DEDICACE

ABBREVIATION	SIGNIFICATION
SSH	Secure Shell
PING	Packet Internet Groper
YML	YAML Ain't Markup Language (ou Yet Another Markup Language)
IP	Internet Protocol
ISE	Identity Services Engine
ESXI	Elastic Sky X (Hyperviseur de VMware)
IPS	Intrusion Prevention System
CLI	Command Line Interface
GNS	Graphical Network Simulator
SDN	Software Defined Networking
ACL	Access Control List
VLAN	Virtual Local Area Network
DSL	Digital Subscriber Line
ZMQ	ZeroMQ (ou ØMQ, une bibliothèque de messagerie)
AWS	Amazon Web Services
VPN	Virtual Private Network

QEMU	Quick Emulator
INI	Initialization File
IOS	Internetwork Operating System (Système d'exploitation de Cisco)
ICMP	Internet Control Message Protocol
ANSIBLE	Outil d'automatisation open source
MOTD	Message of the Day
RSA	Rivest-Shamir-Adleman (algorithme de cryptographie)
VTY	Virtual Teletype
VAULT	Ansible Vault (outil de chiffrement d'Ansible)
RGPD	Règlement Général sur la Protection des Données

RESUME

Ce rapport relate mon stage au sein du groupe **ISARE**, une académie spécialisée dans la formation et la certification en réseaux, ainsi que dans les services TIC. Ce stage s'est déroulé du 10 juillet 2024 au 29 septembre 2024, avec pour objectif de me familiariser avec les infrastructures réseau de l'entreprise et de contribuer à l'amélioration de leurs processus internes. Dès le début, j'ai été immergé dans l'environnement d'ISARE, où j'ai découvert ses méthodes de travail, ses besoins techniques et son fonctionnement en général. Après cette phase d'intégration, j'ai été chargé de plusieurs tâches pratiques liées à **la gestion et à l'optimisation des infrastructures réseau.** Cela m'a permis de développer de nouvelles compétences et d'acquérir une meilleure compréhension des défis rencontrés dans la gestion des réseaux. Tout au long du stage, j'ai collaboré avec les équipes techniques d'ISARE, ce qui m'a permis d'améliorer ma capacité à travailler en équipe et de renforcer mes connaissances dans le domaine des réseaux. Ce rapport retrace l'évolution de mon expérience, depuis la phase d'observation jusqu'à la réalisation des projets concrets, en détaillant les compétences acquises et les enseignements tirés de cette immersion professionnelle.

Mots clés :

Isare : **Institut Supérieur D'administration Des Réseaux D'entreprises**

Tic : Technologie de l'information et de la communication

ABSTRACT

This report outlines my internship experience at **ISARE**, a training and certification academy specializing in network technologies and ICT services, which took place from July 22, 2023, to September 29, 2023. The primary goal of the internship was to gain practical knowledge of network infrastructure management and to contribute to enhancing internal processes. During the initial phase, I became familiar with the operational environment of ISARE, understanding its technical needs and workflows. As I progressed, I was involved in various tasks focused on optimizing network infrastructure, allowing me to develop essential skills and a deeper understanding of the challenges faced in network management. Collaboration with ISARE's technical teams provided valuable insights and improved my ability to work within a team. This report details the evolution of my internship, highlighting the transition from observation to active participation in concrete projects, as well as the skills acquired and lessons learned throughout this professional immersion.

Keywords :

Isare: **Higher Institute of Enterprise Network Administration**

Tic: **Technology of Information and Communication**

INTRODUCTION GENERALE

Dans un contexte où les infrastructures réseaux deviennent de plus en plus complexes et où les exigences de rapidité et de fiabilité des déploiements augmentent, l'automatisation s'impose comme une solution indispensable pour optimiser la gestion des systèmes informatiques. L'automatisation permet non seulement de réduire les erreurs humaines, mais aussi de gagner en efficacité opérationnelle tout en assurant une plus grande cohérence dans la configuration des équipements. Ansible, un outil open-source d'automatisation, se distingue par sa simplicité d'utilisation et sa capacité à gérer des infrastructures complexes de manière centralisée. Basé sur une approche déclarative, Ansible permet d'automatiser des tâches variées telles que la configuration des routeurs et switches, la gestion des VLANs, ou encore la sécurisation des équipements réseau. Grâce à son architecture sans agent et à l'utilisation de fichiers YAML pour la définition des playbooks, Ansible facilite la gestion des configurations tout en étant hautement extensible. Ce rapport a pour objectif de détailler le processus d'implémentation d'Ansible dans un environnement réseau, ainsi que les différents scénarios d'automatisation qui y ont été appliqués. À travers l'analyse de cas pratiques, nous démontrons comment Ansible permet de simplifier les tâches réseau courantes, d'assurer une gestion centralisée et d'améliorer la sécurité des configurations, tout en optimisant les performances globales du réseau.

PARTIE I : PHASE D'INSERTION

CHAPITRE I : PRESENTATION DE L'ENTREPRISE ISARE

PARTIE 2 : PHASE TECHNIQUE

CHAPITRE I : ANALYSE DU PROJET

RÉSUMÉ

Pour garantir le succès optimal d'un projet, il est impératif d'examiner en profondeur tous ses aspects et caractéristiques. Dans ce chapitre, nous analysons le contexte dans lequel notre projet sera mis en œuvre, dans le but de définir ses contraintes et d'identifier la problématique qui en découle.

SOMMAIRE :

INTRODUCTION

1.1 PRESENTATION DU PROJET

1.2. ETUDE DE L'EXISTANT

1.3. CRITIQUE DE L'EXISTANT

1.4. PROBLEMATIQUE

CONCLUSION

INTRODUCTION

Le cahier des charges est un document exhaustif détaillant et décrivant le sujet en question. Il reflète une étroite collaboration entre les utilisateurs et le responsable du projet, avec pour objectif principal de satisfaire pleinement l'ensemble des parties prenantes impliquées de près ou de loin dans la conception du projet. Dans ce chapitre, notre démarche consistera à exposer de manière précise les objectifs à atteindre et les solutions à mettre en œuvre dans le cadre de notre projet, ainsi que tous les autres éléments nécessaires pour assurer sa réalisation réussie.

1.1 PRESENTATION DU PROJET

L'automatisation des processus IT est devenue un enjeu majeur pour les entreprises cherchant à optimiser leurs opérations et à minimiser les erreurs humaines. ISARE, une académie spécialisée dans les formations en réseaux et technologies de l'information, gère une infrastructure réseau complexe comprenant des équipements variés tels que des routeurs, commutateurs, pare-feu, et points d'accès Wi-Fi. Jusqu'à présent, la configuration et la gestion de ces équipements ont été réalisées manuellement, ce qui expose l'infrastructure à des risques d'erreurs, d'incohérences dans les configurations, et à des délais prolongés pour le déploiement de nouvelles configurations. Pour répondre à ces défis, ISARE a décidé de se tourner vers l'automatisation en utilisant Ansible, un outil puissant et flexible capable de gérer et d'automatiser les tâches sur l'ensemble de l'infrastructure réseau. Ce projet vise à standardiser les configurations, réduire les erreurs humaines, et accélérer les processus de gestion réseau, permettant ainsi à ISARE d'améliorer la fiabilité, la sécurité et l'efficacité de son infrastructure tout en libérant les administrateurs réseau pour des tâches à plus forte valeur ajoutée.

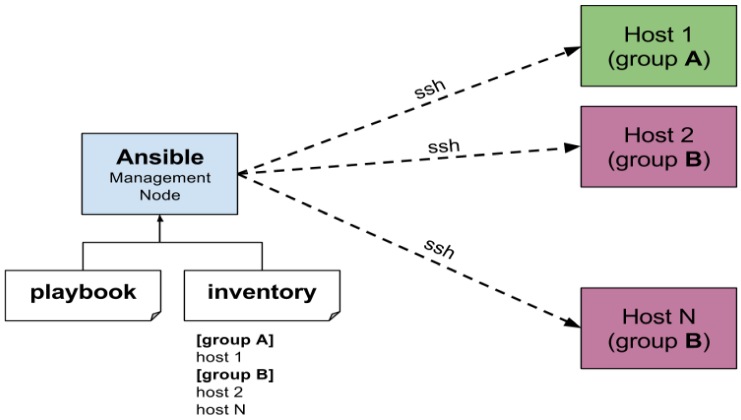


Figure 1 : Fonctionnement ansible

1.2 ETUDE DE L'EXISTANT

L'académie **ISARE**, renommée pour ses formations en réseaux et technologies de l'information, dispose d'une infrastructure réseau robuste et diversifiée pour soutenir ses activités pédagogiques et administratives. Cette infrastructure est essentielle pour la qualité des formations dispensées, notamment celles certifiées par des organismes comme Cisco, Microsoft, et Linux. ISARE gère une gamme variée d'équipements réseau et de systèmes informatiques, qui forment la colonne vertébrale de son environnement technologique.

1.2.1 Architecture du réseau existant

Le réseau informatique est un ensemble d'équipement interconnectés dans le but du partage des informations et du matériel. Quel que soit le type de système informatique utilisé au sein d'une entreprise, aujourd'hui il est obligatoire d'interconnecter les équipements informatiques pour constituer un réseau et voir même aussi les différentes succursales. *La figure 3* nous montre l'architecture du réseau local de l'institut ISARE.

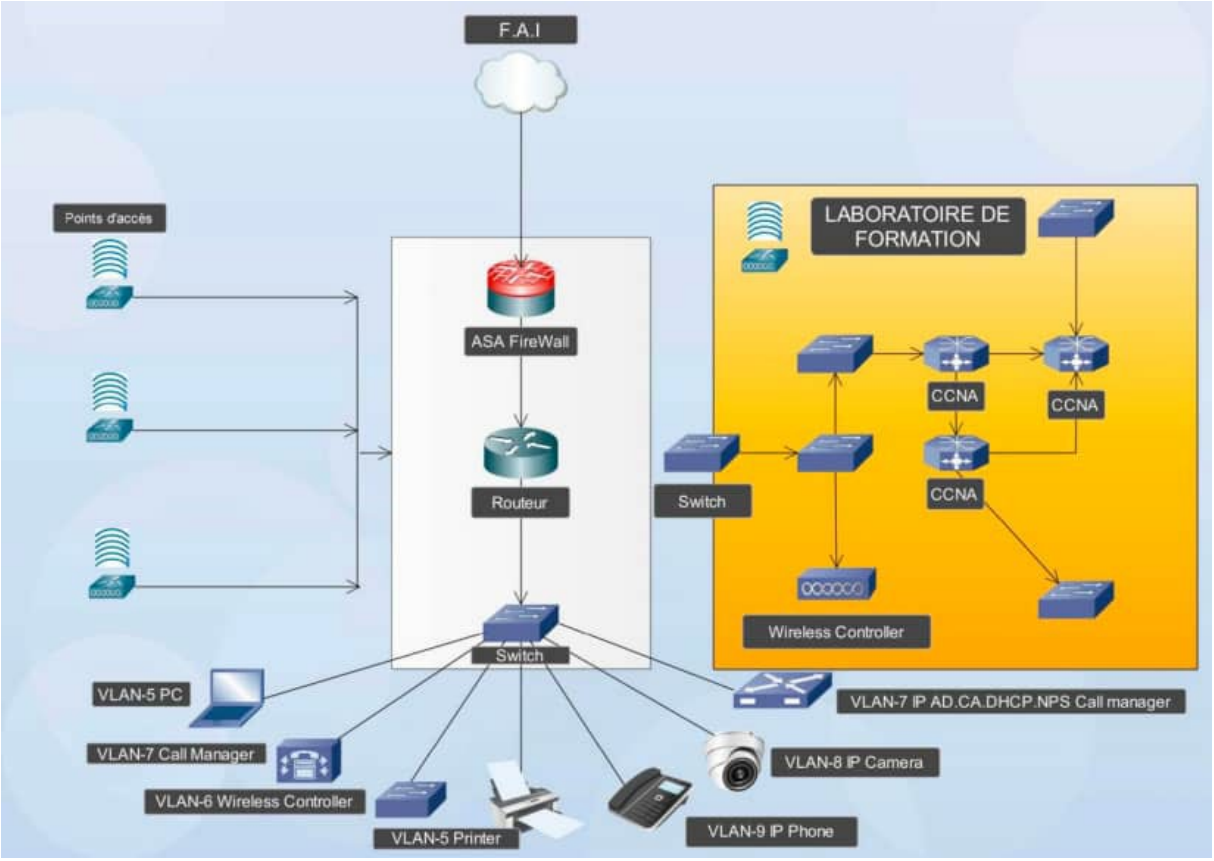


Figure 2 : Architecture D'IZARE

Dans son architecture réseau nous avons :

- **Le pare-feu ASA** : Définit comme un système de sécurité de réseau informatique qui limite le trafic internet entrant, sortant ou à l'intérieur du réseau local ; Le pare-feu ASA pour Adaptive Security Appliance est un pare-feu de nouvelle génération de marque Cisco, il offre entre autres un Système complet de prévention des intrusions (IPS), une protection complète contre les menaces et les programmes malveillants évolué, dispositif autonome adapté aux petites et moyennes entreprises etc. L'institut ISARE dispose d'un pare-feu cisco ASA 5520 ;
- **Le routeur** : Cet équipement réseau permet l'interconnexion de deux ou plusieurs sous réseaux tout en assurant le routage des différents paquets ; L'institut ISARE dispose de plusieurs routeurs de marque Cisco (cisco 2800 et 2900) ;
- **Le switch** : Encore appelé (commutateur) permet de relier plusieurs segments du réseau et d'acheminer les trames sur le bon segment de destination grâce aux adresses contenues dans la table CAM du commutateur. L'institut ISARE dispose de plusieurs switch (couche 3) Catalyst model 3750 avec 48 et 24 ports réseaux PoE ;
- **Des points d'accès :** Pour Access Point en anglais est un équipement informatique permettant de créer un réseau sans fil local et de permettre à tout équipements sans fil de se connecter à internet. L'institut ISARE dispose de plusieurs ;
- **Des ordinateurs portables** : Ses appareils informatiques dédié aux traitements de l'information constitué de plusieurs parties telles la partie logiciel (software) et la partie matériel (hardware) ; L'institut ISARE dispose de plusieurs ordinateurs constitués de marque DELL et HP dont les espaces de stockage varient entre 112 Go à 512 Go ayant comme système d'exploitation Windows 11 et kali et des progiciels comment Windows office 2019 pour le traitement de texte ;

- **Des imprimantes** : Cet équipement est un appareil permettant d'obtenir un document sur papier à partir d'un modèle informatique du document visualiser grâce à un logiciel de traitement de texte ; L'institut ISARE possède deux imprimantes de marque canon ;
- **Des téléphones IP** : Pour la communication entre les différents bureaux ;
- **Des caméras de surveillance** : Ce sont des appareils installés dans un espace, qui permet de visualiser tout ce qui se passe dans cette espace grâce à un écran en passant par un enregistrer ; L'institut ISARE dispose de plusieurs caméras IP de caractéristiques Dôme motorisé ;
- **Des contrôleurs de point d'accès** : Ce sont des équipements informatiques permettant de prendre la bande passante qui provient d'un routeur et de l'étendre pour une adaptation aux besoins du réseau ; L'institut ISARE possède en seule.

1.2.2 Description de l'existant

Actuellement, la gestion de l'infrastructure réseau d'ISARE repose sur des processus manuels. Les administrateurs réseau configurent individuellement chaque équipement via des interfaces CLI (Command Line Interface) ou des interfaces graphiques propres à chaque dispositif. Ce processus inclut la configuration des VLANs, l'application des règles de pare-feu, et la gestion des accès sans fil.

1.3 CRITIQUE DE L'EXISTANT

Bien que la gestion manuelle de l'infrastructure réseau d'ISARE permette un contrôle précis sur les configurations, elle présente des limitations notables. La nécessité de configurer chaque équipement individuellement à l'aide d'interfaces CLI ou graphiques rend le processus long et complexe. Cette approche peut entraîner des incohérences et des erreurs, notamment dans les configurations des VLANs, des règles de pare-feu et des paramètres d'accès sans fil. La gestion manuelle est également moins efficace pour appliquer des mises à jour ou des modifications à grande échelle, ce qui peut retarder les réponses aux besoins évolutifs de l'infrastructure. En outre, le temps et les ressources nécessaires pour gérer ces configurations manuelles peuvent réduire la capacité de l'équipe IT à se concentrer sur des tâches stratégiques, soulignant la nécessité d'une approche plus automatisée pour améliorer l'efficacité et la cohérence.

1.4 PROBLEMATIQUE

La problématique centrale de ce projet réside dans l'efficacité et la fiabilité de la gestion de l'infrastructure réseau d'ISARE. Actuellement, la configuration manuelle des équipements réseau, tels que les routeurs, commutateurs, et pare-feu, est sujette à des erreurs humaines et à des incohérences, ce qui peut compromettre la stabilité et la sécurité du réseau. De plus, cette approche entraîne des délais significatifs pour l'application des mises à jour et des modifications, limitant la réactivité et l'efficacité opérationnelle de l'équipe IT. Comment peut-on automatiser la configuration et la gestion de l'infrastructure réseau d'ISARE en utilisant Ansible pour améliorer la cohérence, réduire les erreurs humaines, et accélérer les processus de gestion tout en optimisant les ressources disponibles ?

CONCLUSION

la gestion manuelle de l'infrastructure réseau d'ISARE présente des défis en matière d'efficacité et de cohérence. L'introduction de l'automatisation via Ansible pourrait considérablement améliorer la précision et la rapidité des opérations réseau. Pour avancer vers cette solution, un cahier des charges détaillé est essentiel pour définir les exigences et les objectifs du projet. Ce document servira de guide pour l'élaboration et la mise en œuvre de l'automatisation, garantissant que les besoins spécifiques d'ISARE sont pleinement pris en compte.

CHAPITRE II : CAHIER DE CHARGE

RÉSUMÉ

Le cahier des charges définit les objectifs pour l'automatisation de la configuration réseau d'ISARE avec Ansible. Il précise les exigences en matière de gestion, de sécurité et de cohérence des configurations réseau, tout en mettant l'accent sur la simplicité d'utilisation et l'efficacité opérationnelle.

SOMMAIRE :

INTRODUCTION

2.1 CONTEXTE

2.2 OBJECTIFS

2.3 EVALUATION FINANCIERE DU PROJET

2.4 PLANIFICATION DU PROJET

2.5 LES LIVRABLES

CONCLUSION

INTRODUCTION

Un cahier de charges (CDC) est un document qui doit être scrupuleusement respecté lors de la conception et la réalisation d'un projet. Le cahier de charges est donc un ensemble d'indications et de consignes pour mener à bien un projet. Il présente un certain nombre de spécificités permettant la définition et la compréhension du sujet. Il sera donc question pour nous ici de ressortir le contexte de notre étude, le planifier, l'évaluer financièrement et enfin dégager les livrables de celui-ci.

2.1 CONTEXTE

Dans un environnement technologique en constante évolution, l'automatisation des processus informatiques devient une priorité pour les entreprises souhaitant gagner en efficacité, en sécurité, et en compétitivité. **ISARE**, spécialisée dans la gestion de services réseaux, fait face à une croissance rapide de son infrastructure et à des besoins de plus en plus exigeants en matière de gestion des configurations réseau.

Afin de relever ces défis, **ISARE** a décidé d'adopter une approche plus moderne en automatisant la configuration et la gestion des équipements réseau à l'aide **d'Ansible**, un outil d'automatisation . Ce projet vise à mettre en place une infrastructure réseau plus flexible, capable de répondre rapidement aux besoins de l'entreprise tout en réduisant les erreurs et en améliorant la cohérence des configurations.

2.2 OBJECTIF GENERAL

Ce projet d'automatisation aura pour objectifs :

- ✓ De faciliter le déploiement et la maintenance des configurations réseau.
- ✓ D'assurer une gestion centralisée des équipements et une meilleure sécurité.
- ✓ D'optimiser les ressources humaines en réduisant les tâches répétitives et chronophages.

2.2.1 OBJECTIFS SPECIFIQUES

- ➔ Configurer automatiquement les équipements réseau
- ➔ Uniformiser toutes les configurations réseau
- ➔ Déployer rapidement les modifications
- ➔ Surveiller et réparer automatiquement les problèmes

- ➔ Faciliter l'ajout de nouveaux équipements**
- ➔ Renforcer la sécurité
- ➔ Réduire le temps de maintenance

2.3 PLANIFICATION DES TACHES

La planification de projet correspond à l'organisation des tâches à réaliser sur une période donnée. L'objectif de la planification est de déterminer le coût, les ressources mobilisées et la meilleure manière d'ordonnancer toutes les tâches à effectuer.

- **Accueil et intégration en entreprise** : Durant cette période nous avons pris le temps de bien comprendre notre environnement ;
- **Rédaction du dossier d'insertion** ;
- **Etude de l'architecture réseau en place dans l'entreprise** : Il était question ici de comprendre le fonctionnement du réseau de l'entreprise dans son ensemble ;
- **Réalisation du cahier de charges** : Nous avons durant cette période évaluer les ressources matériel, humaine et logiciel nécessaire pour la réalisation du projet.
- **Validation du cahier de charges** ;
- **Implémentation de la solution** : C'est à cette étape que nous avons eu à effectuer nos différentes installation et configuration afin de mettre sur pied la solution ;
- **Test et corrections** : Ici nous avons réalisé plusieurs simulations d'attaques afin de relever les manquements et essayer de les ressourdre ;
- **Finalisation de la rédaction du rapport** : Il s'agissait ici de l'étape finale qui consistait à rédiger le reste de notre rapport de stage en respectant le cahier de charges fourni par l'établissement ;

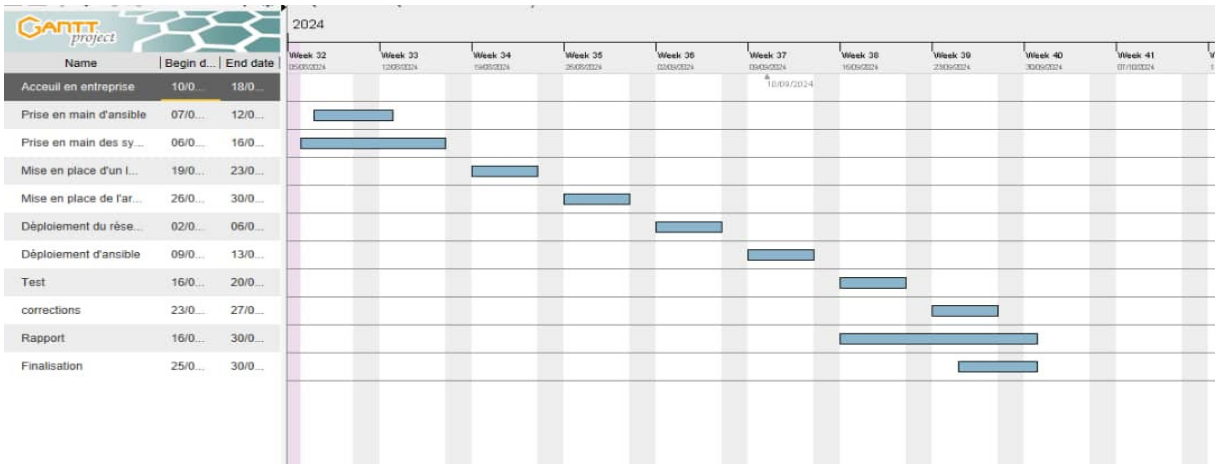


Figure 3 : Diagramme de Gantt

2.4 ESTIMATION ET CONTRAINTES DU PROJET

Parmi les différentes étapes de gestion d'un projet, l'évaluation du coût d'un projet est le plus important et le plus délicat. L'évaluation financière est l'estimation de la valeur (c'està-dire du prix potentiel) du matériel sur le marché.

2.4.1 CONTRAINTES DU PROJET

Outils de travail	Aperçus	Description
Machine hôte		20 GO de RAM 512 SSD I7 8th gen
GNS 3	GNS3	Emulateur réseau pour tester la configuration des équipements comme les routeurs et les commutateurs avant leur mise en production.
GNS VM	GNS3 VM	Control les ressources des équipements dans GNS 3
Une machine ubuntu		Machine de virtualisation C'est la machine de contrôle qui exécutera les scripts Ansible.
Vim, **Nano** ou **VS Code**		pour écrire et éditer les playbooks.
VMware Workstation		Pour virtualiser les systèmes d'exploitation comme Ubuntu (utilisé pour Ansible) et Windows.

Gantt Project		Planificateur de projet
Microsoft office		Rédaction du rapport
Modules ansible		• Version Ansible **2.16.3** ou ultérieure installée sur la machine de contrôle (généralement Ubuntu). •**SSH** activé sur les équipements cibles pour permettre la communication entre Ansible et ces appareils via des scripts automatisés.

Source : www.google.com

2.4.2 EVALUATION FINANCIERE DU PROJET

- Estimation financière des ressources logicielles du projet

  Le tableau ci-après présente le bilan des logiciels utilisés pour la réalisation de ce travail**.**

Tableau 1 : Tableau 5 Estimation des ressources logicielles

DESIGNATION	DESCRIPTION	COUT (FCFA)
Linux	Système d'exploitation	Gratuit
Windows 10	Système d'exploitation	Préinstaller sur PC
Microsoft Office 2021	Editeur de texte /	45000/ans
Gantt Project	Logiciel de planification	Gratuit
VMware Workstation	Logiciel de virtualisation	163000
TOTAL		**207 000 FCFA**

b. Estimation financière des ressources matérielles du projet

Le tableau ci-dessous présente le matériel utilisé pour la réalisation de ce travail.

Tableau 2 : Estimation financière des ressources matérielles

DESIGNATION	UTILITE	Qté	PRIX UNITAIRE (FCFA)	PRIX TOTAL (FCFA)
Unité centrale **HP CORE I7 PRO 300G6** 512Go ssd, 20Go RAM, 04 NIC (Network Interface Card)	Support sur lequel sera installé PfSense	01	300000	300000
Routeur DLINK DI804HV	Pour l'accès à Internet	01	72 450	72 450
TOTAL	**963 450 FCFA**			

Source : Mercurial des prix

c. Estimation financière des ressources humaines du projet

Tableau 3 : tableau de références des prix

REFERENCE. MERCURIALE 2023	DESIGNATION	JOURS	NOMBRE	PRIX/ JOUR	PRIX TOTAL
34-001-190019	**Chef de projet Ingénieur des travaux informatiques**	**07**	**01**	**250 000 FCFA**	**500 000 FCFA**
34-003-180006	**Technicien Supérieur en informatique**	**09**	**01**	**150 000 FCFA**	**300 000 FCFA**
	TOTAL				**800 000 FCFA**

Source : Mercurial des prix

2.5 LES LIVRABLES

Les livrables sont essentiels pour assurer la transparence, la gestion efficace et la réussite globale d'un projet, car ils fournissent des repères clairs pour l'équipe du projet et les parties prenantes afin de s'assurer que les objectifs sont atteints.

Ces livrables sont :

- ✓ Un rapport de stage complet portant sur le projet ;
- ✓ Une présentation du projet (PowerPoint) ;
- ✓ La solution implémentée.

CHAPITRE III : ETAT DE L'ART

RÉSUMÉ

L'état de l'art de ce projet présente une vue d'ensemble des concepts et des technologies essentiels pour l'automatisation des réseaux, avec un focus sur l'outil Ansible. Il décrit les principes de base de l'automatisation réseau, le fonctionnement d'Ansible et son architecture sans agent. Cette partie met en lumière les avantages d'Ansible par rapport à d'autres outils similaires, comme Puppet ou Chef, notamment en termes de simplicité d'utilisation et d'efficacité. Enfin, elle examine les études antérieures et les applications réelles d'Ansible dans des environnements réseau complexes.

SOMMAIRE :

INTRODUCTION

3.1 Présentation d'Ansible

3.2 Architecture d'Ansible

3.3 Applications de l'automatisation avec

3.4 Avantages et limites de l'automatisation des réseaux

CONCLUSION

Introduction

L'automatisation des processus réseau est devenue un pilier essentiel dans les systèmes informatiques modernes. Elle permet d'automatiser des tâches répétitives ou complexes, telles que la configuration des routeurs et des switches, la gestion des règles de sécurité, ou encore la supervision des performances. Cette automatisation réduit les erreurs humaines, améliore l'efficacité opérationnelle, et permet aux équipes IT de se concentrer sur des tâches à plus forte valeur ajoutée.

3.1 Présentation générale de l'automatisation dans le contexte réseau.

Historiquement, la gestion des réseaux était effectuée manuellement, souvent à travers des interfaces en ligne de commande (CLI). Cependant, avec la complexité croissante des environnements réseau — incluant des centaines voire des milliers de dispositifs interconnectés à travers différents sites — les équipes IT se sont retrouvées confrontées à des défis considérables.

Avec l'émergence du Software-Defined Networking (SDN) et des environnements virtualisés, l'automatisation est devenue indispensable. Le SDN permet de séparer le plan de contrôle du réseau (ce qui décide où les données sont envoyées) de son plan de données (ce qui achemine effectivement ces données). En conséquence, cela permet une gestion centralisée du réseau via des logiciels, ouvrant ainsi la porte à une automatisation plus large.

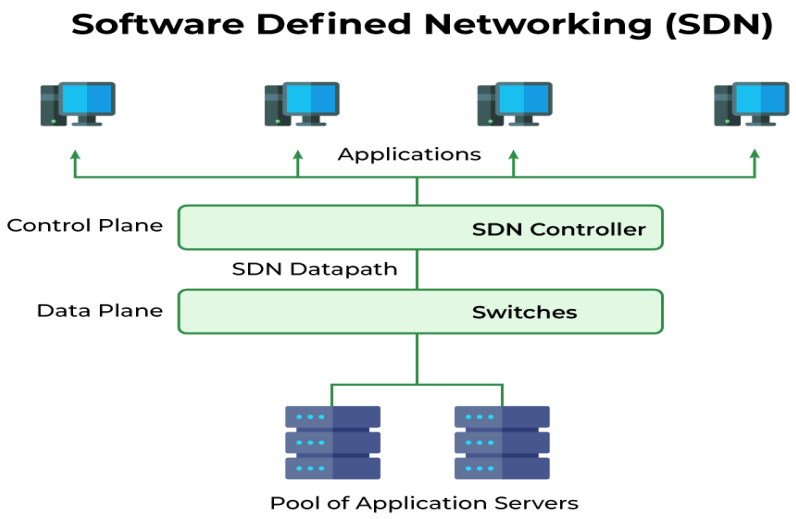


Figure 4 : Software-Defined Networking

L'outil Ansible, par exemple, a gagné en popularité car il offre une interface simple basée sur YAML pour gérer et configurer des réseaux de manière déclarative. Contrairement aux méthodes manuelles, l'automatisation permet de standardiser les configurations, évitant ainsi les erreurs dues à la variation humaine. De plus, les outils comme Ansible sont capables d'automatiser des tâches à grande échelle, comme le déploiement de mises à jour de sécurité ou la reconfiguration de plusieurs switches et routeurs simultanément.

3.1.1 Importance et rôle de l'automatisation dans la gestion des infrastructures réseau modernes.

Dans le contexte actuel, les entreprises font face à des défis sans précédent dans la gestion de leurs infrastructures réseau. Voici une analyse plus détaillée des principaux enjeux :

- **Complexité croissante des infrastructures** : Avec l'essor du cloud hybride et des architectures distribuées, les entreprises gèrent souvent des réseaux qui s'étendent sur plusieurs sites physiques et virtuels. Les configurations manuelles deviennent impraticables, car elles ne peuvent suivre le rythme des besoins évolutifs. Automatiser la configuration des équipements réseau permet aux entreprises de gérer cette complexité tout en garantissant la cohérence des configurations.

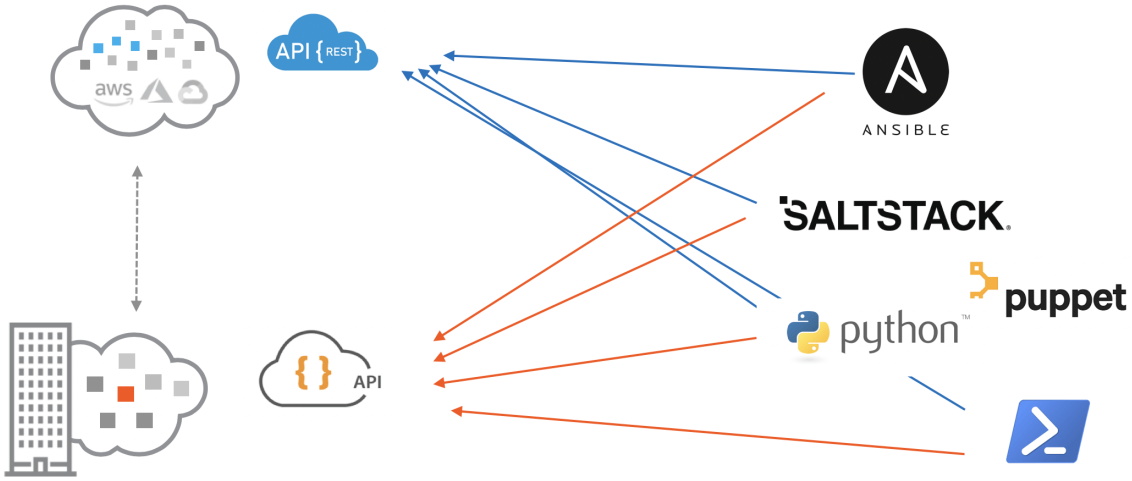


Figure 5 : Croissante des infrastructures

- **Scalabilité et agilité** : Les entreprises doivent être capables de faire évoluer rapidement leurs infrastructures en fonction des nouvelles exigences métiers. L'automatisation permet de répondre à ce besoin en rendant la gestion des réseaux plus flexible. Par exemple, la configuration de nouveaux équipements réseau peut être scriptée et exécutée à grande échelle, réduisant ainsi considérablement les temps de déploiement.

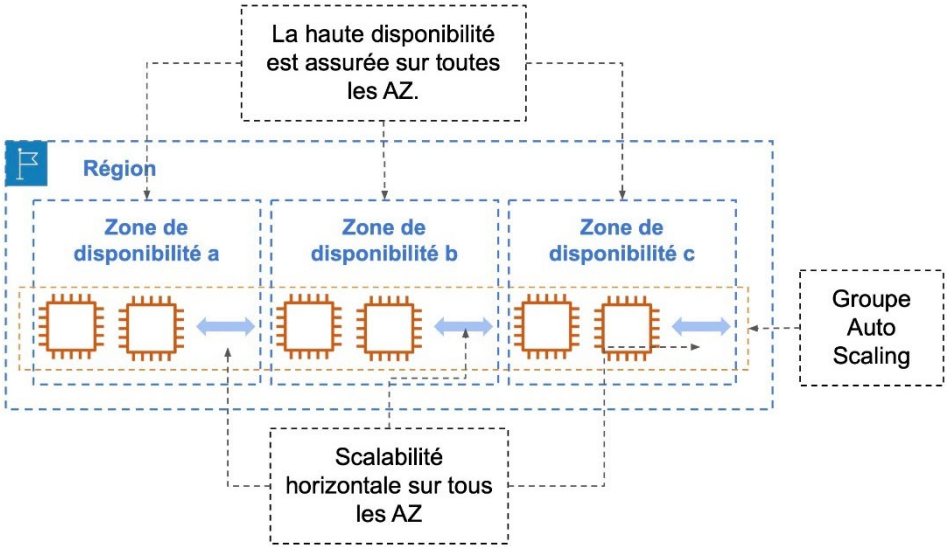


Figure 6 : Scalabilité

- **Sécurité** : La sécurité réseau est une priorité absolue pour toutes les entreprises. En automatisant les processus, notamment la gestion des pare-feux, des listes de contrôle d'accès (ACL) ou des mises à jour de sécurité, il est possible de garantir une conformité accrue avec les standards de sécurité. De plus, l'automatisation peut faciliter la détection des menaces et la réponse aux incidents de manière beaucoup plus rapide.

- **Efficacité opérationnelle** : L'automatisation permet d'éviter des erreurs humaines qui peuvent survenir lors de la configuration manuelle des équipements. En outre, elle améliore la réactivité des équipes IT face aux incidents en rendant les processus plus prédictibles. Par exemple, la configuration automatique des VLANs, des routes ou des politiques de sécurité permet aux équipes de se concentrer sur des tâches plus stratégiques, comme l'optimisation des performances.

- **Réduction des coûts** : Automatiser les processus réseau réduit considérablement le coût de la gestion opérationnelle. En évitant les interventions manuelles répétitives et en

réduisant les erreurs, les entreprises peuvent consacrer moins de ressources aux opérations réseau tout en augmentant leur efficacité.

3.2 Histoire et développement d'Ansible.

Ansible est né de la volonté de simplifier l'automatisation des systèmes informatiques. Son créateur, Michael DeHaan, a conçu l'outil en réponse à la complexité croissante des infrastructures IT et à la prolifération des solutions d'automatisation existantes, comme Puppet et Chef. Ces derniers nécessitaient souvent une architecture complexe, avec des agents sur les systèmes cibles et des configurations difficiles à maintenir. Ansible a été lancé en tant qu'alternative plus simple et légère, adoptant une architecture sans agent. En 2015, Red Hat a acquis Ansible, consolidant ainsi sa place dans le domaine de l'automatisation des infrastructures et des réseaux. Aujourd'hui, Ansible est l'un des outils les plus utilisés dans le DevOps et l'automatisation réseau, bénéficiant d'une communauté active et de nombreuses

Figure 7 : Représentation ansible

extensions.

3.2.1 Caractéristiques principales d'Ansible

Les caractéristiques uniques d'Ansible en font un outil puissant et polyvalent pour la gestion d'infrastructures IT et d'équipements réseau.

- **Infrastructure sans agent** : Contrairement à d'autres outils comme Puppet ou Chef, Ansible ne nécessite pas d'installer d'agent logiciel sur les systèmes cibles. Il utilise des protocoles standardisés tels que SSH (pour Linux) ou WinRM (pour Windows) pour communiquer directement avec les équipements. Cela réduit la complexité et minimise la gestion des agents sur des centaines ou des milliers de systèmes.

- **Langage déclaratif (YAML)** : Ansible utilise un langage déclaratif pour décrire l'état final souhaité d'un système ou d'un réseau, plutôt qu'un langage impératif qui dicterait chaque étape de la configuration. Cela signifie que l'utilisateur peut définir "ce qui doit être" sans se soucier des étapes techniques intermédiaires. Par exemple, un playbook peut spécifier qu'un package doit être installé ou qu'un service doit être démarré, sans décrire les étapes détaillées.

- **Idempotence** : Une des principales forces d'Ansible est son idempotence, c'est-à-dire la capacité à exécuter plusieurs fois le même playbook sans modifier un état déjà correct. Si une tâche est déjà accomplie (par exemple, un VLAN déjà configuré), Ansible ne la réapplique pas, ce qui évite les erreurs liées à des configurations redondantes.

- **Simplicité d'utilisation** : Ansible est conçu pour être utilisé par des équipes IT sans compétences de développement poussées. L'écriture des playbooks (fichiers YAML) est simple, lisible et intuitive. De plus, les modules fournis par la communauté permettent d'étendre rapidement les fonctionnalités d'Ansible pour gérer différents types d'équipements réseau, plateformes cloud et systèmes.

3.2.2 Comparaison avec d'autres outils d'automatisation (Puppet, Chef, SaltStack).

Figure 8 : Puppet, Chef, SaltStack

Ansible se distingue par sa simplicité et sa légèreté par rapport aux solutions comme Puppet, Chef ou SaltStack, qui adoptent des architectures plus complexes :

Puppet : Cet outil est basé sur une architecture client-serveur, où un serveur maître envoie les configurations aux agents installés sur chaque machine cible. Il nécessite une configuration initiale plus complexe, mais offre une gestion fine et puissante des ressources. Puppet est souvent préféré pour des environnements à très grande échelle avec des besoins de conformité rigides.

Chef : Chef, comme Puppet, repose sur une architecture maître-agent, mais il utilise un langage déclaratif basé sur Ruby, ce qui peut représenter une courbe d'apprentissage plus raide. Chef est puissant pour les environnements complexes mais moins accessible aux débutants comparés à Ansible.

SaltStack : SaltStack offre une flexibilité avec ses options d'agent ou de fonctionnement sans agent, mais il reste plus complexe à mettre en place. SaltStack est souvent choisi pour sa rapidité d'exécution dans de grands environnements distribués.

En comparaison, Ansible est beaucoup plus simple à configurer et ne nécessite pas d'infrastructure agent-maître. Cette approche "sans agent" réduit le temps de déploiement et facilite la gestion des petites à moyennes infrastructures, tout en restant compétitif pour des environnements plus complexes.

Tableau 4 tableau comparatif des thecnologie

Critères	Ansible	Puppet	Chef	SaltStack
Architecture	Sans agent	Maître-agent	Maître-agent	Maître-agent ou sans agent
Langage	YAML (déclaratif)	DSL spécifique basé sur Ruby	DSL basé sur Ruby	YAML (déclaratif) + Python (impératif)
Courbe d'apprentissage	Faible (facilité d'utilisation	Moyenne (langage spécifique Puppet)	Forte (langage Ruby nécessaire)	Moyenne à forte (selon les modules)
Modèle d'exécution	Basé sur SSH (sans agent)	Exécution via agents	Exécution via agents	Communication via ZMQ ou SSH
Simplicité de configuration	Très simple, fichiers YAML faciles à lire	Configuration plus complexe (manifests)	Complexité élevée (Cookbooks et Recipes)	Simple avec YAML, complexe avec Python
Idempotence	Oui	Oui	Oui	Oui
Communauté et support	Grande communauté, support Red Hat	Grande communauté	Communauté Ruby et DevOps	Grande communauté, surtout dans DevOps

Cas d'utilisation	Automatisation IT et réseau	Gestion d'infrastructure IT	Gestion de déploiements complexes IT	Déploiements rapides et scalables
Extensions/Modules	Nombreux modules et playbooks disponibles	Bibliothèque riche de modules	Systèmes de cookbooks extensibles	Modules disponibles, flexibilité Python
Gestion du Cloud	Très bon support des clouds publics	Support des clouds via modules	Support natif mais nécessite configuration	Support cloud avec modules
Performances	Moins performant sur de très grands environnements	Performances bonnes mais nécessite agents	Performances stables mais nécessite agents	Très performant pour des environnements distribués
Gestion des réseaux	Excellent pour l'automatisation réseau	Limité (principalement IT)	Limité (principalement IT)	Bon pour IT et réseaux
Prix	**Gratuit pour l'open-source**, versions payantes (Ansible Tower): environ **5 000 $/an**	**10 000 $/an** pour Puppet Enterprise (version open-source gratuite)	**7 200 $/an** pour Chef Automate (version open-source gratuite)	**6 000 $/an** pour SaltStack Enterprise, gratuit pour la version open-source

3.2.3 Analyse des prix :

- **Ansible** est gratuit dans sa version open-source, mais pour des fonctionnalités avancées (comme Ansible Tower), une licence coûte environ 5 000 $/an.
- **Puppet** offre également une version gratuite, mais la version Puppet Enterprise commence autour de 10 000 $/an, idéale pour des infrastructures complexes.
- **Chef** propose une version gratuite, mais Chef Automate coûte environ 7 200 $/an.

- ✓ **SaltStack** a une version gratuite, mais pour l'Enterprise, les prix commencent à 6 000 $/an.

Chaque outil a une version open-source gratuite, idéale pour les tests et les petits environnements, mais les licences payantes offrent plus de fonctionnalités pour les entreprises.

3.2.4 Avantages d'Ansible pour l'automatisation des équipements réseau.

L'utilisation d'Ansible pour l'automatisation réseau présente plusieurs avantages majeurs pour les entreprises cherchant à améliorer leur gestion d'infrastructure :

- ✓ **Standardisation des configurations** : Grâce à Ansible, les entreprises peuvent créer des playbooks qui décrivent une configuration réseau standardisée, évitant ainsi les variations manuelles susceptibles de provoquer des erreurs. Cela permet d'assurer une cohérence sur l'ensemble des équipements réseau (routeurs, commutateurs, pare-feu, etc.).

- ✓ **Réduction du temps de déploiement** : Automatiser des tâches réseau courantes telles que la configuration de VLANs, la mise en place de politiques de sécurité ou la gestion des ACL permet de réduire significativement le temps nécessaire pour déployer de nouvelles infrastructures ou mettre à jour les anciennes.

- ✓ **Évolutivité** : Dans un environnement de grande envergure avec des centaines de nœuds réseau, Ansible permet de déployer des configurations de manière rapide et fiable sur un grand nombre de dispositifs, facilitant ainsi les mises à jour en masse et la gestion de l'infrastructure.

- ✓ **Simplification des processus d'audit** : En automatisant les configurations réseau, Ansible permet aux équipes IT de garder une trace claire des changements effectués, facilitant ainsi les audits de sécurité et de conformité. Chaque playbook devient un document vivant des configurations appliquées, offrant une transparence totale.

- ✓ **Adaptation aux environnements hybrides** : Avec la montée des environnements cloud et hybrides, Ansible peut gérer à la fois les infrastructures traditionnelles et les

services cloud, en permettant une gestion unifiée des réseaux physiques et virtuels, des centres de données privés aux clouds publics comme AWS, Google Cloud, et Azure.

3.3 Architecture d'Ansible

L'architecture d'Ansible repose sur plusieurs composants essentiels, permettant une gestion et une orchestration efficaces des configurations des systèmes. Voici les éléments clés

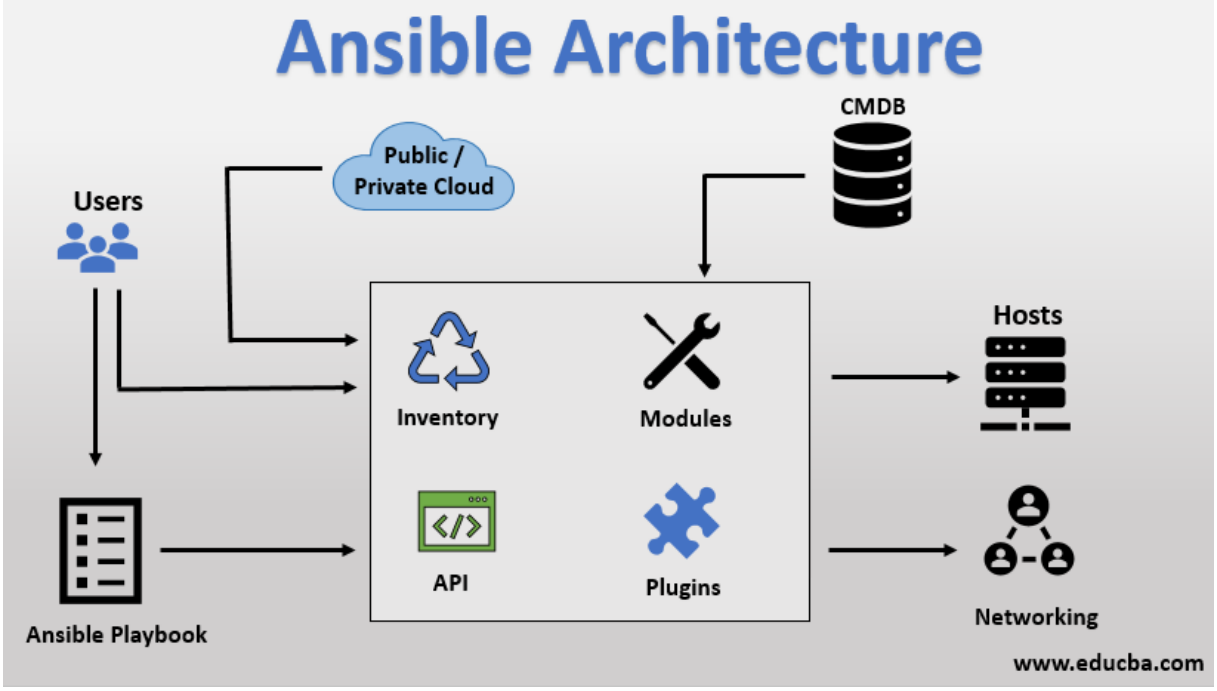


Figure 9 : Ansible architecture

- ✓ **Nœud de contrôle (Control Node) :** Le nœud de contrôle est le serveur ou la machine où Ansible est installé. C'est depuis ce point que les commandes Ansible sont exécutées et envoyées aux hôtes cibles. Le nœud de contrôle ne nécessite qu'une installation minimale d'Ansible, sans aucun agent supplémentaire sur les machines cibles. Cela rend Ansible extrêmement léger en comparaison avec d'autres solutions.
- ✓ **Playbooks :** Les playbooks sont les fichiers YAML qui définissent les actions à réaliser sur les hôtes cibles. Ils sont déclaratifs et permettent d'indiquer l'état souhaité des ressources plutôt que la manière exacte d'y parvenir. Un playbook contient plusieurs tâches (tasks), chacune définissant une action spécifique, comme l'installation d'un

logiciel ou la configuration d'un service réseau. Les tâches sont exécutées de manière séquentielle.

- ✓ **Inventaire :** Le fichier inventaire contient la liste des hôtes cibles organisés en groupes. Les hôtes peuvent être définis par leur adresse IP, leur nom DNS ou via un script générant une liste dynamique.

Exemple de fichier d'inventaire :

ini Copier le code

```ini
[webservers]
server1.example.com
192.168.1.10

[dbservers]
db1.example.com
```

Figure 10 : Rédaction du fichier hosts au format .ini

Chaque groupe d'hôtes peut être associé à des variables spécifiques, facilitant la personnalisation des configurations selon les environnements (développement, production).

Modules :

- ✓ **Les modules** Ansible sont les blocs fonctionnels qui réalisent les tâches sur les hôtes. Chaque module est conçu pour gérer une tâche spécifique : gestion des fichiers, services, configurations réseau, etc. Ansible possède une vaste bibliothèque de modules couvrant

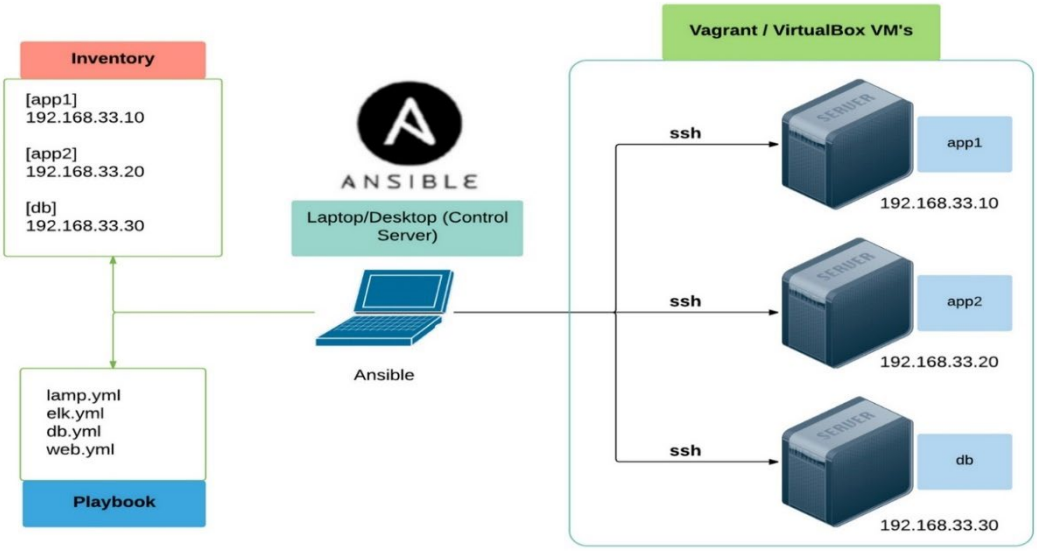


Figure 11 : Les modules ansible

des besoins variés (réseau, cloud, conteneurs, etc.). Par exemple, le module ios_config permet de gérer la configuration des équipements réseau Cisco.

3.3.1 Fonctionnement d'Ansible : utilisation de SSH pour la communication avec les hôtes.

Ansible se distingue par son fonctionnement sans agent (agentless), basé sur le protocole SSH (ou d'autres protocoles selon les besoins, comme WinRM pour Windows). Voici un aperçu détaillé du processus de fonctionnement :

- **Connexion SSH** : Le nœud de contrôle utilise SSH pour se connecter aux hôtes cibles. Cette méthode d'authentification est sécurisée et fiable, et Ansible supporte l'utilisation de clés SSH pour une meilleure gestion des accès.
- Exécution locale des modules : Une fois connecté à l'hôte via SSH, Ansible transfère les modules nécessaires et les exécute localement sur l'hôte cible. Cela signifie que les modules ne sont pas pré-installés sur les machines cibles, mais sont envoyés temporairement lors de l'exécution du playbook.
- Idempotence : Les tâches dans Ansible sont conçues pour être idempotentes. Cela signifie que la même tâche peut être exécutée plusieurs fois sans causer de modifications supplémentaires si l'état final souhaité est déjà atteint. Par exemple, si un service est déjà installé et actif, Ansible ne réinstallera ni ne redémarrera le service inutilement.

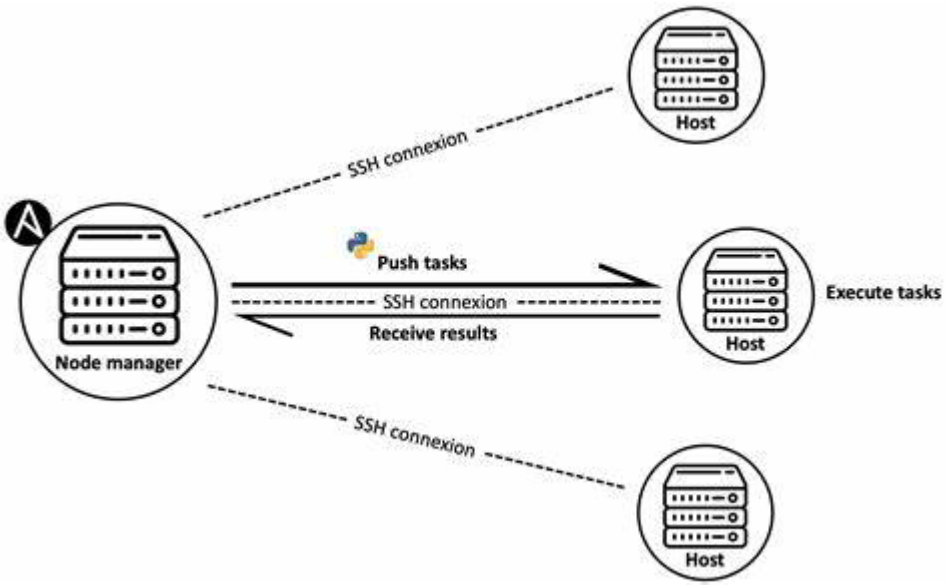


Figure 12 : Communication d'ansible en ssh

➢ **Exemple de flux de travail d'un playbook Ansible pour la configuration réseau.**

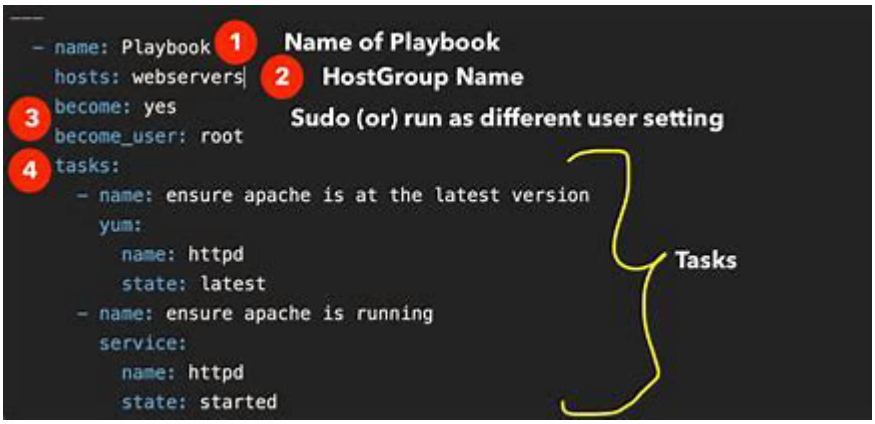


Figure 13 : Rédaction d'un playbook

3.4 Applications de l'automatisation avec Ansible dans les environnements réseau

L'automatisation réseau avec Ansible a de nombreuses applications concrètes dans la gestion des infrastructures modernes. Grâce à sa capacité à configurer, déployer et gérer des équipements réseau de manière centralisée, Ansible permet aux administrateurs réseau de simplifier et d'accélérer les opérations répétitives et complexes. Voici quelques-unes des principales applications d'Ansible dans les environnements réseau :

❖ **Configuration des équipements réseau**

Ansible permet d'automatiser la configuration initiale et continue des équipements réseau comme les routeurs, switches, et firewalls. Les configurations peuvent être déployées à grande échelle avec une réduction des erreurs humaines et une meilleure cohérence.

Exemple : La configuration de VLANs sur plusieurs switches via un playbook unique, en gérant automatiquement les interfaces et les paramètres de sécurité.

❖ **Mises à jour et correctifs**

L'automatisation avec Ansible facilite la gestion des mises à jour du firmware des équipements réseau et l'application de correctifs de sécurité. Ces opérations peuvent être centralisées, programmées et déployées sur des centaines d'appareils sans intervention manuelle répétitive.

Exemple : Mise à jour du firmware de plusieurs commutateurs Cisco en une seule opération grâce à un playbook Ansible.

- **Gestion des configurations réseau dynamiques**

Dans les environnements dynamiques, comme les centres de données ou les infrastructures cloud, Ansible peut gérer les changements fréquents dans les configurations réseau, tels que l'ajout de nouveaux sous-réseaux, la modification de règles de pare-feu ou l'ajustement des politiques de qualité de service (QoS).

Exemple : L'ajout automatique de nouvelles routes sur des routeurs lors de l'extension d'une infrastructure réseau.

- Gestion de la sécurité réseau

Ansible peut aider à configurer les politiques de sécurité réseau, telles que les listes de contrôle d'accès (ACL), les règles de pare-feu, et les tunnels VPN, tout en assurant la conformité aux normes de sécurité internes et externes.

Exemple : Application des règles de pare-feu et des ACL sur tous les routeurs et firewalls pour bloquer ou autoriser certains types de trafic.

- **Surveillance et gestion des incidents**

Grâce à l'automatisation, Ansible peut surveiller l'état des équipements réseau, identifier les pannes, et appliquer automatiquement des correctifs ou des redémarrages pour minimiser les interruptions de service.

Exemple : Détection automatique d'un switch en panne et rétablissement de la configuration sur un autre switch pour assurer la continuité du service.

- **Orchestration de réseau dans les environnements cloud et DevOps**

Dans les architectures cloud, Ansible s'intègre aux environnements DevOps pour orchestrer les déploiements réseau et les ajuster en fonction des besoins des applications. Cela permet de créer des environnements réseau adaptatifs, capables de répondre aux besoins de l'entreprise en temps réel.

Exemple : Orchestration de la configuration réseau dans un environnement multi-cloud pour connecter les services via des VPN et des VLANs.

Conclusion

L'automatisation des processus réseau avec Ansible transforme la gestion des infrastructures réseau, en améliorant la rapidité des déploiements, en minimisant les erreurs et en garantissant une conformité constante. Grâce à sa flexibilité et son intégration fluide avec des équipements multifournisseurs, Ansible est un outil incontournable pour les équipes réseau qui cherchent à moderniser et à simplifier leur gestion des configurations et des opérations réseau.

CHAPITRE IV : IMPLEMENTATION DES SOLUTIONS

RÉSUMÉ :

Dans la partie implémentation, l'objectif est de montrer les étapes concrètes et les outils utilisés pour déployer la solution automatisée avec Ansible.

SOMMAIRE :

INTRODUCTION

4.1 Prérequis D'installation

4.2 Mises En Œuvre Du Système

4.3 Automatisation des tâches

CONCLUSION

INTRODUCTION

Dans cette section, nous aborderons la mise en œuvre concrète de l'automatisation des configurations réseau à l'aide d'Ansible. Après avoir exploré les fondements théoriques et les avantages de l'automatisation, il est essentiel de détailler les différentes étapes pratiques du projet. L'objectif de cette partie est de présenter comment Ansible a été configuré et utilisé pour automatiser les tâches réseau, depuis la création de l'environnement jusqu'à la gestion des configurations sur des équipements spécifiques.

4.1 PREREQUIS D'INSTALLATION

4.1.1 ENVIRONNEMENT MATERIEL

Ici, nous allons décrire l'environnement matériel, logiciel et l'architecture pour la réalisation de la solution. Pour la réalisation de la solution, on a opté pour la création de 4 machines virtuelles sur un ordinateur portable ayant les caractéristiques suivantes : Ordinateur portable : Système d'exploitation :

- ✓ Windows 10 Professional
- ✓ Mémoire vive : 20 Go
- ✓ Disque dur : 500 Go
- ✓ Routeur

4.1.2 ENVIRONNEMENT LOGICIEL

- **Hyperviseur VMware Workstation Pro 17 (version 17.0.2)**

VMware est une plateforme de virtualisation qui permet de consolider des serveurs, d'assurer l'isolation et la sécurité des machines virtuelles, d'offrir une grande flexibilité pour la gestion des ressources, de garantir la haute disponibilité des applications, et de permettre la migration en temps réel des machines virtuelles. C'est un choix privilégié pour optimiser les infrastructures informatiques en entreprise

- **GNS3** :

Un simulateur de réseau qui permet de créer et de tester des topologies réseau virtuelles, sans avoir besoin d'équipements physiques. Il supporte une large gamme d'images de routeurs, commutateurs et pare-feux.

- Vous pouvez intégrer des routeurs et commutateurs Cisco, ou même des machines virtuelles via **QEMU**, **VirtualBox**, ou **VMware**.

- Prérequis pour utiliser Ansible

Avant de commencer la configuration d'Ansible, il est important de s'assurer que certaines conditions sont remplies :

Système de contrôle (Ansible Control Node) : Ansible doit être installé sur une machine (Linux de préférence, mais il peut aussi fonctionner sur macOS ou WSL pour Windows). Cette machine servira à envoyer des commandes vers les équipements réseau.

SSH : Tous les équipements réseau (routeurs, switches) doivent être accessibles via SSH, car Ansible utilise SSH pour communiquer avec eux.

Python : Python doit être installé sur la machine de contrôle. Ansible fonctionne à partir d'un environnement Python.

4.1.3 PRESENTATION DE LA TOPOLOGIE

Dans le cadre de ce projet de recherche sur l'automatisation des réseaux avec Ansible, nous avons mis en place une topologie réseau à l'aide de GNS3 pour valider nos configurations et tester les capacités d'Ansible en matière d'automatisation. Cette section décrit la topologie que nous avons simulée, ainsi que les différentes étapes d'implémentation que nous allons aborder au cours de ce rapport.

- Topologie Réseau Simulée

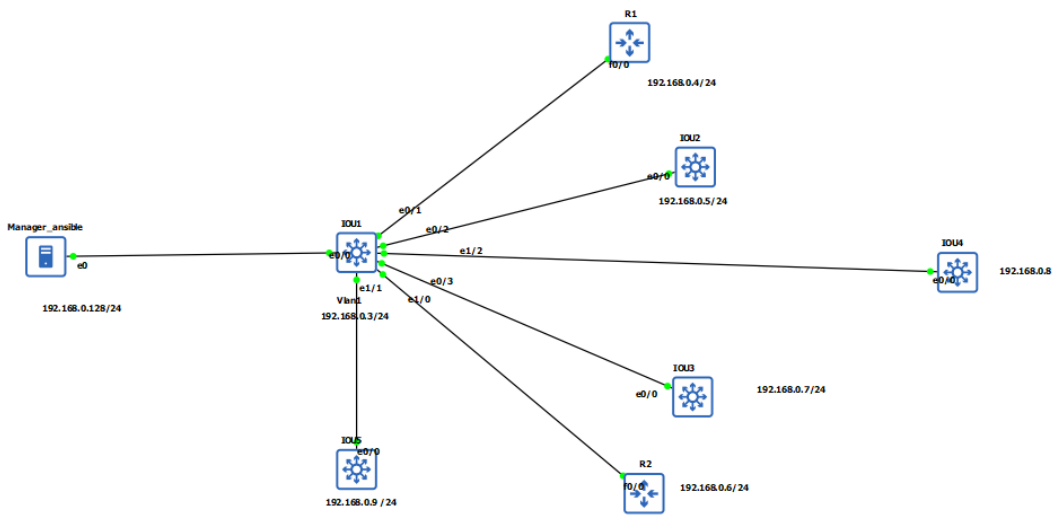


Figure 14 : Topologie automatisation

La topologie simulée comprend plusieurs équipements interconnectés :

- ✓ **Hôte Ubuntu (192.168.0.128/24) :** Il s'agit de la machine sur laquelle Ansible est installé. Cet hôte jouera le rôle de control node pour orchestrer la configuration des équipements du réseau.
- ✓ Commutateur central (IOU1) : Ce switch central est connecté à l'hôte Ubuntu ainsi qu'à plusieurs autres équipements du réseau, formant ainsi le cœur de la communication entre les équipements.
- ✓ Commutateurs secondaires (IOU2, IOU3, IOU4) : Ces commutateurs simulent des segments réseau distincts, permettant la mise en œuvre de VLANs et la séparation logique du trafic réseau.
- ✓ Routeurs (R1, R2) : Les routeurs sont utilisés pour gérer les sous-réseaux et assurer la communication inter-VLAN, offrant une simulation complète d'un environnement multi-réseaux.

4.2 MISES EN ŒUVRE DU SYSTEME

4.2.1 Préparation de l'environnement de travail

Pour mener à bien notre projet d'automatisation avec Ansible, il est essentiel de mettre en place un environnement de travail stable et adéquat. Étant donné que nous ne disposons pas

d'un environnement physique, nous utiliserons un environnement virtuel avec GNS3 pour simuler nos équipements réseau.

Nous avons opté pour une machine virtuelle Ubuntu, un système d'exploitation open-source robuste et bien adapté pour l'utilisation d'Ansible en tant que nœud de contrôle. Ubuntu permet de bénéficier d'une large gamme de ressources et de communautés pour résoudre d'éventuels problèmes. De plus, il est compatible avec Ansible, facilitant l'installation et la gestion des paquets.

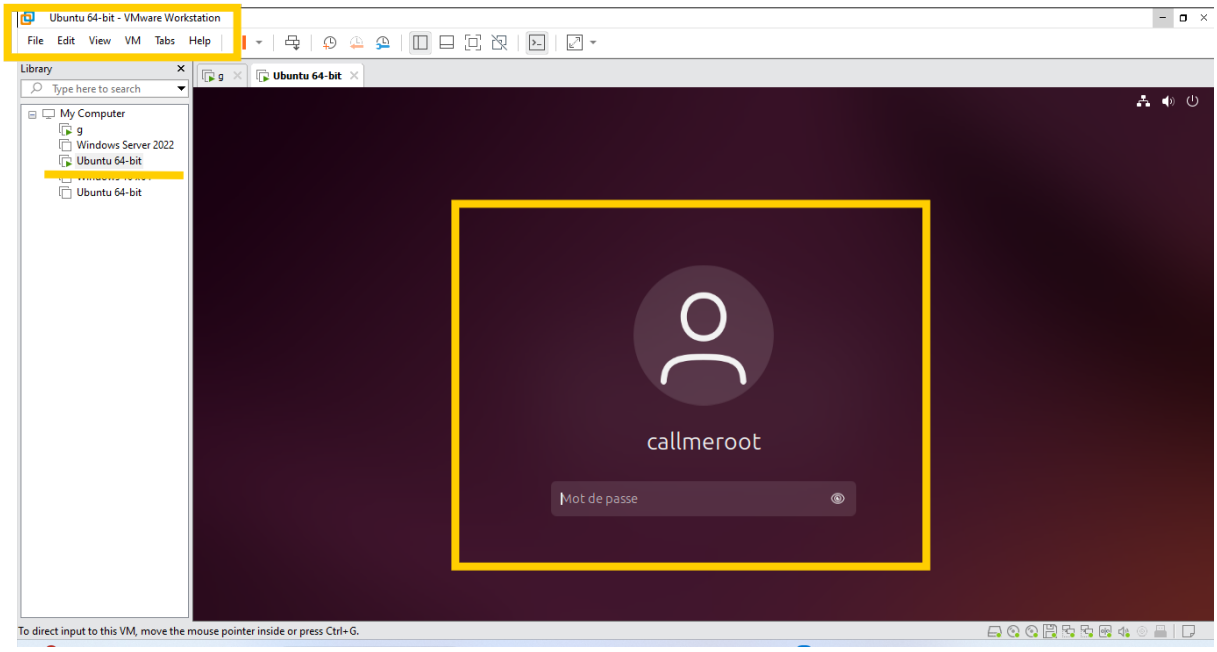


Figure 15 : Control manager

4.2.1.1.1 Configuration de l'environnement virtuel

Nous utiliserons VMware pour exécuter une machine virtuelle sous Ubuntu, et GNS3 pour la simulation réseau. Cela nous permettra de configurer, tester et appliquer les playbooks Ansible dans un environnement entièrement simulé.

Les étapes de configuration sont les suivantes : Installation de VMware Workstation pour la création de la machine virtuelle. Installation de GNS3 pour la simulation réseau.

Création d'une topologie réseau sous GNS3 incluant des routeurs et des switches simulés, tous connectés à notre machine Ansible.

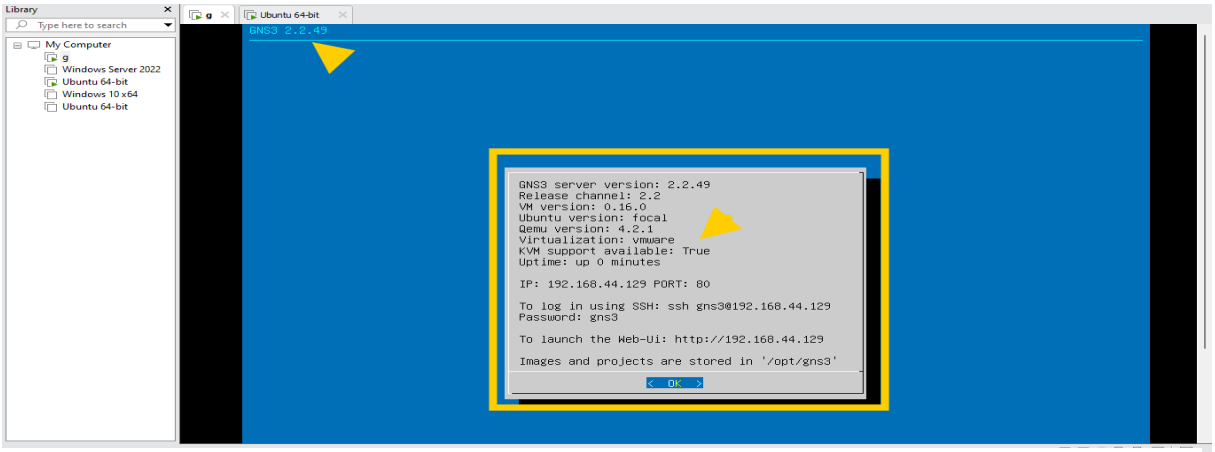


Figure 16 : GNS3 VM machine de liason

[2] **GNS3 VM** est une machine virtuelle qui optimise l'exécution de GNS3, offrant de meilleures performances et une gestion simplifiée des ressources. Elle permet d'héberger des appareils réseau virtuels, facilitant la création de topologies complexes et la simulation d'environnements de production. En intégrant **GNS3 VM avec Ansible**, nous pouvons tester des configurations et des scripts d'automatisation de manière efficace dans un environnement contrôlé.

4.2.1.1.2 Installation ansible sur la machine de contrôle

Comme mentionné précédemment, l'un des principaux avantages d'Ansible est la facilité de son déploiement. Pour cela, nous nous référerons à la documentation officielle afin de guider son installation.

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu

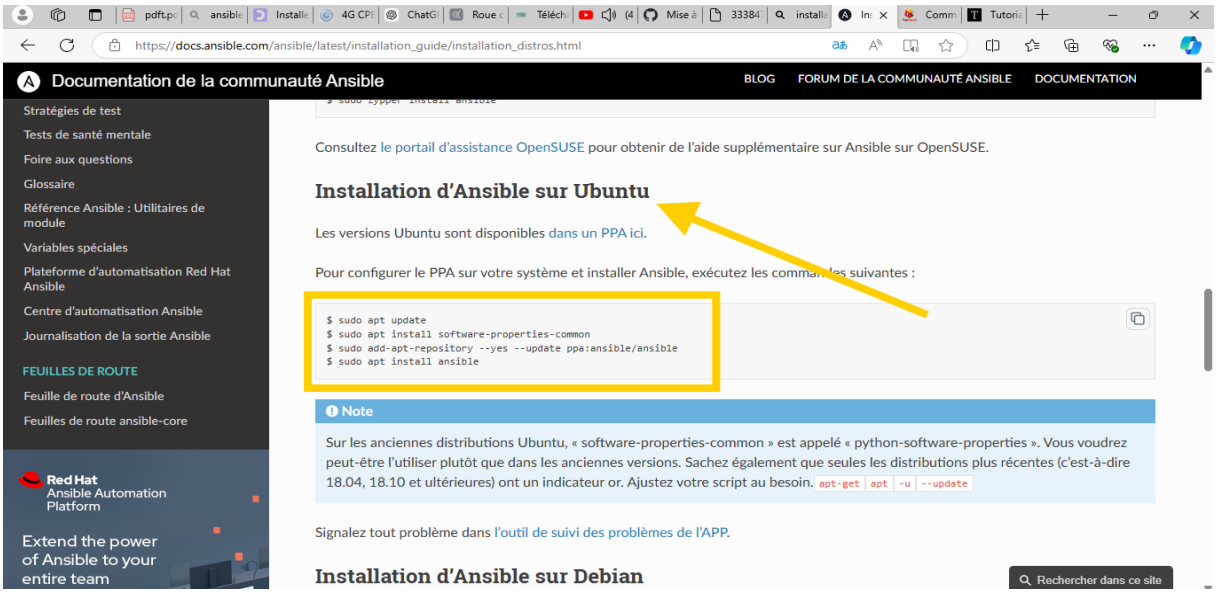


Figure 17 : Documentation Ansible

Nous avons les différentes lignes de commande installation sur Ubuntu nous allons installer

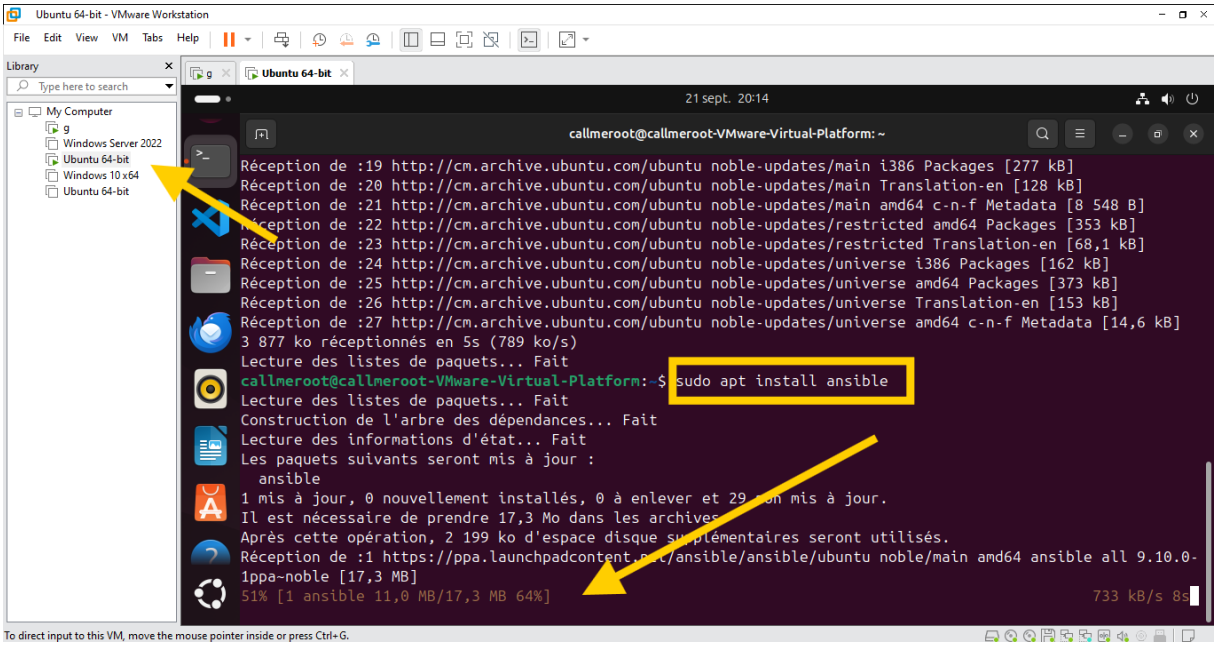


Figure 18 : Installation ansible sur le control manager

Nous pouvons belle et bien observée a travers la commande

➔ **Ansible - -version**

Que le paquet est bien installez

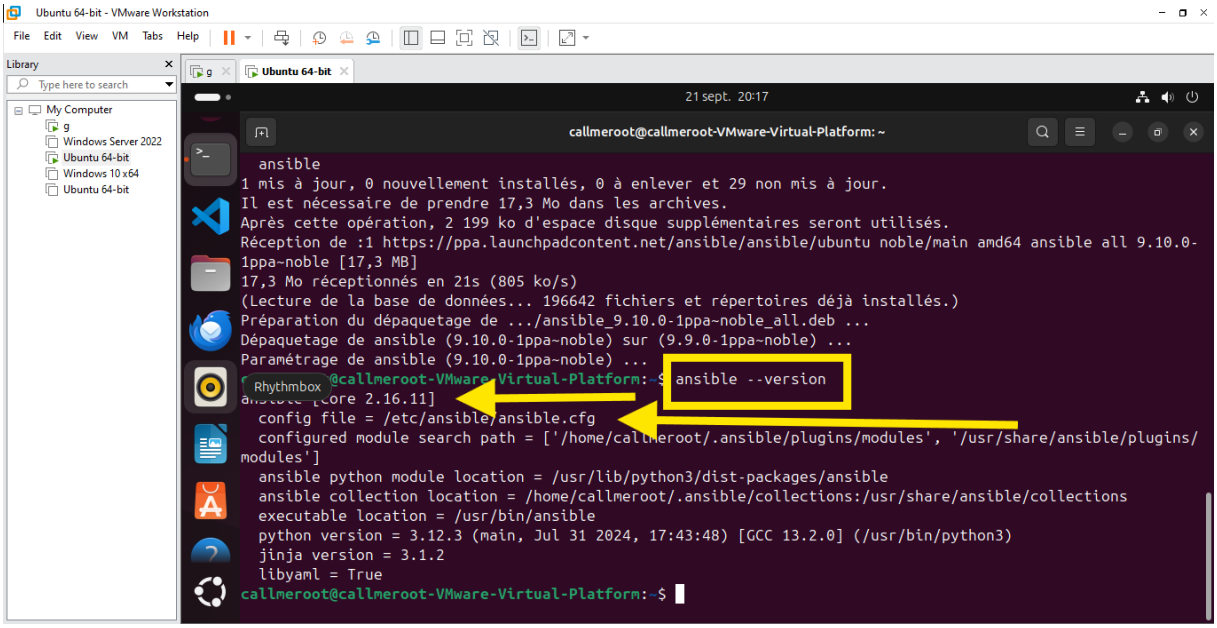


Figure 19 : Vérification de la version et du chemin accés

Le résultat indique que nous utilisons **Ansible version 2.16.11**. Les informations fournies incluent le chemin du fichier de configuration **(`/etc/ansible/ansible.cfg`)**, les répertoires de recherche des modules Ansible, ainsi que l'emplacement du module Python d'Ansible. Il est également précisé que **Python version 3.12.3** est utilisé, ainsi que la version de **Jinja (3.1.2),** ce qui est essentiel pour le traitement des modèles. La bibliothèque **YAML** est également activée, ce qui permet une meilleure gestion des fichiers de configuration.

4.2.1.1.3 Connection de notre machine de contrôle avec gns3

La connexion de notre machine de contrôle avec GNS3 est essentielle pour tirer pleinement parti de la simulation réseau. Cette intégration nous permet d'automatiser la configuration et la gestion des appareils virtuels, facilitant ainsi nos tests et déploiements. En établissant une connexion réseau entre la machine de contrôle, où Ansible est installé, et les appareils virtuels de GNS3, nous pouvons exécuter des scripts d'automatisation et gérer efficacement notre infrastructure réseau. Ce processus est crucial pour valider nos configurations dans un environnement contrôlé et simuler des scénarios réels.

- ➔ Ouvrir GNS3 : Lancez l'application GNS3 sur votre machine.
- ➔ Accéder aux Préférences : Cliquez sur "Édition" puis "Préférences".
- ➔ Activer GNS3 VM : Dans "GNS3 VM", cochez "Activer la GNS3 VM" et choisissez VMware comme hyperviseur.
- ➔ Tester la Connexion : cliquez sur "Appliquer" puis "OK".

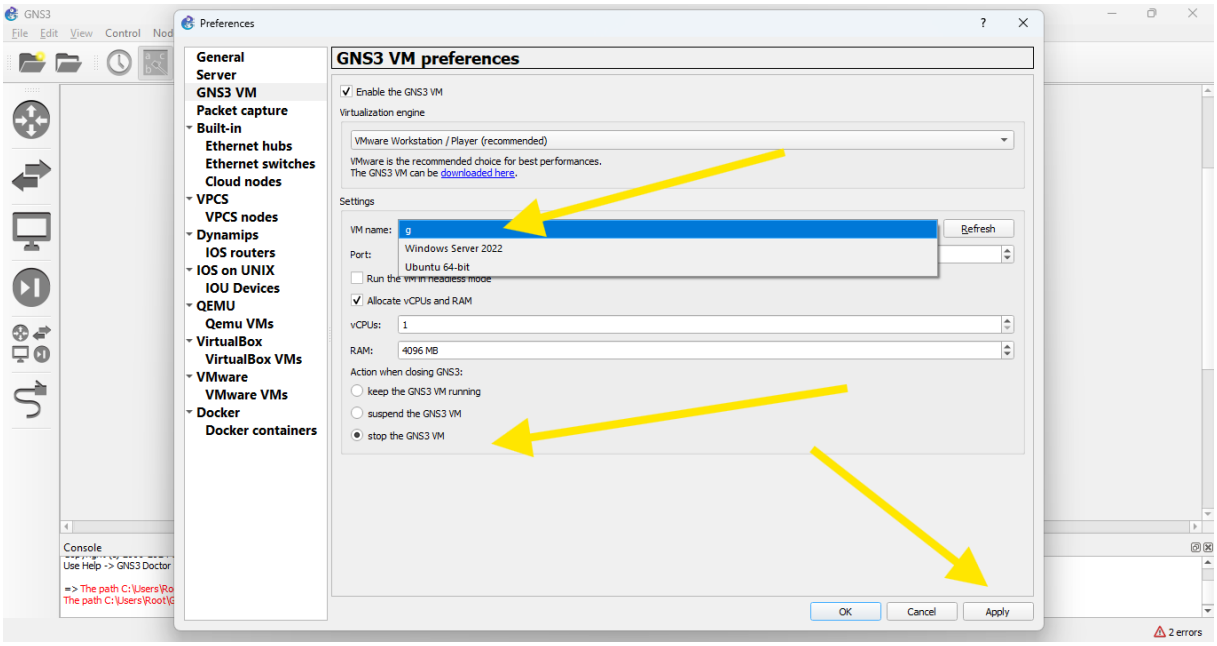


Figure 20 : Connection de notre machine de contrôle avec gns3

Apres notre machine GNS3 VM il nous faut ajoutez notre contrôleur (machine Ubuntu)

- **Accéder aux Préférences** : Cliquez sur le menu "Édition" (ou "Edit") et sélectionnez "Préférences" (ou "Preferences").
- **Configurer VMware** : Dans la fenêtre des préférences, allez dans la section "Dynamips" puis sélectionnez "VMware" dans le menu de gauche. Cliquez sur "Nouveau" (ou "New").
- **Sélectionner l'Image de la VM** : Choisissez la machine virtuelle que vous souhaitez ajouter à GNS3. Assurez-vous que la machine est déjà configurée et prête à être utilisée dans VMware.
- **Configurer les Paramètres** : Suivez les étapes pour configurer les paramètres de la machine virtuelle, comme le nom, le nombre de CPU, la mémoire et les interfaces réseau.
- **Valider les Paramètres** : Cliquez sur "Appliquer" (ou "Apply") puis sur "OK" pour enregistrer la configuration de votre machine virtuelle.

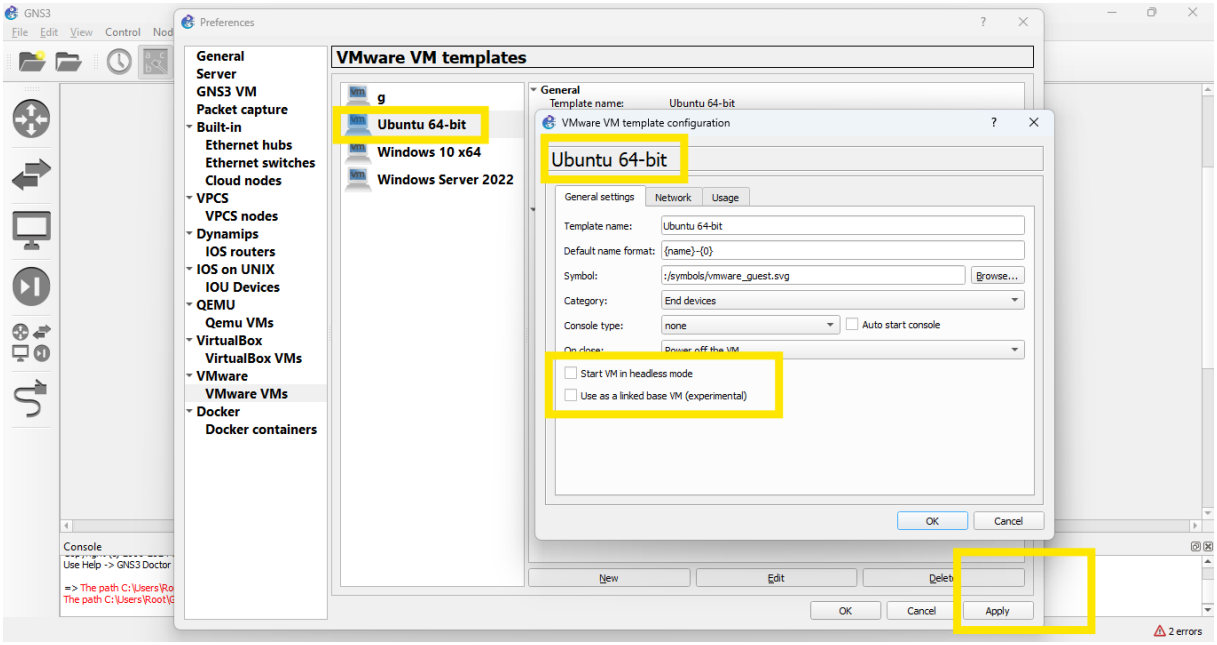


Figure 21 : Connection de notre machine de contrôle avec gns3

4.2.1.1.4 Mise en place de la topologie

Une fois la configuration de l'environnement terminée, nous pouvons relier nos différents appareils pour créer notre topologie.

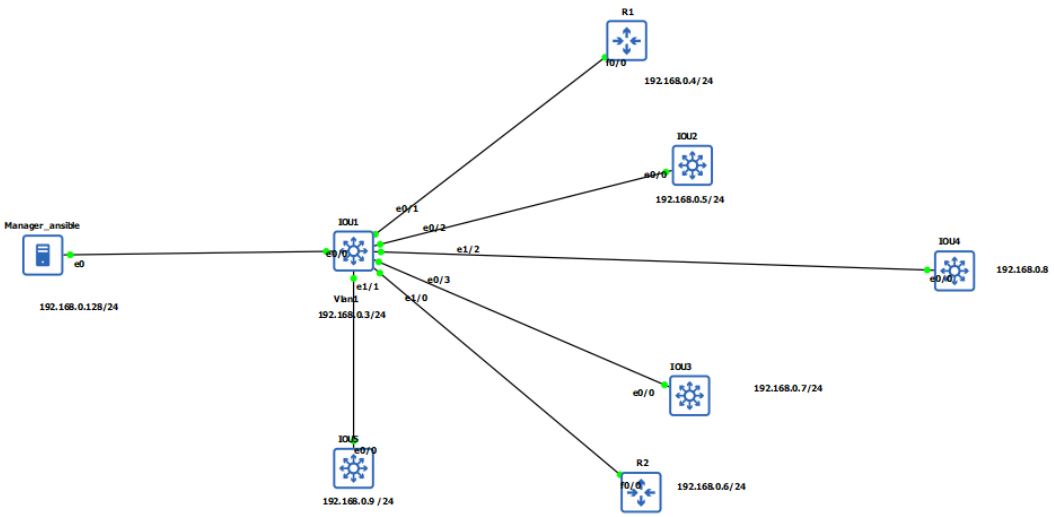


Figure 22 Réalisation de la topologie avec notre control manager et Gns3 VM

Une fois la configuration de l'environnement terminée, nous pouvons relier nos différents appareils pour créer notre topologie.

4.2.1.1.5 Configurations de accès SSH

SSH (Secure Shell) joue un rôle clé dans l'automatisation avec Ansible, car il permet de gérer les équipements distants de manière sécurisée et sans intervention physique. Son rôle principal est de fournir un canal de communication crypté entre la machine de contrôle Ansible et les hôtes (comme des routeurs, des commutateurs ou des serveurs).

Voici ses principales fonctions dans le contexte d'Ansible :

- **Connexion sécurisée** : SSH chiffre les données échangées entre la machine de contrôle et les équipements cibles, garantissant la confidentialité et l'intégrité des informations, notamment des mots de passe et des configurations sensibles.
- **Accès distant sans mot de** passe : Grâce à l'authentification par clé publique/privée, SSH permet à Ansible de se connecter aux hôtes sans avoir à entrer un mot de passe à chaque fois, rendant les opérations d'automatisation plus rapides et efficaces.

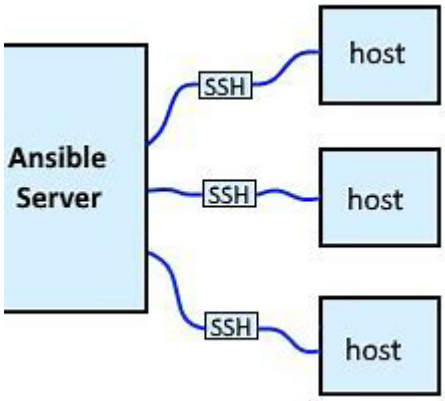


Figure 23 : Accès en SSH

- **Exécution des commandes à distance** : SSH permet à Ansible d'exécuter des tâches à distance, comme la configuration des équipements, l'envoi de fichiers ou la récupération de données, sans nécessiter d'accès direct ou physique aux machines.

Ainsi, SSH est le pilier qui permet à Ansible de déployer des configurations et de gérer les infrastructures réseau de manière automatisée et sécurisée.

- Nous allons configurer l'accès SSH sur tous les routeurs et switches.

```
IOU4(config)#inter
IOU4(config)#interface v
IOU4(config)#interface vl
IOU4(config)#interface vlan 1
IOU4(config-if)#ip addr
IOU4(config-if)#ip address 192.168.0.8 255.255.255.0
IOU4(config-if)#exit
IOU4(config)#ip d
IOU4(config)#ip do
IOU4(config)#ip domain na
IOU4(config)#ip domain name steve
IOU4(config)#crypti
IOU4(config)#crypto
IOU4(config)#crypto k
IOU4(config)#crypto key g
IOU4(config)#crypto key generate r
IOU4(config)#crypto key generate rsa
The name for the keys will be: IOU4.steve
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 4 seconds)

IOU4(config)#
*Sep 23 22:34:57.675: %SSH-5-ENABLED: SSH 1.99 has been enabled
IOU4(config)#ip ssh
IOU4(config)#ip ssh ver
IOU4(config)#ip ssh version 2
IOU4(config)#lin
IOU4(config)#line vty 0 4
IOU4(config-line)#tran
IOU4(config-line)#transport inp
```

Figure 25 : Configuration de accès Ssh sur les équipements

```
IOU4(config)#line vty 0 4
IOU4(config-line)#tran
IOU4(config-line)#transport inp
IOU4(config-line)#transport input ssh
IOU4(config-line)#login
IOU4(config-line)#login  local
IOU4(config-line)#login  local
IOU4(config-line)#userna
IOU4(config-line)#userna
IOU4(config-line)#usernam
IOU4(config-line)#username
                         ^
% Invalid input detected at '^' marker.

IOU4(config-line)#exit
IOU4(config)#user
IOU4(config)#username cisco pr
IOU4(config)#username cisco privilege 15 pas
IOU4(config)#username cisco privilege 15 password 0 vikingsf
IOU4(config)#service password
IOU4(config)#service password-encryption
IOU4(config)#
```

Figure 24 : Configuration de accès Ssh sur les équipements

```
IOU4(config)#ip domain nam steve
IOU4(config)#cri
IOU4(config)#cry
IOU4(config)#crypto  k
IOU4(config)#crypto  key g
IOU4(config)#crypto  key generate  rs
IOU4(config)#crypto  key generate  rsa
The name for the keys will be: IOU4.steve
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 9 seconds)

IOU4(config)#
*Sep 24 06:04:13.607: %SSH-5-ENABLED: SSH 1.99 has been enabled
IOU4(config)#ip shh
IOU4(config)#ip ss
IOU4(config)#ip ssh ver
IOU4(config)#ip ssh version 2
IOU4(config)#lin
IOU4(config)#line v
IOU4(config)#line vty 0 4
IOU4(config-line)#tra
IOU4(config-line)#transport inp
IOU4(config-line)#transport input ss
IOU4(config-line)#transport input ssh
IOU4(config-line)#log
IOU4(config-line)#login
IOU4(config-line)#login lo
IOU4(config-line)#login local
IOU4(config-line)#exit
IOU4(config)#user
IOU4(config)#username cis
IOU4(config)#username cisco
IOU4(config)#username cisco privil
IOU4(config)#username cisco privilege 15 pas
IOU4(config)#username cisco privilege 15 password viking
IOU4(config)#username cisco privilege 15 password 0 vikingsf
```

- Sur cette figure, nous voyons que le ping passe entre la machine de contrôle et le switch IOU1. De même, l'accès SSH est réussi avec le routeur R1.

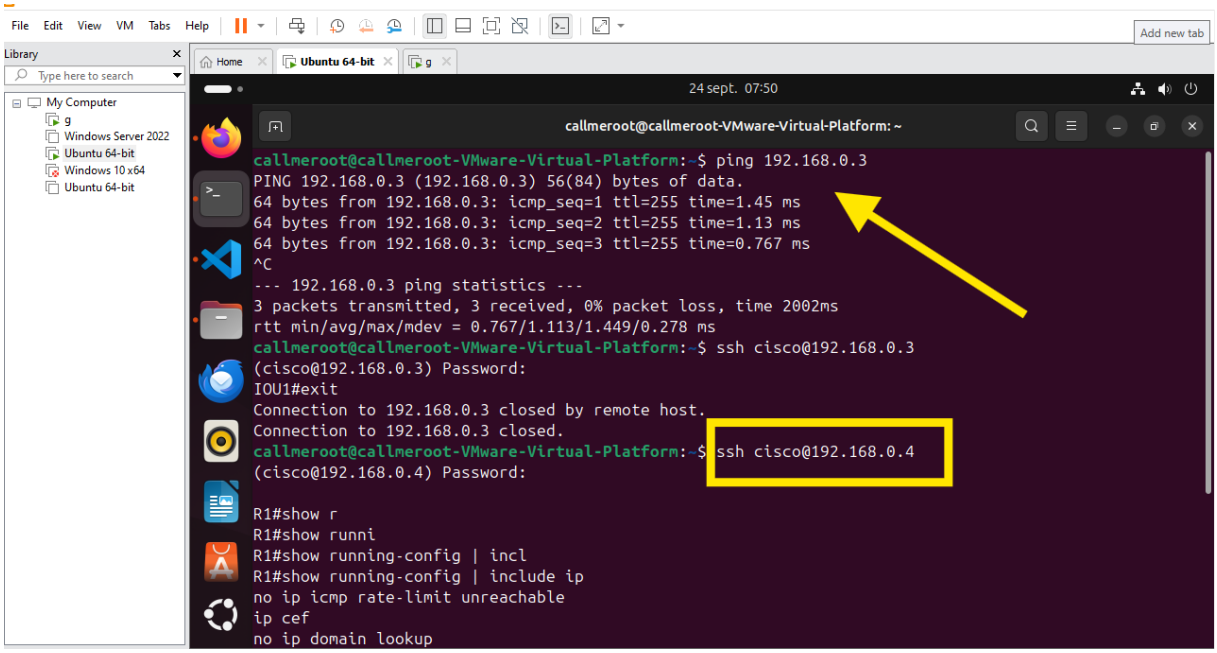


Figure 26 : Ping et accès en SSH depuis le node manager

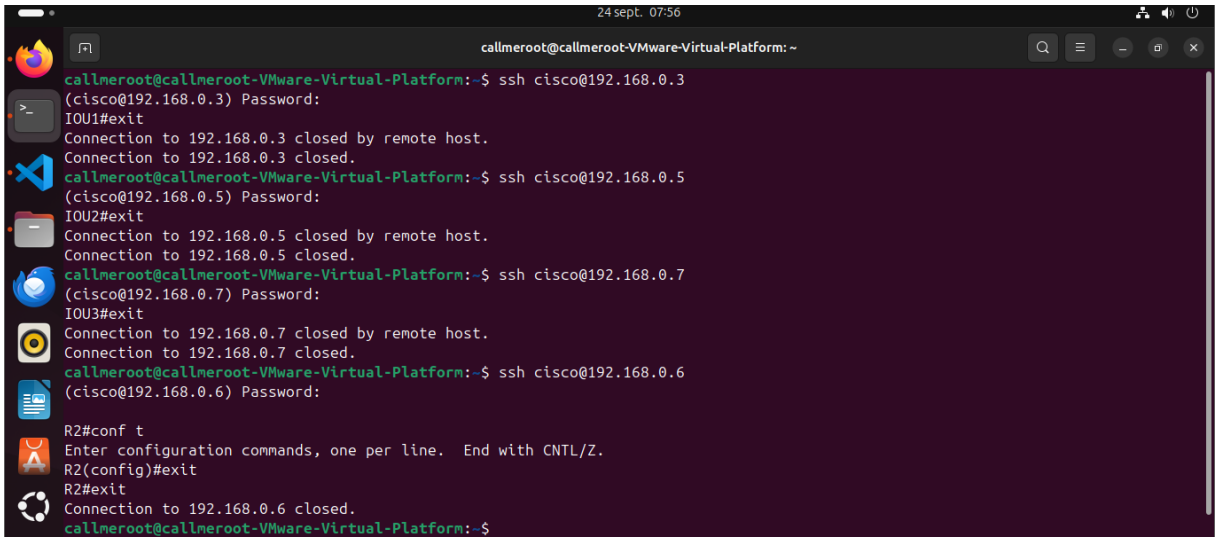
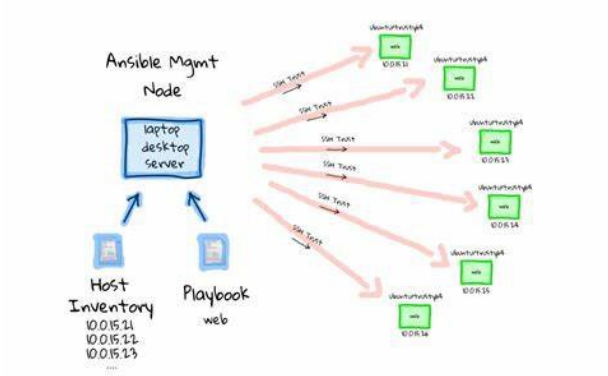


Figure 27 : Connexion à distance réussie

Nous pouvons remarquer que, sur la figure ci-dessus, la machine peut être prise en main à distance. On observe clairement que le SSH est configuré avec **le nom de domaine 'steve'**, la **version 2 du protocole SSH**, et **un utilisateur nommé 'cisco'** avec un mot de passe chiffré.

4.2.1.1.6 Configuration de l'inventaire Ansible

L'inventaire ou **inventory file** est le fichier dans lequel nous allons lister les équipements cibles **(hosts)**, ceux pour lesquels nous allons exécuter des tâches. Avec les playbook, c'est l'un des éléments les plus importants au fonctionnement d'Ansible. Nous allons pouvoir y décrire l'adresse ip d'administration, les identifiants de connexion ou bien encore le type d'équipement. Il peut se présenter sous différents formats et différents niveaux de complexité.



Le fichier d'inventaire peut être dans l'un des nombreux formats, en fonction des plug-ins d'inventaire dont vous disposez. Les formats les plus courants sont INI et YAML. Un INI de base /etc/ansible/hosts pourrait Ressembler à ceci :

```
mail.example.com

[webservers] ➔ correspond au nom de groupe …
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com

[west]
bar.example.com
three.example.com
```

Figure 28 : Syntaxe de rédaction d'un fichier hosts ou inventaire

Le fichier d'inventaire peut être dans l'un des nombreux formats, en fonction des plug-ins d'inventaire dont vous disposez. Les formats les plus courants sont INI et YAML. **Au format**

```
all:                ➔ Le group all imbrique tous les autres groupe. C'est le parent de tous les autres groupes
  children:
    dbservers:
      hosts:
        one.example.com: {}
        three.example.com: {}
        two.example.com: {}
    ungrouped:
      hosts:
        mail.example.com: {}
    webservers:
      hosts:
        bar.example.com: {}
        foo.example.com: {}
```

Figure 29 : Syntaxe de rédaction d'un fichier hosts ou inventaire

YAML.

- Emplacement du fichier d'inventaire

  Il existe deux options pour l'emplacement du fichier d'inventaire :

1. **Fichier global** : Par défaut, Ansible utilise le fichier d'inventaire situé à l'emplacement suivant : /etc/ansible/hosts. Ce fichier peut être utilisé pour gérer tous les hôtes de manière globale, mais il est souvent préférable d'utiliser un fichier dédié pour chaque projet.

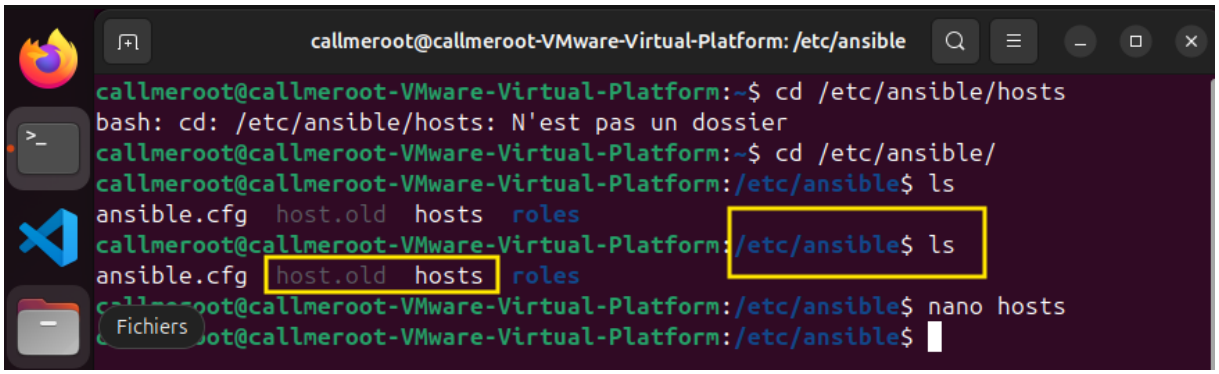


Figure 30 : Fichier hosts global

2. **Fichier spécifique au projet** : Pour une meilleure organisation, il est recommandé de créer un fichier d'inventaire dans un répertoire spécifique au projet. Par exemple, le fichier pourrait être placé dans le répertoire ~/ansible-project/inventory. Cela permet une gestion plus facile et isolée des hôtes d'un projet particulier.

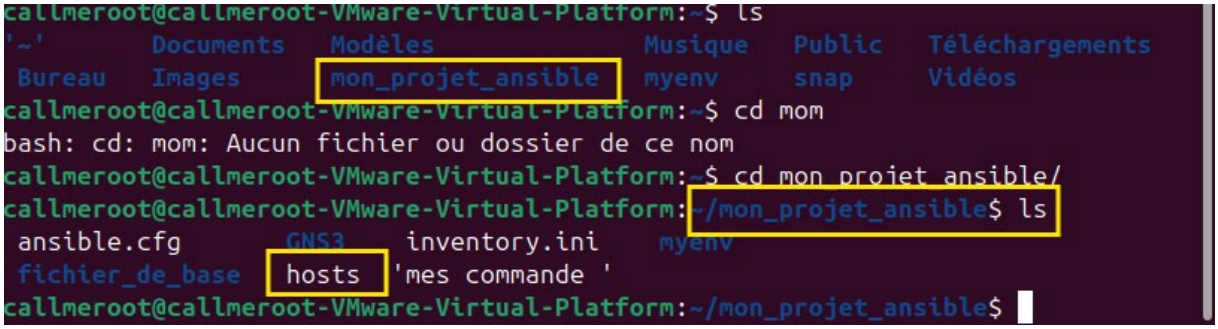


Figure 31 Fichier hosts spécifique au projet

Avant de créer notre **fichier d'inventaire (inventory ou hosts),** nous avons d'abord effectué une résolution de nom au niveau du fichier **racine /etc/hosts** de notre machine de

contrôle Ansible. Cette étape est cruciale pour faciliter la gestion des appareils dans le réseau sans devoir mémoriser ou utiliser leurs adresses IP à chaque fois.

Le fichier **/etc/hosts** permet d'associer des noms d'hôtes à des adresses IP locales. En modifiant ce fichier, nous avons pu attribuer un nom unique à chaque équipement réseau **(routeur, switch, etc.),** ce qui simplifie leur identification tout au long de nos configurations.

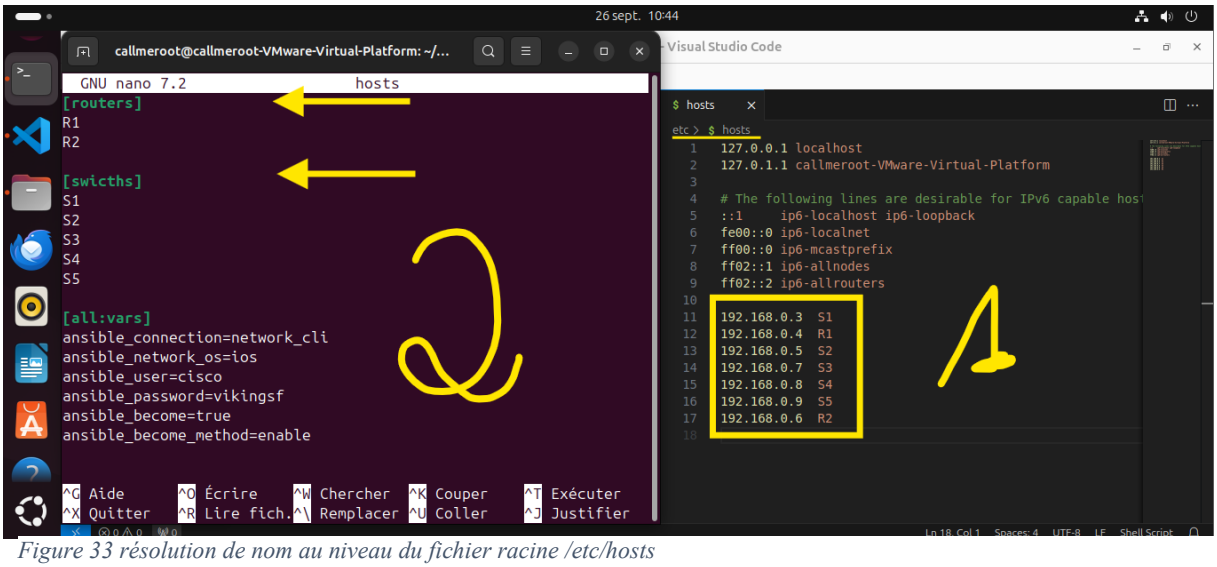


Figure 33 résolution de nom au niveau du fichier racine /etc/hosts

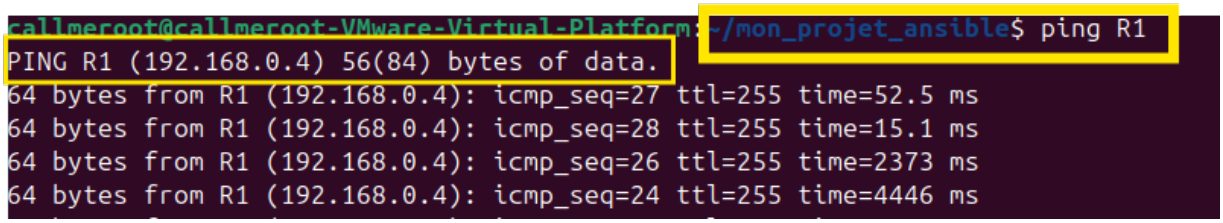


Figure 32 : PING R1(192.168.0.4)

Dans Ansible, il est possible de spécifier directement certaines variables dans le fichier d'inventaire. Ces variables définissent des propriétés essentielles pour la connexion aux hôtes et pour l'exécution des commandes. L'instruction vars permet de définir ces variables au niveau de chaque hôte, d'un groupe d'hôtes ou même d'un sous-groupe, ce qui simplifie grandement la gestion des connexions.

```
  GNU nano 7.2                                                hosts
[routers]
R1
R2

[swicths]
S1
S2
S3
S4
S5

[all:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_user=cisco
ansible_password=vikingsf
ansible_become=true
ansible_become_method=enable

                                                [ Lecture de 18 lignes ]
^G Aide        ^O Écrire      ^W Chercher    ^K Couper      ^T Exécuter    ^C
^X Quitter     ^R Lire fich.  ^\ Remplacer   ^U Coller      ^J Justifier   ^/
```

Figure 34 : Rédaction du fichier host avec paramètre pour la connexion

Prenons l'exemple de la configuration des identifiants de connexion pour un routeur Cisco. Voici comment nous pouvons définir ces variables dans le fichier d'inventaire :

Dans cet exemple :

- **ansible_host** : Spécifie l'adresse IP de l'équipement.
- **ansible_user** : Définit l'utilisateur SSH (ici, "cisco").
- **ansible_ssh_pass** : Le mot de passe utilisé pour se connecter à l'équipement.
- **ansible_network_os** : Indique le système d'exploitation réseau (ici, Cisco IOS).
- **ansible_connection** : Définit la méthode de connexion, ici network_cli pour les équipements réseau.

Pour que les équipements cibles soient correctement pris en charge par Ansible, il faut indiquer qu'il s'agit d'équipements réseau qui ne peuvent pas exécuter de scripts comme des serveurs standards. Voici les deux principales valeurs possibles pour ansible_connection :

network_cli : Utilise un plugin dédié à la gestion des équipements réseau via SSH. Ce plugin nécessite une installation préalable et permet à Ansible de gérer des équipements Cisco, Juniper, et d'autres constructeurs via leur interface en ligne de commande (CLI).

Après avoir configuré ton fichier d'inventaire, tu peux tester la connectivité avec les équipements (routeurs, switches) en utilisant Ansible pour effectuer un ping de type "**handshake**" (ce n'est pas un ping **ICMP** classique comme avec la commande ping sous Linux, mais un ping Ansible pour vérifier que la machine de contrôle peut bien se connecter aux hôtes via SSH).

```bash
ansible all -i <chemin_vers_fichier_inventaire> -m ping
```

Figure 35 : Commande ah-doc (ping) pour la vérification de la connexion ssh

- Test de connectivité

```
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible$ ansible all -m ping
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
S3 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
S1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
R1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
R2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
S2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
S5 | SUCCESS => {
```

Figure 36 : Test de connectivité

Dans cette capture d'écran, nous voyons une exécution de la commande **Ansible ansible all -m ping.** Cette commande a pour but de tester la connectivité entre la machine de contrôle Ansible et tous les hôtes définis dans le fichier d'inventaire. Voici ce qu'elle montre :

Résultats des pings :

Tous les hôtes ont répondu avec succès. Chaque résultat affiche un statut SUCCESS, confirmant que la communication est établie.

Le retour "ping": "pong" montre que chaque hôte a bien répondu au ping envoyé par Ansible. Cela signifie que l'hôte est en ligne et que la connexion SSH est possible.

Le champ "changed": false est toujours false dans ce type d'opérations, car il n'y a aucun changement de configuration effectué, seulement un test de connectivité.

Ainsi, cette image valide que la communication avec les différents équipements réseau (routeurs et switches) via SSH est fonctionnelle, ce qui est un prérequis pour les futures automatisations avec Ansible.

4.3 Automatisation des tâches

Après avoir validé la connectivité SSH entre notre machine de contrôle et les équipements réseau (routeurs et switches), nous pouvons désormais passer à l'étape clé du projet : l'automatisation des tâches de configuration. Grâce à Ansible, nous allons déployer de manière centralisée des configurations réseau sur tous les équipements, ce qui nous permettra de réduire considérablement le temps nécessaire à ces opérations tout en minimisant les risques d'erreurs humaines.

Pour atteindre cet objectif, nous allons utiliser Ansible à travers :

Une tâche (task) : Une tâche est une instruction décrite en YAML dans un fichier de configuration. Chaque tâche utilise un module ainsi que quelques éventuels arguments supplémentaires.

Structure d'une tâche :

```
- name: "Task 1 example"
  module_x:
    argument1: foo
    argument2: bar
--------------------- FIN DU PLAY 1 --------------------------
- name: "Task 2 example"
  module_y:
    argument1: foo
    argument2: bar
```

Figure 37 rédaction d'une tache

Un rôle : Un rôle est une structure arborescente constituée de répertoires et de fichiers de configuration YAML, qui vont avoir pour fonction d'installer tel ou tel système.

Les rôles peuvent être imbriqués et interdépendants les uns des autres. Un rôle est donc un ensemble de fichiers organisés dans une structure arborescente. Ansible Galaxy fait référence au site web de **https://galaxy.ansible.com/** où les utilisateurs peuvent partager des rôles (de la même façon que **GitHub** propose aux développeurs de stocker et de partager, publiquement ou non, le code qu'ils créent). **ansible-galaxy,** quant à lui, est un outil de ligne de commande pour télécharger, créer et gérer les rôles Ansible.

Pour créer un rôle avec ansible galaxy, il faut faire comme suit :

```
$ mkdir roles
$ cd roles


$ ansible-galaxy init apache

armel@ubuntu:~/role$ tree apache/
apache/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

files : tous les fichiers à copier sur le node ;

templates : tous les fichiers de template Jinja ;

tasks : liste des instructions à exécuter (le fichier main.yml est obligatoire) ;

handlers : même chose pour les instructions handlers (le fichier main.yml est obligatoire) ;

vars : fichier contenant des déclarations de variables (le fichier main.yml est obligatoire) ; les variables définies ici sont prioritaires par rapport aux variables définies dans l'inventaire ;

defaults : valeurs par défaut (le fichier main.yml est obligatoire) avec une priorité moindre ;

meta : dépendances du rôle et informations (auteur, licence, plateformes...) sur le rôle (le fichier main.yml est obligatoire).

*Figure 38 : Créations de notre **rôle***

4.3.1 Les playbooks

Un **Playbook** dans Ansible est un fichier écrit en YAML qui permet de définir un ensemble de tâches à exécuter de manière automatique et ordonnée sur un ou plusieurs hôtes (machines cibles). C'est un document central dans Ansible qui décrit **les étapes nécessaires pour atteindre un état de configuration souhaité** sur les équipements réseau ou les serveurs.

Structure d'un Playbook :

Un playbook est structuré autour de plusieurs éléments essentiels :

Le ou les hôtes ciblés (hosts) : Le playbook peut être appliqué à un ou plusieurs groupes d'hôtes définis dans l'inventaire.

Les tâches (tasks) : Chaque tâche est une action spécifique (par exemple : installer un paquet, configurer une interface, redémarrer un service, etc.).

Les modules : Ansible utilise des modules pour exécuter les actions. Chaque tâche utilise un module pour interagir avec les machines. Un module est un programme utilisé pour exécuter une tâche ou une commande Ansible. Chaque tâche utilise un module et un seul, qui peut prendre des arguments pour être exécuté de manière personnalisée. Ansible fournit de nombreux modules, mais vous pouvez créer vos propres modules. Tous les modules sont accessibles sur la documentation d'Ansible ou avec la commande **ansible-doc --list**.

https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections

Les variables (vars) : Ce sont des paramètres que l'on peut personnaliser pour chaque hôte ou groupe d'hôtes.

```
---
- name: Update web servers
  hosts: webservers
  remote_user: root

  tasks:
    - name: Ensure apache is at the latest version
      ansible.builtin.yum:
         name: httpd
         state: latest
    - name: Write the apache config file
      ansible.builtin.template:
         src: /srv/httpd.j2
         dest: /etc/httpd.conf
#---------------------------------------- FIN première instruction --------------------
- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
    - name: Ensure postgresql is at the latest version
      ansible.builtin.yum:
         name: postgresql
         state: latest
    - name: Ensure that postgresql is started
      ansible.builtin.service:
         name: postgresql
         state: started
```

Figure 39 : Syntaxe de notre playbook/ tache

Plus besoin de rappeler le rôle **ANSIBLE** dans les sections à venir, nous allons illustrer cette automatisation à travers plusieurs scénarios distincts. Chaque scénario sera basé sur des

besoins réels rencontrés en entreprise, comme la configuration d'accès SSH, la gestion des VLANs, la sécurisation des interfaces réseau, ou encore la sauvegarde automatique des configurations. Ces tâches seront déployées à l'aide de playbooks Ansible, qui contiennent des instructions détaillées pour automatiser la configuration des équipements.

Chaque scénario montrera :

- L'utilisation de variables pour personnaliser les configurations en fonction des appareils
- L'application de roles pour organiser et structurer les tâches ;
- L'exécution de playbooks qui mettent en œuvre ces configurations de manière automatique et contrôlée.

Scénario 1 : configuration de base des routeurs et switches

Dans ce premier scénario, nous allons aborder la configuration de base des équipements réseau, un passage crucial avant toute mise en service d'un réseau fonctionnel et sécurisé. En entreprise, chaque équipement doit être correctement configuré afin d'assurer un accès sécurisé, une gestion des connexions distantes (SSH), et la mise en place d'éléments de sécurité comme les ACLs (Access Control Lists).

Pour illustrer cette étape, nous allons utiliser un **playbook Ansible** qui va automatiquement configurer les éléments suivants sur l'ensemble des routeurs et switches de l'architecture réseau précédemment définie :

- ✓ **Nom des équipements** : Chaque appareil doit avoir un nom unique pour faciliter son identification au sein du réseau.
- ✓ **Nom de domaine** : Le nom de domaine est important pour la gestion des services DNS et des connexions SSH.
- ✓ **Configuration SSH** : L'accès distant sécurisé via SSH est configuré pour permettre aux administrateurs de gérer les équipements à distance de manière sécurisée.
- ✓ **Sécurisation des mots de passe** : Le cryptage des mots de passe garantit que les informations sensibles ne sont pas stockées en clair.
- ✓ **Mise en place de bannières** : Des messages de bannière (motd et login) sont configurés pour informer les utilisateurs lors de leurs connexions sur les équipements, rappelant qu'il s'agit de matériels protégés.

- ✓ **Création et sécurisation des utilisateurs** : Un utilisateur administrateur est créé avec des privilèges élevés et un mot de passe sécurisé.
- ✓ **ACLs** : Des listes de contrôle d'accès sont mises en place pour filtrer les connexions aux équipements, limitant l'accès aux seuls réseaux autorisés.

L'objectif de ce scénario est de garantir que tous les équipements ont une configuration initiale cohérente, sécurisée et prête à être utilisée. Cette étape automatisée via Ansible permet de déployer ces configurations sur plusieurs équipements en une seule exécution, évitant ainsi les erreurs humaines et accélérant le déploiement réseau.

Les configurations suivantes sont particulièrement importantes dans ce scénario car elles posent les bases nécessaires avant d'aller plus loin avec des configurations plus complexes dans les scénarios suivants.

Dans ce scénario nous allons utilisez les rôles commençons par crée nôtres rôles

- ➢ Création du rôle 1

```
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3/roles$ ansible-galaxy init conf_bases
- Role conf_bases was created successfully
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3/roles$ cd conf
conf-base/  conf_bases/
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3/roles$ cd confes
```

Figure 40 : Création du rôle

- ➢ Vérification du dossier rôle

```
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3/r...   callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet
conf-base/  conf_bases/
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3/roles$ tree conf_bases/
conf_bases/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Figure 41 : Vérification du dossier rôle

Nous pouvons observer les différents fichiers et dossiers qui ont été créés grâce à l'utilisation du module **ansible-galaxy init**, qui permet de générer une structure complète pour une

role Ansible. À présent, nous allons éditer le fichier **main.yml** situé dans le dossier **tasks**, car c'est dans ce fichier que seront définies les différentes tâches à exécuter pour notre scénario

Pour éditer les tâches, nous allons nous référer à la documentation officielle d'Ansible afin de répertorier les différents modules à utiliser. Cela nous permettra de sélectionner les modules les plus adaptés à chaque étape de notre configuration.

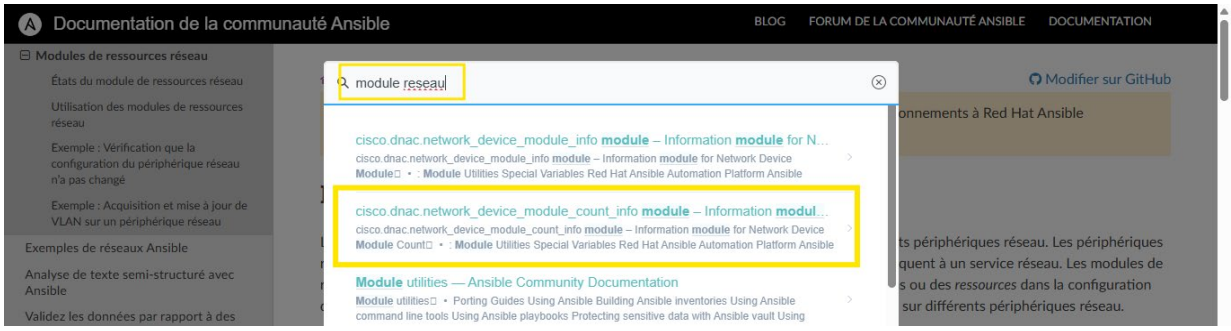


Figure 42 : Documentation d'ansible

- Nous pouvons observer plusieurs modules disponibles sur le site d'Ansible, tels que **`ios_banner`**, **`ios_acls`**, **`ios_interfaces`**, qui permettent de gérer différentes configurations sur les équipements réseau Cisco. Chaque module est conçu pour effectuer des tâches spécifiques, facilitant ainsi l'automatisation des configurations réseau.

Figure 43 : Documentation d'ansible

- Par exemple, dans le module **`ios_banner`**, nous pouvons observer qu'il permet de configurer différentes bannières sur les équipements réseau, comme la bannière de

connexion (`login`) ou la bannière d'avertissement (`motd`). Ce module offre une manière simple et standardisée d'ajouter des messages d'information ou de sécurité aux utilisateurs qui accèdent à nos équipements

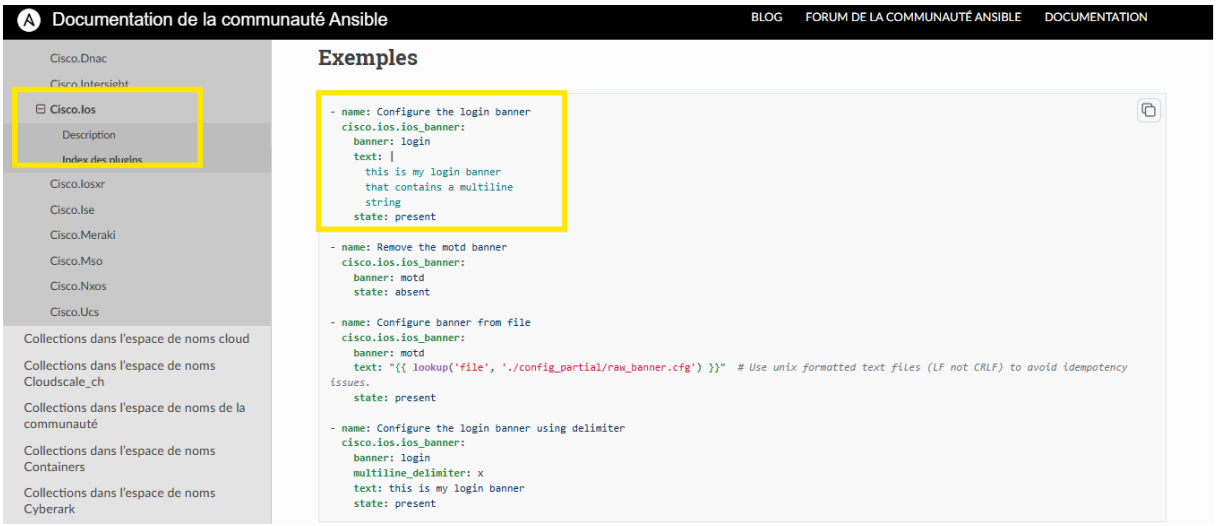


Figure 44 : odule ios_banner

- Nous avons également des exemples d'utilisation du module, qui montrent comment l'appliquer dans des configurations réelles. Par exemple, avec le module `ios_banner`, nous pouvons définir une bannière **motd** ou **login** en utilisant les paramètres suivants

Paramètre	Commentaires
bannière Chaîne / Obligatoire	Spécifie la bannière à configurer sur le périphérique distant. Dans Ansible 2.4 et versions antérieures, seuls *la connexion* et *motd* étaient pris en charge. Choix: • "login" • "motd" • "exec" • "incoming" • "slip-ppp"
multiline_delimiter corde	Spécifiez le caractère de délimitation qui sera utilisé pour la configuration. Faire défaut: "@"
état corde	Spécifie si la configuration est présente ou non dans la configuration active en cours d'exécution des appareils. Choix: • "present" ← (par défaut) • "absent"
SMS corde	Texte de bannière qui doit être présent dans la configuration d'exécution de l'appareil distant. Cet argument accepte une chaîne de plusieurs lignes, sans lignes vides. Nécessite *state=present*.

Figure 45 : Exemples d'utilisation du module

Grâce à cette documentation, nous allons éditer notre fichier de tâches et notre playbook pour exécuter les configurations souhaitées. En nous appuyant sur les exemples fournis, nous pourrons adapter les modules aux besoins spécifiques de notre infrastructure réseau et automatiser efficacement la gestion des équipements.

- fichier tache

```yaml
---

- name: limiter les connexions ssh
  ios_config:
    lines:
    - ip ssh logging events
    - ip ssh time-out 60
    - ip ssh authentication-retries 3

- name: Configurer les ACLs
  ios_config:
    lines:
    - permit tcp 192.168.0.0 0.0.0.255 any log
    parents: ip access-list extended VTY-ACL
    match: strict
    replace: block

- name: reapplication de VTY-ACL sur les ligne 0 4
  ios_config:
    lines:
    - access-class VTY-ACL in
    parents: line vty 0 4

- name: Sauvegarde de la configuration
  ios_config:
    save_when: changed
```

Figure 47 : Rédaction du fichier tache

```yaml
---

- name: Configurer les utilisateurs
  ios_config:
    lines:
    - username admin password admin

- name: mot de passe enbale
  ios_config:
    lines:
    - enable secret cisco


- name: crypter les passwords
  ios_config:
    lines:
    - service password-encryption

- name: creer une clef ssh
  ios_config:
    lines:
    - crypto key generate rsa general-keys modulus 2048

- name: configurer ssh version 2
  ios_config:
    lines:
    - ip ssh version 2

- name: limiter les connexions ssh
  ios_config:
    lines:
    - ip ssh logging events
```

Figure 46 :Rédaction du fichier tache

Résumé des étapes de configuration du fichier tâche :

- **Configuration du nom d'hôte** : Le playbook définit le nom d'hôte pour chaque appareil selon son nom d'inventaire.
- **Configuration du nom de domaine** : Le nom de domaine de l'appareil est configuré en tant que teachmemore.local.
- **Activation de la recherche DNS** : La recherche DNS est activée pour permettre à l'équipement de résoudre les noms de domaine.
- **Configuration de la bannière MOTD** : Un message de type "Message of the Day" (MOTD) est configuré pour avertir les utilisateurs non autorisés.
- **Configuration de la bannière de connexion** : Un message d'accueil s'affiche avant l'authentification de l'utilisateur.
- **Création d'un utilisateur admin** : Un utilisateur admin est ajouté avec des informations d'identification prédéfinies.
- **Configuration du mot de passe enable** : Un mot de passe est configuré pour accéder au mode privilégié de l'équipement.
- **Chiffrement des mots de passe** : Le chiffrement des mots de passe dans la configuration est activé pour plus de sécurité.
- **Génération d'une clé SSH** : Une clé RSA de 2048 bits est générée pour l'authentification SSH sécurisée.
- **Configuration de la version SSH** : La version 2 du protocole SSH est forcée pour toutes les connexions SSH.
- **Limitation des connexions SSH** : Les paramètres de connexion SSH sont restreints, avec un délai de 60 secondes et un maximum de 3 tentatives d'authentification.
- **Configuration des ACL (Access Control Lists)** : Des listes de contrôle d'accès (ACL) sont configurées pour filtrer le trafic réseau autorisé.
- **Ré application des ACL sur les lignes VTY** : Les ACL sont appliquées sur les lignes VTY pour contrôler l'accès via SSH.
- **Sauvegarde de la configuration** : La configuration est sauvegardée automatiquement si des modifications ont été apportées.

Fichier playbook qui nous permettra exécuter le fichier tache

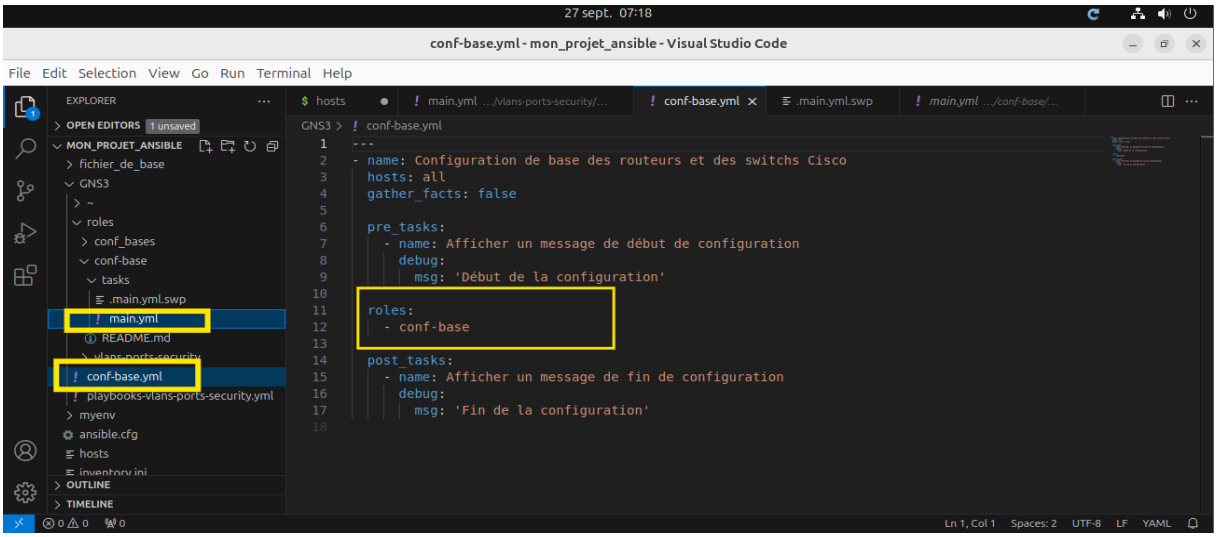


Figure 48 : Playbook exécution

Résultat du scenario

Nous allons exécuter notre fichier playbook ainsi que nos tâches depuis le terminal à l'aide de la commande suivante :

```bash
ansible-playbook -i inventaire.yml playbook.yml
```

Figure 49 : Commande exécution du ROLE conf-base

- **ansible-playbook** : Cette commande permet d'exécuter un playbook Ansible.
- **-i inventaire.yml** : Ici, on spécifie le fichier d'inventaire (`inventaire.yml`), qui contient la liste des hôtes sur lesquels les tâches vont être exécutées.
- **playbook.yml** : Il s'agit du fichier playbook contenant les différentes tâches de configuration à appliquer aux hôtes spécifiés dans l'inventaire.

Dans ce cas, cela lancera toutes les tâches décrites dans le playbook, depuis la configuration du nom des hôtes jusqu'à la sauvegarde de la configuration.

Ensuite, Ansible procédera à l'exécution de chaque tâche définie dans le playbook, en appliquant les configurations aux appareils réseau de la topologie, comme nous l'avons décrit dans les étapes précédentes.

Ceci conclura l'exécution de notre scénario de base.

```
Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
27 sept. 15:28
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ ansible-playbook conf-base.yml -u cisco -k
SSH password:

PLAY [Configuration de base des routeurs et des switchs Cisco] ***************************************************************

TASK [Afficher un message de début de configuration] *************************************************************************
ok: [S1] => {
    "msg": "Début de la configuration"
}
ok: [R1] => {
    "msg": "Début de la configuration"
}
ok: [R2] => {
    "msg": "Début de la configuration"
}
ok: [S2] => {
    "msg": "Début de la configuration"
}
ok: [S3] => {
    "msg": "Début de la configuration"
}
ok: [S4] => {
    "msg": "Début de la configuration"
```

Figure 50 : Exécution du rôle conf-base. YML

```
27 sept. 15:32
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3
TASK [conf-base : configure DNS lookup sources] ******************************************************************************
ok: [S1]
ok: [S2]
ok: [S3]
ok: [S4]
ok: [S5]
changed: [R1]
changed: [R2]

TASK [conf-base : Configure banner] ******************************************************************************************
ok: [S1]
ok: [S2]
ok: [S3]
ok: [S4]
ok: [S5]
changed: [R2]
changed: [R1]

TASK [conf-base : Configure the login banner] ********************************************************************************
ok: [S3]
ok: [S1]
ok: [S2]
ok: [S4]
ok: [S5]
```

Figure 51 : Exécution du rôle conf-base. YML

```
27 sept. 15:33
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3

TASK [conf-base : Configurer les utilisateurs] ***********************************************************************
[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: [S1]
changed: [S3]
changed: [S2]
changed: [S4]
changed: [S5]
changed: [R1]
changed: [R2]

TASK [conf-base : mot de passe enbale] *******************************************************************************
changed: [S1]
changed: [S2]
changed: [S3]
changed: [S4]
changed: [S5]
changed: [R2]
changed: [R1]

TASK [conf-base : crypter les passwords] *****************************************************************************
ok: [S1]
ok: [S2]
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

Figure 52 : Exécution du rôle conf-base. YML

```
27 sept. 15:34
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3
ok: [S2] => {
    "msg": "Fin de la configuration"
}
ok: [S3] => {
    "msg": "Fin de la configuration"
}
ok: [S4] => {
    "msg": "Fin de la configuration"
}
ok: [S5] => {
    "msg": "Fin de la configuration"
}

PLAY RECAP ***********************************************************************************************************
R1                         : ok=16   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
R2                         : ok=16   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S1                         : ok=16   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S2                         : ok=16   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S3                         : ok=16   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S4                         : ok=16   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S5                         : ok=16   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 53 : Exécution du rôle conf-base. YML

- ✓ **Début de la configuration**

Le playbook commence par afficher un message de confirmation sur chaque équipement pour indiquer que la configuration débute correctement. Cela montre que la connexion initiale avec les hôtes (routeurs et switchs) est réussie.

- ✓ **Configuration des noms d'hôtes**

La tâche configure les noms des équipements sur chacun des routeurs et switchs. Tous les hôtes ont été configurés sans problème, ce qui est confirmé par le statut ok.

- ✓ **Configuration du nom de domaine**

Ici, la configuration du nom de domaine sur les équipements est réalisée. Les équipements R1 et R2 ont montré un changement (changed), car ils n'avaient pas cette configuration avant

l'exécution du playbook. Les autres switchs avaient déjà cette configuration en place, donc aucun changement n'a été nécessaire.

- ✓ **Configuration des sources de recherche DNS**

Les serveurs DNS sont configurés sur tous les équipements. Les routeurs R1 et R2 ont été modifiés, tandis que les switchs n'ont pas montré de changement.

- ✓ **Configuration des bannières de connexion**

Les bannières de connexion pour les utilisateurs qui accèdent aux équipements via SSH ou console sont configurées avec succès. Encore une fois, les routeurs R1 et R2 ont subi des changements, tandis que les switchs ont maintenu leur configuration.

- ✓ **Configuration des utilisateurs**

Tous les équipements ont vu leurs utilisateurs configurés, avec des changements sur chaque équipement.

- ✓ **Configuration des mots de passe enable et cryptage des mots de passe**

Cette section montre que les mots de passe enable sont définis et les mots de passe existants ont été cryptés sur tous les équipements.

- ✓ **Création de clés SSH et configuration SSH**

Les clés SSH sont créées pour chaque équipement, permettant un accès sécurisé via SSH. La version SSH est aussi configurée en version 2 sur tous les équipements.

- ✓ **Limitation des connexions SSH**

Des ACLs (Listes de contrôle d'accès) sont appliquées pour restreindre les connexions SSH sur les lignes VTY (virtual teletype), garantissant ainsi que seules les adresses IP autorisées peuvent se connecter.

- ✓ **Application des ACLs et sauvegarde**

Les ACLs sont appliquées aux équipements R1 et R2 pour sécuriser l'accès via SSH. Ensuite, la configuration de tous les équipements est sauvegardée.

- **Fin de la configuration**

Chaque équipement affiche un message indiquant la fin de la configuration.

Résumé du Playbook (PLAY RECAP)

Tous les équipements (R1, R2, S1, S2, S3, S4, S5) ont été configurés avec succès. Les routeurs ont subi plus de changements (changed=10), car ils nécessitaient davantage de

modifications, tandis que les switchs avaient déjà plusieurs configurations en place (changed=4)

Scénario 2 : Vérification de la Configuration des Équipements Réseau via Ansible

Après avoir déployé la configuration de base sur les routeurs et switchs Cisco à l'aide d'Ansible, il est crucial de vérifier que ces changements ont bien été appliqués. Ce second scénario se concentre sur l'interrogation des équipements réseau pour s'assurer que la configuration est conforme aux attentes.

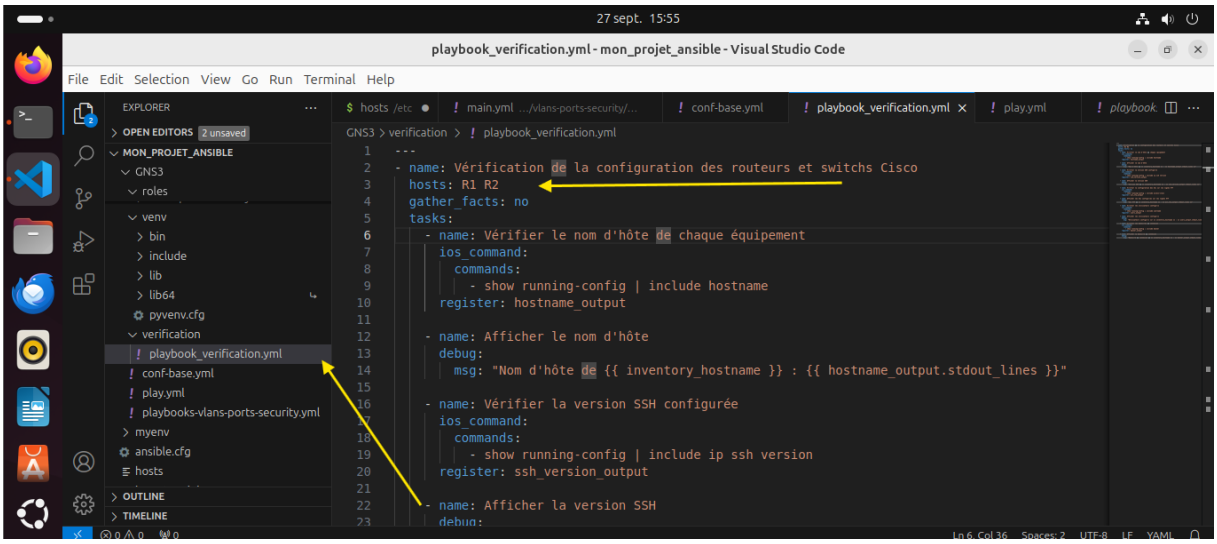
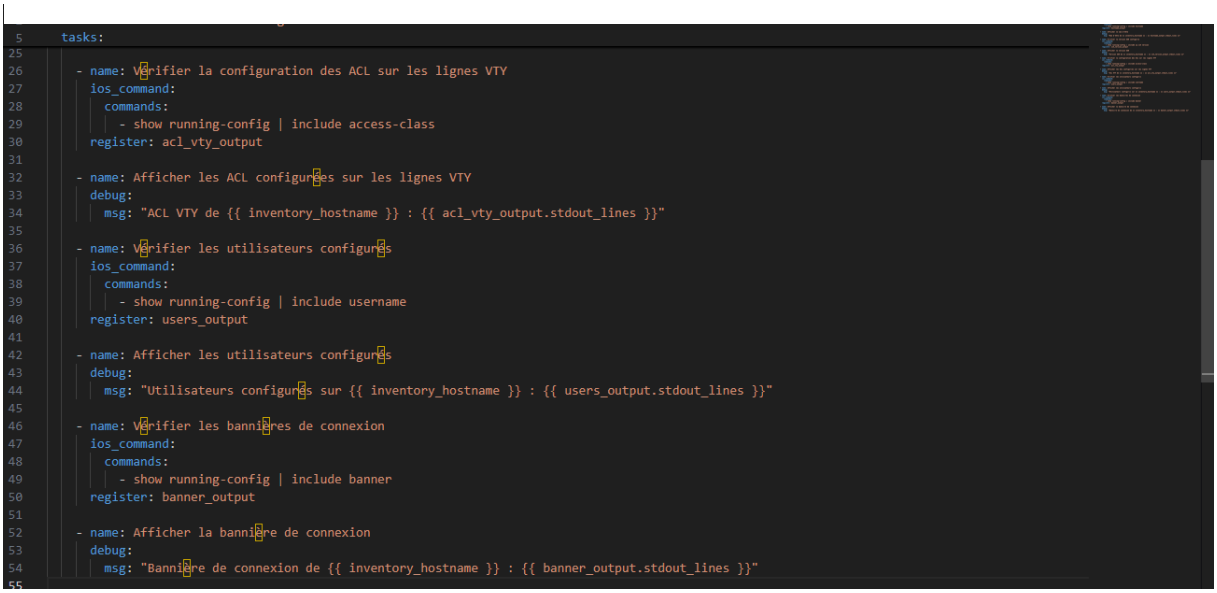


Figure 54 : Vérification de la Configuration des Équipements Réseau via Ansible



Grâce à ce playbook, nous allons vérifier plusieurs éléments essentiels tels que le nom d'hôte, la version SSH, les ACL sur les lignes VTY, les utilisateurs configurés, et les bannières de connexion. Ce processus permet de garantir la sécurité et l'intégrité de la configuration

```
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3/verification $ ansible-playbook playbook_verification.ym
l

PLAY [Vérification de la configuration des routeurs et switchs Cisco] ************************************************************

TASK [Vérifier le nom d'hôte de chaque équipement] ***********************************************************************************
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
ok: [R2]
ok: [R1]

TASK [Afficher le nom d'hôte] ********************************************************************************************************
ok: [R1] => {
    "msg": "Nom d'hôte de R1 : [['hostname R1']]"
}
ok: [R2] => {
    "msg": "Nom d'hôte de R2 : [['hostname R2']]"
}

TASK [Vérifier la version SSH configurée] ********************************************************************************************
ok: [R1]
ok: [R2]

TASK [Afficher la version SSH] *******************************************************************************************************
```

Figure 55 : Résultat de la vérification

```
TASK [Afficher la version SSH] *******************************************************************************************************
ok: [R2] => {
    "msg": "Version SSH de R2 : [['ip ssh version 2']]"
}
ok: [R1] => {
    "msg": "Version SSH de R1 : [['ip ssh version 2']]"
}

TASK [Vérifier la configuration des ACL sur les lignes VTY] **************************************************************************
ok: [R1]
ok: [R2]

TASK [Afficher les ACL configurées sur les lignes VTY] *******************************************************************************
ok: [R1] => {
    "msg": "ACL VTY de R1 : [['access-class VTY-ACL in']]"
}
ok: [R2] => {
    "msg": "ACL VTY de R2 : [['access-class VTY-ACL in']]"
}

TASK [Vérifier les utilisateurs configurés] ******************************************************************************************
ok: [R1]
ok: [R2]
```

Figure 56 : Résultat de la vérification

```
ok: [R1]
ok: [R2]

TASK [Afficher les utilisateurs configurés] ******************************************************************************************
ok: [R1] => {
    "msg": "Utilisateurs configurés sur R1 : [['username cisco privilege 15 password 7 111F100E1E1C0C1F02', 'username admin pass
word 7 11081D081E1C']]"
}
ok: [R2] => {
    "msg": "Utilisateurs configurés sur R2 : [['username cisco privilege 15 password 7 0719284747071E1611', 'username admin pass
word 7 11081D081E1C']]"
}

TASK [Vérifier les bannières de connexion] *******************************************************************************************
ok: [R2]
ok: [R1]

TASK [Afficher la bannière de connexion] *********************************************************************************************
ok: [R1] => {
    "msg": "Bannière de connexion de R1 : [['banner login ^C', 'banner motd ^C']]"
}
```

Figure 57 : Résultat de la vérification

Explication de l'Exécution du Playbook de Vérification

L'exécution du playbook playbook_verification.yml a permis de vérifier que la configuration appliquée aux routeurs R1 et R2 correspond bien aux attentes. Les différentes tâches réalisées sont les suivantes :

Vérification du nom d'hôte : Le nom d'hôte de chaque routeur a été interrogé et correspond à la configuration appliquée :

R1 : hostname R1

R2 : hostname R2

Vérification de la version SSH : La version SSH configurée sur les deux routeurs est la version 2, comme attendu :

R1 : ip ssh version 2

R2 : ip ssh version 2

Vérification des ACL sur les lignes VTY : Les ACL appliquées aux lignes VTY sont correctes et correspondent à la règle access-class VTY-ACL in sur les deux routeurs.

Vérification des utilisateurs configurés : Les utilisateurs avec les niveaux de privilège et mots de passe chiffrés ont été correctement configurés :

R1 : Utilisateurs cisco et admin

R2 : Utilisateurs cisco et admin

Vérification des bannières de connexion : Les bannières de connexion (login et message du jour) ont également été configurées avec succès sur les deux routeurs.

En résumé, ce playbook de vérification a confirmé que la configuration initiale appliquée sur les routeurs R1 et R2 est en place et respecte les attentes, sans aucune modification supplémentaire nécessaire.

Scénario 3 : Configuration des VLANs et Sécurité des Ports sur les Switches Cisco

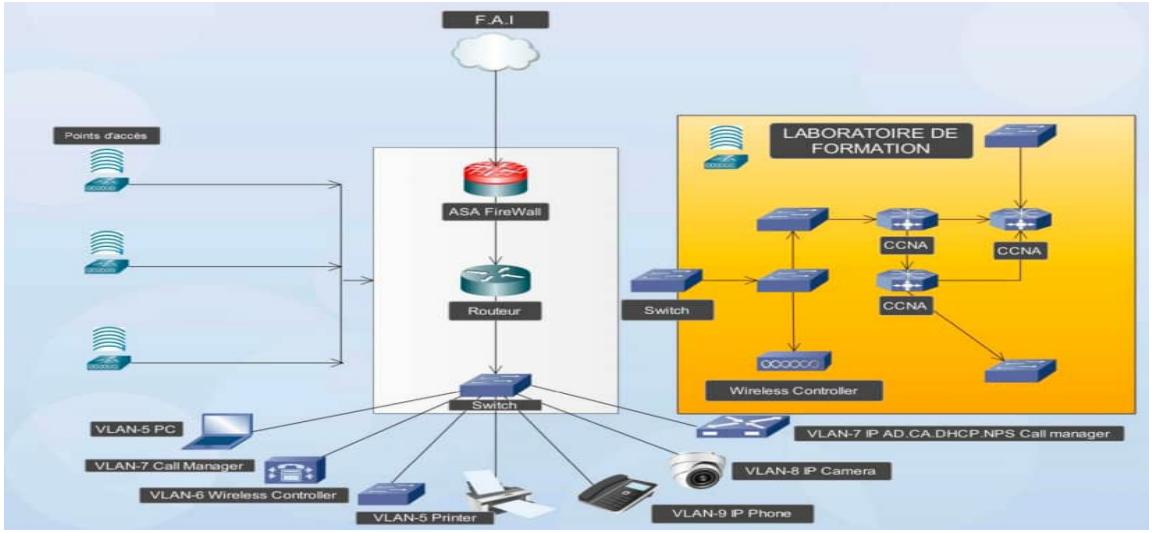


Figure 58 :Topologie Vlan ISARE

ISARE dispose d'un réseau composé de plusieurs services connectés à des switches. Chaque service a ses propres besoins en termes de connectivité et de sécurité. L'objectif est de segmenter le réseau à l'aide de VLANs, de configurer l'encapsulation trunk pour les liaisons entre les switches, et de mettre en place des mesures de sécurité pour les ports, notamment la limitation du nombre de périphériques connectés et la sécurisation des adresses MAC.

L'entreprise a les services suivants :

- **PC des employés** : Connectés au VLAN 5.
- **Contrôleur sans fil** : Géré sur le VLAN 6.
- **Gestion d'appels (Call Manager)** : Configuré sur le VLAN 7.
- **Caméras de sécurité IP** : Connectées au VLAN 8.
- **Téléphones IP** : Connectés au VLAN 9.

Objectifs du scénario :

- **Créer des VLANs pour chaque service.**
- **Configurer les interfaces des switches pour attribuer les bons VLANs.**
- **Configurer les interfaces `trunk` entre les switches pour permettre le passage des VLANs.**

- **Sécuriser les ports pour limiter le nombre de périphériques et activer la fonctionnalité de port security.**

- Rédaction du playbook ici nous allons utilisez le playbook directement et nom les rôles

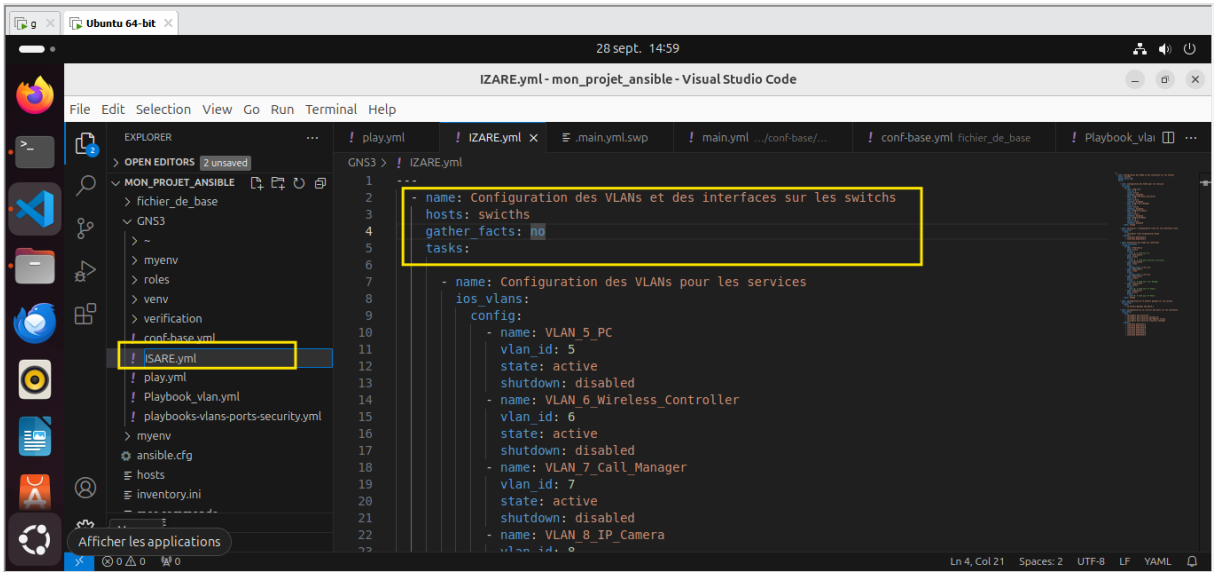


Figure 61 :Rédaction du playbook

Figure 60 : Rédaction du playbook

- Explication de ce Playbook :

VLANs pour les services :

Figure 59/ Rédaction du playbook

- Chaque VLAN est configuré avec un nom qui correspond au service (PC, Wireless Controller, Call Manager, IP Camera, IP Phone).
- Les VLANs sont activés sur tous les switches.

Configuration des Trunks :

- Les interfaces Ethernet2/0 et Ethernet2/1 sont configurées en trunk pour permettre la transmission des VLANs (5, 6, 7, 8, 9).

Affectation des VLANs :

- Les interfaces sont affectées aux VLANs correspondants :
  - VLAN 5 pour les PC.
  - VLAN 6 pour le Wireless Controller.
  - VLAN 7 pour le Call Manager.
  - VLAN 8 pour les IP Cameras.
  - VLAN 9 pour les IP Phones.

Configuration de la passerelle par défaut :

- La passerelle est configurée pour permettre aux switches de communiquer avec d'autres réseaux via le routeur.

Sécurité des ports :

- La sécurité des ports est appliquée sur les interfaces en mode access pour limiter le nombre d'adresses MAC par port et prévenir des attaques de type "MAC Flooding".
- Les ports seront mis en shutdown si une violation de sécurité survient.

RESULTAT DE L'automatisation

```
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ ansible-playbook IZARE.yml

PLAY [Configuration des VLANs et des interfaces sur les switchs] ***********************************************************

TASK [Configuration des VLANs pour les services] ******************************************************************************
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
fatal: [S4]: FAILED! => {"changed": false, "msg": "[Errno None] Unable to connect to port 22 on 192.168.0.8"}
changed: [S1]
changed: [S5]
changed: [S2]
changed: [S3]

TASK [Configurer l'encapsulation trunk sur les interfaces trunk] ******************************************************************
[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: [S1]
changed: [S3]
changed: [S5]
changed: [S2]

TASK [Affectation des VLANs aux interfaces] ***********************************************************************************
changed: [S1]
```

Figure 63 : RESULTAT DE L'automatisation

```
28 sept. 15:16
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3
TASK [Affectation des VLANs aux interfaces] ***********************************************************************************
changed: [S1]
changed: [S2]
changed: [S5]
changed: [S3]

TASK [Configuration de la default gateway sur les switchs] ********************************************************************
changed: [S1]
changed: [S2]
changed: [S3]
changed: [S5]

TASK [Configuration de la sécurité des ports sur les interfaces] **************************************************************
changed: [S2]
changed: [S5]
changed: [S1]
changed: [S3]

PLAY RECAP ********************************************************************************************************************
S1                         : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S2                         : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S3                         : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S4                         : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
S5                         : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 62 : RESULTAT DE L'automatisation

Résumé des résultats

- **S4 a échoué** : L'erreur indique une impossibilité de se connecter à l'équipement S4 via le port 22 (SSH). Cela peut être dû à un problème de réseau ou à une mauvaise configuration du service SSH sur ce switch.
- **S1, S2, S3, S5 ont réussi** : Les autres switches ont correctement exécuté les tâches.

Détails des tâches réussies :

1. **Configuration des VLANs**

- ✓ Les VLANs 10, 20, et 30 ont été configurés avec succès sur les interfaces des switches S1, S2, S3 et S5. Chaque switch a bien pris en compte la configuration des VLANs pour le trafic.

2. **Configuration de l'encapsulation trunk** :

   - ✓ L'encapsulation dot1q sur les interfaces trunk a été appliquée. Un avertissement indique que pour assurer l'idempotence (pour éviter que la tâche ne soit réappliquée inutilement), les lignes de configuration doivent correspondre à celles présentes dans la configuration active. Toutefois, cela n'a pas affecté l'exécution.

3. **Affectation des VLANs aux interfaces** :

   - ✓ Chaque interface associée aux VLANs d'accès et trunk a été configurée correctement. Le script a modifié les interfaces comme prévu pour inclure les VLANs 10, 20, et 30.

4. **Configuration de la passerelle par défaut** :

   - ✓ La configuration de la passerelle par défaut a été déployée sur tous les switches sauf S4. Cela garantit que les switches peuvent router les paquets vers l'extérieur du réseau local.

5. **Sécurité des ports** :

   - ✓ La sécurité des ports a été activée sur toutes les interfaces désignées, limitant le nombre de périphériques autorisés à se connecter et activant le mode de sécurité shutdown en cas de violation de la sécurité (comme un port qui voit trop de nouvelles adresses MAC).

Vérification avec un Playbook

```yaml
---
- name: Vérification de la configuration des switches
  hosts: switches
  gather_facts: no
  tasks:

    - name: Vérification des VLANs configurés sur le switch
      ios_command:
        commands:
          - show vlan brief
      register: vlan_output

    - name: Afficher la liste des VLANs configurés
      debug:
        msg: "{{ vlan_output.stdout_lines }}"

    - name: Vérification de l'état des interfaces trunk
      ios_command:
        commands:
          - show interfaces trunk
      register: trunk_output
```

Figure 64 :Playbook de Vérification

```yaml
    - name: Afficher l'état des interfaces trunk
      debug:
        msg: "{{ trunk_output.stdout_lines }}"

    - name: Vérification de la configuration de la passerelle par défaut
      ios_command:
        commands:
          - show ip route
      register: gateway_output

    - name: Afficher la configuration de la passerelle par défaut
      debug:
        msg: "{{ gateway_output.stdout_lines }}"

    - name: Vérification de la sécurité des ports
      ios_command:
        commands:
          - show port-security interface
      register: port_security_output

    - name: Afficher l'état de la sécurité des ports
      debug:
        msg: "{{ port_security_output.stdout_lines }}"
```

Figure 65 : Playbook de Vérification

```
PLAY [Vérification de la configuration des switches] ***********************************************************************

TASK [Vérification des VLANs configurés sur le switch] *********************************************************************
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
ok: [S1]

TASK [Afficher la liste des VLANs configurés] ******************************************************************************
ok: [S1] => {
    "msg": [
        [
            "VLAN Name                             Status    Ports",
            "---- -------------------------------- --------- -------------------------------",
            "1    default                          active    Et0/0, Et0/1, Et1/0, Et2/3, Et3/2, Et3/3",
            "5    VLAN_5_PC                        active    Et1/2",
            "6    VLAN_6_Wireless_Controller       active    Et1/3",
            "7    VLAN_7_Call_Manager              active    Et2/2",
            "8    VLAN_8_IP_Camera                 active    Et3/0",
            "9    VLAN_9_IP_Phone                  active    Et3/1",
```

Figure 66 : Exécution du playbook

```
        },
        "item": "Ethernet1/2",
        "stdout": [
            "Port Security              : Enabled\nPort Status                 : Secure-up\nViolation Mode              : Shut
down\nAging Time                  : 0 mins\nAging Type                  : Absolute\nSecureStatic Address Aging : Disabled\nMaximum
 MAC Addresses      : 10\nTotal MAC Addresses        : 2\nConfigured MAC Addresses   : 0\nSticky MAC Addresses       : 2\nLast S
ource Address:Vlan   : aabb.cc00.0400:5\nSecurity Violation Count   : 0"
        ],
        "stdout_lines": [
            [
                "Port Security              : Enabled",
                "Port Status                : Secure-up",
                "Violation Mode             : Shutdown",
                "Aging Time                 : 0 mins",
                "Aging Type                 : Absolute",
                "SecureStatic Address Aging : Disabled",
                "Maximum MAC Addresses      : 10",
                "Total MAC Addresses        : 2",
                "Configured MAC Addresses   : 0",
                "Sticky MAC Addresses       : 2",
                "Last Source Address:Vlan   : aabb.cc00.0400:5",
                "Security Violation Count   : 0"
            ]
```

Figure 67 : Exécution du playbook

Conclusion

L'automatisation via Ansible a permis de configurer avec succès les VLANs, les trunks, la passerelle par défaut et la sécurité des ports sur quatre switches. Nous pouvons diagnostic doit être effectué pour résoudre le problème de connexion avec S4.

SCÉNARIO 4 : SÉCURITÉ AVEC ANSIBLE VAULT

Dans un environnement réseau d'entreprise, l'administrateur système doit assurer la sécurité des données sensibles tout en garantissant un accès approprié aux différents équipements réseau. L'entreprise utilise des routeurs et des commutateurs Cisco, où plusieurs VLANs ont été configurés pour segmenter le trafic réseau en fonction des départements (comme le service de finance, de développement et des ressources humaines).

CONTEXTE ET DÉFIS

- **Protection des Informations Sensibles :** L'administrateur doit gérer les mots de passe des équipements réseau, notamment ceux des switchs et des routeurs, qui sont souvent stockés en clair dans les fichiers de configuration. Cette pratique peut conduire à des failles de sécurité en cas d'accès non autorisé aux fichiers.

- **Gestion des Accès :** Les différents départements ont des niveaux d'accès différents. Les employés du service de finance ne devraient pas avoir accès aux équipements utilisés par le service de développement. Cela nécessite une gestion fine des droits d'accès.
- **Conformité et Audits :** L'entreprise doit se conformer aux normes de sécurité des données (comme le RGPD ou PCI-DSS) qui imposent des exigences strictes sur la gestion des informations sensibles.

Objectif du Scénario

L'administrateur réseau va utiliser Ansible Vault pour chiffrer les mots de passe des équipements Cisco, garantissant ainsi que ces informations sensibles ne soient pas exposées. Le but est de démontrer comment Ansible Vault peut être intégré dans des playbooks pour déployer des configurations sécurisées sans compromettre la sécurité des données.

Dans notre projet Ansible, plusieurs fichiers sont considérés comme vulnérables en raison du stockage d'informations sensibles. Voici une liste des fichiers les plus à risque :

host_vars/ : Ce répertoire contient des fichiers qui définissent des variables spécifiques pour des groupes d'hôtes ou des hôtes individuels. Les paramètres de connexion, y

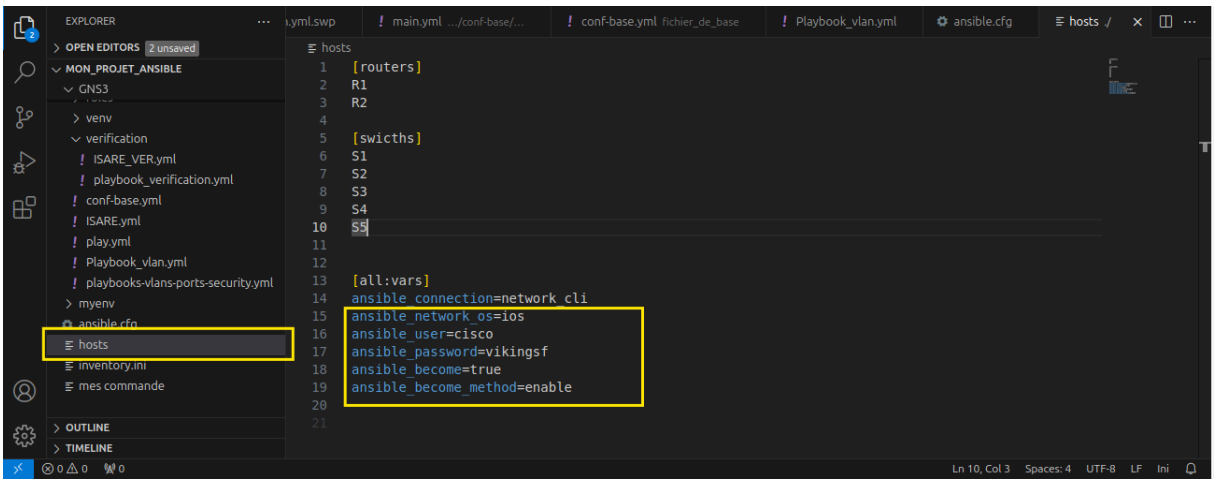


Figure 68 : Fichier hosts claisre

compris les adresses IP, les mots de passe et les, sont souvent stockés ici. En raison de la nature sensible de ces informations, il est crucial de sécuriser ce répertoire afin de prévenir toute exposition accidentelle.

```
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3/v...   callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible$ ansible-vault
usage: ansible-vault [-h] [--version] [-v] {create,decrypt,edit,view,encrypt,encrypt_string,rekey} ...
ansible-vault: error: the following arguments are required: action

usage: ansible-vault [-h] [--version] [-v] {create,decrypt,edit,view,encrypt,encrypt_string,rekey} ...

encryption/decryption utility for Ansible data files

positional arguments:
  {create,decrypt,edit,view,encrypt,encrypt_string,rekey}
    create              Create new vault encrypted file
    decrypt             Decrypt vault encrypted file
    edit                Edit vault encrypted file
    view                View vault encrypted file
    encrypt             Encrypt YAML file
    encrypt_string      Encrypt a string
    rekey               Re-key a vault encrypted file

options:
  --version             show program's version number, config file location, configured module search path, module location,
                        executable location and exit
  -h, --help            show this help message and exit
```

Figure 69 : Paramètre Ansible vault

Ansible Vault est un utilitaire permettant de chiffrer et de déchiffrer des fichiers de données sensibles dans des projets Ansible. Il garantit que des informations sensibles, telles que des mots de passe ou des clés d'API, restent protégées tout en étant intégrées dans des playbooks Ansible.

Utilisation de la commande

La commande ansible-vault nécessite un argument d'action, qui définit l'opération que vous souhaitez effectuer. Voici un aperçu des actions possibles :

- ✓ **create** : Créer un nouveau fichier chiffré.
- ✓ **decrypt** : Déchiffrer un fichier chiffré.
- ✓ **edit** : Éditer un fichier chiffré.
- ✓ **view** : Afficher le contenu d'un fichier chiffré sans le déchiffrer complètement.
- ✓ **encrypt** : Chiffrer un fichier YAML existant.
- ✓ **encrypt_string** : Chiffrer une chaîne de caractères.
- ✓ **rekey** : Changer la clé utilisée pour chiffrer un fichier.

➢ Nous allons crypter ce fichier pour le protéger.

Nous avons utilisé la commande `ansible-vault encrypt` pour chiffrer le fichier `hosts`, rendant son contenu illisible sans le mot de passe approprié. Ce processus protège les données sensibles, telles que des mots de passe ou des identifiants, et empêche tout accès non autorisé.

```
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3/v...   callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible$ ls
 ansible.cfg   fichier_de_base   GNS3   hosts   inventory.ini   'mes commande '   myenv
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible$ ansible-vault encrypt hosts
New Vault password:
Confirm New Vault password:
Encryption successful
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible$ cat hosts
$ANSIBLE_VAULT;1.1;AES256
6635323430643565366366165336336366262626139633234346339633838363761323934366466 39
6435386164356365643531336131313762353838373762640a396166326266393836333662343064
3030356633303396165376463373465383331356432306131353138363031363363346564613430
3836393163393532620a313538303161666332373435633135323965633446362356231323333636
3830353462346239633261616634343438303233353331343837353531366639356616336646338
3531373134343035633344386131373231633134363334616337613663366465355658363343437 30
3634376532623466363630353638386130616362373434383762323164333430373633231393161
3839373364373234373236303732643137313365363066343039373232346261336663362646138
3736316532646634653365356364366664653535386366306465343632313062396532396366633 3
3164623362663666303862373536613965376332346133313366376533383633376653932653534
3933303139376635656431656430393863353337326664663563306430616363666433865306264
3433373961633034653265363233666563333432323530303961613263303630626535373330613 9
3334333462353765343233643531333161653265653339646530306436313164663534353939393 9
3139316562623331656162363037663162633636131353934 63
```

Figure 70 : Cryptage du fichier hosts

Comme le fichier ISARE comporte adresse de la passerelle réseaux nous allons crypter le fichier correspondant au Vlan et a la sécurité des ports.

```
1 oct. 18:01
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3
callmeroot@callmeroot-VMware-Virtual-Platform: ~/mon_projet_ansible/GNS3   callmeroot@callmeroot-VMware-Virtual-Platform: /etc/ansible
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ ansible-vault encrypt ISARE.yml
New Vault password:
Confirm New Vault password:
Encryption successful
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ cat ISARE.yml
$ANSIBLE_VAULT;1.1;AES256
6264663366323137306230396639303338666164653635393035613363636332633066376166 3263
3463326236353861376538306539653564623831623031610a363166663362383565316534653465
3665346333366393666663861313536306136386635376333643264636365376665393732366396 3
3236363636313465610a343639653737643233323766396138373166383433393533303265323262
3933623361366130383639326538343464303332643037386536313962356435326132653736306 4
6362363932326439303963373137666533373033373165356137623561366332343932653533393 8
3232653365316436303366303130323536363135363932313463633839613934376533231393865
3239306338313430386337646430333061313836616562316233633235356365633464386531663 2
6539363537646136653738333738316634613665306439626665323033363639346234643965376 6
3334353531393036336464633765613432393561353137383535393933366136376537653366333 2
6635363637643835323066636561396137393065346137363034346466636337356466663034623 0
3131323837306139663631343464623338393330316433623565323163363037656334653437616 4
3164393230663065616534336238323231343665343033376631386332363964663638623063646 1
6261313361613062373833353062313932316133323966623866636562363031343632366662303 1
3635356653039303361313533366303732616132393539323338636131306363666234326233 3
3636633061383837666432333665656336343038393033653663393633373061316531316464643 3
```

Figure 71 : Cryptage du fichier ISARE.yml

Ici nous allons utiliser option **View** pour visualiser les dossiers crypter avec l'outil **Vault**

```
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ ansible-vault view ISARE.yml
Vault password:
ERROR! Decryption failed (no vault secrets were found that could decrypt) on ISARE.yml for ISARE.yml
callmeroot@callmeroot-VMware-Virtual-Platform:~/mon_projet_ansible/GNS3$ ansible-vault view ISARE.yml
Vault password:
---
  - name: Configuration des VLANs et des interfaces sur les switchs
    hosts: swicths
    gather_facts: no
    tasks:

      - name: Configuration des VLANs pour les services
        ios_vlans:
          config:
            - name: VLAN_5_PC
              vlan_id: 5
              state: active
              shutdown: disabled
            - name: VLAN_6_Wireless_Controller
              vlan_id: 6
              state: active
```

Conclusion

En conclusion, l'intégration d'Ansible dans notre infrastructure réseau a permis une automatisation efficace et sécurisée à travers divers scénarios, notamment la configuration des routeurs et switches, la gestion des VLANs, et la sécurisation des ports via Ansible Vault. Chaque scénario a démontré la flexibilité d'Ansible dans l'automatisation des tâches récurrentes, réduisant les risques d'erreurs humaines et améliorant la cohérence des configurations sur l'ensemble des équipements. Grâce à des fonctionnalités comme le chiffrement des informations sensibles et l'exécution de commandes à grande échelle, Ansible s'est imposé comme un outil essentiel pour une gestion réseau centralisée, tout en garantissant la sécurité et la rapidité des déploiements.

CONCLUSION GÉNÉRALE

Ce rapport a synthétisé mon expérience de stage au sein du groupe **ISARE**, où j'ai eu l'opportunité de travailler sur **L'AUTOMATISATION DES TÂCHES DE GESTION RÉSEAU À L'AIDE D'ANSIBLE**. Ce stage a été non seulement une occasion d'appliquer mes connaissances théoriques dans un environnement professionnel, mais aussi un moyen de comprendre les défis et les exigences liés à la gestion moderne des infrastructures réseau. J'ai pu observer l'impact significatif de l'automatisation sur l'efficacité des opérations, la réduction des erreurs humaines et l'amélioration de la sécurité des configurations. En intégrant des outils d'automatisation, j'ai contribué à rationaliser les processus de gestion réseau, offrant ainsi à ISARE une approche plus dynamique et réactive face aux évolutions technologiques. À l'avenir, une question cruciale à explorer serait **: comment pourrions-nous intégrer des solutions de conteneurisation, comme Docker, avec Ansible pour améliorer encore la flexibilité et la portabilité des configurations réseau ?** Cette approche pourrait non seulement faciliter la gestion des environnements multi-cloud, mais aussi favoriser une innovation continue dans la manière dont ISARE aborde l'automatisation des infrastructures.

ANNEXE

GLOSSAIRE

Ansible : Un outil d'automatisation open source utilisé pour la gestion de configurations, le déploiement d'applications et l'orchestration de tâches sur des infrastructures informatiques.

Automatisation : Le processus d'utiliser des technologies pour effectuer des tâches sans intervention humaine, visant à améliorer l'efficacité et à réduire les erreurs.

Playbook : Un fichier de configuration en YAML utilisé par Ansible pour définir les tâches à exécuter sur des hôtes distants. Les playbooks contiennent des instructions pour la gestion des configurations, l'installation de logiciels, et plus encore.

Module : Un composant de base dans Ansible qui effectue une tâche spécifique, comme la gestion d'utilisateurs, le déploiement de logiciels, ou la configuration de services.

SSH (Secure Shell) : Un protocole de réseau utilisé pour accéder et gérer des systèmes informatiques à distance de manière sécurisée.

VLAN (Virtual Local Area Network) : Une méthode de segmentation de réseau qui permet de regrouper des dispositifs sur le même réseau logique, indépendamment de leur emplacement physique.

Inventaire : Un fichier ou une liste qui contient des informations sur les hôtes et les groupes d'hôtes gérés par Ansible, spécifiant leur adresse IP et d'autres paramètres.

Ansible Vault : Une fonctionnalité d'Ansible qui permet de chiffrer des fichiers ou des variables, garantissant la sécurité des informations sensibles telles que les mots de passe et les clés API.

Orchestration : La coordination de plusieurs tâches ou services pour automatiser des processus complexes, souvent impliquant plusieurs systèmes et technologies.

CI/CD (Continuous Integration/Continuous Deployment) : Un ensemble de pratiques qui permettent aux développeurs de fréquenter et de déployer des modifications de code de manière continue, réduisant ainsi les délais de livraison et améliorant la qualité.

Configuration Management : Une pratique de gestion qui vise à maintenir les systèmes informatiques dans un état souhaité, en automatisant la configuration et la maintenance des ressources.

Idempotence : Une propriété des opérations dans Ansible où l'exécution répétée de la même tâche produira le même résultat, sans causer d'effets indésirables supplémentaires.

Role : Une structure dans Ansible qui regroupe des playbooks, des modules, et d'autres fichiers en un ensemble réutilisable, facilitant l'organisation et la gestion des configurations.

Task : Une action unique exécutée par Ansible dans un playbook, définie pour accomplir une tâche spécifique sur un ou plusieurs hôtes.

Handler : Un type de tâche qui ne s'exécute que si elle est notifiée par une autre tâche, généralement utilisé pour redémarrer des services après une modification de configuration.

WEBOGRAPHIE

d'Ansible, documentation officielle. 2024. https://docs.ansible.com/. [En ligne] 14 septembre 2024.

Galaxy, Ansible. 2024. https://galaxy.ansible.com/. [En ligne] 20 aout 2024.

2024. https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html. [En ligne] 15 Aout 2024.

Modules,Ansible.Networking.2024. https://docs.ansible.com/ansible/latest/collections/ansible/netcommon/index.html. [En ligne] 26 septembre 2024.

Platform,RedHat.Ansible.Automation.2024. https://www.redhat.com/en/ansible-automation-platform. [En ligne] 25 juillet 2024.

Table des matières

DEDICACE I

RESUME III

ABSTRACT IV

INTRODUCTION GENERALE 1

PARTIE I : PHASE D'INSERTION 2

CHAPITRE I : PRESENTATION DE L'ENTREPRISE ISARE 3

PARTIE 2 : PHASE TECHNIQUE 3

CHAPITRE I : ANALYSE DU PROJET 4

1.1 PRESENTATION DU PROJET 5

1.2 ETUDE DE L'EXISTANT 6

1.2.1 Architecture du réseau existant 6

1.2.2 Description de l'existant 8

1.3 CRITIQUE DE L'EXISTANT 8

1.4 PROBLEMATIQUE 9

CHAPITRE II : CAHIER DE CHARGE 10

2.1 CONTEXTE 11

2.2 OBJECTIF GENERAL 11

2.2.1 OBJECTIFS SPECIFIQUES 11

2.3 PLANIFICATION DES TACHES 12

2.4 ESTIMATION ET CONTRAINTES DU PROJET 13

2.4.1 CONTRAINTES DU PROJET 13

2.4.2 EVALUATION FINANCIERE DU PROJET 14

❖ Estimation financière des ressources logicielles du projet Le tableau ci-après présente le bilan des logiciels utilisés pour la réalisation de ce travail. 14

2.5 LES LIVRABLES 16

CHAPITRE III : ETAT DE L'ART 17

3.1 Présentation générale de l'automatisation dans le contexte réseau. 18

3.1.1 Importance et rôle de l'automatisation dans la gestion des infrastructures réseau modernes. 19

3.2 Histoire et développement d'Ansible. 21

3.2.1 Caractéristiques principales d'Ansible 21

3.2.2 Comparaison avec d'autres outils d'automatisation (Puppet, Chef, SaltStack). 23

3.2.3 Analyse des prix : 25

3.2.4 Avantages d'Ansible pour l'automatisation des équipements réseau. 26

3.3 Architecture d'Ansible 27

3.3.1 Fonctionnement d'Ansible : utilisation de SSH pour la communication avec les hôtes. 29

3.4 Applications de l'automatisation avec Ansible dans les environnements réseau 30

CHAPITRE IV : IMPLEMENTATION DES SOLUTIONS 33

4.1 PREREQUIS D'INSTALLATION 34

4.1.1 ENVIRONNEMENT MATERIEL 34

4.1.2 ENVIRONNEMENT LOGICIEL 34

4.1.3 PRESENTATION DE LA TOPOLOGIE 35

4.2 MISES EN ŒUVRE DU SYSTEME 36

4.2.1 Préparation de l'environnement de travail 36

4.3 Automatisation des tâches 51

4.3.1 Les playbooks 52

CONTEXTE ET DÉFIS 73

CONCLUSION GÉNÉRALE 78

ANNEXE I

GLOSSAIRE II

WEBOGRAPHIE IV