

UNIVERSITÉ MOHAMMED V DE RABAT
Faculté des Sciences



Département d'Informatique
Filière Licence Fondamentale
en Sciences Mathématiques et Informatique

PROJET DE FIN D'ÉTUDES

Intitulé :

Mise en place d'Ansible et automation des
configurations

Présenté par :
ATBIR RIHAB
HAOUDI MARWA

soutenu le 15 Juin 2024 devant le Jury

M. Latifa Oufkir	Professeur à la Faculté des Sciences - Rabat	<i>Encadrante</i>
M. Lamyae Sardi	Professeur à la Faculté des Sciences - Rabat	<i>Examineur</i>

Année Universitaire 2023-2024

Remerciements

Avant tout développement sur cette expérience académique, il apparaît opportun de commencer ce rapport de projet de fin d'étude par des remerciements, à ceux qui nous ont beaucoup appris au cours de ce projet, et même à ceux qui ont eu la gentillesse de faire de ce PFE un moment très profitable.

Nous tenons à exprimer nos sincères remerciements, dans un premier temps, à notre encadrante Mme Latifa Oufkir pour son accompagnement tout au long du projet ainsi que pour le temps et l'attention qu'elle a consacrés à nos questions et à nos préoccupations. Nous exprimons également notre profonde gratitude à toute l'équipe pédagogique de notre faculté des sciences ; département informatique qui a contribué à notre réussite pendant cette année.

Nous adressons aussi nos vifs remerciements aux membres des jurys pour avoir bien voulu examiner et juger notre travail.

Enfin, nous tenons à exprimer notre gratitude envers nos familles et nos amis pour leur soutien inconditionnel tout au long de cette expérience.

À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

Résumé

De nos jours, il est temps d'apprendre à automatiser l'infrastructure des réseaux. C'est un changement important dans la façon dont les experts pensent des réseaux, de la conception à la création et à la gestion. En tant qu'étudiants, nous n'aurions pas pu choisir un meilleur moment pour explorer l'automatisation.

Au fur et à mesure que les réseaux s'étendent, le nombre de périphériques doit être dissocié du nombre de personnes.

En mettant en œuvre l'automatisation, non seulement les erreurs sont moins probables et les temps d'arrêt sont réduits, mais certaines tâches peuvent être effectuées plus rapidement et plus efficacement. Dans l'ensemble, ce projet constitue une bonne base pour démontrer que ce style d'automatisation est réalisable et laisse les possibilités ouvertes pour de nouvelles idées à automatiser à l'avenir.

Abstract

In today's fast-evolving technological landscape, network automation has become a critical aspect of infrastructure management. This project focuses on leveraging Ansible, an open-source automation tool, to streamline network configurations and management tasks. By automating ACLs, SNMP configurations, logging, and OSPF routing on routers, as well as VLAN naming on switches, we aim to enhance efficiency, reduce errors, and minimize downtime. This project not only showcases the practical implementation of Ansible in a network environment but also highlights its potential in solving common network management challenges.

Keywords :

Table des matières

Remerciements	i
Résumé	ii
Abstract	iii
Introduction	vii
1 Cadre général du projet et planification	1
1.1 Cadre général du projet :	1
1.1.1 Contexte du projet :	1
1.1.2 Problématique :	1
1.1.3 Objectifs du projet :	1
1.1.4 Démarche :	2
2 Étude comparative des différentes solutions	3
2.1 Benchmarking des outils disponibles :	3
2.1.1 Ansible :	3
2.1.2 Chef :	4
2.1.3 Puppet :	4
2.1.4 Choix des critères :	4
2.1.5 Comparaison et synthèse :	5
2.1.6 Pourquoi Ansible ?	5
2.2 Présentation générale de Ansible :	6
2.2.1 Présentation générale :	6
2.2.2 Le terme Ansible :	7
2.2.3 Notions et Concepts	7
2.2.4 Progression de Ansible avec le Réseau	7
3 : Etude théorique d’Ansible	9
3.1 Composants principaux de Ansible	9
3.1.1 Architecture de Ansible :	9
3.2 Composants supplémentaires :	11
3.2.1 Modèles Jinja2 :	11
3.2.2 Ansible Vault :	11
3.2.3 Mode ad-hoc :	11
3.2.4 Langage YAML :	11

4	Virtualisation	13
4.1	Introduction à la virtualisation :	13
4.1.1	Les types principaux de virtualisation :	13
4.1.2	Les avantages :	14
4.2	Mise en place de l'environnement virtuel :	14
4.2.1	VMware :	14
4.2.2	Avantages de VMware :	15
4.3	Mise en place d'un simulateur :	15
4.3.1	Présentation EVE-NG (Emulated Virtual Environment - Next Generation) :	15
4.3.2	Avantages de EVE-ng :	16
4.3.3	Présentation Panetlab :	16
4.3.4	Avantages de Panetlab :	16
4.4	Conclusion :	16
5	Environnement de travail	18
5.1	Installation des Packages APT :	18
5.2	Installation d'Ansible sous Ubuntu :	19
5.3	Installation d'EVN-EG dans VMware	21
5.4	Installation d'une image d'appareil réseau dans EVE-NG :	24
6	Mise en place de la solution et automatisation	28
6.1	La topologie réseau adoptée	28
6.2	Configuration de l'accès SSH	28
6.3	Configuration de mappage sur <code>inventory.ini</code>	29
6.3.1	Connexion à distance via SSH	29
6.4	Configuration d'ansible et tests	31
6.4.1	Playbook : Obtention des informations sur le switch	31
6.4.2	Playbook : Configuration des VLANs sur le switch	33
6.4.3	Playbook : activation d'ospf sur un routeur cisco ios	37
6.4.4	Playbook pour la Configuration des logs	38
6.4.5	Playbook pour configurer une ACL pour filtrer le trafic sortant sur l'interface wan	39
6.4.6	Playbook Configuration snmp	40
6.4.7	Playbook pour la vérification de l'état du routeur	41
6.4.8	Playbook pour la vérification de ping	43
6.5	Déploiement de GIT avec GitHub	43
6.5.1	Introduction	44
6.5.2	Prérequis pour le Déploiement	44
6.5.3	Installation et Configuration de Git	45
6.5.4	Configuration initiale de Git :	46
6.6	Création et gestion des workflows GitHub Actions	47
6.6.1	Présentation de GitHub Actions	47
6.6.2	Avantages et cas d'utilisation	47
6.6.3	Intégration de GitHub Actions et Molecule pour le CI/CD	49
6.6.4	Structure des Répertoires	49

6.6.5	Fichier ci-cd.yml de GitHub Actions	49
6.6.6	Fichier molecule.yml de Molecule	52
Conclusion		54

Introduction

Dans le cadre de notre parcours universitaire en informatique, Le projet réalisé s'est avéré très intéressant et très enrichissant pour notre expérience académique et professionnelle.

En effet, ce sujet complète le thème de nos études puisqu'elle se déroule à propos de l'automatisation des configurations.

Dans ce rapport, je vais tout d'abord présenter le cadre général du projet et étude comparative des différentes solutions, après étude théorique d'Ansible et préparation d'environnement de travail et enfin la mise en place de la solution et automatisation.

Chapitre 1

Cadre général du projet et planification

Dans ce chapitre, nous allons présenter notre projet de fin d'études. Nous allons nous pencher sur les différents aspects. Nous allons également décrire le contexte dans lequel notre projet s'inscrit et mettre en évidence les principales caractéristiques de l'environnement de travail dans lequel nous avons évolué.

1.1 Cadre général du projet :

1.1.1 Contexte du projet :

Le contexte du projet d'automatisation de configuration est ancré dans la nécessité croissante des entreprises de simplifier et d'accélérer le déploiement ainsi que la gestion de leurs infrastructures informatiques. Dans un environnement où la demande de flexibilité et de rapidité est primordiale, les méthodes traditionnelles de configuration manuelle deviennent inefficaces et coûteuses.

Ainsi, l'automatisation de la configuration émerge comme une solution incontournable pour répondre à ces défis, en permettant aux entreprises d'automatiser et de standardiser les processus de déploiement, de configuration et de gestion des ressources informatiques.

1.1.2 Problématique :

La problématique principale que ce sujet vise à résoudre est l'automatisation et la gestion efficace des configurations réseau qui relèvent du domaine NetDevOps pour un environnement composé de routeurs, de commutateurs et de serveurs.

La configuration manuelle de ces équipements est une tâche sujette aux erreurs, surtout lorsque le nombre d'équipements augmente. De plus, la diversité des technologies et des plateformes utilisées complique davantage la gestion et la coordination des configurations. L'automatisation des configurations réseau à l'aide d'un outil comme Ansible permet de rationaliser et de standardiser ce processus, réduisant ainsi les risques d'erreurs humaines.

1.1.3 Objectifs du projet :

Les objectifs du projet sont les suivants :

1. Réduire les délais de déploiement : L'automatisation accélère le déploiement des infrastructures en éliminant les étapes manuelles, réduisant ainsi les délais de mise en service.
2. Améliorer la cohérence et la fiabilité : Les scripts automatisés garantissent des configurations cohérentes et fiables, éliminant les erreurs humaines.
3. Accroître la flexibilité et l'adaptabilité : L'automatisation permet des mises à jour rapides, le scaling des infrastructures et l'intégration de nouvelles technologies sans perturbations.
4. Renforcer la sécurité : L'automatisation applique des standards de configuration sécurisés et assure une surveillance constante de la conformité.
5. Optimiser les ressources : En réduisant les tâches manuelles, l'automatisation optimise l'utilisation des ressources humaines et matérielles, réduisant les coûts opérationnels.

1.1.4 Démarche :

La démarche pour atteindre ces objectifs implique plusieurs étapes :

1. Proposition et étude de la solution Ansible
2. Étude des technologies utilisées, y compris les IOS (switch et routeur)
3. Installation d'Eve-ng et création du lab
4. Mise en place de la solution Ansible et déploiement
5. Installation d'Ansible sur Ubuntu

Chapitre 2

Étude comparative des différentes solutions

Introduction :

Dans ce chapitre, nous examinerons plusieurs solutions d'automatisation de configuration afin de déterminer laquelle répond le mieux aux besoins spécifiques de notre projet. Cette étude comparative est essentielle pour prendre une décision éclairée et choisir la solution la plus adaptée à nos exigences en termes de flexibilité, de performance et de facilité d'utilisation.

2.1 Benchmarking des outils disponibles :

2.1.1 Ansible :

- a Qu'est-ce qu'Ansible ? Ansible est une plateforme d'automatisation de configuration open-source qui se distingue par sa simplicité et sa facilité de déploiement. Il utilise une architecture sans agent et fonctionne sur SSH, ce qui le rend particulièrement adapté aux environnements hétérogènes. Ansible offre une syntaxe YAML claire et facile à comprendre pour décrire les tâches à exécuter, ce qui le rend accessible même aux débutants.
- b Avantages d'Ansible :
 - Automatisation simplifiée : Ansible est simple d'utilisation, facile à installer, à configurer et à maîtriser. En moins de 30 minutes, il est possible d'installer et de configurer le système et d'exécuter des commandes ad hoc pour les serveurs pour résoudre un problème spécifique.
 - Courbe d'apprentissage basse : Ansible est facile à déployer car il n'utilise aucun agent ni infrastructure de sécurité personnalisée supplémentaire. Il s'appuie également sur YAML, un langage simple pour décrire votre travail d'automatisation via les playbooks.
 - Automatisation immédiate : Dès que vous pouvez envoyer une requête ping aux hôtes via Ansible, vous pouvez commencer à automatiser votre environnement. Commencez par de petites tâches, suivez les bonnes pratiques, hiérarchisez

les tâches sources de valeur pour l'entreprise, résolvez des problèmes majeurs, gagnez du temps et améliorez la productivité.

2.1.2 Chef :

Créé en 2009, Chef est un logiciel de gestion de configurations pour le développement informatique. Il permet d'automatiser des infrastructures serveur comme les systèmes d'exploitation, à partir de "recettes". Elles entrent en ligne de compte pour le packaging et le provisioning. Il utilise un modèle client-serveur avec un agent installé sur les nœuds à gérer. Chef propose une grande flexibilité grâce à son langage de configuration DSL (Domain Specific Language), mais peut être plus complexe à mettre en place et à gérer que d'autres solutions.

2.1.3 Puppet :

Conçu comme un projet open-source et développé en langage Ruby, Puppet est un outil de gestion de configuration et d'orchestration de serveurs. On l'utilise pour automatiser la configuration et la gestion de systèmes informatiques, en assurer la cohérence et réduire les erreurs.

Il est notamment exploité pour installer et configurer des logiciels, gérer des utilisateurs et des groupes, configurer des services réseau, surveiller l'état des systèmes et bien plus encore.

Un cas d'usage courant est la gestion de configuration de grands parcs de serveurs. En effet, Puppet permet de déployer des configurations cohérentes à grande échelle et garantit que les systèmes répondent aux normes de l'entreprise.

Au fil des années, Puppet est devenu très populaire auprès des administrateurs système et des entreprises de toutes tailles. L'entreprise Puppet Labs (désormais appelée Puppet Inc) fondée par Kanies pour son développement a levé 5 millions de dollars en 2011 en financement de série B.

L'outil a évolué pour s'enrichir de fonctionnalités d'orchestration, permettant de gérer des tâches à travers plusieurs serveurs. Il est disponible en version open source et en version entreprise.

2.1.4 Choix des critères :

Dans la catégorie des outils de gestion de configuration, trois outils sont les plus connus et utilisés par les entreprises : Ansible, Puppet et Chef.

Les critères qui vont orienter notre choix d'outil de gestion de configuration et d'automatisation sont :

- Sans Agent : Pouvoir contrôler des machines sans configuration a priori ou installation d'agent.
- Langage
- Dépendance client : les logiciels prérequis pour configurer une machine.
- Mécanisme : de partage de configuration, push ou pull.
- Installation : Complexité de mettre en place l'outil pour un environnement de production.

2.1.5 Comparaison et synthèse :

Dans cette section, nous allons comparer les performances des trois outils d'automatisation de la gestion des configurations - Ansible, Chef et Puppet - en fonction des critères définis précédemment. Nous évaluerons également leur adéquation par rapport à nos besoins spécifiques en matière d'automatisation de la gestion des configurations.




<div>Critère \ Outil</div>	 ANSIBLE	 CHEF	 puppet
Sans Agent	Oui	Non	Non
Langage	Python	Ruby	Ruby
Dépendance client	Python, sshd, bash	Ruby, sshd, bash	Ruby
Mécanisme	Push	Pull	Pull
Approche	Procédural	Procédural	Déclarative
Installation	Facile	Peu facile	Difficile
Contributeur	3432	522	492

FIGURE 2.1.1 – Description de l'image

- CHEF et Puppet ont une courbe d'apprentissage légèrement plus abrupte en raison de leur approche basée sur l'état désiré du système et de la nécessité d'installer des agents sur les nœuds cibles. Cependant, ils offrent un contrôle plus précis sur les configurations.
- CHEF et Puppet offrent également de bonnes performances, bien que leur utilisation d'agents sur les nœuds cibles puisse entraîner une légère surcharge réseau et des temps de déploiement légèrement plus longs.
- CHEF et Puppet offrent également une grande flexibilité, mais peuvent être plus complexes à configurer pour certains cas d'utilisation spécifiques en raison de leur approche plus axée sur l'état.
- CHEF et Puppet nécessitent l'installation d'agents sur chaque nœud, ce qui peut poser des défis de gestion à grande échelle.
- CHEF et Puppet nécessitent une configuration et une gestion plus complexes, ce qui peut augmenter les efforts de maintenance, en particulier dans les environnements distribués.
- CHEF et Puppet ont également des communautés actives, bien que légèrement moins étendues que celle d'Ansible.

2.1.6 Pourquoi Ansible ?

Après une analyse approfondie des différentes solutions d'automatisation de la gestion des configurations disponibles, nous avons conclu que Ansible est la solution la plus

adaptée à nos besoins.

L’avantage d’Ansible par rapport aux autres outils de gestion de configuration est son aspect sans agent, c’est-à-dire, il n’a pas besoin d’une configuration a priori pour contrôler une machine. Donc, Ansible reste le plus adapté et facile à mettre en œuvre en comparaison avec Chef et Puppet.

Conclusion :

En conclusion, Ansible offre une combinaison unique de simplicité, de flexibilité, de performance et de support communautaire qui en fait la solution idéale pour automatiser la gestion des configurations dans notre environnement. Son déploiement facile, son architecture sans agent et sa grande adaptabilité en font un choix solide qui répond à nos besoins présents et futurs en matière d’automatisation.

2.2 Présentation générale de Ansible :

2.2.1 Présentation générale :

Ansible est un outil open-source de provisionnement de logiciels, de gestion des configurations et de déploiement d’applications qui permet de coder une infrastructure IT, y compris le support de ses applications. Il permet d’automatiser la plupart des tâches de gestion d’infrastructures. Il fonctionne sur de nombreux systèmes de type Unix et peut configurer aussi bien des systèmes de type Unix que Microsoft Windows ou autres. Il comprend son propre langage déclaratif pour décrire la configuration du système. Ansible est sans agent, se connectant temporairement à distance via SSH ou Windows Remote Management (permettant l’exécution à distance de PowerShell) par exemple pour effectuer ses tâches.

Ansible gère les différents noeuds avec un accès à distance natif (tels que les protocoles SSH ou Remote PowerShell ou encore des APIs natives, et bien d’autres). Il ne nécessite l’installation d’aucun logiciel supplémentaire à distance. Il offre des capacités de parallélisation, de collecte de métadonnées et de gestion des états. Cet aspect de conception “sans agent” installé sur le périphérique est important car il réduit les besoins sous-jacents d’infrastructure pour démarrer une gestion. Les modules fonctionnent grâce à JSON et à la sortie standard et ils peuvent être écrits dans n’importe quel langage de programmation. Ansible utilise notamment YAML pour exprimer des descriptions réutilisables de systèmes, il fournit des sorties en JSON, il traite les variables grâce à des modèles Jinja2.

Le logiciel Ansible a été conçu par un ancien employé de Red Hat, Michael DeHaan, également auteur de l’application de serveur de “provisionnement” Cobbler et co-auteur du framework Func pour l’administration à distance. Le code source du logiciel est sous licence GNU General Public v3.0. Red Hat a racheté la société Ansible, Inc. en octobre 2015.

2.2.2 Le terme Ansible :

Une “ansible” est un dispositif théorique permettant de réaliser des communications à une vitesse supraluminique (supérieure à la vitesse de la lumière) imaginé en 1966 par Ursula K. Le Guin dans son roman de science-fiction, *Le Monde de Rocannon*. Elle en détaillera plus tard le concept dans *Les Dépossédés* (1974). L’idée est notamment reprise par d’autres auteurs de livres de science-fiction et des jeux vidéos, la communication étant basée sur l’état d’énergie réciproque de deux particules jumelles. Par ailleurs, Ansible est le titre d’un magazine anglo-saxon consacré à la science-fiction. Enfin, le terme “ansible” peut faire référence à un système de communication hyperspace instantané fictif.

Comme une “ansible”, une tâche Ansible se déploie simultanément sous forme de module sur les cibles désignées.

2.2.3 Notions et Concepts

Ansible repose sur plusieurs notions et concepts clés :

- **Playbooks** : Les playbooks sont des fichiers YAML qui décrivent les tâches à effectuer par Ansible sur un ensemble de nœuds cibles. Ils définissent l’état désiré du système et les actions à entreprendre pour atteindre cet état.
- **Modules** : Les modules sont des morceaux de code exécutables par Ansible sur les nœuds cibles pour réaliser des actions spécifiques. Ils peuvent être utilisés dans les playbooks pour effectuer des opérations telles que l’installation de logiciels, la configuration de services, la gestion des fichiers, etc.
- **Inventaire** : L’inventaire est un fichier qui répertorie les hôtes sur lesquels Ansible doit agir. Il peut être statique ou dynamique et permet de regrouper les nœuds en fonction de différents critères tels que leur rôle, leur emplacement géographique, etc.
- **Adhoc commandes** : Les commandes ad-hoc permettent d’exécuter des tâches rapidement sans avoir à créer de playbook. Elles sont utiles pour des actions ponctuelles telles que la vérification de l’état d’un service ou la mise à jour d’un package sur un ensemble de nœuds.

2.2.4 Progression de Ansible avec le Réseau

Au fil du temps, Ansible a évolué pour inclure des fonctionnalités spécifiques au réseau, lui permettant de gérer efficacement les équipements réseau. Ces fonctionnalités comprennent :

- **Modules réseau** : Ansible fournit une collection de modules spécifiques au réseau qui permettent de configurer, surveiller et gérer des équipements réseau tels que des commutateurs, des routeurs, des pare-feu, etc.
- **Intégration avec les API réseau** : Ansible peut interagir avec les API exposées par les équipements réseau pour automatiser des tâches de configuration et de gestion. Cela permet d’étendre les capacités d’automatisation d’Ansible pour répondre aux besoins spécifiques des environnements réseau.
- **Gestion centralisée** : Ansible permet de gérer l’infrastructure réseau à partir d’une plateforme centralisée, ce qui facilite la configuration et la coordination des équipements réseau à grande échelle.

En résumé, Ansible offre une solution complète et flexible pour l'automatisation des tâches de gestion des configurations, avec des fonctionnalités spécifiques au réseau qui lui permettent de s'intégrer parfaitement dans les environnements informatiques modernes.

Chapitre 3

: Etude théorique d'Ansible

Introduction

Pour bien mener ce projet, il est nécessaire de faire une étude théorique détaillée sur Ansible. Ce chapitre sera composé de cinq parties. La première portera sur les Composants principaux d'Ansible, la deuxième sur les composants supplémentaires, la troisième sur les éléments de Configuration d'Ansible, la quatrième sur l'Interface Graphique Utilisateur, et enfin, la dernière partie concernera la Documentation d'Ansible.

3.1 Composants principaux de Ansible

3.1.1 Architecture de Ansible :

L'architecture d'Ansible est facile à comprendre.

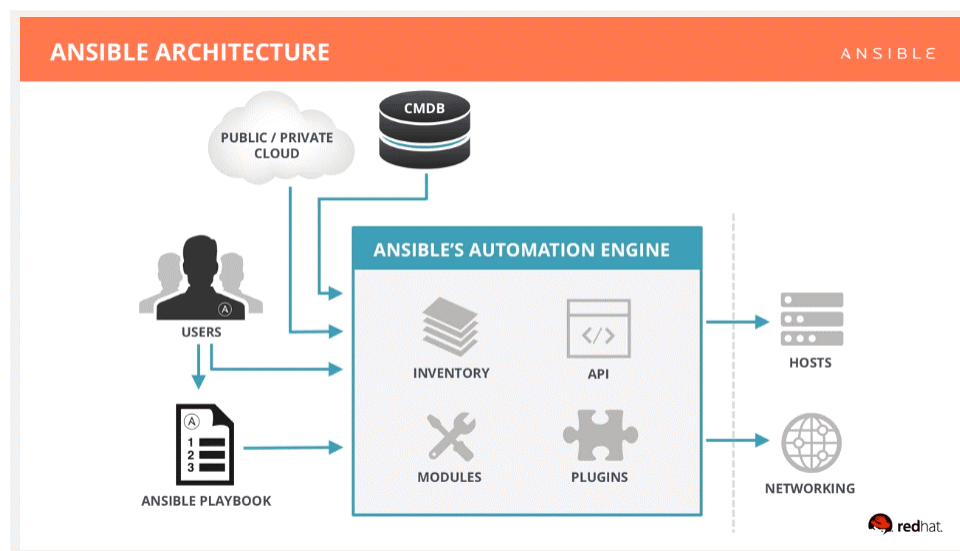


FIGURE 3.1.1 – Architecture générale de la solution Ansible

Inventory :

Le fichier inventory contient une liste des hôtes (hosts :) (généralement leurs adresses IP ou DNS) que vous souhaitez configurer ou gérer. Il est au format INI par défaut. Les hôtes d'un inventory peuvent être divisés en groupes plus petits pour faciliter la gestion et la configuration, il y a aussi des variables (vars :) attachées à ces hôtes et à ces groupes. Chaque groupe peut exécuter différentes tâches.

L'option `-i` suivie du chemin du fichier d'inventory est utilisée avec `ansible-playbook` pour préciser l'emplacement du fichier inventory. On peut aussi préciser le chemin du fichier d'inventory par défaut dans le fichier de configuration `ansible.cfg` avec l'entrée `inventory` sous la section `[defaults]`.

Playbook :

Un playbook est un fichier script de configuration d'Ansible écrit en langage YAML pour exécuter l'ensemble des tâches et étapes de configuration à effectuer sur les hôtes de manière séquentielle sur tout ou partie de l'inventory, c'est-à-dire un groupe déjà défini dans l'inventory.

Chaque playbook est composé d'un ou plusieurs "playbooks" exposés sous forme d'une liste. Un "playbook" organise des tâches (tasks) en play. Il est de bonne pratique de l'organiser en "rôles". Un "rôle" ajoute un niveau d'abstraction dans l'exécution des tâches d'un playbook.

Modules :

Les modules Ansible sont des programmes utilisés pour exécuter des tâches, ils sont principalement écrits en Python. Chaque tâche utilise un module qui peut prendre des paramètres pour être exécuté de manière personnalisée. Ansible exécute ensuite ces modules (via SSH par défaut) grâce au format JSON sur la sortie standard et les supprime lorsque l'action est terminée.

Facts :

Les Facts sont des informations collectées par Ansible et que l'on peut appeler via des variables `ansible`, et pour les périphériques de Cisco, appeler via des variables `ansible_net`. Par défaut, Ansible collecte les Facts des cibles avant l'exécution des tâches du playbook. Ce comportement se contrôle sur le playbook avec la directive `gather_facts` une valeur booléenne.

Rôles :

Les rôles sont des playbooks génériques, qui peuvent être intégrés dans d'autres playbooks. Ce concept est essentiel pour créer des tâches complexes. Pour rendre les playbooks lisibles, il vaut mieux construire des rôles qui peuvent être partagés, distribués et assemblés, plutôt que de créer un playbook qui sera difficile à maintenir, complexe à utiliser et peu lisible.

3.2 Composants supplémentaires :

3.2.1 Modèles Jinja2 :

Ansible utilise un module de "Templates" ou modèles, permettant de créer des fichiers scripts pour la gestion des équipements, et met en forme des données, il s'appuie sur Jinja2 qui permet de gérer des listes ou des variables, des boucles, des tests logiques, et donc d'étendre la programmabilité dans l'automatisation du réseau. Donc Jinja2 est un langage de gestionnaire des templates écrit pour Python.

Les modèles de données sont transformés en configuration de périphériques. -Les sorties des périphériques peuvent être transformées en documentation (dynamique). - On doit mettre l'extension j2 qui désigne ce fichier comme étant au format Jinja2 .

3.2.2 Ansible Vault :

Ansible vault est une fonctionnalité d'Ansible qui nous permet de crypter les fichiers qui contiennent des données sensibles telles que les mots de passe ou les clés, plutôt que de les laisser en clair dans les Playbooks ou les rôles.

Pour exécuter un Playbook avec des fichiers de données cryptés par Vault, nous devons fournir le mot de passe Vault, on peut stocker dans un fichier (dans ce exemple : pass vault.txt).

3.2.3 Mode ad-hoc :

Mode Ad-Hoc est la commande ansible en une ligne qui exécute une tâche sur l'hôte cible. Elle vous permet d'exécuter une tâche simple en une ligne sur un hôte ou un groupe d'hôtes définis dans la configuration du fichier d'inventary. Une commande Ad-Hoc n'aura que deux paramètres, le groupe d'hôtes sur lequel vous souhaitez effectuer la tâche et le module Ansible à exécuter.

La commande Ad-Hoc vous donne plus d'avantages pour l'exploration d'Ansible. Vous pouvez d'effectuer de tâche à manière rapide sans créer un playbook.

Dans cette partie, je vais montrer l'utilisation de base du mode Ad-Hoc, je vais l'utiliser pour effectuer une tâche simple :

```
Ansible -i inventorycisco -c local -m ios command -a "commands='show running-config' .
```

3.2.4 Langage YAML :

YAML, ou YAML Ain't Markup Language, est un langage de sérialisation de données couramment utilisé dans Ansible pour définir les configurations et les tâches à exécuter.

Voici quelques points clés sur l'utilisation de YAML dans Ansible :

- Syntaxe YAML : YAML utilise une syntaxe claire et lisible. Il est basé sur l'indentation pour définir la structure des données. Les listes sont représentées par des tirets (-) et les dictionnaires (ou "maps") sont représentés par des paires clé-valeur.
- Fichiers YAML Ansible : Les fichiers YAML dans Ansible sont utilisés pour définir les playbooks, les rôles, les inventaires, etc.
- Playbooks : Les playbooks Ansible sont des fichiers YAML qui décrivent une série de tâches à exécuter sur des hôtes distants.

Conclusion

Chapitre 4

Virtualisation

4.1 Introduction à la virtualisation :

L'introduction à la virtualisation consiste à comprendre les concepts et les principes de base de cette technologie. La virtualisation est une méthode qui permet de créer des environnements virtuels, indépendants les uns des autres, sur une seule machine physique. Cela signifie qu'une machine physique peut héberger plusieurs machines virtuelles (VM) qui fonctionnent de manière isolée les unes des autres.

L'objectif principal de la virtualisation est d'optimiser l'utilisation des ressources matérielles, telles que le processeur, la mémoire et le stockage, en les partageant entre plusieurs machines virtuelles. Chaque VM a son propre système d'exploitation et ses applications, ce qui donne l'impression qu'il s'agit de machines physiques distinctes.

La virtualisation est réalisée à l'aide d'un logiciel appelé hyperviseur, également connu sous le nom de moniteur de machine virtuelle (VMM). L'hyperviseur permet de créer, gérer et exécuter les machines virtuelles. Il fournit une interface entre le matériel physique et les machines virtuelles, permettant ainsi à chaque VM d'accéder aux ressources matérielles de manière contrôlée.

4.1.1 Les types principaux de virtualisation :

Virtualisation de type 1 (native) : Dans ce type de virtualisation, l'hyperviseur s'exécute directement sur le matériel physique, sans système d'exploitation hôte. Les machines virtuelles sont créées et exécutées directement sur l'hyperviseur. Cela offre une performance élevée et une isolation maximale entre les machines virtuelles. Exemples d'hyperviseurs de type 1 : VMware ESXi, Microsoft Hyper-V, Xen.

Virtualisation de type 2 (hébergée) : Dans ce type de virtualisation, l'hyperviseur s'exécute comme une application sur un système d'exploitation hôte. Les machines virtuelles sont créées et exécutées sur l'hyperviseur, qui utilise les ressources du système d'exploitation hôte. Cela offre une flexibilité et une facilité d'utilisation, mais peut entraîner une légère perte de performance. Exemples d'hyperviseurs de type 2 : VMware Workstation, Oracle VirtualBox.

4.1.2 Les avantages :

- Consolidation des serveurs physiques : La virtualisation permet de consolider plusieurs serveurs physiques en un seul serveur physique, réduisant ainsi les coûts d'achat, de maintenance et de consommation d'énergie.
- Isolation des environnements : Chaque machine virtuelle fonctionne de manière isolée, ce qui signifie que les problèmes sur une VM n'affectent pas les autres. Cela améliore la sécurité et la stabilité du système.
- Flexibilité et évolutivité : Les machines virtuelles peuvent être créées, supprimées ou déplacées facilement, ce qui offre une grande flexibilité et une évolutivité rapide en fonction des besoins.
- Gestion simplifiée : Les outils de gestion de la virtualisation permettent de gérer efficacement les machines virtuelles, y compris leur déploiement, leur surveillance et leur sauvegarde.
- Test et développement : La virtualisation offre un environnement idéal pour le test et le développement d'applications, permettant de créer des environnements

4.2 Mise en place de l'environnement virtuel :

4.2.1 VMware :

VMware est une société de logiciels de virtualisation et de cloud computing basée à Palo Alto, en Californie. Fondée en 1998, VMware est une filiale de Dell Technologies. EMC Corporation a initialement acquis VMware en 2004, et EMC a été par la suite acquise par Dell Technologies en 2016. Les technologies de virtualisation de VMware sont basées sur son hyperviseur bare-metal ESX/ESXi dans l'architecture x86.

Avec la virtualisation de serveur de VMware, un hyperviseur est installé sur le serveur physique pour permettre à plusieurs machines virtuelles (VMs) de fonctionner sur le même serveur physique. Chaque VM peut exécuter son propre système d'exploitation (OS), ce qui signifie que plusieurs OS peuvent fonctionner sur un seul serveur physique. Toutes les VMs sur le même serveur physique partagent des ressources, telles que la mise en réseau et la RAM. En 2019, VMware a ajouté la prise en charge à son hyperviseur pour exécuter des charges de travail conteneurisées dans un cluster Kubernetes de la même manière. Ces types de charges de travail peuvent être gérés par l'équipe d'infrastructure de la même manière que les machines virtuelles, et les équipes DevOps peuvent déployer des conteneurs comme ils étaient habitués.

VMware a été fondée par Diane Greene, Scott Devine, Mendel Rosenblum, Edward Wang et Edouard Bugnion, et a lancé son premier produit, VMware Workstation, en 1999. La société a publié son deuxième produit, VMware ESX, en 2001. Le PDG actuel de VMware est Patrick Gelsinger, nommé en 2012.

Les produits VMware incluent des outils de virtualisation, de gestion de réseau et de sécurité, des logiciels de centre de données à logiciel défini et des logiciels de stockage.

4.2.2 Avantages de VMware :

En tant que leader du domaine de la virtualisation, VMware offre un certain nombre de produits et services spécialement conçus pour virtualiser un datacenter. Avantages d'utiliser VMware pour la virtualisation :

Des avantages économiques grâce à la division d'un seul serveur physique en plusieurs machines virtuelles Une agilité et une flexibilité informatiques accrues pour répartir votre charge de travail dans l'ensemble de votre infrastructure selon vos besoins

Déploiement, gestion et maintenance des machines virtuelles rationalisés Virtualisation granulaire des réseaux de stockage (SAN) et des matrices de stockage en réseau (NAS) avec volumes virtuels (vVols)



FIGURE 4.2.1 – Exemple du fichier d'inventory

4.3 Mise en place d'un simulateur :

4.3.1 Présentation EVE-NG (Emulated Virtual Environment - Next Generation) :

EVE-NG est une plate-forme de virtualisation réseau qui permet de créer des environnements réseau virtuels complexes en utilisant des machines virtuelles pour simuler des équipements réseau tels que des routeurs, des commutateurs, des pare-feu, etc. Il offre une interface graphique conviviale pour la conception et la gestion de ces topologies virtuelles.

4.3.2 Avantages de EVE-ng :

Flexibilité : EVE-NG prend en charge une large gamme d'appareils réseau virtuels et offre une grande flexibilité pour créer des topologies personnalisées.

Interface graphique conviviale : Son interface utilisateur intuitive permet de créer, configurer et gérer facilement des topologies réseau complexes.

Communauté active : EVE-NG bénéficie d'une communauté active qui partage des topologies prêtes à l'emploi et des connaissances.

Évolutivité : Il peut être déployé sur des serveurs locaux ou sur des plates-formes cloud, offrant ainsi une évolutivité pour répondre aux besoins des entreprises de toutes tailles.

4.3.3 Présentation Penetlab :

Penetlab est une plate-forme de formation à la cybersécurité qui offre des environnements virtuels prêts à l'emploi pour la pratique de divers scénarios d'attaque et de défense. Il propose une série de laboratoires où les utilisateurs peuvent pratiquer des techniques de piratage éthique, des analyses de vulnérabilités et des tests de sécurité.

4.3.4 Avantages de Panetlab :

Scénarios prêts à l'emploi : Penetlab fournit des environnements de laboratoire complets avec des scénarios de formation prêts à l'emploi, ce qui facilite la mise en place rapide de simulations de cyberattaques et de défense.

Orientation vers la sécurité : Contrairement à EVE-NG qui se concentre sur la virtualisation réseau en général, Penetlab est spécifiquement axé sur la cybersécurité, offrant ainsi des outils et des exercices ciblés sur la sécurité informatique.

Didactique : Il est conçu pour être utilisé dans un contexte pédagogique, avec des guides d'utilisation et des instructions détaillées pour aider les apprenants à tirer le meilleur parti des laboratoires.

Accessibilité : Penetlab peut être facilement déployé localement ou sur des plates-formes cloud, offrant ainsi une grande accessibilité aux utilisateurs.

4.4 Conclusion :

EVE-NG et Penetlab sont deux outils complémentaires, mais avec des objectifs différents. EVE-NG est idéal pour la simulation et la formation en matière de réseaux, tandis que Penetlab se concentre spécifiquement sur la cybersécurité et offre des environnements de laboratoire prêts à l'emploi pour la pratique de scénarios d'attaque et de défense. Le

choix entre les deux dépendra des besoins spécifiques de l'utilisateur en termes de formation et de simulation. Mais pour nos objectifs, nous utiliserons Eve-ng.

Chapitre 5

Environnement de travail

Introduction :

Ce chapitre consistera en la préparation de notre environnement, en expliquant toutes les étapes qui contribueront au bon fonctionnement de notre projet.

Ubuntu qui est un système d'exploitation de type Unix, très connu dans le monde de la programmation sera notre système d'exploitation principal.

D'abord, nous aurons besoin d'y installer principalement VMWare Workstation pro qui nous permettra d'installer des machines virtuelles, afin de pouvoir gérer différents nœuds. Ainsi que EVN-EG qui est utilisé pour émuler plusieurs systèmes d'exploitation dans un environnement virtuel à l'aide des systèmes d'exploitation Cisco Inter-network. Nous pourrons donc y architecturer notre réseau et le configurer. Ensuite, nous expliquerons toutes les configurations de bases de notre réseau, les logiciels ainsi que les commandes nécessaires afin de connecter toutes les machines et tous les équipements entre eux.

Enfin, nous montrerons comment installer l'outil d'automatisations de notre projet.

5.1 Installation des Packages APT :

Lorsque nous débutons sur UBUNTU il est important de mettre à jour et d'installer les packages APT qui nous seront utiles dans nos prochaines opérations. Advanced Packaging Tool est un système complet et avancé de gestion de paquets, permettant une recherche facile et efficace, une installation simple et une désinstallation propre de logiciels et utilitaires.

Il permet également de faciliter la mise à jour de la distribution Ubuntu avec les paquets en versions les plus récentes et de passer à une nouvelle version de Ubuntu,

lorsque celle-ci est disponible.

Les commandes à implémenter pour charger APT se présentent comme suit :

Installation du package APT

```
pfe@pfe-virtual-machine:~$ sudo apt-get install
```

FIGURE 5.1.1 – Installation du package APT.

Mise à jour des packages.

```
pfe@pfe-virtual-machine:~$ sudo apt-get upgrade
```

FIGURE 5.1.2 – Mise à jour des packages.

Mise à jour système

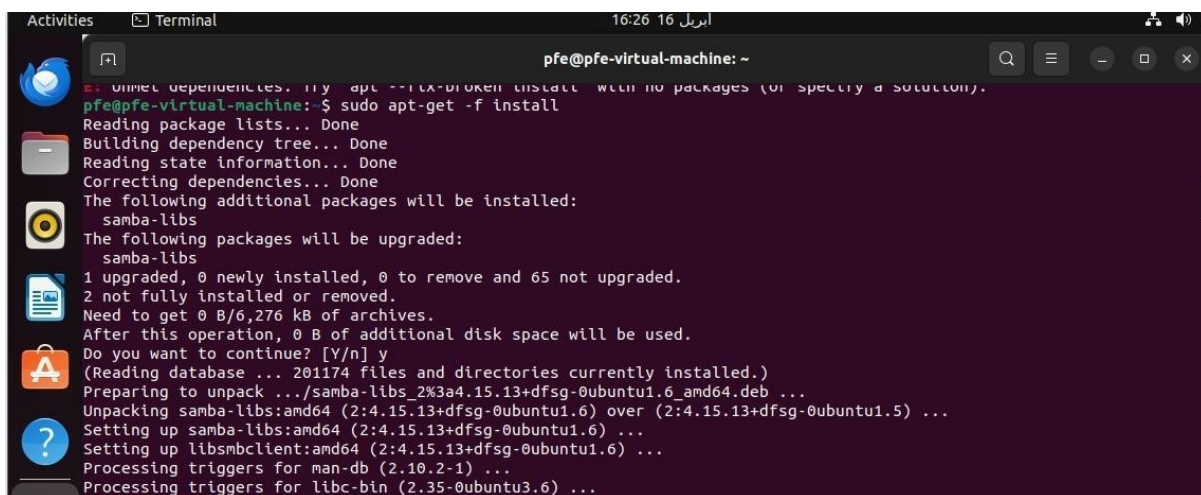
```
pfe@pfe-virtual-machine:~$ sudo apt update
```

FIGURE 5.1.3 – Mise à jour système.

5.2 Installation d'Ansible sous Ubuntu :

Vérification des dépendances

Pour commencer, vérifiez et résolvez les éventuelles dépendances cassées.



A terminal window titled 'Terminal' with the prompt 'pfe@pfe-virtual-machine: ~'. The user enters the command 'sudo apt-get -f install'. The terminal output shows the process of reading package lists, building a dependency tree, and correcting dependencies. It lists 'samba-ls' as the package to be installed and 'samba-ls' as the additional package to be installed. The output also shows the disk space requirements and the confirmation to continue. The installation process is shown as successful, with the package being unpacked and set up.

```
pfe@pfe-virtual-machine: ~  
$ sudo apt-get -f install  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Correcting dependencies... Done  
The following additional packages will be installed:  
  samba-ls  
The following packages will be upgraded:  
  samba-ls  
1 upgraded, 0 newly installed, 0 to remove and 65 not upgraded.  
Need to get 0 B/6,276 kB of archives.  
After this operation, 0 B of additional disk space will be used.  
Do you want to continue? [Y/n] y  
(Reading database ... 201174 files and directories currently installed.)  
Preparing to unpack .../samba-ls_2%3a4.15.13+dfsg-0ubuntu1.6_amd64.deb ...  
Unpacking samba-ls:amd64 (2:4.15.13+dfsg-0ubuntu1.6) over (2:4.15.13+dfsg-0ubuntu1.5) ...  
Setting up samba-ls:amd64 (2:4.15.13+dfsg-0ubuntu1.6) ...  
Setting up libsmclient:amd64 (2:4.15.13+dfsg-0ubuntu1.6) ...  
Processing triggers for man-db (2.10.2-1) ...  
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
```

FIGURE 5.2.1 – Vérification d’une dépendance.



A terminal window showing the installation of python3 and its dependencies. The user enters the command 'sudo apt install python3 python3-pip'. The terminal output shows the process of reading package lists, building a dependency tree, and installing python3. It lists a large number of additional packages that will be installed, including binutils, fakeroot, g++, gcc, javascript-common, libalgorithm-diff-perl, libalgorithm-diff-xs-perl, libalgorithm-merge-perl, libasan6, libbinutils, libc-dev-bin, libc-devtools, libc6-dev, libcc1-0, libcrypt-dev, libctf-nobfd0, libctf0, libdpkg-perl, libexpat1, libexpat1-dev, libfakeroot, libfile-fcntllock-perl, libgcc-11-dev, libitm1, libjs-jquery, libjs-sphinxdoc, libjs-underscore, liblsan0, libnsl-dev, libpython3-dev, libpython3.10-dev, libstdc++-11-dev, libtirpc-dev, libtsan0, libubsan1, linux-libc-dev, lto-disabled-list, make, manpages-dev, python3-dev, python3-distutils, python3-setuptools, python3-wheel, python3.10-dev, rpcsvc-proto, zlib1g-dev. Suggested packages are also listed.

```
pfe@pfe-virtual-machine:~$ sudo apt install python3 python3-pip  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
python3 is already the newest version (3.10.6-1~22.04).  
python3 set to manually installed.  
The following additional packages will be installed:  
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev  
  fakeroot g++ g++-11 gcc gcc-11 javascript-common libalgorithm-diff-perl  
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libbinutils  
  libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0  
  libctf0 libdpkg-perl libexpat1 libexpat1-dev libfakeroot  
  libfile-fcntllock-perl libgcc-11-dev libitm1 libjs-jquery libjs-sphinxdoc  
  libjs-underscore liblsan0 libnsl-dev libpython3-dev libpython3.10-dev  
  libstdc++-11-dev libtirpc-dev libtsan0 libubsan1 linux-libc-dev  
  lto-disabled-list make manpages-dev python3-dev python3-distutils  
  python3-setuptools python3-wheel python3.10-dev rpcsvc-proto zlib1g-dev  
Suggested packages:  
  binutils-doc debian-keyring g++-multilib g++-11-multilib gcc-11-doc  
  gcc-multilib autoconf automake libtool flex bison qcc-doc qcc-11-multilib
```

FIGURE 5.2.2 – Installer les dépendances nécessaires .

l'installation via pip :

Ensuite, vous pouvez installer Ansible via pip (le gestionnaire de packages Python).

```
pfe@pfe-virtual-machine:~$ sudo pip3 install ansible
Collecting ansible
  Downloading ansible-9.4.0-py3-none-any.whl (46.4 MB)
    46.4/46.4 MB 8.9 MB/s eta 0:00:00
Collecting ansible-core~=2.16.5
  Downloading ansible_core-2.16.6-py3-none-any.whl (2.3 MB)
    2.3/2.3 MB 14.6 MB/s eta 0:00:00
Collecting resolvelib<1.1.0,>=0.5.3
  Downloading resolvelib-1.0.1-py2.py3-none-any.whl (17 kB)
Collecting packaging
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
    53.5/53.5 KB 11.9 MB/s eta 0:00:00
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible-core~=2.16.6)
Collecting jinja2>=3.0.0
  Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
    133.2/133.2 KB 3.2 MB/s eta 0:00:00
```

FIGURE 5.2.3 – Installer Ansible via pip .

5.3 Installation d'EVN-EG dans VMware

la Configuration des paramètres

la Configuration des paramètres de la machine virtuelle eve ng tels que la quantité de RAM, la capacité du disque dur, le nombre de cœurs de processeur, etc.

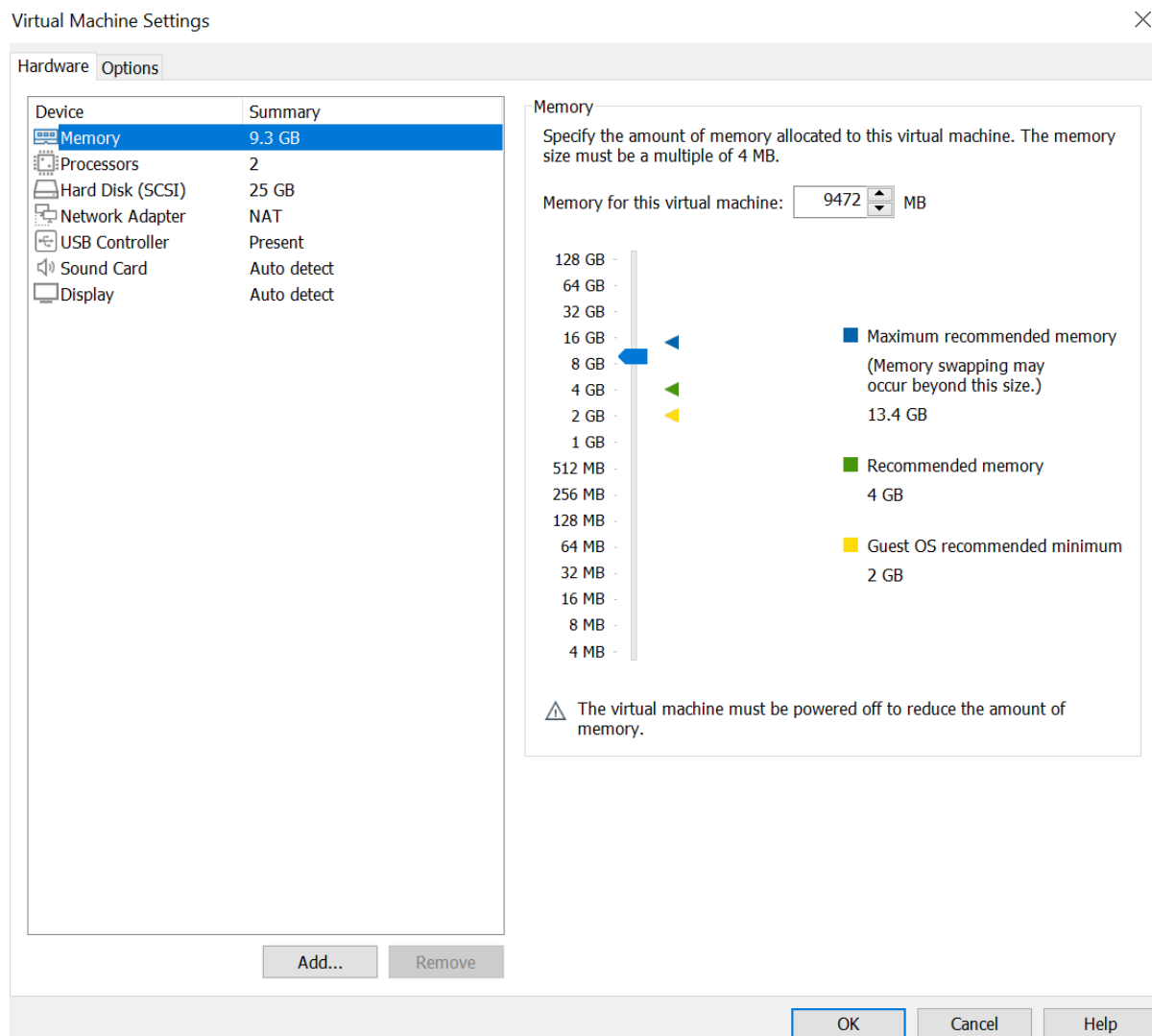


FIGURE 5.3.1 – la Configuration des paramètres d'éveng.

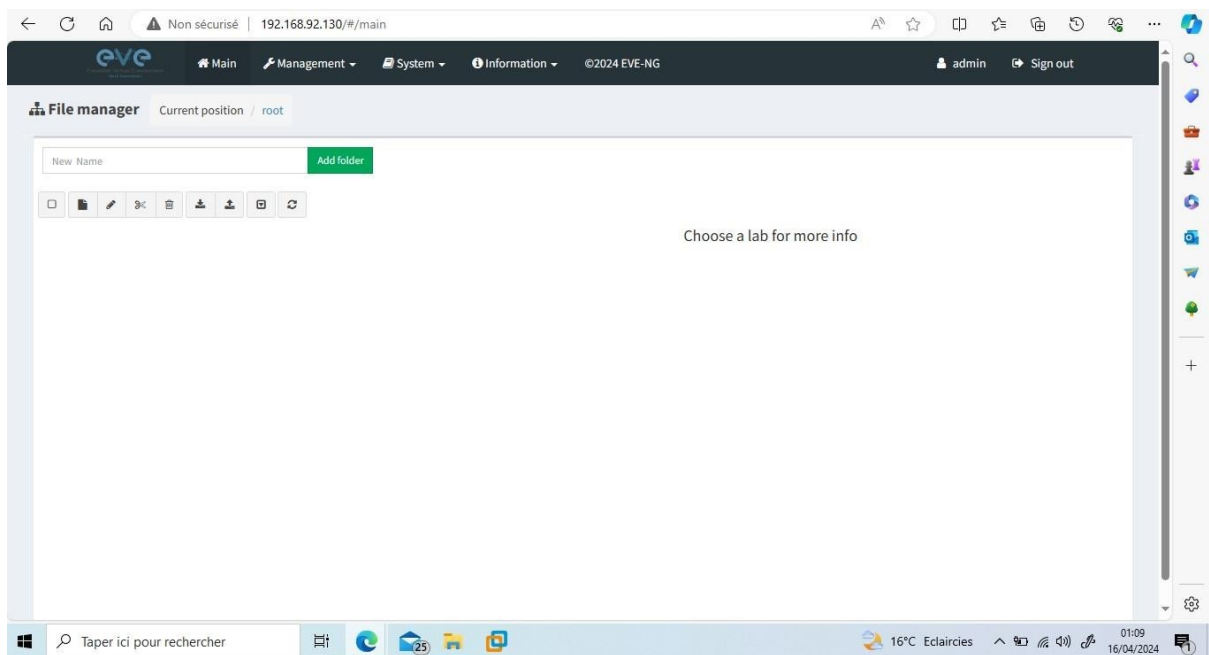


FIGURE 5.3.2 – l’interface web d’EVE-NG

....

5.4 Installation d'une image d'appareil réseau dans EVE-NG :

Téléchargement de l'image :

- Téléchargement de l'image : Tout d'abord, nous sommes assurés de télécharger une image du périphérique réseau que nous souhaitons utiliser lors de l'événement, et nous sommes également assurés que l'image était compatible avec l'événement.

Connectez-vous à EVE-NG à l'aide de PuTTY :

- Nous avons exécuté PuTTY sur notre ordinateur.
- Dans la fenêtre PuTTY, nous avons saisi l'adresse IP de notre appareil EVE-NG dans le champ Nom d'hôte (ou adresse IP) .
- Nous nous sommes assurés que le port était défini sur 22 (valeur par défaut pour SSH).
- Cliquer sur Ouvrir pour ouvrir la session SSH.

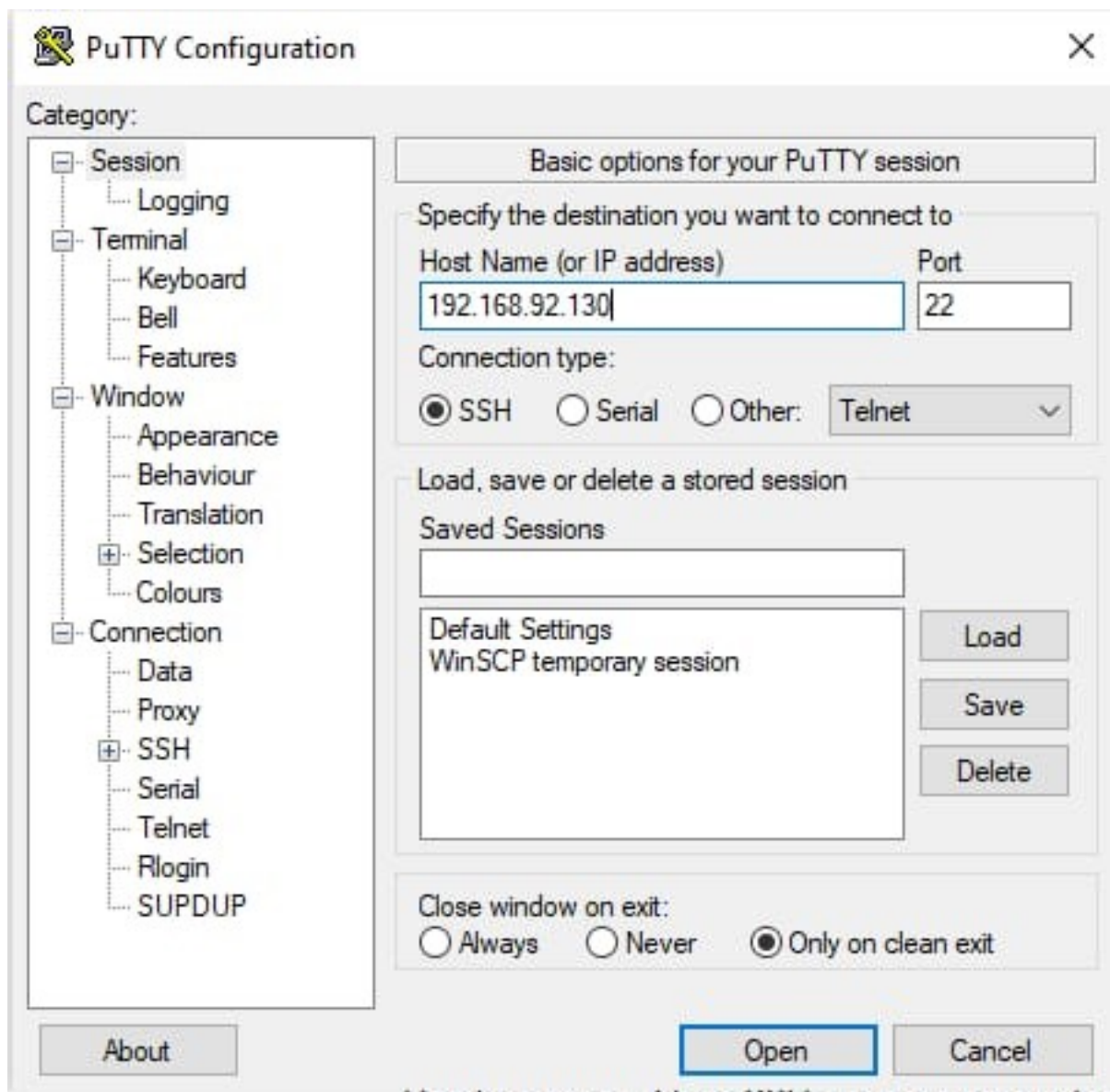


FIGURE 5.4.1 – Connexion à EVE-NG avec PuTTY

Exemple d'installation d'une image de routeur

la création le répertoire d'images sur `/opt/unetlab/addons/qemu` d'EVE, cet exemple concerne la première image du tableau ci-dessus. Il s'agit de l'image du routeur vios L3. Selon notre table de dénomination d'image , nous devons créer un dossier d'images commençant par vios- .

```
root@eve-ng:/opt/unetlab/addons/qemu/vios-adventerprisek9-m.SPA.159-3.M6# ls
virtioa.qcow2
root@eve-ng:/opt/unetlab/addons/qemu/vios-adventerprisek9-m.SPA.159-3.M6#
```

FIGURE 5.4.2 – le répertoire d'images `/opt/unetlab/addons/qemu` d'EVE

Transfert de l'image à l'aide de WinSCP :

- Nous avons exécuté WinSCP sur notre ordinateur.
- Dans la fenêtre de connexion WinSCP, nous avons entré l'adresse IP, le nom d'utilisateur et le mot de passe de l'appareil EVE-NG.
- Nous avons cliqué sur Connexion pour nous connecter à notre appareil EVE-NG.

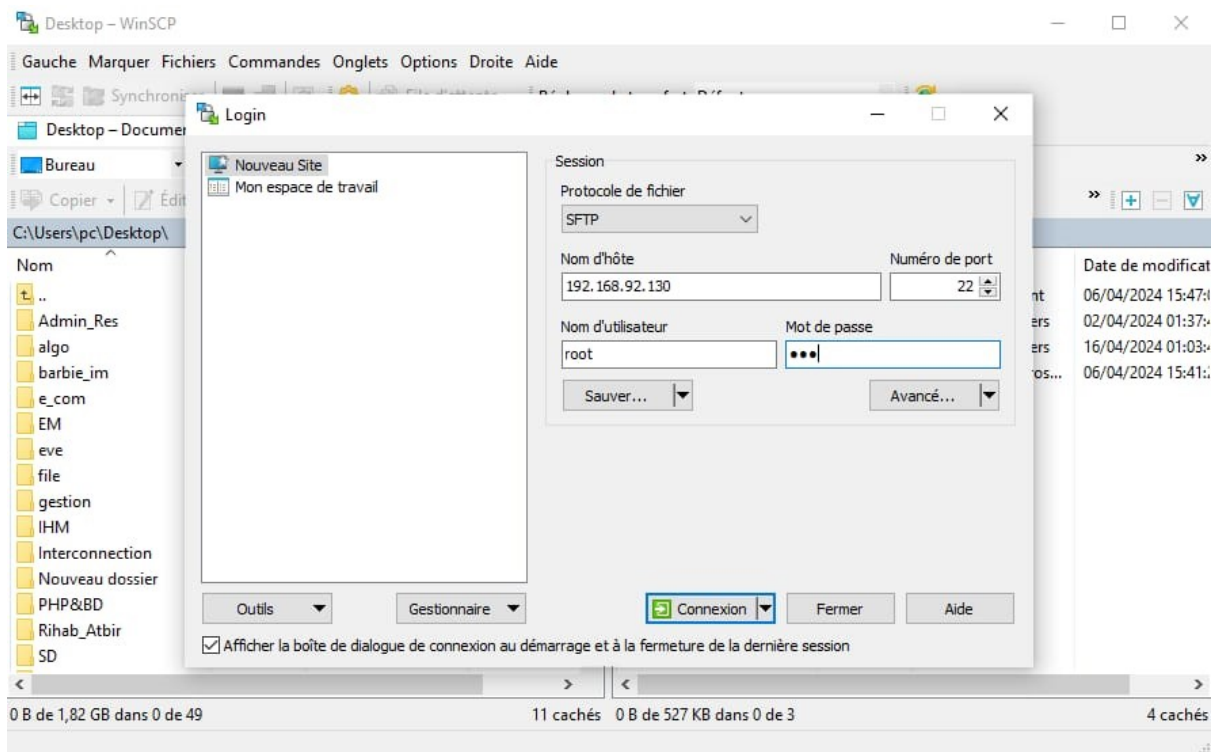


FIGURE 5.4.3 – Connexion WinSCP pour se connecter à notre appareil EVE-NG

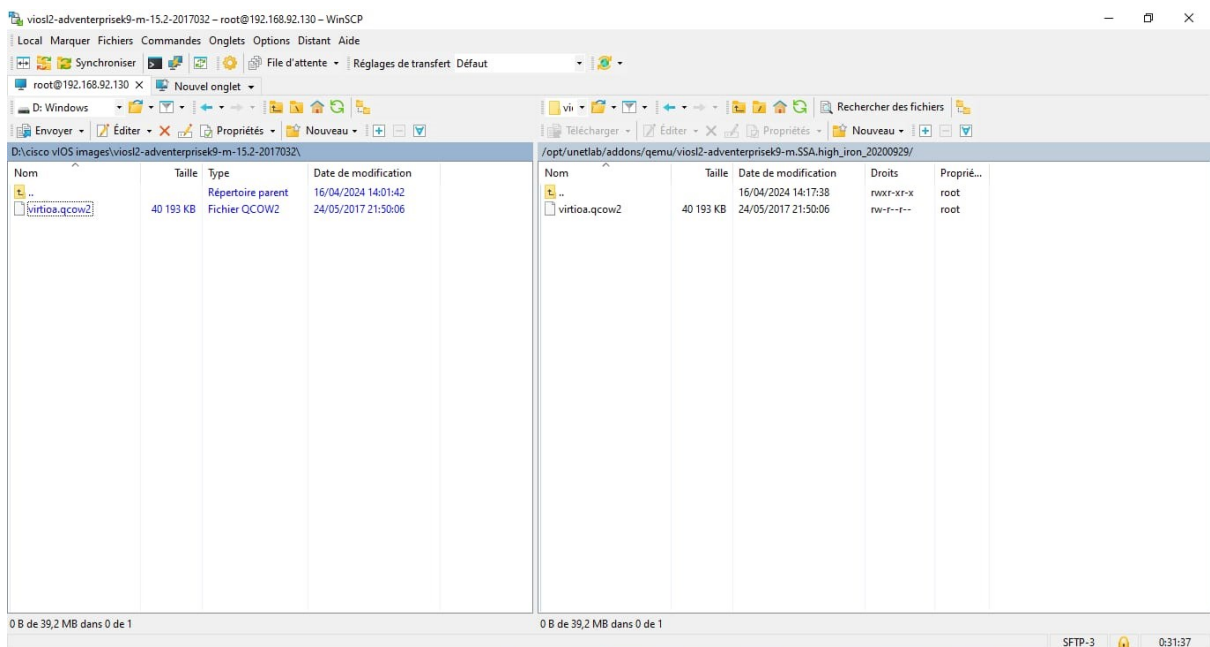


FIGURE 5.4.4 – Transfert de l'image avec WinSCP

Chapitre 6

Mise en place de la solution et automatisatisation

6.1 La topologie réseau adoptée

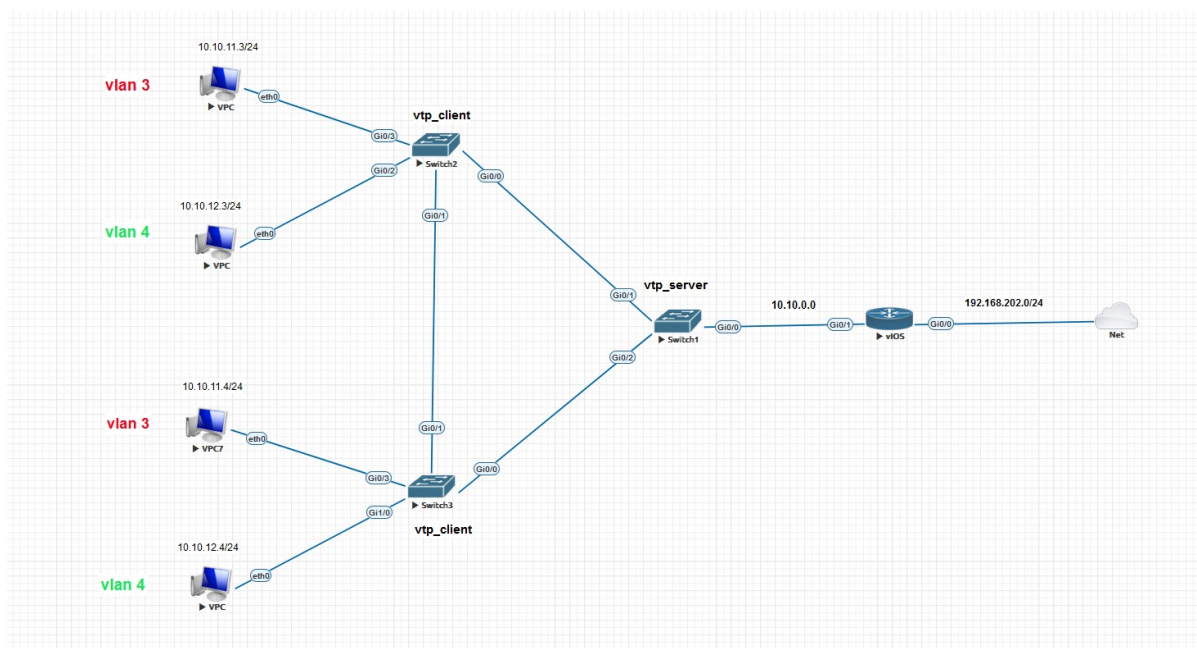


FIGURE 6.1.1 – Topologie réseau

6.2 Configuration de l'accès SSH

SSH (Secure Shell) est un protocole de communication sécurisé largement utilisé dans les environnements informatiques pour administrer à distance des appareils réseau tels que les switches Cisco. Il fournit un moyen sécurisé d'accéder et de gérer des équipements réseau à partir d'un emplacement distant, en établissant une connexion cryptée entre le client SSH et le serveur SSH.

Avant de commencer l'automatisation avec Ansible, il est essentiel de s'assurer que SSH est configuré et fonctionnel sur les appareils cibles afin de :

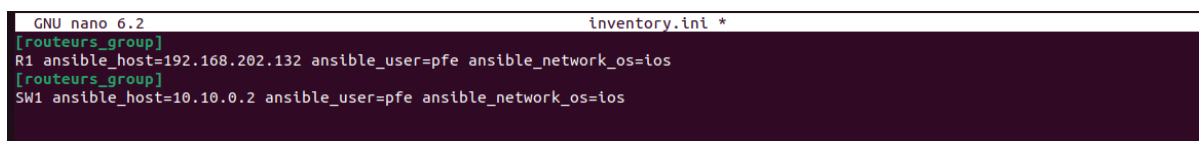
- **Établissement de la connexion** : Ansible établit une connexion SSH avec les appareils cibles pour envoyer et exécuter des commandes de configuration. Par conséquent, SSH doit être configuré et activé sur les appareils pour permettre ces connexions.
- **Authentification** : SSH fournit un mécanisme d'authentification sécurisé pour se connecter aux appareils distants. Lors de l'utilisation d'Ansible, les clés SSH (publiques et privées) sont généralement utilisées pour l'authentification. Vous devrez configurer les clés publiques sur les appareils cibles afin qu'Ansible puisse s'authentifier sans avoir à saisir de mots de passe à chaque connexion.
- **Sécurité des communications** : La configuration de SSH garantit que les communications entre Ansible et les appareils cibles sont sécurisées et chiffrées. Cela protège les informations sensibles transmises pendant le processus d'automatisation.

Pour configurer SSH sur les switches, nous utilisons les étapes suivantes :

1. Configurer le nom de domaine par défaut.
2. Générer la clé RSA.
3. Activer le protocole SSH.
4. Configurer les lignes de terminal virtuelles (VTY).

6.3 Configuration de mappage sur inventory.ini

Le fichier `inventory.ini` est utilisé pour spécifier les hôtes et les groupes d'hôtes que Ansible va gérer. Dans notre cas, nous avons configuré le mappage des routeurs ainsi que l'adresse du switch où les VLANs ont été configurés. Cette configuration permet à Ansible d'identifier et de se connecter aux dispositifs réseau de manière simplifiée et organisée.



```
GNU nano 6.2                                inventory.ini *
[routeurs_group]
R1 ansible_host=192.168.202.132 ansible_user=pfe ansible_network_os=ios
[switch_group]
SW1 ansible_host=10.10.0.2 ansible_user=pfe ansible_network_os=ios
```

FIGURE 6.3.1 – Le fichier inventaire

6.3.1 Connexion à distance via SSH

En premier lieu il est préférable de faire un test de connectivité entre les deux machines en utilisant la commande `ping`.

Dans notre cas nous avons déjà un agent SSH ajouté par défaut par notre système d'exploitation UBUNTU. Il suffit alors d'introduire les commandes comme illustrées sur la figure 6.4

```

pfe@pfe-virtual-machine:/etc$ ping 192.168.202.132
PING 192.168.202.132 (192.168.202.132) 56(84) bytes of data.
64 bytes from 192.168.202.132: icmp_seq=1 ttl=255 time=6.98 ms
64 bytes from 192.168.202.132: icmp_seq=2 ttl=255 time=4.71 ms
64 bytes from 192.168.202.132: icmp_seq=3 ttl=255 time=0.977 ms
64 bytes from 192.168.202.132: icmp_seq=4 ttl=255 time=3.55 ms
64 bytes from 192.168.202.132: icmp_seq=5 ttl=255 time=1.68 ms
^C
--- 192.168.202.132 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 0.977/3.577/6.976/2.153 ms
pfe@pfe-virtual-machine:/etc$ █

```

FIGURE 6.3.2 – Ping entre la machine ansible et R1

On peut voir que nous avons réussi à accéder à notre machine via le protocole SSH. Nous pouvons aussi nous déconnecter en utilisant la commande logout et revenir facilement sur notre machine locale.

```

pfe@pfe-virtual-machine:/etc$ ssh -o HostKeyAlgorithms=ssh-rsa -o KexAlgorithms=diffie-hellman-group1-sha1 pfe@192.168.202.132
*****
* IOSv - Cisco Systems Confidential *
*
* This software is provided as is without warranty for internal
* development and testing purposes only under the terms of the Cisco
* Early Field Trial agreement. Under no circumstances may this software
* be used for production purposes or deployed in a production
* environment.
*
* By using the software, you agree to abide by the terms and conditions
* of the Cisco Early Field Trial Agreement as well as the terms and
* conditions of the Cisco End User License Agreement at
* http://www.cisco.com/go/eula
*
* Unauthorized use or distribution of this software is expressly
* Prohibited.
(pfe@192.168.202.132) Password: *****
(pfe@192.168.202.132) Password:
*****
* IOSv - Cisco Systems Confidential *
*
* This software is provided as is without warranty for internal
* development and testing purposes only under the terms of the Cisco
* Early Field Trial agreement. Under no circumstances may this software
* be used for production purposes or deployed in a production
* environment.
*
* By using the software, you agree to abide by the terms and conditions
* of the Cisco Early Field Trial Agreement as well as the terms and
* conditions of the Cisco End User License Agreement at
* http://www.cisco.com/go/eula
*
* Unauthorized use or distribution of this software is expressly
* Prohibited.
*****
R1# █

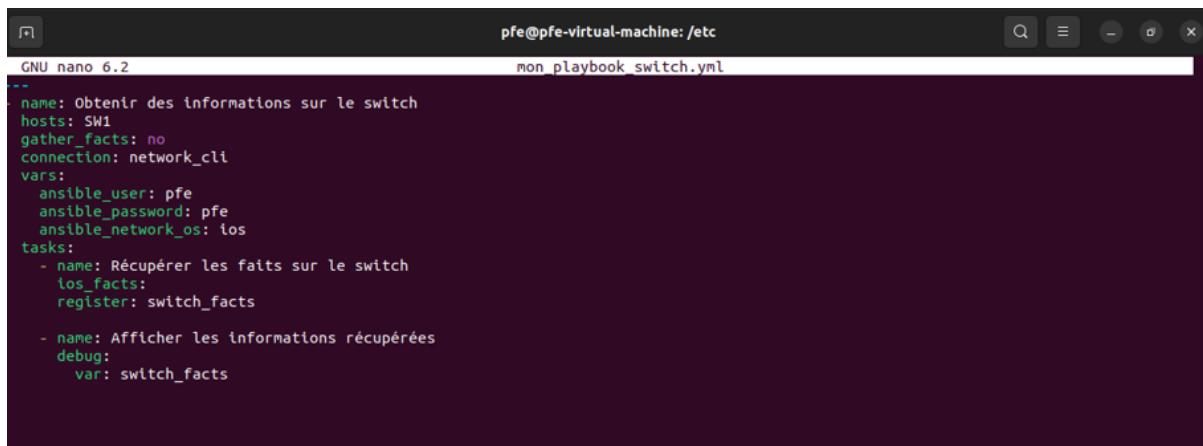
```

FIGURE 6.3.3 – Test de connectivité via SSH

6.4 Configuration d'ansible et tests

6.4.1 Playbook : Obtention des informations sur le switch

Le playbook `playbook_switch.yml` contient les instructions pour obtenir des informations sur le switch à l'aide d'Ansible. Ces informations peuvent inclure des détails sur la configuration actuelle du switch, son état, ses interfaces, etc.



```
pfe@pfe-virtual-machine: /etc
GNU nano 6.2 mon_playbook_switch.yml
--
name: Obtenir des informations sur le switch
hosts: SW1
gather_facts: no
connection: network_cli
vars:
  ansible_user: pfe
  ansible_password: pfe
  ansible_network_os: ios
tasks:
  - name: Récupérer les faits sur le switch
    ios_facts:
    register: switch_facts

  - name: Afficher les informations récupérées
    debug:
      var: switch_facts
```

FIGURE 6.4.1 – Contenu du playbook `playbook_switch.yml`

Résultat du test :

```

pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini mon_playbook_switch.yml

PLAY [Obtenir des informations sur le switch] *****

TASK [Récupérer les faits sur le switch] *****
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
ok: [SW1]

TASK [Afficher les informations récupérées] *****
ok: [SW1] => {
  "switch_facts": {
    "ansible_facts": {
      "ansible_net_api": "clitconf",
      "ansible_net_gather_network_resources": [],
      "ansible_net_gather_subset": [
        "default"
      ],
      "ansible_net_hostname": "SW1",
      "ansible_net_image": "flash0:/vios_l2-adventerprisek9-m",
      "ansible_net_iostype": "IOS",
      "ansible_net_model": "IOSv",
      "ansible_net_operatingmode": "autonomous",
      "ansible_net_python_version": "3.10.12",
      "ansible_net_serialnum": "9TMDMOE3GKE",
      "ansible_net_system": "ios",
      "ansible_net_version": "15.2(20170321:233949)",
      "ansible_network_resources": {}
    },
    "changed": false,
    "failed": false
  }
}

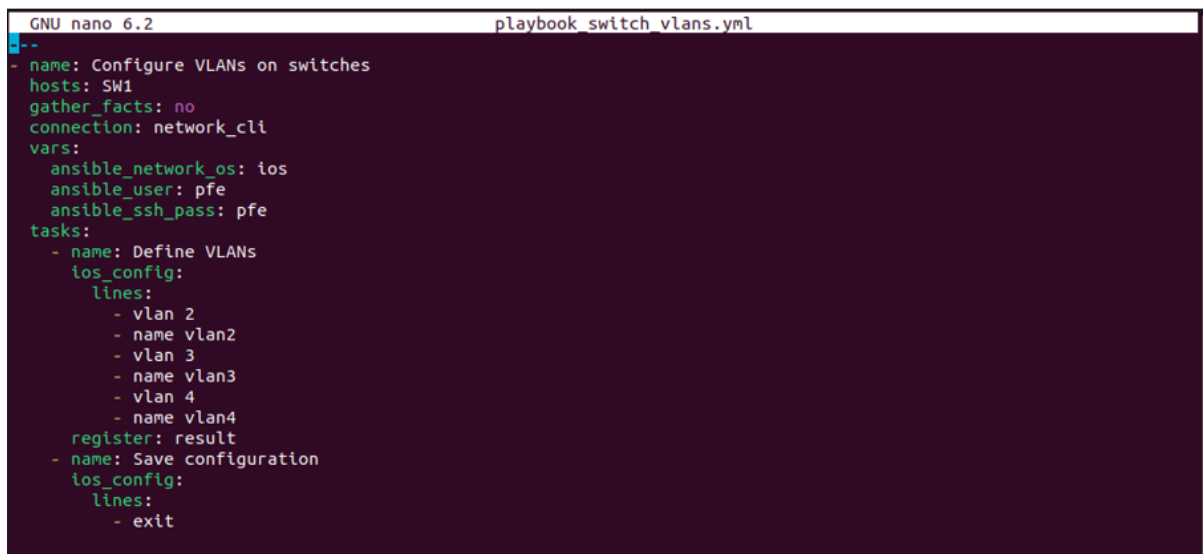
PLAY RECAP *****
SW1 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

```

FIGURE 6.4.2 – Résultat du test pour le playbook playbook_switch.yml

6.4.2 Playbook : Configuration des VLANs sur le switch

Le playbook `playbook_switch_vlans.yml` décrit les étapes pour configurer les VLANs sur le switch à l'aide d'Ansible. Cela implique généralement la création, la modification ou la suppression de VLANs.



```
GNU nano 6.2                                playbook_switch_vlans.yml
--
- name: Configure VLANs on switches
  hosts: SW1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_ssh_pass: pfe
  tasks:
    - name: Define VLANs
      ios_config:
        lines:
          - vlan 2
          - name vlan2
          - vlan 3
          - name vlan3
          - vlan 4
          - name vlan4
        register: result
    - name: Save configuration
      ios_config:
        lines:
          - exit
```

FIGURE 6.4.3 – Contenu du playbook `playbook_switch_vlans.yml`

Résultat du test :



```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_switch_vlans.yml

PLAY [Configure VLANs on switches] *****

TASK [Define VLANs] *****
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they
appear if present in the running configuration on device
changed: [SW1]

TASK [Save configuration] *****
changed: [SW1]

PLAY RECAP *****
SW1                : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

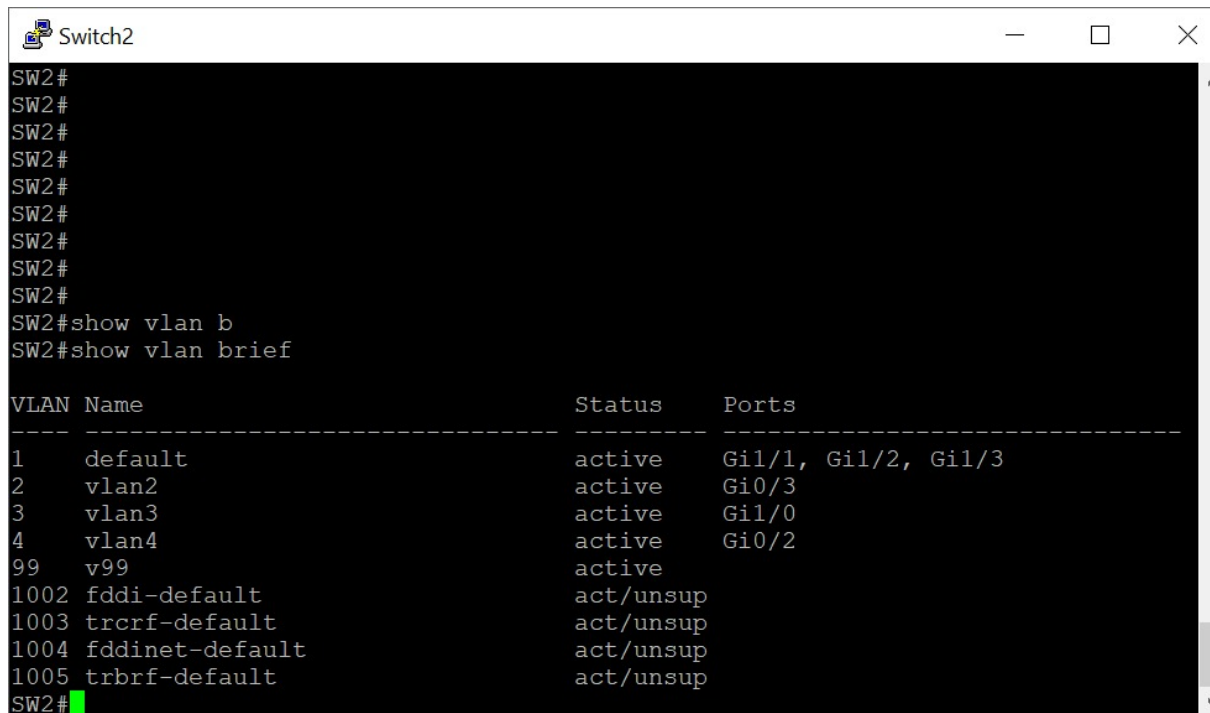
FIGURE 6.4.4 – Résultat du test pour le playbook `playbook_switch_vlans.yml`

```
SW1#show vlan brief
```

VLAN	Name	Status	Ports
1	default	active	Gi0/0, Gi0/3, Gi1/0, Gi1/1 Gi1/2, Gi1/3
2	vlan2	active	
3	vlan3	active	
4	vlan4	active	
99	v99	active	
1002	fddi-default	act/unsup	
1003	trcrf-default	act/unsup	
1004	fddinet-default	act/unsup	
1005	trbrf-default	act/unsup	

```
SW1#
```

FIGURE 6.4.5 – Vérification de l'exécution du playbook sur switch3 en mode server



```
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#
SW2#show vlan b
SW2#show vlan brief
```

VLAN	Name	Status	Ports
1	default	active	Gi1/1, Gi1/2, Gi1/3
2	vlan2	active	Gi0/3
3	vlan3	active	Gi1/0
4	vlan4	active	Gi0/2
99	v99	active	
1002	fddi-default	act/unsup	
1003	trcrf-default	act/unsup	
1004	fddinet-default	act/unsup	
1005	trbrf-default	act/unsup	

```
SW2#
```

FIGURE 6.4.6 – Vérification de l’exécution du playbook sur sur switch2 en mode client

```
Switch3
*****
* IOSv is strictly limited to use for evaluation, demonstration and IOS *
* education. IOSv is provided as-is and is not supported by Cisco's *
* Technical Advisory Center. Any use or disclosure, in whole or in part, *
* of the IOSv Software or Documentation to any third party for any *
* purposes is expressly prohibited except as otherwise authorized by *
* Cisco in writing. *
*****
SW3>en
SW3#show vlan br
SW3#show vlan brief

VLAN Name                Status    Ports
----
1    default              active    Gi1/1, Gi1/2, Gi1/3
2    vlan2                active    Gi0/2
3    vlan3                active    Gi0/3
4    vlan4                active    Gi1/0
99   v99                  active
1002 fddi-default          act/unsup
1003 trcrf-default        act/unsup
1004 fddinet-default      act/unsup
1005 trbrf-default        act/unsup
SW3#
```

FIGURE 6.4.7 – Vérification de l'exécution du playbook sur switch3 en mode client

6.4.3 Playbook : activation d'ospf sur un routeur cisco ios

Le but de ce playbook Ansible `playbook_ping.yml` est d'activer le protocole OSPF (Open Shortest Path First) sur un routeur Cisco IOS (R1) , et aussi ce playbook automatise la configuration initiale d'OSPF sur le routeur R1 en définissant le processus OSPF et en spécifiant les réseaux à annoncer dans l'aire 0, ce qui est essentiel pour l'établissement et la maintenance des routes dynamiques dans un réseau IP

```
GNU nano 6.2                                playbook_router_ospf.yml
---
- name: Activer OSPF sur un routeur Cisco IOS
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe

  tasks:
    - name: Configurer OSPF
      ios_config:
        lines:
          - router ospf 1
          - network 192.168.202.0 0.0.0.255 area 0
        parents: router ospf 1
      register: ospf_config
```

FIGURE 6.4.8 – Contenu du playbook `playbook_ospf.yml`

Résultat du test :

```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_router_ospf.yml

PLAY [Activer OSPF sur un routeur Cisco IOS] *****

TASK [Configurer OSPF] *****
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they appear if present in
the running configuration on device
changed: [R1]

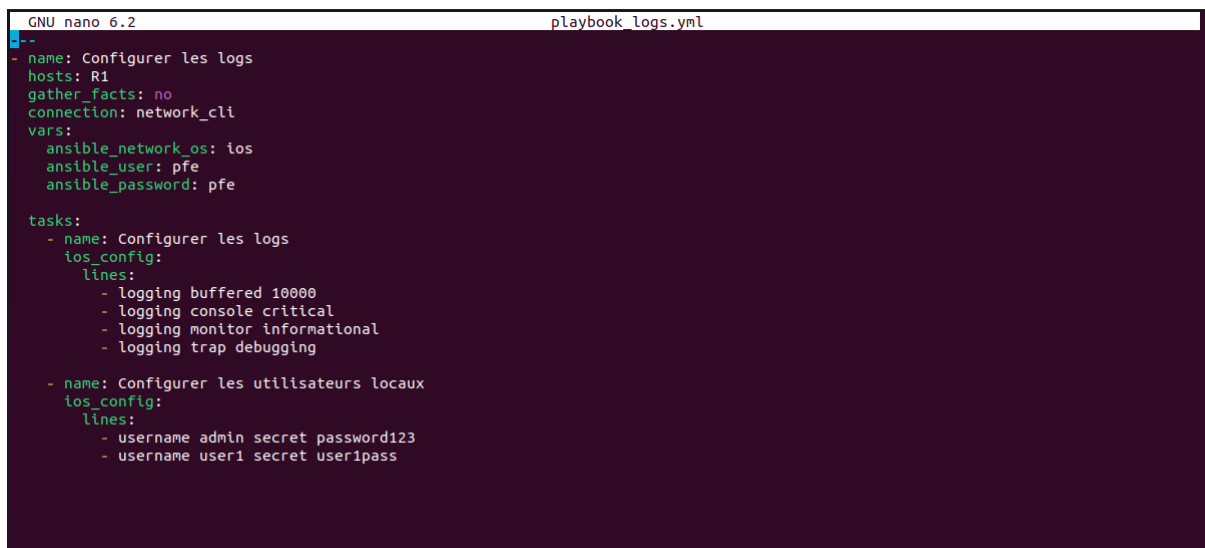
PLAY RECAP *****
R1                        : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

pfe@pfe-virtual-machine:/etc$
```

FIGURE 6.4.9 – Résultat du test pour le playbook `playbook_ospf.yml`

6.4.4 Playbook pour la Configuration des logs

Le but de ce playbook `playbook_logs.yml` est de configurer les paramètres de journalisation (logs) et de définir des utilisateurs locaux sur le routeur R1 et aussi ce playbook vise à améliorer la gestion et la sécurité du routeur R1 en configurant des options de journalisation pour surveiller les événements importants et en définissant des utilisateurs locaux avec des accès sécurisés.



```
GNU nano 6.2                                playbook_logs.yml
--
- name: Configurer les logs
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe

  tasks:
    - name: Configurer les logs
      ios_config:
        lines:
          - logging buffered 10000
          - logging console critical
          - logging monitor informational
          - logging trap debugging

    - name: Configurer les utilisateurs locaux
      ios_config:
        lines:
          - username admin secret password123
          - username user1 secret useripass
```

FIGURE 6.4.10 – Contenu du playbook `playbook_logs.yml`

Résultat du test :



```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_logs.yml

PLAY [Configurer les logs] *****

TASK [Configurer les logs] *****
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they appear if present in the running configuration on device
changed: [R1]

TASK [Configurer les utilisateurs locaux] *****
changed: [R1]

PLAY RECAP *****
R1                : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

FIGURE 6.4.11 – Résultat du test pour le playbook `playbook_logs.yml`

6.4.5 Playbook pour configurer une ACL pour filtrer le trafic sortant sur l'interface wan

Le but de ce playbook `playbook_ACL.yml` est de configurer une Liste de Contrôle d'Accès (ACL) sur l'interface WAN du routeur R1 afin de filtrer le trafic sortant. Les ACL sont utilisées pour contrôler quel type de trafic est autorisé à quitter le réseau local et à travers l'interface WAN vers Internet ou d'autres réseaux.

```
GNU nano 6.2                                playbook_acl.yml
- name: Configurer une ACL pour filtrer le trafic sortant sur l'interface WAN
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe
  tasks:
    - name: Configurer une liste de contrôle d'accès (ACL) pour le trafic sortant
      ios_config:
        lines:
          - access-list 101 deny    tcp any any eq telnet
          - access-list 101 permit tcp any any
          - access-list 101 permit udp any any
          - access-list 101 permit icmp host 192.168.202.132 any
          - access-list 101 permit icmp any any echo-reply
          - access-list 101 permit icmp any any time-exceeded
          - access-list 101 permit icmp any any unreachable
        parents: ip access-list extended 101
```

FIGURE 6.4.12 – Contenu du playbook `playbook_ACL.yml`

Résultat du test :

```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_acl.yml
PLAY [Configurer une ACL pour filtrer le trafic sortant sur l'interface WAN] *****

TASK [Configurer une liste de contrôle d'accès (ACL) pour le trafic sortant] *****
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar to how they appear if present in
the running configuration on device
changed: [R1]

PLAY RECAP *****
R1                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

FIGURE 6.4.13 – Résultat du test pour le playbook `playbook_ACL.yml`

6.4.6 Playbook Configuration snmp

Le but de ce playbook `playbook_snmp.yml` est de configurer le Service de Surveillance de Réseau Simple (SNMP) sur le routeur R1 et aussi ce playbook vise à activer et configurer le SNMP sur le routeur R1 en définissant des communautés SNMP pour l'accès aux informations de gestion et en spécifiant un hôte de gestion SNMP pour recevoir les informations SNMP générées par le routeur

```
GNU nano 6.2                                playbook_snmp.yml
---
- name: Configuration de SNMP
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe

  tasks:
    - name: Configurer le serveur SNMP
      ios_config:
        lines:
          - snmp-server community public RO
          - snmp-server community private RW
          - snmp-server host 192.168.1.100 version 2c public
```

FIGURE 6.4.14 – Contenu du playbook `playbook_snmp.yml`

Résultat du test :

```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_snmp.yml

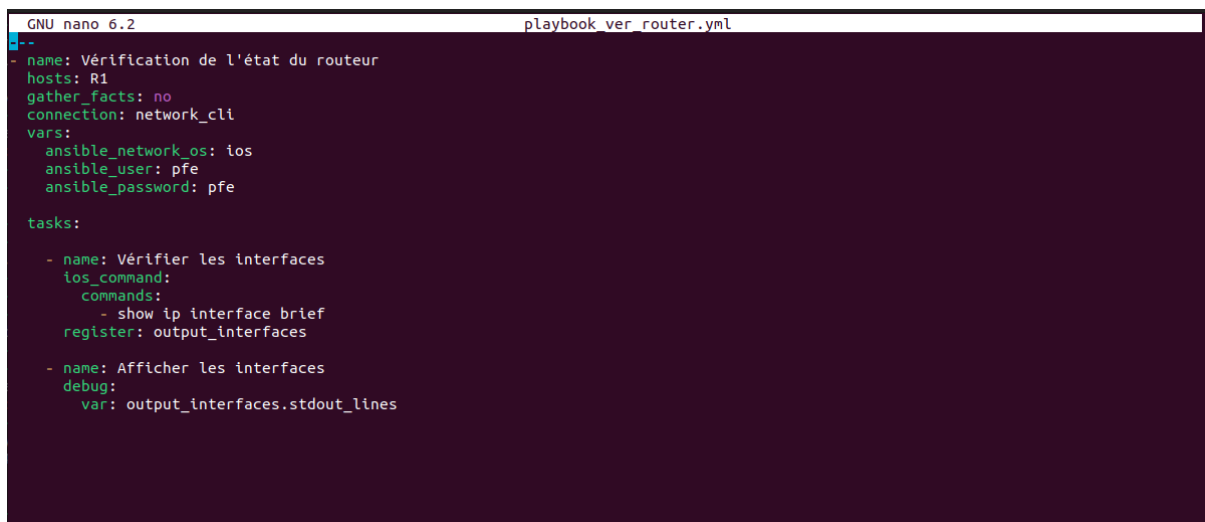
PLAY [Configuration de SNMP] *****
TASK [Configurer le serveur SNMP] *****
ok: [R1]

PLAY RECAP *****
R1                : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

FIGURE 6.4.15 – Résultat du test pour le playbook `playbook_snmp.yml`

6.4.7 Playbook pour la vérification de l'état du routeur

Le but de ce playbook `playbook_ver_router.yml` est de vérifier l'état des interfaces réseau du routeur R1 et d'afficher les résultats obtenus et aussi ce playbook permet de surveiller et d'afficher l'état des interfaces réseau du routeur R1, ce qui est essentiel pour diagnostiquer les problèmes de connectivité ou de configuration sur le réseau.



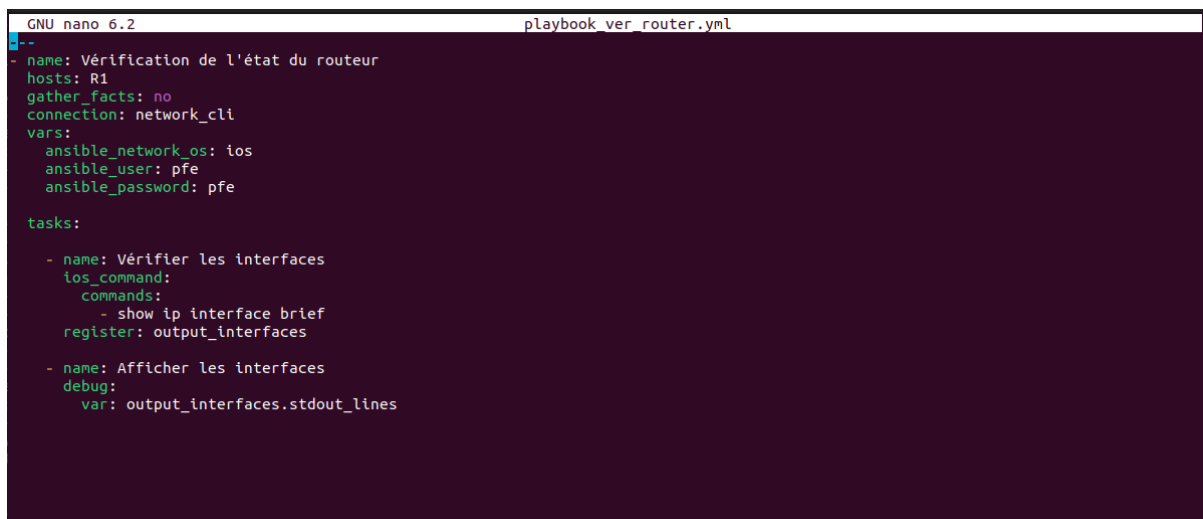
```
GNU nano 6.2                                playbook_ver_router.yml
--
- name: Vérification de l'état du routeur
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe

  tasks:
    - name: Vérifier les interfaces
      ios_command:
        commands:
          - show ip interface brief
      register: output_interfaces

    - name: Afficher les interfaces
      debug:
        var: output_interfaces.stdout_lines
```

FIGURE 6.4.16 – Contenu du playbook `playbook_ver_router.yml`

Résultat du test :



```
GNU nano 6.2                                playbook_ver_router.yml
--
- name: Vérification de l'état du routeur
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_network_os: ios
    ansible_user: pfe
    ansible_password: pfe

  tasks:
    - name: Vérifier les interfaces
      ios_command:
        commands:
          - show ip interface brief
      register: output_interfaces

    - name: Afficher les interfaces
      debug:
        var: output_interfaces.stdout_lines
```

FIGURE 6.4.17 – Résultat du playbook `playbook_ver_router.yml`

6.4.8 Playbook pour le vérification de ping

Ce playbook permet de vérifier la disponibilité des périphériques réseau accessibles depuis l'hôte R1 en utilisant des pings ICMP, ce qui est utile pour diagnostiquer rapidement les problèmes de connectivité réseau.

```
GNU nano 6.2                                playbook_ping_router.yml
--
- name: Ping hosts
  hosts: R1
  gather_facts: no
  connection: network_cli
  vars:
    ansible_user: pfe
    ansible_password: pfe
    ansible_network_os: ios
  tasks:
    - name: Ping hosts
      ping:
```

FIGURE 6.4.18 – Contenu du playbook `playbook_veri.yml` et Résultat du test

Résultat du test :

```
pfe@pfe-virtual-machine:/etc$ ansible-playbook -i inventory.ini playbook_ping_router.yml

PLAY [Ping hosts] *****
TASK [Ping hosts] *****
ok: [R1]

PLAY RECAP *****
R1                  : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

pfe@pfe-virtual-machine:/etc$
```

FIGURE 6.4.19 – Résultat du ping `playbook_ping_router.yml`

6.5 Déploiement de GIT avec GitHub

6.5.1 Introduction

Git et GitHub sont des outils essentiels pour le développement logiciel moderne. Git est un système de contrôle de version distribué qui permet de suivre les modifications du code source et de faciliter la collaboration entre les développeurs. GitHub, quant à lui, est une plateforme de développement collaboratif qui héberge des dépôts Git et offre des fonctionnalités supplémentaires telles que les issues, les pull requests et les actions GitHub pour l'intégration continue (CI/CD).



FIGURE 6.5.1 – Git



FIGURE 6.5.2 – GitHub

6.5.2 Prérequis pour le Déploiement

Avant de déployer Git et de configurer un dépôt GitHub, certains prérequis doivent être remplis :

1. Environnement :
 - Un serveur ou une machine virtuelle avec accès à Internet.
 - Accès SSH configuré pour l'utilisateur avec privilèges sudo.
2. Logiciels :
 - Git doit être installé sur le serveur.
 - Un compte GitHub doit être créé et configuré.

6.5.3 Installation et Configuration de Git

```
pfe@pfe-virtual-machine:/$ sudo apt install git
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
git est déjà la version la plus récente (1:2.34.1-1ubuntu1.10).
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 11 non mis à jour.
```

FIGURE 6.5.3 – Installation de Git

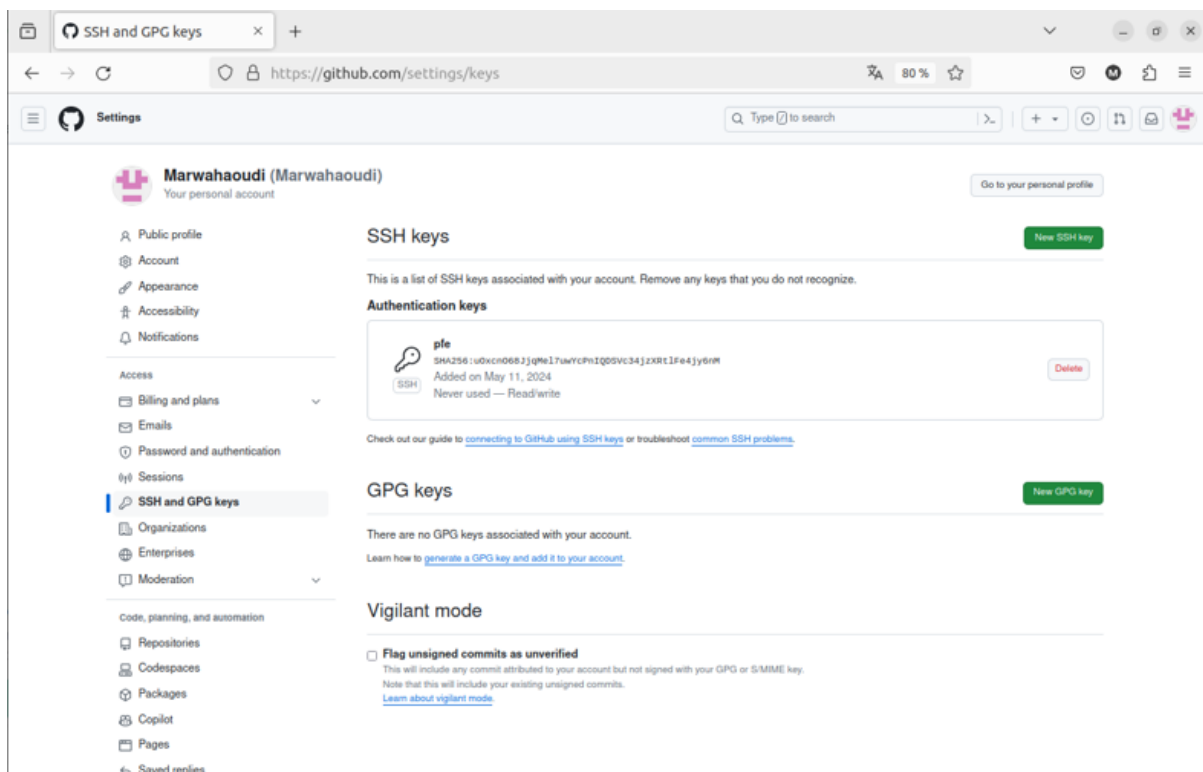


FIGURE 6.5.4 – Connection SSH entre github et notre environnement

6.5.4 Configuration initiale de Git :

Cette configuration permet à Git de connaître votre identité, ce qui est utile pour l'historique des commits.

```
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git config --global user.name "Marwahaoudi"
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git config --global user.email "marouahaoudi@gmail.com"
```

FIGURE 6.5.5 – Configuration initiale de Git

Ensuite on crée une branche dev où on sauvegarde l'ensemble de nos fichiers. Cette branche sert à stocker les différentes versions de notre fichier avant de les migrées dans la branche principale master.

```
pfe@pfe-virtual-machine:/$ sudo git clone https://github.com/rhabatbir/PFE_Automatisation_Configuration_Ansible_EVE-NG.git
Clonage dans 'PFE_Automatisation_Configuration_Ansible_EVE-NG'...
Username for 'https://github.com': Marwahaoudi
Password for 'https://Marwahaoudi@github.com':
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 16 (delta 3), reused 0 (delta 0), pack-reused 0
Réception d'objets: 100% (16/16), 2.70 Mio | 3.34 Mio/s, fait.
Résolution des deltas: 100% (3/3), fait.
pfe@pfe-virtual-machine:/$ ls
bin      lib32    PFE_Automatisation_Configuration_Ansible_EVE-NG  swapfile
boot     lib64    proc                                              sys
cdrom    libx32   root                                             tmp
dev      lost+found  run                                              usr
etc      media    sbin                                            var
home     mnt      snap
lib      opt      srv
pfe@pfe-virtual-machine:/$ cd PFE_Automatisation_Configuration_Ansible_EVE-NG/
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ ls
les.etapes.de.l.installation.2.pdf  README.md
'rapport_de_projet_fin_d_etude (5).pdf'
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git config --global user.name "Marwahaoudi"
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git config --global user.email "marouahaoudi@gmail.com"
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ cat ~/.gitconfig
[user]
    name = Marwahaoudi
    email = marouahaoudi@gmail.com
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ ls
les.etapes.de.l.installation.2.pdf  README.md
'rapport_de_projet_fin_d_etude (5).pdf'
pfe@pfe-virtual-machine:/PFE_Automatisation_Configuration_Ansible_EVE-NG$ history
1 sudo apt install python3 python3-pip
2 sudo apt install python3 python3-pip
3 sudo mount /dev/cdrom /cdrom
4 sudo apt install python3 python3-pip
5 sudo pip3 install ansible
6 ansible --version
```

FIGURE 6.5.6 – Vérification de l'exécution du playbook sur le périphérique

Maintenant les fichiers sont bien sauvegardés dans la branche dev.

```

pfe@pfe-virtual-machine:/etc/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git commit -m "Ajout du mon_playbook_switch.yml"
[main 27a9167] Ajout du mon_playbook_switch.yml
1 file changed, 18 insertions(+)
create mode 100644 mon_playbook_switch.yml
pfe@pfe-virtual-machine:/etc/PFE_Automatisation_Configuration_Ansible_EVE-NG$ git push
Username for 'https://github.com': Marwahaoudi
Password for 'https://Marwahaoudi@github.com':
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression par delta en utilisant jusqu'à 2 fils d'exécution
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 503 octets | 503.00 Kio/s, fait.
Total 3 (delta 1), réutilisés 0 (delta 0), réutilisés du pack 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rihabatbir/PFE_Automatisation_Configuration_Ansible_EVE-NG.git
8738c2f..27a9167 main -> main

```

FIGURE 6.5.7 – Vérification de l'exécution du playbook sur le périphérique

6.6 Création et gestion des workflows GitHub Actions

Introduction à GitHub Actions

6.6.1 Présentation de GitHub Actions

GitHub Actions est une fonctionnalité intégrée à GitHub qui permet aux développeurs d'automatiser des tâches de développement telles que l'intégration continue (CI) et le déploiement continu (CD). Avec GitHub Actions, vous pouvez créer des workflows personnalisés pour automatiser la construction, le test, et le déploiement de votre code directement depuis votre dépôt GitHub.

Les workflows sont définis à l'aide de fichiers YAML, et ils peuvent être déclenchés par divers événements, tels que des commits, des pull requests, ou des actions planifiées.

6.6.2 Avantages et cas d'utilisation

Avantages :

1. Intégration étroite avec GitHub :

Les workflows sont définis et gérés directement dans le dépôt GitHub. Accès facile aux événements GitHub tels que les commits, les issues, et les pull requests.

2. Flexibilité et personnalisation :

Les workflows peuvent être entièrement personnalisés pour répondre aux besoins spécifiques des projets. Utilisation de scripts personnalisés ou d'actions disponibles sur GitHub Marketplace.

3. Écosystème riche :

Accès à une vaste bibliothèque d'actions créées par la communauté et disponibles sur GitHub Marketplace. Possibilité de créer et partager des actions personnalisées.

4. Automatisation complète du cycle de vie du développement :

Prise en charge de l'intégration continue (CI), du déploiement continu (CD), et de nombreuses autres automatisations. Capacités de déploiement sur diverses plateformes (cloud, serveurs, etc.).

Cas d'utilisation :

1. Intégration continue (CI) :

Construction et test automatiques du code à chaque commit ou pull request.
Validation des modifications avant leur intégration dans la branche principale.

2. Déploiement continu (CD) :

Déploiement automatique des applications après validation des tests. Prise en charge de divers environnements de déploiement (staging, production, etc.).

3. Automatisation des tâches répétitives :

Exécution de scripts de maintenance, mise à jour des dépendances, nettoyage des branches obsolètes. Gestion automatique des versions et des releases.

4. Assurance qualité et sécurité :

Intégration d'outils de linting, de formatage, et d'analyse statique du code.
Exécution de tests de sécurité et de vérifications de conformité.

5. Collaboration et revue de code :

Automatisation des processus de revue de code, comme l'ajout de labels ou l'affectation de reviewers. Notification automatique des équipes via Slack, email, ou autres outils de communication.

6. Gestion de projet :

Automatisation de la création et de la gestion des issues et des projets GitHub.
Mise à jour automatique des tableaux de bord et des rapports de progression.

6.6.3 Intégration de GitHub Actions et Molecule pour le CI/CD

L'automatisation des tests et des déploiements est essentielle pour garantir la qualité et la fiabilité des applications. GitHub Actions permet de définir des workflows CI/CD, tandis que Molecule est un outil de test pour Ansible. Cette présentation explique comment intégrer ces deux outils en utilisant des fichiers de configuration spécifiques.

6.6.4 Structure des Répertoires

1. `.github/workflows/ci-cd.yml` : Contient la définition des workflows GitHub Actions.
2. `molecule/default/molecule.yml` : Contient la configuration des scénarios de test Molecule.

6.6.5 Fichier ci-cd.yml de GitHub Actions

Objectif du Flux de Travail Ce flux de travail vise à automatiser les tests et le déploiement de configurations Ansible à chaque fois qu'un changement est apporté à la branche principale (main) du dépôt GitHub. Il se compose de deux jobs principaux :

Job de Test (test) :

Ce job est responsable de vérifier la syntaxe des playbooks Ansible, d'exécuter des tests automatisés sur les rôles Ansible à l'aide de Molecule, et de signaler tout problème potentiel avant le déploiement.

Job de Déploiement (deploy) :

Conditionné par le succès du job de test, ce job déploie les configurations validées en production en utilisant un playbook Ansible spécifique .

Déclencheurs :

Le flux de travail est déclenché à chaque push ou pull request sur la branche main, garantissant ainsi que tout changement est testé avant d'être fusionné dans la branche principale.

```

11 jobs:
12   test:
13     runs-on: ubuntu-latest
14
15     steps:
16       - name: Checkout code
17         uses: actions/checkout@v2
18
19       - name: Set up Python
20         uses: actions/setup-python@v2
21         with:
22           python-version: '3.10'
23
24       - name: Install dependencies
25         run: |
26           python -m pip install --upgrade pip
27           pip install ansible ansible-lint molecule[docker] docker
28
29       - name: Install Ansible Galaxy roles
30         run: ansible-galaxy install -r molecule/default/requirements.yml
31
32       - name: Lint Ansible playbooks
33         run: ansible-lint .
34
35       - name: Test Ansible roles with Molecule
36         run: molecule test
37
38   ~

```

FIGURE 6.6.1 – Job de Test

```

38 deploy:
39   needs: test
40   runs-on: ubuntu-latest
41   if: github.ref == 'refs/heads/main'
42
43   steps:
44     - name: Checkout code
45       uses: actions/checkout@v2
46
47     - name: Set up Python
48       uses: actions/setup-python@v2
49       with:
50         python-version: '3.10'
51
52     - name: Install Ansible
53       run: |
54         python -m pip install --upgrade pip
55         pip install ansible
56
57     - name: Deploy to production
58       run: ansible-playbook -i inventory/production playbook_ping_router.yml
59       env:
60         ANSIBLE_VAULT_PASSWORD: ${ secrets.ANSIBLE_VAULT_PASSWORD }

```

FIGURE 6.6.2 – Job de Déploiement (deploy)

```
1  name: CI/CD Pipeline
2
3  on:
4    push:
5      branches:
6        - main
7    pull_request:
8      branches:
9        - main
```

FIGURE 6.6.3 – Déclencheurs

6.6.6 Fichier molecule.yml de Molecule

Ce fichier configure les tests de Molecule. Molecule (molecule.yml) :

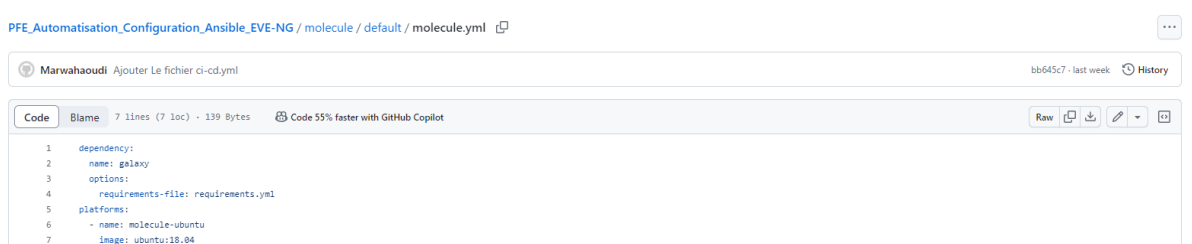


FIGURE 6.6.4 – test

- Dépendances : Utilisation de Galaxy pour gérer les dépendances avec un fichier requirements.yml.
- Plateformes : Définition des plateformes de test, ici ubuntu :18.04.

Avantages de l'Intégration

- Automatisation : Réduction des tâches manuelles répétitives et amélioration de la productivité.
- Qualité : Assurance que les changements sont testés et validés avant d'être fusionnés.
- Cohérence : Maintien d'une norme de qualité uniforme dans les configurations et les déploiements.

Dans GitHub Actions, l'exécution ressemblerait à la capture d'écran ci-dessous :

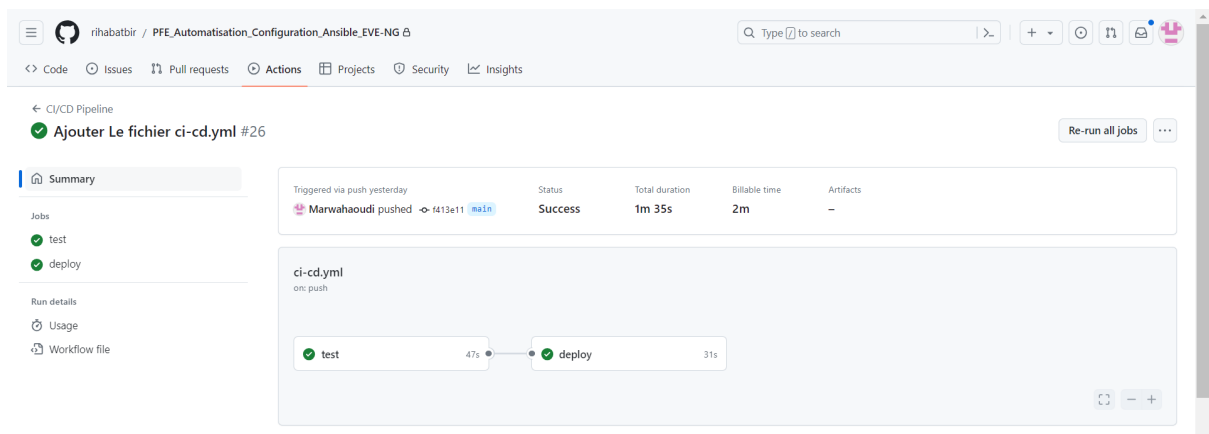


FIGURE 6.6.5 – test

Conclusion

Notre projet démontre les avantages significatifs de l'utilisation d'Ansible pour l'automatisation des réseaux. En automatisant la configuration des dispositifs réseau, nous avons atteint une cohérence, une efficacité et une évolutivité accrues. Les configurations standardisées ont minimisé les variations et garanti des déploiements uniformes.

L'efficacité opérationnelle s'est améliorée, les tâches répétitives étant exécutées de manière fiable et rapide, libérant ainsi des ressources humaines pour des tâches stratégiques et réduisant le temps d'indisponibilité.

L'utilisation d'Ansible a facilité la gestion de notre infrastructure en croissance, permettant des déploiements et des modifications à grande échelle avec simplicité et rapidité. Cette automatisation a transformé la gestion des réseaux, la rendant plus fiable et réduisant les erreurs humaines, tout en posant les bases d'une gestion plus proactive et agile.

Bibliographie

- [1] Andrew Tanenbaum (Université libre d'Amsterdam), « Systèmes d'exploitation », 3ème édition, Nouveaux Horizons, 2008.
- [2] <http://www.linux.org/>
- [3] https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html / <https://www.redhat.com/en/topics/automation/what-is-configuration-management>
- [4] <https://www.eve-ng.net/index.php/documentation/howtos/>