

- **Objectif n°1** : Analyse des besoins
  L'objectif est d'évaluer les ressources matérielles et logicielles nécessaires afin de garantir la faisabilité de la solution.
- **Objectif n°2** : Configuration et Déploiement d'Openstack
  Après avoir défini les besoins, cette étape consiste à installer Openstack ainsi que ses composants essentiels.
- **Objectif n°3** : Intégration de l'annuaire LDAP avec Keystone
  Relier l'annuaire LDAP à Openstack pour une gestion centralisée des identités et des rôles.

- **Objectif n°4** : Implémentation d'une connexion sécurisée via TLS/HTTPS
  Configurer le protocole TLS afin de garantir un accès sécurisé en HTTPS aux différents services de la plateforme.
- **Objectif n°5** : Déploiement d'instances et validation
  Créer et configurer une instance afin de vérifier le bon fonctionnement de l'environnement.
- **Objectif n°6** : Déployer Grafana et Prometheus pour la supervision

1.2 Cadre méthodologique

1.2.1 Délimitation du champ d'étude

L'étude présentée dans ce rapport n'a pas été réalisée dans le cadre d'un stage en entreprise, mais dans une logique d'apprentissage personnel et pédagogique en s'appuyant sur des recherches et l'exploitation de diverses ressources documentaires disponibles en ligne.

L'objectif principal est de mettre en œuvre une architecture simplifiée, mais représentative, d'un cloud privé basé sur Openstack. Ainsi l'étude reste donc volontairement centrée sur un déploiement local mais pleinement opérationnel, permettant de comprendre et de maîtriser l'ensemble des étapes du déploiement.

Afin de mener à bien ce projet, différentes approches de recherche ont été mises en œuvre pour acquérir les connaissances nécessaires.

1.2.2 Techniques de la recherche

Pour avancer dans ce projet et comprendre les différents aspects techniques, nous avons eu recours à plusieurs sources d'information accessibles et variées :

Tout d'abord des vidéos pédagogiques sur YouTube ont été consultés afin de mieux comprendre le processus d'installation, de configuration et de déploiement d'Openstack. Ces vidéos ont permis de visualiser les étapes critiques du projet et de contourner certains blocages techniques rencontrés. Ensuite nous nous sommes appuyés sur la documentation officielle d'Openstack qui reste la référence la plus fiable. Il a fourni des guides détaillés, des exemples de configuration, ainsi qu'un guide pour comprendre le rôle des différents services. Enfin nous avons consultes des mémoires et travaux d'étudiants disponibles en ligne et à la bibliothèque.

Ces différentes ressources théoriques et pratiques ont été adaptées au contexte spécifique de notre rapport afin de déployer une solution fonctionnelle et compréhensible.

En résumé, ce premier chapitre a permis de poser les bases du projet : nous avons clarifié le contexte, défini les objectifs, présenté la problématique et expliqué la démarche choisie.

Ainsi, dans le chapitre suivant, nous approfondirons les concepts fondamentaux liés à la virtualisation et du cloud computing, présenterons les principaux modèles de services et de déploiement, puis introduirons Openstack et son architecture.

Chapitre 2 : Cadre conceptuel

2.1 Généralités sur la virtualisation et le cloud computing

Aujourd'hui, les entreprises ont besoin de technologies rapides, flexibles et puissantes. Deux de ces technologies sont la virtualisation et le cloud computing. Elles sont souvent utilisées ensemble, mais ce ne sont pas la même chose. Elles reposent sur des principes différents, tout en étant complémentaires. Pour bien comprendre comment créer un cloud privé, il faut d'abord connaître les bases de ces deux technologies.

2.1.1 Fondamentaux de la virtualisation

La virtualisation est une technologie qui consiste à simuler un environnement informatique, généralement matériel, au sein d'un système physique réel. En d'autres termes, elle permet de créer plusieurs machines virtuelles sur une seule et même machine physique, grâce à un hyperviseur. Cette approche permet d'optimiser l'utilisation des ressources, de réduire les coûts d'infrastructure, et de faciliter la gestion des systèmes informatiques. Après avoir compris le principe de base de la virtualisation, il est important de connaître ses différentes formes, chacune adaptée à des besoins spécifiques.

2.1.2 Les différents types de virtualisation

Il existe deux grands types de virtualisation selon le type d'hyperviseur utilisé :

- **La virtualisation de type 1**

Également connue sous le nom d'hyperviseur bare-metal, il utilise un hyperviseur qui s'exécute directement sur le matériel physique, sans passer par un système d'exploitation.

Cela permet de meilleures performances, une utilisation efficace des ressources et une sécurité renforcée. C'est le type d'hyperviseur le plus utilisé dans les centres de données et les environnements professionnels. Des exemples incluent :

- VMware ESXi : un hyperviseur très répandu dans les entreprises. Il s'installe directement sur le serveur physique et permet une gestion performante des ressources sans avoir besoin d'un système d'exploitation.

- Microsoft Hyper-V : intégré à Windows Server, il permet de créer et gérer des machines virtuelles dans un environnement Windows. Il est très utilisé dans les infrastructures Microsoft grâce à ses outils de gestion et de réplication.
- KVM (Kernel-based Virtual Machine) : intégré au noyau Linux, KVM transforme un système Linux en un hyperviseur puissant et sécurisé. Il est particulièrement utilisé dans les environnements cloud et les datacenters.

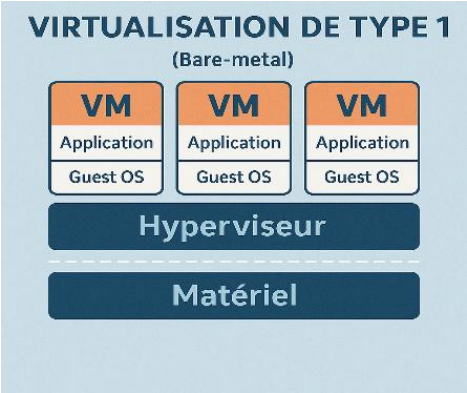


Figure 2. 1 : Architecture de virtualisation de type 1

- **La virtualisation de type 2 :**

La virtualisation de type 2 fonctionne au-dessus d'un système d'exploitation déjà installé sur l'ordinateur. On parle d'hyperviseur hébergé. Ce type de virtualisation est plus facile à mettre en place, mais il est moins performant que le type 1, car il dépend du système d'exploitation hôte. Il est souvent utilisé pour les tests, les environnements de développement ou un usage personnel. Parmi les exemples de virtualisation de type 2, nous pouvons citer:

- Oracle VirtualBox : solution gratuite et open source, VirtualBox fonctionne sur différents systèmes d'exploitation (Windows, Linux, macOS). Il est très utilisé pour les tests et les environnements de développement grâce à sa simplicité et sa flexibilité.
- VMware Workstation : conçu pour exécuter plusieurs systèmes d'exploitation sur un seul ordinateur, cet hyperviseur s'installe comme un logiciel classique sur Windows ou Linux. Il est pratique pour les développeurs ou les administrateurs qui veulent tester plusieurs systèmes en parallèle.

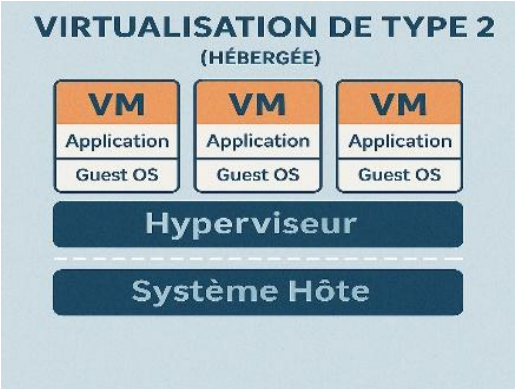


Figure 2.2 Architecture de virtualisation de type 2

Après avoir exploré les principes de la virtualisation, il est maintenant temps de s'intéresser à une technologie étroitement liée et souvent complémentaire : le cloud computing.

2.2 Généralités sur le cloud computing

Le cloud computing, ou informatique en nuage, est une technologie qui permet d'accéder à distance à des ressources informatiques tels que des serveurs, du stockage, des bases de données ou des logiciels via Internet. Contrairement aux solutions locales, cette approche ne nécessite pas d'avoir tout le matériel sur place. Cette technologie a profondément changé la façon dont les entreprises et les particuliers utilisent l'informatique. Elle offre de nombreux avantages comme la flexibilité, la scalabilité et une réduction des coûts.

Le cloud computing repose sur trois caractéristiques fondamentales que sont : la mutualisation des ressources, l'élasticité rapide et service mesuré mais aussi la facturation à l'usage. Ce modèle permet aux organisations de gagner en agilité, de réduire leurs coûts d'exploitation et d'accélérer leur innovation. Toutefois, pour bien en comprendre la portée, il est essentiel d'examiner les modèles de services proposés par le cloud. Ainsi, le cloud computing s'impose aujourd'hui comme une évolution logique et puissante des pratiques informatiques traditionnelles, portée par la virtualisation.

Dans cette dynamique, examinons à présent les différents modèles de services proposés dans le cloud.

2.2.1 Modèles de services

Le cloud computing se décline en trois grands modèles de services, chacun offrant un niveau d'abstraction différent selon les besoins des utilisateurs :

Modèle	Description	Responsabilité de l'utilisateur	Exemples
IaaS *(*Infrastructure as a Service)	Fournit des ressources matérielles virtualisées (serveurs, stockage, réseaux) via Internet.	Gérer le système d'exploitation, les applications et les données.	Amazon EC2, Google Compute Engine, Openstack
PaaS (Platform as a Service)	Fournit un environnement complet pour développer et déployer des applications. L'infrastructure est gérée par le fournisseur.	Développer, configurer et déployer les applications.	Google App Engine, Heroku, Azure App Services
SaaS (Software as a Service)	Met à disposition des applications prêtes à l'emploi accessibles via un navigateur.	Utiliser directement l'application.	Gmail, Microsoft 365, Dropbox

Tableau 2.1 : Les différents modèles de services cloud

En résumé, ces modèles de services montrent toute la flexibilité du cloud computing. Ils permettent à chaque profil qu'il s'agisse d'un administrateur, d'un développeur ou d'un utilisateur final de choisir le niveau de contrôle et de responsabilité qui lui convient.

Maintenant que nous avons vu comment les services cloud sont consommés, intéressons-nous aux différents modèles de déploiement qui définissent où et comment ces services sont hébergés.

2.2.2 Modèles de déploiement

Les services cloud peuvent être mis en œuvre selon plusieurs modèles de déploiement, chacun répondant à des exigences spécifiques en matière de sécurité, de contrôle, de coût et de flexibilité :

Modèle	Description	Avantages	Inconvénients
Cloud public	Infrastructure partagée, hébergée par un fournisseur tiers et accessible via Internet.	Coût réduit Haute flexibilité	Moins de contrôle Sécurité dépendante du fournisseur
Cloud privé	Infrastructure dédiée à une seule organisation, hébergée en interne ou par un prestataire.	Contrôle total Sécurité renforcée	Coûts plus élevés Moins flexible
Cloud hybride	Combine cloud public et privé pour tirer parti des deux environnements.	Équilibre entre sécurité et souplesse Optimisation des ressources	Plus complexe à gérer et intégrer

Tableau 1.2 : Les différents modèles de déploiement

En définitive, chaque modèle de déploiement répond à des besoins spécifiques. Le choix dépend des priorités de l'organisation en matière de sécurité, de coût, de flexibilité et de conformité réglementaire.

Maintenant que nous avons posé les bases du cloud computing, il est pertinent de comparer les solutions existantes sur le marché.

2.3 Étude comparative d'Openstack face aux autres solutions de cloud computing

Dans un écosystème cloud de plus en plus concurrentiel, plusieurs plateformes se disputent le marché de l'infrastructure à la demande. Si des géants comme Amazon Web Services (AWS), Microsoft Azure ou Google Cloud Platform (GCP) dominent le cloud public, d'autres solutions comme Openstack s'illustrent dans le domaine du cloud privé.

Critère	Openstack	AWS / Azure / GCP	VMware vSphere
Type	Open source, cloud privé	Cloud public	Virtualisation / cloud privé
Licence	Open source (Apache 2.0)	Propriétaire	Propriétaire
Modularité	Très modulaire	Intégrée mais moins personnalisable	Bonne, mais fermée
Coût	Gratuit (hors infrastructure)	Élevé selon l'usage	Licence coûteuse
Contrôle / Personnalisation	Très élevé	Limité	Moyen
Communauté	Active, mondiale	Très vaste (support commercial)	Moins communautaire
Cas d'usage idéal	Souveraineté, personnalisation, cloud privé maîtrisé	Scalabilité rapide, déploiement mondial, services managés	Virtualisation d'entreprise, modernisation de datacenter

Tableau 2.3 : Comparaisons des différentes solution cloud

En somme, Openstack se démarque par sa nature open source, sa modularité, et sa capacité à s'adapter à des environnements exigeant un haut niveau de contrôle, de sécurité et de souveraineté.

Forts de cette comparaison, nous comprenons mieux pourquoi Openstack est souvent choisi pour les projets de cloud privé sécurisé.

2.4 Présentation d'Openstack : architecture et composants

Openstack a vu le jour en 2010, fruit d'une collaboration entre Rackspace Hosting et la NASA. Son objectif : offrir une solution de cloud computing open source capable de rivaliser avec les plateformes propriétaires telles qu'Amazon Web Services (AWS).

Openstack est principalement utilisé pour déployer des infrastructures IaaS (Infrastructure as a Service). Il repose sur une architecture modulaire composée de services

interconnectés, chacun chargé d'un rôle spécifique dans la gestion du cloud : calcul, réseau, stockage, authentification, orchestration, etc.

En résumé, Openstack est une solution ouverte, évolutive et modulaire qui permet de construire un cloud privé maîtrisé, adapté aux besoins des entreprises et des institutions.

Partant de ce principe, il est essentiel de comprendre comment ces services s'organisent et interagissent pour assurer le fonctionnement d'Openstack.

2.4.1 Architecture d'Openstack

Dans la structure d'Openstack, chaque service dispose de ses propres composants et API. Cette conception modulaire permet une grande flexibilité et facilite l'intégration avec d'autres outils ou systèmes d'information.

- **Architecture conceptuelle**

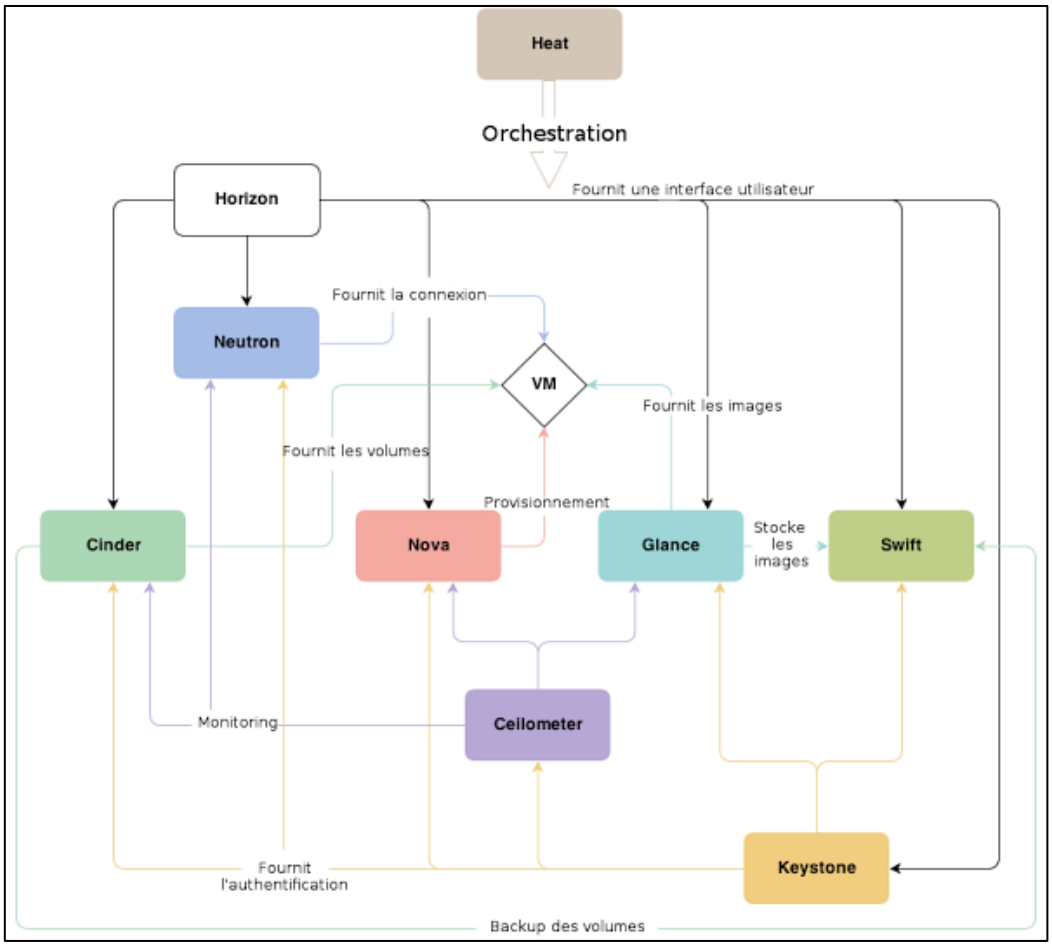


Figure 2.3 Architecture conceptuelle d'Openstack

Cette illustration présente la vision conceptuelle d'Openstack du point de vue de l'utilisateur et met en évidence les interactions entre ses différents services tout en assurant une communication standardisée et fluide entre les services.

- **Architecture simple**

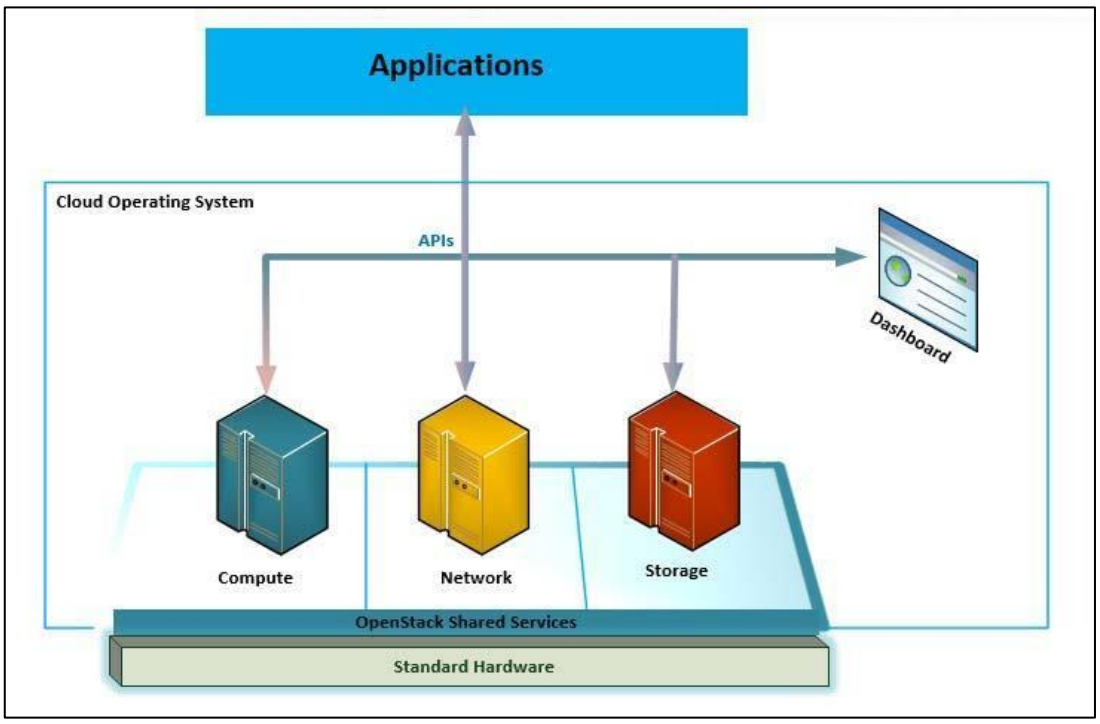


Figure 2.4 : Architecture simple d'Openstack

2.4.2 Les différents composants d'Openstack

Openstack est constitué d'un ensemble de services modulaires, chacun dédié à une fonction spécifique du cloud. Ces services communiquent entre eux principalement via des API REST, formant une infrastructure IaaS complète et flexible. Parmi ses principaux composants nous avons :

- **Nova (Service de calcul):**

Nova gère la création, l'exécution et l'administration des machines virtuelles. Il orchestre le déploiement des instances sur les nœuds de calcul et propose une API permettant aux utilisateurs de démarrer, arrêter ou supprimer leurs VMs.

- **Neutron (Service réseau):**

Neutron fournit la connectivité réseau aux instances. Il permet la création de réseaux virtuels, sous-réseaux, routeurs et l'application de règles de sécurité. Ce service joue un rôle central dans l'interconnexion des ressources.

- **Cinder & Swift (Services de stockage):**
  - **Cinder** fournit un stockage bloc que l'on peut attacher aux instances, idéal pour les bases de données ou systèmes nécessitant un stockage persistant.
  - **Swift** fournit un stockage objet hautement disponible, adapté au stockage massif de fichiers, images ou sauvegardes.

- **Keystone (Authentification et gestion des accès):**

Keystone assure l'authentification et l'autorisation des utilisateurs et services. Il centralise la gestion des identités et des rôles, garantissant un contrôle d'accès sécurisé.

- **Glance (Gestion des images):**

Glance gère les images de machines virtuelles. Il permet de stocker, partager et distribuer ces images, utilisées par Nova lors du déploiement des instances.

- **Horizon (Interface utilisateur):**

Horizon est le tableau de bord web d'Openstack. Il offre une interface graphique intuitive pour gérer les ressources (instances, réseaux, volumes, images) directement via un navigateur.

En conclusion, Openstack s'appuie sur une architecture modulaire et distribuée qui lui confère flexibilité, évolutivité et robustesse. Ces caractéristiques en font une solution particulièrement adaptée à la mise en place d'un cloud privé sécurisé et performant.

Après avoir établi les bases théoriques, nous pouvons désormais passer à la phase pratique, en abordant la mise en œuvre et le déploiement de l'infrastructure Openstack.

Chapitre 3 : Mise en place de la solution Openstack

3.1 Les différentes architecture de déploiement d'Openstack

Le déploiement d'Openstack peut suivre plusieurs architectures en fonction des besoins, des ressources disponibles et de la finalité du projet. Deux grandes approches se distinguent : l'architecture Multi-nœuds, adaptée aux environnements distribués, et l'architecture Tout-en-un (All-in-One), privilégiée pour des environnements plus compacts ou pour des phases de test.

3.1.1 Le déploiement en Multi-nœuds

Dans un déploiement Multi-nœuds, les services Openstack sont répartis sur plusieurs machines dédiées (compute, storage, controller), ce qui améliore la scalabilité et la résilience en évitant qu'une panne n'affecte l'ensemble de la plateforme. Cette architecture, plus complexe à mettre en œuvre, exige un réseau performant, une gestion fine de la configuration et une supervision rigoureuse. Elle est généralement utilisée en production pour assurer la haute disponibilité et permettre une évolution progressive de l'infrastructure.

3.1.2 Le déploiement tout-en-un (All-in-One)

L'architecture All-in-One regroupe tous les services Openstack sur un seul serveur. Cette configuration simplifie grandement l'installation et la gestion, réduit les besoins matériels et facilite les expérimentations. Elle est particulièrement adaptée aux environnements de test, de formation ou de développement. Toutefois, elle présente des limites importantes : performances réduites sous forte charge, absence de haute disponibilité et impossibilité de répartir les rôles, ce qui rend la plateforme vulnérable à une panne unique.

3.1.3 Comparaison entre le tout en un et le multi-nœuds

Critère	Multi-nœuds	All-in-One
Complexité	Élevée	Faible
Coût matériel	Plus important	Réduit
Scalabilité	Excellente	Très limitée
Haute disponibilité	Possible	Non

Performances	Optimisées par la répartition des charges	Limitées par une seule machine
Cas d'usage	Production, grandes entreprises	Tests, maquettes, formations

Tableau 3.1 : Comparaison des différentes architectures de déploiement

En définitive, le choix entre un déploiement Multi-nœuds et All-in-One dépend principalement des objectifs et des contraintes du projet : le premier offre robustesse, performance et évolutivité pour la production, tandis que le second privilégie simplicité et rapidité pour des environnements limités ou expérimentaux.

3.2 Les différentes méthodes de déploiement

Plusieurs méthodes permettent de déployer Openstack, chacune présentant des avantages et des limites selon l'objectif recherché, le niveau de maîtrise technique et les ressources disponibles.

- **Le déploiement manuel :** Il consiste à installer et configurer chaque composant d'Openstack étape par étape. Cette méthode offre un contrôle total et une compréhension fine du système, mais elle est longue, complexe et sujette aux erreurs.
- **Packstack** : c'est un outil automatisé qui simplifie l'installation d'Openstack via des scripts Puppet. Facile à utiliser, il est adapté aux environnements de test ou de démonstration, mais peu recommandé pour la production.
- **DevStack:** Il fournit un environnement rapide à installer pour le développement et les tests fonctionnels. Très flexible, il n'est cependant pas destiné à la mise en production, notamment à cause de sa faible résilience.
- **MicroStack:** C'est une version compacte d'Openstack, facile à installer et à administrer sur un seul nœud, conçue pour des environnements légers ou des prototypes.
- **Juju:** Il propose un déploiement basé sur des paquets de configuration permettant d'automatiser et d'orchestrer Openstack de manière modulaire. Il convient aussi bien aux environnements de test qu'à la production.
- **TripleO:** Il utilise Openstack pour déployer et gérer Openstack lui-même, avec une approche **«** Openstack on Openstack » optimisée pour les environnements de production.
- **Kolla-Ansible:** Il permet de déployer Openstack en utilisant des conteneurs Docker orchestrés via Ansible, offrant une installation rapide, modulaire et adaptée à la production.

- **Tableau comparatif des différents types de déploiement**

Méthode	**Complexité**	**Production** prête	**Rapidité d'installation**	**Flexibilité**
Manuel	Élevée	Oui	Très faible	Très forte
Packstack	Moyenne	Limité	Rapide	Moyenne
DevStack	Faible	Non	Très rapide	Faible
MicroStack	Faible	Non	Très rapide	Faible
Juju	Moyenne	Oui	Moyenne	Moyenne
TripleO	Élevée	Oui	Moyenne	Forte
Kolla-Ansible	Moyenne	Oui	Rapide	Forte

Tableau 3.2 : Comparaison des différents types de déploiement

En conclusion, le choix de la méthode de déploiement dépend du contexte et des besoins : les solutions comme TripleO et Kolla-Ansible sont privilégiées pour la production grâce à leur robustesse et leur flexibilité, tandis que DevStack, Packstack ou MicroStack se prêtent mieux aux environnements de test, de formation ou de prototypage.

Après analyse des différentes méthodes de déploiement, nous avons opté pour Kolla-Ansible en mode Multi-nœuds, qui offre une architecture distribuée, résiliente et évolutive, adaptée aux exigences de production. Ce choix garantit une meilleure scalabilité, une gestion optimisée des charges de travail et une tolérance accrue aux pannes grâce à la séparation des rôles entre nœuds contrôleurs, nœuds de calcul et nœuds de stockage.

C'est dans cette logique que nous avons entrepris notre implémentation, présentée dans le chapitre suivant.

3.3 Travaux réalisés

Dans cette section, nous présentons la mise en œuvre concrète de notre solution en déployant Openstack avec Kolla-Ansible en mode Multi-nœuds. Nous détaillons les étapes suivies, les configurations réalisées ainsi que les principales difficultés rencontrées et les solutions apportées.

3.3.1 Structure du réseau physique:

- Architecture physique du réseau

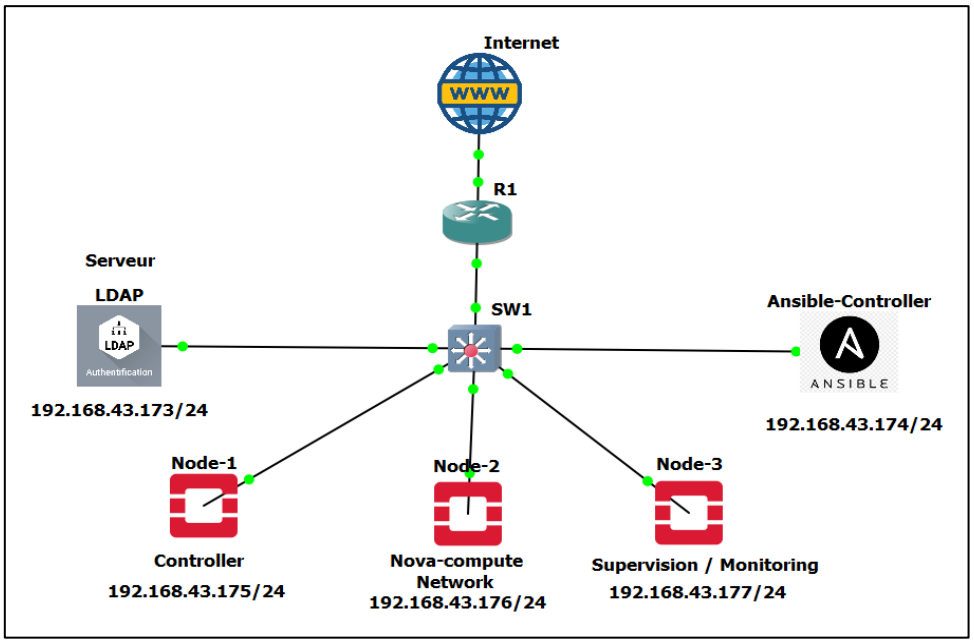


Figure 3.1 : Architecture physique du réseau

3.3.2 Prérequis pour le déploiement Multi-nœuds d'Openstack avec Kolla-Ansible

Dans le cadre de notre projet, le déploiement multi-nœuds d'Openstack a été réalisé

sur quatre serveurs virtuels distincts, installés sous Ubuntu 24.04 LTS sur VMware Workstation. Chacun de ces serveurs joue un rôle bien défini dans l'infrastructure :

- **Serveur Ansible Controller**
  - Rôle : hébergement de l'environnement Ansible et gestion du déploiement via Kolla-Ansible.
  - Ressources : 2 Go RAM, 30 Go disque, 2 CPU.
  - 2 interfaces réseaux : bridge, host-only
- **Nœud Controller**
  - Rôle : exécution des services de gestion Openstack (Keystone, Glance, Horizon, placement, scheduler, etc.).
  - Ressources : 5 Go RAM, 50 Go disque, 4 CPU.
  - 2 interfaces réseaux : bridge, host-only
- **Nœud Compute + Network**
  - Rôle : hébergement des machines virtuelles et gestion du réseau (Neutron).

- Ressources : 3 Go RAM, 30 Go disque, 4 CPU.
- 3 interfaces réseaux : bridge, host-only, Nat

- **Nœud Supervision/Monitoring**
  - Rôle : collecte de métriques, supervision et suivi des performances de l'infrastructure.
  - Ressources : 2 Go RAM, 30 Go disque, 2 CPU.
  - 2 interfaces réseaux : bridge, host-only

3.3.3 Configuration et déploiement d'Openstack via Kolla-Ansible

➢ **Mise en place de l'environnement réseau**

Dans notre architecture multi-nœuds, les interfaces réseau sont configurées en fonction du rôle de chaque serveur :

- **Ansible Controller** (nœud d'orchestration) : 2 interfaces
  - **ens33** → bridge
  - **ens34** → host-only
- **Controller** (nœud principal avec Neutron) : 2 interfaces
  - **ens33** → bridge
  - **ens34** → host-only
- **Compute + Network** (nœud 2) : 3 interfaces
  - **ens33** → bridge
  - **ens34** → host-only
  - **ens35** → NAT pour l'accès Internet, mises à jour et téléchargement
- **Supervision / Monitoring** (nœud 3) : 2 interfaces
  - **ens33** → bridge
  - **ens34** → host-only

Cette configuration permet à chaque nœud d'avoir la connectivité nécessaire selon son rôle, tout en limitant le nombre d'interfaces physiques aux besoins réels.

➢ **Mise en place de la configuration sur ansible Controller**

- Mise à jour du système :

```
apt update && apt upgrade -y
```

- Installation des dépendances et bibliothèques nécessaires :

```
apt install -y python3-dev libffi-dev gcc libssl-dev python3-venv git
libdbus-glib-1-dev docker-compose net-tools openvswitch-switch
```

- Installation et activation de Docker:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh
sudo systemctl start docker && sudo systemctl enable docker
```

- Activation d'Open vSwitch:

```
systemctl start openvswitch-switch && systemctl enable
openvswitch-switch
```

- Création de l'utilisateur **di-med** pour exécuter les commandes **sudo** sans mot de passe:

Cela sert à automatiser entièrement l'installation et la configuration, sans interruptions dues à des demandes de mot de passe.

```
sudo adduser di-med
sudo visudo
```

Ajouter la directive suivante :

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
di-med ALL=(ALL) NOPASSWD:ALL
```

Figure 3.2 : Configuration du fichier /etc/sudoers

Maintenant nous allons nous connecter avec le nouvel utilisateur avec :

```
su - di-med
```

- Création d'un environnement virtuel Python et installation des modules :

```
python3 -m venv ~/kolla-venv
source ~/kolla-venv/bin/activate
```

- Installation des modules Python et Docker

```
pip install --upgrade pip
pip install dbus-python docker
```

- Installation de Kolla-Ansible :

```
pip install git+https://opendev.org/openstack/kolla-ansible@stable/2025.1
```

- Copie des fichiers de configuration et de l'inventaire Kolla Ansible:

```
mkdir -p /etc/kolla
chown $USER:$USER /etc/kolla
cp -r ~/kolla-venv/share/kolla-ansible/etc_examples/kolla/* /etc/kolla
cp ~/kolla-venv/share/kolla-ansible/ansible/inventory/multinode ./multinode
```

- Génération des mots de passe :

```
kolla-genpwd
```

```
(kolla-venv) di-med@ansible-controller:~$ kolla-genpwd
(kolla-venv) di-med@ansible-controller:~$
```

Figure 3.3 : Génération des mots de passe

- Modification des paramètres globaux de Kolla-Ansible :

Ouvrir le fichier de configuration principal globals.yml

Les paramètres essentiels comme l'adresse IP virtuelle, les interfaces réseau, les services à activer, et la configuration du backend réseau sont définis ici.

```
vim /etc/kolla/globals.yml
```

Décommenter et éditer les directives suivantes :

```
39 ###############
40 # Kolla options
41 ###############
42 # Valid options are [ COPY_ONCE, COPY_ALWAYS ]
43 config_strategy: "COPY_ALWAYS"
44
45 # Valid options are ['centos', 'debian', 'rocky', 'ubuntu']
46 kolla_base_distro: "ubuntu"
47
48 # Do not override this unless you know what you are doing.
49 openstack_release: "2025.1"
50
51 # Location of configuration overrides
52 node_custom_config: "/etc/kolla/config"
53 # 'network_interface' as set in the Networking section below.
54 kolla_internal_vip_address: "192.168.43.80"
55
56 # This should be a VIP, an unused IP on your network that will float between
57 # the hosts running keepalived for high-availability. If you want to run an
58 # All-In-One without haproxy and keepalived, you can set enable_haproxy to no
59 # in "OpenStack options" section, and set this value to the IP of your
60 # 'network_interface' as set in the Networking section below.
61 kolla_internal_vip_address: "192.168.43.80"
62 kolla_external_vip_address: "{{ kolla_internal_vip_address }}"
```

Figure 3.4 : Configuration du fichier globals.yml

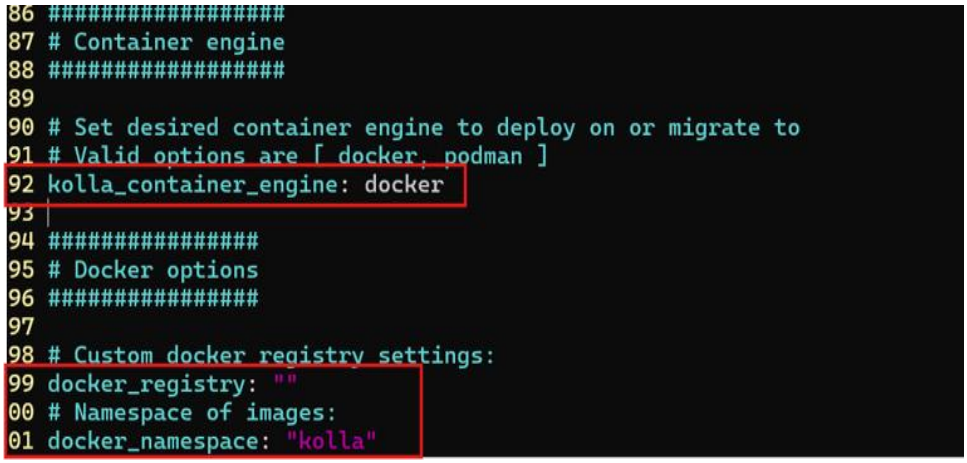


Figure 3.5 : Configuration du fichier globals.yml

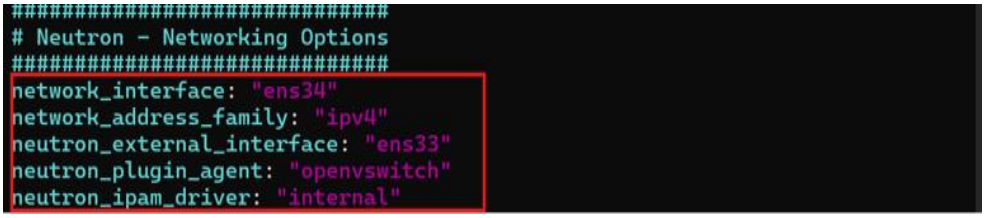


Figure 3.6 : Configuration du fichier globals.yml

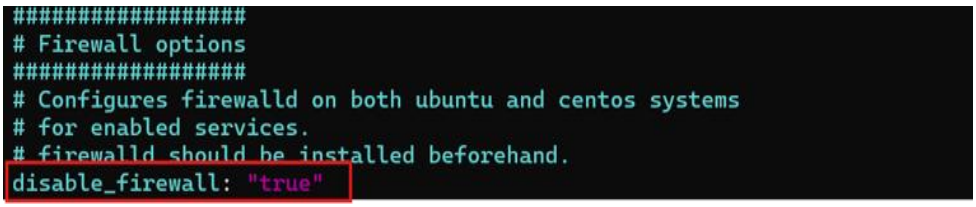


Figure 3.7 : Configuration du fichier globals.yml

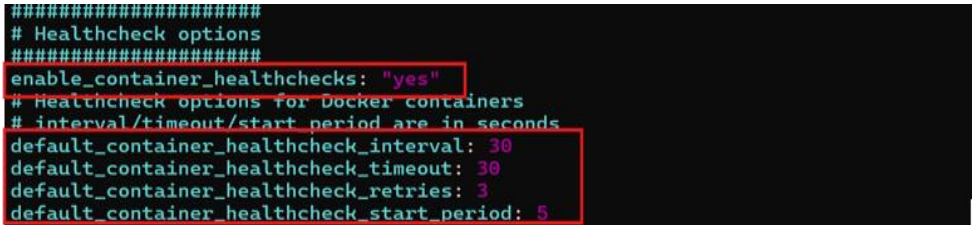


Figure 3.8 : Configuration du fichier globals.yml

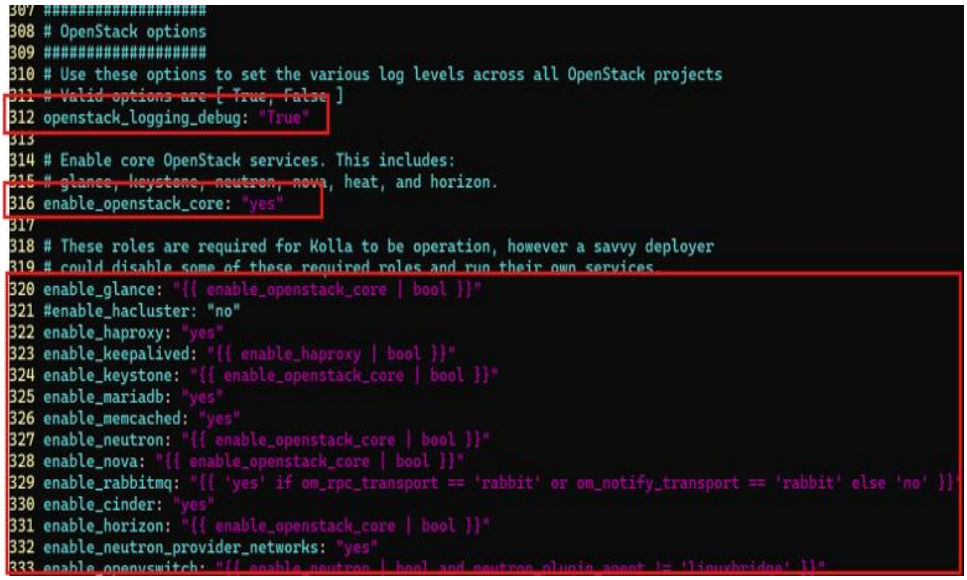


Figure 3.9 : Configuration du fichier globals.yml

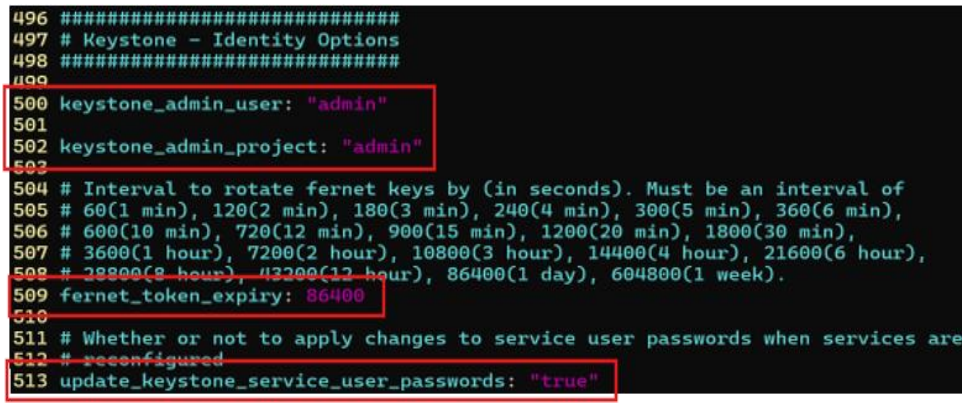


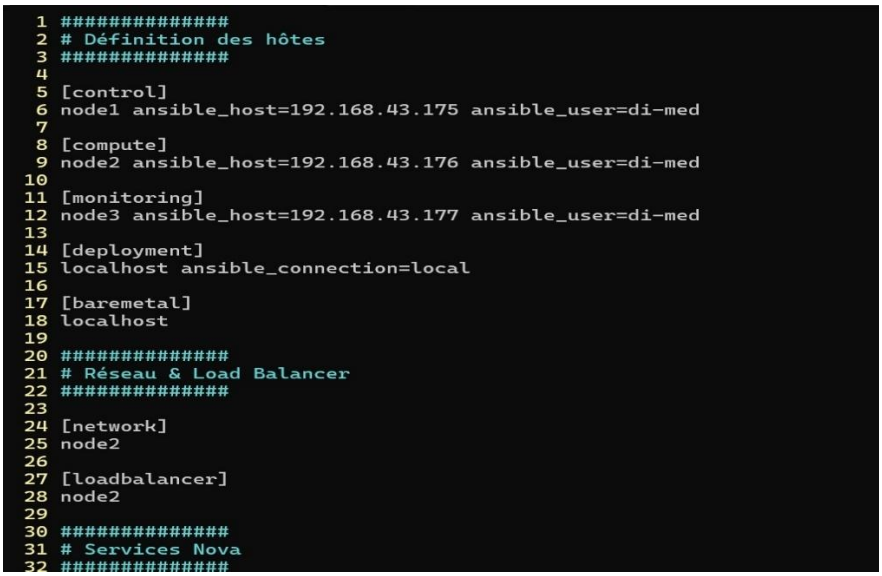
Figure 3.10 : Configuration du fichier globals.yml

- Installation des dépendances et déploiement complet d'Openstack

```
kolla-ansible install-deps
```

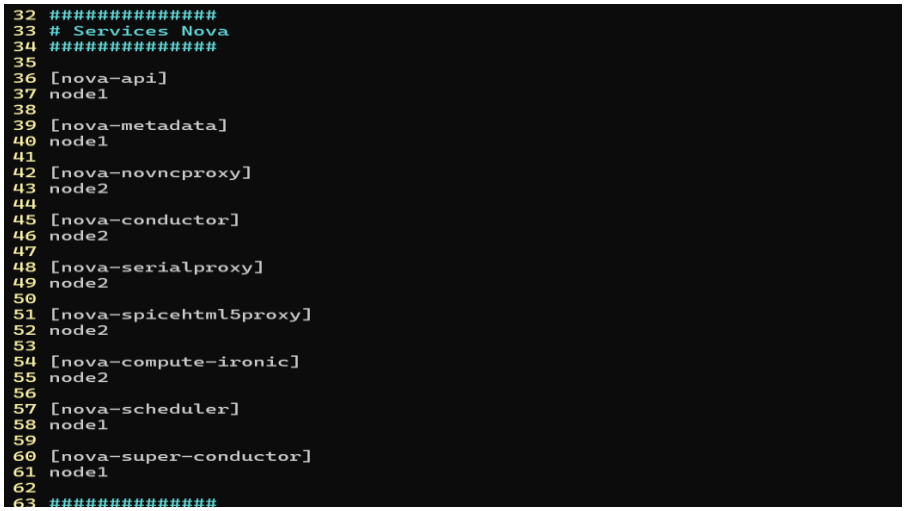
- Edition du fichier d'inventaire multinode

  Le fichier d'inventaire multinode a été édité afin de définir le rôle, l'adresse IP et les paramètres de chaque nœud dans l'architecture Openstack, permettant à Kolla-Ansible d'orchestrer correctement le déploiement sur plusieurs serveurs.



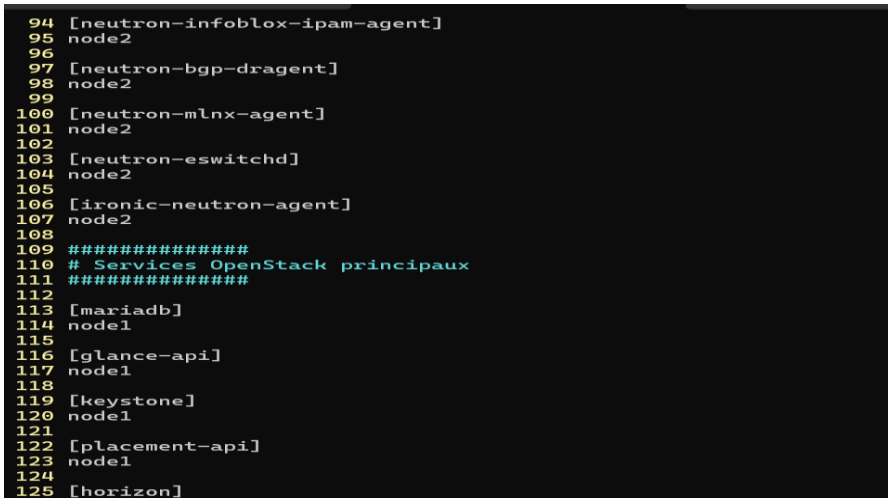
```
###############
# Définition des hôtes
###############

[control]
node1 ansible_host=192.168.43.175 ansible_user=di-med

[compute]
node2 ansible_host=192.168.43.176 ansible_user=di-med

[monitoring]
node3 ansible_host=192.168.43.177 ansible_user=di-med

[deployment]
localhost ansible_connection=local

[baremetal]
localhost

###############
# Réseau & Load Balancer
###############

[network]
node2

[loadbalancer]
node2

###############
# Services Nova
###############
```

Figure 3.11 : Configuration du fichier multinode



```
###############
# Services Nova
###############

[nova-api]
node1

[nova-metadata]
node1

[nova-novncproxy]
node2

[nova-conductor]
node2

[nova-serialproxy]
node2

[nova-spicehtml5proxy]
node2

[nova-compute-ironic]
node2

[nova-scheduler]
node1

[nova-super-conductor]
node1

###############
```

Figure 3.12 : Configuration du fichier multinode



```
[neutron-infoblox-ipam-agent]
node2

[neutron-bgp-dragent]
node2

[neutron-mlnx-agent]
node2

[neutron-eswitchd]
node2

[ironic-neutron-agent]
node2

###############
# Services OpenStack principaux
###############

[mariadb]
node1

[glance-api]
node1

[keystone]
node1

[placement-api]
node1

[horizon]
```

Figure 3.13 : Configuration du fichier multinode

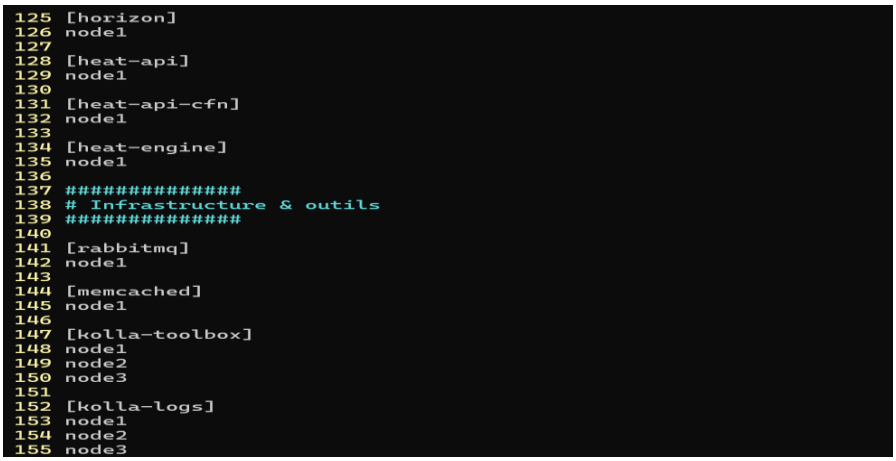


Figure 3.14 : Configuration du fichier multinode

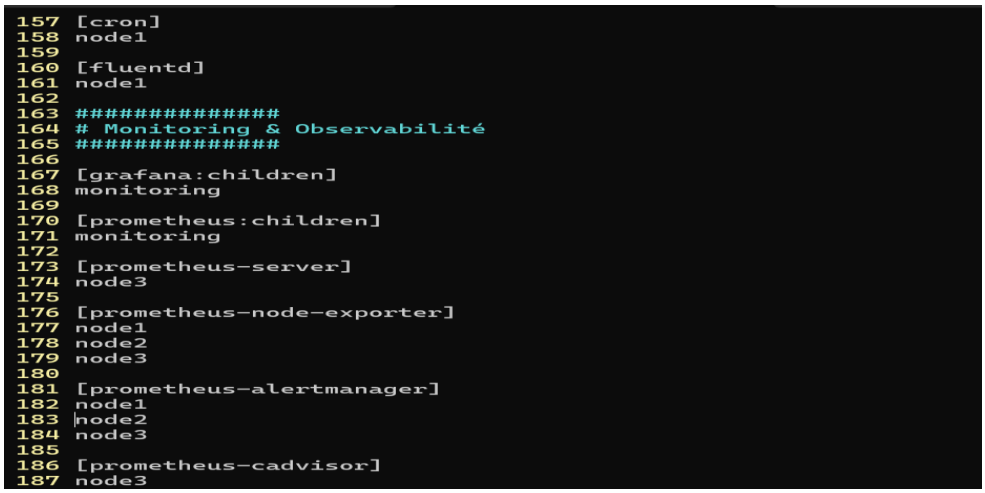


Figure 3.15 : Configuration du fichier multinode

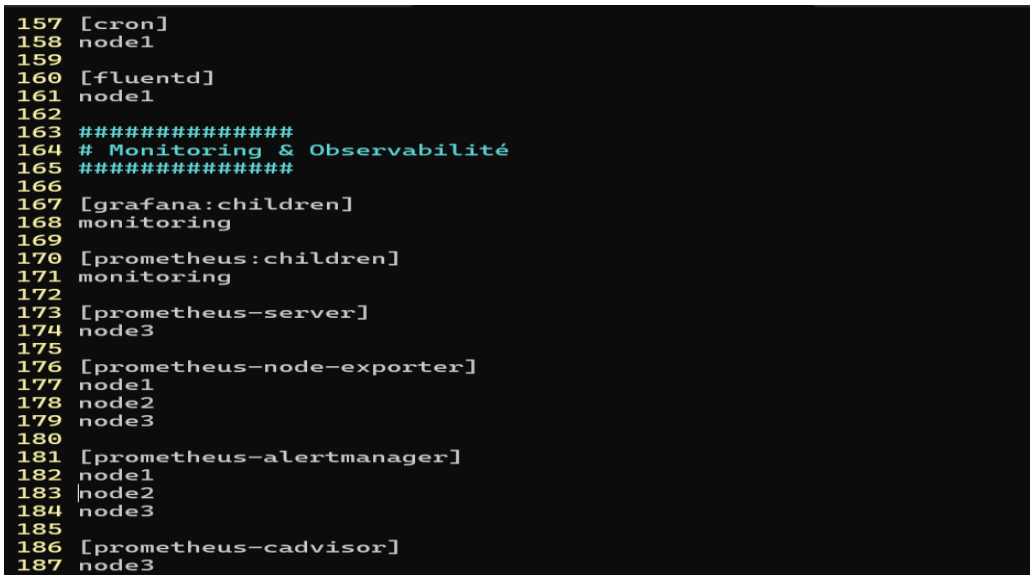


Figure 3.16 : Configuration du fichier multinode

```
189 [prometheus-mysqld-exporter]
190 node3
191
192 [prometheus-memcached-exporter]
193 node3
194
195 [prometheus-openstack-exporter]
196 node3
197
198 [prometheus-elasticsearch-exporter]
199 node3
200
201 [prometheus-blackbox-exporter]
202 node3
203
204 [prometheus-libvirt-exporter]
205 node3
206
207 ###############
208 # TLS & Sécurité
209 ###############
210
211 [tls-backend:children]
212 control
213 network
```

Figure 3.17 : Configuration du fichier multinode

- Configuration du SSH pour les trois nœuds

```
ssh-keygen -t rsa -b 4096
ssh-copy-id di-med@192.168.43.175
ssh-copy-id di-med@192.168.43.176
ssh-copy-id di-med@192.168.43.177
```

- Initialisation des serveurs avec Kolla-Ansible

```
(kolla-venv) di-med@ansible-controller:~$ kolla-ansible bootstrap-servers -i multinode
Bootstrapping servers
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [Gather facts for all hosts] **********************************************************

TASK [Group hosts to determine when using --limit] *****************************************
ok: [node2]
ok: [localhost]
ok: [node1]
ok: [node3]
```

Figure 3.18 : Initialisation des serveurs avec Kolla-Ansible

Après avoir installé toutes les dépendances, cette sortie indique que l'environnement des hôtes est désormais conforme aux prérequis Kolla, prêt à accueillir les conteneurs Openstack.

```
PLAY RECAP *********************************************************************************************************************
localhost                  : ok=37   changed=3    unreachable=0    failed=0    skipped=30   rescued=0    ignored=0
node1                      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node3                      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 3.19 : Résultat de l'initialisation des serveurs

- Ici nous faisons une vérification préalable au déploiement pour voir s'il n'y a pas d'erreur dans le fichier de configuration

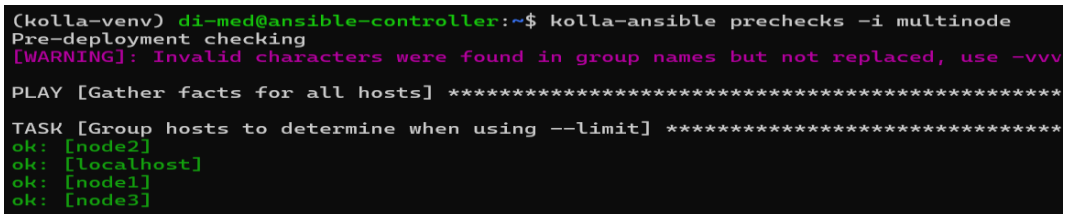


Figure 3.20 : Vérification pré-déploiement des serveurs

Tous les prérequis ont été validés, ce qui garantit que les hôtes sont prêts pour le déploiement sans erreurs bloquantes

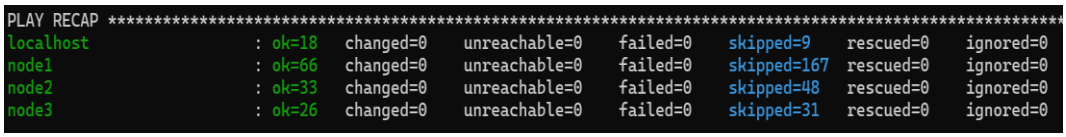


Figure 3.21 : Résultat de la vérification

- Déploiement des services Openstack :

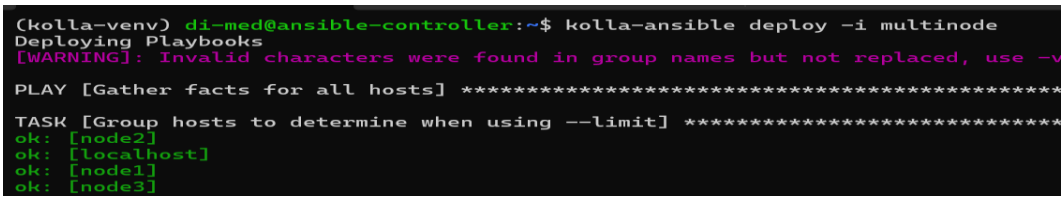


Figure 3.22 : Lancement du déploiement d'Openstack

À l'issue du processus de déploiement, la sortie affichée confirme que la solution Openstack a bien été installée et configurée avec succès.

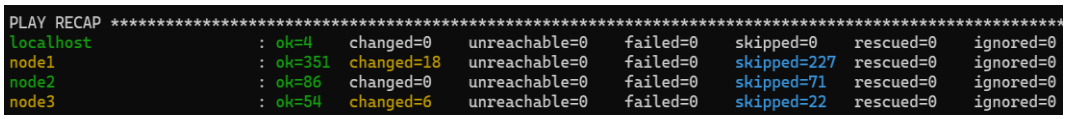


Figure 3.23 : Fin du déploiement d'Openstack

- Post-déploiement d'Openstack :

  Ceci est l'étape de finalisation qui configure l'accès et prépare un environnement utilisable.

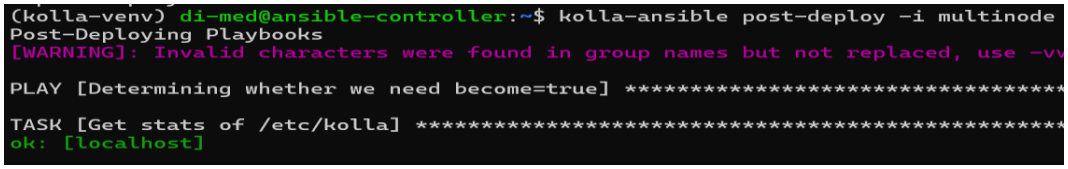


Figure 3.24 : Post-déploiement d'Openstack

Résultat:

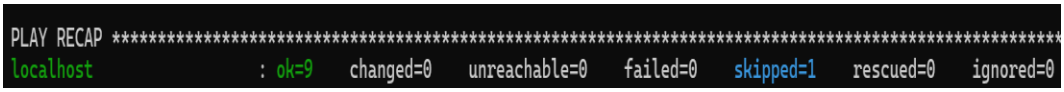


Figure 3.25 : Résultat Post-déploiement

La sortie suivante confirme que les fichiers d'accès tels que admin-openrc.sh et clouds.yaml ont été générés avec succès, permettant l'interaction avec les services Openstack déployés.

- Chargement des variables d'environnement nécessaires pour utiliser la CLI Openstack.

```
source /etc/kolla/admin-openrc.sh
```

- Récupération du mot de passe admin d'Openstack à partir du fichier password.yml.

```
(kolla-venv) di-med@ansible-controller:~$ cat /etc/kolla/passwords.yml | grep keystone_admin
keystone_admin_password: hwszV2QcFpqgzCXUoR1QWK1U1Q68WvOJnv4jSAiN
```

Figure 3.26 : Récupération du mot de passe admin d'Openstack

- Connexion à horizon via l'adresse IP VIP http://192.168.43.80 avec l'utilisateur admin et son mot de passe

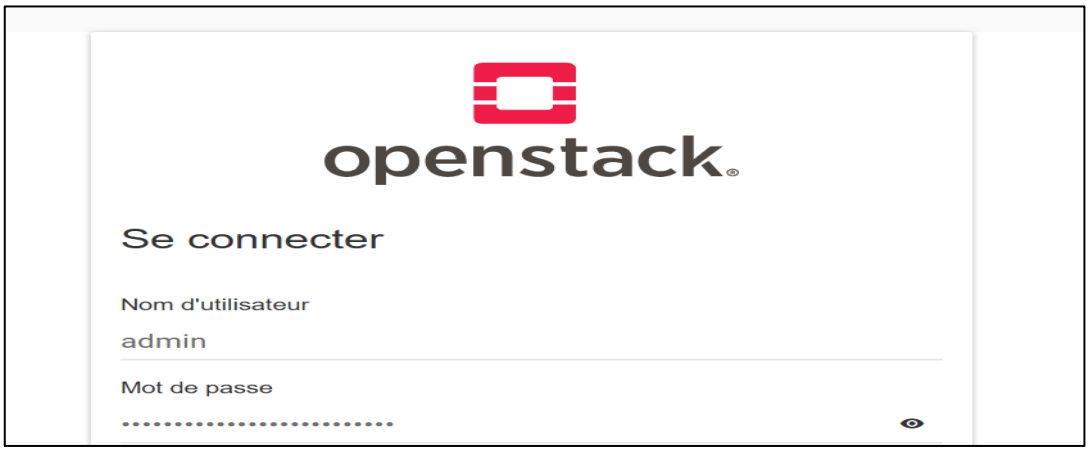


Figure 3.27: Connexion au tableau de bord Horizon

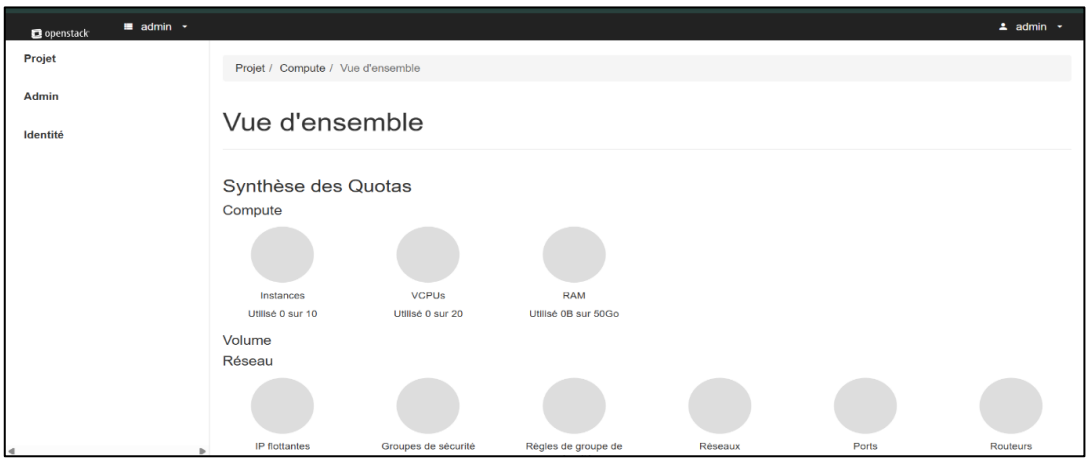


Figure 3.28 : Interface web d'Horizon

3.3.4 Intégration de l'annuaire LDAP avec Keystone

Pour centraliser la gestion des identités, nous avons intégré un annuaire LDAP à Keystone.

- Création du fichier de configuration LDAP : keystone.groupeisi.edu.com.conf

```
mkdir -p /etc/kolla/config/keystone/domains/
vi /etc/kolla/config/keystone/domains/keystone.groupeisi.edu.com.conf
```

Ajouter l'Identity et les directives de notre serveur LDAP

```
[identity]
driver = ldap

[ldap]
url = ldap://192.168.43.173
user = cn=admin,dc=openstack,dc=sn
password = passer
suffix = dc=openstack,dc=sn
user_tree_dn = ou=Utilisateurs,dc=openstack,dc=sn
user_objectclass = inetOrgPerson
user_id_attribute = uid
user_name_attribute = uid
user_mail_attribute = mail
user_enabled_attribute = None
user_enabled_default = True
user_enabled_mask = 0
user_filter = (objectClass=inetOrgPerson)
```

Figure 3.29 : Configuration du fichier keystone.groupeisi.edu.com.conf

- Attribution des permissions

```
sudo chmod -R 755 /etc/kolla/config/keystone
```

- Redéploiement de Keystone

```
(kolla-venv) di-med@ansible-controller:~$ kolla-ansible reconfigure -i multinode
Reconfigure OpenStack service
[WARNING]: Invalid characters were found in group names but not replaced, use -v

PLAY [Gather facts for all hosts] **********************************************

TASK [Group hosts to determine when using --limit] *****************************
ok: [node2]
ok: [localhost]
ok: [node1]
ok: [node3]
```

Figure 3.30 : Redéploiement de Keystone

Nous aurons la sortie suivante à la fin :

```
PLAY RECAP *********************************************************************************************************
localhost                  : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node1                      : ok=351  changed=18   unreachable=0    failed=0    skipped=227  rescued=0    ignored=0
node2                      : ok=86   changed=0    unreachable=0    failed=0    skipped=71   rescued=0    ignored=0
node3                      : ok=54   changed=6    unreachable=0    failed=0    skipped=22   rescued=0    ignored=0
```

Figure 3.31 : Fin du redéploiement de keystone

- Création d'un domaine nommé ldap

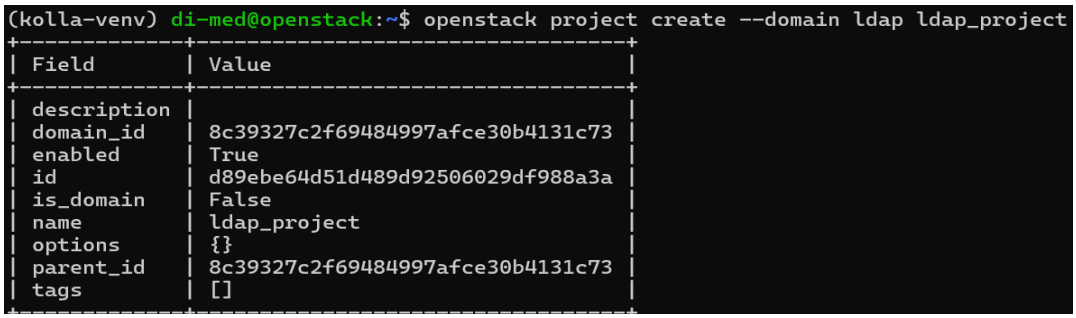
```
openstack domain create groupeisi.edu.com
```

Field	Value
description	
enabled	True
id	6b1c05575ca0472c82193d261c484f2e
name	groupeisi.edu.com
options	{}
tags	[]

Figure 3.32 : Création du domaine groupeisi.edu.com

- Création d'un projet nommé ldap_projet_soutenance

```
openstack domain create –domain groupeisi.edu.com ldap_projet_soutenance
```



```
(kolla-venv) di-med@openstack:~$ openstack project create --domain ldap ldap_project
```

Field	Value
description	
domain_id	8c39327c2f69484997afce30b4131c73
enabled	True
id	d89ebe64d51d489d92506029df988a3a
is_domain	False
name	ldap_project
options	{}
parent_id	8c39327c2f69484997afce30b4131c73
tags	[]

Figure 3.33 : Création du projet ldap_projet_soutenance

- Affectation d'un rôle à nos deux utilisateurs LDAP Ahmad et Dieyna pour qu'ils soient membre du projet ldap_projet_soutenance

```
openstack role add --user dieyna --user-domain groupeisi.edu.com --project ldap_projet_soutenance  --project-domain groupeisi.edu.com member
openstack role add --user ahmad --user-domain groupeisi.edu.com --project ldap_projet_soutenance --project-domain groupeisi.edu.com member
```

- Test d'authentification via CLI

```
openstack --os-auth-url https://192.168.43.80:5000/v3 --os-username dieyna --os-password passer --os-user-domain-name ldap --os-project-name ldap_project  --os-project-domain-name ldap token issue
```

```
(kolla-venv) di-med@openstack:~$ openstack --os-auth-url https://192.168.43.80:5000/v3 \
  --os-username dieyna \
  --os-password passer \
  --os-user-domain-name ldap \
  --os-project-name ldap_project \
  --os-project-domain-name ldap \
  token issue
```

Field	Value
expires	2025-08-11T14:59:00+0000
id	gAAAAABomLO01AKXgfhom1nNJrJWQ2p9FA7lIkpPuHRlBMWDYyGZgcpTAjWC7H_ODr3KkIlClEemPr6ZoaUP0qphIqaE13gid-nX-vicJJCNzsIsyWqy nkixHdGahxS9d1kIqbA6hylbV9hMoIyjH0AqKGjPy1oHJs_GexBvCSqlQVfovAtRg8MMIYK0R27SxWO_4_INQVaCV_WZpFwATKbapjNzZkiaFcfBP- TAizRCfFkAEIFHIFI
project_id	d89ebe64d51d489d92506029df988a3a
user_id	93b4c4ad1ad3b41291b07d3c93152d072cc1dd1dc1d8f4dc2e4110f33c877a1a

Figure 3.34 : Test d'authentification a keystone via CLI

- Vérification de l'ajout des utilisateurs au domaine groupeisi.edu.com

```
(kolla-venv) di-med@ansible-controller:~$ openstack user list --domain groupeisi.edu.com
+----------------------------------------------------------------+--------+
| ID                                                             | Name   |
+----------------------------------------------------------------+--------+
| 84f709875c217b7e189071cbda14fd88e22bedc0012bf9d31026d132c5ce845b | dieyna |
| 107c5ef096df03b422fb8c9f154c4edf6109f8df74fc95c5b78677d37d173b84 | ahmad  |
+----------------------------------------------------------------+--------+
```

Figure 3.35 : Vérification des utilisateurs dans le domaine groupeisi.edu.com

- Activation du champ multi-domaine sur Horizon, suivie d'un redéploiement d'Horizon

  Cela permet d'activer dans l'interface web Horizon la gestion de plusieurs domaines Keystone, avec un menu déroulant pour choisir entre eux.
  Nous avons créé un dossier de configuration personnalisé pour Horizon et y avons copié le fichier de paramètres personnalisés. Ainsi, lors du redéploiement, Kolla-Ansible utilisera ces réglages personnalisés au lieu de ceux par défaut.

```
mkdir -p /etc/kolla/config/horizon

vi /etc/kolla/config/horizon/_9999-custom-settings.py
```

  Ajouter les directives suivantes :

```
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DOMAIN_DROPDOWN = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_API_VERSIONS = {
    "identity": 3,
}
OPENSTACK_KEYSTONE_DOMAIN_CHOICES = (
    ('Default', 'default'),
    ('groupeisi.edu.com', 'groupeisi.edu.com'),
)
```

Figure 3.36 : Configuration du fichier /_9999-custom-settings.py

- Maintenant nous allons reconfigurer Horizon

```
(kolla-venv) di-med@ansible-controller:~$ kolla-ansible reconfigure -i multinode --t horizon
Reconfigure OpenStack service
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [Gather facts for all hosts] **********************************************************

TASK [Group hosts to determine when using --limit] *****************************************
ok: [node2]
ok: [localhost]
ok: [node1]
```

Figure 3.37 : Reconfiguration d'Horizon

La sortie suivante montre que la configuration s'est passe avec succès.

```
PLAY RECAP *********************************************************************************************************
localhost                  : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node1                      : ok=31   changed=0    unreachable=0    failed=0    skipped=17   rescued=0    ignored=0
node2                      : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 3.38 : Fin de la configuration

- Accédons à Horizon avec l'utilisateur LDAP : dieyna

Figure 3.39 : Connexion avec un utilisateur LDAP

L' utilisateur s'est bien connecté et peut maintenant accéder au projet **ldap_project_soutenance**

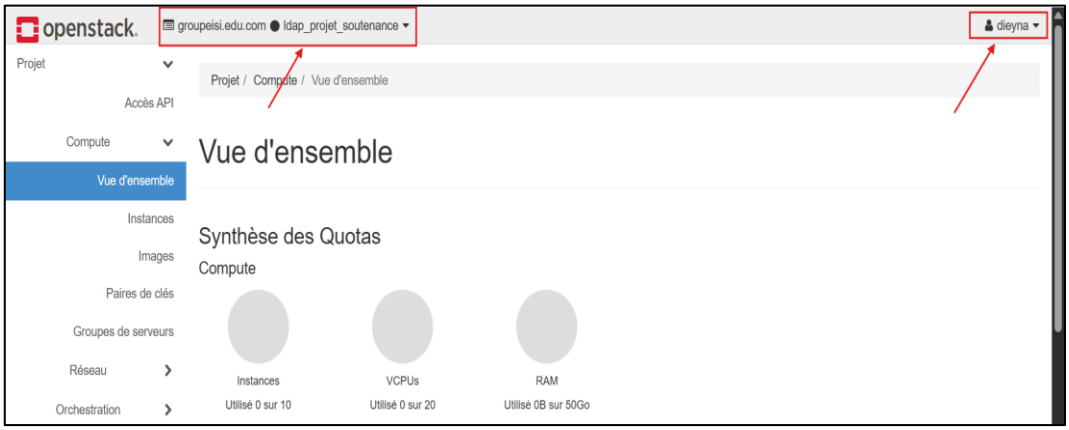


Figure 3.40 : Accès au projet par l'utilisateur LDAP

3.3.5 Mise en place d'une connexion sécurisée via TLS/HTTPS

Nous avons activé le chiffrement TLS pour sécuriser les échanges entre les services.

- Activation du TLS dans globals.yml

```
vim /etc/kolla/globals.yml
```

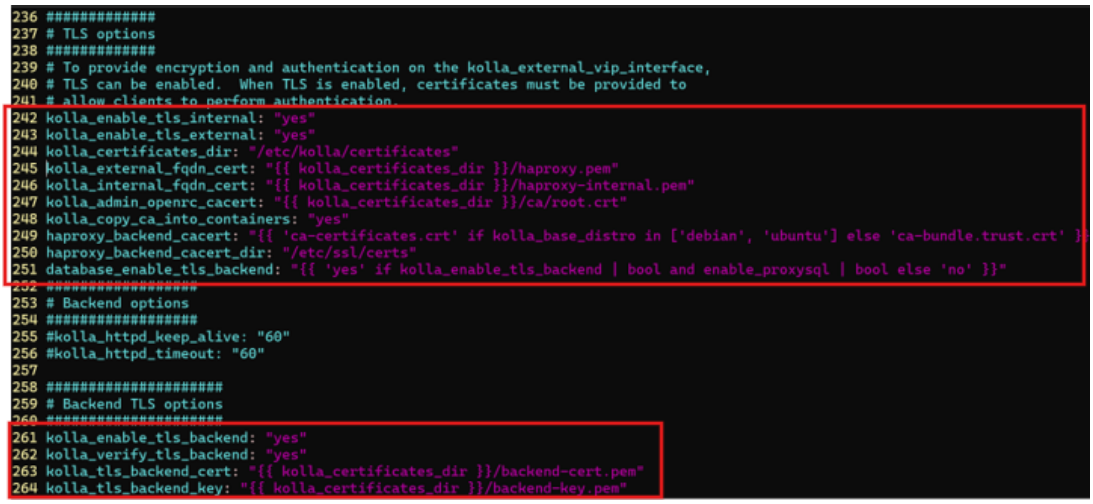


Figure 3.41 : Configuration du fichier globals.yml

- Génération et ajouts des certificats autosignés avec « kolla-ansible certificates »

```
kolla-ansible certificates -i multinode
```

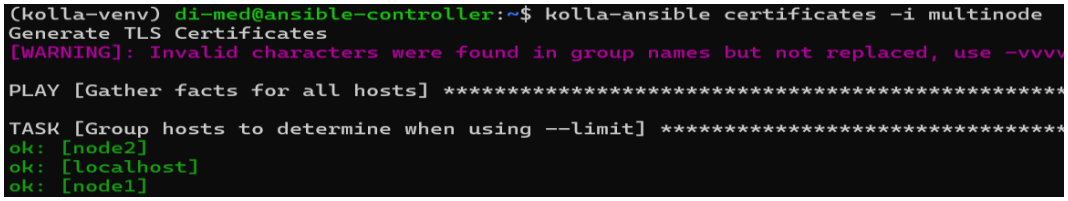


Figure 3.42 : Génération des certificats auto signés

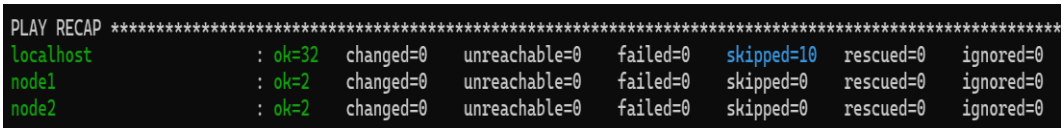


Figure 3.43 : Fin de la génération des certificats

Nous avons ajouté le certificat racine (CA) d'Openstack au magasin de certificats du système. Ainsi, les services et clients du serveur feront confiance aux connexions TLS signées par cette CA.

```
sudo cp /etc/kolla/certificates/ca/root.crt /usr/local/share/ca-certificates/kolla-ca.crt
sudo update-ca-certificates
```

- Reconfiguration du déploiement :

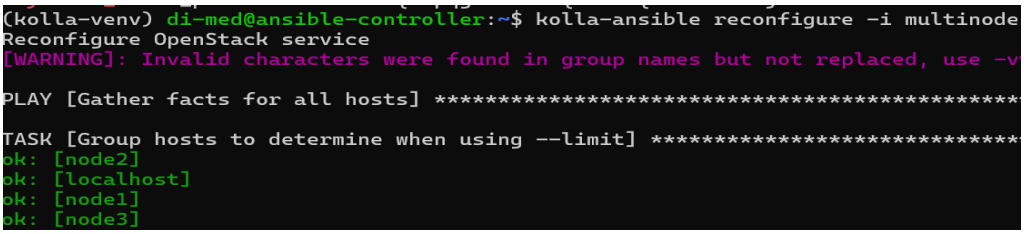


Figure 3.44 : Reconfiguration du déploiement

Nous pouvons désormais accéder à Horizon en utilisant le protocole HTTP

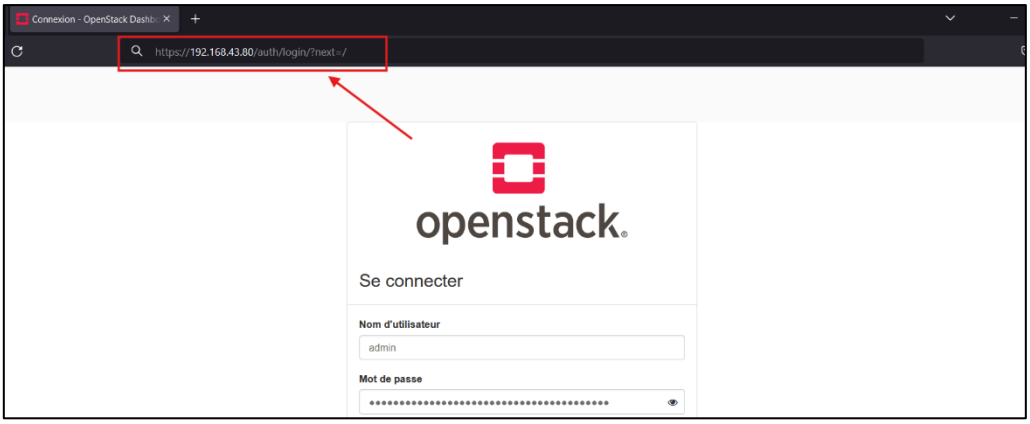


Figure 3.45 : Connexion à Horizon via HTTPS

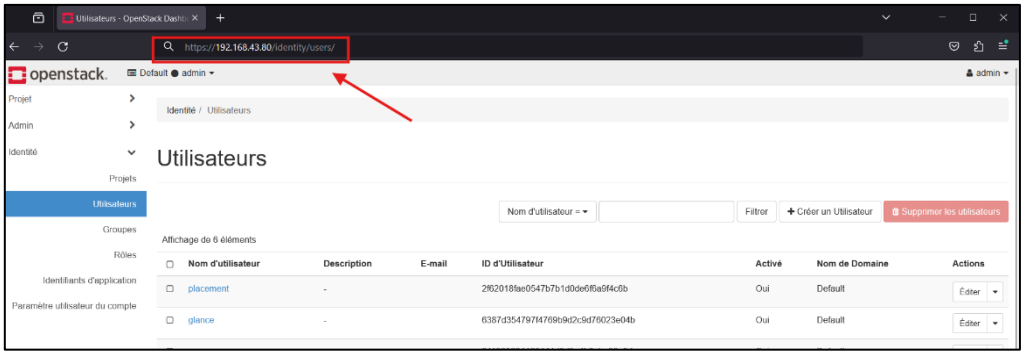


Figure 3.46 : Connexion sécurisée réussie

3.3.6 Déploiement d'instances et validation

Afin de vérifier le bon fonctionnement de notre déploiement , nous avons déployé une instance Ubuntu Jammy. Des réseaux publics et privés ont été mis en place avec Neutron, chacun accompagné de son sous-réseau. Un routeur a été configuré pour assurer la connectivité entre le réseau interne et l'extérieur. De plus, un flavor a été créé afin de spécifier les ressources (CPU, mémoire vive et stockage) allouées aux instances.

- Création d'images et ajout dans Glance
  - Téléchargement de l'image cloud ubuntu-jammy :

```
wget https://cloud-images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img -o ubuntu.img
```

```
(kolla-venv) di-med@openstack:~$ wget https://cloud-images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img -O ubuntu.img
--2025-08-10 17:57:26--  https://cloud-images.ubuntu.com/jammy/current/jammy-server-cloudimg-amd64.img
Resolving cloud-images.ubuntu.com (cloud-images.ubuntu.com)... 185.125.190.40, 185.125.190.37, 2620:2d:4000:1::1a, ...
Connecting to cloud-images.ubuntu.com (cloud-images.ubuntu.com)|185.125.190.40|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 677302272 (646M) [application/octet-stream]
Saving to: 'ubuntu.img'

ubuntu.img                    9%[====>                                                ]  64,56M  3,91MB/s    eta 2m 23s
```

Figure 3.47 : Téléchargement d'une image cloud Ubuntu

- Ajout de l'image dans Glance :

```
openstack image create "ubuntu-jammy" --file ubuntu.img  --disk-format qcow2  --container-format bare   --public
```

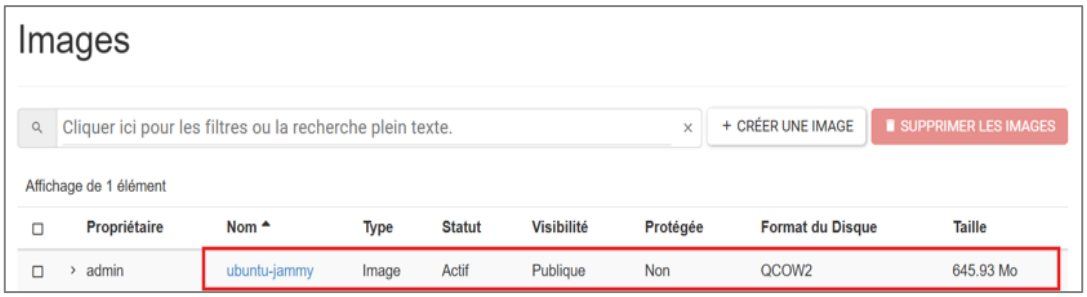


Figure 3.48 : Image cloud Ubuntu

- Création de flavor nommé m1.small avec une mémoire RAM de 2 Go, un stockage de 10Go et 2 vCPUs.

```
openstack flavor create --id 1 --ram 2048 --disk 10 --vcpus 2 m1.small
```

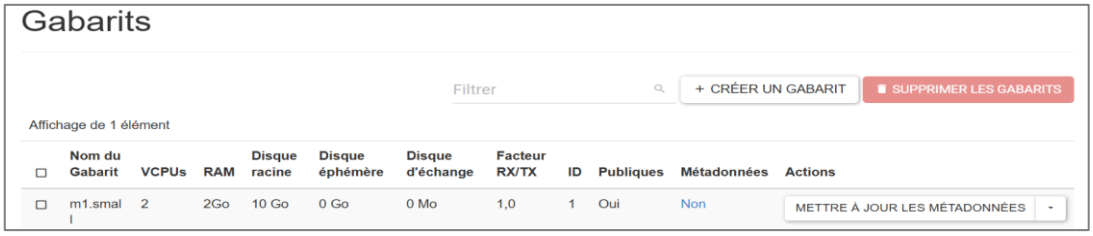


Figure 3.49 : Flavor Openstack

- Création d'un réseau prive dans Openstack :

```
openstack network create private-network
```

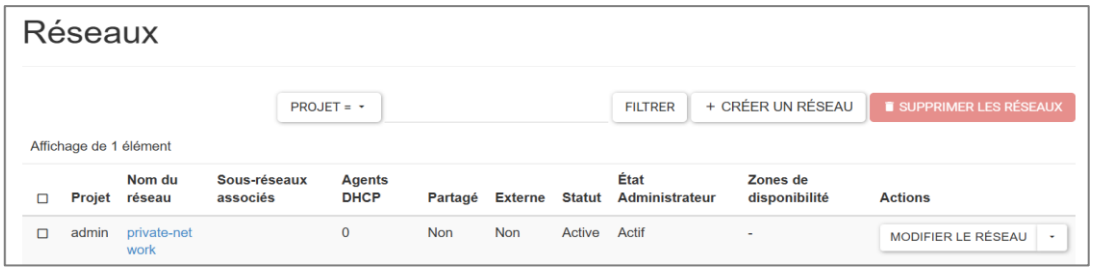


Figure 3.50 : Réseau privé

- Création d'un sous-réseau privé dans Openstack :

```
openstack subnet create private-subnet --network private-network --subnet-range 10.0.0.0/24 --dns-nameserver 8.8.8.8 --gateway 10.0.0.1
```

Figure 3.51 : Sous-réseau privé

- Création d'un réseau public dans Openstack :

```
openstack network create public-net --external --provider-network-type flat --provider-physical-network physnet1
```

Figure 3.52 : Réseau public

- Création d'un sous-réseau public dans Openstack :

```
openstack subnet create public-subnet --network public-net --allocation-pool start=192.168.1.15,end=192.168.1.35 --subnet-range 192.168.1.0/24 --gateway 192.168.1.1 --no-dhcp --dns-nameserver 8.8.8.8
```

Figure 3.53 : Sous-réseau public

- Création d'un routeur dans Openstack :

```
openstack router create demo-router
```

Figure 3.54 : Routeur

- Association des réseaux :

```
openstack router add subnet demo-router private-subnet
openstack router set demo-router --external-gateway public-net
```

Figure 3.55 : Association du Routeur au réseaux public

Ci-dessous, nous présentons la topologie réseau mise en place dans Openstack :

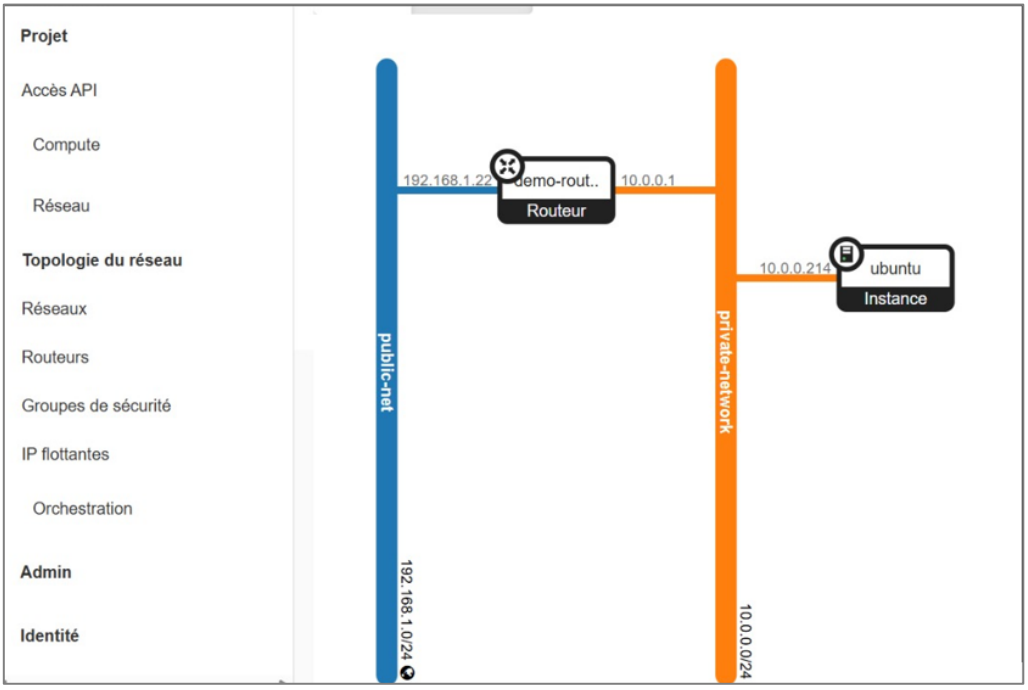


Figure 3.56 : Topologie du cloud

- Création de paires de clés et groupes de sécurité :

Une paire de clés a été générée pour permettre l'authentification sécurisée en SSH vers les instances. Des groupes de sécurité ont également été configurés afin de définir les règles de filtrage réseau, notamment les ports et protocoles autorisés.

- Création d'une paire de clés SSH

```
ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
openstack keypair create ubuntu-key --public-key ~/.ssh/id_rsa.pub
```

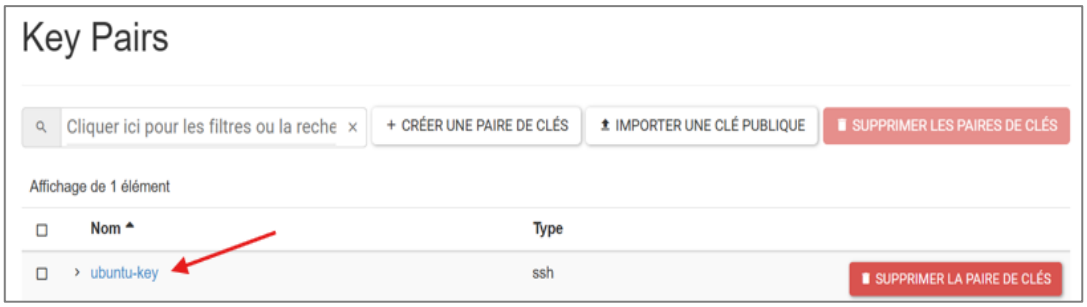


Figure 3.57 : Clé de connexion SSH

- Création d'un groupe de sécurité

```
openstack security group create openstack-security-group --des-
cription "Groupe de sécurité avec ICMP et SSH"
```

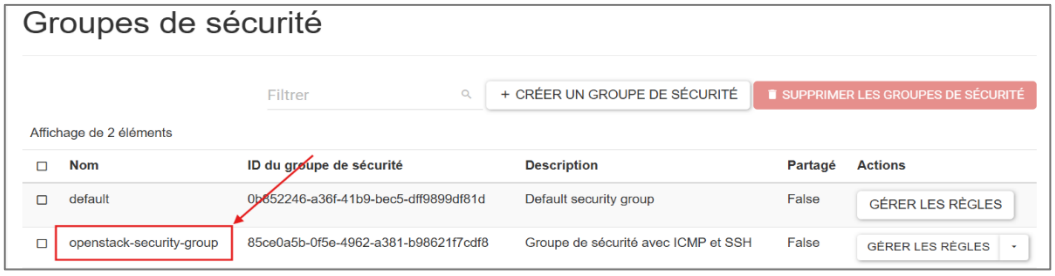


Figure 3.58 : Groupe de sécurité

- Configuration des règles de sécurité réseau dans Opentack pour autoriser les connexions SSH et le trafic ICMP depuis le réseau 192.168.1.0/24

```
openstack security group rule create openstack-security-group --
protocol tcp --dst-port 22 --remote-ip 192.168.1.0/24 --description
"Autorise SSH"
openstack security group rule create openstack-security-group --
protocol icmp --remote-ip 192.168.1.0/24 --description "Autorise
ICMP (ping)"
```

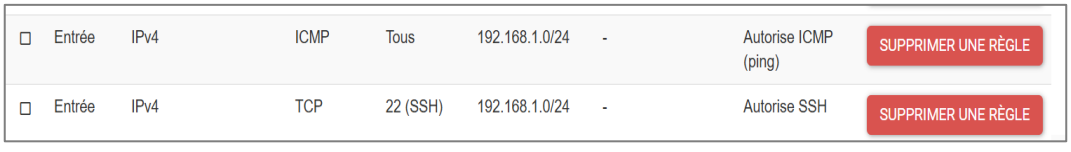


Figure 3.59 : Règles de sécurité

➢ Création et Association d'une IP flottante et lancement de l'instance

L'instance a été déployée puis associée à une adresse IP flottante afin de permettre son accès depuis l'extérieur du réseau privé

- Création d'une Instance Ubuntu

```
openstack server create ubuntu --image ubuntu-jammy --flavor m1.small --key-name ubuntu-key --network private-network  --security-group openstack-security-group
```

Figure 3.60 : Instance Ubuntu

L'instance a été déployée puis associée à une adresse IP flottante afin de permettre son accès depuis l'extérieur du réseau privé

- Création d'une IP flottante

```
FLOATING_IP1=$(openstack floating ip create public-net -f value -c floating_ip_address)
openstack server add floating ip ubuntu $FLOATING_IP1
```

Figure 3.61 : IP flottante

- Test de l'instance

La connectivité de l'instance a été vérifiée à l'aide de commandes de ping et d'accès SSH, confirmant son bon fonctionnement et sa disponibilité sur le réseau.

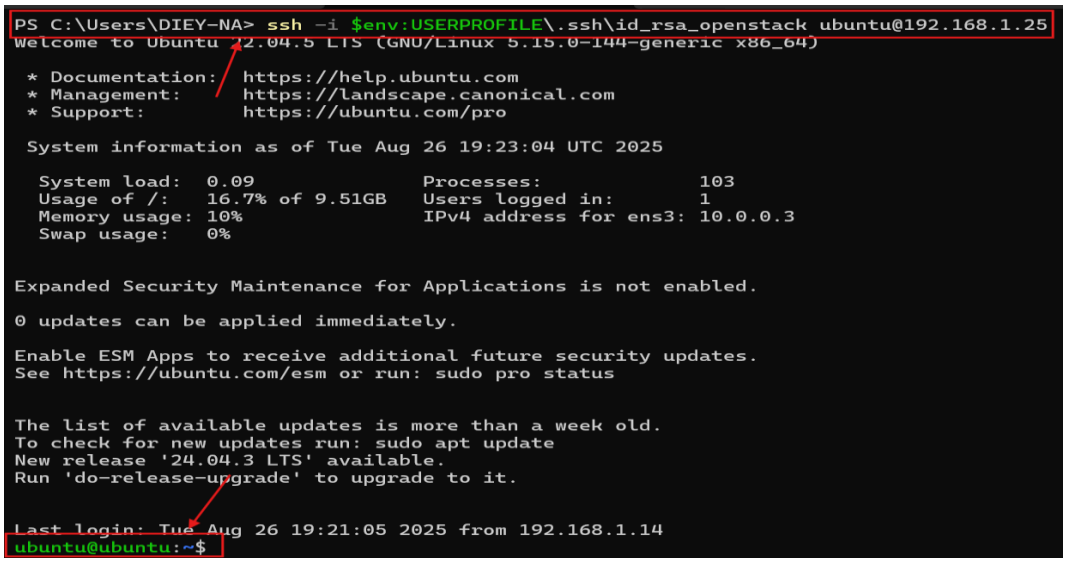


Figure 3.62 : Connexion SSH à l'instance Ubuntu

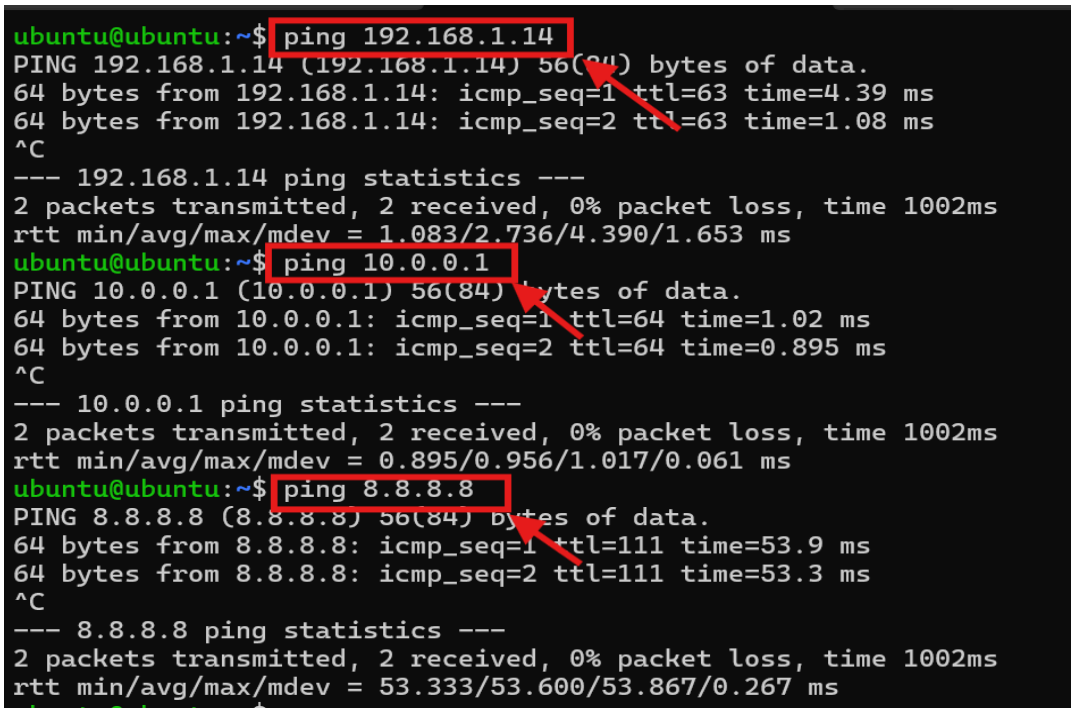


Figure 3.63 : Test ICMP

Pour démontrer la pleine opérationnalité de l'instance, un site web complet basé sur WordPress, connecté à MariaDB, a été déployé. Le site est accessible via

192.168.1.25/wordpress, permettant de vérifier le bon fonctionnement des services web et de la base de données.

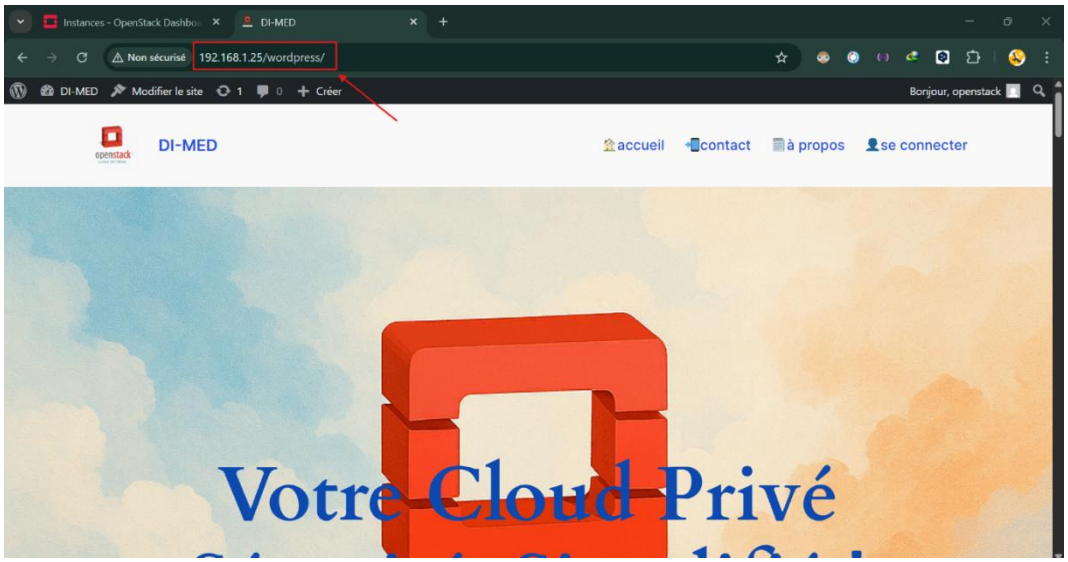


Figure 3.64 : Interface web du site WordPress

3.3.7 Activation des serveurs de supervision et mise en place du monitoring

Les paramètres relatifs aux serveurs de supervision ont été activés dans le fichier globals.yml, notamment pour Grafana et Prometheus, afin d'assurer un suivi complet de l'environnement Openstack

vim /etc/kolla/globals.yml

```
365 enable_grafana: "yes"
366 enable_grafana_external: "{{ enable_grafana | bool }}"
367 enable_prometheus: "yes"
```

Figure 3.65 : Configuration du fichier globals.yml

```
732 ############
733 # Prometheus
734 ############
735 enable_prometheus_server: "{{ enable_prometheus | bool }}"
736 enable_prometheus_haproxy_exporter: "{{ enable_haproxy | bool }}"
737 enable_prometheus_mysqld_exporter: "{{ enable_mariadb | bool }}"
738 enable_prometheus_node_exporter: "{{ enable_prometheus | bool }}"
739 enable_prometheus_cadvisor: "{{ enable_prometheus | bool }}"
740 enable_prometheus_alertmanager: "{{ enable_prometheus | bool }}"
741 enable_prometheus_openstack_exporter: "{{ enable_prometheus | bool }}"
```

Figure 3.66 : Configuration du fichier globals.yml

- Reconfiguration de l'environnement pour le déploiement de Grafana et Prometheus

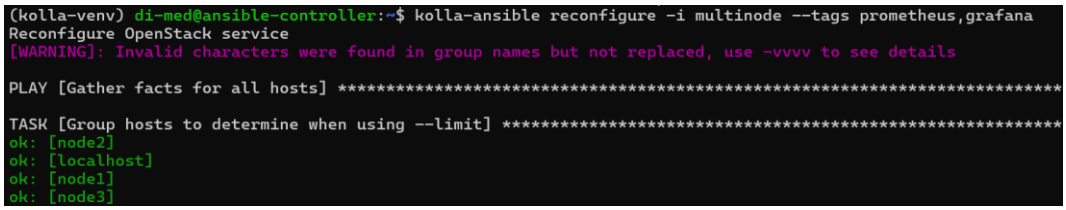


Figure 3.67 : Déploiement de Grafana et Prometheus

- Récupération des mots de passe Prometheus et Grafana :

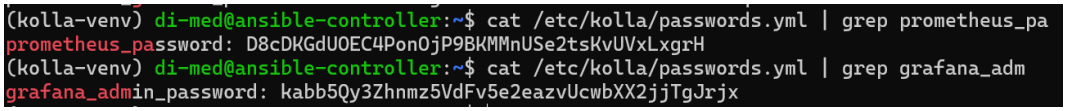


Figure 3.68 : Mots de passe Grafana et Prometheus

L'interface de Grafana est désormais accessible via l'adresse IP VIP 192.168.43.80 sur le port 3000

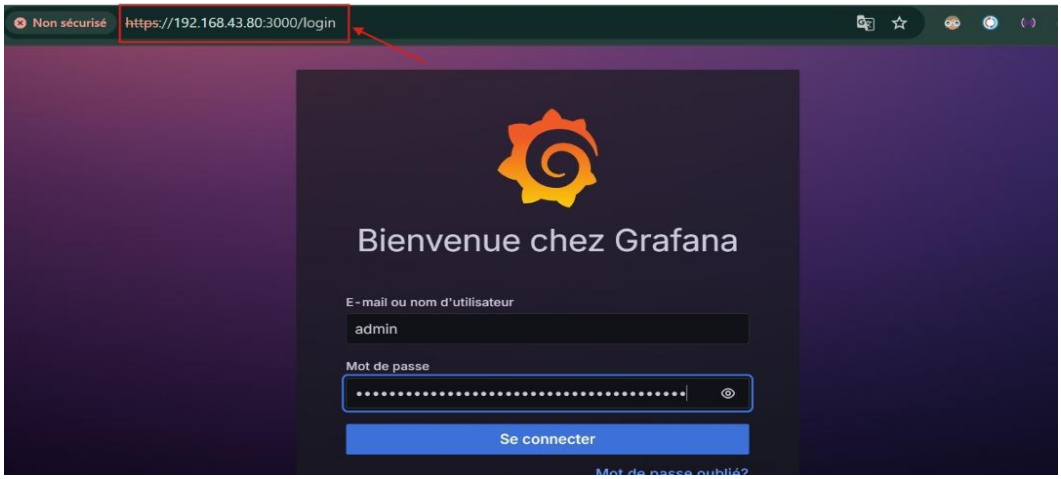


Figure 3.69 : Connexion à Grafana

Il est observé que Prometheus a été automatiquement configuré comme source de données après le déploiement

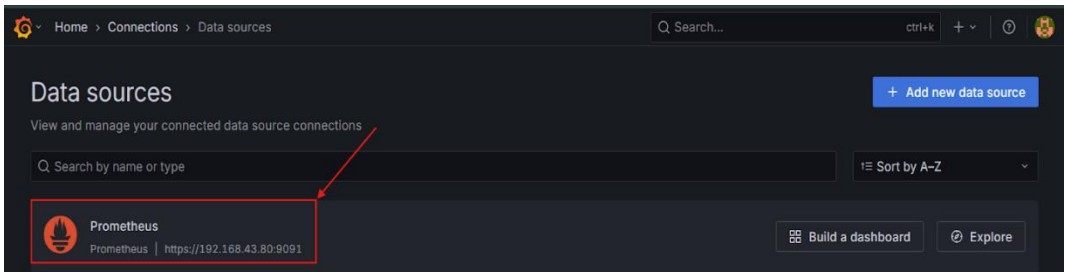


Figure 3.70 : Source de donnée Prometheus

Nous avons ajouté un tableau de bord pour la visualisation des agents et renseigné l'UID 1860 pour importer le node-exporter pour la collecte des donnés.

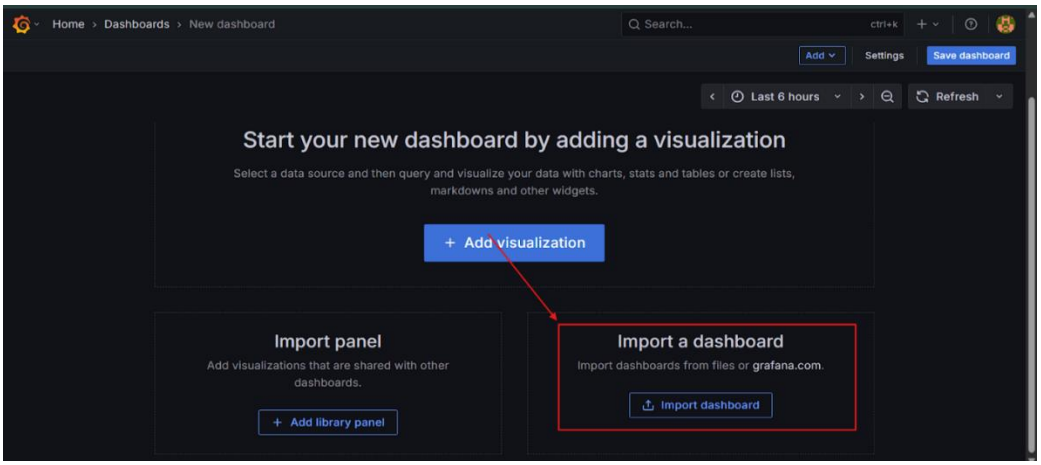


Figure 3.71 : Création des tableaux de bord

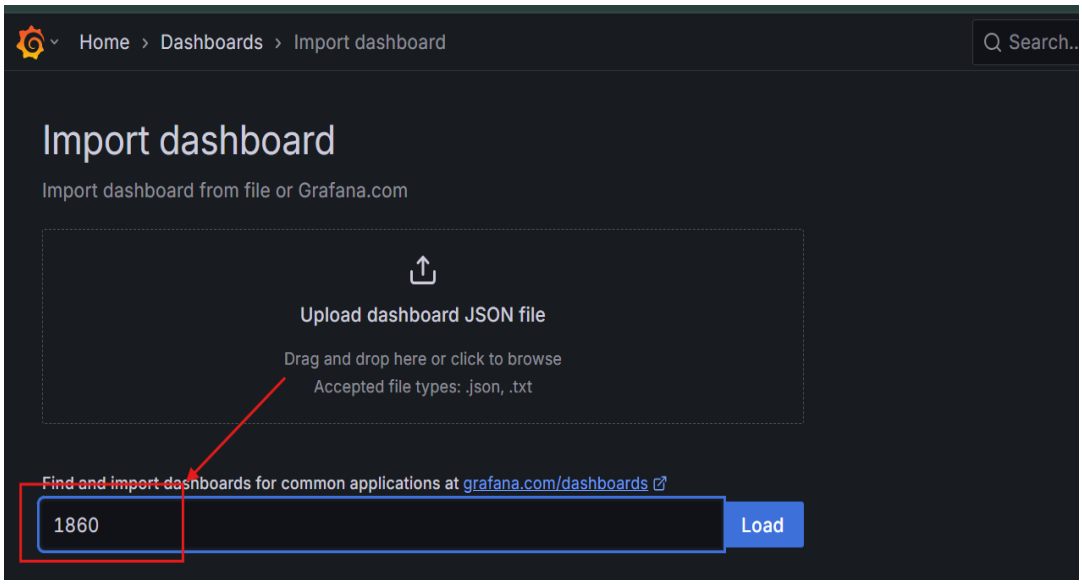


Figure 3.72 : Importation d'un dashboard

- Sélection de Prometheus comme source et importation

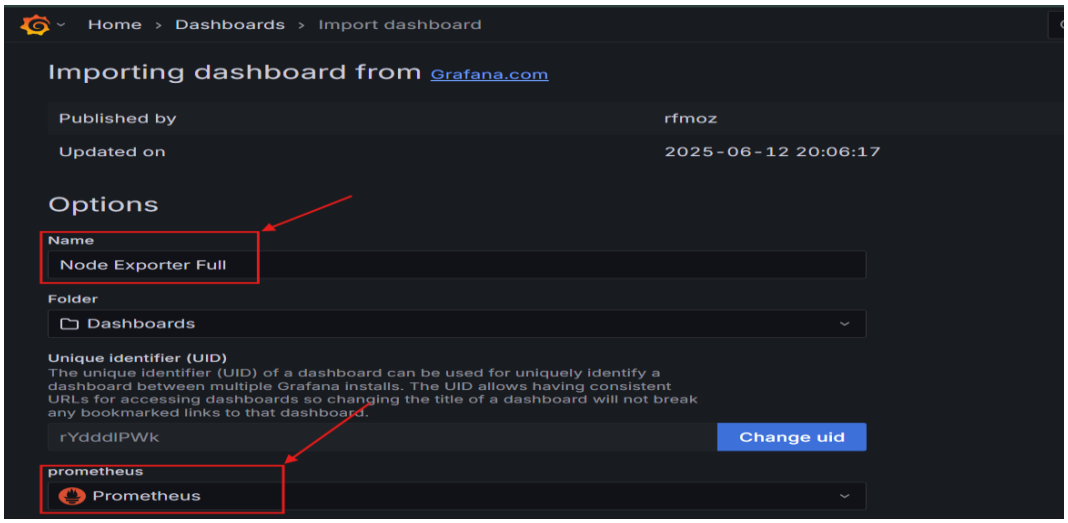


Figure 3.73 : Création du Dashboard

Il est désormais possible de visualiser les métriques de nos 3 nœuds qui ont été collectés via Prometheus.

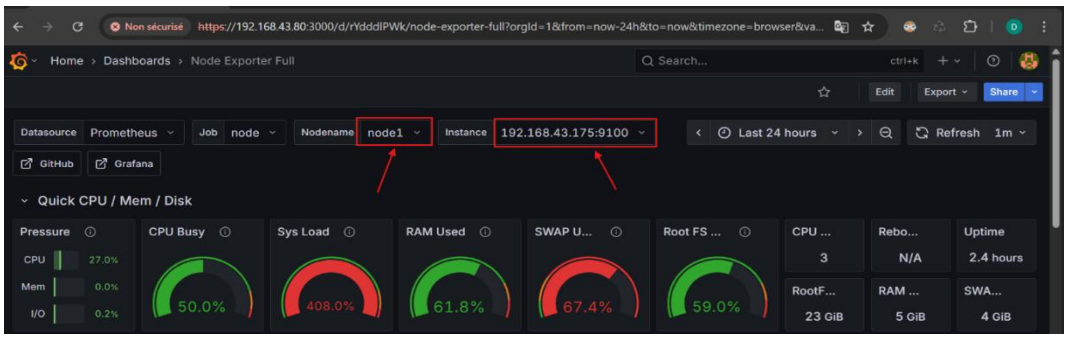


Figure 3.74 : Tableau de bord du 1er nœud Openstack

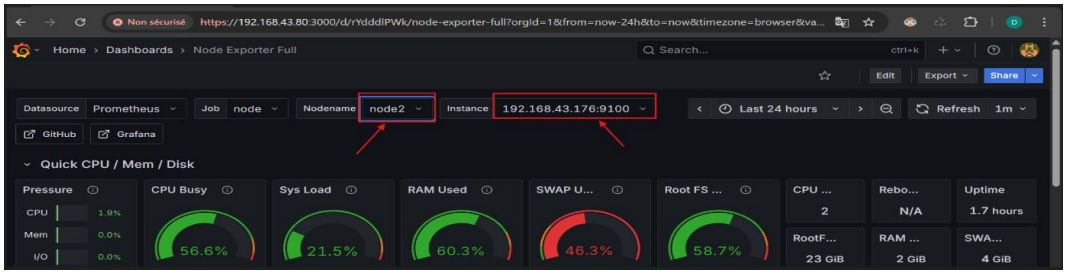


Figure 3.75 : Tableau de bord du 2e nœud Openstack

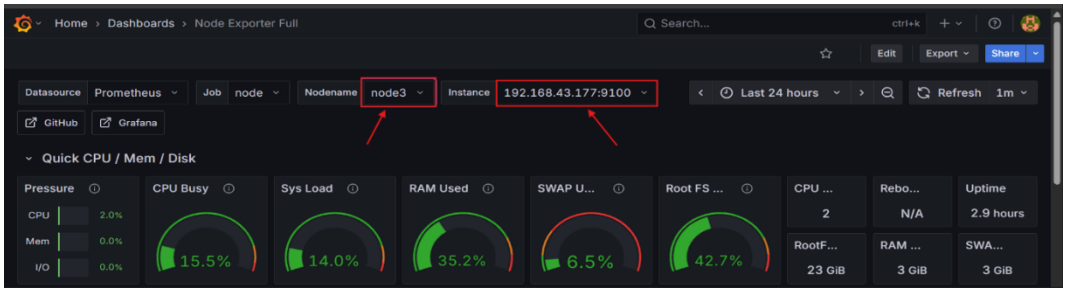


Figure 3.76 : Tableau de bord du 3e nœud Openstack

Par la suite, un tableau de bord personnalisé pour Openstack a été créé, affichant le nombre d'instances, le nombre de VCPU utilisés et la quantité de RAM consommée.

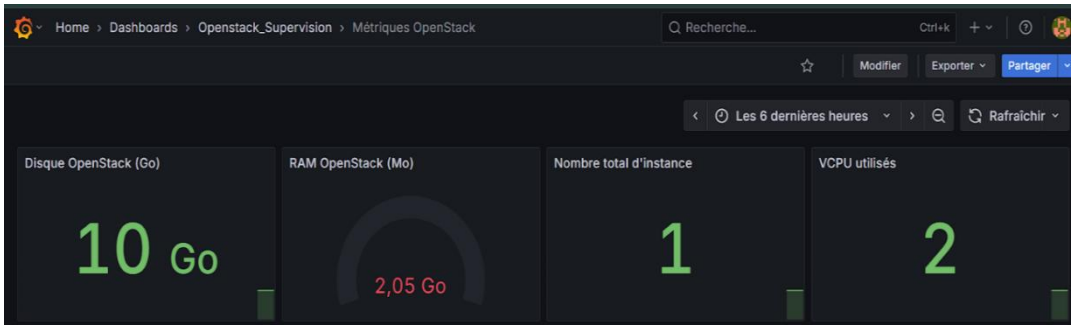


Figure 3.77 : Tableau de bord des métriques dans Openstack

- Configuration d'alertes dans Grafana

Pour superviser les nœuds Openstack, une alerte sur l'utilisation de la mémoire a été définie dans Grafana. La requête utilisée: **(node_memory_MemTotal_bytes - node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100** permet de calculer le pourcentage de mémoire utilisée. Une alerte se déclenche lorsque ce pourcentage dépasse un seuil critique, assurant ainsi la stabilité et la performance des services.

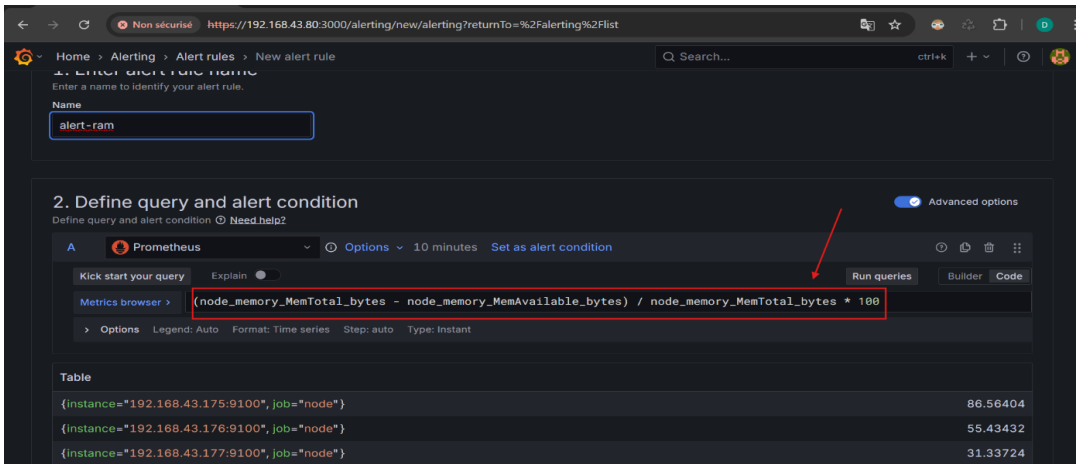


Figure 3.78 : Définition d'une condition alerte

- Déclenchement de l'alerte

  Lorsque le pourcentage de mémoire utilisée dépasse le seuil configuré, l'alerte passe en état « Firing », indiquant qu'une action de surveillance ou de remédiation doit être entreprise afin d'éviter une saturation ou un dysfonctionnement du serveur.

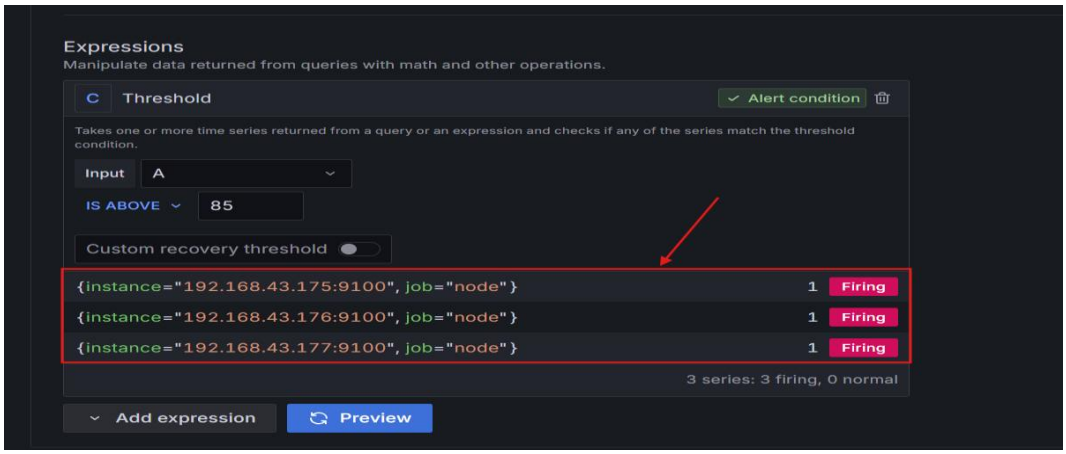


Figure 3.79 : Déclenchement de l'alerte

- Création d'un point de contact

  Un point de contact a été configuré dans Grafana afin de recevoir automatiquement les notifications lorsque l'alerte passe en état « Firing ». Cela permet de garantir une intervention rapide en cas d'incident.

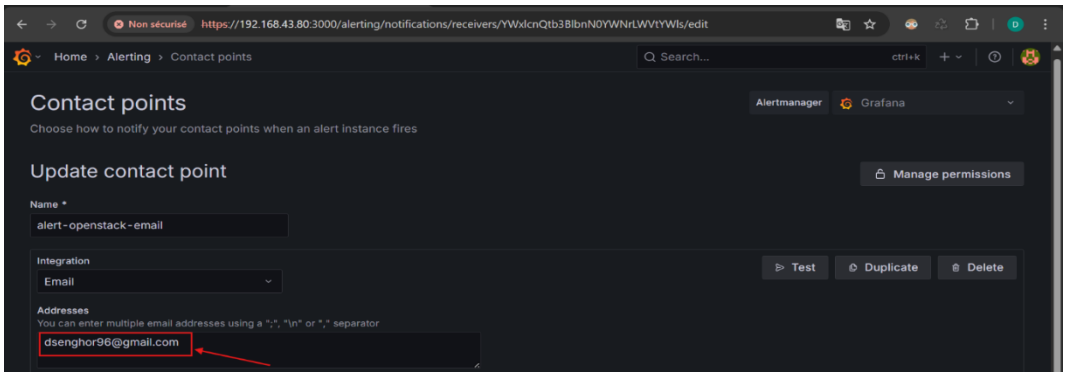


Figure 3.80 : Création du point de contact

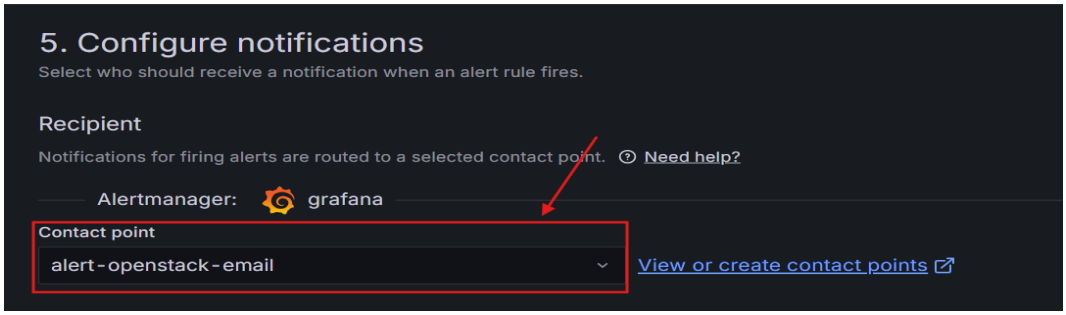


Figure 3.81 : Ajout du point de contact

- Création d'un fichier « override » pour Grafana pour configurer SMTP

  Un fichier « override » a été créé pour Grafana afin de configurer le service SMTP, permettant l'envoi des alertes par e-mail.

- Création du dossier Grafana et édition du fichier grafana.ini

```
mkdir -p /etc/kolla/config/grafana/
vi /etc/kolla/config/grafana/grafana.ini
```

```
[smtp]
enabled = true
host = smtp.gmail.com:587
user = dsenghor96@gmail.com
password = rhhm uuug pzuy howk
from_address = dsenghor96@gmail.com
from_name = "Grafana Alerts"
skip_verify = true
startTLS_policy = OpportunisticStartTLS
```

Figure 3.82 : Configuration de Grafana

- Réception d'alerte dans la boite mail indiquer

  La réception de l'alerte a été vérifiée dans la boîte mail configurée, confirmant le bon fonctionnement du système de notification par e-mail.

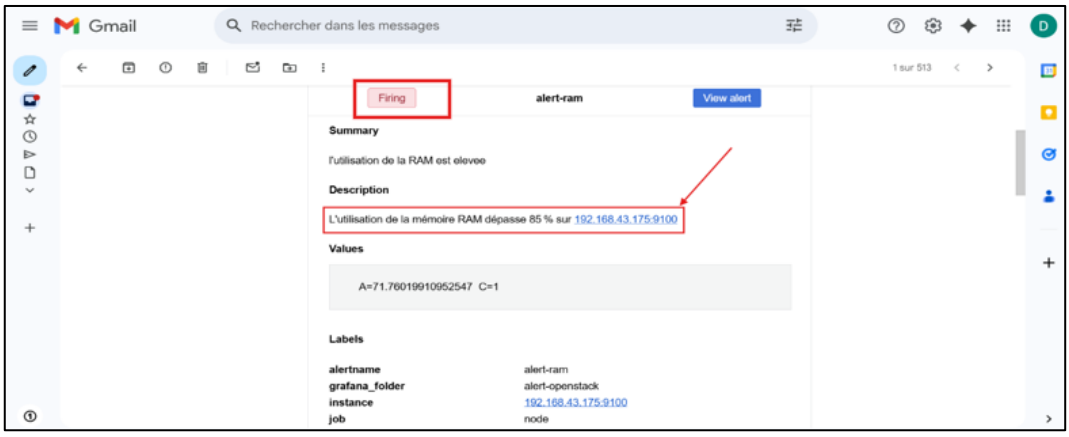


Figure 3.83 : Réception d'une alerte via E-mail

- Une seconde notification est également envoyée lorsque l'alerte est résolue, attestant du retour à la normale.

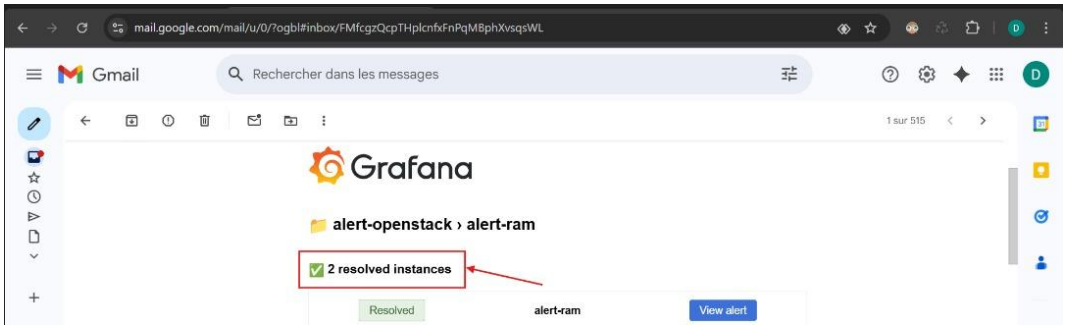


Figure 3.84 : Alerte Grafana résolue

3.4 Technologies utilisées

Pour mener à bien ce projet, nous avons mobilisé un ensemble diversifié de logiciels et d'outils technologiques, essentiels à la mise en place et à la gestion

de notre environnement de cloud privé. Ci-dessous se trouve une présentation détaillée des outils employés :

- **Machine physique Dell**
    - Processeur : Intel Core i5
    - Mémoire vive (RAM) : 12 Go
    - Stockage : Disque SSD de 1To
    - Rôle : Héberge l'hyperviseur et assure la puissance de calcul nécessaire pour faire fonctionner l'environnement virtuel Openstack.
- **VMware Workstation**
    - Rôle : Création et gestion des machines virtuelles hébergées sur la machine physique, permettant l'exécution simultanée de plusieurs systèmes d'exploitation pour les besoins du projet.
- **Ubuntu Server 24.04 LTS**
    - Rôle : Système d'exploitation principal des serveurs virtuels. Il offre une base stable, sécurisée et optimisée pour l'installation et l'exploitation d'Openstack.
- **Windows 11**
    - Rôle : Utilisé comme poste client pour tester l'accessibilité et le bon fonctionnement des services du cloud privé depuis un environnement utilisateur différent
- **Openstack**
    - Rôle : Fournit une suite complète de services (gestion des ressources, instances, réseaux, stockage, etc.) pour la construction et la gestion du cloud privé.
- **Kolla-Ansible**
    - Rôle : Automatisation du déploiement d'Openstack dans des conteneurs Docker, facilitant la gestion et la mise à jour de l'infrastructure.
- **Prometheus**
    - Rôle : Collecte des métriques système et applicatives afin de surveiller en temps réel l'état de l'infrastructure.
- **Grafana**
    - Rôle : Visualisation et analyse des données collectées par Prometheus à travers des tableaux de bord dynamiques et interactifs.

- **phpMyAdmin**
    - Rôle : Interface web permettant la gestion, l'administration et l'exploration graphique des bases de données.
- **Docker**
    - Rôle : Exécution d'Openstack et de ses composants dans des conteneurs isolés, garantissant portabilité et facilité de maintenance.
- **Open vSwitch (OVS)**
    - Rôle : Gestion avancée du trafic réseau entre les machines virtuelles et les composants d'Openstack, avec prise en charge de fonctionnalités réseau telles que VLAN, tunneling et routage.

CONCLUSION GÉNÉRALE

1. Vérification des objectifs

L'objectif principal de ce projet était de concevoir et de déployer une solution de cloud computing privé sécurisé basée sur Openstack. Les différentes étapes prévues ont été réalisées avec succès :

- ✓ Analyse des besoins : identification précise des ressources matérielles et logicielles requises pour garantir la faisabilité de la solution.
- ✓ Configuration et déploiement d'Openstack : installation complète de la plateforme et de ses composants essentiels.
- ✓ Intégration LDAP avec Keystone : mise en place d'une authentification centralisée pour une gestion optimisée des identités et des rôles.
- ✓ Implémentation TLS/HTTPS : sécurisation des accès aux services via un chiffrement SSL/TLS fiable.
- ✓ Déploiement et validation d'instances : création et configuration d'instances fonctionnelles confirmant la stabilité de l'environnement.
- ✓ Sécurisation et supervision : mise en œuvre de règles de contrôle d'accès, configuration des pare-feu et déploiement d'outils de monitoring pour une surveillance proactive.

La réalisation de ces objectifs n'a pas été exempte de défis. Nous avons dû faire face à la configuration initiale complexe d'Openstack, à la recherche d'une version stable adaptée au déploiement avec Kolla-Ansible, ainsi qu'à la sélection

de versions compatibles de Docker et d'Ansible. L'intégration de l'annuaire LDAP avec Keystone a nécessité plusieurs ajustements pour assurer une authentification fluide, et la gestion des certificats TLS/HTTPS a demandé une attention particulière afin de garantir la sécurité des communications. Ces obstacles ont été surmontés grâce à une recherche approfondie, des tests successifs et une collaboration efficace, renforçant ainsi notre expertise technique et notre capacité à résoudre des problèmes complexes.

2. Intérêt personnel

Sur le plan personnel, ce projet nous a permis de renforcer nos compétences techniques et notre autonomie. Nous avons acquis une expérience concrète dans l'installation, la configuration et la sécurisation d'Openstack, tout en développant une maîtrise des technologies connexes telles que LDAP, TLS/HTTPS et les outils de supervision. La gestion des incidents et la résolution des problèmes rencontrés nous ont appris à adopter une démarche méthodique, rigoureuse et orientée solution.

Au-delà des aspects techniques, ce travail a consolidé notre capacité à collaborer efficacement, à documenter nos actions et à mener un projet complexe de bout en bout. Ces acquis constituent un atout majeur pour notre future carrière dans le domaine des infrastructures cloud et de la cybersécurité.

En perspective, l'infrastructure mise en place pourrait évoluer en intégrant un système de gestion centralisée des logs avec Loki et Promtail, permettant une meilleure traçabilité et une supervision plus fine de l'environnement. Elle pourrait également bénéficier d'un niveau accru d'automatisation via Ansible, ainsi que de solutions avancées d'orchestration et de haute disponibilité. Ces améliorations ouvriraient la voie à une plateforme plus performante, résiliente et conforme aux meilleures pratiques du cloud computing moderne.

Bibliographie

I. Ouvrages

Belamaric, Richard, & Paden, *Learning OpenStack*, Birmingham, Packt Publishing, 2015, 404 pages.

Cohen, Michael, *OpenStack Cloud Security*, Birmingham, Packt Publishing, 2015, 286 pages.

Jordan, Cody B., & Taylor, Raymond, *OpenStack Administration with Ansible*, Birmingham, Packt Publishing, 2018, 360 pages.

Silverman, Ben, & Solberg, Michael, *OpenStack for Architects: Design Production-Ready Private Cloud Infrastructure*, 2nd Edition, Birmingham, Packt Publishing, 2018, 256 pages.

Khedher, Omar, *Mastering OpenStack: Implement the Latest Techniques for Designing and Deploying an Operational, Production-Ready Private Cloud*, 3rd Edition, Birmingham, Packt Publishing, 2025, 392 pages.

II. Mémoires

MOUSSIDEN Meriem, *Etude sur la sécurité d'OpenStack*, Université SAAD DAHLAB de BLIDA, Année Universitaire 2021-2022, 91 pages.

AHMED Slim, *Mise en place d'une infrastructure cloud basée sur OpenStack*, TEKUP, 2017-2018, 77 pages.

MOUHYADIN Hassan, *Etude et mise en place d'une solution de cloud computing sécurisé*, ISI, 2021-2023, 136 pages.

Webographie

https://www.youtube.com/watch?v=b-XgSPuedro&list=LL&index=15 Consulté le 12 mars 2025 à 14h25

https://docs.openstack.org/project-deploy-guide/kolla-ansible/2024.1/quickstart.html Consulté le 18 mars 2025 à 02h05

https://github.com/guolunwei/kolla-ansible-2023.1-jammy/blob/main/install.sh Consulté le 25 mars 2025 à 20h10

https://stackoverflow.com/questions/68038263/failed-to-create-openstack-instance Consulté le 02 avril 2025 à 9h30

https://docs.openstack.org/kolla-ansible/latest/admin/tls.html Consulté le 08 avril 2025 à 21h05

https://openstack.goffinet.org/02-00-projet-openstack#1-pr%C3%A9sentation Consulté le 15 avril 2025 à 10h45

https://www.redhat.com/fr/topics/openstack Consulté le 22 avril 2025 à 22h30

https://miakassissa.com/comment-installer-openstack-ubuntu/ Consulté le 29 avril 2025 à 00h15

https://docs.openstack.org/kolla-ansible/latest/reference/logging-and-monitoring/index.html Consulté le 05 mai 2025 à 13h15

https://docs.openstack.org/kolla-ansible/latest/user/troubleshooting.html Consulté le 12 mai 2025 à 18h50

https://stackoverflow.com/questions/69490712/openstack-error-unable-to-establish-connection-to-endpoint Consulté le 20 mai 2025 à 7h40

https://serverfault.com/questions/589865/connectionerror-errno-111-connection-refused-openstack-installation-through Consulté le 28 mai 2025 à 12h30

https://stackoverflow.com/questions/68135710/deploy-kolla-ansible-openstask-wuth-ldap-integration Consulté le 05 juin 2025 à 14h10

https://bugs.launchpad.net/kolla-ansible/+bug/2054923 Consulté le 12 juin 2025 à 11h55

https://docs.openstack.org/kolla-ansible/latest/reference/shared-services/horizon-guide.html Consulté le 20 juin 2025 à 16h45

Table des matières