

Dédicaces

A nos chers parents et A nos familles,

Dont les mérites, les sacrifices, l'amour et l'affection nous ont permis de finaliser ce travail. Les termes nous manquent pour exprimer toute la reconnaissance, la fierté et le profond amour que nous vous portons pour les sacrifices que vous avez faits pour notre réussite.

Que dieu vous préserve et vous réserve une bonne santé et une longue vie. Tous nos sentiments de reconnaissance.

A nos enseignants tout au long de notre cursus universitaire,

A nos chers collègues,

A tous nos collègues et particulièrement ceux qui nous ont apporté leur aide et leur soutien moral aux moments propices.

A tous ceux que nous aimons et ceux qui nous aiment.



**Yosr
Safa**

Remerciements

C'est avec un grand plaisir que nous nous permettons d'exprimer nos vives et respectueuses reconnaissances envers tous ceux qui nous ont aidés à passer cette période dans les conditions les plus favorables.

Nous remercions Dieu le tout puissant de nous avoir permis de mener à terme ce projet de fin d'études qui est pour nous le point de départ d'une merveilleuse aventure, celle de la recherche, source de remise en cause permanente et de perfectionnement perpétuel.

Nous tenons tout d'abord à remercier notre encadrant Monsieur Mustapha HAMZA maître assistant à l'ISETCOM pour sa patience, son assistance et son suivi incessant par ses directives et ses conseils précieux tout au long de notre travail.

Nous tenons aussi à remercier ISETCOM pour la formation qu'elle nous a fournie non seulement au niveau technique mais aussi relationnel. Les tâches auxquelles nous avons été associées, dans le cadre de ce projet de fin d'études, nous ont permis de consolider nos connaissances et d'en développer de nouvelles.

Nous adressons nos profondes gratitude à M. Boulbaba LABIADH et M. Chafik HAMMAMI, pour l'opportunité qu'il nous a offerte pour travailler sur ce projet, pour son aide et son soutien.

Nos vifs remerciements s'adressent aux membres du jury pour avoir accepté de juger notre travail.

Nous tenons finalement à remercier toutes les personnes ayant apporté leurs aides et collaborations pour la bonne réussite de notre projet de fin d'études.

Table des matières

Introduction générale	1
Chapitre 1 : Contexte du projet.....	2
Introduction.....	2
1. Présentation de l'entreprise d'accueil : NextStep IT.....	2
1.1 Présentation générale	2
1.2 Services de NextStep IT	2
1.2.1 Audit et conseil	2
1.2.2 Intégration	2
1.2.3 Assistance technique	3
1.2.4 Support et maintenance	3
2. Cadre du projet.....	3
2.1 Description de l'existant	3
2.2 Problématique et motivations.....	4
2.3 Objectifs du projet	4
2.4 Solutions du Cloud existantes	4
2.4.1 CloudStack	5
2.4.2 OpenNebula	5
2.4.3 OpenStack.....	5
2.5 Solution d'automatisation	7
3. Planification du projet	7
Conclusion	8
Chapitre 2 : Etat de l'art	9
Introduction.....	9
1. Cloud Computing.....	9
1.1 Services du Cloud Computing.....	9
1.2 Modèles de déploiement du Cloud Computing	10
1.3 Virtualisation : le socle du Cloud Computing	10
1.3.1 Présentation	10
1.3.2 Hyperviseurs	11
1.3.2 Formes de virtualisation	12
1.3 Automatisation des Clouds.....	13
2. OpenStack.....	14
2.1 Présentation	14
2.2 Architecture d'OpenStack	14
2.2.1 Keystone (Identité)	14
2.2.2 Nova (Compute).....	15
2.2.3 Glance	15
2.2.4 Quantum (Neutron).....	15

2.2.5 Cinder.....	15
2.2.6 Swift	16
2.2.7 Horizon.....	16
2.3 Installateurs d'OpenStack.....	16
2.3.1 Devstack.....	16
2.3.2 Packstack	16
2.3.3 Fuel	17
2.4 Architectures de déploiement d'OpenStack	17
3. Ansible	18
3.1 Présentation	18
3.2 Architecture d'Ansible	18
4. AWX	19
Conclusion	19
Chapitre 3 : Analyse et spécification des besoins	20
Introduction.....	20
1. Besoins fonctionnels.....	20
1.1 Gestion des projets.....	20
1.2 Gestion des ressources	20
1.3 Exécution des tâches	20
1.4 Gestion des paiements	20
1.5 Gestion des utilisateurs	21
2. Besoins non fonctionnels	21
3. Identification des acteurs.....	21
4. Conception	21
4.1 Diagramme de cas d'utilisation générale	22
4.2 Raffinements des Cas d'Utilisations	22
4.2.1 Cas d'utilisation « Consulter l'état du Cloud »	22
4.2.2 Cas d'utilisation « Gérer les instances »	23
4.2.3 Cas d'utilisation « Gérer les services Clouds »	24
4.2.4 Cas d'utilisation « Gérer les gabarits »	24
4.2.5 Cas d'utilisation « Gérer les images ».....	25
4.2.6 Cas d'utilisation « Consulter la liste des utilisateurs »	25
4.3 Diagramme de séquence	26
4.3.1 Diagramme de séquence « Authentification ESXI »	26
4.3.2 Diagramme de séquence « Création d'une VM »	27
4.3.3 Diagramme de séquence « Authentification OpenStack »	27
4.3.4 Diagramme de séquence « Ajout d'une règle de filtrage IP ».....	28
4.3.5 Diagramme de séquence « Lancement d'une instance »	29
4.3.6 Diagramme de séquence « Automatisation d'OpenStack à de l'outil Ansible ».....	30
Conclusion	30
Chapitre 4 : Réalisation	31

Introduction.....	31
1. Mise en place de l'IaaS : OpenStack.....	31
1.1 Installation et configuration d'OpenStack.....	31
1.3 Manipulations des fonctionnalités d'OpenStack.....	31
1.2.1 Création du projet	32
1.2.2 Création de l'utilisateur « ABF »	32
1.2.3 Création du réseau externe	33
1.2.4 Création du réseau interne.....	34
1.2.5 Création d'un routeur	35
1.2.6 Création d'un groupe de sécurité.....	36
1.2.7 Création d'une paire de clés	36
1.2.8 Allocation d'une IP flottante.....	36
1.2.9 Lancement de l'instance	37
1.2.10 Définition des quotas.....	38
1.2.11 Création d'un conteneur	39
2. Implémentation d'Ansible	40
3. Playbooks Ansible.....	42
4. AWX	55
4.1 Exécution d'un playbook sur le serveur AWX	55
4.2 Avantages d'AWX	57
4.2.1 Multi-playbooks workflows	57
4.2.2 Enquête.....	58
Conclusion	60
Conclusion générale	61
Bibliographie	62
Annexe 1 : Création d'une machine virtuelle sur VMware ESXI	63
Annexe 2 : Installation d'OpenStack	65
Annexe 3 : Implémentation d'AWX.....	70
Annexe 4 : Ajout des enquêtes d'AWX.....	72

Table des figures

Figure 2.1 : Les services du Cloud Computing.....	10
Figure 2.2 : Hyperviseur Type 1.....	11
Figure 2.3 : Hyperviseur Type 2.....	11
Figure 2.4 : Architecture d'OpenStack	14
Figure 2.5 : L'architecture d'Ansible.....	18
Figure 3.1 : Diagramme de cas d'utilisation générale.....	22
Figure 3.2 : Cas d'utilisation "Consulter l'état du Cloud"	23
Figure 3.3 : Cas d'utilisation "Gérer les instances"	23
Figure 3.4 : Cas d'utilisation "Gérer les services Clouds"	24
Figure 3.5 : Cas d'utilisation "Gérer les gabarits"	24
Figure 3.6 : Cas d'utilisation "Gérer les images"	25
Figure 3.7 : Cas d'utilisation "Consulter la liste des utilisateurs "	25
Figure 3.8 : Diagramme de séquence de l'authentification sur l'hyperviseur ESXI	26
Figure 3.9 : Diagramme de séquence de la création d'une VM.....	27
Figure 3.10 : Diagramme de séquence de l'authentification OpenStack	28
Figure 3.11 : Diagramme de séquence d'ajout d'une règle de filtrage IP.....	28
Figure 3.12 : Diagramme de séquence du lancement d'une instance.....	29
Figure 3.13: Diagramme de séquence de l'automatisation d'OpenStack	30
Figure 4.1 : Les étapes de la création d'un environnement Cloud sur OpenStack.....	32
Figure 4.2 : Création du projet	32
Figure 4.3 : Création de l'utilisateur "ABF"	32
Figure 4.4 : Le réseau externe d'ABF.....	33
Figure 4.5 : Le sous-réseau externe d'ABF	33
Figure 4.6 : Le réseau interne d'ABF	34
Figure 4.7 Le sous-réseau interne d'ABF.....	34
Figure 4.8 : le routeur ABF	35
Figure 4.9 : L'ajout de l'interface qui relie le routeur avec le réseau interne.....	35
Figure 4.10 : Les règles du groupe de sécurité ABF	36
Figure 4.11 : La paire de clés ABF.....	36
Figure 4.12 : Allocation d'une IP flottante	37
Figure 4.13 : L'instance ABF	37
Figure 4.14 : Le volume ABF.....	38
Figure 4.15 : Attachement de volume ABF	38
Figure 4.16 : Les quotas de service nova	39
Figure 4.17 : Le quota de service cinder	39
Figure 4.18 : Génération des clés SSH.....	40
Figure 4.19 : Inventaire Ansible	41
Figure 4.20 : Test de connectivité	41
Figure 4.21 : Playbook "projet.yml"	42

Figure 4.22 : Exécution du playbook "projet.yml"	43
Figure 4.23 : Playbook "utilisateur.yml »	43
Figure 4.24: Playbook "attacher_un_utilisateur.yml"	44
Figure 4.25 : Playbook "quotas.yml"	45
Figure 4.26 : Playbook "réseau_interne.yml"	46
Figure 4.27 : Playbook "réseau_externe.yml"	47
Figure 4.28 : Playbook "routeur.yml"	48
Figure 4.29: Playbook "groupe_de_sécurité.yml"	49
Figure 4.30: Playbook "paire_de_clés.yml "	50
Figure 4.31 : Playbook "instance.yml"	51
Figure 4.32 : Ansible_instance	51
Figure 4.33 : Playbook "ip_flottante.yml "	52
Figure 4.34 : Playbook "instantané.yml"	53
Figure 4.35 : Playbook "volume.yml "	53
Figure 4.36 : Playbook "conteneur.yml"	54
Figure 4.37 : Création d'une nouvelle organisation	55
Figure 4.38 : Création de l'inventaire OpenStack	55
Figure 4.39 : l'identifiant d'OpenStack	56
Figure 4.40 : L'ajout d'un projet dans AWX.....	56
Figure 4.41 : Le modèle "projet"	57
Figure 4.42 : Le modèle "utilisateur"	57
Figure 4.43 : L'exécution du flux de travail "test"	58
Figure 4.44 : L'enquête de projet	58
Figure 4.45 : Saisie de la variable "projet"	59
Figure 4.46 : Saisie des variables nécessaires pour créer un utilisateur.....	59
Figure 4.47 : Saisie des variables de l'enquête "quotas"	59

Table des tableaux

Tableau 1.1 : Comparaison entre les solutions du Cloud.....	6
Tableau 1.2 : Planification prévisionnelle du projet	7
Tableau 2.1 : Les installateurs d'OpenStack.....	17

Liste des acronymes

AWX : Ansible Worx

API : Application Programming Interface

APT : Advanced Packaging Tool

CE : Docker Compose

ESI : Elastic Sky X Integrated

FSC : Fournisseur de Service Cloud

GPG : GNU Privacy Guard

HA : Haute Disponibilité

IaaS : Infrastructure as a Service

ICMP : Internet Control Message Protocol

ISCSI : Internet Small Computer Systems Interface

KVM : Kernel-based Virtual Machine

LXC : Linux Containers

NAT : Network Address Translation

NPM : Node Package Manager

PaaS : Platform as a Service

PPA : Personal Package Archives

PXE : Preboot eXecution Environment

QEMU : Quick EMUlator

QoS : Quality of Service

RAM : Random Access Memory

RSA : Rivest Shamir Adleman

SaaS : Software As A Service

SDN : Software-Defined Networking

SSH : Secure Shell

UML : Unified Modeling Language

VCPU : Virtual Central Processing Unit

VM : Virtual Machine

VLAN : Virtual Local Area Network

VPN : Virtual Private Network



Introduction générale

Le développement remarquable du Cloud Computing (nuage informatique) attire l'intérêt de plusieurs acteurs dans le monde informatique. Certes, il promet un changement dans l'univers d'investissement et d'exploitation des ressources IT.

Avec le Cloud, les utilisateurs n'ont plus besoin d'investir lourdement dans des ressources informatiques. Ils ont la possibilité à les louer et à les exploiter par le biais de l'Internet en mode service à la demande.

Le Cloud permet aux entreprises de se focaliser sur leurs business sans se soucier de l'infrastructure matérielle sous-jacente. En effet, les ressources allouées sont en permanence contrôlées et optimisées par le fournisseur du Cloud, ce qui renforce le rendement et la disponibilité des services Clouds.

Le nuage n'étant pas juste une utilisation d'infrastructure mais aussi une livraison des services à une clientèle de plus en plus grande et soucieuse d'une rapidité de traitement des besoins avec une bonne qualité de service. Suite à cette évolutivité, il est temps d'appliquer aux Clouds un processus d'automatisation.

L'automatisation dans le Cloud permet de configurer les ressources IT avec une intervention humaine très réduite et de répondre aux besoins des clients au fur et à mesure qu'ils se présentent. Cela permet de faciliter l'utilisation du Cloud et de garantir une haute disponibilité des services.

Ayant pour but d'accompagner l'évolution technologique du marché en termes du Cloud Computing et étant conscient des bénéfices qu'il peut apporter à ses clients, NextStep IT a décidé d'enrichir son offre commerciale en offrant des solutions Cloud innovantes et open source.

Le présent rapport retrace l'approche que nous avons adoptée pour aborder le projet, en parallèle avec les outils dont nous avons fait usage et qui ont fait l'objet d'études bien minutieuses.

Le premier présente l'entreprise d'accueil NextStep IT ainsi que le cadre général du projet.

Dans le deuxième chapitre, nous allons introduire les techniques de virtualisation et du Cloud Computing et nous allons présenter la solution choisie.

Le troisième chapitre comporte en premier lieu la spécification des besoins fonctionnels et non fonctionnels et la dernière partie sera consacrée pour la conception.

Le quatrième chapitre sera dédié à la mise en place de l'IaaS OpenStack et à l'automatisation des tâches offertes par cette infrastructure.

Ce rapport se termine par une conclusion générale sur la totalité du travail achevé durant le stage.

Chapitre 1 : Contexte du projet

Introduction

Le Cloud est devenu un outil incontournable dans le monde informatique. Chacun souhaite pouvoir disposer à tout moment des ressources IT nécessaires pour répondre à ses besoins et atteindre ses objectifs.

Avant d'entamer les aspects techniques du projet, nous allons traiter trois parties : la présentation de la société où s'est déroulé notre stage, une étude de l'architecture Cloud existant de NextStep IT et une planification des tâches effectuées.

1. Présentation de l'entreprise d'accueil : NextStep IT

1.1 Présentation générale

NextStep IT est créé en 2012. Elle propose à ses clients des outils et des solutions sur mesure basées sur les dernières innovations, et ce, afin de les aider à transformer, à différencier et à développer leur entreprise. Elle conçoit, déploie et met en exploitation des solutions adaptées aux environnements de travail [1].

NextStep IT dispose de deux pôles : le premier est dédié à l'étude, au conseil et à l'assistance technique et le deuxième permet d'intégrer des solutions innovantes du Cloud.

1.2 Services de NextStep IT

1.2.1 Audit et conseil

NextStep IT intègre le service d'audit des infrastructures des systèmes d'information. Les bénéfices d'audit permettent d'établir un état des lieux des infrastructures informatiques existant dans le but d'évaluer les besoins et mettre en évidence tous les aspects nécessaires afin d'améliorer l'environnement et l'organisation [1].

1.2.2 Intégration

NextStep IT est spécialisé dans l'intégration des solutions de sécurité, dans l'infrastructure réseau et dans les communications et système. Elle affecte une équipe composée d'un chef de projet, des ingénieurs et techniciens et d'un ingénieur commercial pour la durée du projet. Tous les membres du projet de NextStep IT sont certifiés et formés en continu sur ses solutions et sur les différentes technologies, par les constructeurs partenaires [1].

Tous les projets d'intégration de NextStep IT doivent passer par une étape de préparation et de planification, une phase de conception, une phase d'implémentation et de migration et enfin un transfert de compétences et une assistance au démarrage.

1.2.3 Assistance technique

NextStep IT met à la disposition de ses clients la compétence technique de son équipe et les outils informatiques nécessaires dans le but d'apporter l'assistance adéquate.

L'expertise et le professionnalisme de NextStep IT assurent à ses clients :

- La réactivité : les équipes agissent directement là où ses clients le souhaitent, en fonction de leurs contraintes.
- L'efficacité : les techniciens fournissent la solution la plus adéquate dans des meilleurs délais.
- La transparence : les actes techniques font l'objet de rapports et de transferts de compétences.
- Le savoir-faire : le problème du client de NextStep IT est pris en charge par une équipe certifiée, qualifiée et expérimentée.

1.2.4 Support et maintenance

L'efficacité du centre de support dans le maintien du support réseau pour les clients assure la résolution rapide des incidents et la maîtrise de l'évolution de la solution.

Cette efficacité vient de :

- La rapidité de réponse à la demande.
- La capacité des intervenants qui traitent la demande.
- Ressources matérielles utilisées pour remplacer les équipements défectueux.
- L'établissement d'un fort partenariat avec les constructeurs.

2. Cadre du projet

Notre étude sera menée essentiellement sur le département Cloud de NextStep IT, où s'est déroulé notre stage. Le but de notre projet est de mettre en place un environnement Cloud et l'automatiser afin d'assurer une agilité et une fiabilité d'utilisation.

2.1 Description de l'existant

NextStep IT dispose d'une large gamme des services Clouds basée sur les dernières innovations. Elle offre à ses clients des solutions complètes et propriétaires dont le coût d'achat des licences est très important.

Avec les infrastructures propriétaires, les utilisateurs sont totalement dépendants des sociétés éditrices ce qui affecte la continuité et la disponibilité des services.

Ces facteurs ont permis aux solutions open source de devenir une alternative intéressante. C'est dans ce cadre que NextStep IT vise à les adopter afin d'enrichir son offre commerciale.

2.2 Problématique et motivations

NextStep IT cherche à déployer une solution Cloud open source automatisée permettant de remédier aux faiblesses suivantes :

- La non-interopérabilité entre les différents types d'hyperviseurs.
- Un coût élevé des solutions propriétaires existantes.
- Un nombre d'effectifs important pour gérer les ressources des infrastructures.
- Les administrateurs sont responsables de la maintenance et la surveillance des systèmes, donc ils peuvent commettre des erreurs qui affaiblissent l'efficacité et la stabilité des ressources.

Les principales motivations pouvant influencer l'adoption d'une solution alternative aux infrastructures Clouds existant de NextStep IT sont les suivantes :

- Une flexibilité massive.
- Un ensemble varié de fonctionnalités.
- Une réduction des coûts.
- Un gain du temps grâce au processus de l'automatisation afin de permettre aux équipes de l'entreprise de se focaliser sur des tâches à valeur ajoutée.

2.3 Objectifs du projet

L'objectif principal de notre projet est de garantir une meilleure exploitation des infrastructures Clouds d'une façon automatisée, plus souple et flexible. En effet, notre solution doit assurer les services suivants :

- La gestion des ressources IT par le biais d'une interface graphique.
- L'infrastructure Cloud doit réagir pour fournir des services performants dans les plus brefs délais selon les besoins des entreprises.
- L'indépendance des contraintes techniques des sociétés éditrices des solutions propriétaires (par exemple le middleware spécifique pour certaines infrastructures physiques).
- L'automatisation de la création des environnements Cloud via des scripts ainsi que leur configuration. En effet, elle rend les services plus réactifs et elle évite aussi les risques d'accidents liés aux erreurs humaines.

2.4 Solutions du Cloud existantes

Le service de base le plus connu du Cloud Computing est l'IaaS (infrastructure as a service), qui fournit un socle d'infrastructure informatique virtualisé, en offrant des ressources informatiques (serveurs, stockage, réseaux) à la demande selon les besoins des clients. Ceci permet aux entreprises d'externaliser leurs systèmes informatiques et donc de réduire leurs coûts informatiques tout en étant plus efficaces et plus flexibles [2].

Dans le domaine du Cloud Computing, il existe différentes solutions open source qui sont conçues pour les architectures privées internes.

2.4.1 CloudStack

CloudStack est un logiciel conçu pour être une plateforme du Cloud Computing hautement disponible et évolutive.

Il permet de déployer et de gérer de grands réseaux de machines virtuelles. De nombreux fournisseurs de services l'utilisent pour fournir des services du Cloud public, privé et hybride.

2.4.2 OpenNebula

OpenNebula est un logiciel sous licence Apache, qui fournit un ensemble de fonctions pour gérer le Cloud Computing.

Ce logiciel organise le fonctionnement d'un groupe de serveurs physiques et fournit des ressources pour les machines virtuelles. Il coordonne et gère le cycle de vie de toutes les machines virtuelles.

OpenNebula permet la virtualisation des réseaux, virtualisation des serveurs ainsi que l'intégration d'autres nœuds.

2.4.3 OpenStack

OpenStack est une plateforme logicielle permettant principalement le déploiement d'un service IaaS, où des serveurs virtuels et des ressources IT sont mises à la disposition des clients.

La gestion d'OpenStack est effectuée par le biais d'un tableau de bord web ou d'une ligne de commande.

Les solutions du Cloud existantes se caractérisent par un ensemble de critères qui sont :

- Le type du Cloud : il existe trois modèles de déploiement de services Cloud qui sont public, privé et hybride.
- L'architecture supportée : un seul composant ou modulaire.
- La disponibilité des services en ligne.
- Le balancement des charges qui permet une meilleure gestion des ressources Cloud.
- Le stockage d'objets qui permet la gestion d'un très grand volume de données non structurées.
- Le SDN qui est une approche de l'architecture réseau qui permet de contrôler ou de programmer le réseau de manière intelligente et centralisée.

Le tableau 1.1 présente une étude comparative entre les solutions du Cloud Computing.

Tableau 1.1 : Comparaison entre les solutions du Cloud

Solutions Caractéristiques		OpenStack	CloudStack	OpenNebula
Type du Cloud		Public , Hybride , Privé	Public , Hybride , Privé	Public, Hybride , Privé
Architecture		Modulaire	1 composant	1 composant
Contrôleurs	Haute Disponibilité	Oui	Oui	Oui
	Balancement de charge	Oui	Oui	Non
Stockage d'objets		swift, cinder		
Hyperviseurs		KVM, XenServer, Xen, QEMU, LXC, VMWare, Hyper-V, Docker, Baremetal	KVM, XenServer, Xen, QEMU, LXC, VMWare, Hyper-V	KVM , XenServer
Haute disponibilité d'instances		Oui	Oui	Non
Documentation		Forte	Faible	Moyenne
Communauté		Grande	Moyenne	Faible
Activité développement		Très élevée	Moyenne	Faible
Isolation Simple (type VLAN)		Oui	Oui	Oui
SDN		Oui	Non	Oui
QoS		Oui	Non	Oui
Firewalls		Oui	Oui	Oui
VPN		Oui	Oui	Non

Dans le tableau 1.1, nous avons présenté les solutions open source permettant la création d'un Cloud privé.

Notre choix s'est fixé sur OpenStack pour les raisons suivantes :

- OpenNebula ne supporte que les hyperviseurs KVM et XenServer.
- CloudStack ne dispose pas d'une forte documentation. De plus, il n'est pas utilisable par une forte communauté.
- CloudStack et OpenNebula possèdent des modules d'automatisation intégrés.

2.5 Solution d'automatisation

L'évolutivité d'OpenStack dépend de la valeur du temps nécessaire pour configurer et exécuter des environnements Cloud, ainsi que la réduction opérationnelle des coûts. Pour tout cela, il faut un outil automatisé de déploiement et de configuration.

Dans ce cadre, NextStep IT nous propose d'étudier la plateforme logicielle Ansible fournie par Red Hat (premier éditeur mondial des solutions logicielles open source).

Ansible permet d'apporter une souplesse et une agilité à OpenStack. En effet, il lui permet de configurer ses tâches et de gérer ses ressources d'une manière permanente.

3. Planification du projet

Parmi les différentes phases d'un projet, la planification est sans doute l'une des plus importantes. Elle consiste à décrire les tâches à réaliser.

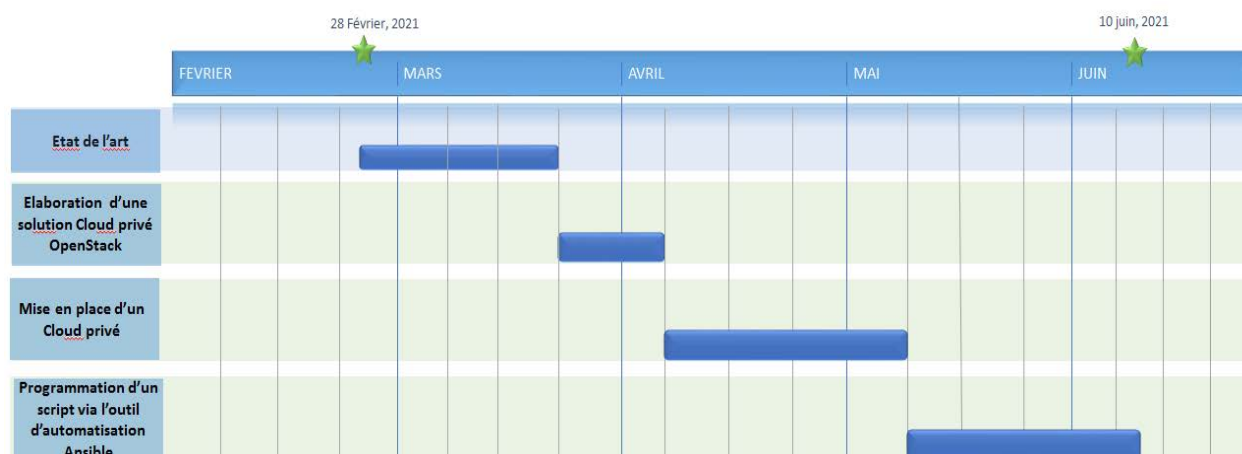
Un planning prévisionnel est dressé afin de bien suivre les différentes étapes de déroulement du projet. Il sera ainsi possible de déterminer si les objectifs fixés sont atteints et de suivre l'avancement à travers un tableau de bord de suivi de projet. Une planification des tâches, rigoureuse et réaliste, sera donc le garant de la progression et de la réussite du projet.

Notre projet se compose principalement de 4 phases :

- Etude du Cloud Computing en terme de principes, du type et des technologies utilisées.
- Elaboration d'une solution Cloud Computing qui répond aux besoins de NextStep IT.
- Mise en place d'un Cloud privé opérationnel qui optimise l'utilisation des ressources de l'entreprise.
- Programmation d'un script via l'outil d'automatisation Ansible (en lançant des instances avec une image précise aux besoins du client ainsi qu'une connectivité SSH).

Le diagramme de Gantt 1.2 représente la planification prévisionnelle définie pour l'implémentation de la solution et la rédaction du rapport associé.

Tableau 1.2 : Planification prévisionnelle du projet



Comme le montre le tableau 1.2, la phase de l'état de l'art a été effectuée pendant trois semaines et demi. La deuxième phase a duré deux semaines, la troisième phase a été accomplie en un mois et la dernière a été réalisée pendant le reste du projet.

Conclusion

Dans ce chapitre, nous avons présenté la société d'hébergement NextStep IT du projet. Ensuite, nous avons donné le cadre général du notre projet en précisant les défis et les motivations.

Dans le chapitre suivant, nous allons étudier les concepts théoriques du projet.

Chapitre 2 : Etat de l'art

Introduction

Afin de mieux cerner l'étendue de notre problématique, nous allons revenir sur les aspects techniques liés au Cloud Computing et à la virtualisation. Ensuite, nous présentons la plateforme d'orchestration des ressources OpenStack ainsi que le projet Red Hat Ansible et nous clôturons par la description de l'interface graphique AWX.

1. Cloud Computing

Le Cloud Computing désigne la livraison des services et des ressources à la demande par le biais d'internet pour offrir des ressources flexibles, des économies d'échelle et une innovation plus rapide [2].

Le Cloud Computing est caractérisé par :

- La disponibilité mondiale en libre-service : en Cloud Computing la demande est automatique et la réponse est immédiate.
- L'ouverture : les services de l'informatique en nuage sont accessibles via Internet.
- La mutualisation : Cloud permet de combiner des ressources hétérogènes. Ainsi, elle permet l'adaptation des ressources en se basant sur les besoins des clients.
- Le paiement à l'usage : la quantité des services en Cloud Computing est mesurée.

1.1 Services du Cloud Computing

Les services cloud peuvent prendre la forme d'une infrastructure, d'une plateforme ou d'un logiciel, hébergés par des fournisseurs tiers et mis à la disposition des utilisateurs par Internet [3].

Les principaux services proposés en Cloud Computing sont :

- IaaS (Infrastructure as a Service) qui permet la libération de l'entreprise de toute infrastructure informatique (serveurs, machines virtuelles, stockage, réseaux, systèmes d'exploitation). Les utilisateurs n'ont pas besoin d'investir dans le matériel, ils vont louer une infrastructure informatique auprès d'un fournisseur de services Cloud, avec un paiement en fonction de l'utilisation.
- PaaS (Platform as a Service) qui propose à l'utilisateur un environnement à la demande pour développer, tester, fournir et gérer des applications logicielles. Elle fournit en plus de l'infrastructure technique des composants logiciels intégrés.
- SaaS (Software as a Service) qui fournit des applications prêtes à l'emploi s'exécutant sur l'infrastructure du fournisseur et accessibles via le navigateur du client.

La figure 2.1 représente la différence entre les trois services du Cloud.

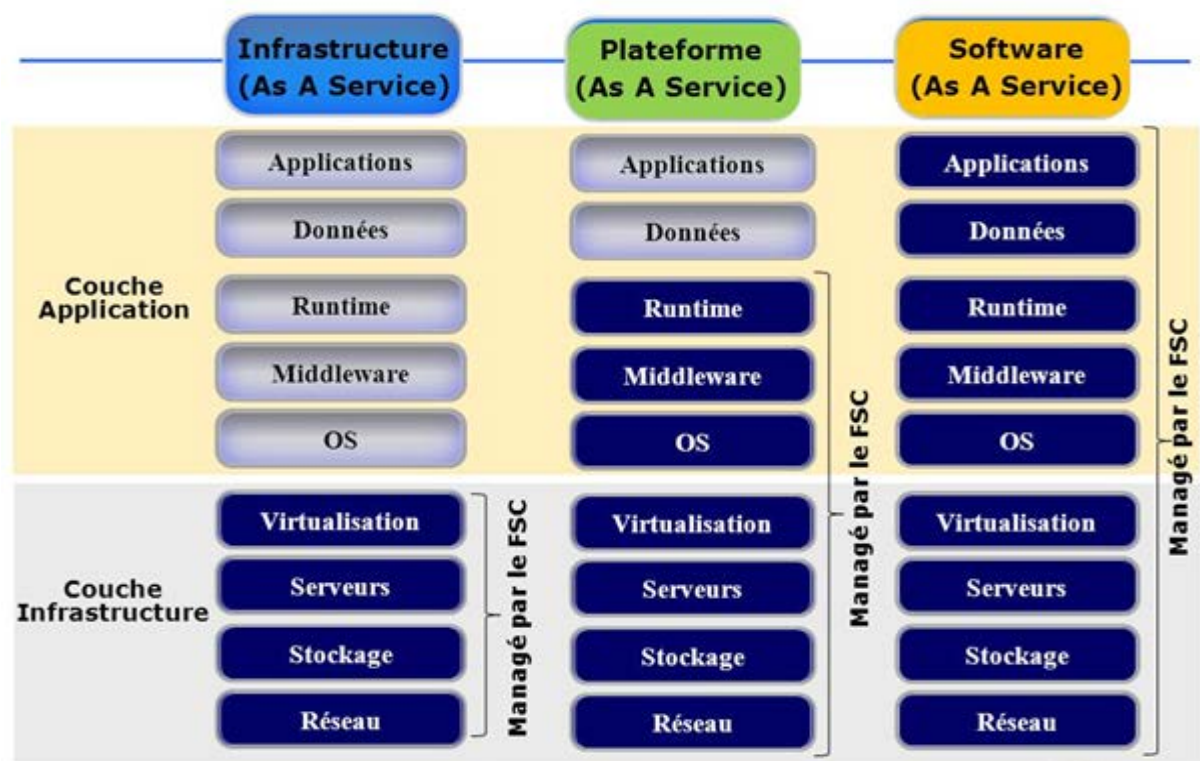


Figure 2.1 : Les services du Cloud Computing

1.2 Modèles de déploiement du Cloud Computing

Il existe trois modes de déploiement de services Cloud :

- Le Cloud public correspond à une externalisation de l'infrastructure chez un fournisseur tiers avec des ressources mutualisées ouvertes à tous. Microsoft Azure est un exemple du Cloud public [2].
- Le Cloud privé est un ensemble de ressources du Cloud Computing dédié à une entreprise ou à une organisation. Il peut être physiquement situé dans le centre de données local de l'entreprise (Cloud privé externe). Certaines entreprises paient également des fournisseurs de services pour héberger leurs Clouds privés (Cloud privé interne).
- Le Cloud hybride rassemble des Clouds publics et privés. Il permet de disposer de ressources d'infrastructure privées en interne et publiques en externe.
- Le Cloud communautaire consiste à partager un espace donné entre plusieurs entreprises ayant les mêmes exigences en matière de sécurité et de confidentialité [2].

1.3 Virtualisation : le socle du Cloud Computing

1.3.1 Présentation

La virtualisation et le Cloud Computing sont deux concepts différents, mais complémentaires.

La virtualisation est un ensemble de techniques et d'outils permettant de faire fonctionner plusieurs systèmes d'exploitation sur une même machine physique grâce à un hyperviseur.

1.3.2 Hyperviseurs

Un hyperviseur (aussi appelé Virtual Machine Monitor, soit VMM) est un logiciel qui permet à plusieurs systèmes invités de fonctionner sur un seul système hôte [4]. Son rôle est d'assurer le contrôle du processeur et des ressources de la machine hôte. Il permet d'allouer alternativement à chaque VM ses besoins.

D'ailleurs, il existe deux types d'hyperviseurs :

Hyperviseur type 1 (natif, bare métal) : est un logiciel qui s'exécute directement sur le hardware. Il s'agit d'un noyau allégé et optimisé pour ne faire tourner que des noyaux de systèmes d'exploitation invités à cette architecture spécifique. Ces systèmes invités ayant conscience d'être virtualisés [4].

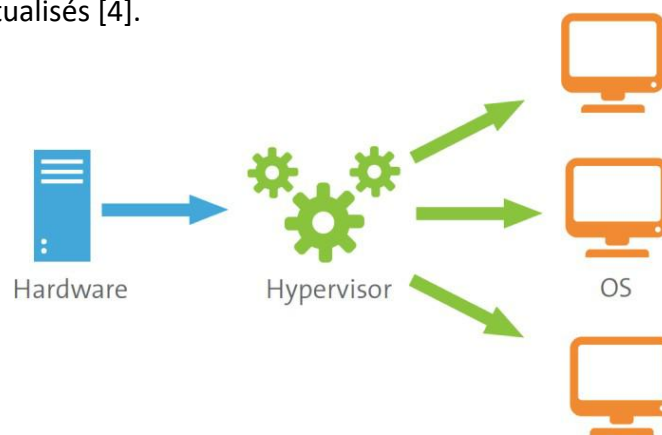


Figure 2.3 : Hyperviseur Type 1

Le système d'exploitation invité n'a pas besoin d'être modifié pour pouvoir fonctionner sur un hyperviseur du type 1.

Hyperviseur type 2 (hébergé) : est un logiciel qui s'exécute à l'intérieur d'un système d'exploitation hôte. Par conséquent, les systèmes invités traversent deux couches logicielles pour accéder au hardware. L'installation et la configuration de ce type de système de virtualisation présentent un gros avantage [4].

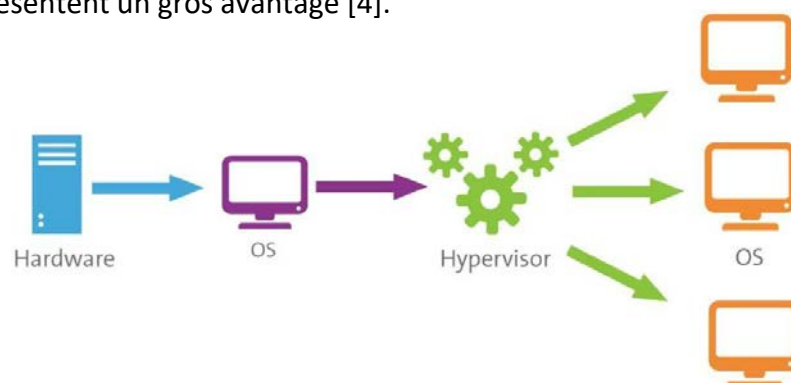


Figure 2.4 : Hyperviseur Type 2

1.3.2 Formes de virtualisation

Il y a diverses formes de virtualisation qui se réfèrent à l'abstraction des ressources informatiques telles que les logiciels, la mémoire, les données ou les composants réseaux.

Une distinction est donc faite entre :

- Virtualisation matérielle
- Virtualisation logicielle
- Virtualisation de la mémoire
- Virtualisation des données
- Virtualisation du réseau

a) Virtualisation matérielle

Le terme de virtualisation matérielle désigne les technologies qui permettent de fournir des composants matériels à l'aide des logiciels, quelle que soit leur base physique. En effet, l'exemple classique d'une virtualisation matérielle est la machine virtuelle.

Une VM est une machine virtuelle qui se comporte pour l'utilisateur final comme une machine physique, y compris avec un matériel et un système d'exploitation. Elle fonctionne sur un ou plusieurs systèmes physiques appelés hôtes. Avec la virtualisation matérielle, la couche d'abstraction entre la base physique et le système virtuel est gérée par un hyperviseur [4].

b) Virtualisation logicielle

Si les composants logiciels sont virtualisés au lieu de composants matériels, nous parlons de virtualisation logicielle.

Les approches communes de ce concept de virtualisation sont :

- Virtualisation des applications : elle consiste à encapsuler l'application et son contexte d'exécution système dans un environnement cloisonné (isolé) et de se répartir sur plusieurs systèmes sans avoir à modifier les systèmes de fichiers, les systèmes d'exploitation locaux ou le registre.
- Virtualisation de postes de travail : est un concept avec lequel les environnements de bureau sont fournis de manière centralisée et accessible via un réseau. Les entreprises utilisent souvent cette approche. Microsoft et VMware sont les principaux fournisseurs de solutions logicielles de virtualisation de postes de travail qui s'appuient sur des technologies propriétaires [4].
- Virtualisation au niveau du système d'exploitation : elle s'appuie sur les fonctions natives du noyau des systèmes d'exploitation Unix, qui permettent d'exécuter plusieurs instances d'espace utilisateur isolé en parallèle [4].

c) Virtualisation de la mémoire

La virtualisation du stockage vise à cartographier d'une façon virtuelle les ressources de stockage d'une entreprise telle que la mémoire flash, les disques durs ou les lecteurs de bandes et à les rendre disponibles comme un pool de stockage lié.

d) Virtualisation des données

La virtualisation des données est une couche logique de données intégrant toutes les données d'entreprise réparties entre les différents systèmes. Elle gère l'ensemble de données unifiées pour les sécuriser et les gouverner de manière centralisée, puis de les fournir en temps réel aux utilisateurs [4].

e) Virtualisation des réseaux

La virtualisation du réseau permet d'extraire les ressources matérielles et logicielles du réseau à partir de leur fondement physique.

Deux objectifs sont au premier plan :

- Les ressources physiques du réseau doivent être combinées en une unité logique grâce à la virtualisation.
- Les ressources physiques du réseau doivent être divisées en différentes unités virtuelles grâce à la virtualisation.

Le SDN est un bon exemple de virtualisation du réseau.

1.3 Automatisation des Clouds

Le déploiement et l'exploitation de charge de travail dans les entreprises étaient un processus manuel fastidieux. Il implique souvent des tâches répétitives : dimensionnement, provisionnement et configuration de ressources telles que des machines virtuelles, l'équilibrage de charge, l'élaboration de réseaux virtuels et la surveillance et la gestion de la disponibilité de ces ressources.

Ces processus manuels et répétitifs s'avèrent globalement improductifs et comportent des erreurs qui peuvent nécessiter un dépannage, et donc retarder la disponibilité des services [5].

Pour répondre à tous ces besoins, les environnements Clouds doivent être automatisés. Cette robotisation des activités permettant l'exploitation des ressources informatiques a plusieurs apports bénéfiques :

- Une accélération du traitement des processus.
- Une diminution des coûts des opérations de services informatiques.
- Une réduction significative du nombre d'erreurs humaines.
- Une standardisation des méthodes.
- Une simplification des contrôles [6].

2. OpenStack

2.1 Présentation

OpenStack est une plateforme open source, libre qui permet de créer et de gérer des Clouds privés ou publics à partir des pools de ressources virtuelles. Les projets qui constituent la plateforme OpenStack assurent les principaux services du Cloud Computing : le calcul, la mise en réseau, le stockage et la gestion des identités et des images [7].

2.2 Architecture d'OpenStack

L'architecture d'OpenStack se compose de multiples composantes dont chacune assure un service d'OpenStack [8].

La figure 2.4 montre les services essentiels qui assurent la puissance de calcul, la mise en réseau, le stockage, la gestion des identités et la gestion des images.

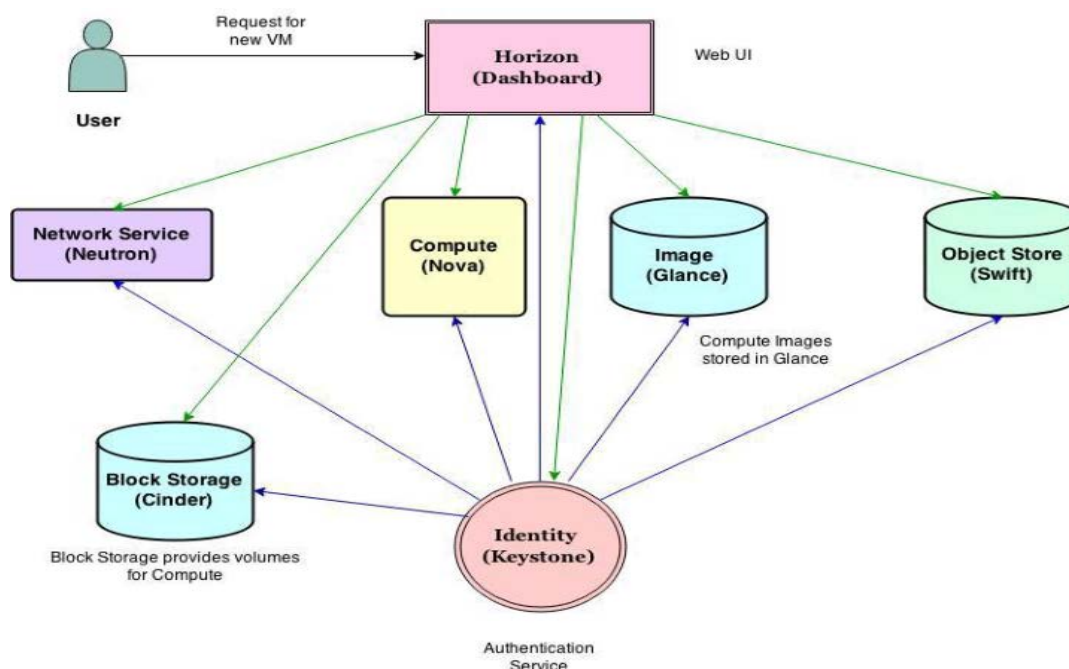


Figure 2.3 : Architecture d'OpenStack

2.2.1 Keystone (Identité)

Keystone permet une gestion de l'identité et des autorisations d'accès pour les différents services d'OpenStack grâce à des jetons temporaires générés par lui-même lors de l'authentification [9].

Il se compose de quatre entités :

- Un domaine qui représente un ensemble de projets, de groupes et d'utilisateurs qui définissent les limites administratives pour la gestion des entités OpenStack.
- Un rôle qui comprend un ensemble de droits et privilèges. Un utilisateur assumant ce rôle hérite de ces droits et privilèges.

- Un utilisateur qui peut être associé à des rôles, des projets ou les deux [9].
- Un projet (tenant) qui est l'unité de base d'OpenStack, car toutes les ressources IT doivent appartenir à un projet spécifique. En effet, chaque projet appartient à un domaine spécifique.

2.2.2 Nova (Compute)

Nova fournit un moyen de provisionner des instances. Il prend en charge la création de machines virtuelles qui seront exécutées sur les différents nœuds de calcul [9].

Nova se compose principalement de :

- Nova API qui reçoit et traite des requêtes concernant le composant.
- Planificateur (scheduler) qui permet de répartir les ressources à allouer à travers la plateforme IaaS, en fonction des ressources disponibles.
- Chef d'orchestre (conductor) qui contient les infos sur la VM créée.
- Nova compute qui crée et met fin aux VM.

2.2.3 Glance

Glance permet de découvrir, de sauvegarder et de récupérer des images de disques durs utilisées par les machines virtuelles [9].

Glance se compose de :

- Registre glance qui est un démon serveur qui sert des métadonnées d'images via une API du type REST.
- Glance API qui reçoit et traite des requêtes concernant le composant.

2.2.4 Quantum (Neutron)

Quantum fournit un réseau en tant que service. Il peut créer des réseaux, attribuer des adresses IP aux machines virtuelles de ces réseaux et gérer les règles des pare-feu.

Neutron se compose de :

- Quantum server qui reçoit et traite des requêtes concernant le composant.
- Quantum DHCP client qui permet d'offrir le service DHCP aux instances virtuelles de nova.
- Quantum plugin agent qui est un plugin utilisé par neutron pour créer le réseau défini.
- Quantum L3 agent qui réalise le routage, le NAT et le forwarding réseau de niveau 3 [9].

2.2.5 Cinder

Cinder permet aux utilisateurs de créer des volumes, des disques durs externes virtuels, qui peuvent être attachés à une instance de machines virtuelle.

Il se compose de :

- Cinder API qui reçoit et traite des requêtes concernant le composant.
- Cinder planificateur qui permet de répartir les volumes à allouer à travers la plateforme en fonction des ressources disponibles.
- Cinder volume qui permet de gérer les volumes.
- Cinder de secours qui crée et restaure des sauvegardes de disques [9].

2.2.6 Swift

Dans les très grands environnements de stockage, où la quantité totale de données se compte par centaines de téraoctets, il est recommandé d'utiliser le mode de stockage d'objet assuré par swift.

En mode objet, contrairement au mode fichier, il n'y a pas de concept d'arborescence. En effet, chaque objet est accordé à des métadonnées qui permettent de le retrouver.

De plus, cinder ou glance peuvent utiliser l'espace de stockage fourni par swift [9].

2.2.7 Horizon

C'est une application web permettant aux utilisateurs et aux administrateurs de gérer les composants regroupés dans OpenStack via une interface graphique.

2.3 Installateurs d'OpenStack

Les méthodes d'installation d'OpenStack relèvent généralement de deux grandes catégories : le test d'un côté et le passage en production de l'autre.

2.3.1 Devstack

Devstack est un projet d'installation d'OpenStack à base de scripts bash maintenu par une communauté de développeurs. Il permet la construction des environnements complets destinés au développement et aux tests. Devstack peut être installé en machine virtuelle ainsi que sur une machine physique. Il supporte la distribution Ubuntu en versions courantes [9].

2.3.2 Packstack

Packstack utilise les modules puppet pour déployer OpenStack. Il définit l'environnement avec la minimale configuration qui est préférable pour une activité pratique pas pour un environnement de production.

L'OpenStack déployé à partir de Packstack n'est pas assez stable pour gérer les exigences du centre de données. Packstack soutient Centos et Red Hat Enterprise Linux (RHEL).

2.3.3 Fuel

Ce sont des modules testés par Mirantis. Fuel se base sur le nœud Fuel-Master qui se trouve au-dessus de tous les nœuds, y compris le contrôleur, et il permet de surveiller leur fonctionnement. Il est directement connecté au contrôleur et aux nœuds via un réseau séparé PXE (réseau admin).

Fuel fournit une interface d'utilisateur graphique pour faciliter le contrôle et la gestion des nœuds. En outre, il supporte Centos et Ubuntu et il peut être étendu pour soutenir d'autres distributions.

Le tableau 2.1 résume les caractéristiques des installateurs d'OpenStack : Devstack, Packstack et Fuel.

Tableau 2.1: Les installateurs d'OpenStack

Méthodes d'installation Caractéristiques	Devstack	Packstack	Fuel
Déploiement	Développement	Développement	Production
Système d'exploitation supporté	Ubuntu	Fedora, CentOS et RHEL	Ubuntu et CentOS
open source	Oui	Oui	Non
La stabilité de déploiement	Non	Oui	Oui
Le niveau de difficulté	Absence	Normal	Présence

En se basant sur le tableau 2.1, nous avons choisi d'étudier les deux méthodes d'installation d'OpenStack Devstack et Packstack pour les raisons suivantes :

- Devstack et Packstack sont open source.
- Facile à manipuler.
- Ce sont des plateformes destinées aux tests.

2.4 Architectures de déploiement d'OpenStack

Il existe de nombreuses architectures pour le déploiement d'OpenStack :

- Nœud unique : il s'agit d'un seul serveur qui exécute tous les services d'OpenStack et conduit également toutes les instances virtuelles.
- Deux nœuds : un nœud de contrôleur nuage exécutant les services nova, et un nœud de calcul gérant nova-compute.
- Plusieurs nœuds : quatre nœuds au minimum, c'est la meilleure architecture pour exécuter plusieurs instances virtuelles qui nécessitent beaucoup de puissance de traitement. Nous pouvons ajouter un nœud de contrôle, de compute, de stockage et de réseau [9].

Notre choix s'est fixé pour l'architecture de nœud unique. Nous avons utilisé cette configuration afin de tester les fonctionnalités offertes par OpenStack. En outre, Devstack et Packstack ne supportent que l'architecture all-in-one.

3. Ansible

3.1 Présentation

Ansible est un moteur d'automatisation informatique Open Source qui automatise l'approvisionnement, la gestion des configurations, le déploiement des applications, l'orchestration et bien d'autres processus informatiques [10].

Ansible repose sur le protocole d'authentification réseau SSH et l'architecture Push pour pousser les configurations décrites dans les playbooks vers les clients.

3.2 Architecture d'Ansible

L'architecture d'Ansible se compose :

- Des hôtes (des nodes) auxquelles Ansible se connecte et pousse les tâches d'automatisation via le protocole SSH.
- Du nœud de contrôle sur lequel l'outil Ansible est installé.
- De l'inventaire qui présente une liste de toutes les adresses IP des hôtes.
- Des playbooks qui définissent les tâches à accomplir sur les ressources que nous souhaitons les automatiser. Chaque tâche a un nom et appelle des modules.
- Des plugins qui sont des types spéciaux de modules. Nous les exécutons avant l'exécution des modules sur les hôtes comme le montre la figure 2.5.

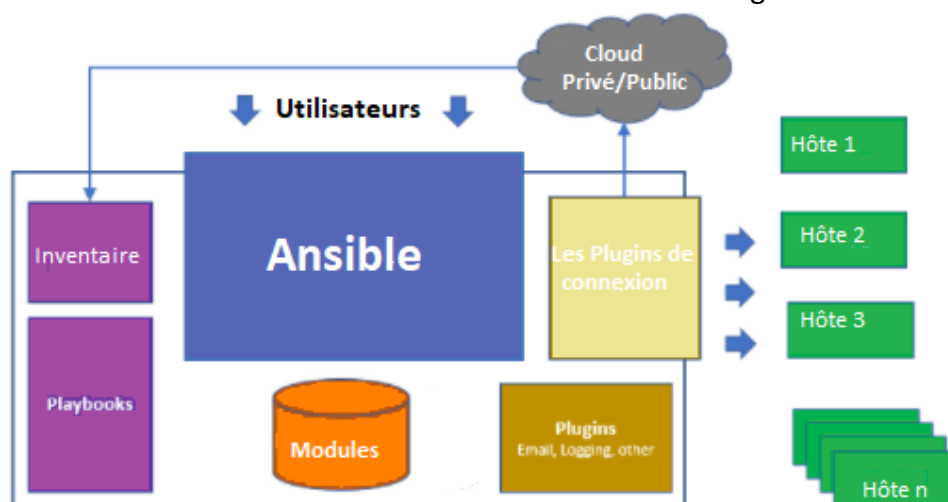


Figure 2.4 : L'architecture d'Ansible

Toutefois, Ansible présente quelques lacunes. Il ne permet pas ni la délégation de l'exécution de playbook à des équipes tierces ni la définition des extras variables. De plus, il ne crée pas des pipelines de playbooks Ansible permettant d'éviter le gaspillage du temps.

Pour remédier à ces faiblesses, nous avons utilisé l'outil d'orchestration des playbooks Ansible : AWX.

4. AWX

AWX est la version open source derrière Red Hat Ansible Tower. Il est conçu pour fonctionner au-dessus du projet Ansible, améliorant ainsi le moteur d'automatisation.

AWX ajoute une interface utilisateur web, la planification des travaux, la gestion des stocks, la création de rapports, l'automatisation du flux de travail, le partage des informations d'identification et des outils pour activer la délégation [11].

AWX dispose d'un système de workflow multi-playbooks permettant de réaliser des pipelines de playbooks Ansible et de paralléliser leur exécution.

AWX permet également aux utilisateurs de fournir des variables supplémentaires d'une manière conviviale, sans avoir besoin de les transmettre à partir de la boîte « EXTRA VARIABLES ».

Conclusion

Au cours de ce chapitre, nous avons présenté la plateforme d'orchestration des ressources IT OpenStack et les outils d'automatisation Ansible et AWX pour la mise en place de la solution envisagée. Les contraintes de ces outils vont influencer la conception qui sera étudiée dans le chapitre suivant. .

Chapitre 3 : Analyse et spécification des besoins

Introduction

La phase de conception vise à déterminer les objectifs et les caractéristiques techniques et architecturales de la solution envisagée. En effet, nous allons spécifier et élaborer les différents cas d'utilisation qui vont annoncer les exigences fonctionnelles et non fonctionnelles.

1. Besoins fonctionnels

L'ensemble de fonctionnalités d'OpenStack doit répondre aux exigences fonctionnelles suivantes :

1.1 Gestion des projets

OpenStack à travers keystone permet la gestion des projets. L'administrateur d'OpenStack peut alors créer, supprimer, activer, désactiver et modifier les caractéristiques des projets.

1.2 Gestion des ressources

Les utilisateurs d'OpenStack gèrent l'ensemble des ressources disponibles sur le Cloud qui peuvent être des services, des plateformes de développement ou des infrastructures informatiques. Ces ressources sont définies selon les besoins des clients et affectées aux projets.

1.3 Exécution des tâches

Les administrateurs ou les utilisateurs disposant des autorisations d'administrateur peuvent exécuter les tâches suivantes :

- Gérer les instances via le tableau de bord d'OpenStack.
- Gérer l'accès aux environnements.
- Gérer les images.
- Vérifier l'état des ressources.
- Gérer les gabarits.

1.4 Gestion des paiements

L'administrateur d'OpenStack gère les paiements selon la quantité des services consommée en Cloud Computing. En effet, il envoie des lettres de relances à leurs clients pour leur rappeler la date de facturation.

1.5 Gestion des utilisateurs

L'administrateur d'OpenStack ou de projet a le droit de consulter la liste d'utilisateurs enregistrés, de les ajouter ou éditer leurs métadonnées.

Chaque utilisateur appartient à un seul projet où il dispose des droits lui permettant d'exploiter les ressources d'une manière sécurisée.

2. Besoins non fonctionnels

Il s'agit des besoins qui caractérisent l'infrastructure Cloud :

- Allocation de ressources immédiate sur demande : un utilisateur peut avoir à sa disposition les ressources informatiques dont il a besoin (temps de calcul, de serveurs, capacité de stockage, etc.) sans intervention humaine.
- Flexibilité : les ressources disponibles ont généralement une capacité forte et rapide à s'adapter à l'évolution des besoins d'une manière transparente pour les utilisateurs.
- Supervision centralisée de l'infrastructure.
- Paiement à l'usage : facturation des utilisateurs en fonction de métriques précises.
- Accès léger : aucun matériel ou logiciel propriétaire n'est requis pour accéder aux ressources via une application disponible sur internet.
- Amélioration de la qualité de service grâce au processus d'automatisation de l'infrastructure Cloud.

3. Identification des acteurs

Un acteur est une personne physique ou morale qui représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système.

Pour notre projet, nous avons identifié deux acteurs :

- L'administrateur : Il dispose de droits administratifs, ce qui lui permet de contrôler l'ensemble du Cloud et lui permet d'accéder au portail « tableau de bord » et à l'interface de gestion sur tous les autres projets.
- Le membre du projet : C'est un membre d'un ou de plusieurs projets. Il ne pourra accéder qu'à ses projets.

4. Conception

Pour faciliter la conception des documents nécessaires au développement d'OpenStack, nous avons utilisé le langage UML.

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage commun de modélisation visuelle, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que par leur comportement [12].

4.1 Diagramme de cas d'utilisation générale

Le diagramme des cas d'utilisation permet de modéliser le comportement du système sous forme d'actions et de réactions du point de vue de l'utilisateur. Il permet de définir les limites du système et la relation entre le système et l'environnement.

Le diagramme 3.1 représente tous les cas d'utilisation relatifs à notre projet.

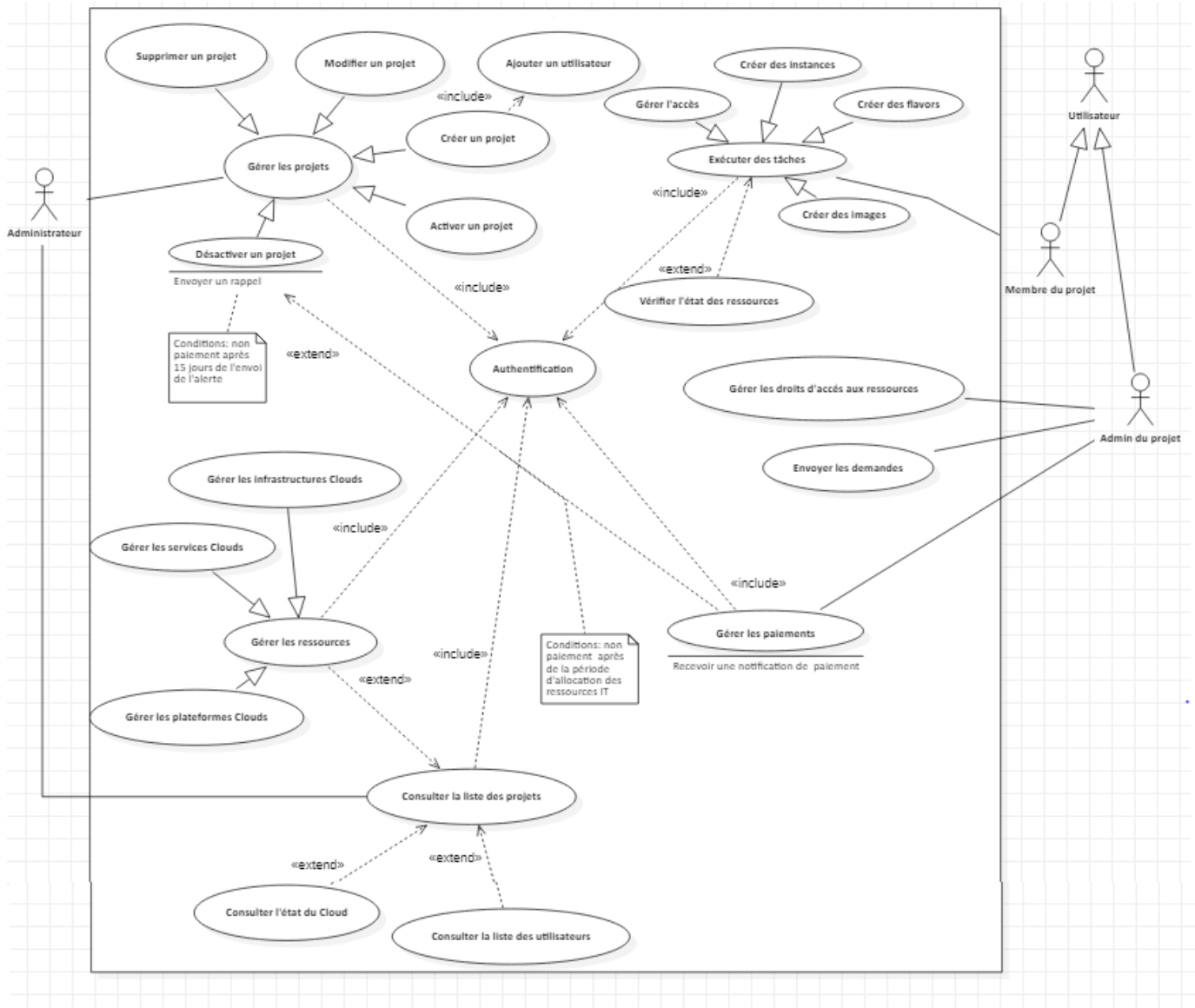


Figure 3.1 : Diagramme de cas d'utilisation générale

D'après la figure ci-dessus, l'administrateur dispose de tous les privilèges permettant la gestion du Cloud.

4.2 Raffinements des Cas d'Utilisations

4.2.1 Cas d'utilisation « Consulter l'état du Cloud »

L'administrateur peut :

- Consulter l'utilisation des serveurs par projet, la consommation de CPU virtuels, le RAM et l'espace disque.
- Générer des rapports au format CSV.

La figure 3.2 présente le cas d'utilisation « Consulter l'état du Cloud ».

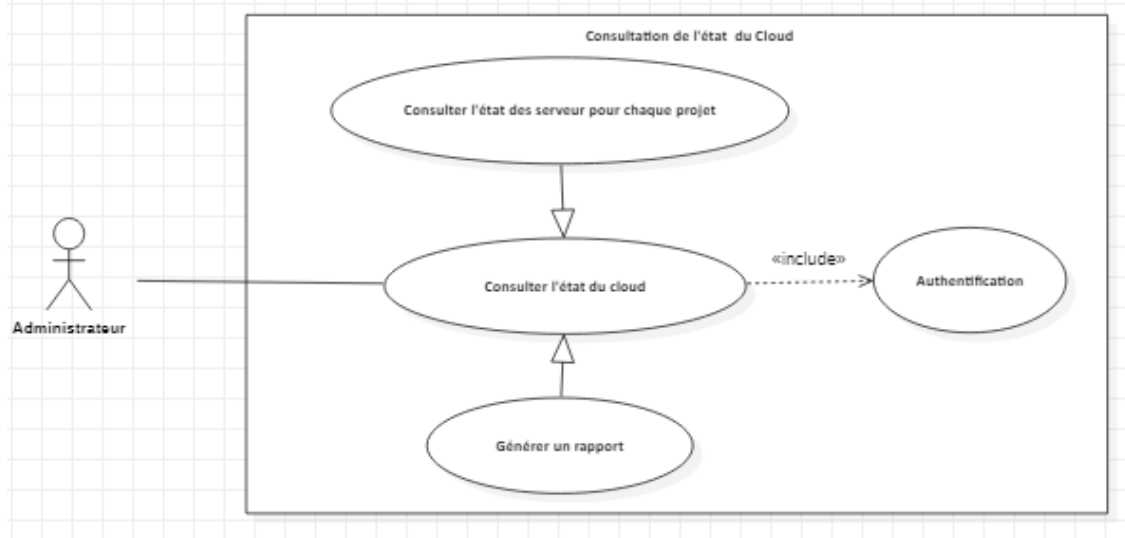


Figure 3.2 : Cas d'utilisation « Consulter l'état du Cloud »

4.2.2 Cas d'utilisation « Gérer les instances »

L'administrateur peut gérer les instances existantes sur le Cloud. En effet, il a la possibilité de :

- Créer une nouvelle instance.
- Éditer les différents détails d'une instance.
- Mettre fin à une instance.
- Exécuter, redémarrer ou suspendre une instance.
- Accéder à l'instance via sa console.
- Consulter la liste des instances du Cloud.

La figure 3.3 montre le cas d'utilisation « Gérer les instances »

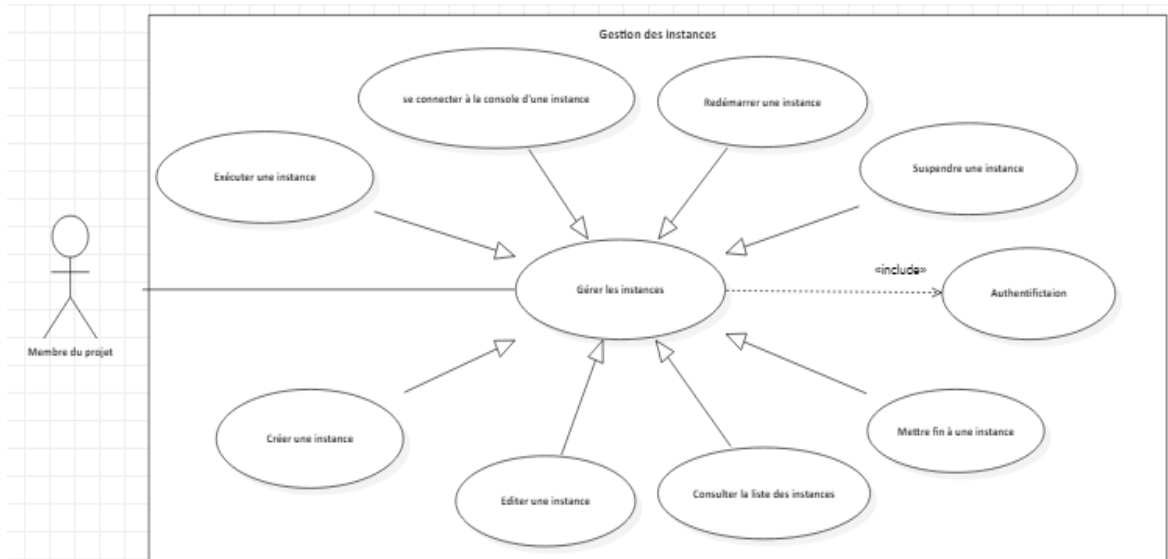


Figure 3.3: Cas d'utilisation « Gérer les instances »

4.2.3 Cas d'utilisation « Gérer les services Clouds »

L'administrateur d'OpenStack peut :

- Consulter la liste des services (nova, keystone, glance, cinder et neutron) qui sont activés.
- Modifier l'état du service c'est-à-dire activer ou désactiver un service.
- Gérer le statut de serveurs hôtes (activé, désactivé).

La figure 3.4 montre le cas d'utilisation « Gérer les services Clouds »

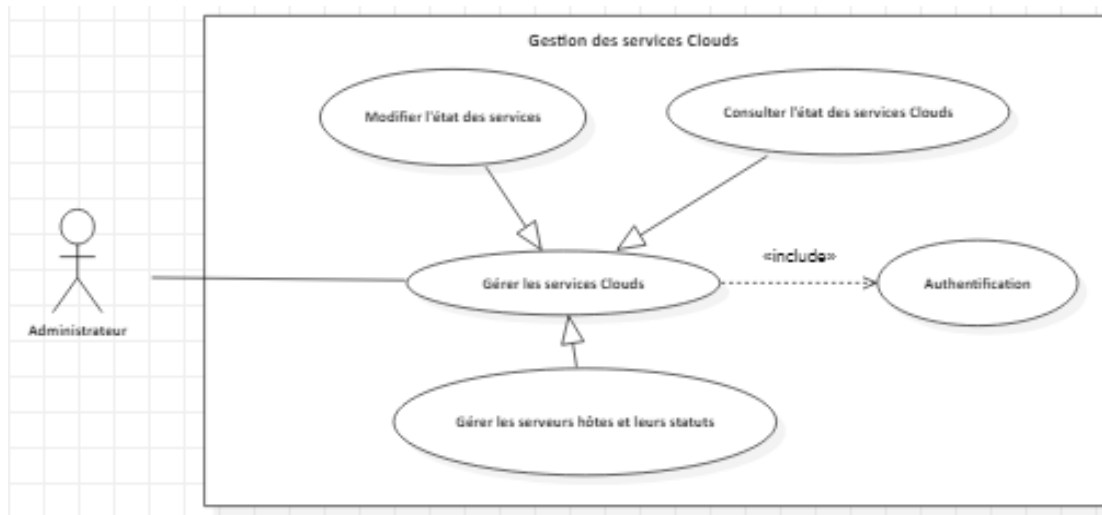


Figure 3.4 : Cas d'utilisation « Gérer les services Clouds »

4.2.4 Cas d'utilisation « Gérer les gabarits »

Un membre de projet peut :

- Consulter la liste des gabarits actuellement disponibles qui peuvent être utilisés pour démarrer l'instance.
- Créer, éditer ou supprimer des gabarits personnalisés.

La figure 3.5 présente le cas d'utilisation « Gérer les gabarits »

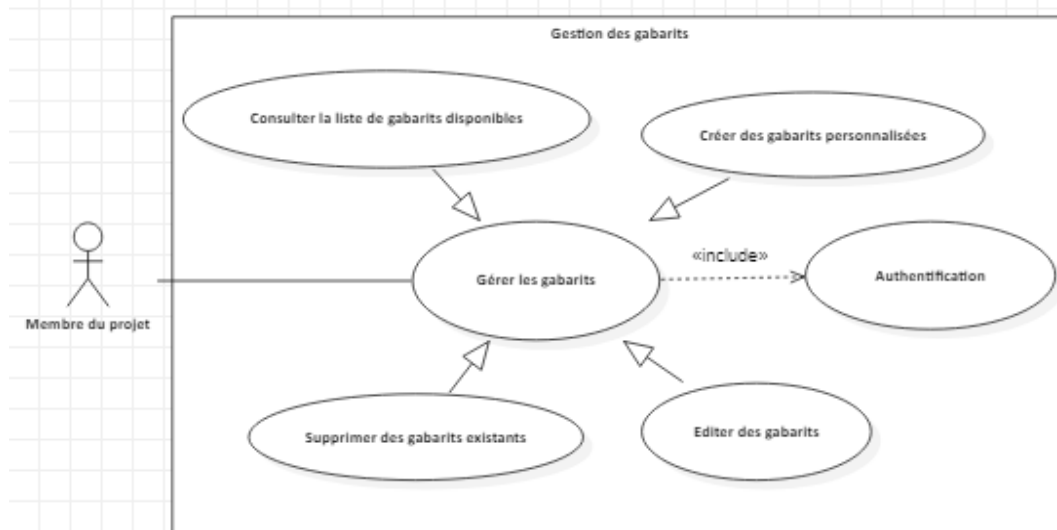


Figure 3.5 : Cas d'utilisation « Gérer les gabarits »

4.2.5 Cas d'utilisation « Gérer les images »

Un membre de projet pourra :

- Voir la liste des images disponibles.
- Créer et supprimer des images si elles ne sont plus nécessaires.
- Modifier les caractéristiques de l'image (nom, ID noyau, ID ramdisk, architecture, format, public ou privé).

La figure 3.6 présente le cas d'utilisation « Gérer les images ».

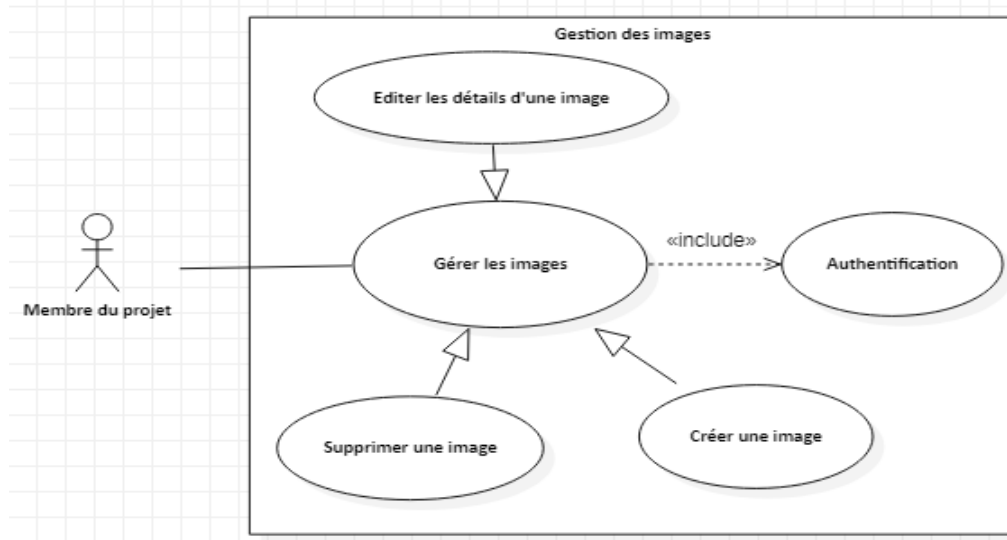


Figure 3.6 : Cas d'utilisation « Gérer les images »

4.2.6 Cas d'utilisation « Consulter la liste des utilisateurs »

L'administrateur du tenant gère les utilisateurs des projets, il est autorisé à :

- Affecter un utilisateur à un projet et le détacher.
- Ajouter un nouvel utilisateur à un projet.
- Contrôler les privilèges des utilisateurs.

La figure 3.7 présente le cas d'utilisation « Gérer les utilisateurs des projets ».

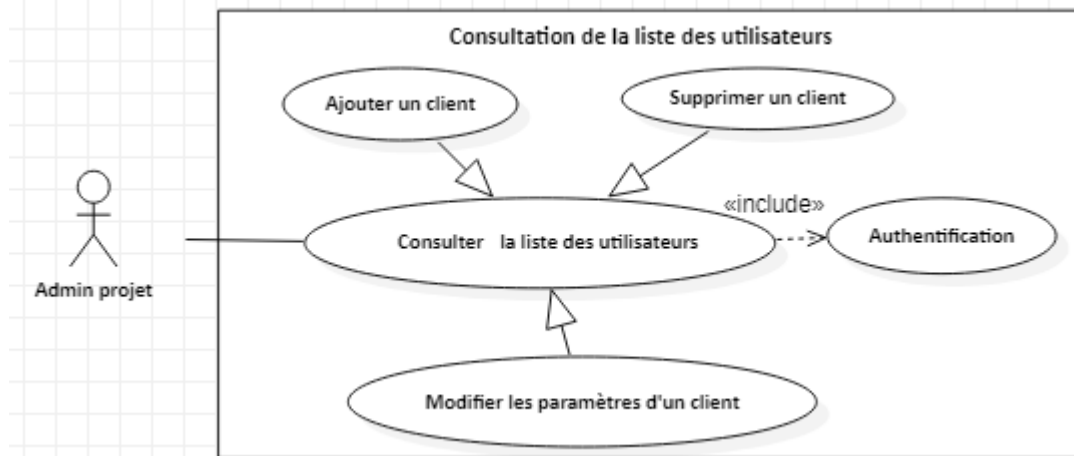


Figure 3.7 : Cas d'utilisation « Consulter la liste des utilisateurs »

4.3 Diagramme de séquence

A moins que le système à modéliser soit extrêmement simple, nous ne pouvons pas modéliser la dynamique globale du système dans un seul diagramme. Nous ferons donc appel à un ensemble de diagrammes de séquences chacun correspondant à une sous-fonction du système, généralement d'ailleurs pour illustrer un cas d'utilisation [13].

Avec les diagrammes de séquences système, l'UML fournit un moyen graphique pour représenter les interactions entre un acteur et le système au cours de l'exécution du cas d'utilisation.

Le diagramme de séquence fait partie des diagrammes comportementaux (dynamique) et plus précisément des diagrammes d'interactions.

Dans ce contexte, nous allons présenter quelques diagrammes de séquences les plus significatifs.

4.3.1 Diagramme de séquence « Authentification ESXI »

L'hyperviseur ESXI est sécurisé par défaut. Pour renforcer la protection des hôtes ESXI, NextStep IT exige l'utilisation d'un login et un mot de passe pour un accès à partir de l'interface DCUI (Direct Console User Interface).

La figure 3.8 présente la phase d'authentification sur l'hyperviseur.

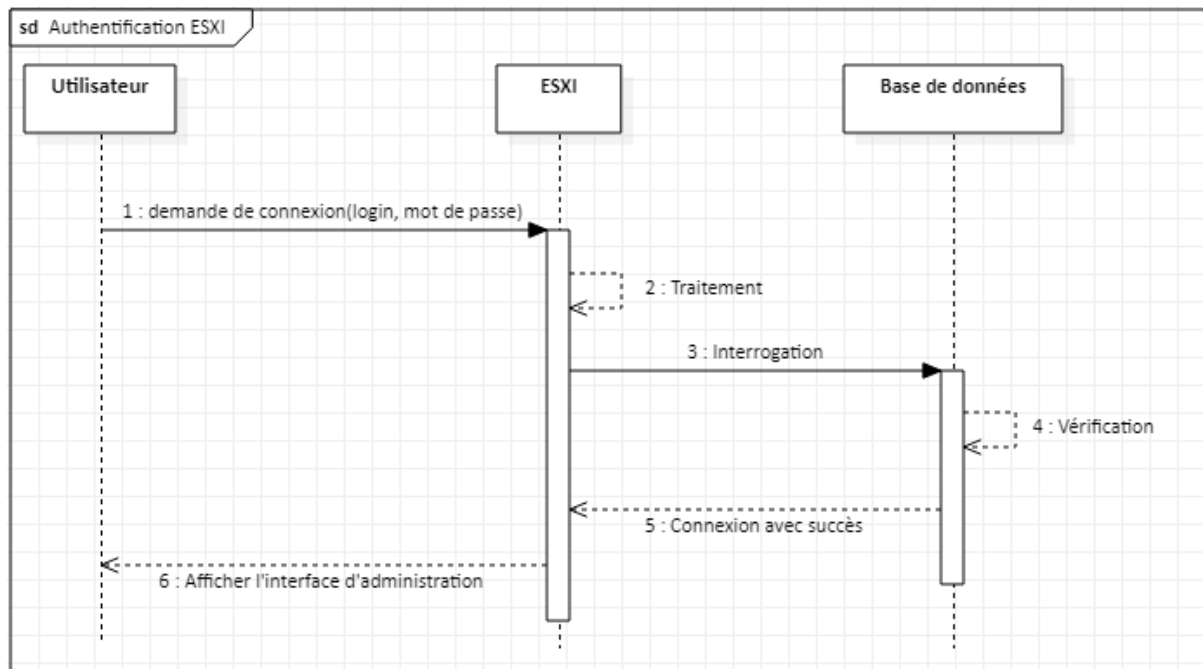


Figure 3.8 : Diagramme de séquence de l'authentification sur l'hyperviseur ESXI

La phase d'authentification se déroule comme suit :

- L'utilisateur accède à l'interface DCUI et il entre les paramètres d'accès qui sont le login et le mot de passe.

- Après la vérification et la validation des données saisies, l'interface graphique de ESXI sera affichée.

4.3.2 Diagramme de séquence « Création d'une VM »

Le processus de la création d'une machine virtuelle sur l'hyperviseur VMware ESXI est représenté dans la figure 3.9.

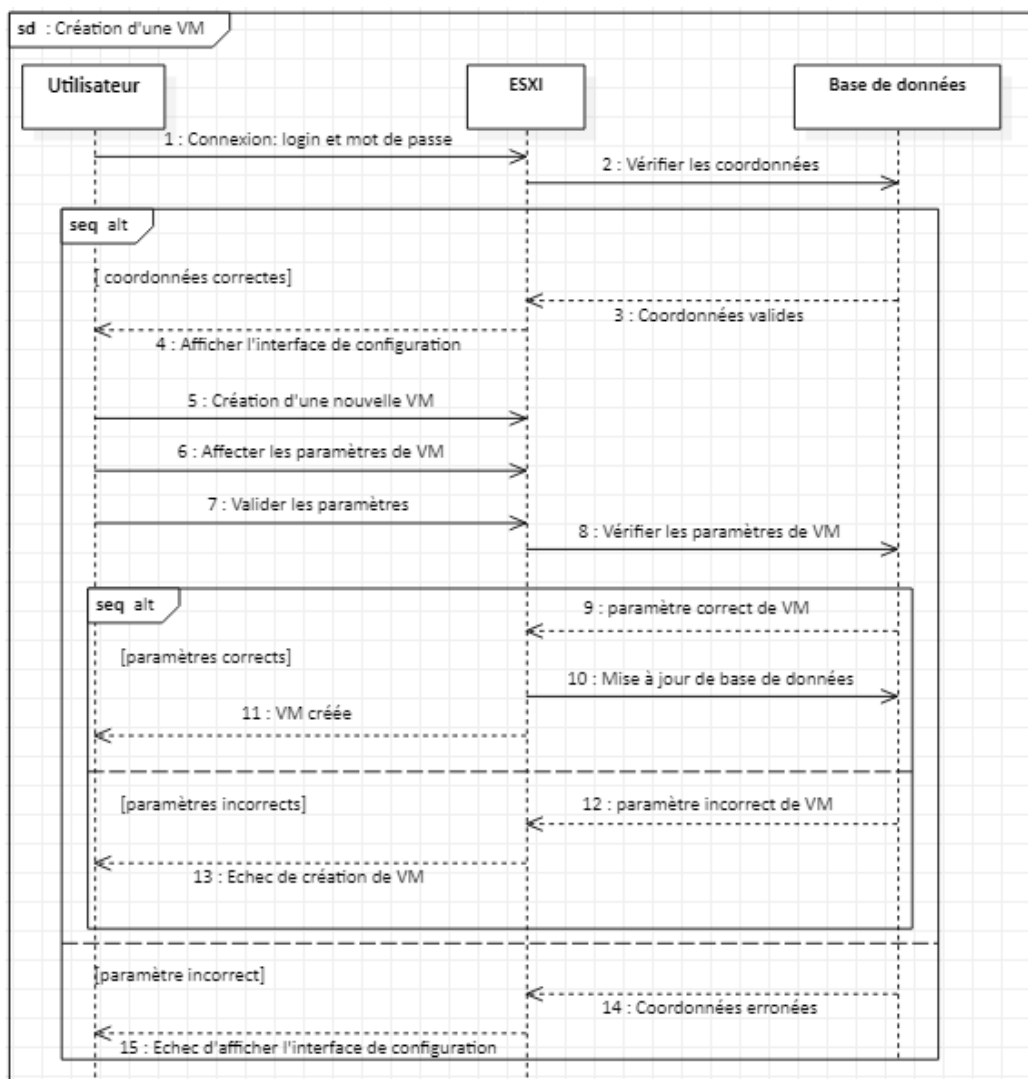


Figure 3.9 : Diagramme de séquence de la création d'une VM

L'acteur de ce scénario est l'utilisateur.

Suite à la phase d'authentification, l'utilisateur souhaite créer une machine virtuelle. Il doit définir les paramètres de la VM qui seront traités et ajoutés à la base de données de l'entreprise en cas de validation des paramètres. Par la suite, la machine sera créée, sinon un message d'erreur sera affiché.

4.3.3 Diagramme de séquence « Authentification OpenStack »

Pour exploiter les différents services du Cloud, l'utilisateur (administrateur, membre d'un projet) doit s'identifier. Ainsi après la saisie de ses informations (nom d'utilisateur , mot de

passé) dans l'interface d'authentification d'horizon, le service d'identité de keystone vérifie les données d'utilisateur. En cas de validation, keystone lui accorde un jeton. Selon ce dernier, une liste des projets s'affichera, comme le montre la figure 3.10.

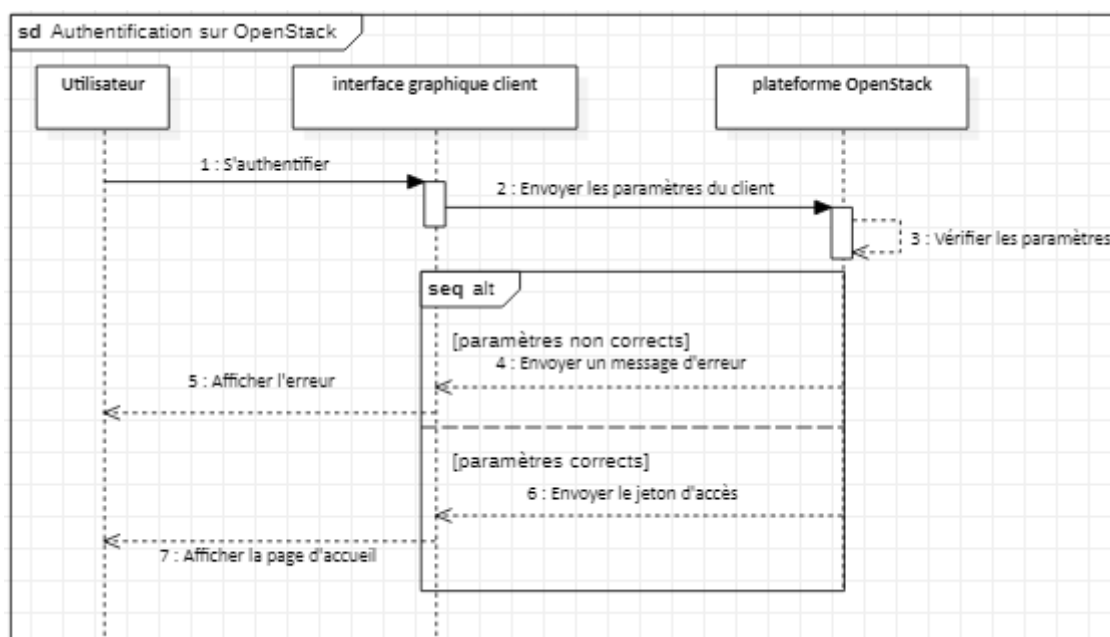


Figure 3.10 : Diagramme de séquence de l'authentification OpenStack

4.3.4 Diagramme de séquence « Ajout d'une règle de filtrage IP »

La figure 3.11 représente le processus de la phase d'ajout d'une règle de filtrage IP.

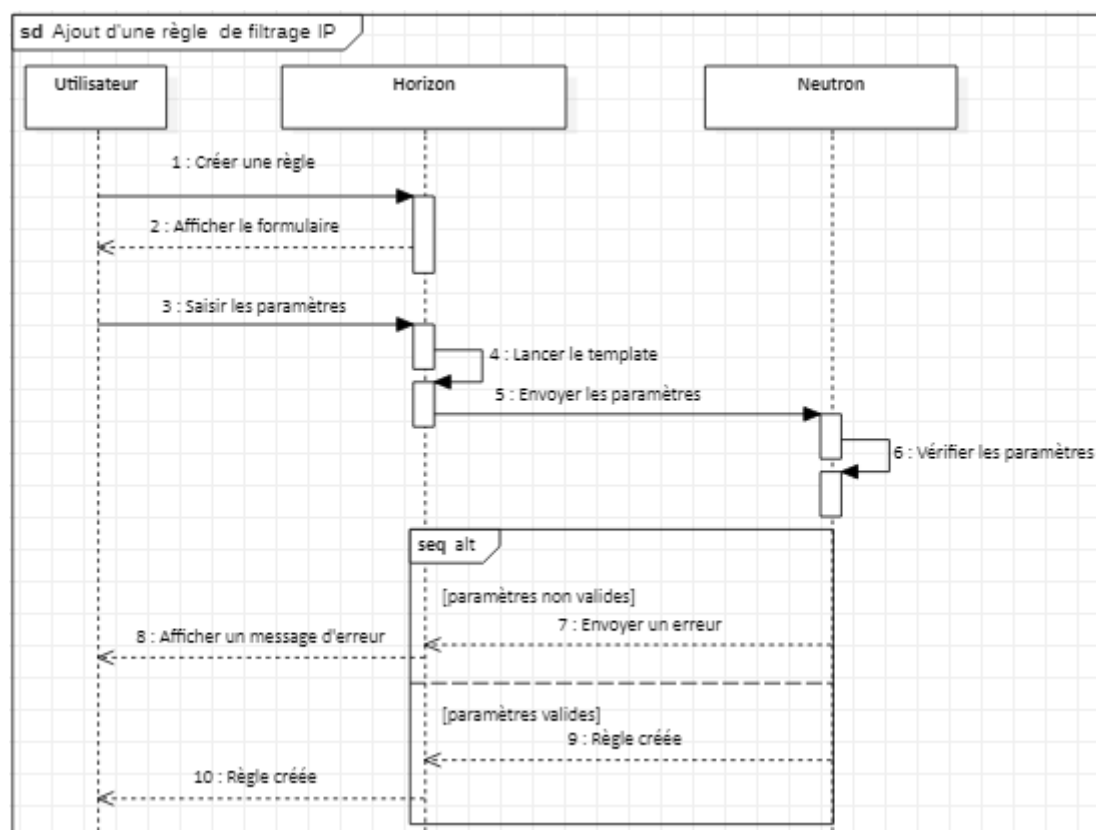


Figure 3.11 : Diagramme de séquence d'ajout d'une règle de filtrage IP.

Le service neutron d'OpenStack permet de définir l'architecture réseau qui fournit la connectivité entre tous les composants. Il gère les règles de filtrage IP ajoutées par l'utilisateur via horizon.

4.3.5 Diagramme de séquence « Lancement d'une instance »

Dans ce qui suit nous admettons que les quotas sont définis.

La création d'une instance nécessite la communication d'horizon avec keystone en premier lieu et keystone avec glance et nova en second lieu. Après la création de cette instance l'utilisateur peut lui accéder via SSH ou VNC comme le montre la figure 3.12.

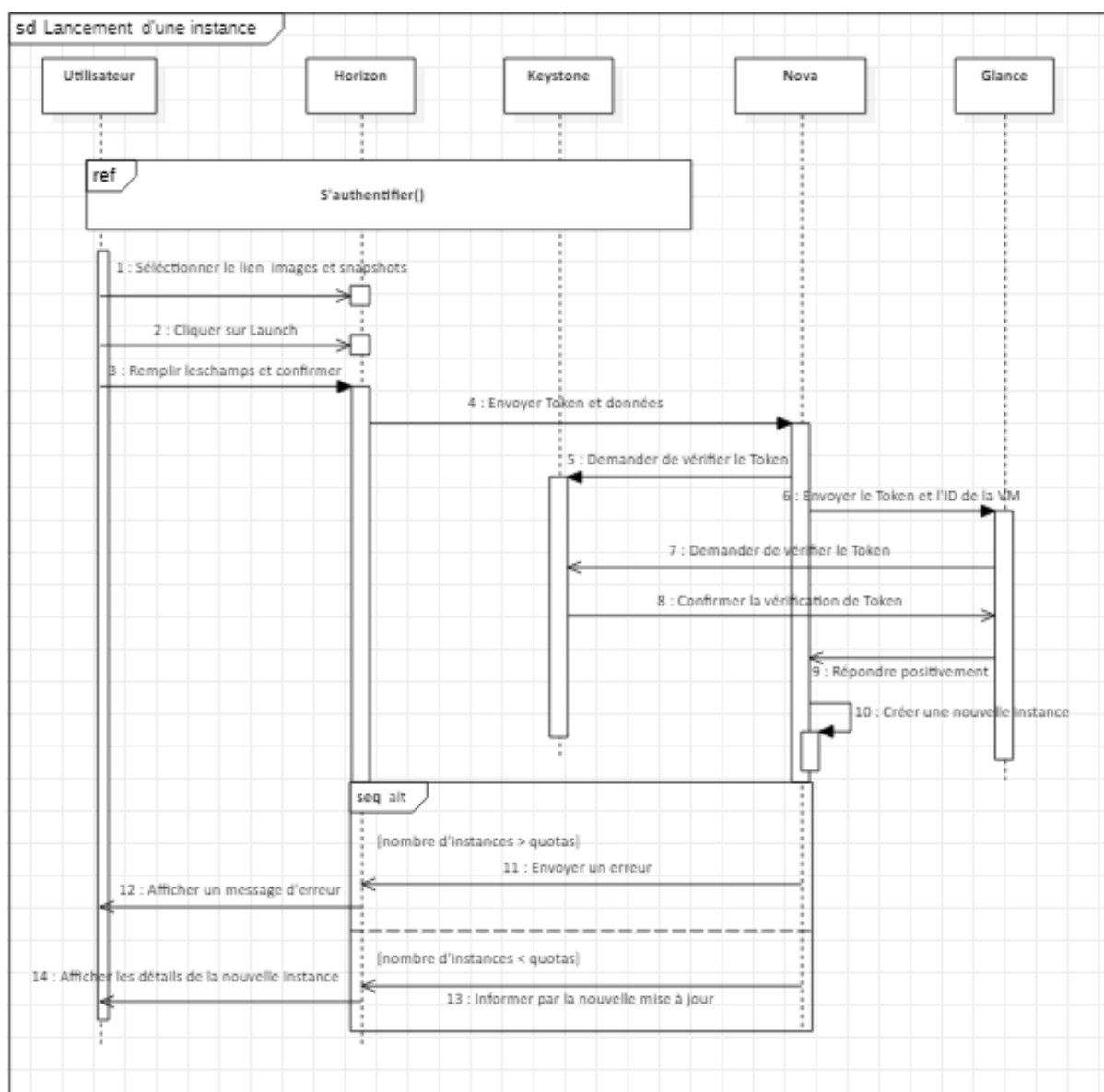


Figure 3.12 : Diagramme de séquence du lancement d'une instance

4.3.6 Diagramme de séquence « Automatisation d'OpenStack à de l'outil Ansible »

Le processus d'automatisation du Cloud consiste à utiliser des playbooks afin de réaliser les tâches répétitives d'OpenStack, avec une intervention humaine très réduite, pour unifier les processus de gestion du Cloud.

La figure 3.13 décrit le processus d'automatisation à base de l'outil de l'Ansible.

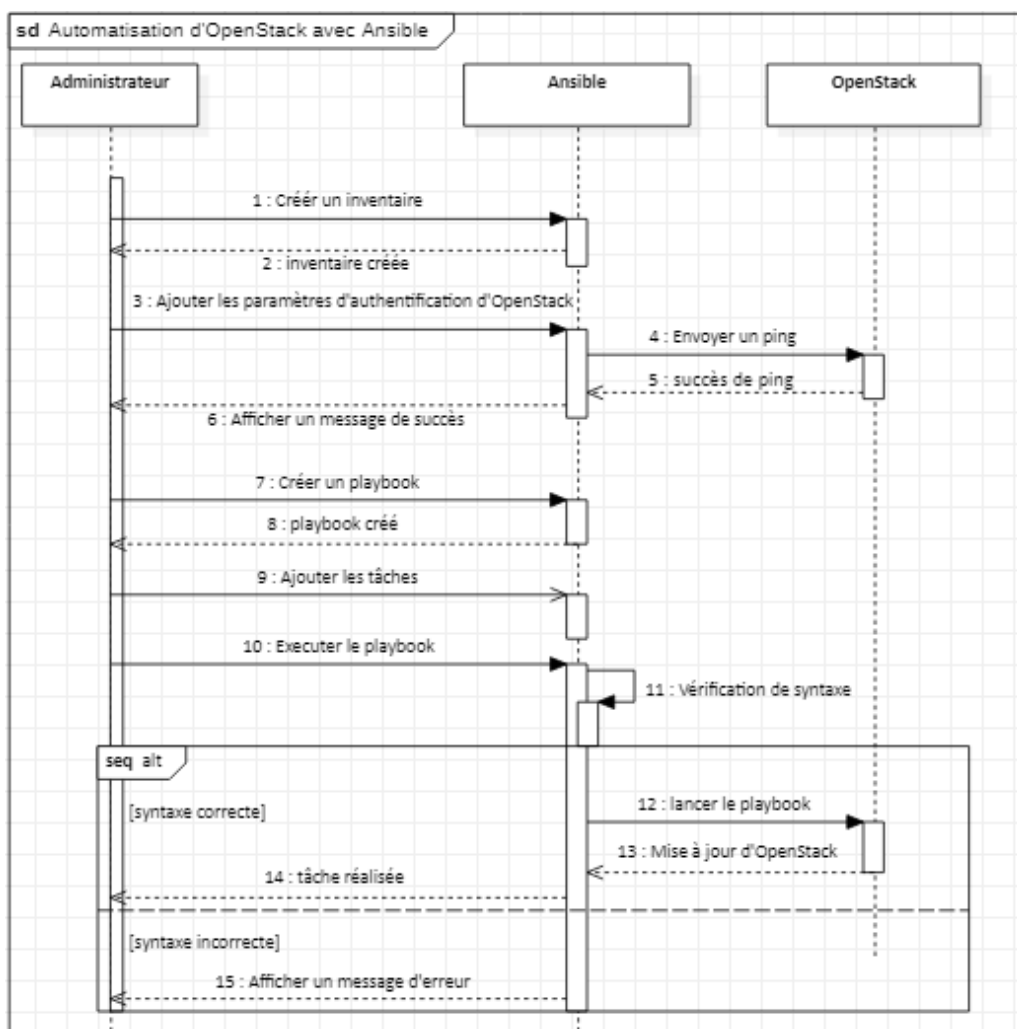


Figure 3.13 : Diagramme de séquence de l'automatisation d'OpenStack

L'administrateur doit définir les paramètres d'accès au tableau de bord d'OpenStack dans un inventaire. Par la suite, il a créé un playbooks où il a défini les tâches sur la machine ciblée. Ensuite, il va vérifier sa syntaxe, en cas de succès, il sera lancé sur l'infrastructure OpenStack, sinon il sera annulé et rejeté.

Conclusion

Ce chapitre est une phase nécessaire pour le déroulement du projet. Nous avons dégagé les différents besoins fonctionnels et non fonctionnels de la solution envisagée. Ensuite, nous avons détaillé les cas d'utilisations relatives aux tâches de l'infrastructure Cloud.

Le chapitre suivant sera consacré à la description détaillée de la phase de la réalisation.

Chapitre 4 : Réalisation

Introduction

Après avoir achevé la phase de l'analyse et de la spécification des besoins, nous allons enchaîner avec la mise en place d'une infrastructure Cloud basée sur OpenStack. Ensuite, nous allons automatiser les tâches répétitives d'OpenStack.

1. Mise en place de l'IaaS : OpenStack

NextStep IT a mis à notre disposition un hyperviseur type 1 appelé VMware ESXi 7.0 où nous avons créé nos machines virtuelles pour des raisons de souplesse et de fiabilité (voir annexe 1).

1.1 Installation et configuration d'OpenStack

Le déploiement d'OpenStack est décrit dans l'annexe 2. Nous avons choisi d'étudier les deux installateurs d'OpenStack : Devstack et Packstack. Ces derniers sont déployés sur un seul nœud qui permet l'exécution de tous les services d'OpenStack et également la conduite des instances virtuelles.

1.3 Manipulations des fonctionnalités d'OpenStack

Lors de cette phase, nous allons créer un environnement Cloud qui répond aux besoins du client « ABF » : 10 Go de RAM ,6 VCPUs, 50 Go d'espace de stockage et une instance virtuelle. Le schéma 4.1 présente les étapes que nous allons suivre pour répondre aux besoins du client « ABF ».

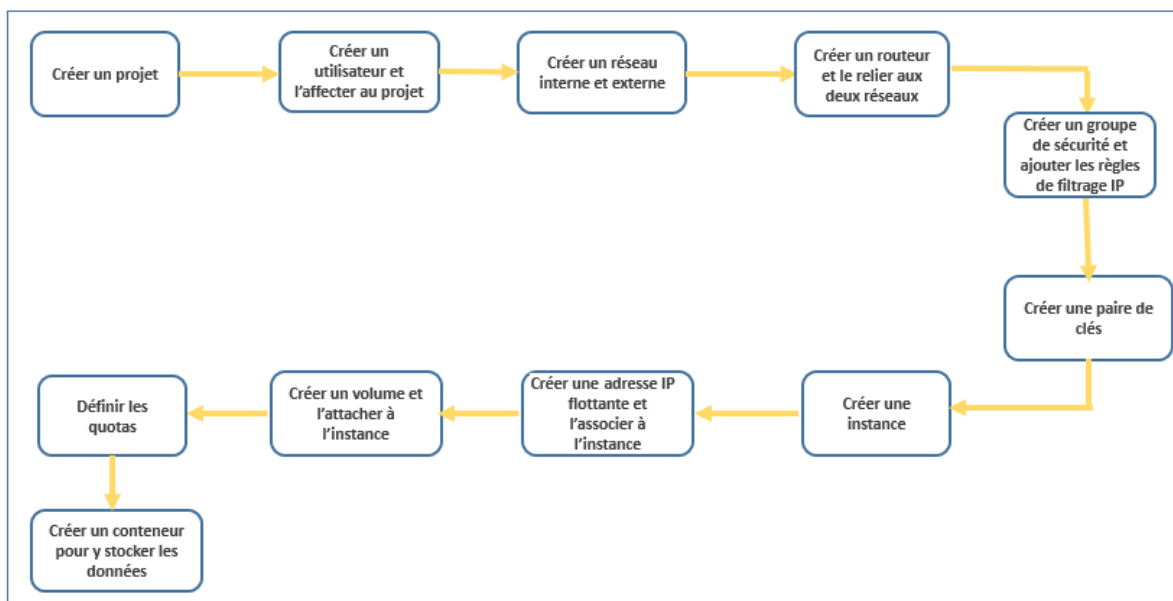


Figure 4.1 : Les étapes de la création d'un environnement Cloud sur OpenStack

1.2.1 Création du projet

L'administrateur d'OpenStack peut créer des projets pour les nouveaux utilisateurs via tableau de bord d'OpenStack. Les projets possèdent des pools de ressources virtuelles définies selon le besoin des utilisateurs.

Nous avons créé un projet nommé projet ABF comme l'indique l'image 4.2.

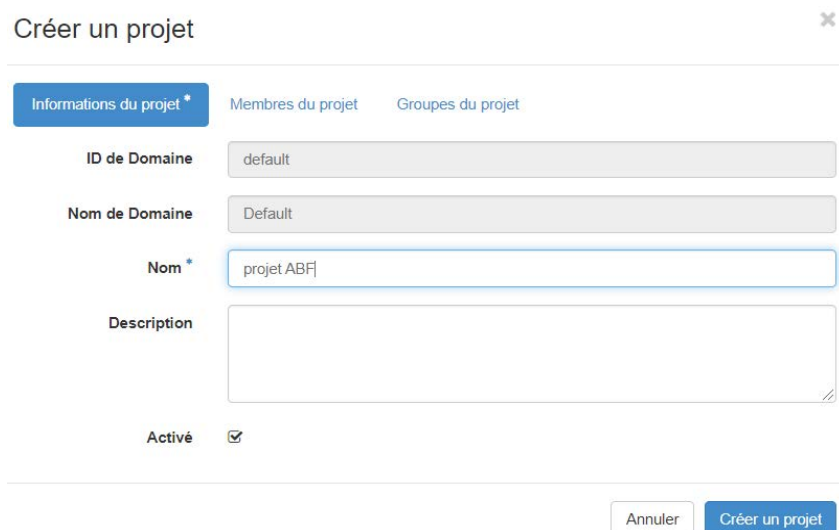


Figure 4.2 : Création du projet

Par défaut, le projet ABF sera créé dans le domaine « Default ».

1.2.2 Création de l'utilisateur « ABF »

Il existe deux types d'utilisateurs du Cloud : administrateur ou membre d'un projet. « ABF » est un membre du tenant « ABF ». La figure 4.3 représente les champs à remplir pour créer l'utilisateur « ABF ».

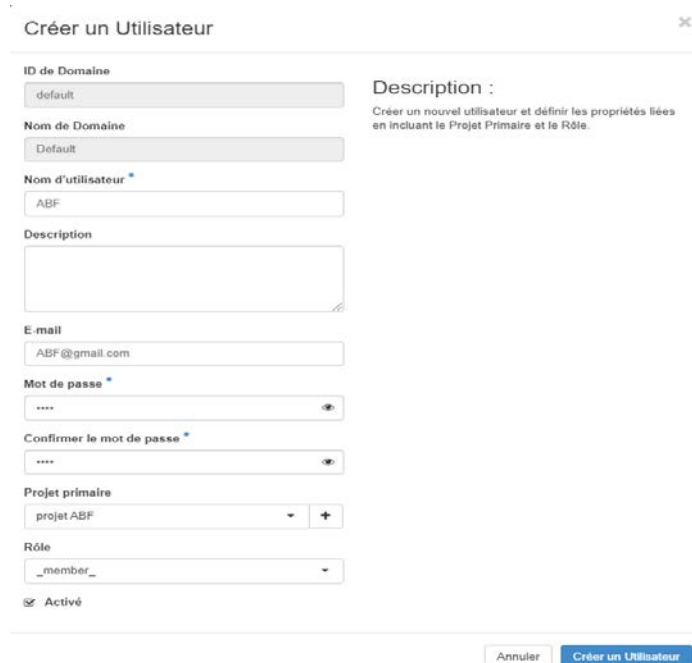


Figure 4.3 : Création de l'utilisateur "ABF"

Au cours de cette étape, nous avons affecté l'utilisateur « ABF » au tenant ABF et nous avons défini son rôle en tant que membre du projet.

1.2.3 Création du réseau externe

Durant cette phase, nous allons créer un réseau externe appelé réseau externe « ABF ». C'est un réseau partagé et géré par l'administrateur du projet comme le montre la figure 4.4.

Créer un réseau ✕

Réseau * Sous-réseau Détails du sous-réseau

Nom

Projet *

☒ État Administratif Actif

☒ Partagé

☒ Réseau externe

☒ Créer un sous-réseau

Indications de zone de disponibilité ?

Créez un nouveau réseau. En plus, un sous-réseau associé à ce réseau pourra être créé dans les étapes suivantes de cet assistant.

Figure 4.4 : Le réseau externe d'ABF

A ce réseau, nous avons associé un sous-réseau appelé sous-réseau externe « ABF » comme l'indique la figure 4.5.

Créer un réseau ✕

Réseau * **Sous-réseau** Détails du sous-réseau

Nom du sous-réseau

Adresse réseau ?

Version IP

Adresse IP de la passerelle ?

☐ Désactiver la passerelle

Crée un sous-réseau associé à un réseau. Vous devez entrer une "Adresse réseau" et une "Adresse IP de la passerelle" valide. Si vous n'entrez pas d'"Adresse IP de la passerelle", la première valeur (IP) de votre réseau sera assignée par défaut. Si vous ne souhaitez pas de passerelle, veuillez cocher "Désactiver la passerelle". Cliquez sur l'onglet "Détails Sous-réseaux" pour configurer des options avancées.

Figure 4.5 : Le sous-réseau externe d'ABF

Nous avons utilisé l'adresse réseau 172.25.52.0/24 et la passerelle 172.25.52.254.

1.2.4 Création du réseau interne

Chaque projet dispose d'un réseau interne. Le réseau privé de projet ABF est appelé réseau interne « ABF » comme le montre la figure 4.6.

Créer un réseau

Réseau Sous-réseau Détails du sous-réseau

Nom du réseau

réseau_interne_ABF

Créez un nouveau réseau. En plus, un sous-réseau associé à ce réseau pourra être créé dans les étapes suivantes de cet assistant.

☒ État Administratif Actif ?

☒ Créer un sous-réseau

Indications de zone de disponibilité ?

nova

Annuler « Retour Suivant »

Figure 4.6 : Le réseau interne d'ABF

A ce réseau, nous avons associé le sous-réseau 10.0.16.0/24 comme le montre la figure 4.7.

Créer un réseau

Réseau Sous-réseau Détails du sous-réseau

Nom du sous-réseau

sous-réseau_interne_ABF

Adresse réseau ?

10.0.16.0/24

Version IP

IPv4

Adresse IP de la passerelle ?

10.0.16.1

☐ Désactiver la passerelle

Crée un sous-réseau associé à un réseau. Vous devez entrer une "Adresse réseau" et une "Adresse IP de la passerelle" valide. Si vous n'entrez pas d'"Adresse IP de la passerelle", la première valeur (IP) de votre réseau sera assignée par défaut. Si vous ne souhaitez pas de passerelle, veuillez cocher "Désactiver la passerelle". Cliquez sur l'onglet "Détails Sous-réseaux" pour configurer des options avancées.

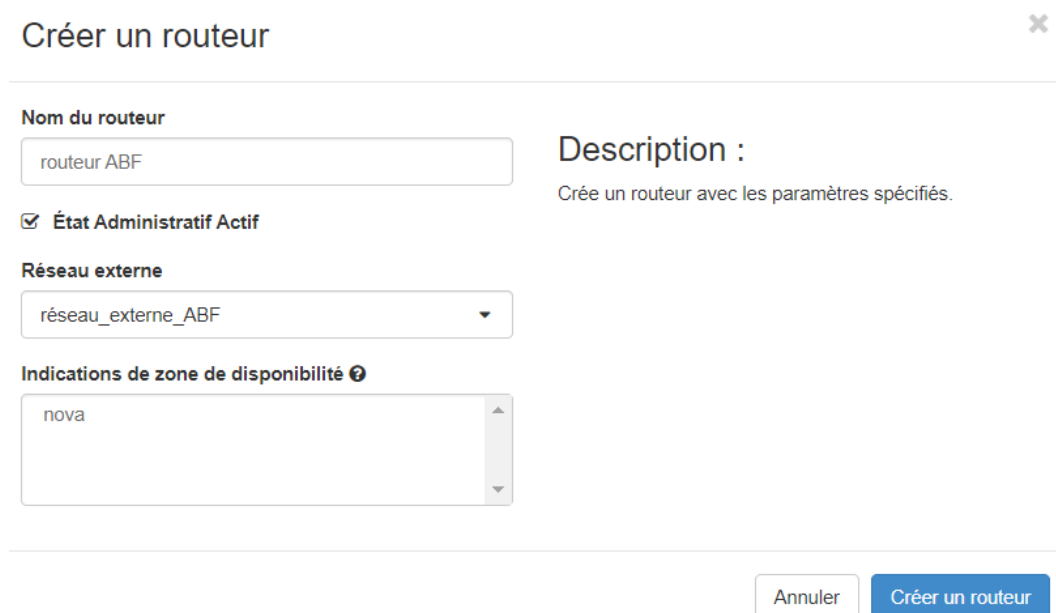
Annuler « Retour Suivant »

Figure 4.7 : Le sous-réseau interne d'ABF

Ensuite, nous avons alloué un pool d'adresses allant de 10.0.16.10 jusqu'à 10.0.16.100 au projet ABF.

1.2.5 Création d'un routeur

Les réseaux internes se connectent aux externes via un routeur virtuel qui effectue du NAT bidirectionnel. La figure 4.8 présente le routeur spécifié au projet « ABF ».



Créer un routeur

Nom du routeur

routeur ABF

☒ État Administratif Actif

Réseau externe

réseau_externe_ABF

Indications de zone de disponibilité ?

nova

Description :

Crée un routeur avec les paramètres spécifiés.

Annuler Créer un routeur

Figure 4.8: le routeur ABF

Le routeur « ABF » possède une passerelle sur le réseau externe « ABF » et une interface reliée au réseau interne « ABF » comme l'indique la figure 4.9.



Ajouter une interface

Sous-réseau *

réseau_interne_ABF: 10.0.16.0/24 (sous-réseau...)

Adresse IP (facultatif) ?

Description :

Vous pouvez connecter un sous-réseau spécifique au routeur.

Si aucune adresse IP n'est spécifiée ici, l'adresse IP de la passerelle du sous-réseau sera utilisée pour la nouvelle interface créée dans le routeur. Si l'IP de la passerelle est déjà utilisée, vous devez spécifier une adresse différente qui appartient au sous-réseau sélectionné.

Annuler Envoyer

Figure 4.9 : L'ajout de l'interface qui relie le routeur avec le réseau interne

Par défaut, l'adresse IP de l'interface qui permet de relier le routeur « ABF » au réseau interne est celle de la passerelle de sous-réseau.

1.2.6 Création d'un groupe de sécurité

Les groupes de sécurité sont des ensembles de règles de filtrage IP qui sont appliqués aux paramètres de l'interface réseau de la machine virtuelle.

La figure 4.10 représente le groupe de sécurité « ABF » qui autorise l'ICMP et Secure Shell (SSH).

Gérer les règles du groupe de sécurité : groupe_de_sécurité_ABF
(1be4c011-4ffc-4311-ac5f-e391701e4a68)

+ Ajouter une règle Supprimer les Règles

Affichage de 4 éléments

<input type="checkbox"/> Direction	Type de protocole (EtherType)	Protocole IP	Plage de ports	Préfixe IP distante	Groupe de sécurité distant	Description	Actions
<input type="checkbox"/> Sortie	IPv4	Tous	Tous	0.0.0.0/0	-	-	Supprimer une Règle
<input type="checkbox"/> Sortie	IPv6	Tous	Tous	:::/0	-	-	Supprimer une Règle
<input type="checkbox"/> Entrée	IPv4	ICMP	Tous	0.0.0.0/0	-	-	Supprimer une Règle
<input type="checkbox"/> Entrée	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Supprimer une Règle

Affichage de 4 éléments

Figure 4.10 : Les règles du groupe de sécurité ABF

Les règles ne sont pas figées. En effet, l'utilisateur « ABF » peut les modifier selon ses besoins.

1.2.7 Création d'une paire de clés

Les paires de clés sont le moyen pour se connecter à notre instance après son lancement. Nous avons créé une paire de clés comme l'indique la figure 4.11.

Créer une paire de clés

Nom de la paire de clés *

cle|ABF

✕ Annuler
+ Créer une paire de clés

Figure 4.11 : La paire de clés ABF

Lors de la création de la paire de clés ABF, un fichier qui contient la clé privée RSA sera stocké sur la machine. Cette clé privée va nous permettre d'accéder à l'instance via Putty.

Putty est un client SSH et Telnet.

1.2.8 Allocation d'une IP flottante

Les IPs flottantes sont configurées uniquement au niveau du routeur pour qu'il puisse réaliser le NAT. En effet, ces adresses sont utilisées pour communiquer en dehors du Cloud.

Pour cela, nous avons alloué une IP flottante comme le montre la figure 4.12.

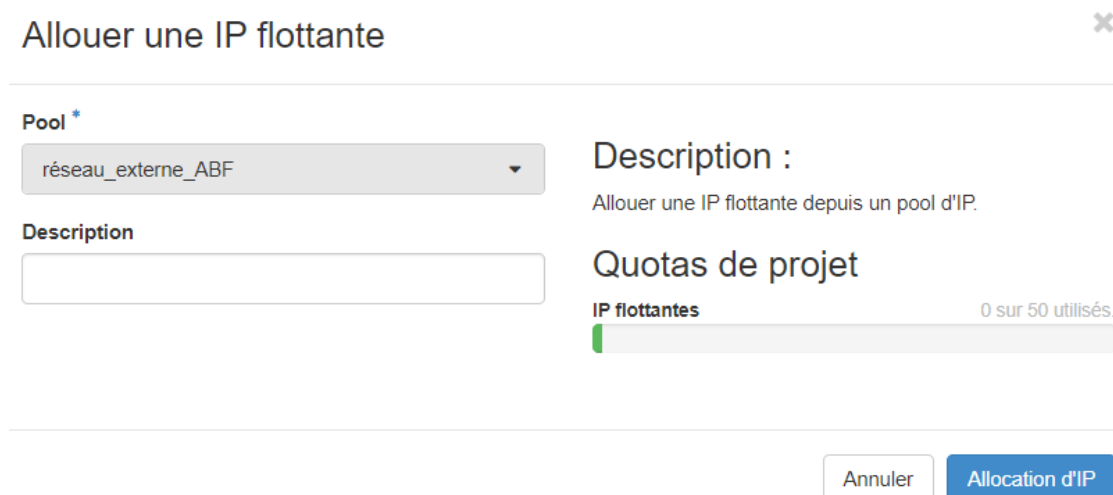


Figure 4.12 : Allocation d'une IP flottante

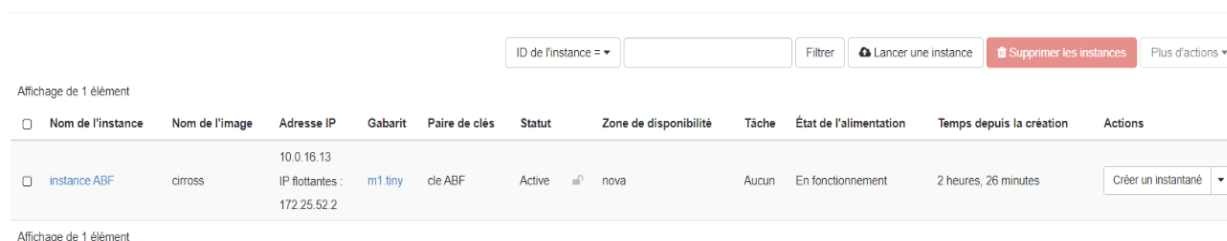
Les IPs flottantes allouées au projet ABF peuvent être associées à une instance ou dissociées d'elle à tout moment.

1.2.9 Lancement de l'instance

Pour créer une nouvelle instance, il faut préciser le réseau, l'image source, le groupe de sécurité, le gabarit et éventuellement la clef SSH à utiliser.

Dans cet exemple, nous avons utilisé l'image cirros et le gabarit m1.tiny qui dispose de 512 Mo de RAM, 1 VCPU et 1Go de stockage pour créer l'instance ABF comme l'indique la figure 4.13.

Instances



Nom de l'instance	Nom de l'image	Adresse IP	Gabarit	Paire de clés	Statut	Zone de disponibilité	Tâche	État de l'alimentation	Temps depuis la création	Actions
instance ABF	cirros	10.0.16.13 IP flottantes : 172.25.52.2	m1.tiny	cle ABF	Active	nova	Aucun	En fonctionnement	2 heures, 26 minutes	Créer un instantané

Figure 4.13 : L'instance ABF

L'instance « ABF » dispose de deux adresses

- Une adresse IP fixe privée 10.0.16.13 attribuée lors de sa création. Elle est utilisée pour la communication entre les instances au sein du même projet.
- Une adresse IP flottante 172.25.52.2 avec laquelle l'instance « ABF » peut communiquer en dehors du Cloud.

De plus, la zone de disponibilité des instances est par défaut nova.

Néanmoins, si l'instance a besoin d'un volume supplémentaire, nous pouvons l'attacher à un disque dur externe virtuel.

Pour cela, nous avons créé le volume « ABF » de taille 4 Go comme l'indique la figure 4.14.

Figure 4.14 : Le volume ABF

Le volume « ABF » utilise le protocole de stockage en réseau iscsi qui permet de relier les installations de stockage de données.

Ensuite, nous l'avons attaché à l'instance « ABF » via la commande « add volume » suivie de l'ID de volume et du projet « ABF », comme le montre la figure 4.15.

```
[root@localhost ~(keystone_admin)]# openstack server add volume 413928e5-cc40-4821-9799-12afd4ffc5bf 654e9ec7-f0b9-416a-9d2a-e46002a20b37
[root@localhost ~(keystone_admin)]# openstack volume list
```

ID	Name	Status	Size	Attached to
654e9ec7-f0b9-416a-9d2a-e46002a20b37	volume ABF	in-use	4	Attached to 413928e5-cc40-4821-9799-12afd4ffc5bf on /dev/vdb
ae8bf659-5ba0-4558-8aa2-f106ecf2f8f0	volume1	in-use	2	Attached to 42c0830d-0c1b-4fe3-821e-11b284ed5b1f on /dev/vdb

```
[root@localhost ~(keystone_admin)]#
```

Figure 4.15 : Attachement de volume ABF

1.2.10 Définition des quotas

Pour éviter que les capacités du système ne soient épuisées sans notification, nous avons défini les quotas du projet « ABF » via la ligne de commande OpenStackclient.

En effet, nous avons réservé 6 VCPUs , 10 Go de RAM et une instance virtuelle dans le service nova en utilisant la commande « nova quota-update », comme le montre la figure 4.16.

```
[root@localhost ~](keystone_admin)]# nova quota-update --ram 10240 7c8ee5f26d6f475e9450085f76b43d4e
[root@localhost ~](keystone_admin)]# nova quota-update --cores 6 7c8ee5f26d6f475e9450085f76b43d4e
[root@localhost ~](keystone_admin)]# nova quota-update --instances 1 7c8ee5f26d6f475e9450085f76b43d4e
[root@localhost ~](keystone_admin)]# nova quota-show --tenant 7c8ee5f26d6f475e9450085f76b43d4e
```

Quota	Limit
instances	1
cores	6
ram	10240
metadata_items	128
key_pairs	100
server_groups	10
server_group_members	10

```
[root@localhost ~](keystone_admin)]#
```

Figure 4.16 : Les quotas de service nova

7c8ee5f26d6f475e9450085f76b43d4e est l’ID du projet ABF.

Ensuite, nous avons vérifié la mise à jour des quotas à l’aide de la commande « quota-show » suivie de l’ID du projet « ABF ».

De plus, nous avons affecté 50 Go de disque au projet « ABF » dans le service cinder comme l’indique la figure 4.17.

```
[root@localhost ~](keystone_admin)]# cinder quota-update --gigabytes 50 7c8ee5f26d6f475e9450085f76b43d4e
```

Property	Value
backup_gigabytes	1000
backups	10
gigabytes	50
gigabytes_iscsi	-1
groups	10
per_volume_gigabytes	-1
snapshots	10
snapshots_iscsi	-1
volumes	10
volumes_iscsi	-1

```
[root@localhost ~](keystone_admin)]#
```

Figure 4.17 : Le quota de service cinder

1.2.11 Création d’un conteneur

Un conteneur est un compartiment permettant de stocker les données et il constitue une façon de les organiser. Dans ce contexte, le service swift d’OpenStack nous permet de créer des conteneurs via la CLI OpenStackclient en utilisant la commande « post » :

```
$ swift post conteneur1
```

Par la suite, nous avons attribué des autorisations en lecture et en écriture à l’utilisateur « ABF » comme suit :

```
$ swift post conteneur1 -r "admin : admin , projet ABF : ABF"
```

```
$ swift post conteneur1 -w "admin : admin , projet ABF : ABF"
```

Ensuite, nous avons stocké la clé « ABF » dans ce conteneur grâce à la commande « upload ».

```
$ swift upload conteneur1 clé_ABf.pem
```

Le service swift est offert seulement par l'installateur Packstack. En effet, Devstack ne permet pas le stockage d'objet.

La configuration de l'environnement Cloud, dédié au client « ABF », nécessite un temps de traitement important qui peut affecter la qualité de service et la disposition des ressources informatiques, d'où l'importance de l'automatisation des environnements Cloud sur OpenStack.

2. Implémentation d'Ansible

Ansible permet d'automatiser les tâches répétitives d'OpenStack lors de la création d'un environnement Cloud.

Pour installer Ansible, nous avons besoin de :

- Un nœud de contrôle Ansible : un système Ubuntu 18.04 dont ses caractéristiques sont 25 Go de disque dur, 12 Go de RAM et 6 VCPUs.
- Un hôte où nous avons installé OpenStack.

Depuis notre nœud de contrôle Ansible, nous avons exécuté la commande suivante pour inclure le PPA du projet officiel dans la liste des sources de notre système :

```
$ sudo apt-add-repository ppa:ansible/ansible
```

Puis, nous rafraichissons notre système avec :

```
$ sudo apt update
```

Suite à cette mise à jour, nous avons installé Ansible en utilisant cette commande :

```
$ sudo apt install ansible
```

Dès que l'installation d'Ansible sera terminée, nous générerons une paire de clés SSH comme l'indique la figure 4.18 pour relier le nœud de contrôle Ansible avec l'hôte 172.25.52.123.

```
ansible@ansible-virtual-machine:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
/home/ansible/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa.
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:d+Zwr39kLfzwtjXhmXaBTSSfHQJmCkOTzre7binhdY ansible@ansible-virtual-machine
The key's randomart image is:
+---[RSA 2048]---+
|      .+=+o o |
|      .ooo =+ |
|      * .  +. |
|      oo  +  |
|      .S + +o +. |
|      . ooB . = O |
|      ooE. .&o |
|      ...o .. O |
|      o+ ...+. |
+---[SHA256]-----+
ansible@ansible-virtual-machine:~$
```

Figure 4.18 : Génération des clés SSH

Une fois que la paire de clés sera générée, nous copierons la clé publique dans le fichier `authorized_keys` de l'hôte cible avec la commande « `ssh-copy-id` ».

Après avoir établi la liaison entre le nœud de contrôle Ansible et l'hôte cible, nous devons modifier le contenu du fichier inventaire Ansible. Nous avons créé un groupe nommé « `client` » dans lequel nous avons défini l'adresse de l'hôte Devstack, et les paramètres d'accès SSH (`ansible_ssh_user`, `ansible_ssh_pass`).

Nous avons également spécifié les paramètres d'authentification au tableau de bord d'OpenSatck :

- `OS_PROJECT_NAME` est le nom du tenant.
- `OS_USERNAME` est le nom de l'utilisateur.
- `OS_PASSWORD` est le mot de passe de l'utilisateur.
- `OS_REGION_NAME` est le nom de la région où nous avons créé le projet qui est par défaut « `RegionOne` ».
- `OS_AUTH_URL` est l'URL d'authentification au tableau de bord d'OpenStack.
- `OS_PROJECT_DOMAIN_NAME` est le nom de domaine où nous avons créé le projet qui est par défaut « `Default` ».
- `OS_USER_DOMAIN_NAME` est le nom de domaine où nous avons créé l'utilisateur qui est par défaut « `Default` ».

La figure 4.19 montre le contenu du fichier `/etc/ansible/hosts`.

```
all:
  hosts:
    client:
      ansible_host: 172.25.52.123
      ansible_ssh_user: ansible
      ansible_ssh_pass: root15987
      OS_USERNAME: "admin"
      OS_PASSWORD: "devstack"
      OS_REGION_NAME: "RegionOne"
      OS_AUTH_URL: "http://172.25.52.123/identity"
      OS_USER_DOMAIN_NAME: "Default"
      OS_PROJECT_DOMAIN_NAME: "Default"
      OS_PROJECT_NAME: "admin"
```

Figure 4.19 : Inventaire Ansible

Suite à l'ajout des paramètres de l'hôte ciblé, nous allons tester la connectivité entre les deux machines comme le montre la figure 4.20.

```
ansible@ansible-virtual-machine:~$ ansible all -m ping -u root
172.25.52.123 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
ansible@ansible-virtual-machine:~$
```

Figure 4.20 : Test de connectivité

Nous avons obtenu une réponse « `pong` » de l'hôte contenant OpenStack (172.25.52.123), cela signifie que nous sommes prêts à exécuter les commandes Ansible et les playbooks sur ce serveur.

3. Playbooks Ansible

Tous les playbooks sont écrits en langage YAML. Chacun est constitué au minimum d'une variable hôtes qui désigne les machines cibles (OpenStack) et une tâche qui définit une action à accomplir dans OpenStack. Parmi ceux nous citons :

a) Playbook « projet.yml »

Pour créer un projet dans OpenStack, nous avons utilisé le playbook « projet.yml » présenté dans la figure 4.21.

```
---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest"
  tasks:
    - name: Create a project
      os_project:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        region_name: RegionOne
        domain_id: default
        name: "{{ OS_PROJECT }}"
...

```

Figure 4.21 : Playbook « projet.yml »

Dans ce playbook, nous avons appelé le module `os_project` responsable de la gestion de projet OpenStack. Il permet de créer, de mettre à jour ou de supprimer un projet. Ce module utilise comme variables :

- State : état du projet (présent, absent).
- Auth : dictionnaire qui contient les paramètres d'authentification d'OpenStack.
- Region_name : nom de région qui est par défaut « RegionOne ».
- Domain_id : id de domaine dans lequel nous allons créer le projet qui est par défaut « Default ».
- Name : nom de projet créée.

Puis, nous exécutons le playbook en tapant la commande suivante :

```
$ cd /etc/ansible
$ ansible-playbook -i inventaire.yml projet.yml
```

La sortie de la commande précédente est présentée dans la figure 4.22.

```
TASK [Gathering Facts] *****
ok: [172.25.52.123]

TASK [Create a project] *****
ok: [172.25.52.123]

PLAY RECAP *****
172.25.52.123 : ok=2 changed=0 unreachable=0 failed=0 s
kipped=0 rescued=0 ignored=0
```

Figure 4.22 : Exécution du playbook "projet.yml"

Le mot clé « ok » indique que le projet « ansibletest » est créé dans l'infrastructure OpenStack.

b) Playbook « utilisateur.yml »

Pour créer un utilisateur OpenStack, nous exécutons le playbook « utilisateur.yml » qui utilise le module `os_user` dont ses paramètres sont :

- Name : nom de l'utilisateur
- Email : email de l'utilisateur
- Password : le mot passe de l'utilisateur « chafik »

La figure 4.23 présente le playbook « utilisateur.yml ».

```
---
- hosts: client
  vars:
    USERMEMBER: "chafik"
    PASSWORD_USER: "root"
    EMAIL_USER: "chafik@gmail.com"
    OS_PROJECT: "ansibletest"
  tasks:
    - name: Create user
      os_user:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        name: "{{ USERMEMBER }}"
        password: "{{ PASSWORD_USER }}"
        email: "{{ EMAIL_USER }}"
        domain: default
        default_project: "{{ OS_PROJECT }}"
```

Figure 4.23 : Playbook « utilisateur.yml »

c) Playbook « attacher_un_utilisateur.yml »

Le playbook, présenté dans la figure 4.24, permet d'attacher l'utilisateur « chafik » au projet « ansibletest » grâce au module `os_user_role`. Nous devons également attacher l'administrateur pour qu'il puisse gérer l'environnement.

```
---
- hosts: client
  vars:
    USERMEMBER: "chafik"
    PASSWORD_USER: "root"
    EMAIL_USER: "chafik@gmail.com"
    OS_PROJECT: "ansibletest"
  tasks:|
    - name: Attach user to project
      os_user_role:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        role: _member_
        region_name: RegionOne
        project: "{{ OS_PROJECT }}"
        user: "{{ USERMEMBER }}"
    - name: Attach admin to project
      os_user_role:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        user: admin
        role: admin
        project: "{{ OS_PROJECT }}"
```

Figure 4.24 : Playbook « attacher_un_utilisateur.yml »

L'utilisateur « chafik » a les privilèges d'un membre. Il a le droit de modifier la configuration de son infrastructure Cloud y compris les instances, les IPs flottantes, les clés et les images, tout en respectant les quotas définis par l'administrateur d'OpenStack.

d) Playbook « quotas.yml »

Les quotas définissent les limites d'utilisation des ressources disponibles telles que la quantité de CPU ou le nombre d'instances. Ils sont appliqués au projet.

Nous avons précisé les quotas de projet « ansibletest » grâce au module `os_quotas`. Par exemple, dans ce playbook l'utilisateur « chafik » a réservé 10 VCPUs et 4 Go de RAM.

La figure 4.25 représente le playbook « quotas.yml »

```
---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest"
#vars quota
  INSTANCES_PROJECT: "5"
  CORES_PROJECT: "8"
  RAM_PROJECT: "10240"
  VOLUMES_PROJECT: "5"
  SNAPSHOTS_PROJECT: "5"
  GIGABYTES_PROJECT: "50"
  FLOATING_IPS: "5"
  ROUTER: "5"
  tasks:
    - name: set a project quotas
      os_quota:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        name: "{{ OS_PROJECT }}"
        instances: "{{ INSTANCES_PROJECT }}"
        cores: "{{ CORES_PROJECT }}"
        ram: "{{ RAM_PROJECT }}"
        volumes: "{{ VOLUMES_PROJECT }}"
        snapshots: "{{ SNAPSHOTS_PROJECT }}"
        gigabytes: "{{ GIGABYTES_PROJECT }}"
        floating_ips: "{{ FLOATING_IPS }}"
        router: "{{ ROUTER }}"
```

Figure 4.25 : Playbook « quotas.yml »

En se basant sur ce playbook, nous avons défini aussi le nombre d'instances, de volumes, et des instantanées pour l'utilisateur « chafik » dans la tâche « set project quotas » qui appelle également le module `os_quotas`.

e) Playbook « réseau_interne.yml »

Le playbook réseau interne est présenté dans la figure 4.26.

```
---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest"
#vars interne network
  NAME_INT_NETWORK: "int_network"
#vars interne subnet
  NAME_INT_SUBNET: "int_subnet"
  CIDR_INT_SUBNET: "10.6.0.0/24"
  tasks:
    - name : create an interne network
      os_network:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          name: "{{ NAME_INT_NETWORK }}"
          project: "{{ OS_PROJECT }}"
    - name: create an interne subnet
      os_subnet:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          network_name: "{{ NAME_INT_NETWORK }}"
          name: "{{ NAME_INT_SUBNET }}"
          cidr: "{{ CIDR_INT_SUBNET }}"
          dns_nameservers: "{{ OS_DNS_NAMESERVER1 }}"
          project: "{{ OS_PROJECT }}"
```

Figure 4.26 : Playbook "réseau_interne.yml"

Ce playbook appelle deux modules :

- `os_network` qui permet de créer le réseau interne et de l'associer au projet « `ansibletest` ».
- `os_subnet` qui permet d'ajouter un sous-réseau. Il utilise le paramètre « `cidr` » qui est une représentation de sous-réseau ainsi que l'option « `enable_dhcp` » qui est activée.

g) Playbook « réseau_externe.yml »

De même pour la création du réseau externe, nous avons utilisé les deux modules `os_network` et `os_subnet`. Cependant, nous avons utilisé le paramètre « `external` » pour indiquer qu'il s'agit d'un réseau public.

De plus, nous avons spécifié un pool d'adresses IP dédiées comme le montre la figure 4.27.

```
---
- hosts: client
  vars:
#vars externe network
  NAME_EXT_NETWORK: "ext_network"
#vars externe subnet
  NAME_EXT_SUBNET: "ext_subnet"
  CIDR_SUBNET: "172.25.52.0/24"
  DNS_NAMESERVERS: "8.8.8.8"
  ALLOCATION_POOL_START: "172.25.52.122"
  ALLOCATION_POOL_END: "172.25.52.138"
  tasks:
    - name : create an externe network
      os_network:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          name: "{{ NAME_EXT_NETWORK }}"
          external: true
          shared: yes
    - name: create an externe subnet
      os_subnet:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          network_name: "{{ NAME_EXT_NETWORK }}"
          name: "{{ NAME_EXT_SUBNET }}"
          cidr: "{{ CIDR_SUBNET }}"
          enable_dhcp: yes
          allocation_pool_start: "{{ ALLOCATION_POOL_START }}"
          allocation_pool_end: "{{ ALLOCATION_POOL_END }}"
          dns_nameservers: "{{ DNS_NAMESERVERS }}"
```

Figure 4.27 : Playbook "réseau_externe.yml"

h) Playbook « routeur.yml »

Le routeur permet d'effectuer la traduction d'adresses. Il s'agit d'un élément indispensable d'une architecture réseau. La figure 4.28 décrit la syntaxe du palybook routeur.

```

---
- hosts: client
  vars:
#vars router
    NAME_ROUTER: "router1"
    OS_PROJECT_NAME: "admin"
    OS_PROJECT: "ansibletest"
    NAME_EXT_NETWORK: "ext_network"
    NAME_INT_NETWORK: "int_network"
    NAME_INT_SUBNET: "int_subnet"
    NAME_EXT_SUBNET: "ext_subnet"
  tasks:
    - name: create a router
      os_router:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        name: "{{ NAME_ROUTER }}"
        network: "{{ NAME_EXT_NETWORK }}"
        external_fixed_ips:
          - subnet: "{{ NAME_EXT_SUBNET }}"
        interfaces:
          - "{{ NAME_INT_SUBNET }}"
        enable_snat: yes
        project: "{{ OS_PROJECT }}"
...

```

Figure 4.28 : Playbook "routeur.yml"

Le playbook routeur appelle le module `os_router` pour créer un routeur. L'option « `external_fixed_ips` » lui permet de spécifier le nom du réseau externe.

En outre, le paramètre « `interfaces` » ajoute une interface sur le routeur pour le relier avec le réseau interne.

j) Playbook « groupe_de_sécurité.yml »

Dès que nous avons créé le routeur, nous avons défini les règles de filtrage IP pour protéger l'infrastructure Cloud des dangers externes. Dans ce contexte, nous avons autorisé tout accès SSH aux entités ayant l'autorisation d'accéder aux instances.

La figure 4.29 représente ce playbook.

```
---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest"
#vars security_group
  SECURITY_GROUP: "groupe1"
  tasks:
    - name: Create a security group
      os_security_group:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          name: "{{ SECURITY_GROUP }}"
          project: "{{ OS_PROJECT }}"
    - name: Create a security group rule
      os_security_group_rule:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        security_group: "{{ SECURITY_GROUP }}"
        protocol: tcp
        direction: ingress
        port_range_min: 22
        port_range_max: 22
        remote_ip_prefix: 0.0.0.0/0
        project: "{{ OS_PROJECT }}"
```

Figure 4.29 : Playbook "groupe_de_sécurité.yml"

Ce playbook appelle deux modules :

- `os_security_group` pour créer un groupe de sécurité que nous avons nommé « groupe1 »
- `os_security-group_rule` pour spécifier les règles de filtrage IP. Nous devons spécifier le protocole, les ports et la direction.

k) Playbook « paire_de_clés.yml »

La paire de clés nous permet d'accéder aux instances de l'extérieur. La figure 4.30 représente le palybook utilisé pour créer la clé « ansible_key ».

```
---
- hosts: client
  vars:
    OS_PROJECT_NAME: "ansibletest"
#vars key
  KEY_NAME: ansible_key|
  tasks:
    - name: create a key pair
      os_keypair:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        name: "{{ KEY_NAME }}"
```

Figure 4.30 : Playbook "paire_de_clés.yml "

La paire de clés sera créée grâce au module os_keypair dans le projet « ansibletest ».

k) Playbook « instance.yml »

Pour lancer une instance, nous avons utilisé le module os_serer. Ce dernier nécessite l'ajout de quelques paramètres :

- Auth : c'est un dictionnaire contenant les paramètres d'accès au OpenStack.
- State : indique l'état de l'instance créée.
- Name : nom de l'instance qui est « ansible_instance ».
- Image : l'id de l'image. Nous avons utilisé l'image cirros-0.5.1-x86_64-disk.
- Key_name : c'est le nom de paire de clés (ansible_key).
- Security_groups : c'est nom de groupe de sécurité associé à cette instance (groupe 1).
- Network : indique le réseau interne auquel l'instance est attachée.
- Auto_ip : permet aux instances d'avoir une adresse du pool de réseau interne (int_net).
- Flavor : c'est le nom de gabarit dans lequel l'instance doit être créée.

La figure 4.31 indique la structure de playbook « instance.yml »

```

---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest"
#vars instance
    NAME_INSTANCE: "ansible_instance"
    NAME_IMAGE: "cirros-0.5.1-x86_64-disk"
    NAME_INT_NETWORK: "int_network"
    SECURITY_GROUP: "groupe1"
    KEY_NAME: ansible_key
    NAME_FLAVOR: "flavor1"
  tasks:
    - name: create a new instance
      os_server:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        name: "{{ NAME_INSTANCE }}"
        image: "{{ NAME_IMAGE }}"
        key_name: "{{ KEY_NAME }}"
        flavor: "{{ NAME_FLAVOR }}"
        network: "{{ NAME_INT_NETWORK }}"
        security_groups: "{{ SECURITY_GROUP }}"
        auto_ip: yes
        region_name: RegionOne

```

Figure 4.31 : Playbook « instance.yml »

Après l'exécution du playbook, l'instance « ansible_instance » est créée. Elle est présente dans la liste des instances au niveau de Dashboard d'OpenStack comme l'indique la figure 4.32.

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State
<input type="checkbox"/>	ansible_instance	cirros-0.5.1-x86_64-disk	10.0.16.136	cirros256	ansible_key	Active	nova	Aucun	En fonctionnement

Figure 4.32 : Ansible_instance

l) Playbook « ip_flottante.yml »

Une fois que nous avons créé une instance, nous lui associons une IP flottante grâce au module `os_floating_ip` comme le montre la figure 4.33.

```
---
- hosts: client
  vars:
    OS_PROJECT_NAME: "ansibletest"
    OS_PROJECT: "ansibletest"
    NAME_INSTANCE: "ansible_instance"
    NAME_EXT_NETWORK: "ext_network"

  tasks:
    - name: create a floating_ip
      os_floating_ip:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        reuse: yes
        server: "{{ NAME_INSTANCE }}"
        network: "{{ NAME_EXT_NETWORK }}"
        fixed_address: 10.0.16.157
        destination_nat: int_network
...|
```

Figure 4.33 : Playbook "ip_flottante.yml "

L'adresse 172.25.52.134 est affectée à « ansible_instance » que nous utilisons pour accéder à la console de la VM via SSH.

m) Playbook « instantané.yml »

Un instantané est un mécanisme qui vous permet de créer une nouvelle image à partir d'une instance en cours d'exécution.

Cela sert principalement à deux fins :

- En tant que mécanisme de sauvegarde : nous enregistrons le disque principal de notre instance dans une image et nous démarrons plus tard une nouvelle instance à partir de cette image avec les données enregistrées.
- En tant que mécanisme de création de modèles : nous personnalisons une image de base et l'enregistrons pour l'utiliser comme modèle pour de nouvelles instances [14].

La figure 4.34 présente la syntaxe de playbook instantané.

```
---
- hosts: client
  vars:
#vars snapshot
    OS_PROJECT: "ansibletest"
    NAME_SNAPSHOT: "snapshot1"
    NAME_VOLUME: "volume1"
  tasks:
    - name : create a volume
      os_volume_snapshot:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          display_name: "{{ NAME_SNAPSHOT }}"
          volume: "{{ NAME_VOLUME }}"
...|
```

Figure 4.34 : Playbook "instantané.yml"

n) Playbook « volume.yml »

Le palybook volume permet de créer un espace de disque supplémentaire que l'utilisateur associe à « ansibletest » en utilisant le module `os_volume` comme l'indique la figure 4.35.

```
---
- hosts: client
  vars:
#vars project
    OS_PROJECT: "ansibletest"
#vars volume
    NAME_VOLUME: "volume1"
    SIZE_VOLUME: "2"
  tasks:
    - name : create a volume
      os_volume:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
          display_name: "{{ NAME_VOLUME }}"
          size: "{{ SIZE_VOLUME }}"
...|
```

Figure 4.35: Playbook "volume.yml "

Nous avons spécifié la taille de volume en gigaoctet en utilisant l'option « size ». Cet espace de stockage sera attaché à une instance du projet « ansibletest ».

o) Playbook « conteneur.yml »

Pour le projet ABF, nous avons créé un conteneur où nous avons stocké la clé ABF. Nous avons opté pour ce playbook qui utilise le module `os_object` comme l'indique la figure 4.36.

```
---
- hosts: client
  vars:
    OS_PROJECT: "ansibletest-project1"
  #vars container
    CONTAINER: "conteneur2"

  tasks:
    - name : create a container
      os_object:
        state: present
        auth:
          auth_url: "{{ OS_AUTH_URL }}"
          username: "{{ OS_USERNAME }}"
          password: "{{ OS_PASSWORD }}"
          project_name: "{{ OS_PROJECT_NAME }}"
          os_user_domain_name: "{{ OS_USER_DOMAIN_NAME }}"
          os_project_domain_name: "{{ OS_PROJECT_DOMAIN_NAME }}"
        container: "{{ CONTAINER }}"
        container_access: public
        filename: /home/packstack/downloads/ansible_key.pem
```

Figure 4.36 : Playbook "conteneur.yml"

Nous avons défini le chemin du fichier local à ajouter dans le « conteneur1 » grâce à l'option « filename ». Nous avons également modifié les droits d'accès pour le rendre accessible de l'extérieur via l'URL correspondante.

Donc Ansible nous permet de simplifier l'administration d'OpenStack. IL vise à fournir des gains de productivité importants. C'est un outil simple à utiliser grâce à sa syntaxe écrite en YAML.

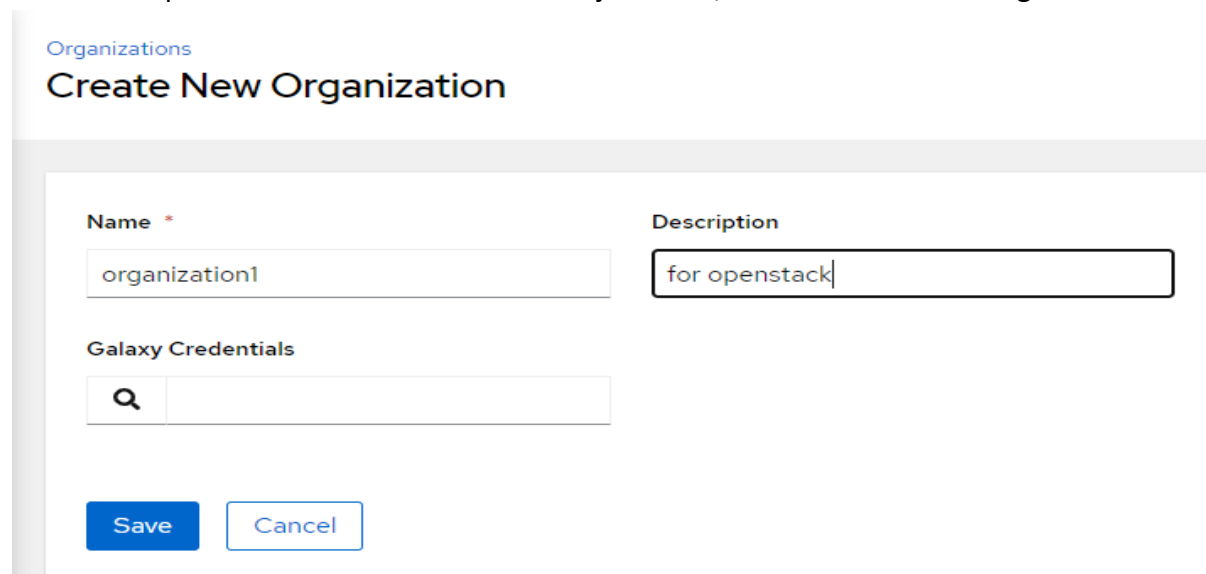
De plus, il ne nécessite pas l'installation d'autres logiciels sur les systèmes clients que nous souhaitons automatiser.

Cependant, Ansible ne délègue pas l'exécution de plusieurs playbooks à des équipes tierces. En outre, il est nécessaire de modifier manuellement les variables des playbooks dès que nous voulons créer un autre environnement Cloud. Pour éviter ces problèmes, nous avons utilisé l'interface graphique AWX (voir annexe 3) qui permet de mieux gérer et automatiser OpenStack.

4. AWX

4.1 Exécution d'un playbook sur le serveur AWX

Pour exécuter un playbook sur le serveur AWX, nous avons suivi les étapes suivantes :
Tout d'abord, nous avons ajouté une organisation nommée « organization1 ». Elle constitue le niveau le plus élevé de la hiérarchie des objets AWX, comme le montre la figure 4.37.



Organizations

Create New Organization

Name *

Description

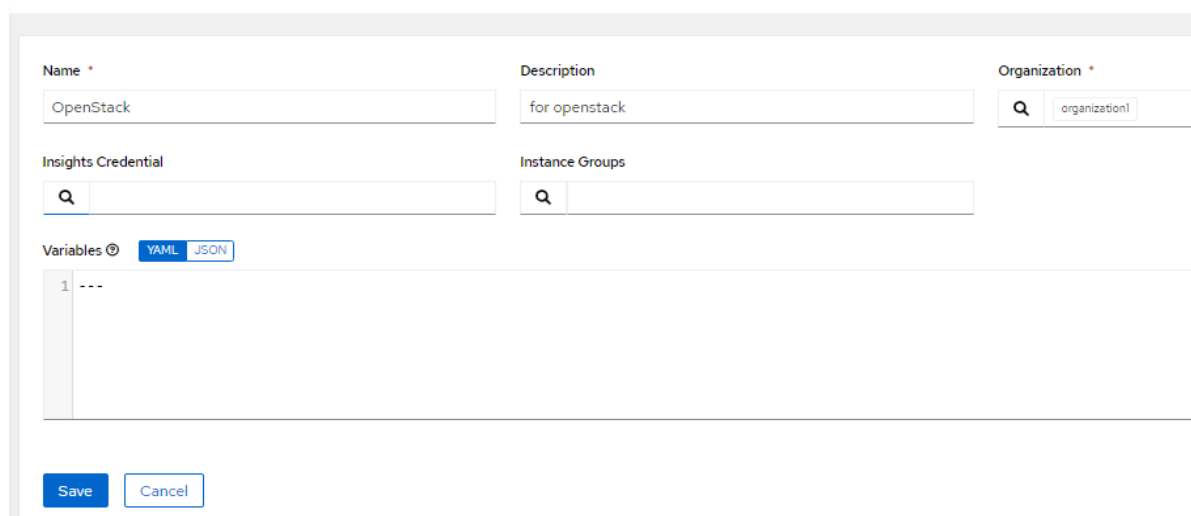
Galaxy Credentials

Figure 4.37 : Création d'une nouvelle organisation

L'organisation1 inclut l'utilisateur « OpenStack_User », le projet « OpenStack_Ansible » et l'inventaire « OpenStack ».

Ensuite, nous avons créé un inventaire nommé « OpenStack » comme l'indique la figure 4.38. Ce dernier contient l'hôte « 172.25.52.123 ».

Create new inventory



Name *

Description

Organization *

Insights Credential

Instance Groups

Variables

1 ---

Figure 4.38 : Création de l'inventaire OpenStack

Avec le serveur AWX, les paramètres d'authentification sont stockés séparément. Pour ce faire, nous avons créé un identifiant contenant les paramètres d'accès au tableau de bord OpenStack comme l'indique la figure 4.39.

Credentials

Create New Credential



Figure 4.39 : l'identifiant d'OpenStack

Nous avons créé un projet nommé « OpenStack_Ansible » dont son type d'informations d'identification de contrôle du code source est git. Nous avons ajouté un référentiel GitHub public : <https://github.com/yosr-ghazouani/ansibletest> comme le montre la figure 4.40.

Projects

Create New Project



Figure 4.40 : L'ajout d'un projet dans AWX

Par la suite, nous avons ajouté un modèle grâce auquel le playbook aura la spécificité d'exécuter à partir du projet OpenStack Ansible. La figure 4.41 présente le modèle « projet » qui exécute le playbook « projet.yml ».

[Templates > project](#)
Details

◀ Back to Templates **Details** Access Notifications Schedules Completed Jobs Survey

Name	project	Description	create project	Job Type	run
Organization	organization1	Inventory	openstack	Project	OpenStack_Ansible
Playbook	project.yml	Forks	0	Verbosity	0 (Normal)
Timeout	0	Show Changes	Off	Job Slicing	1
Options	Enable Privilege Escalation	Created	29/04/2021, 15:21:37 by admin	Last Modified	29/04/2021, 15:21:37 by admin
Credentials	Cloud: openstack SSH: test				

Figure 4.41 : Le modèle "projet"

Pour lancer le modèle « projet », nous avons cliqué sur l'icône du lanceur de travaux.

De plus, nous avons mis en place le modèle « utilisateur » permettant de créer un utilisateur OpenStack présenté dans la figure 4.42.

[Templates > user](#)
Details

◀ Back to Templates

Details

Access

Notifications

Schedules

Completed Jobs

Survey

Name	user	Description	create user	Job Type	run
Organization	organization1	Inventory	openstack	Project	OpenStack_Ansible
Playbook	useryml	Forks	0	Verbosity	0 (Normal)
Timeout	0	Show Changes	Off	Job Slicing	1
Options	Enable Privilege Escalation	Created	29/04/2021, 15:29:36 by admin	Last Modified	29/04/2021, 15:29:36 by admin
Credentials	Cloud: openstack	SSH: test			

Figure 4.42 : Le modèle "utilisateur"

Ce modèle exécute le playbook « utilisateur.yml » hébergé dans le référentiel GitHub où nous avons défini les privilèges de l'utilisateur « ansibleuser ».

Nous avons aussi inventé d'autres modèles permettant de créer un environnement Cloud sur OpenStack (voir annexe 3).

4.2 Avantages d'AWX

Parmi les valeurs ajoutées par le serveur AWX c'est que nous pouvons exécuter plusieurs playbooks en parallèle et que nous pouvons modifier les variables de l'environnement sans toucher au contenu des playbooks dans le référentiel GitHub public.

4.2.1 Multi-playbooks workflows

Les flux de travail permettent la configuration d'une séquence de modèles de travail hétérogène. Nous avons testé cette fonctionnalité avec les modèles « projet », « utilisateur »

« quotas », « réseau_interne », « réseau_externe », « routeur », « groupe_de_sécurité », « volume », « instance » et « ip_flottante ».

L'exécution du flux de travail « test » est présentée dans la figure 4.43.

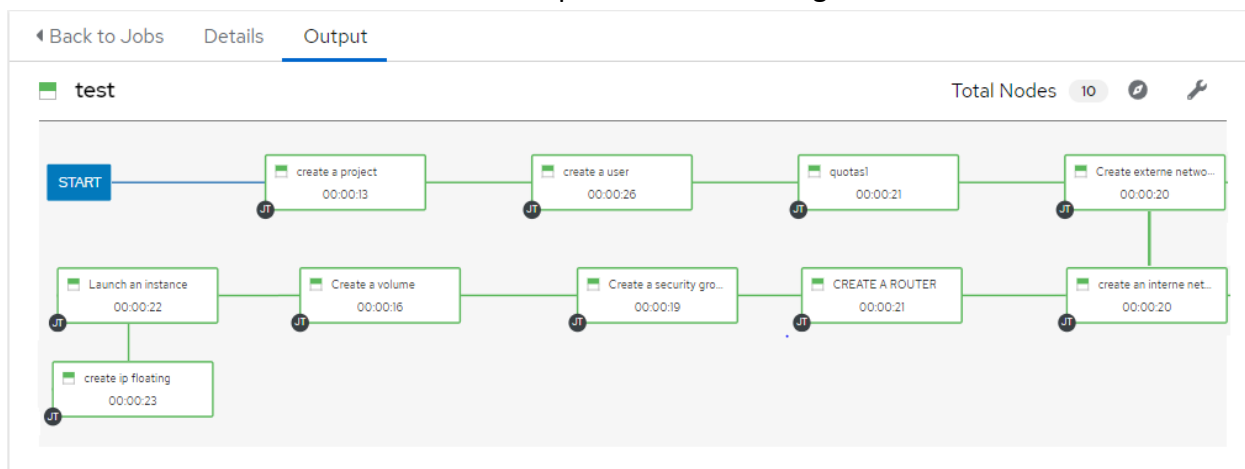


Figure 4.43 : L'exécution du flux de travail "test"

La couleur verte indique que l'exécution de modèle a réussi c'est-à-dire le projet, l'utilisateur et les quotas sont définis.

Donc grâce au flux de travail, nous pouvons exécuter plusieurs playbooks en parallèle ce qui nous permet d'éviter la perte du temps.


4.2.2 Enquête

Les enquêtes permettent de définir des variables supplémentaires pour les playbooks, mais sous une forme de question et de réponse conviviale.

Au cours de cette phase, nous avons ajouté une enquête pour renseigner la valeur de la variable « projet » comme le montre la figure 4.44.

Figure 4.44 : L'enquête de projet

Une fois que le modèle « projet » est exécuté, on nous demande d'entrer le nom du projet comme c'est indiqué dans la figure 4.45.



Prompts

1 Survey

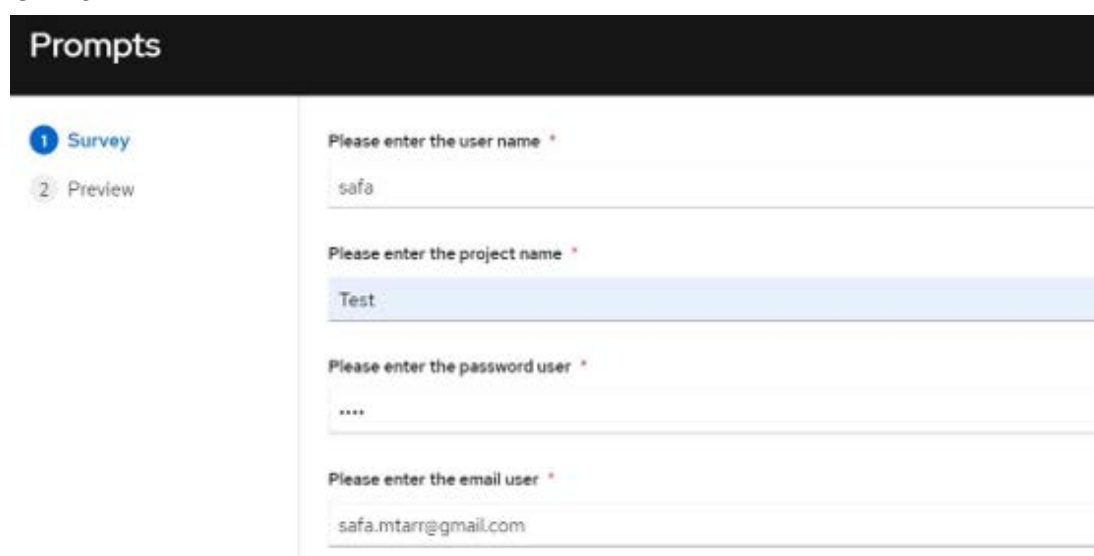
2 Preview

please enter the name of project *

Test

Figure 4.45 : Saisie de la variable "projet"

De même, pour le modèle « utilisateur » nous allons saisir les variables telles que le nom de l'utilisateur, l'email, le mot de passe et le nom du projet à attacher comme l'indique la figure 4.46.



Prompts

1 Survey

2 Preview

Please enter the user name *

safa

Please enter the project name *

Test

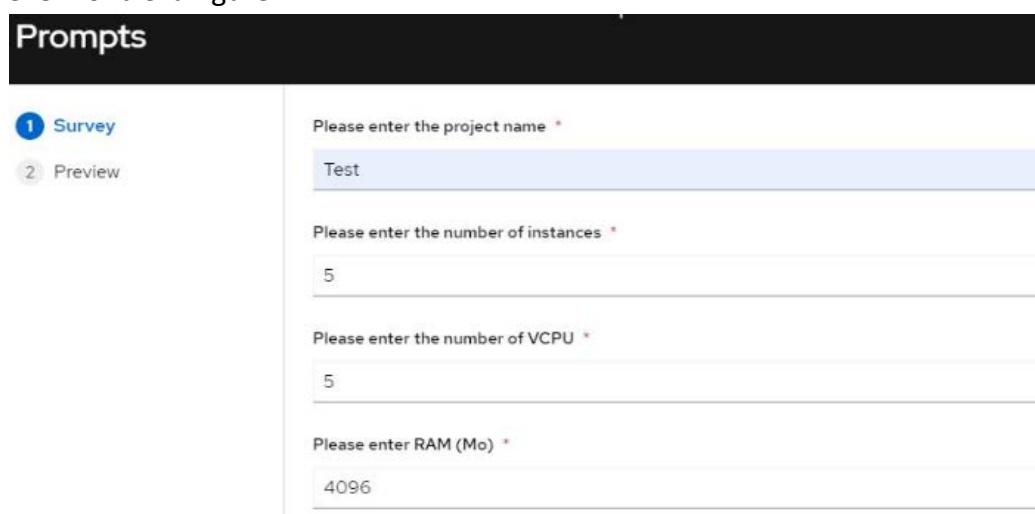
Please enter the password user *

Please enter the email user *

safa.mtarr@gmail.com

Figure 4.46 : Saisie des variables nécessaires pour créer un utilisateur

De plus, nous avons ajouté des enquêtes qui demandent les valeurs des variables telles que le nom de projet, le nombre d'instances et VCPU et le RAM, selon le besoin du client, comme le montre la figure 4.47.



Prompts

1 Survey

2 Preview

Please enter the project name *

Test

Please enter the number of instances *

5

Please enter the number of VCPU *

5

Please enter RAM (Mo) *

4096

Figure 4.47 : Saisie des variables de l'enquête "quotas"

Conclusion

Dans ce chapitre, nous avons détaillé les étapes nécessaires pour la mise en place de l'IaaS OpenStack. Nous avons également présenté la configuration de l'outil Ansible pour qu'il puisse automatiser les tâches répétitives d'OpenStack et l'interface graphique AWX pour orchestrer les playbooks Ansible.

Conclusion générale

Le Cloud Computing répond à un grand nombre de besoins des entreprises. L'investissement dans ce domaine attire donc de plus en plus les techniciens réseaux qui ne cessent de proposer des infrastructures du Cloud variées en qualité de service fiable et rentable.

Dans le cadre de stage de fin d'études, nous avons configuré un environnement Cloud sur OpenStack. En effet, nous avons testé les services : nova, keystone, horizon, neutron, cinder, glance et swift.

Keystone nous a permis d'authentifier des utilisateurs en leur accordant des accès individualisés.

Le service nova est responsable de la gestion de nos instances virtuelles en utilisant l'hyperviseur sans licence KVM.

Glance a mis à notre disposition les images des supports de données des machines virtuelles.

Via neutron, nous avons assuré la communication entre les instances au sein du projet et en dehors du Cloud. Nous avons également défini les règles de filtrage IP pour sécuriser les environnements Cloud.

Cinder nous a fourni un espace de stockage en bloc adapté aux besoins des utilisateurs. A l'instar de cinder, swift crée des espaces de stockage objet.

Horizon nous a offert une interface utilisateur graphique dans le but de gérer les composants regroupés dans OpenStack.

Toutefois, l'évolutivité d'OpenStack dépend de la rapidité de la livraison et de la qualité des services. C'est pour cela, les fournisseurs du Cloud multiplient les procédés destinés à concevoir des infrastructures automatisées pour accompagner la croissance des entreprises sans accroître significativement les effectifs IT.

Ansible, outil d'orchestration des ressources IT, nous a permis d'automatiser les tâches d'OpenStack. En effet, nous les avons traduites en des playbooks écrits en langage yaml.

Au-dessus du projet Ansible, nous avons ajouté une interface utilisateur : AWX. Cette dernière nous a permis de créer des pipelines de playbooks et de définir des extras variables afin de renforcer le moteur d'automatisation.

Notre travail peut être poursuivi de différentes manières. Ainsi, la principale perspective découverte est l'étude du mode HA d'OpenStack pour offrir un nuage plus fonctionnel et avec des nœuds multiples.

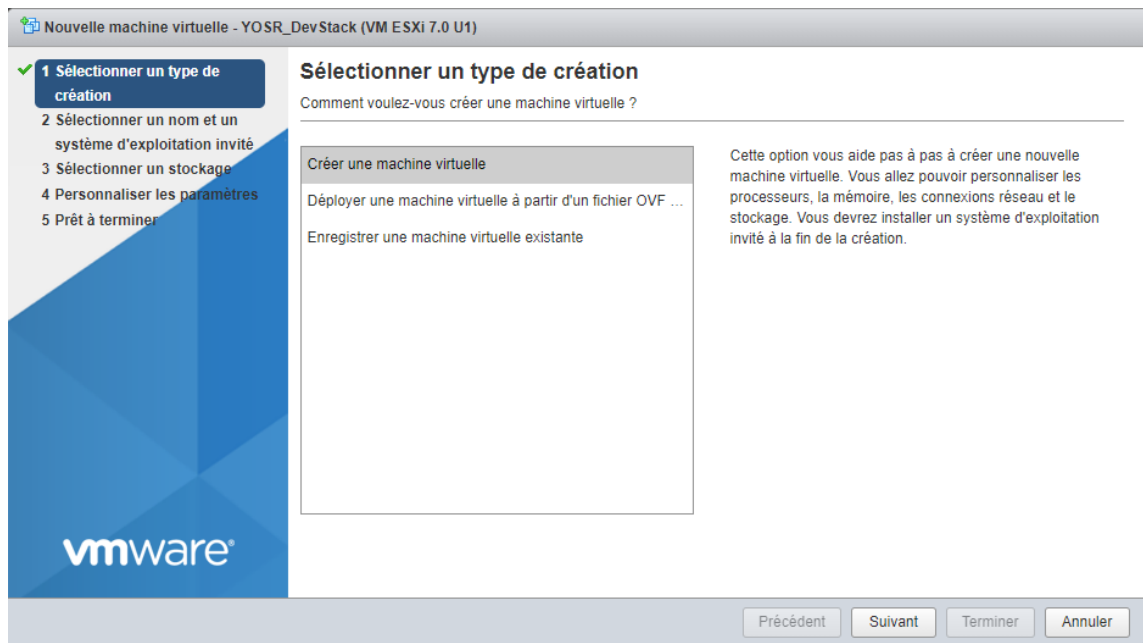
Bibliographie

- [1] « NextStep IT », [En ligne]. Disponible: <https://www.nextstep-it.com/> [consulté le 02/03/2021]
- [2] Abdelmajid, « Les différents types de cloud computing ». Advancia, 2019.
- [3] Red Hat, « Les services cloud, qu'est-ce que c'est ? ».
- [4] Digital Guide IONOS « Aperçu des concepts de virtualisation ». 2019.
- [5] Tech Target, « Automatisation du cloud ». LEMAGIT, 2019.
- [6] Emilie Ravet, « Automatisation et Cloud : le duo gagnant », 2018.
- [7] Red Hat, « Comprendre OpenStack ».
- [8] Shalmali Suhas Sahasrabudhe, Shilpa Sonawani, « Algorithme de pondération de filtre amélioré pour la planification des ressources tenant compte de l'utilisation dans OpenStack ». Conférence internationale sur le traitement de l'information (ICIP), Université William Paix, Décembre 2015.
- [9] « OpenStack », [En ligne]. Disponible: <https://docs.openstack.org> , [consulté le 18/04/2021].
- [10] Chris Houseknecht, « 5 Things You Can Do With AWX ». Red Hat Ansible, 2017.
- [11] Red Hat, « Apprendre les bases d'Ansible ».
- [12] « Online Diagram Software & Visual Solution | Lucidchart », [En ligne]. Disponible: <https://www.lucidchart.com/pages/fr> [consulté le 12/05/2021].
- [13] Remy Manu, « Cours n°5 : Diagramme de séquences ». PDF, <http://remy-manu.no-ip.biz/UML/Cours/coursUML5.pdf> .
- [14] Pierre Gaxatte, « Créer et utiliser des instantanés OpenStack ». OVHcloud, 2020.

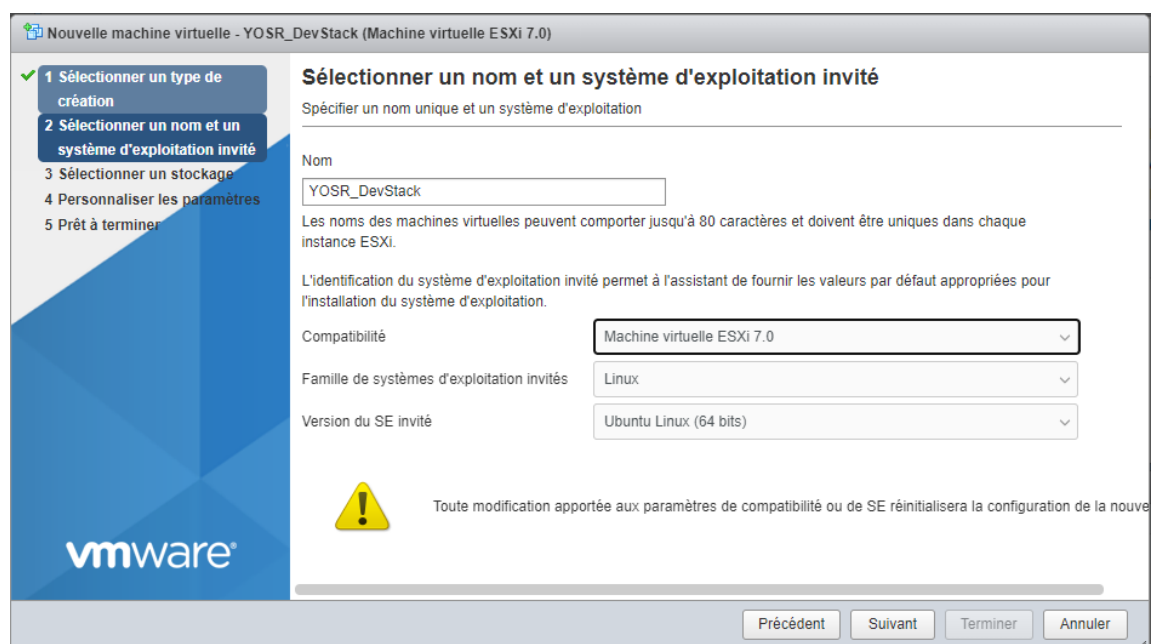
Annexe 1 : Création d'une machine virtuelle sur VMware ESXi

Toutes les machines virtuelles utilisées pour la réalisation du projet sont créées sur l'hyperviseur VMware ESXi de l'entreprise. Pour ce faire, nous avons configuré le matériel virtuel, y compris les processeurs, les disques durs et la mémoire comme suit :

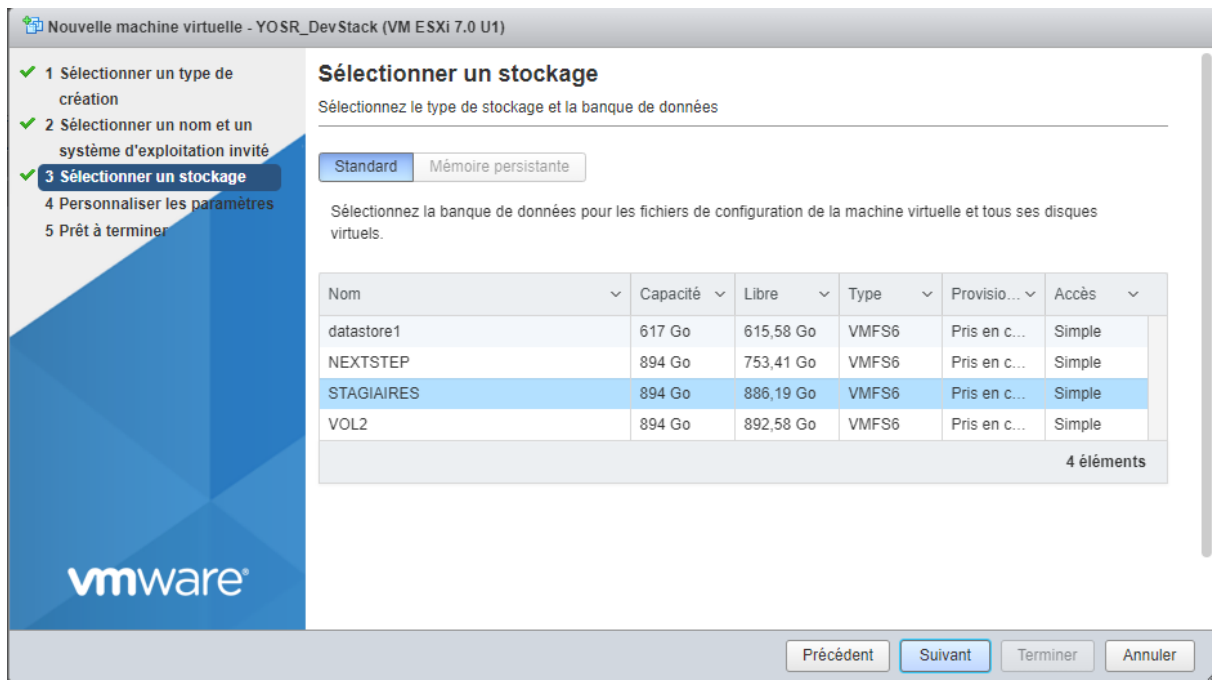
Tout d'abord, nous avons sélectionné le type de création qui est une machine virtuelle sans modèle ou clone, comme l'indique la figure ci-dessous.



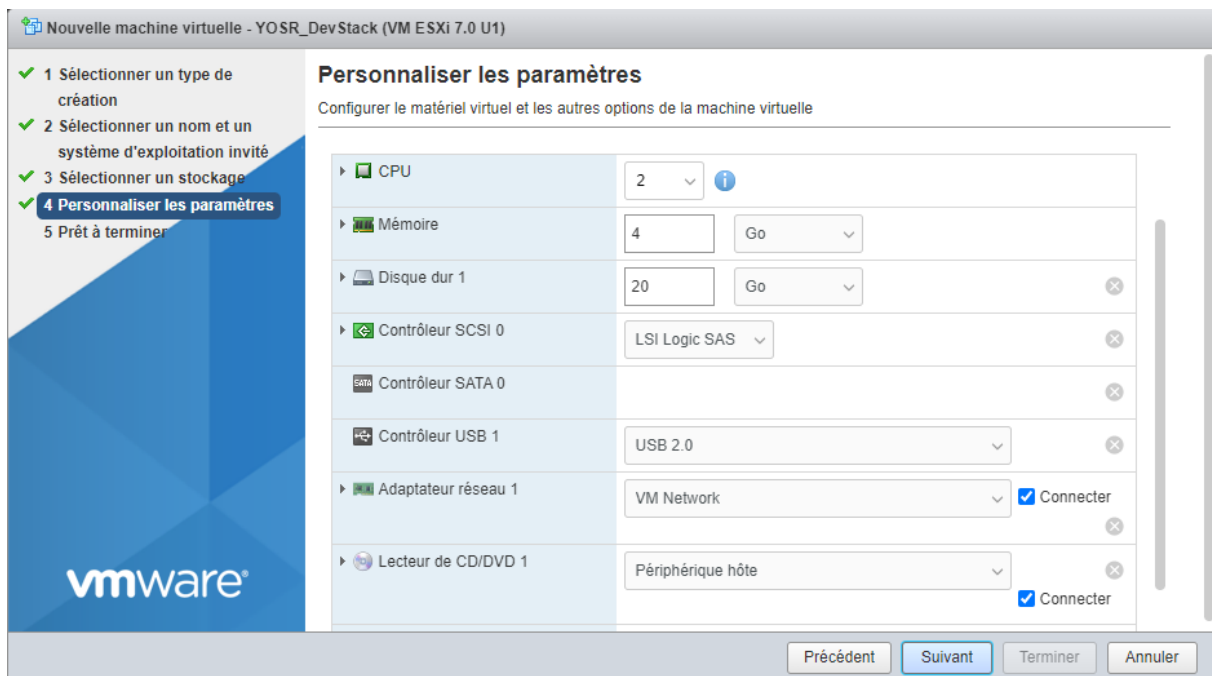
Ensuite, nous avons entré un nom unique pour la VM et nous avons sélectionné le système d'exploitation invité adéquat comme le montre la figure suivante :



Puis, sur la page sélectionner un stockage, nous avons sélectionné la banque de données « STAGIAIRES » où nous allons stocker les fichiers de la machine virtuelle comme le représente la figure ci-dessous.



Enfin, nous avons configuré le matériel et les options de la machine virtuelle y compris la mémoire, le processeur et le disque dur comme l'indique la figure ci-dessous.



Dès que nous cliquons sur terminer, La VM apparaît dans l'inventaire de vSphere Client.

Annexe 2 : Installation d'OpenStack

Pour le déploiement d'Openstack, nous avons utilisé les deux installateurs : Devstack et Packstack.

1. Installation d'OpenStack avec Devstack

Devstack est un ensemble de scripts extensibles facilitant le déploiement d'OpenStack.

1.1 Exigences minimales

Pour installer OpenStack en utilisant l'installateur Devstack, nous avons besoin d'un système Ubuntu 18.04, 12 Go de RAM, 6 VCPUs et un espace de stockage de 20 Go.

1.2 Mise à jour du système

Nous avons mis à niveau les référentiels systèmes à l'aide de la commande suivante :

```
$ sudo apt update -y && apt upgrade -y
```

1.3 Création d'un utilisateur pile (Stack) et attribution des privilèges

Les meilleures pratiques imposent l'exécution de Devstack en tant qu'utilisateur régulier avec les droits sudo. Dans cet esprit, nous avons créé un utilisateur appelé « stack » en exécutant la commande ci-dessous :

```
$ sudo adduser -s /bin/bash -d /opt/stack -m stack
```

Ensuite, nous avons attribué les privilèges sudo à l'utilisateur « stack » grâce à cette commande :

```
$ sudo echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

1.4 Clonage du dépôt git de Devstack

Une fois que nous avons créé avec succès l'utilisateur, nous avons cloné le dépôt git de Devstack comme indiqué :

```
$ git clone https://git.openstack.org/openstack-dev/devstack
```

1.5 Création d'un fichier de configuration Devstack

Lors de cette étape, nous avons accédé au répertoire « devstack » et créé un fichier de configuration appelé « local.conf ».

Ce dernier contient les informations indiquées dans la figure ci-dessous.

```
[[local|localrc]]  
  
# Password for Keystone, Database, RabbitMQ and Service  
ADMIN_PASSWORD=devstack  
DATABASE_PASSWORD=$ADMIN_PASSWORD  
RABBIT_PASSWORD=$ADMIN_PASSWORD  
SERVICE_PASSWORD=$ADMIN_PASSWORD  
  
# Host IP - get your Server/VM IP address from ip addr command  
HOST_IP=172.25.52.123
```

1.6 Exécution du script « stack »

Pour commencer le déploiement d'OpenStack sur Ubuntu 18.04, nous avons exécuté le script « stack » dans le répertoire « devstack », en utilisant la commande suivante :

```
$ ./stack.sh
```

Ce script permet l'installation de ces services :

- Tableau de bord OpenStack - horizon
- Service de calcul - nova
- Service d'image - glance
- Service réseau - neutron
- Service d'identité - neystone
- Placement - API de placement

La sortie de la commande ci-dessus est la suivante :

```
This is your host IP address: 172.25.52.123  
This is your host IPv6 address: ::1  
Horizon is now available at http://172.25.52.123/dashboard  
Keystone is serving at http://172.25.52.123/identity/  
The default users are: admin and demo  
The password: devstack  
  
Services are running under systemd unit files.  
For more information see:  
https://docs.openstack.org/devstack/latest/systemd.html  
  
DevStack Version: wallaby  
Change: ff895cc787cd58ed20e4e281cb2e7484ec42f8db Merge "Add a variable to confi  
gure the Tempest venv upper constraints" 2021-03-09 21:50:25 +0000  
OS Version: Ubuntu 18.04 bionic  
  
2021-03-11 17:17:48.424 | stack.sh completed in 1174 seconds.
```

Cela confirme que tout s'est bien configuré et que nous pouvons accéder au tableau de bord horizon via l'URL : <http://172.25.52.123/dashboard>.

2. Installation d'OpenStack avec Packstack

Packstack est un utilitaire de ligne de commande qui utilise des modules puppet pour déployer OpenStack.

2.1 Prérequis

Pour cette installation, nous utilisons une machine virtuelle ESXI 7.0 dont ses caractéristiques sont

- 16 Go de RAM,
- 60 Go de disque dur
- 6 VCPUs.

2.2 Configuration du réseau

D'abord, nous avons désactivé le service FirewallD pour démarrer automatiquement au démarrage du système :

```
$ sudo systemctl disable firewalld
```

Suite à l'exécution de la commande ci-dessus, nous avons obtenu la sortie suivante :

```
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.  
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
```

Ensuite, nous avons arrêté le service FirewallD avec :

```
$ sudo systemctl stop firewalld
```

Puis, nous avons désactivé le gestionnaire de réseau (définitivement) pour démarrer automatiquement lors du démarrage du système en utilisant la commande suivante :

```
$ sudo systemctl disable NetworkManager
```

La sortie de cette commande est :

```
Removed symlink /etc/systemd/system/multi-user.target.wants/NetworkManager.service.  
Removed symlink /etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service.  
Removed symlink /etc/systemd/system/network-online.target.wants/NetworkManager-wait-online.service.
```

Ensuite, nous avons arrêté le service de gestionnaire de réseau avec :

```
$ sudo systemctl stop NetworkManager
```

Lors du redémarrage du système, nous avons activé le service gestionnaire de réseau avec :

```
$ sudo systemctl enable network
```

Enfin, nous avons démarré le serveur réseau avec :

```
$ sudo systemctl start network
```

1.4 Mise à jour du système

Dès que nous avons redémarré notre système et rétabli la connexion Internet, nous avons exécuté la commande suivante :

```
$ sudo yum update -y
```

2.4 Installation des référentiels de logiciels

Nous avons installé le package Rocky pour activer le référentiel OpenStack avec la commande ci-dessous.

```
$ sudo yum install -y centos-release-openstack-rocky
```

2.5 Installation du programme Packstack

Une fois l'installation du package rocky est terminée, nous avons installé Packstack avec :

```
$ sudo yum install -y openstack-packstack
```

2.6 Exécution du Packstack

Pour déployer OpenStack avec un seul nœud, nous avons exécuté la commande suivante :

```
$ sudo packstack --allinone
```

Nos paramètres de connexion sont stockés dans le fichier « keystone_admin » comme le montre la figure ci-dessous.

```
[root@localhost ~]# cat keystone_admin
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD='b6bb4b7857fa4119'
export OS_REGION_NAME=RegionOne
export OS_AUTH_URL=http://172.25.52.127:5000/v3
export PS1='\u@\h \W(keystone_admin)]\$ '

export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
[root@localhost ~]#
```

Une fois le processus d'installation est terminé, nous pouvons accéder au tableau de bord OpenStack en accédant à « <http://172.25.52.127/dashboard> ».

← → ↻ ⚠ Non sécurisé | 172.25.52.127/dashboard/auth/login/?next=/dashboard/ 🔑 ☆ ⚙️ 👤 ⋮


openstack®

Se connecter

Nom d'utilisateur

Mot de passe

Se connecter

Annexe 3 : Implémentation d'AWX

Nous avons installé AWX sur le nœud de contrôle Ansible.

Tout d'abord, nous avons mis à jour le système Ubuntu en utilisant la commande suivante :

```
$ sudo apt update && sudo apt -y upgrade  
$ sudo reboot
```

Ensuite, les services Ansible AWX s'exécutent en tant que conteneurs Docker. C'est pour cela que nous avons besoin de Docker Engine pour alimenter l'environnement. La commande ci-dessous installe les dépendances de base permettant la configuration des packages Docker.

```
$ sudo apt -y install apt-transport-https ca-certificates curl software-properties-common
```

Puis, nous avons ajouté la clé GPG du référentiel Docker à notre système et aux sources APT en tapant les commandes suivantes :

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu  
$(lsb_release -cs) stable "
```

Par la suite, nous avons installé Docker CE grâce à cette commande :

```
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

```
$ sudo usermod -aG docker ${USER}
```

De plus, pour que nous évitions de taper « sudo » chaque fois que nous exécutons une commande Docker, nous avons ajouté notre utilisateur au groupe Docker :

Une fois que nous avons vérifié l'installation de Docker Engine, nous avons installé Node.js, NPM, le module python docker-py et docker-compose en exécutant les commandes ci-dessous :

```
$ sudo apt install -y nodejs npm  
$ sudo npm install npm --global  
$ sudo apt install python-pip git pwgen vim  
$ sudo pip install requests==2.14.2  
$ sudo pip3 install docker-compose == 1.29.1
```

Maintenant, nous pouvons installer AWX en clonant le code source à partir de GitHub à l'aide de la commande « git ».

```
$ sudo git clone -b 17.1.0 https://github.com/ansible/awx.git
```


Après, nous avons accédé au répertoire du programme d'installation AWX et nous avons modifié le fichier inventaire en indiquant les paramètres d'accès à l'interface graphique et la clé secrète générée par la commande « pwgen ».

```
# then these default values are used
admin_user=admin
admin_password=root

# Whether or not to create preload data for demonstration purposes
create_preload_data=True

# AWX Secret key
# It's *very* important that this stay the same between upgrades or you will lose
# the ability to decrypt
# your credentials
secret_key= 3na9BcFDmQYHRi24fstNZ0ii330ZV7
```

Enfin, nous avons exécuté le palybook « install.yml » qui permet d'installer AWX et nous avons listé les conteneurs en cours d'exécution en utilisant la commande « docker ps ».

```
root@awx-virtual-machine:~/awx/installer# docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED        STATUS
US            PORTS                                NAMES
0b472cb195b7   ansible/awx:17.1.0                 "/usr/bin/tini -- /u..."             46 seconds ago Up 4
0 seconds     8052/tcp                             awx task
04ca2c2d03d6   ansible/awx:17.1.0                 "/usr/bin/tini -- /b..."             54 seconds ago Up 4
3 seconds     0.0.0.0:80->8052/tcp                 awx web
c345189faa0a   redis                               "docker-entrypoint.s..."             56 seconds ago Up 4
2 seconds     6379/tcp                             awx_redis
4069d87ccf9c   postgres:12                         "docker-entrypoint.s..."             56 seconds ago Up 4
1 seconds     5432/tcp                             awx_postgres
```

Nous pouvons alors accéder au tableau de bord via l'adresse IP suivante :

- [Http:// 172.25.52.134/#/awx](http://172.25.52.134/#/awx)

Le tableau de bord présente des informations sur notre serveur AWX qui sont :

- Les hôtes qui ont exécuté avec succès les playbooks
- Les hôtes qui n'ont pas réussi à exécuter les playbooks
- Les inventaires
- Les projets et état de la synchronisation

Annexe 4 : Ajout des enquêtes d'AWX

Dans AWX, une enquête est un ensemble de questions que vous pouvez configurer pour poser avant d'exécuter le script et stocker ses réponses sous forme de variables. Les étapes de configuration peuvent être résumées comme suit :

1. Ajout d'une enquête pour le modèle « utilisateur »

Une enquête peut comprendre n'importe quel nombre de questions. L'enquête spécifiée au modèle « utilisateur » se compose de quatre questions.

La figure ci-dessus représente la première question permettant la saisie du nom d'utilisateur.

◀ Back to Templates Details Access Notifications Schedules Completed Jobs Survey

Question *	Description	Answer variable name * ?
<input type="text" value="Please enter the user name"/>	<input type="text"/>	<input type="text" value="member"/>
Answer type * ?	<input checked="" type="checkbox"/> Required	
<input type="text" value="Text"/>		
Minimum length	Maximum length	Default answer
<input type="text" value="0"/>	<input type="text" value="1024"/>	<input type="text"/>

La deuxième question qui demande la saisie de nom du projet, est présentée dans la figure ci-dessous.

Question *	Description	Answer variable name * ?
<input type="text" value="Please enter the project name"/>	<input type="text"/>	<input type="text" value="project"/>
Answer type * ?	<input checked="" type="checkbox"/> Required	
<input type="text" value="Text"/>		
Minimum length	Maximum length	Default answer
<input type="text" value="0"/>	<input type="text" value="1024"/>	<input type="text"/>

La troisième question permet de saisir le mot de passe de l'utilisateur OpenStack comme indiqué dans cette figure.

The screenshot shows the configuration interface for a new question in AWX. The form is divided into several sections:

- Question:** A text input field containing "Please enter the password user".
- Description:** An empty text input field.
- Answer variable name:** A text input field containing "password".
- Answer type:** A dropdown menu set to "Password".
- Required:** A checkbox labeled "Required" which is checked.
- Minimum length:** A text input field containing "0".
- Maximum length:** A text input field containing "50".
- Default answer:** A text input field with a clear icon (X) and an empty field.
- Buttons:** "Save" (blue) and "Cancel" (white with blue border) buttons at the bottom left.

Le type de la réponse est password. Elle sera cryptée et traitée comme des informations sensibles.

La dernière question relève de l'email de l'utilisateur comme le montre la figure ci-dessous.

The screenshot shows the configuration interface for a new question in AWX, similar to the previous one but for an email question. The form is divided into several sections:

- Question:** A text input field containing "Please enter the email user".
- Description:** An empty text input field.
- Answer variable name:** A text input field containing "email".
- Answer type:** A dropdown menu set to "Text".
- Required:** A checkbox labeled "Required" which is checked.
- Minimum length:** A text input field containing "0".
- Maximum length:** A text input field containing "1024".
- Default answer:** An empty text input field.
- Buttons:** "Save" (blue) and "Cancel" (white with blue border) buttons at the bottom left.

La réponse de l'utilisateur à la question ci-dessus sera stockée dans la variable « email ». Il s'agit de la variable à utiliser par le playbook « utilisateur » qui est hébergé dans le référentiel git : <https://github.com/yosr-ghazouani/ansibletest>.

2. Ajout d'une enquête pour le modèle « quotas »

Cette enquête se compose de quatre questions qui permettent de spécifier les caractéristiques de l'environnement Cloud y compris RAM, VCPUs et l'espace de stockage comme l'indiquent la figure ci-dessous.

Question *	Description	Answer variable name *
Please enter the project name		project
Answer type *	Minimum length	Maximum length
Text	0	1024
Question *	Description	Answer variable name *
Please enter RAM (Mo)		ram
Answer type *	Minimum	Maximum
Integer	1	51200
Question *	Description	Answer variable name *
Please enter the storage of volumes (Go)		storage
Answer type *	Minimum	Maximum
Integer	1	1000
Question *	Description	Answer variable name *
Please enter the storage of volumes (Go)		storage
Answer type *	Minimum	Maximum
Integer	1	1000