République Tunisienne
Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et des Technologies de l'Information et de la Communication

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Département de Génie Informatique et de Mathématiques Appliquées

Année universitaire : 2024-2025

# Rapport de Stage Technicien

***presenté à :***

École Nationale d'Ingénieurs de Sfax
(Département de Génie Informatique et de Mathématiques Appliquées)

***Elaboré par :***

**Hiba Hriz**

## Automatisation de la configuration d'openstack avec ansible

***Établissement d'accueil :***


***Encadrant académique :***

**Mr Riadh Ben Halima**

***Encadrant industriel :***

**Mr Taoufik Keskes**

# Remerciements

C'est avec un grand plaisir que je réserve cette page en signe de gratitude et de reconnaissance à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Tout d'abord, j'adresse mes remerciements à mon professeur Monsieur **Riadh Ben Halima** pour avoir été mon encadrant académique durant mon stage d'été.

Je souhaite également remercier chaleureusement Monsieur **Taoufik Keskes**, mon encadrant industriel, pour l'expérience enrichissante et pleine d'intérêt qu'il m'a fait vivre durant mon stage au sein de *GroupeRif*. Je lui suis reconnaissante pour son accueil, le temps qu'il m'a consacré et le partage quotidien de son expertise. Grâce à sa confiance, j'ai pu accomplir l'ensemble de mes missions. il a été d'un soutien précieux dans les moments les plus délicats.

Je remercie également l'équipe du *GroupeRif* qui m'a offert un environnement sain et motivant, propice à la réussite de ce stage.

Enfin, j'adresse mes remerciements à toutes les personnes qui m'ont conseillée et accompagnée lors de la rédaction de ce rapport de stage.

# Table des matières

# Liste des figures

# Acronymes

**IaaS :** Infrastructure as a Service

**PaaS :** Platform as a Service

**SaaS :** Software as a Service

**VM :** Machine virtuelle

**SSH :** Shell sécurisé

**API :** Interface de programmation d'application

**RGPD :** Règlement Général sur la Protection des Données

**NAT :** Network Address Translation (Traduction dadresse réseau)

**YAML :** Yet Another Markup Language (langage de sérialisation lisible)

**DevOps :** Développement et opérations

**IaC :** Infrastructure as Code (Infrastructure en tant que code)

**AWS :** Amazon Web Services (plateforme de services cloud dAmazon)

# Introduction générale

La gestion des infrastructures informatiques, la continuité des activités et la réduction des coûts constituent des priorités essentielles pour les entreprises modernes. Cependant, les solutions commerciales disponibles sur le marché sont souvent complexes à mettre en oeuvre et peu flexibles. Par ailleurs, l'évolution constante des technologies et l'arrivée de nouveaux équipements rendent le déploiement des infrastructures informatiques de plus en plus complexe, nécessitant des compétences techniques pointues et des efforts répétés pour garantir la stabilité, la sécurité et la performance.

C'est dans ce contexte que le Cloud Computing s'est imposé comme une solution innovante, permettant de répondre à ces défis tout en offrant une grande flexibilité, une scalabilité accrue et une réduction potentielle des coûts. En effet, le cloud repose sur la virtualisation des ressources informatiques matérielles et logicielles et permet une allocation dynamique des ressources selon les besoins. Grâce à cette approche, les entreprises peuvent disposer de services informatiques à la demande, sans avoir à investir massivement dans des infrastructures physiques.

Dans ce cadre, les infrastructures cloud prêtes à l'emploi , souvent fournies sous forme de services (IaaS, PaaS, SaaS), se sont multipliées. Elles permettent à de nombreuses organisations de déployer rapidement des environnements virtuels. Cependant, certaines entreprises, pour des raisons de sécurité, de conformité légale ou de contrôle total sur leurs données , préfèrent opter pour des clouds privés , qu'elles gèrent elles-mêmes.

C'est ici qu'intervient OpenStack , une plateforme open source largement adoptée pour la gestion d'infrastructures cloud privées. OpenStack permet de virtualiser et de gérer les ressources informatiques, en offrant une alternative flexible, modulaire et personnalisable aux solutions propriétaires. Toutefois, sa mise en oeuvre reste complexe, notamment en raison du nombre important de services interdépendants qu'il intègre, ainsi que des configurations spécifiques requises pour chaque déploiement.

Face à ces défis, l'automatisation des processus de déploiement et de configuration devient incontournable. C'est là que Ansible , outil d'automatisation puissant et simple d'usage entre en jeu. En utilisant des playbooks, Ansible permet de décrire de manière claire et reproductible les étapes nécessaires à la configuration d'un environnement OpenStack, assurant ainsi une mise en place rapide, cohérente et facilement maintenable.

Chapter 1

# Introduction

## 1.1 *Contexte*

La virtualisation a profondément transformé l'infrastructure informatique en permettant l'exécution simultanée de plusieurs systèmes sur une même machine physique. Cette avancée a ouvert la voie au cloud computing, qui propose un accès flexible et à la demande aux ressources informatiques. Dans ce cadre, OpenStack s'impose comme une solution open source majeure pour la mise en place de clouds privés. Cependant, la mise en oeuvre et la configuration d'OpenStack restent des tâches complexes, nécessitant une expertise technique solide. C'est pourquoi l'automatisation des tâches d'installation et de configuration devient une nécessité. L'outil Ansible , reconnu pour sa simplicité et sa fiabilité, s'impose comme une solution efficace pour répondre à ces enjeux.

## 1.2 *Problématiques*

Malgré les nombreux avantages offerts par OpenStack, sa mise en oeuvre manuelle reste un processus long, coûteux et sujet à des erreurs humaines . En effet, OpenStack est composé de plusieurs services interdépendants (Keystone, Glance, Nova, Neutron, Cinder, etc.) nécessitant des configurations précises et souvent répétitives.

Dans un environnement dynamique où les infrastructures évoluent fréquemment, cette approche manuelle n'est plus adaptée . Elle manque de reproductibilité , de standardisation , et ne permet pas une réponse rapide aux besoins opérationnels. De plus, elle rend la maintenance complexe et augmente le risque d'incohérences entre les environnements de développement, de test et de production.

## 1.3 *Objectifs*

### 1.3.1 Réduction des efforts de déploiement

L'automatisation permet de réduire considérablement le temps et les efforts nécessaires à la mise en place d'une infrastructure OpenStack. Elle élimine les tâches répétitives et manuelles, sources potentielles d'erreurs, et garantit une exécution plus fluide et cohérente des étapes de déploiement. Grâce à des outils comme Ansible, le processus devient plus rapide, plus fiable et plus accessible même aux équipes moins expérimentées.

### 1.3.2 Reproductibilité et standardisation des environnements

Ansible permet d'assurer une configuration homogène sur l'ensemble des environnements grâce à l'exécution des mêmes *playbooks*. Cela garantit que chaque déploiement respecte les mêmes standards, réduisant les divergences et les erreurs de configuration. La standardisation facilite aussi la maintenance et le passage d'un environnement à un autre (test, préproduction, production).

### 1.3.3 Maintenance simplifiée et évolutivité

Les *playbooks* Ansible sont modifiables et réutilisables, facilitant l'évolution de l'infrastructure au fil des besoins. Cela permet une mise à jour rapide et une *scalabilité* optimale.

### 1.3.4 Objectifs spécifiques du projet

- Déployer une architecture OpenStack complète via Ansible.
- Automatiser la configuration des services principaux (Keystone, Nova, Neutron, etc.).
- Garantir la sécurité, la stabilité et la reproductibilité du déploiement.
- Documenter les *playbooks* et les bonnes pratiques pour faciliter la maintenance.

## 1.4 *Organisation du rapport*

Ce rapport est organisé autour des principales sections suivantes :

- **Introduction** : Elle présente le contexte du stage, la problématique abordée, les motivations du projet ainsi que la méthodologie adoptée.
- **Présentation de l'entreprise** : Elle fournit un aperçu de la structure d'accueil du stage, ses domaines d'activité et son intérêt pour la solution mise en place.
- **Technologies utilisées** : Cette partie présente les fondements techniques sur lesquels repose le projet, notamment la plateforme OpenStack ainsi que l'outil d'automatisation Ansible.
- **État de l'art** : Cette section analyse les différentes solutions existantes pour le déploiement d'OpenStack, en comparant des approches avant de motiver le choix retenu.
- **Mise en oeuvre du projet** : Cette partie détaille les étapes techniques suivies pour l'automatisation du déploiement, les outils utilisés, la configuration des environnements, ainsi que les résultats obtenus.
- **Conclusion générale** : Elle propose un bilan du stage, une réflexion critique sur les réalisations, ainsi que des perspectives d'amélioration.

## 1.5 *Conclusion*

Cette introduction a permis de situer le contexte technologique actuel, d'identifier les défis liés à la configuration manuelle d'OpenStack, et de justifier l'utilisation d'Ansible pour automatiser ces tâches.

Chapter 2

# Présentation de l'entreprise

GroupeRif est une entreprise de services numériques innovante, engagée dans le développement de solutions digitales sur mesure pour répondre aux besoins croissants des entreprises dans leur transformation numérique. Elle se distingue par une approche centrée sur la qualité, l'innovation et l'accompagnement personnalisé de ses clients, quels que soient leur taille ou leur secteur d'activité.

Fondée avec la volonté de rapprocher les entreprises des technologies les plus avancées, GroupeRif intervient dans plusieurs domaines stratégiques tels que le développement web et mobile, le cloud souverain, les systèmes ERP, les solutions d'intelligence artificielle et la gestion de projets informatiques. Grâce à cette expertise diversifiée, elle propose un large éventail de services allant de la création de sites web modernes et d'applications mobiles performantes à la mise en place de plateformes cloud sécurisées et d'outils de gestion internes sur mesure.

L'une des particularités de GroupeRif réside dans son engagement à fournir des solutions évolutives et intelligentes, capables de s'adapter aux besoins présents et futurs des clients. L'entreprise veille à intégrer dans ses projets les dernières avancées technologiques, tout en maintenant un haut niveau d'exigence en termes de performance, de sécurité et d'expérience utilisateur.

GroupeRif accorde une grande importance à une culture d'entreprise ouverte et collaborative, où chacun est encouragé à s'exprimer, à partager ses idées et à contribuer au succès

de l'équipe. L'environnement de travail y est à la fois stimulant, humain et formateur, favorisant l'apprentissage continu, le partage de compétences et le développement personnel.

Durant mon stage, j'ai pu constater cette dynamique d'innovation et d'entraide au sein des équipes. J'ai eu l'opportunité de collaborer avec des professionnels passionnés, dans un cadre propice à l'initiative et à la montée en compétences. Ce cadre de travail m'a permis de comprendre concrètement les méthodes de gestion de projets numériques et les attentes liées au métier, tout en évoluant dans un environnement à la fois bienveillant et axé sur les résultats.

En somme, GroupeRif ne se contente pas de développer des solutions technologiques: elle s'engage pleinement dans le succès de ses clients et dans l'évolution de ses collaborateurs. Grâce à cette double ambition, elle se positionne aujourd'hui comme un acteur clé de la transformation digitale et un partenaire de confiance pour bâtir l'avenir numérique.


**Figure 1.** *Logo de GroupeRif [1]*

Chapter 3

# Technologies utilisées

## 3.1 *Aperçu d'OpenStack et de l'infrastructure cloud*

### 3.1.1 Le Cloud Computing : concepts clés

Le Cloud Computing désigne un modèle de fourniture à la demande de ressources informatiques (serveurs, stockage, bases de données, réseaux, logiciels, etc.), accessibles à distance via Internet. Ce paradigme permet une gestion plus souple, évolutive et économique des infrastructures IT, en réduisant les coûts liés à l'achat et à la maintenance du matériel.

Il existe trois principaux modèles de services dans le cloud :

- Infrastructure as a Service (IaaS) : Ce modèle fournit une infrastructure informatique virtualisée, telle que des machines virtuelles, du stockage et des réseaux. L'utilisateur gère les systèmes d'exploitation et les applications installées, tandis que le fournisseur gère le matériel sous-jacent.
- Platform as a Service (PaaS) : Fournit un environnement de développement et de déploiement complet, sans que l'utilisateur ait à gérer l'infrastructure sous-jacente. Cela inclut les serveurs, le stockage, les systèmes d'exploitation, et même certains outils de développement.

- Software as a Service (SaaS) : Met à disposition des applications accessibles via un navigateur web. L'utilisateur consomme directement le service sans avoir à gérer l'installation, la maintenance ou l'infrastructure.

### 3.1.2 OpenStack comme solution IaaS

Parmi les différentes solutions IaaS disponibles, OpenStack se distingue comme une plateforme open source robuste et flexible, conçue pour construire et gérer des infrastructures cloud, notamment privées et hybrides. Son adoption est particulièrement pertinente dans les contextes où un haut niveau de contrôle, de personnalisation et de sécurité est requis.

OpenStack est ainsi bien adapté pour :

- Les organisations souhaitant garder un contrôle total sur leur infrastructure, notamment lorsqu'il s'agit de données sensibles (santé, finance, défense) ;
- Les environnements soumis à des réglementations strictes (ex. : RGPD) ;
- Les cas d'usage nécessitant une forte personnalisation des composants et des configurations ;
- Les projets où la sécurité et la conformité sont des priorités.

### 3.1.3 Architecture générale d'OpenStack

L'architecture d'OpenStack repose sur une organisation modulaire où chaque service est responsable d'un aspect spécifique de l'infrastructure cloud. Ces services communiquent entre eux via des APIs et sont répartis sur plusieurs noeuds physiques ou virtuels.

Un déploiement de base d'OpenStack se compose généralement des noeuds suivants :

- **Noeud contrôleur (Controller Node)** : centralise les services de gestion et de contrôle. Il héberge notamment Keystone, Glance, Horizon, le scheduler de Nova, le serveur de message (RabbitMQ), la base de données (ex. : MariaDB), et parfois Neutron server.

- **Noeud de calcul (Compute Node)** : exécute les machines virtuelles. Il héberge Nova Compute et les agents Neutron nécessaires pour la connectivité réseau.
- **Noeud de stockage (Storage Node)** : utilisé pour le stockage persistant. Il peut héberger les services Cinder (stockage en blocs) et/ou Swift (stockage en objets).

### 3.1.4 Les principaux services OpenStack

OpenStack est composé d'un ensemble de services modulaires et interconnectés, chacun étant responsable d'un aspect spécifique de l'infrastructure cloud. Voici les principaux services :

- **Keystone** : Service d'authentification et d'autorisation. Il délivre des jetons (tokens) permettant aux utilisateurs ou systèmes d'accéder aux autres services OpenStack via les API.
- **Nova** : Service de calcul (*Compute*). Il gère le cycle de vie des machines virtuelles (lancement, arrêt, redémarrage, suppression) et alloue les ressources de calcul. Il repose sur plusieurs composants :
  - **Nova API** : reçoit les requêtes d'exécution ;
  - **Nova Scheduler** : détermine le noeud d'exécution optimal ;
  - **Nova Compute** : assure l'exécution effective de la machine virtuelle.
- **Glance** : Service de gestion des images disques utilisées pour le démarrage des machines virtuelles.
- **Neutron** : Service de gestion réseau. Il assure la connectivité des machines virtuelles entre elles ainsi qu'avec l'extérieur.
- **Cinder** : Gère le stockage en blocs. Il fournit des disques durs virtuels qui se comportent comme des disques physiques dans un environnement virtualisé.

- **Swift** : Gère le stockage en objets. Il permet de stocker des fichiers (images, vidéos, sauvegardes, etc.) sous forme d'objets accessibles via une URL ou une API.
- **Horizon** : Tableau de bord web facilitant l'administration et la visualisation des services OpenStack.
- **Heat** : Service d'orchestration. Il permet d'automatiser la création d'infrastructures cloud complètes via un fichier de description. Au lieu de créer une machine virtuelle, un réseau, un volume ou une règle de sécurité manuellement, ces éléments peuvent être définis dans un fichier que Heat interprète pour effectuer le déploiement automatiquement.

Chaque service peut exposer plusieurs types de points d'accès (*endpoints*) :

- **Public** : Accessible depuis l'extérieur, utilisé par les clients ou utilisateurs.
- **Internal** : Réservé aux communications internes entre services OpenStack.
- **Admin** : Utilisé pour les opérations d'administration.

Les services OpenStack communiquent entre eux via des appels API et des files de messages (comme RabbitMQ). Par exemple :

- Nova utilise Keystone pour s'authentifier.
- Nova utilise Glance pour récupérer les images.
- Nova utilise Neutron pour configurer le réseau des instances.
- Cinder est utilisé par Nova pour attacher des volumes persistants.

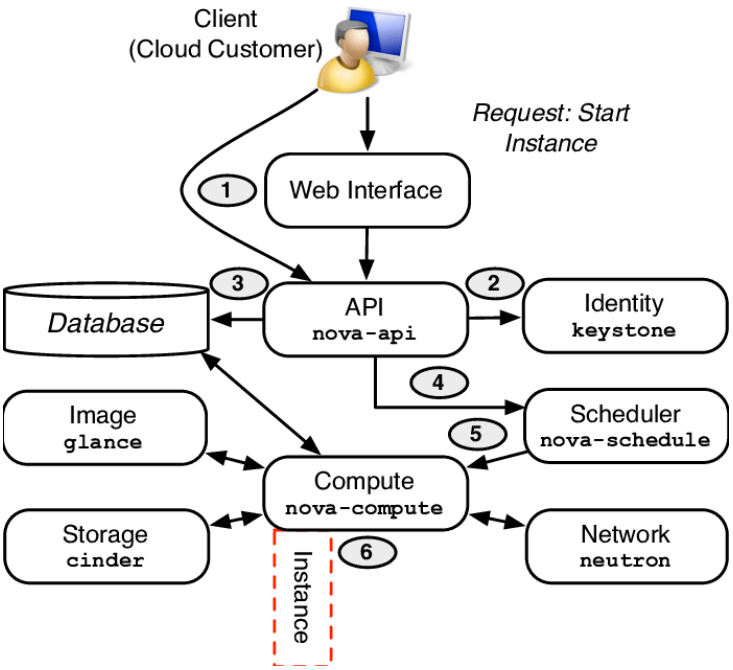


**Figure 2.** *Flux de création d'une instance dans OpenStack [2]*

Cette figure illustre les composants clés d'OpenStack et les interactions entre ses différents services.

### 3.1.5 Réseaux dans OpenStack : Management et Provider Network

#### 3.1.5.1 Réseau de gestion (Management Network) :

Le réseau de gestion est utilisé pour permettre la communication entre les différents services internes d'OpenStack (comme Nova, Neutron, Cinder, etc.). Ce réseau n'est pas exposé aux utilisateurs finaux. Il est principalement utilisé pour :

- Le contrôle et la coordination entre les composants internes (ex. : entre le contrôleur et les noeuds de calcul).
- La gestion des hôtes via SSH ou d'autres outils d'administration.
- Le transfert des messages via le bus de communication (RabbitMQ, etc.).

Exemple d'usage : lorsqu'un utilisateur demande la création d'une VM, c'est par le réseau de gestion que les instructions circulent entre les composants pour planifier et lancer cette VM.

#### 3.1.5.2 Réseau fournisseur (Provider Network) :

Le réseau fournisseur permet aux machines virtuelles (VMs) créées dans OpenStack de communiquer avec l'extérieur, c'est-à-dire avec le réseau physique réel (par exemple Internet ou un réseau d'entreprise). Il est directement connecté à une infrastructure réseau existante, sans NAT ni routage OpenStack. Caractéristiques :

- le réseau fournisseur (Provider Network) est directement associé (ou lié) à une carte réseau physique de la machine hôte.
- Les VMs obtiennent des adresses IP du réseau physique réel.

Exemple d'usage : déployer une VM qui doit être accessible par l'utilisateur depuis Internet.

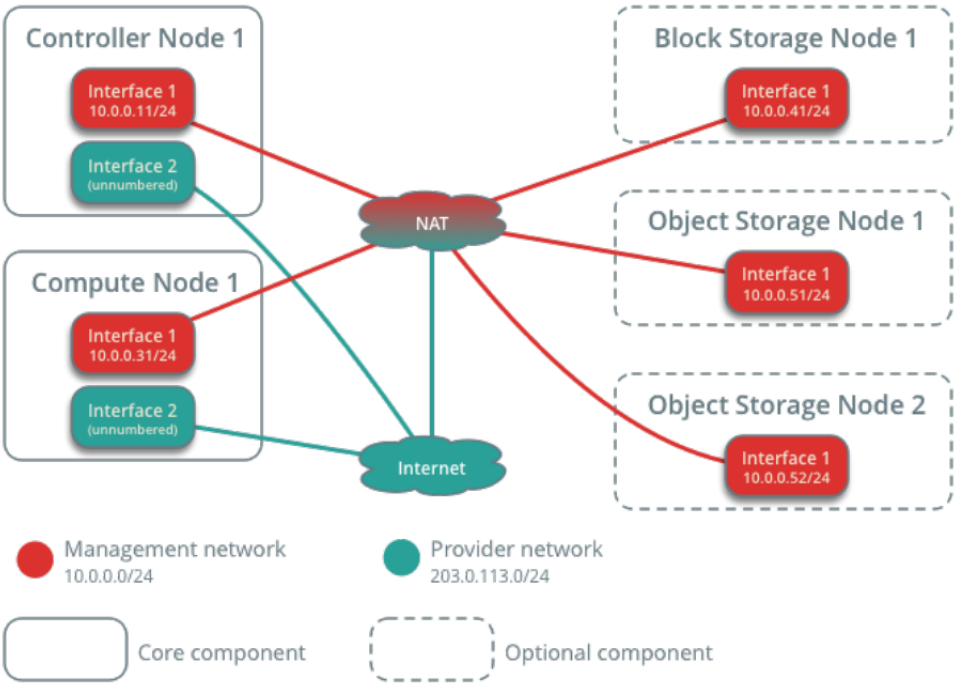


**Figure 3.** *Architecture réseau d'une infrastructure OpenStack [3]*

Dans le schéma ci-dessus, Chaque noeud est connecté au réseau de gestion via une interface spécifique et un mécanisme de NAT. Ils sont également connectés au réseau fournisseur pour permettre l'accès à Internet ou à des clients externes.

## 3.2 *Automatisation avec Ansible*

### 3.2.1 Qu'est-ce qu'Ansible et pourquoi l'utiliser ?

Ansible est un outil open source d'automatisation qui permet de configurer des serveurs, déployer des applications et orchestrer des tâches IT de manière simple, rapide et sans agent. Il utilise des scripts simples et lisibles par l'humain, appelés playbooks, pour automatiser les tâches. On déclare l'état souhaité d'un système local ou distant dans le playbook. Ansible veille à ce que le système reste dans cet état. Son utilisation est préférable à l'installation et à la configuration manuelles d'OpenStack, car déployer OpenStack manuellement est long, complexe et sujet aux erreurs. Ansible automatise tout le processus:

- Il élimine la répétition et simplifie les flux de travail
- Il permet de gérer et maintenir la configuration des systèmes
- Il facilite le déploiement continu de logiciels complexes
- Il rend possibles des mises à jour progressives sans interruption

### 3.2.2 Architecture et fonctionnement d'Ansible

Ansible s'intègre naturellement dans une démarche DevOps et de Infrastructure as Code. Il permet de versionner, tester et automatiser la configuration d'infrastructures comme on gère du code logiciel, assurant la traçabilité, reproductibilité et automatisation continue. Les playbooks sont des fichiers écrits en YAML où nous définissons les tâches que nous souhaitons exécuter sur nos noeuds gérés. Un playbook est composé de "plays", et chaque

"play" contient plusieurs tâches. Les tâches utilisent des modules Ansible pour effectuer diverses actions comme l'installation de packages, la copie de fichiers, ou la gestion de services. Les playbooks permettent de définir des processus automatisés de manière claire et répétable.

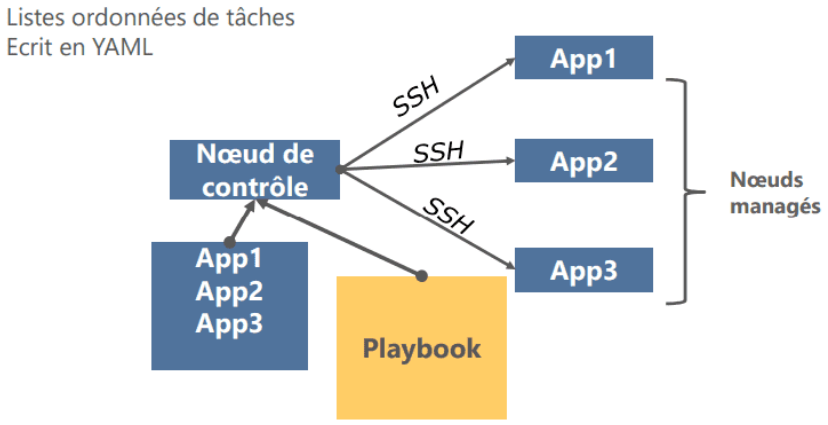


**Figure 4.** *Architecture d'Ansible [4]*

Comme illustré dans la figure précédente, la plupart des environnements Ansible comportent trois composants principaux :

- Control node: Un système sur lequel Ansible est installé. On exécute les commandes Ansible telles que ansible ou ansible-inventory sur ce noeud de contrôle en utilisant SSH pour se connecter aux managed nodes.
- Inventory: Une liste de managed nodes qui sont organisés de manière logique. On crée un inventory sur le control node pour décrire les déploiements des hôtes à Ansible.
- Managed node: Un système distant, ou hôte, qu'Ansible contrôle.

### 3.2.3 Concepts fondamentaux

Ansible repose sur plusieurs concepts clés qui structurent la manière dont les tâches d'automatisation sont définies et exécutées. Voici un aperçu des éléments fondamentaux:

**a) Playbooks :**

Ils contiennent des Plays (qui sont l'unité de base d'exécution dans Ansible). C'est à la fois un concept d'exécution et une façon de désigner les fichiers sur lesquels la commande ansible-playbook agit.

**b) Plays :**

C'est le contexte principal d'exécution dans Ansible. Un Play associe les noeuds gérés (hôtes) à des tâches. Il contient des variables, des rôles et une liste ordonnée de tâches, et peut être exécuté plusieurs fois. Il consiste essentiellement en une boucle implicite sur les hôtes et les tâches, et définit comment les parcourir.

**c) Tâches (Tasks) :**

Définissent une action à appliquer à l'hôte géré.

**d) Handlers :**

Ce sont des tâches spéciales que tu écris dans ton playbook, qui ne s'exécutent que si une autre tâche précédente a fait un changement sur la machine cible. On l'utilise pour éviter de répéter des actions inutiles. Par exemple, si plusieurs tâches modifient une configuration, tu ne veux pas redémarrer le service plusieurs fois dans le même playbook, mais seulement une fois à la fin.

**e) Modules :**

C'est le code qu'Ansible envoie et exécute sur la machine distante (appelée managed node) pour faire une tâche précise. Dans un playbook, chaque tâche utilise un module pour dire ce qu'elle veut faire. Par exemple, une tâche peut utiliser le module copy pour copier un fichier, une autre le module user pour créer un utilisateur. Les modules sont

regroupés dans des collections, un peu comme des "paquets" qui contiennent plusieurs modules liés à un domaine (exemple : modules pour Linux, pour Cisco, pour AWS, etc.).

**f) Plugins :**

Les plugins ajoutent des fonctionnalités supplémentaires à Ansible, au-delà des modules classiques. Ils interviennent notamment dans :

- la connexion aux hôtes (plugins de connexion : SSH, WinRM, etc.),
- la transformation ou le filtrage des données (plugins de filtre),
- l'affichage des résultats en console (plugins de rappel).

**g) Collections :**

Une collection est un paquet tout-en-un qui regroupe plusieurs types de contenus Ansible, par exemple : des playbooks (scénarios d'automatisation), des rôles (groupes de tâches organisées), des modules (actions spécifiques à exécuter), des plugins (extensions pour Ansible). Au lieu d'installer ou gérer ces éléments séparément, tu peux télécharger ou partager une collection complète qui contient tout ce dont tu as besoin pour un usage spécifique (par exemple, une collection pour gérer des équipements Cisco, ou pour déployer sur AWS).

### 3.2.4 Méthodologie DevOps et Infrastructure as Code (IaC)

Dans le cadre de ce projet, la méthodologie adoptée repose sur les principes du DevOps et de l'Infrastructure as Code (IaC), avec l'utilisation d'Ansible pour l'automatisation de la configuration et du déploiement.

#### 3.2.4.1 Méthodologie DevOps

La démarche DevOps vise à rapprocher les équipes de développement et d'exploitation, afin de favoriser une collaboration efficace tout au long du cycle de vie logiciel. Elle repose sur plusieurs piliers fondamentaux :

- **Automatisation** des processus de build, test, intégration et déploiement.
- **Livraison continue** pour déployer rapidement et fréquemment les nouvelles fonctionnalités.
- **Collaboration renforcée** entre les développeurs, les administrateurs systèmes et les utilisateurs finaux.

### 3.2.5 Infrastructure as Code (IaC)

L'Infrastructure as Code est une approche qui consiste à définir et gérer l'infrastructure à l'aide de fichiers de configuration, versionnés comme du code source.

Cette pratique permet de :

- Garantir la **reproductibilité** des environnements.
- **Faciliter la collaboration** au sein des équipes grâce au versioning.
- **Automatiser les tests et les déploiements**, réduisant ainsi les erreurs humaines.

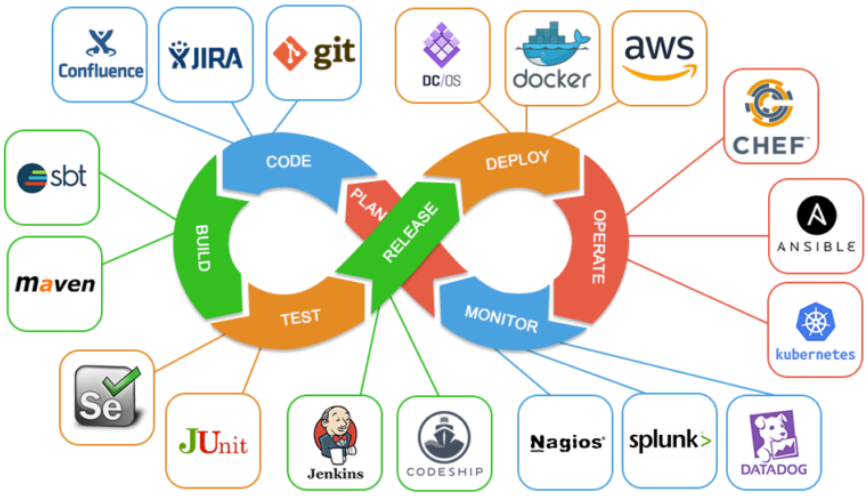


**Figure 5.** *Cycle DevOps intégrant l'Infrastructure as Code avec Ansible [5]*

Cette figure illustre les différentes étapes du cycle DevOps et montre comment Ansible, en tant qu'outil d'IaC, s'intègre dans cette chaîne pour automatiser l'ensemble du processus de déploiement.

Chapter 4

# État de l'art

## 4.1 *Introduction*

Avant de se lancer dans la conception et l'implémentation technique, il est essentiel de réaliser un état de l'art  c'est-à-dire une revue des technologies, outils et méthodes existants susceptibles de répondre aux besoins du projet. Cette phase permet de :

- Identifier les solutions disponibles sur le marché.
- Comprendre leurs forces et faiblesses.
- Justifier les choix techniques retenus.
- Positionner le projet dans un contexte technologique et organisationnel.

## 4.2 *Solutions existantes pour le déploiement d'OpenStack*

Plusieurs outils permettent de déployer et configurer une infrastructure OpenStack. Ils se distinguent par leur complexité, flexibilité et mode d'installation :

### 4.2.1 DevStack

Outil principalement destiné aux environnements de développement et de test. Il permet d'installer rapidement une instance d'OpenStack sur une seule machine. Toutefois, il n'est pas recommandé pour une utilisation en production en raison de son manque de robustesse, de sécurité et de persistance.

### 4.2.2 Kolla-Ansible

Utilise Ansible en combinaison avec Docker pour déployer les services OpenStack sous forme de conteneurs. Kolla-Ansible est orienté vers les environnements où la conteneurisation est privilégiée. Il offre des avantages en termes d'isolation, de mise à l'échelle et de facilité de mise à jour.

### 4.2.3 Packstack

Outil simple d'installation basé sur Puppet, utilisé pour des déploiements rapides à des fins d'apprentissage ou de test. Il est moins maintenu que les autres solutions et ne convient pas aux besoins en production.

### 4.2.4 Puppet

Puppet est un langage déclaratif utilisé pour la configuration d'OpenStack et la gestion du cycle de vie des noeuds. Il fonctionne en mode client/serveur ou sans serveur, selon une logique "écrire une fois, déployer partout". Basé sur Ruby, Puppet nécessite une certaine expertise en programmation. Il propose une interface Web, des outils de reporting et permet l'automatisation à grande échelle via des modules prédéfinis. Puppet Enterprise offre une gestion en temps réel, des rapports détaillés, une conformité renforcée et une application centralisée des correctifs de sécurité.

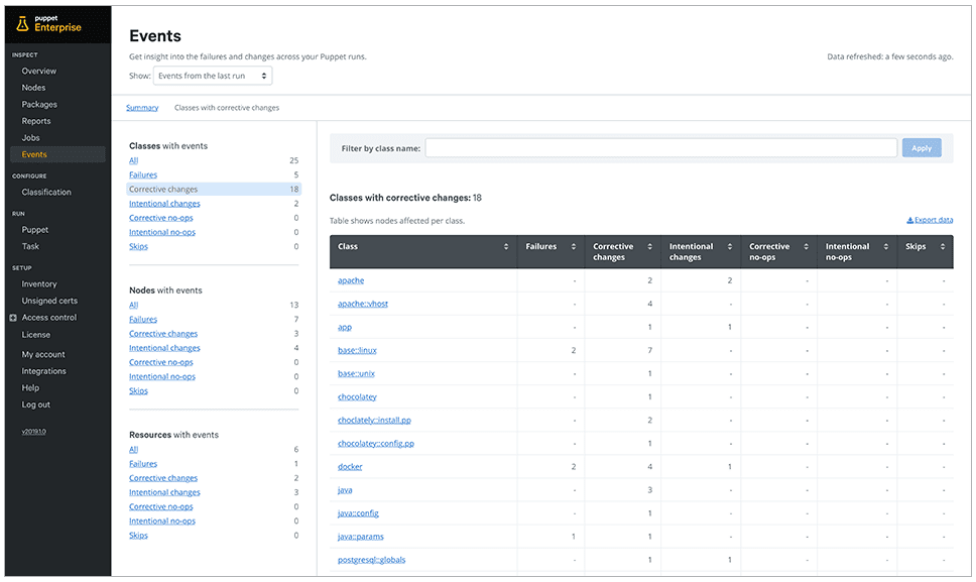


**Figure 6.** *Interface de monitoring des événements Puppet Enterprise montrant le suivi des changements et des échecs de configuration [6]*

### 4.2.5 Compass

Compass est un outil d'automatisation du déploiement et de la gestion d'OpenStack. Il simplifie la gestion des serveurs dans les centres de données en réduisant les erreurs et le temps nécessaire aux configurations. Compass permet :

- Le déploiement d'OpenStack depuis des serveurs physiques (bare metal) vers n'importe quelle plateforme cloud.
- L'identification du matériel, l'installation du système d'exploitation, de l'hyperviseur et la gestion de la configuration.
- L'implémentation de différentes configurations via des métadonnées.
- La programmabilité de l'infrastructure pour les opérateurs.
- L'extensibilité grâce à l'intégration d'outils tiers pour la découverte, la planification et le déploiement.

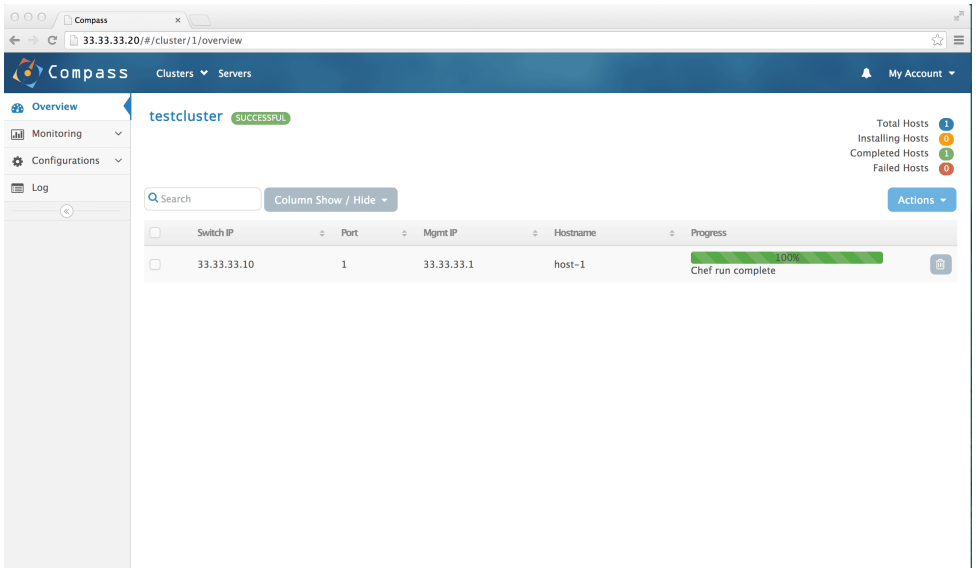


**Figure 7.** *Interface web Compass montrant le statut du cluster OpenStack après déploiement automatisé réussi. [6]*

### 4.2.6 OpenStack-Ansible

Solution d'automatisation basée sur Ansible qui permet le déploiement d'OpenStack en production sur plusieurs noeuds. Elle offre une grande flexibilité, une configuration modulaire, ainsi qu'un bon niveau de sécurité. Elle est adaptée aux environnements complexes et reproductibles.

## 4.3 *Comparaison et justification du choix technologique*

Après avoir étudié les différentes solutions de déploiement d'OpenStack, nous avons retenu **OpenStack-Ansible** pour les raisons suivantes :

- **Approche production-ready** : Contrairement à DevStack ou Packstack, OpenStack-Ansible est conçu pour des environnements de production, offrant une meilleure stabilité et sécurité.
- **Automatisation complète et flexibilité** : Grâce à l'utilisation d'Ansible, il est possible d'automatiser tout le cycle de déploiement avec des configurations personnalisables et réutilisables.

- **Gestion multi-noeuds** : OpenStack-Ansible permet de déployer des environnements distribués, ce qui correspond à notre besoin de simuler un cluster composé de plusieurs noeuds.
- **Communauté active et documentation** : La solution est bien documentée et dispose d'une large communauté, facilitant la résolution de problèmes et l'apprentissage.
- **Alignement avec les principes DevOps et IaC** : Le choix d'Ansible s'inscrit dans une logique d'Infrastructure as Code (IaC), favorisant la reproductibilité, la versioning des configurations, et l'intégration continue.

En conclusion, OpenStack-Ansible représente un compromis idéal entre complexité de mise en oeuvre et robustesse, tout en étant compatible avec les bonnes pratiques modernes de gestion d'infrastructure.

## 4.4 *Conclusion*

Cette section a permis de connaitre des outils et technologies disponibles pour automatiser le déploiement d'OpenStack . Elle a montré que plusieurs solutions existent, mais qu'Ansible se démarque par sa simplicité, sa puissance et son intégration naturelle avec OpenStack .

Chapter 5

# Mise en oeuvre du projet

## 5.1 *Introduction*

Ce chapitre est consacré à la mise en oeuvre pratique du projet, constituant la phase centrale où les choix méthodologiques et théoriques présentés précédemment trouvent leur application concrète. Il expose de manière détaillée le processus de déploiement de la plateforme, en mettant l'accent sur la logique d'exécution adoptée ainsi que sur l'organisation des différents playbooks Ansible utilisés.

L'objectif de cette partie est non seulement de décrire les étapes techniques ayant permis la mise en place de l'infrastructure, mais également de souligner l'importance de l'ordre d'exécution, des dépendances entre composants et des outils mobilisés pour assurer la fiabilité et la cohérence du déploiement.

## 5.2 *Étapes de l'implémentation*

### 5.2.1 Préparation de l'hôte de déploiement

L'hôte de déploiement, agissant comme machine de contrôle, a été soigneusement configuré pour assurer une gestion centralisée de l'ensemble de l'infrastructure. Les dépendances logicielles suivantes ont été installées :

- Python 3 et `pip` pour l'exécution des scripts et la gestion des paquets,
- Ansible, outil d'automatisation central du projet,
- Le code source d'`OpenStack-Ansible`, récupéré depuis le dépôt Git officiel.

Une configuration réseau appropriée a été mise en place afin d'assurer la connectivité avec tous les noeuds cibles. En outre, des clés SSH ont été générées sur l'hôte de contrôle et distribuées sur les noeuds distants, permettant un accès sécurisé et sans saisie de mot de passe via SSH.

### 5.2.2 Préparation des noeuds cibles

Les noeuds destinés à héberger les services OpenStack ont été préparés selon une configuration standardisée. Chaque machine a fait l'objet :

- d'un partitionnement disque adapté aux besoins des services,
- d'une attribution d'adresses IP statiques (sur les réseaux de gestion et de conteneur),
- d'une configuration DNS correcte pour assurer la résolution des noms d'hôtes.

Un test de connectivité SSH a été effectué depuis l'hôte de déploiement vers chaque noeud cible, afin de s'assurer de leur accessibilité et de la bonne configuration du réseau.

### 5.2.3 Configuration du déploiement

La personnalisation de l'infrastructure a été réalisée via deux fichiers de configuration principaux :

- `/etc/openstack_deploy/openstack_user_config.yml`
- `/etc/openstack_deploy/user_variables.yml`

Ces fichiers ont été soigneusement configurés afin de refléter la topologie physique et logique de l'infrastructure déployée. Ils définissent de manière déclarative tous les aspects essentiels du système, notamment :

- **Les adresses IP de gestion et celles dédiées aux conteneurs** : Chaque noeud se voit attribuer des adresses IP sur différents réseaux (gestion pour l'administration, conteneur pour la communication interne entre services). Cette séparation permet d'isoler les flux critiques et d'améliorer la sécurité et la stabilité du système.
- **Les noms d'hôtes et la répartition des rôles** : La fonction de chaque machine (noeud contrôleur, noeud compute, noeud de stockage, etc.) est explicitement définie. Cette approche modulaire permet une architecture claire, évolutive et facile à maintenir.
- **La configuration réseau (interfaces, VLANs, passerelles)** : Une segmentation réseau rigoureuse est indispensable dans OpenStack pour isoler les différents types de trafic (gestion, données, stockage, tunneling, etc.) et garantir la performance, sécurité et scalabilité.
- **Les paramètres spécifiques à chaque service OpenStack** : Des options fines sont définies pour chaque composant (Keystone, Glance, Nova, Cinder, etc.), telles que les tailles d'instances par défaut, les backends de stockage, les politiques de sécurité, ou encore les mécanismes de haute disponibilité. Ces paramètres permettent d'ajuster le comportement de la plateforme aux besoins fonctionnels et opérationnels du déploiement.

Cette étape de configuration est cruciale : elle rend le déploiement entièrement déclaratif, automatisé, reproductible et traçable. Elle constitue la base d'un processus fiable, facile à auditer, et adaptable à différentes topologies d'infrastructure.

### 5.2.4 Exécution des playbooks Ansible

Le déploiement a été exécuté selon l'ordre recommandé par la documentation OpenStack-Ansible, en respectant les dépendances entre les composants. Les playbooks principaux ont été lancés successivement :

1. `ansible-playbook setup-hosts.yml`

   Ce playbook initialise les noeuds cibles : installation des dépendances système, configuration des conteneurs LXC, et préparation de l'environnement d'exécution.
2. `ansible-playbook setup-infrastructure.yml`

   Il déploie les services d'infrastructure critiques, tels que les bases de données (MariaDB), le système de messagerie (RabbitMQ), et les services réseau internes (HAProxy, Keepalived, etc.).
3. `ansible-playbook setup-openstack.yml`

   Cette étape installe et configure les services OpenStack proprement dits : Keystone (identité), Glance (images), Nova (calcul), Neutron (réseau), Cinder (stockage bloc), ainsi que Horizon (interface web).

Chaque phase du déploiement a fait l'objet d'un suivi attentif, accompagné d'une analyse des journaux d'exécution afin de détecter d'éventuelles anomalies et de s'assurer de la conformité du processus.

### 5.2.5 Vérification du bon fonctionnement d'OpenStack

Après le déploiement, il est nécessaire de procéder à une série de vérifications afin de s'assurer du bon fonctionnement de la plateforme. Ces vérifications portent à la fois sur l'API d'OpenStack et sur l'interface graphique Horizon.

#### Vérification de l'API

La vérification de l'API s'effectue depuis le conteneur *utility*, qui fournit un environnement en ligne de commande adapté à la configuration et aux tests. La première étape consiste

à identifier le nom du conteneur en utilisant la commande `lxc-ls` avec un filtrage par `grep utility`.

Une fois le conteneur identifié, on procède à l'attachement au conteneur utility pour accéder à son environnement. À l'intérieur du conteneur, il convient de charger les variables d'environnement de l'utilisateur administrateur, définies dans le fichier `openrc`.

Pour vérifier le bon fonctionnement de l'API, on peut créer un utilisateur de test, comme illustré dans la figure 8, où l'utilisateur "hiba" est créé avec succès dans le domaine par défaut.

```
root@controller-utility-container-462dce16:/# openstack user create hiba --password 'motdepasse123' --email hiba@cloud.l
ocal --enable
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| default_project_id  | None                             |
| domain_id           | default                          |
| email               | hiba@cloud.local                 |
| enabled             | True                             |
| id                  | d592d5de083e4f359faa8dfcaf00cbf4 |
| name                | hiba                             |
| description         | None                             |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

**Figure 8.** *Création d'un utilisateur OpenStack via l'API confirmant son accessibilité.*

Ensuite, on peut exécuter la commande `openstack user list` pour afficher la liste des utilisateurs OpenStack. Le retour de cette commande doit présenter l'ensemble des utilisateurs, incluant ceux créés par défaut dans l'environnement ainsi que les nouveaux utilisateurs créés, comme le montre la figure 9.

```
root@controller-utility-container-462dce16:/# exit
exit
root@controller-virtual-machine:/home/controller# lxc-ls | grep utility
controller-repo-container-a106a4e3          controller-utility-container-462dce16
root@controller-virtual-machine:/home/controller# lxc-attach -n controller-utility-container-462dce16
root@controller-utility-container-462dce16:/# . ~/openrc
root@controller-utility-container-462dce16:/# openstack user list --os-cloud=default
+----------------------------------+-------+
| ID                               | Name  |
+----------------------------------+-------+
| 82ef36874e49422b8dff23e575b0f67f | admin |
| d592d5de083e4f359faa8dfcaf00cbf4 | hiba  |
+----------------------------------+-------+
root@controller-utility-container-462dce16:/#
```

**Figure 9.** *Affichage de la liste des utilisateurs OpenStack confirmant le bon fonctionnement de l'API.*

Cette vérification confirme l'accessibilité et le bon fonctionnement de l'API OpenStack, démontrant que les services principaux sont opérationnels et que les opérations de gestion des utilisateurs s'exécutent correctement.

### Vérification du tableau de bord Horizon

La seconde étape consiste à vérifier l'accès au tableau de bord web Horizon. Celui-ci est disponible via l'adresse IP ou le nom de domaine du répartiteur de charge externe (`external_lb_vip_address`), telle que définie dans le fichier `/etc/openstack_deploy /openstack_user_config.yml`. Le tableau de bord est accessible en HTTPS sur le port 443.

Pour s'authentifier, on utilise le compte administrateur (`admin`) ainsi que le mot de passe spécifié dans le fichier `/etc/openstack_deploy/user_secrets.yml`. L'accès réussi à l'interface Horizon, ainsi que la possibilité de naviguer entre les différents services proposés, constituent une validation supplémentaire du bon fonctionnement de la plateforme.

Afin de valider le bon fonctionnement du service de calcul Nova, deux instances de test ont été crées via l'interface Horizon : `test-vm-1` et `test-vm-2`.

Comme illustré dans la figure ci-dessous, le tableau de bord affiche ces instances dans la section "Instances" du service Nova. On peut y observer leurs caractéristiques : adresses IP internes et externes, leur état de fonctionnement, ainsi que leur âge de création.

La création réussie de ces instances de test, leur affichage correct dans l'interface Horizon, ainsi que la possibilité de naviguer entre les différents services constituent une validation complète du bon fonctionnement de la plateforme OpenStack.

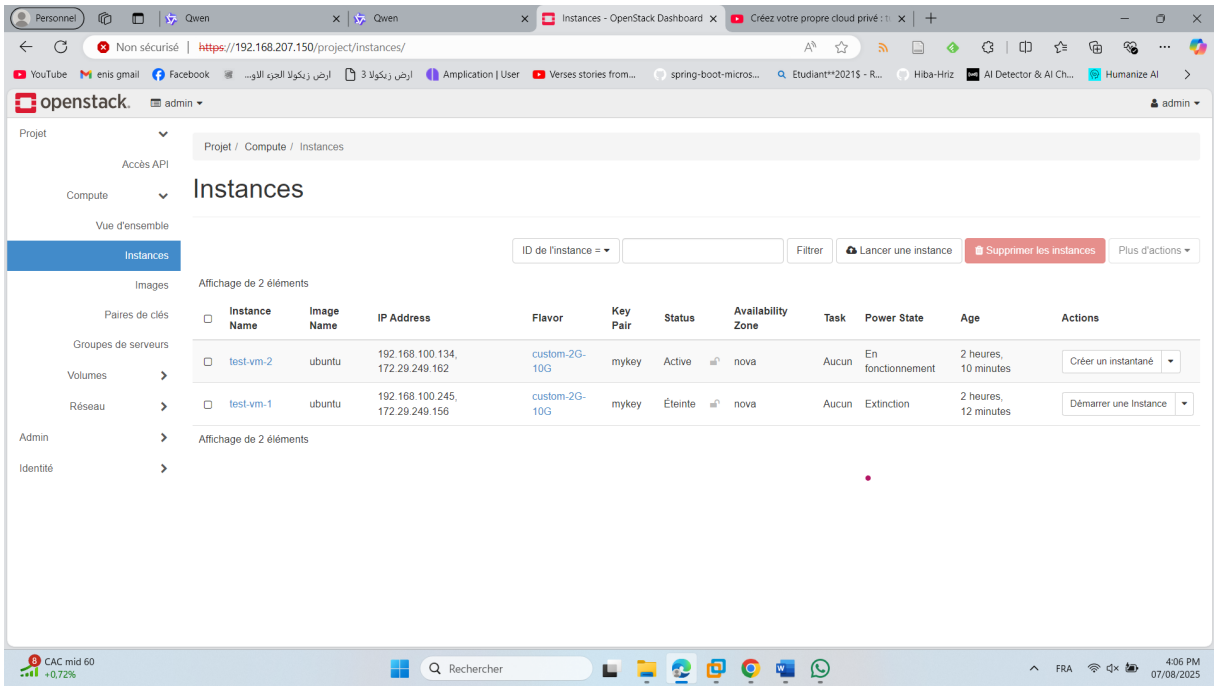


**Figure 10.** *Tableau de bord Horizon: Liste des instances*

Pour confirmer la connectivité réseau et valider le bon fonctionnement des services réseau Neutron, des tests de connectivité ont été effectués depuis l'instance `test-vm-2`. La figure 11 présente les résultats de ces tests de validation.

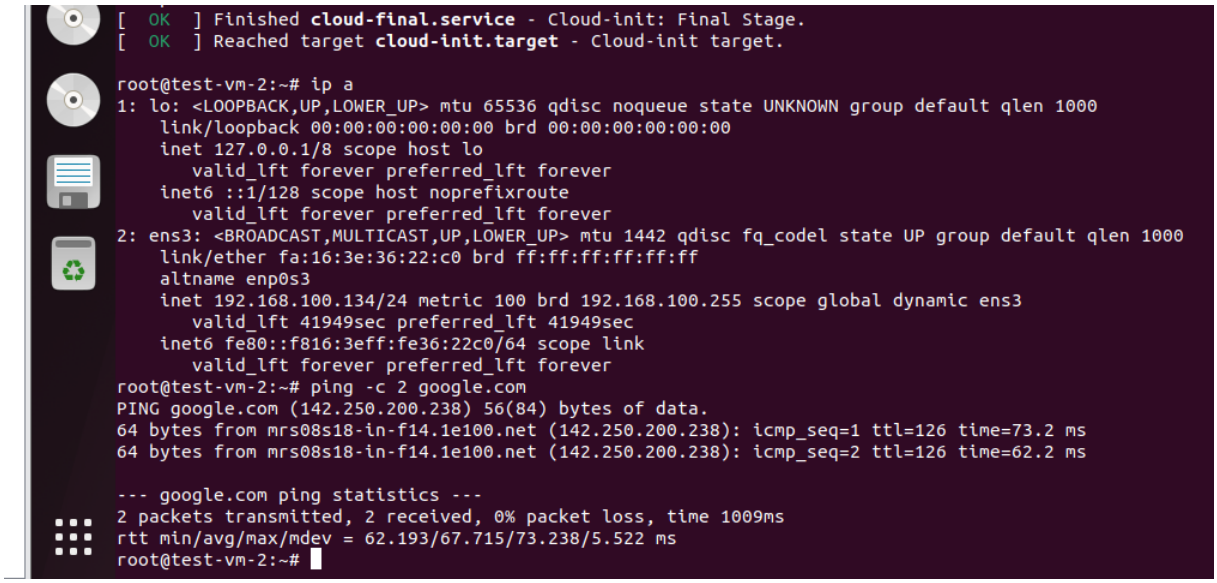


**Figure 11.** *Tests de connectivité réseau depuis l'instance test-vm-2*

Les tests effectués comprennent :

- **Configuration réseau** : La commande `ip a` révèle que l'instance dispose d'une interface de loopback (127.0.0.1) et d'une interface réseau principale avec l'adresse IP 192.168.100.134/24, confirmant l'attribution correcte des adresses IP par le service DHCP de Neutron.
- **Test de connectivité externe** : Le test de ping démontre une connectivité Internet fonctionnelle, avec des temps de réponse satisfaisants et aucune perte de paquets, validant ainsi le bon fonctionnement du routage et de la translation d'adresses (NAT).
- **Résolution DNS** : La résolution successful du nom de domaine `google.com` vers son adresse IP confirme le bon fonctionnement du service DNS configuré dans l'environnement OpenStack.

Ces résultats confirment que l'infrastructure réseau OpenStack est correctement configurée et opérationnelle, permettant aux instances de communiquer aussi bien au sein du cloud privé qu'avec l'extérieur via Internet.

## 5.3 *Conclusion*

La mise en oeuvre du projet a permis de déployer une infrastructure cloud OpenStack de manière automatisée, fiable et reproductible grâce à Ansible. Toutes les étapes, de la préparation des hôtes à la validation fonctionnelle, ont été menées avec rigueur. Les objectifs de simplicité, de maintenabilité et de robustesse ont été atteints. Ce déploiement constitue une base solide pour des extensions futures, telles que l'intégration de nouveaux services, la mise en place de la haute disponibilité ou l'ajout de mécanismes de supervision.

# Conclusion

Ce projet a permis de mettre en évidence l'importance de l'automatisation dans la gestion des infrastructures cloud privées. OpenStack, bien que puissant et flexible, reste complexe à déployer en raison de l'interdépendance de ses nombreux services et des configurations spécifiques nécessaires.

Grâce à Ansible, il est possible de simplifier et d'optimiser ce processus. L'utilisation de playbooks et de rôles modulaires permet de décrire clairement les étapes de configuration, d'assurer une mise en place rapide et cohérente, tout en réduisant les risques d'erreurs humaines et en facilitant la maintenance.

Ainsi, l'automatisation avec Ansible apparaît comme un outil indispensable pour les administrateurs cloud et les ingénieurs systèmes, garantissant la fiabilité, l'efficacité et l'adaptabilité. Ce projet illustre également l'importance croissante des pratiques DevOps et de l'automatisation dans la gestion des infrastructures modernes, contribuant à un déploiement plus sûr, plus rapide et plus contrôlé des environnements cloud.

# Webographie

[1] “Site web de GroupeRif”. In: (2025). URL: https://www.grouperif.com/.

[2] “Inevitable Failure: The Flawed Trust Assumption in the Cloud”. In: (2014). URL: https://www.researchgate.net/publication/281442656_Inevitable_Failure_The_Flawed_Trust_Assumption_in_the_Cloud.

[3] “Host networking”. In: (2025). URL: https://docs.openstack.org/install-guide/environment-networking.html.

[4] “Éléments clés de larchitecture Ansible”. In: (11/2024). URL: https://blog.alphorm.com/architecture-ansible-noeuds-inventaire-playbooks.

[5] “SRE and DevOps: Separate but Connected”. In: (3/2023). URL: https://www.corpit.org/sre-devops-separate-but-connected/.

[6] “Top 5 des outils Open Source pour gérer le serveur OpenStack”. In: (2022). URL: https://toptips.fr/top-5-des-outils-open-source-pour-gerer-le-serveur-openstack/.

[7] “Environment”. In: (2025). URL: https://docs.openstack.org/install-guide/environment.html.

[8] “Ansible community documentation”. In: (7/2025). URL: https://docs.ansible.com/ansible/latest/getting_started/index.html.