Table des matières

Liste des figures

Sigles et Abréviations

Abréviation	Signification
DHCP	Dynamic Host Configuration Protocol
DLP	Data Lost Prevention
DNS	Domain Name System
ICMP	Internet Control Message Protocol
FTP	File Transport Protocol
IGMP	Internet Group Management Protocol
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
TTL	Time To Live
UDP	User Datagram Protocol
RIP	Routing Information Protocol
VoIP	Voice over IP
NAT	Network Address Translation
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
VPN	Virtual Private Network
NFS	Network File System
SNMP	Simple Network Management Protocol
ARP	Address Resolution Protocol
CPU	Central Processing Unit
POP	Post Office Protocol
EPEL	Extra Packages for Entreprise Linux
RHEL	Red Had Entreprise Linux
SSL	Secure Sockets Layer
TLS	Transport Layer Security
NTP	Network Time Protocol

INTRODUCTION GENERALE

Une entreprise utilise des ordinateurs pour faire du travail. Ces ordinateurs sont souvent connectés entre eux pour faciliter le partage des données, des matériels et logiciels. Une des techniques pour améliorer ce partage, la circulation et les conditions d'utilisation des données est la mise en place d'un réseau intranet dans cette entreprise.

L'intranet est un réseau informatique local et privé propre à une organisation (entreprise, administration …) qui utilise les techniques de communication d'internet IP mais ne s'ouvre pas aux connexions publiques. L'intranet offre beaucoup d'avantages, de plus, il utilise les services de l'Internet mais internes à l'entreprise.

Au sein de l'intranet pourront tourner des applications, des logiciels, des serveurs, … et tous auront besoin de configuration, de suivi, de maintenance, ….

Il serait vraiment difficile pour une personne de devoir gérer lui-même toutes ces tâches d'où l'intérêt de penser à une manière de réduire l'interaction humaine avec les systèmes : les logiciels d'automatisation.

On est dans une ère de cloud, de conteneurs et micro services et tous favorisent l'automatisation. Cette dernière accélère la fourniture d'infrastructures et d'applications informatiques en automatisant les processus manuels qui nécessitaient auparavant une intervention humaine. Un logiciel est utilisé pour configurer les instructions, processus ou règles répétables qui font gagner du temps et libèrent le personnel informatique pour les tâches plus stratégiques. Dans notre cas Ansible sera utilisé pour gérer l'automatisation au niveau de notre intranet.

Par ailleurs, les attaques informatiques deviennent de plus en plus fréquentes, pour cela, il est nécessaire de penser à la sécurité au niveau de l'intranet pour une infrastructure sûre.

Les utilisateurs auront besoin d'accéder à Internet. Ce qui ouvre une porte pour les menaces d'où la nécessité de mettre en place un firewall qui est une forme de protection permettant à un réseau d'être connecté à l'Internet tout en maintenant un certain degré de sécurité.

Cette partie de sécurité sera mise en place à l'aide d'Ansible.

Alors de quoi s'agit-il ? Que proposons-nous d'accomplir ?

Tout cela sera illustré en quatre chapitres :

Le premier chapitre consiste à parler de la problématique et des objectifs du sujet.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

Le second chapitre aborde une étude théorique des généralités sur les réseaux et sur un aperçu sur le système d'exploitation Linux. Il est nécessaire de connaitre un réseau, son fonctionnement et ses architectures et vu que notre intranet fonctionnera sur le système d'exploitation Linux, il devient nécessaire d'avoir une idée de ce système d'exploitation.

Le troisième chapitre porte sur les généralités d'intranet. Nous jugeons nécessaire de connaître ce qu'est un intranet, son fonctionnement afin de pouvoir mener à bien son déploiement.

Le quatrième chapitre parle de l'automatisation et des généralités de l'outil Ansible qui sera utilisé pour automatiser certains processus et pour sécuriser l'infrastructure.

Enfin le cinquième et dernier chapitre élabore le déploiement de l'intranet. Pour cela nous allons illustrer la mise en place d'un intranet d'entreprise à l'aide de MediaWiki, projet libre de référence, puisque c'est celui utilisé par Wikipédia. Il sera question d'aborder tous les aspects d'une telle mise en place, en commençant par la mise en place d'un serveur web (Apache dans notre cas) et d'une base de données (Maria DB) afin de déployer MediaWiki pour outiller notre intranet. Nous ne nous limiterons pas qu'à ces aspects fonctionnels, il sera nécessaire aussi de parler de l'optimisation du serveur et la gestion de sa maintenance. Dans cette partie, sera abordée la configuration de SSH afin de disposer d'un accès à distance sûr et aussi la personnalisation du serveur pour s'assurer d'avoir le meilleur confort d'utilisation possible lors de l'accomplissement des tâches de maintenance. Il est impossible de passer sous silence la difficile et ardue tâche de la sécurisation du système. Nous aborderons le rôle des différents outils impliqués, évoquant la mise en place du « firewall », mais aussi de SELinux et le renforcement de la configuration de notre serveur web (Apache).

Chapitre 1 : Approche méthodologique du sujet

1.1. Problématique

Les entreprises ne cessent de grandir. Le nombre d'employés, de matériels, d'infrastructures augmentent tous les jours, ainsi la communication au sein de l'entreprise devient une priorité.

Le besoin de communication est plus fort, les données grandissent de plus en plus, le besoin partage d'informations entre les collaborateurs de l'entreprise devient une nécessité. Il faut donc penser à une manière de communication propre à l'entreprise c'est-à-dire accessible uniquement aux membres de l'entreprise d'où l'idée d'intranet.

1.2. Objectifs du projet

Les principaux objectifs du projet sont :

- ✓ Faciliter la communication ;
- ✓ Faciliter le partage des connaissances et des compétences ;
- ✓ Permettre aux membres de l'entreprise d'avoir accès à l'information en temps réel ;
- ✓ Permettre le décloisonnement des services ;
- ✓ Diminuer l'interaction humaine au niveau des taches informatiques.

1.3. Généralités sur la sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques, et humains nécessaires et mis en place pour conserver, rétablir, et garantir la sécurité des systèmes informatiques.

La sécurité informatique vise généralement cinq principaux objectifs :

- **Disponibilité** : permettant de maintenir le bon fonctionnement du système informatique ;
- **Confidentialité** : consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ;
- **Intégrité** : c'est-à-dire garantir que les données sont bien celles que l'on croit être ;
- **Authentification** : garantissant que l'information est associée à son auteur légitime ;
- **Non-répudiation** : garantissant que l'information ne peut faire l'objet d'un déni de la part de son auteur. ;

Chapitre 2 : Généralités sur les réseaux et le système d'exploitation linux

2.1. GENERALITES SUR LES RESEAUX

2.1.1. Définitions d'un réseau

Un réseau est un ensemble d'objets interconnectés les uns avec les autres. Il permet de faire circuler des éléments entre chacun de ces objets selon des règles bien définies.

Un réseau, en général, est le résultat de la connexion de plusieurs machines entre elles, afin que les utilisateurs et les applications fonctionnant puissent échanger des informations.

Le terme réseau, en fonction de son contexte, peut désigner plusieurs choses. Il peut désigner l'ensemble des machines, infrastructure informatique d'une organisation avec les protocoles qui sont utilisés, ce qui est le cas lorsque l'on parle de l'Internet.

C'est un ensemble d'ordinateurs (ou de périphériques) autonomes connectés entre eux et qui sont situés dans un certain domaine géographique.

Le terme réseau peut également être utilisé pour décrire la façon dont les machines d'un site sont interconnectées. C'est le cas lorsque l'on dit que les machines d'un site sont sur Ethernet, Token Ring, réseau en étoile, réseau en bus, ….

Le terme réseau peut également être utilisé pour spécifier le protocole qui est utilisé pour que les machines communiquent. On peut parler de réseau TCP/IP, Netbeni (protocole Microsoft), Dec Net (protocole DEC), IPX/SPX, …

Le terme réseau possède différentes significations. Alors chaque mot réseau il faut comprendre son sens.

Les Réseaux permettent :

- De partager les fichiers ;
- Le partage d'application : compilateur, système de gestion de base de données (SGBD) ;
- Partage d'imprimante ;
- L'interaction avec les utilisateurs connectés : messagerie électronique, conférence
- électronique, Talk, ... ;
- Le transfert de données en général (réseaux informatiques) ;
- Le transfert de la parole (réseaux téléphoniques) ;

- Le transfert de la parole, de la vidéo et des données (réseaux à intégration de services ou multimédia).

On compte généralement quatre catégories de réseaux informatiques différenciées par la distance maximale séparant les points les plus éloignés du réseau.

On distingue :

- Les PAN : Personal Area Network, ces réseaux personnels interconnectent sur quelques mètres les équipements personnels tels que GSM, portables, etc … ;
- Les LAN : Local Area Network, correspondent par leur taille aux réseaux intra-entreprises.

Ils servent au transport de toutes les informations numériques de l'entreprise.

En règle générale, les bâtiments à câbler s'étendent sur plusieurs centaines de mètres.

Les débits de ces réseaux vont aujourd'hui de quelques mégabits à plusieurs centaines de mégabits par seconde ;

- Les MAN : Metropolitain Area Network, permettent l'interconnexion des entreprises ou éventuellement des particuliers sur un réseau spécialisé à haut débit qui est géré à l'échelle d'une métropole. Ils doivent être capables d'interconnecter les réseaux locaux de différentes entreprises pour leur donner la possibilité de dialoguer avec l'extérieur ;
- Les WAN : Wide Area Network, sont destinés à transporter des données numériques sur des distances à l'échelle d'un pays, voire d'un continent ou de plusieurs continents ; Le réseau est soit terrestre, et il utilise en ce cas des infrastructures au niveau du sol, essentiellement de grands réseaux de fibre optique, soit hertzien, comme les réseaux satellites.

2.1.2. Différentes techniques de commutation

Le réseau doit permettre l'échange de messages entre les abonnés quelle que soit leur localisation. La commutation rassemble toutes les techniques qui réalisent la mise en relation de deux abonnés quelconques.

Il existe quatre techniques de commutation :

2.1.2.1. Commutation de circuits

Un chemin physique est établi à l'initialisation de la communication entre l'émetteur et le récepteur et reste le même pendant toute la durée de la communication. Si les deux correspondants n'ont pas de données à transmettre pendant un certain temps, la liaison restera

inutilisée. L'idée est de concentrer plusieurs correspondants sur une même liaison. Dans le cas où les communications seraient nombreuses, il faut prévoir des mémoires pour stocker des informations en attendant que la liaison soit disponible.

2.1.2.2. Commutation de messages

Un message est un ensemble d'information logique formant un tout (fichier, mail) qui est envoyé de l'émetteur vers le récepteur en transitant nœud à nœud à travers le réseau. On a un chemin logique par message envoyé. Le message ne peut être envoyé au nœud suivant tant qu'il n'est pas reçu complètement et sans erreur par le nœud actuel.

Remarque : La commutation de message nécessite la mise en place d'algorithmes de routage.

2.1.2.3. Commutation de paquets

Optimisation de la commutation de message qui consiste à découper les messages en plusieurs paquets pouvant être acheminés plus vite et indépendamment les uns des autres. Cette technique nécessite la mise en place de la numérotation des paquets.

2.1.2.4. Commutation de cellule

Commutation de paquets particulière. Tous les paquets ont une longueur fixe de 53 octets (1 paquet = 1 cellule de 53 octets). C'est la technique utilisée dans les réseaux ATM où un chemin est déterminé pour la transmission des cellules.

Commutation de cellule = superposition de deux types de commutation :

- commutation de circuit
-commutation de paquets.

Il utilise le mode connecté

- mode connecté : Demande explicite de connexion et de déconnexion.
- mode non connecté : Pas de demande de connexion.

2.1.3. Architecture des réseaux

Pour assurer le bon transfert de l'information avec une quantité de service suffisante, il faut prévoir une architecture logicielle. Une normalisation de l'architecture logicielle s'impose.

Deux grandes familles d'architecture se disputent dans le marché. La première provient de l'ISO. La deuxième est TCP/IP. Une troisième est celui de l'UIT-T, il s'agit de l'adaptation du modèle OSI pour prendre en compte les réseaux hauts-débit (réseau ATM).

2.1.3.1. Modèle de référence OSI

Le modèle de référence OSI publié en 1984 fut le modèle descriptif de réseau créé par l'ISO. Ce modèle propose aux fournisseurs un ensemble de normes assurant une compatibilité et une interopérabilité accrues entre divers types de technologies réseau produites par de nombreuses entreprises à travers le monde.

Elle comporte sept couches numérotées, chacune illustrant une fonction réseau bien précise.

Cette répartition des fonctions réseau est appelée organisation en couches. Le découpage du réseau en sept couches présente les avantages suivants :

- Division des communications sur le réseau en éléments plus petits et plus simples ;
- Uniformisation les éléments du réseau afin de permettre le développement et le soutien multi-constructeur ;
- Communication de différents types de matériel et de logiciel réseau entre eux ;
- Empêchement des changements apportés à une couche d'affecter les autres couches, ce qui assure un développement plus rapide ;
- Division les communications sur le réseau en éléments plus petits, ce qui permet de les comprendre plus facilement.

La figure suivante représente le modèle sept couches d'OSI :

Figure 1 : Couches fonctionnelles du modèle OSI

Source : www.rapport-gratuit.com

Cette figure permet de voir que les protocoles forment un empilement de blocs de constructions posés les uns sur les autres. C'est en cette raison de cette apparence que la structure est souvent appelée une pile (stack) ou pile de protocoles.

Pour permettre l'acheminement des données entre l'ordinateur source et l'ordinateur de destination, chaque couche possède sa propre fonction et la couche au niveau de l'ordinateur source doit communiquer avec sa couche homologue sur l'ordinateur de destination, chaque couche dépend de la fonction de service de la couche sous-jacente.

Examinons les fonctions de chaque couche :

- La couche physique détermine la caractéristique des matériels à utiliser pour la liaison physique entre équipement d'un réseau (câble, connecteur, concentrateur, commutateur). Elle a en charge la transmission des suites des bits sur les moyens physiques d'interconnexion mais aussi le traitement de signal (modulation, amplification, …) ;
- La couche liaison des données détermine la technique d'accès au media qui varie en fonction du type de réseau (Ethernet ou Token-Ring). Elle assure le transfert fiable de données par le média c'est-à-dire sans erreurs, sans duplication ni perte entre systèmes adjacentes (donc sur un seul circuit de données), l'adressage physique, notification des erreurs, contrôle de flux ;

- La couche réseau a pour rôle de gérer, d'établir et de maintenir les connexions de réseau entre deux systèmes d'extrémité et la sélection du meilleur chemin possible, c'est-à-dire l'adressage logique et le routage. Par ailleurs elle se charge des moyens fonctionnels et les procédures nécessaires pour échanger, entre les entités de transport, des unités du service de réseau ;
- La couche transport assure la connexion bout à bout des informations du réseau, transport des données entre les hôtes. Elle établisse l'ouverture et la fermeture des circuits virtuels, détecte les pannes et reprise des erreurs, gère le contrôle de flux ;
- La couche session ouvre, gère et ferme les sessions entre les applications. Cela comprend le lancement, l'arrêt et la resynchronisation de deux ordinateurs qui communiquent. Elle coordonne les applications lorsqu'elles interagissent sur deux hôtes qui communiquent ;
- La couche présentation assure la lisibilité des données pour le système de destination, indique le format des données, structure les données. Elle négocie la syntaxe de transfert des données pour les applications ;
- La couche application fournit les services de communication aux utilisateurs (opérateurs, périphériques, programme d'application). Elle fournit également des services réseaux aux processus d'applications (courrier électronique, transfert des fichiers et émulations des terminal).

2.1.3.2. Architecture du modèle TCP-IP

La pile de protocoles TCP/IP a été développée dans le cadre des recherches de la DARPA. Elle était initialement destinée à assurer les communications au sein de la DARPA. Ensuite, elle a été intégrée à la distribution Berkeley du système d'exploitation Unix. Aujourd'hui, la suite de protocoles TCP/IP est devenue la norme des communications inter-réseaux et sert de protocole de transport à Internet, ce qui permet à des millions d'ordinateurs de communiquer entre eux.

L'architecture TCP/IP prend comme modèle de référence le modèle OSI, mais avec seulement quatre couches fonctionnels : la couche application, la couche transport, la couche Internet et la couche d'accès au réseau. Autrement dit, certaines couches du modèle TCP/IP portent le même nom que des couches du modèle OSI. Il ne faut pas confondre les couches des deux modèles, car la couche application comporte des fonctions différentes dans chaque modèle. La figure suivante permet de visualiser ces quatre couches du modèle TCP/IP.

Figure 2 : Couches fonctionnelles du modèle TCP-IP

Source : www.rapport-gratuit.com

Le terme TCP/IP est un ensemble de protocole qui permet au réseau d'échanger des informations. Ces protocoles assurent la bonne circulation des flux à travers le réseau.
De même que pour le modèle OSI, les données sont passées vers le bas de la pile quand elles sont envoyées vers le réseau, et vers le haut quand elles sont reçues. La structure à quatre niveaux de TCP-IP est vue de la façon dont les données sont gérées alors qu'elles traversent la pile de protocoles vers le réseau physique. Chacune des couches de la pile ajoute des informations de contrôle afin d'assurer une livraison correcte. Ces informations de contrôle sont appelées un en-tête parce qu'elles sont placées devant les données à transmettre. Chaque couche traite toutes les informations qu'elle reçoit des couches supérieures en tant que données et place son propre en-tête avant elles. L'addition d'informations de distribution à chaque couche est appelée encapsulation.

Quand les données sont reçues, c'est l'inverse qui se produit. Chaque couche enlève son en-tête avant de passer les données à celle du dessus. Les informations reçues d'une couche sont interprétées en tant qu'en-tête et données.
Illustrons à l'aide d'une figure l'encapsulation des données (respectivement la désencapsulation) et les protocoles utilisés.

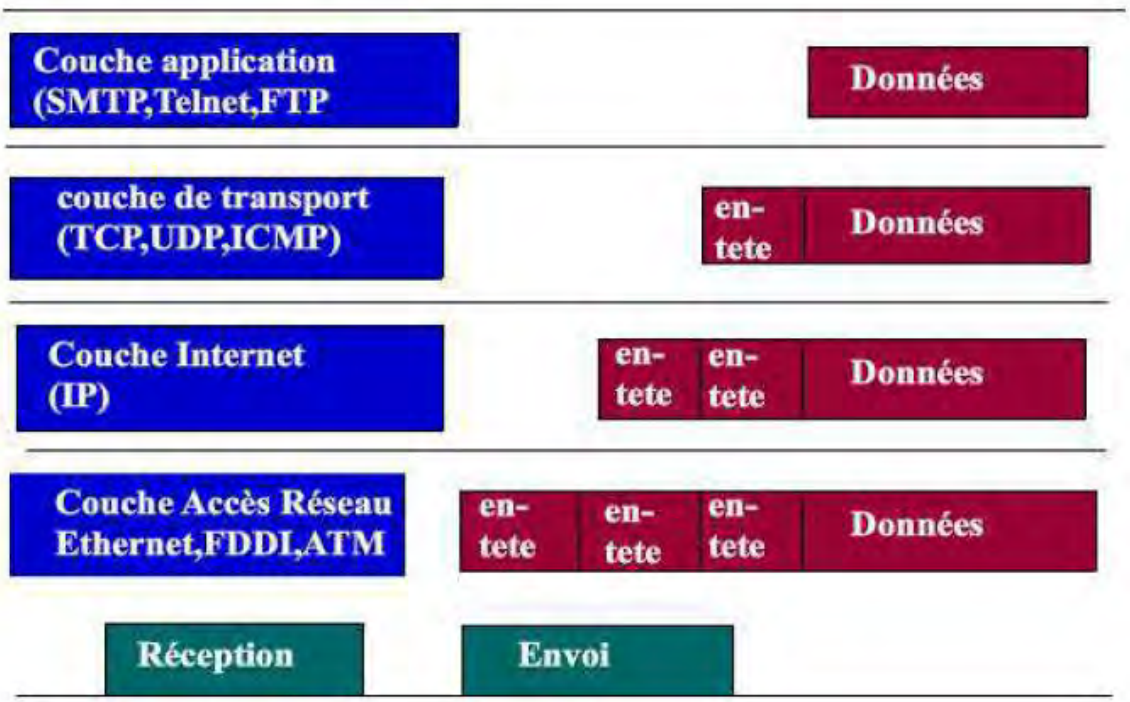

Figure 3 : Encapsulation des données

Source : www.rapport-gratuit.com

2.1.3.2.1. Couche d'accès réseau

La couche d'Accès Réseau est le plus bas niveau de la hiérarchie des protocoles TCP/IP. Les protocoles de cette couche fournissent le moyen de délivrer des données aux autres systèmes directement rattachés au réseau. Il définit la façon d'utiliser le réseau pour transmettre un datagramme IP.

La couche d'Accès Réseau doit connaitre les détails du réseau sous-jacent (sa structure de paquets, son adressage, etc.) Pour formater correctement les données transmises afin de se conformer aux contraintes du réseau. La couche d'Accès Réseau TCP/IP peut regrouper les fonctions des trois couches les plus basses du modèle OSI (Réseau, Liaison et Physique).

Les fonctions assurées à ce niveau comprennent l'encapsulation des datagrammes IP dans les trames transmises par le réseau et la correspondance des adresses IP vers les adresses physiques utilisées par le réseau.

2.1.3.2.2. Couche Internet

La couche située au-dessus de la couche d'Accès Réseau dans la hiérarchie des protocoles est la couche Internet. Le protocole Internet (IP), est au coeur de TCP/IP et le protocole le plus important de la couche Internet. IP fournit les services de livraison de paquets de base sur lesquels les réseaux TCP/IP sont construits. Tous les protocoles des couches supérieures (TCP, UDP) et Inferieures (Ethernet, FDDI, ATM, etc..) utilisent IP pour délivrer les données. Toutes les données TCP/IP passent par IP, qu'elles soient entrantes ou sortantes quelque soit sa destination finale.

Les fonctions d'IP sont :

- La définition du datagramme, qui est l'unité de base de transmission de l'internet ;
- La définition du principe d'adressage de l'Internet ;
- Le passage des données entre la couche d'Accès Réseau et la couche transport Hôte à Hôte ;
- Le routage des datagrammes vers les machines distantes ;
- La fragmentation et le réassemblage des datagrammes ;

Les données qui franchissent la couche IP, alias couche Internet, sont appelées " datagramme IP", datagramme Internet ou datagramme tout court. Voyons les caractéristiques de ces datagrammes.

2.1.3.2.3. Les datagrammes

Les données circulent sur l'Internet sous forme des datagrammes ou des paquets. Les datagrammes sont des données encapsulées, c'est-à-dire des données auxquelles on a ajouté des en-têtes correspondant à des informations sur leur transport ; telles que l'adresse IP destination, adresse IP source, etc.

Les données contenues dans les datagrammes sont analysées et éventuellement modifiées par les routeurs permettant leur transit.

Voici ce en quoi ressemble un datagramme :

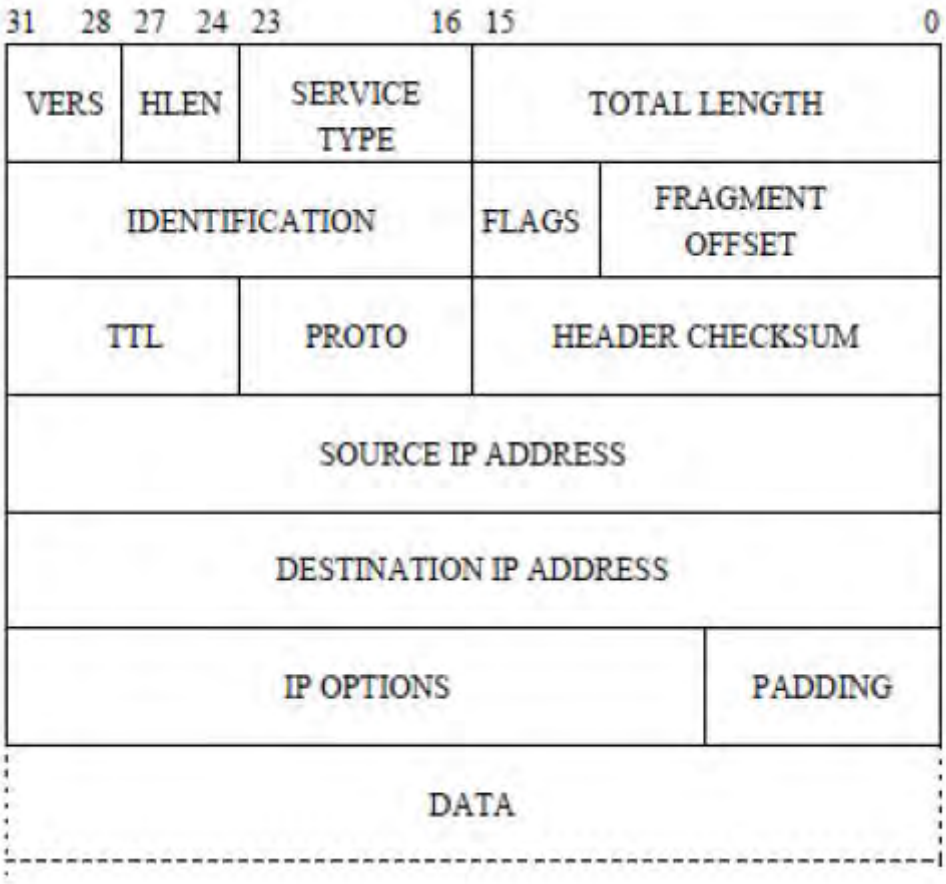

Figure 4 : Format d'un datagramme

Source : www.rapport-gratuit.com

Signification des différents champs :

- VERS (Version) : 4 bits qui spécifient la version du protocole IP. L'objet de ce champ est la vérification que l'´émetteur et le destinataire des datagrammes sont bien en phases avec la même version. Actuellement c'est la version 4 qui est principalement utilisé sur l'Internet, bien que quelques implémentations de la version 6 existent et soient déjà en expérimentation ;
- HLEN (Longueur d'en-tête) :4bits qui donnent la longueur de l'en-tête en mots de 4 octets ou 32bits. La taille standard de cet en-tête fait 5 mots, la taille maximale fait :(23 + 22 + 21 + 20) × 4 = 60 octets ;
- SERVICE TYPE (Type de service) : 8bits (4utiles), il indique la façon dont le datagramme doit être traité Suivant les valeurs de ce champ, le routeur peut privilégier un datagramme par rapport à un autre ;
- TOTAL LENGHT (Longueur totale) : Il indique la taille totale du datagramme en octets. La taille de ce champ étant de 2 octets, la taille totale du datagramme ne peut dépasser

65536 octets. Utilisé conjointement avec la taille de l'en-tête, ce champ permet de déterminer où sont situées les données ;

- IDENTIFICATION, FLAGS et FRAGMENT OFFSET (Identification, drapeau, déplacement de fragment) Ces mots sont prévus pour contrôler la fragmentation des datagrammes. Les données sont fragmentées car les datagrammes peuvent avoir à traverser des réseaux avec des supports physiques (MTU) plus petits que celui du premier support physique employé ;

- TTL " Time To Live " 8 bits, 255 secondes maximum de temps de vie pour un datagramme sur le réseau. Prévu à l'origine pour décompter un temps, ce champ n'est qu'un compteur décrémenté d'une unité a chaque passage du datagramme dans un routeur. Couramment la valeur de départ est 32 ou même 64. Son objet est d'´éviter la présence de paquets fantômes circulant indéfiniment. Si un routeur passe le compteur à zéro avant délivrance du datagramme, un message d'erreur est renvoyé à l'émetteur avec l'indication du routeur, le paquet en lui-même est perdu ;

- PROTO(Protocole) 8 bits pour identifier le format et le contenu des données. Ce champ permet de savoir de quel protocole est issu le datagramme, soit :

ICMP, IGMP, TCP, UDP ;

- HEADER CHECKSUM (Somme de contrôle de l'en-tête) : codée sur 16 bits permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été modifié, altéré pendant la transmission ;

- SOURCE ADDRESS (Adresse IP source) : ce champ représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre ;

- DESTINATION IP ADDRESS (Adresse IP destination) : Adresse IP du destinataire du datagramme ;

- IP OPTIONS : 24 bits pour préciser des options de comportement des couches IP traversées et destinatrices.

Les options les plus courantes concernent :

– Des problèmes de sécurité ;

– Des enregistrements de routes ;

– Des enregistrements d'heure ;

– Des spécifications de route à suivre ;

- PADDING Remplissage pour aligner sur 32 bits ;

- DATA : les données ;

2.1.3.2.4. Couche Transport

La couche transport est chargée des questions de qualité de service touchant la fiabilité, le contrôle de flux et la correction des erreurs. Les protocoles de transport définissent comment transmettre les messages entre les hôtes. Les deux protocoles de transport les plus courants sont le protocole TCP et le protocole UDP. Le protocole IP utilise ces protocoles de transport pour permettre aux hôtes de communiquer et de transmettre des données.

2.1.3.2.4.1. User Datagram Protocol (UDP)

L'User Datagram Protocol donne aux programmes d'application un accès direct à un service de transmission de datagrammes, comme celui que fournit IP. Les applications peuvent ainsi échanger des messages sur le réseau avec un minimum de surcharge due au protocole. UDP est un système d'acheminement « au mieux » qui ne nécessite pas d'accusé de réception. UDP est à préférer, notamment pour la lecture audio en continu, la vidéo et la voix sur IP (VoIP). Les accusés de réception ralentiraient la livraison, et les retransmissions ne sont pas souhaitables.

Voici le format du message UDP :

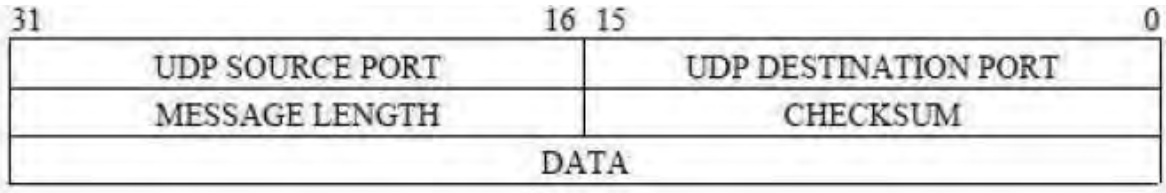

Figure 5 : Format de message UDP

Source : *www.rapport-gratuit.com*

- UDP SOURCE PORT Le numéro de port de l'´émetteur du paquet. Ce champ est optionnel, quand il est spécifié il indique le numéro de port que le destinataire doit employer pour sa réponse. La valeur zéro indique qu'il est inutilisé, le port 0 n'est donc pas celui d'un service valide ;
- UDP DESTINATION PORT Le numéro de port du destinataire du paquet ;
- MESSAGE LENGTH C'est la longueur du paquet, donc comprenant l'en-tête
- et le message
  - La longueur minimale est 8
  - La longueur maximale est 65 535 − $H(IP)$;
- CHECKSUM Le checksum est optionnel et toutes les implémentations ne l'utilisent pas.

S'il est employé, il porte sur un pseudo en-tête, Ce pseudo en-tête est prévu initialement pour apporter une protection en cas de datagrammes mal routés ;

- DATA : les données.

2.1.3.2.4.2. Transmission Control Protocol

Une application qui a besoin d'un accusé de réception, pour s'assurer que le message est bien transmis, utilise TCP. TCP découpe un message en petits morceaux appelés segments. Les segments, numérotés en séquence, sont ensuite passés au processus IP pour être assemblés en paquets. TCP conserve une trace du nombre de segments qui ont été envoyés à un hôte donné à partir d'une application spécifique. Si l'expéditeur ne reçoit pas d'accusé de réception au bout d'un certain temps, il suppose que les segments ont été perdus, et il les retransmet. Seule la partie du message qui a été perdue est renvoyée, pas l'intégralité. Sur l'hôte récepteur, TCP est responsable de la reconstitution des segments de message et de leur transmission à l'application. TCP est orienté connexion. Il établit une connexion logique de bout en bout entre les deux hôtes communiquant entre eux.

Voici un format de segment TCP :

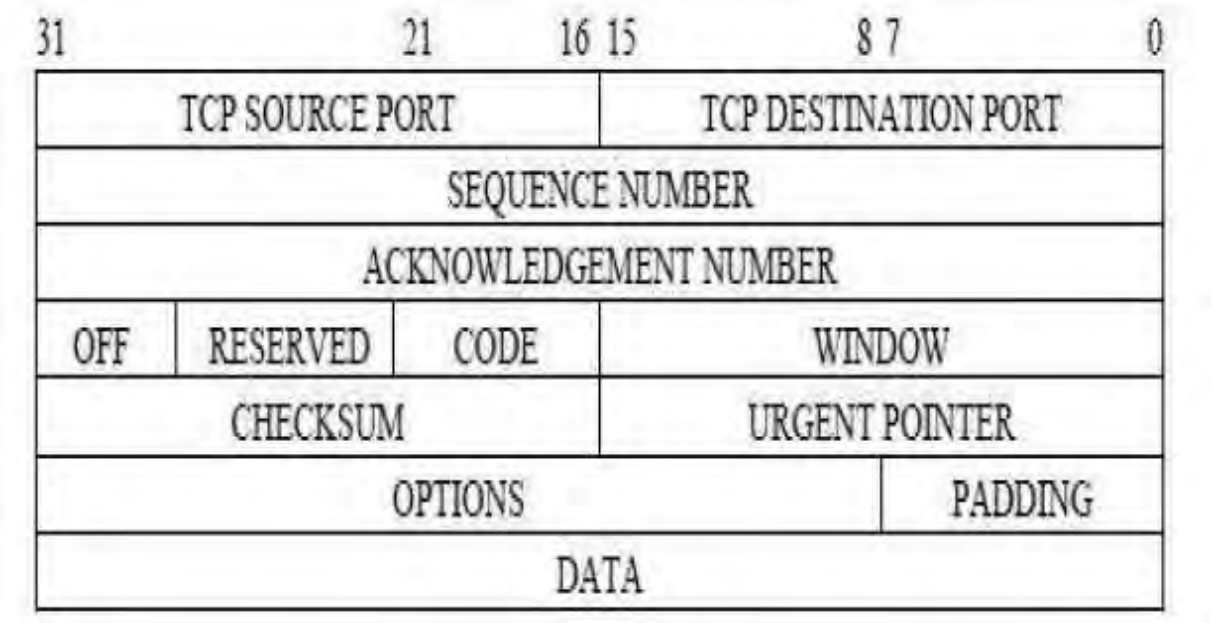

Figure 6 : Format d'un segment TCP

Source : *www.rapport-gratuit.com*

- TCP SOURCE PORT Le numéro de port de l'application locale ;
- TCP DESTINATION PORT Le numéro de port de l'application distante ;

- SEQUENCE NUMBER C'est un nombre qui identifie la position des données à transmettre par rapport au segment original ;

- ACKNOWLEDGEMENT NUMBER C'est un numéro qui identifie la position du dernier

- octet reçu dans le flux entrant ;

- OFF pour OFFSET, il s'agit d'un déplacement qui permet d'atteindre les données quand il y a des options. Codé sur 4 bits, il s'agit du nombre de mots de 4 octets qui composent l'en-tête. RESERVED Six bits réserves pour un usage futur ;

- CODE Six bits pour influer sur le comportement de TCP en caractérisant l'usage du segment ;

- WINDOW Le flux TCP est contrôlé de part et d'autre pour les octets compris dans une zone bien délimitée et nommée " fenêtre ". La taille de celle-ci est définie par un entier non signé de 16 bits, qui en limite donc théoriquement la taille à 65 535 octets ;

- CHECKSUM Un calcul qui porte sur la totalité du segment, en-tête et données ;

- URGENT POINTER Ce champ n'est valide que si le drapeau URG est armé ;

- OPTIONS C'est un paramétrage de TCP. Sa présence est détectée d`es lors que l'OFFSET est supérieur à 5 ;

- PADDING Remplissage pour se caler sur un mot de 32 bits ;

- DATAS Les données transportées. Cette partie est de longueur nulle à l'établissement de la connexion.

2.1.3.2.5. Couche Application

La couche application se trouve au sommet de l'architecture des protocoles TCP/IP. Cette couche comprend tous les processus qui emploient les protocoles de la couche transport pour délivrer les données. Ils existent de nombreux protocoles d'application. Les plus connues sont :

- FTP : ce protocole est un service fiable orienté connexion qui utilise le protocole TCP pour transférer des fichiers entre des systèmes qui le prennent en charge. Il gère les transferts bidirectionnels des fichiers ;

- TFTP : ce protocole est un service non orienté connexion qui utilise le protocole de datagramme utilisateur UDP. Il est utilisé sur le routeur pour transférer des fichiers de configuration et des images de la plate-forme logicielle, ainsi que pour transférer des fichiers entre des systèmes qui le prennent en charge. Il est utile dans certains LAN, car il s'exécute plus rapidement que le protocole FTP dans un environnement stable ;

- SMTP : ce protocole régit la transmission du courrier électronique sur les réseaux informatiques. Il ne permet pas de transmettre des données autres que du texte en clair ;
- DNS : ce protocole est utilisé par Internet pour convertir en adresses IP les noms de domaine et leurs nœuds de réseau annoncés publiquement ;
- Telnet : ce protocole permet d'accéder à distance à un autre ordinateur. Cela permet à un utilisateur d'ouvrir une session sur un hôte Internet et d'exécuter diverses commandes ;

Un client Telnet est qualifié d'hôte local. Un serveur Telnet est qualifié d'hôte distant.

- RIP : employé par des périphériques réseau pour échanger des informations de routages ;
- NFS : ce protocole est un ensemble de protocoles pour systèmes de fichiers distribués, développé par Sun Microsystems, permettant un accès aux fichiers d'un équipement de stockage distant, tel qu'un disque dur, dans un réseau ;
- SNMP : ce protocole permet de surveiller et de contrôler les équipements du réseau, ainsi que de gérer les configurations, les statistiques, les performances et la sécurité ;

TCP/IP emploie trios principe pour parvenir à délivrer les données entre deux hôtes :

- L'adressage
- Le routage
- Le multiplexage

2.1.3.2.5.1. L'adressage

L'adressage consiste à affecter une adresse IP sur l'hôte. Les adresses IP sont divisées en groupes de 8 bits séparées par des points, et représentées dans un format décimal. Une adresse IP contient une partie réseau et une partie hôte. On distingue cinq classes d'adresse :

La classe A et B : réservée pour des applications de trafic très élevé ;

La classe C : utilisé pour les réseaux privés ;

La classe D : pour l'application multidiffusion ;

La classe E : réservée à un usage ultérieur ;

On peut représenter ces différentes classes par la figure suivante :

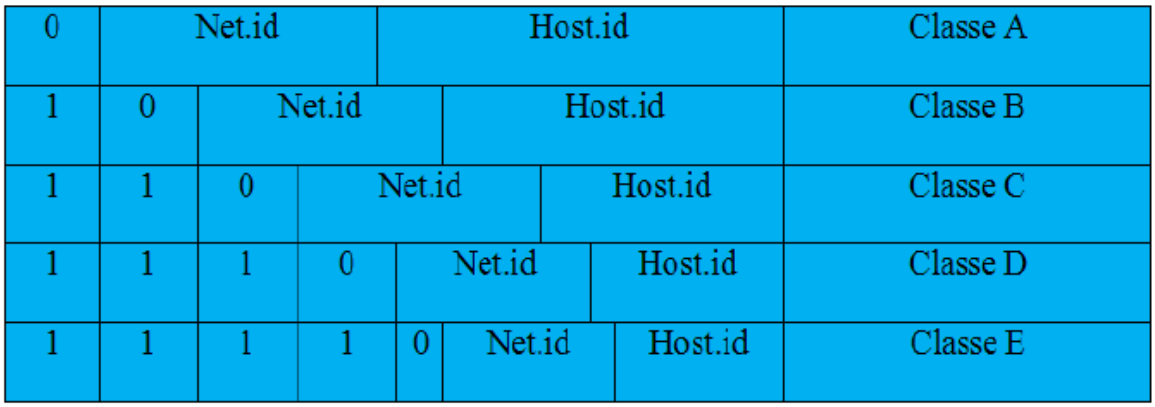

Figure 7 : Classification d'adresse IP

Source : www.rapport-gratuit.com

Pour définir la décomposition de l'adresse IP à 32 bits d'un ordinateur, un second numéro de 32 bits, appelé masque de sous-réseau, est utilisé. Ce masque fournit un guide pour l'interprétation de l'adresse IP. Il indique combien de bits sont réservés à l'identification du réseau dont fait partie l'ordinateur. La partie gauche du masque de sous-réseau est formée d'une série de 1 successifs. Tous les bits du masque qui correspondent à l'adresse du réseau ont la valeur 1, tandis que le reste du masque comporte des zéros. Les bits du masque de sous-réseau qui portent la valeur 0 identifient l'hôte.

Pour la classe A, le masque de sous réseau est : 255.0.0.0

Pour la classe B, le masque de sous réseau est de 255.255.0.0

Pour la classe C, le masque de sous réseau est de 255.255.255.0

2.1.3.2.5.2. Le routage

Le routage consiste à trouver une ligne de sortie pour émettre et recevoir des données. Le but c'est de chercher le meilleur chemin possible pour acheminer les paquets.

On divise le routage en deux grandes familles :

- Le routage direct : Il s'agit de délivrer un datagramme `a une machine raccordée au même LAN. L'émetteur trouve l'adresse physique du correspondant (ARP), encapsule le datagramme dans une trame et l'envoie ;

- Le routage indirect : Le destinataire n'est pas sur le même LAN comme précédemment. Il est absolument nécessaire de franchir une passerelle connue d'avance ou d'employer un chemin par défaut.

Les opérations de routage se font grâce à une table, dite " table de routage ", dans une table de routage on peut trouver les champs suivants :

Destination : le réseau ou l'hôte de destination Gateway : la passerelle à employer pour atteindre la destination simplifiée

Flags : les flags décrivent certaines caractéristiques de cette route. Leurs valeurs possibles sont :

- D : La route a été créé dynamiquement ;
- G : La route désigne une passerelle, sinon c'est une route directe ;
- H : La route est vers une machine, sinon elle est vers un réseau ;
- L : Désigne la conversion vers une adresse physique ;
- S : La route a été ajoutée manuellement ;
- U : La route est active ;
- W : La route est le résultat d'un clonage.

· Numéros de protocole

Le numéro de protocole est un octet unique dans le troisième mot de l'en-tête du datagramme. La valeur identifie le protocole de la couche au-dessus d'IP à laquelle les données doivent être passées.

· Numéro du port

Les numéros de port sont des valeurs de 16 bits qui identifient les processus d'application ou des services réseau. Dans chaque premier mot de l'en-tête de chaque datagramme TCP et chaque datagramme UDP existe le numéro de port source et numéros de port destination. Le numéro de port source identifie le processus ayant envoyé les données. Le numéro destination identifie les recevant.

Les numéros de port ne sont pas uniques entre les protocoles de transport. TCP et UDP peuvent se voir assigner les mêmes numéros de port.

C'est la combinaison des numéros de port et de protocole qui identifie de manière unique le processus spécifique auquel les données doivent être délivrées.

· Socket

La combinaison d'une adresse IP et d'un numéro de port est appelé socket. Un socket identifie un processus réseau de façon unique dans tout l'Internet. Une paire de socket, l'une pour l'hôte récepteur et l'autre pour l'hôte émetteur, définit la connexion pour les protocoles orientés connexion comme TCP.

2.1.4. Architecture Client-Serveur

2.1.4.1. Présentation du système client-serveur

De nombreuses applications fonctionnent selon un environnement client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, …

Toutes les architectures informatiques Client/serveur présentes des caractéristiques communes :

Elles intègrent une interface utilisateur (UI) souvent graphique (GUI)

Elles fonctionnent grâce à des applications

Les applications qui les animent manipulent des données

C'est la répartition de ces 3 composantes entre le client ou le serveur qui caractérise les différentes architectures.

Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes. On parle ainsi, par exemple, de client FTP, client de messagerie, ..., lorsque l'on désigne un programme, tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès du serveur (dans le cas du client FTP il s'agit de fichiers, tandis que pour le client messagerie il s'agit de courrier électronique).

2.1.4.2. Avantages et Inconvénients du système client-serveur

Les avantages de ce système sont :

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

Des ressources centralisées : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.

Une meilleure sécurité : car le nombre de points d'entrée permettant l'accès aux données est moins important ;

Une administration au niveau serveur : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés ;

Un réseau évolutif : grâce à cette architecture on peut supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures ;

Les inconvénients sont :

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

Un coût élevé dû à la technicité du serveur ;

Un maillon faible : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui.

2.1.4.3. Fonctionnement du système client-serveur

Un système client/serveur fonctionne selon le schéma suivant :

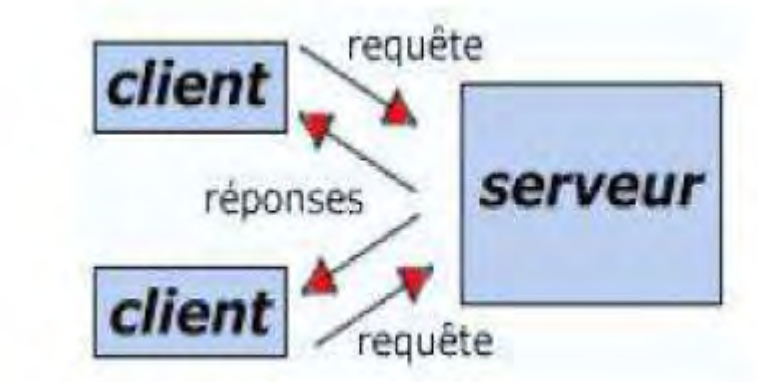

Figure 8 : Fonctionnement du système client-serveur

Source : www.rapport-gratuit.com

- Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur ;
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port.

2.1.4.3.1. Présentation de l'architecture à deux niveaux

L'architecture à deux niveaux (aussi appelée architecture 2-tier, tier signifiant étage en anglais) caractérise les systèmes clients/serveurs dans lesquels le client demande une ressource et le serveur la lui fournit directement. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir le service.

Voici une figure permet de représenter cette architecture à deux niveaux :

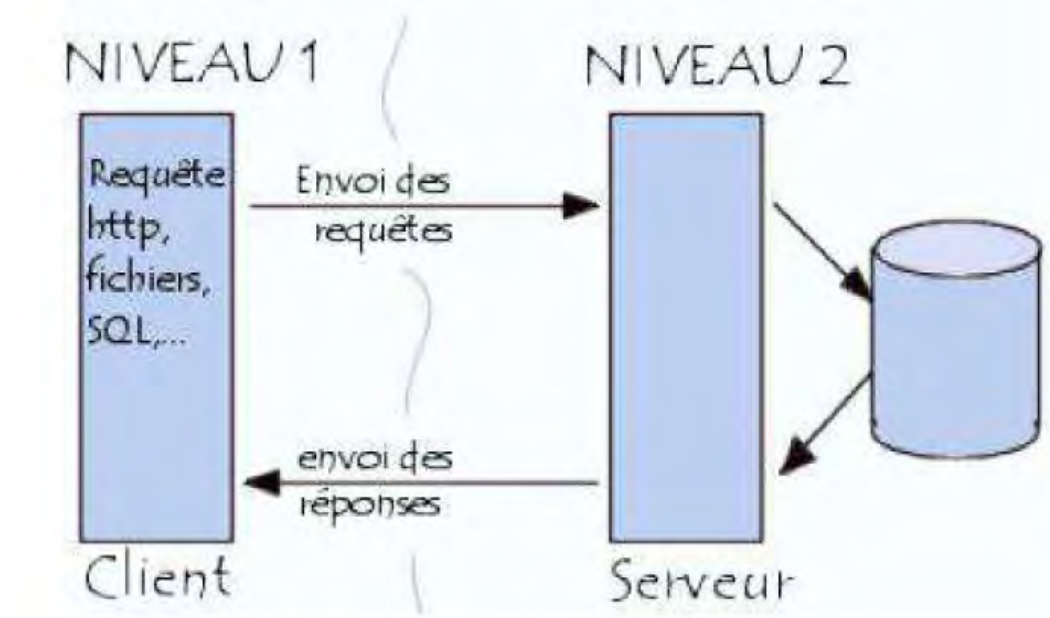

Figure 9 : représentation de l'architecture à deux niveaux

Source : www.rapport-gratuit.com

2.1.4.3.2. Présentation de l'architecture à trois niveaux

Dans l'architecture à 3 niveaux (appelées architecture 3-tiers), il existe un niveau intermédiaire, c'est-à-dire que l'on a généralement une architecture partagée entre :
Le client : le demandeur de ressources
Le serveur d'application (appelé aussi middleware) : le serveur chargé de fournir la ressource mais faisant appel à un autre serveur.
Le serveur secondaire (généralement un serveur de base de données), fournissant un service au premier serveur.
Voici une figure permettant de représenter cette architecture à trois niveaux :

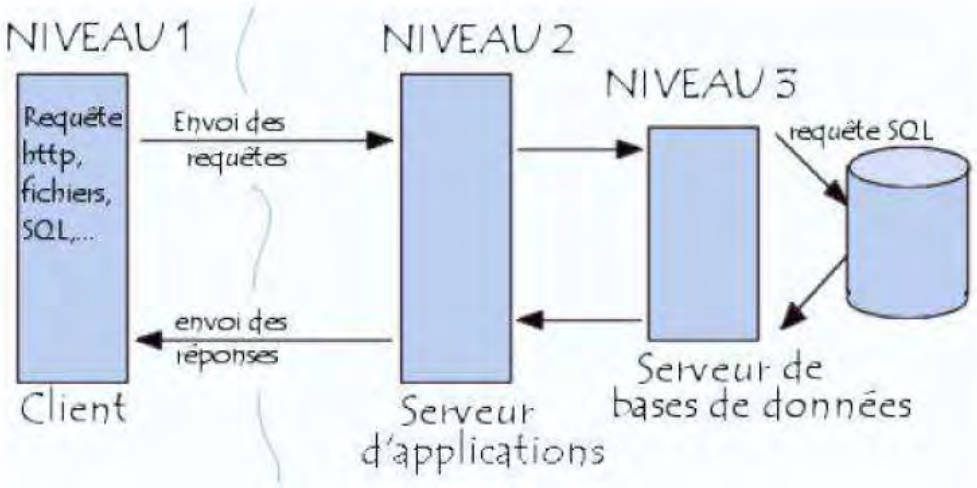

Figure 10 : représentation de l'architecture à trois niveaux

Source : www.rapport-gratuit.com

Etant donné l'emploi massif du terme d'architecture à 3 niveaux, celui-ci peut parfois désigner aussi les architectures suivantes :

Partage d'application entre client, serveur intermédiaire, et serveur d'entreprise.

Partage d'application entre client, base de données intermédiaire, et base de données d'entreprise.

L'architecture à deux niveaux est donc une architecture client/serveur dans laquelle le serveur est polyvalent, c'est-à-dire qu'il est capable de fournir directement l'ensemble des ressources demandées par le client. Dans l'architecture à trois niveaux par contre, les applications au niveau serveur sont délocalisées, c'est-à-dire que chaque serveur est spécialisé dans une tâche (serveur web/serveur de base de données par exemple). Ainsi, l'architecture à trois niveaux permet :

- Une plus grande flexibilité/souplesse ;
- Une plus grande sécurité (la sécurité peut être définie pour chaque service) ;
- De meilleures performances (les tâches sont partagées).

2.1.4.3.3. Présentation de l'architecture à multi niveaux

Dans l'architecture à 3 niveaux, chaque serveur (niveaux 1 et 2) effectue une tâche (un service) spécialisée. Ainsi, un serveur peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service. Par conséquence, l'architecture à trois niveaux est potentiellement une architecture à N niveaux...ou multi-niveaux. Voici une figure permet de représenter cette architecture à multi-niveaux :

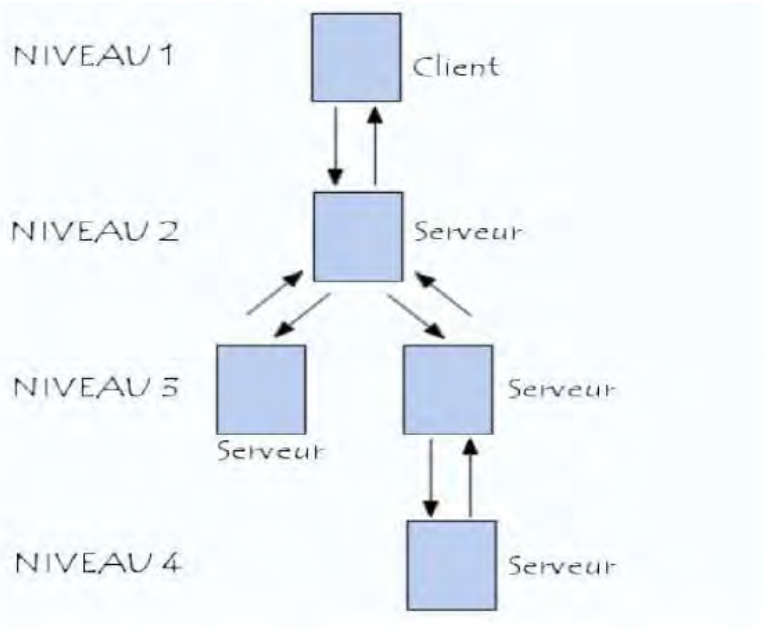

Figure 11 : Architecture à multi niveaux

Source : www.rapport-gratuit.com

2.2. Le système d'exploitation linux

Linux est un système d'exploitation moderne bénéficiant de l'ensemble des fonctionnalités d'Unix. Ce n'est pas un produit commercial : c'est un logiciel libre que l'on peut obtenir gratuitement. Il est livré avec toutes les fonctionnalités, les outils et les utilitaires habituellement livrés avec les variantes commerciales d'Unix :

- C'est un système 32 bits ;
- Il est multiutilisateurs ;
- Il est multitâche ;
- Dans le domaine des réseaux, il prend parfaitement en charge la famille des protocoles TCP/IP et possède bien plus de caractéristiques que la plupart des variantes commerciales d'Unix ;
- …

2.2.1. Pourquoi utiliser linux comme serveur Intranet ?

2.2.1.1. Le système d'exploitation linux

Linux est le système qui connaît actuellement le plus grand développement sur l'Internet ou Intranet. Principalement pour les raisons suivantes :

- Linux est le système de prédilection pour l'installation de trois logiciels serveurs leaders sur l'Internet : Apache en serveur Web, Postfix ou sendmail en serveur courrier et Bind en serveur DNS ;
- Le logiciel Samba qui lui permet d'être serveur de fichier et d'impression en environnement Microsoft ;
- La stabilité et la sécurité que lui confère le développement de son architecture et de ses modules au sein de la communauté Open Source.
- Le large choix d'applications dans de très nombreux domaines. Par exemple, la dernière distribution Debian donne accès à plus de 2000 logiciels différents.
- Moins d'interruptions de service grâce à une gestion intelligente de l'installation des logiciels.

Un serveur sous Linux ne doit être redémarré que lors d'une modification matérielle comme l'ajout d'un disque ou d'une carte.

- Logiciel Libre. Linux est gratuit et librement re-copiable. Cela signifie que l'on peut télécharger une version de Linux ou l'emprunter et l'installer sur n'importe quel nombre d'ordinateur.
- Accès aux sources des logiciels. Tous les utilisateurs peuvent modifier le fonctionnement des programmes ou engager un programmeur pour le faire.
- Linux est plus efficace et consomme moins de ressources CPU et mémoire que Windows. On peut par exemple faire un serveur d'impression avec un vieux 486.

On peut résumer que Linux comme serveur permet d'avoir une réduction des couts, une sécurité et performance.

2.2.1.2. Linux et le serveur apache

Le serveur Web Apache propose une qualité de service que peu d'offres commerciales peuvent concurrencer.

Apache tourne sur Unix, que ce soit Linux ou un UNIX BSD, ainsi que sur WindowsNT, W2K, et WXP. Une nette majorité des serveurs web tournent sous Unix, pour des raisons de performance et surtout de fiabilité. Le serveur Web Apache peut être utilisé comme simple serveur web, ou bien comme serveur d'application et interface de base de données avec les logiciels PHP et MySQL. De plus utiliser des logiciels libres, par opposition à des logiciels payants, est d'une part nettement moins chère, et un moyen de préserver l'indépendance technologique des pays.

Linux en tant que serveur Intranet / Internet peut devenir l'ensemble des solutions suivantes et il est bien entendu possible qu'un seul et même ordinateur gère toutes ces possibilités :

- Un serveur WEB classique (HTTP) ;
- Un serveur FTP ;
- Un serveur de mail (SMTP, POP) ;
- Un serveur Proxy ;
- Un Firewall ;
- Un serveur DNS ;
- Un routeur, etc.…

Linux peut gérer un réseau d'entreprise, comme :

- Un serveur de fichiers ;
- Un serveur d'impression ;
- Un serveur de fax ;
- Un serveur de connexion Dial−Up (permet de devenir fournisseur d'accès à Internet)
- Un serveur de partage de connexion ;
- Un serveur de sauvegarde, etc.

Pour transformer, par exemple, un serveur Linux en serveur de base de données, il suffit de coupler le logiciel de base de données (comme MySQL) avec le serveur Web Apache via un langage comme PHP. Un simple navigateur Web suffit alors pour accéder à l'application voulue, ce qui permet d'alimenter et de consulter très facilement des bases de données.

2.2. La distribution CentOS

CentOS ou Community enterprise Operating System est une distribution issue du monde Red Hat Entreprise Linux ou RHEL qui est l'un des systèmes d'exploitation les plus prisées actuellement parce qu'il est haut de gamme est performant. Red Hat Linux est très populaire grâce à ses nombreuses distributions indépendantes adaptées aux ordinateurs personnels et serveurs informatiques. La version CentOS, quant à elle, est très stable et elle est principalement dédiée aux serveurs.

Aujourd'hui, CentOS est l'une des distributions linux les plus connues pour les serveurs web.

Ce n'est pas le meilleur système au monde, comme les autres elle a ses qualités et ses défauts. Ce qui nous intéresse c'est de savoir quels avantages pourrait-on tirer de son utilisation :

- Tout d'abord, ce système est très stable, sur ce point, il est comparable à la distribution Red Hat utilisé dans les environnements de production de grande taille ;
- Il est doté d'un support gratuit par l'intermédiaire de la communauté et de patchs de sécurité réguliers ;
- Les mises à jour de ce système sont applicatives ;
- L'exploitation et la gestion des paquets au format RPM est plus facile grâce à l'outil YUM ;
- Les manuels en ligne de Red Hat compatibles avec CentOS sont disponibles en anglais et en français.

Chapitre 3 : Généralités sur l'intranet

3.1. Qu'est-ce que l'intranet ?

C'est un réseau informatique au sein d'une même entreprise ou organisation, il se compose d'un ensemble de pages accessibles en réseau et un ensemble d'applications hébergées sur des serveurs.

Les ressources au sein de ce réseau ne sont pas consultables à partir d'autres postes de travail en dehors de l'entreprise. C'est donc l'équivalent d'un Internet propre à une entreprise. Il est utilisé par l'ensemble des travailleurs ou des collaborateurs de cette entreprise, il permet par exemple de :

- Déployer des fonctionnalités et des applications internes à l'entreprise ;
- Discuter à travers un chat ou un forum entre collaborateurs dispersés sur plusieurs sites ;
- Partager les agendas de l'ensemble des collaborateurs ;
- Consulter ses mails ;
- Échanger des documents et des fichiers entre collaborateurs ;
- Rechercher des documents à travers un moteur de recherche interne ;
- Gérer des projets ;
- Fournir un outil d'aide à la décision ;
- Mettre à disposition un annuaire du personnel ;
- Réunir plusieurs collaborateurs pour une visioconférence, …

Cette liste non exhaustive de fonctionnalités donne un aperçu des possibilités d'un intranet. Un intranet peut-être plus ou moins présent dans les process d'une entreprise : les collaborateurs peuvent, depuis leurs postes de travail respectifs, accéder à des documents internes tels que des comptes-rendus de réunion, des grilles de rémunérations, des demandes de congé… C'est donc une solution pour accéder à l'intégralité du système informatique d'une entreprise au format universel du Web.

Un intranet est hébergé sur un ou plusieurs serveurs et peut être consulté depuis n'importe quel endroit dans le monde.

3.2. Quels objectifs pour l'intranet ?

Un intranet présente de nombreux avantages parmi lesquels on peut citer :

- Une centralisation des données ;
- Une fluidité d'accès à l'information ;
- Une mise à disposition des indicateurs clés & reporting ;
- Une optimisation de gestion des process interne.

3.3. Pourquoi adopter un intranet pour une entreprise ?

La mise en place d'un intranet dans une entreprise permet dans la grande majorité des cas des gains de productivité. Il s'avère également très précieux pour favoriser la collaboration de salariés dispersés sur plusieurs sites.

On peut aussi citer quelques raisons liées à l'adoption d'un intranet :

- Un déploiement et une administration centralisée des courriers électroniques ;
- La rapidité de la consultation et de la mise à jour de documents au format HTML ;
- Une mise à jour centralisée de l'ensemble des documents de l'entreprise ;
- La grande facilité de publication, de diffusion et de consultation de documents ;
- La transparence dans l'organisation du travail et la gestion de projets avec une visibilité sur l'état d'avancement ;
- Une meilleure gestion des ressources de l'entreprise ;
- Une bonne visibilité sur les disponibilités de chacun, …

Chapitre 4 : Généralités sur Ansible

4.1. Qu'est-ce que l'automatisation ?

L'automatisation informatique, aussi appelée automatisation de l'infrastructure, consiste à utiliser des logiciels pour créer des instructions et des processus reproductibles dans le but de remplacer ou de réduire l'interaction humaine avec les systèmes informatiques. Les logiciels d'automatisation s'exécutent dans les limites de ces instructions, outils ou structures afin de réaliser des tâches avec une intervention humaine minimale, voire nulle.

L'automatisation est un élément clé de l'optimisation de l'environnement informatique et de la transformation numérique. Les environnements informatiques dynamiques et modernes doivent pouvoir évoluer plus rapidement que jamais, et l'automatisation informatique joue là un rôle essentiel.

4.2. Que couvre l'automatisation informatique ?

En théorie, toute tâche informatique peut être automatisée dans une certaine mesure. L'automatisation peut donc être intégrée et s'appliquer à toute tâche, de l'automatisation du réseau à la gestion des configurations et au déploiement d'applications en passant par l'approvisionnement du cloud, de l'infrastructure et des environnements d'exploitation standard.

Les fonctionnalités d'automatisation et les applications peuvent elles-mêmes s'étendre à des technologies spécifiques telles que les conteneurs, des méthodes comme le DevOps et des domaines plus vastes comme le cloud, l'edge computing, la sécurité, les tests et la surveillance ou les alertes.

4.3. L'automatisation : les avantages

Une approche globale de l'automatisation informatique peut délester l'équipe de l'entreprise de processus manuels et répétitifs. Elle permet ainsi d'augmenter la productivité des équipes, de réduire le nombre d'erreurs, d'améliorer la collaboration et de consacrer plus de temps à des tâches plus importantes.

4.4. Les taches de l'automatisation

4.4.1. Approvisionnement

L'approvisionnement figure parmi les tâches les plus lourdes, qu'il s'agisse d'approvisionner un système nu ou un cloud privé, hybride ou public. Le fonctionnement des systèmes métier nécessite la mise en place et la configuration d'une infrastructure adaptée. De nos jours, les racks, boîtiers et câbles des datacenters ont presque complètement cédé la place aux ressources virtualisées : réseaux, stockage et datacenters définis par logiciel, machines virtuelles et conteneurs.

La plupart des tâches réalisées aujourd'hui sont définies dans des logiciels, et cette transition vers les logiciels a permis d'accroître considérablement le potentiel d'évolutivité et de capacité. Cette transition permet et, nécessairement, exige aussi la codification de processus et vous aide ainsi à satisfaire les besoins de votre entreprise tout en maîtrisant les coûts et en respectant les délais.

C'est justement à ce niveau qu'intervient l'automatisation. Pourquoi perdre du temps à configurer manuellement ces environnements avec des modèles ? Avec la codification, vous avez à votre disposition un modèle à suivre pour réaliser des tâches. Alors pourquoi ne pas transmettre ces règles à un système automatisé pour qu'il les exécute ? Automatisez les déploiements dans votre datacenter à l'aide de solutions compatibles avec votre infrastructure et vos outils de gestion existants afin de tirer pleinement parti des ressources dont vous disposez et d'atteindre vos objectifs futurs.

4.4.2. Gestion des configurations

Toutes les applications ne sont pas créées de la même manière. Elles peuvent avoir besoin d'éléments qui diffèrent : paramètres, systèmes de fichiers, ports, utilisateurs, etc. Une fois l'approvisionnement automatisé, on doit être en mesure d'indiquer à ces ressources ce qu'elles doivent faire. L'enregistrement de la définition de notre environnement d'applications dans un document, une feuille de calcul, un fichier texte ou même un e-mail ne permettra pas de mettre en place un environnement solide et reproductible pour héberger les applications. Face à la multiplication des systèmes et des instances et à l'augmentation de la complexité, on doit adopter un moyen plus approprié d'enregistrer les informations relatives à vos systèmes pour les gérer plus efficacement.

On a besoin pour cela d'une solution robuste de gestion des configurations, qui permet aux développeurs de définir l'infrastructure (système nu, ressources virtualisées, cloud,

conteneurs, etc.) de manière simple et compréhensible pour tous les membres de votre équipe informatique. Plus l'automatisation de pratiques et de scripts adaptés à la gestion des systèmes sera simple, plus le travail sera facilité.

4.4.3. Orchestration

Il y a de fortes chances que l'on ne déploie pas qu'un service sur une seule machine. L'environnement informatique est probablement bien plus complexe. Il nécessite sans doute de gérer et d'assurer la maintenance d'une multitude d'applications dans plusieurs datacenters et infrastructures. Sans oublier les déploiements de clouds privés, publics et hybrides.

Plus l'environnement informatique est complexe, plus la gestion de tous ses éléments mobiles l'est, elle aussi, et plus on a besoin de combiner des tâches automatisées et leurs configurations dans différents groupes de systèmes et de machines. C'est le principe même de l'orchestration. Cerise sur le gâteau, on a la possibilité de contrôler ces orchestrations avec des solutions d'automatisation robustes. Ceci nous permet d'en effectuer le suivi, de les relier les unes aux autres et d'exécuter des systèmes autonomes et plus avancés, en toute simplicité.

4.4.4. Déploiement d'application

Qu'on utilise une méthode traditionnelle de déploiement d'applications ou qu'on suive une approche de CI/CD (intégration et déploiement continus), les pipelines de développement doivent s'appuyer sur des systèmes automatisés robustes pour satisfaire les nouvelles exigences. Le succès du déploiement d'applications dépend donc de l'efficacité d'un ensemble de fonctionnalités et de tâches essentielles automatisées, en particulier lors de la phase de tests. L'automatisation peut nous aider à passer des phases de validation et de création à celles de tests et de déploiement d'une manière codifiée, fiable et éprouvée. Cette démarche réduit les risques d'erreur humaine tout en améliorant l'efficacité et la rapidité.

L'automatisation informatique nous permet de déployer des applications en toute confiance, de configurer dès le début les services requis et de lancer des applications et leurs artéfacts selon une approche commune, à la fois transparente et compréhensible par tous les membres de l'équipe informatique.

4.4.5. Sécurité et conformité

Définir des politiques de sécurité, de conformité et de gestion des risques, les mettre en application et corriger les problèmes en intégrant ces politiques dans des étapes automatisées à l'échelle de l'infrastructure. Placer la sécurité au cœur de vos processus informatiques et soyez plus proactif grâce à l'automatisation.

La standardisation des workflows et des processus de sécurité facilite la mise en conformité et la réalisation d'audits. On sait exactement comment les politiques sont appliquées et on peut en vérifier la cohérence. On peut également mettre facilement en œuvre de nouvelles exigences de conformité dans l'ensemble de l'environnement informatique.

4.5. Les outils d'automatisation

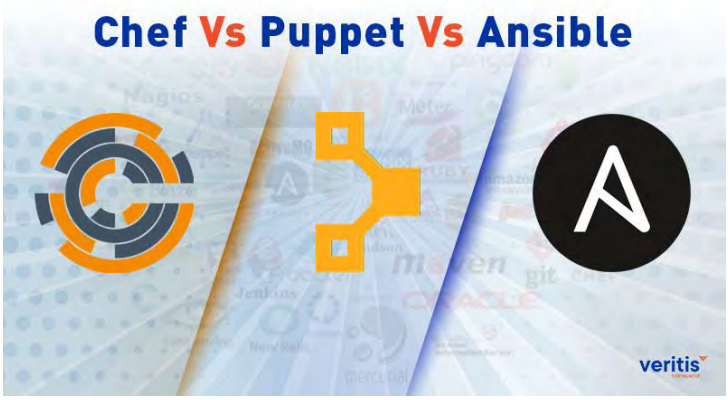

Figure 12 : Outils d'automatisation

Source : https://www.veritis.com/

Il existe sur le marché beaucoup d'outils d'automatisation parmi lesquels on a :

- Ansible

Ansible est un logiciel libre et idempotent d'automatisation, de déploiement et de gestion des configurations sous licence GNU/GPL, écrit en Python et est multiplateforme (GNU/Linux, Microsoft Windows).

Ce logiciel est maintenu par son créateur Michael DeHaan et sa communauté d'utilisateur qui comprend, entre autres Red Hat et Hewlett-Packard ;

Il existe une version graphique et payante d'Ansible : Ansible Tower. Cette version est soutenue par RedHat et offre des fonctions de contrôle d'accès basé sur les rôles, de planification des tâches et de gestion graphique des inventaires.

La plus petite unité de valeurs d'Ansible est une tâche décrite en syntaxe YAML, un ensemble de tâches constitue un play et un ensemble de play constitue un playbook. Il est aussi possible d'utiliser la syntaxe Jinja2 dans des templates ou filtres. Dans les tâches ou les templates, il est possible d'utiliser des variables que l'on peut créer soi-même ou utiliser les « facts ».

Ces facts sont un ensemble de variables décrivant les spécificités d'une machine.

Ansible dispose aussi d'un certain nombre de modules pouvant être exécutés directement sur les machines distantes ou par les Playbook. Ces modules sont assez variés et permettent plusieurs interactions.

Les principaux points positifs d'Ansible sont :

- La clarté de la documentation officielle et sa facilité d'installation (Ansible ne nécessite pas d'installation d'agent supplémentaire sur les machines distantes.) ;
- La communication entre les machines se fait par le biais de tunnel SSH. Et le seul prérequis pour permettre cette communication est le déploiement de la clef publique de l'utilisateur Ansible sur les machines distantes et la présence de python 2.7 et sudo sur les machines distantes ;
- Le fait qu'Ansible soit écrit dans un langage assez connu. Cela permet si le besoin se fait sentir de développer plus rapidement et efficacement nos propres modules ;
- Cela est moins contraignant que de déployer un agent supplémentaire sur des machines.

➢ Puppet

Puppet est un logiciel, sous licence Apache 2, permettant d'automatiser un grand nombre de tâches d'administration, comme l'installation de logiciels, de services ou encore de modifier des fichiers. Puppet fonctionne sous le mode client /serveur, il a été écrit en Ruby ce qui lui permet d'être multi-plate-forme (BSD, SUN, Linux, Windows). Les échanges entre le maître et les esclaves se font par le biais de communications HTTPS. La société éditant Puppet, Reductive Labs, dispose d'une version commerciale permettant en plus des fonctionnalités de la version open-source :

- d'avoir un support sur Puppet et les modules maintenus par Reductive Labs ;
- d'avoir une interface graphique de gestion ;
- de gérer les machines virtuelles Vmware ;
- …

En plus de Puppet, Reductive Labs a développé le logiciel nommé Facter. Celui-ci permet de lister les éléments propres aux systèmes administrés, comme le nom de la machine, les adresses MAC et IP, le système d'exploitation ainsi que les variables d'environnement utilisables dans les « templates » de Puppet.

Ces éléments peuvent ensuite être utilisés dans les « templates » Puppet sous la forme de variable. Il est très souvent utilisé par les concurrents pour la même fonction. Comme bon nombre de ses concurrents, Puppet dispose d'une « forge » permettant de partager, rechercher et récupérer des modules.

➢ Chef

Tout comme Puppet, Chef est aussi un outil de gestion de configuration multi-plate-forme, écrit en Ruby et sous licence Apache 2. Chef dispose d'un serveur central qui détient les configurations des machines clientes. Comme Puppet, il est nécessaire d'installer un agent sur chaque machine que l'on veut configurer avec ce logiciel.

Après l'installation du client, les machines esclaves vont comparer leur configuration par rapport à celles référencées sur le serveur maître et effectuent si besoin les changements nécessaires pour coller à la configuration décrite dans le serveur maître. Comme le nom du logiciel, l'appellation des outils est en rapport avec la cuisine. Ainsi, l'installation d'un programme est décrite par une recette (« recipes » en anglais).

On trouve par exemple une recette pour « MySQL », une autre pour « Apache » etc. L'ensemble des recettes, pour configurer un logiciel, est contenu dans un livre de recette (« cookbook » en anglais). « Knife », traduction du mot français « couteau » en anglais, est l'outil pour interagir avec Chef. Nous aurons aussi les « rôles » qui désignent l'ensemble des livres de recettes ayant la même finalité.

Par exemple, le rôle « serveur web » décrira les recettes pour installer le serveur web mais aussi la configuration du pare-feu de la machine.

Les ingénieurs PROJIXI Europe intègrent des solutions propriétaires et open source chez des grands comptes, grâce à leurs expériences sur des projets significatifs.

4.6. Qu'est-ce que Ansible ?

Dans l'environnement informatique actuel, les applications d'entreprise peuvent être complexes, évolutives, distribuées, basées sur des composants et sont souvent critiques. Ils peuvent être déployés sur une variété de plates-formes dans le cloud privé, le cloud public ou le cloud hybride. Ils peuvent accéder à des données sensibles, ils peuvent être soumis à des directives réglementaires et à des politiques de sécurité strictes, et doivent cependant être aussi conviviaux que possible. En bref, ces applications sont très complexes. Il est necessaire de voir la façon dont ces considérations s'harmonisent avec l'utilisation d'outils d'automatisation comme Ansible.

Ansible est un logiciel libre de gestion des configurations qui automatise le déploiement des applications et la livraison continue des mises à jour. Disponible sous licence GPL v3, il s'adosse au protocole de cryptage réseau SSH (pour Secure Socket Shell) pour déployer les mises en production de code via des fichiers décrivant les configurations applicatives cibles au format Json (pour JavaScript Object Notation).

On peut utiliser Ansible pour automatiser trois types de tâches :

- Provisioning : configurer les différents serveurs dont on a besoin dans une infrastructure ;
- Configuration : modifier la configuration d'une application, d'un système d'exploitation ou d'un périphérique, démarrer et arrêter les services, installer ou mettre à jour des applications, mettre en œuvre une politique de sécurité, ou effectuer une grande variété d'autres tâches de configuration ;
- Déploiement d'applications : adopter une démarche DevOps en automatisant le déploiement d'applications développées en interne sur nos environnements de production.

4.7. Quelques notions d'Ansible

➢ Module

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 le vôtre, personnalisé.

- 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.

  Le but des rôles est de pouvoir agglomérer des opérations cohérentes (dans les fichiers YAML), afin de pouvoir les réutiliser de façon modulaire. On peut voir un rôle comme un ensemble d'opérations qui ont un rôle commun, comme par exemple le rôle d'installer Apache, ou le rôle de configurer MariaDB.

  Les répertoires sont tous optionnels, excepté le répertoire **tasks** qui doit contenir le fichier **main.yml**. Ansible va traiter en premier ce fichier à l'appel d'un rôle.

- Tache

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

- Playbook

  Un playbook est un fichier de configuration YAML contenant une suite de jeux d'instructions, ou plays en anglais. Chacun peut être constitué d'options, et fait appel à un ou plusieurs rôles. Il permet de décrire une stratégie de déploiement, ou de configuration, en structurant les actions nécessaires.

  En utilisant les playbooks, on a la possibilité de conserver le code dans un fichier et de le réutiliser à votre façon, contrairement à la commande ansible qui est volatile

De façon schématique, on peut retenir que :

- Un rôle contient un ou plusieurs fichiers de configuration (YAML) ;
- Un fichier de configuration contient une ou plusieurs tâches ;
- Une tâche fait appel à un module.

4.8. Fonctionnement et avantages d'Ansible

Dans Ansible, il existe deux catégories d'ordinateurs : le nœud maître (master) et les nœuds esclaves (slaves). Le nœud maître est une machine sur laquelle est installé l'outil Ansible. Il doit y avoir au moins un nœud maître, bien qu'un nœud maître de sauvegarde puisse également exister.

Ansible fonctionne en se connectant aux nœuds en SSH et en y poussant de petits programmes, appelés modules. Ces modules sont définis dans un fichier nommé le Playbook. Le nœud maître, se base sur un fichier d'inventaire qui fournit la liste des hôtes sur lesquels les modules Ansible doivent être exécutés.

Ansible exécute ces modules en SSH et les supprime une fois terminer. La seule condition requise pour cette interaction est que le nœud maître Ansible dispose d'un accès de connexion aux nœuds esclaves. Les clés SSH sont le moyen le plus courant de fournir un accès, mais d'autres formes d'authentification sont également prises en charge.

Voici un schéma qui reprend notre explication :

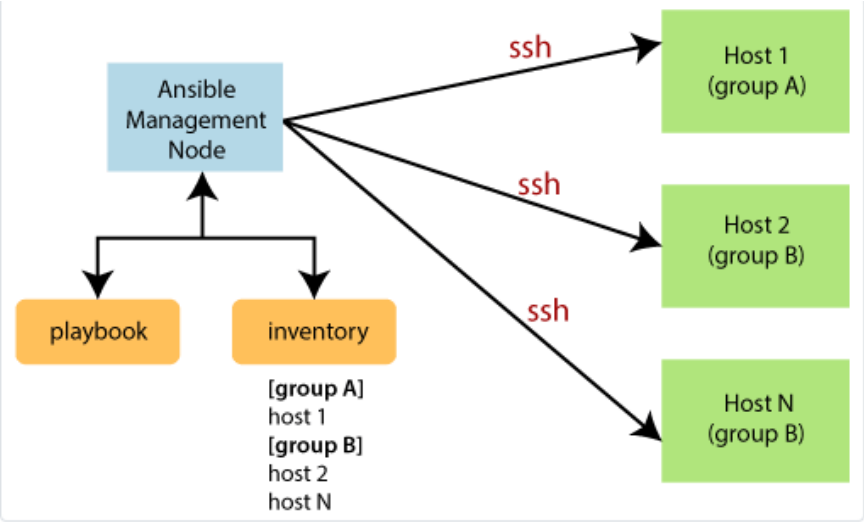

Figure 13 : Fonctionnement d'Ansible

Source : https://devopssec.fr/

Voici une liste des avantages d'Ansible :

- Gratuit : Ansible est un outil open source ;
- Simple : Ansible utilise une syntaxe simple écrite en YAML. Aucune compétence en programmation particulière n'est nécessaire pour créer les playbooks d'Ansible. Il est également simple à installer ;
- Puissant : Ansible vous permet de modéliser des workflows très complexes ;
- Flexible : Ansible fournit des centaines de modules prêts à l'emploi pour gérer nos tâches, quel que soit l'endroit où ils sont déployés. On peut réutiliser le même playbook sur un parc de machines Red Hat, Ubuntu ou autres ;
- Agentless : on n'a pas besoin d'installer d'autres logiciels ou d'ouvrir des ports de pare-feu supplémentaires sur les systèmes clients que vous souhaitez automatiser. Ansible réduit encore l'effort requis pour que notre équipe commence à automatiser immédiatement ;
- Efficace : Parce qu'on n'a pas besoin d'installer de logiciel supplémentaire, il y a plus de place pour les ressources d'application sur notre serveur.

4.9. Que peut faire Ansible ?

Ansible peut être utilisé de différentes manières. En voici quelques-unes ci-dessous :

4.9.1. Déploiement d'application

Ansible permet de déployer rapidement et facilement des applications à plusieurs niveaux. On n'a pas besoin d'écrire du code personnalisé pour automatiser les systèmes ; on liste les tâches à effectuer en écrivant un playbook, et Ansible trouvera comment amener les systèmes à l'état dans lequel on veut qu'ils soient. En d'autres termes, on n'aura pas à configurer manuellement les applications sur chaque machine. Lorsque on exécute un playbook à partir de votre machine de contrôle, Ansible utilisera le protocole SSH pour communiquer avec les hôtes distants et exécuter toutes les tâches (Tasks).

4.9.2. Orchestration

Comme son nom l'indique, l'orchestration consiste à amener différents éléments à interagir ensemble sans incohérence. Par exemple, avec le déploiement d'applications, on doit gérer non seulement les services frontend, mais également les services backend comme les bases de données, le réseau, le stockage, etc. on doit également nous assurer que toutes les tâches sont gérées dans le bon ordre. Grâce à Ansible on orchestre les éléments de notre infrastructure à

l'aide des playbooks Ansible, et on peut les réutiliser sur différents types de machines, grâce à la portabilité des modules Ansible.

4.9.3. Sécurité et conformité

Comme pour le déploiement d'applications, des politiques de sécurité de l'entreprise (telles que des règles de pare-feu ou le verrouillage des utilisateurs) peuvent être mises en œuvre avec d'autres processus automatisés. Si on configure les détails de sécurité sur la machine de contrôle et exécute le playbook associé, tous les hôtes distants seront automatiquement mis à jour avec ces détails. Cela signifie qu'on n'aura pas besoin de surveiller chaque machine pour vérifier la conformité de la sécurité en continu manuellement. De plus, tous les identifiants (identifiants et mots de passe des utilisateurs admin) qui sont stockés dans les playbooks ne sont récupérables en brut par aucun utilisateur.

4.9.4. Provisionnement du cloud

La première étape de l'automatisation du cycle de vie de vos applications consiste à automatiser l'approvisionnement de votre infrastructure. Avec Ansible, on peut provisionner des plateformes cloud, des hôtes virtualisés, des périphériques réseau et des serveurs physiques.

Chapitre 5 : Déploiement

5.1. Prérequis

Dans le tout le long de notre travail on va utiliser Apache comme serveur web, MaraiaDB comme base de données et MediaWiki pour l'intranet.

- **Apache**

Le projet Apache HTTP Server est un effort pour développer et maintenir un serveur HTTP open-source pour les systèmes d'exploitation modernes, y compris UNIX et Windows. Le but de ce projet est de fournir un serveur sécurisé, efficace et extensible qui fournit des services HTTP synchronisés avec les standards HTTP actuels.

Le serveur HTTP Apache (« httpd ») a été lancé en 1995 et est le serveur Web le plus populaire sur Internet depuis avril 1996.

Il existe d'autres serveurs web comme Nginx mais notre choix s'est porté sur Apache vu que par défaut CentOS prpopose Apache Httpd comme serveur web.

- **MariaDB**

MariaDB Server est l'une des bases de données relationnelles open source les plus populaires. Il est fabriqué par les développeurs originaux de MySQL et garanti pour rester open source. Il fait partie de la plupart des offres cloud et est la valeur par défaut de la plupart des distributions Linux.

Il existe d'autres bases de données comme PostgreSQL mais nous avons choisi d'utiliser MariaDB, car c'est l'une des bases de données les plus couramment utilisées conjointement avec MediaWiki. Dans le cadre de ce projet, nous allons utiliser MariaDB dont la syntaxe et l'utilisation sont plus simples et plus explicites (ce qui ne retire en rien ses qualités à PostgreSQL).

- **MediaWiki**

Le logiciel MediaWiki est utilisé par des dizaines de milliers de sites Web et des milliers d'entreprises et d'organisations. Il alimente Wikipedia. MediaWiki nous aide à collecter et à organiser les connaissances et à les mettre à la disposition des gens. Il est puissant, multilingue, gratuit et ouvert, extensible, personnalisable, fiable et gratuit.

Pour commencer on présente notre architecture qui comprend trois (3) serveurs avec CentOS comme système d'exploitation :

- Deux serveurs (Apache :192.168.17.6/24 et MySQL :192.168.17.7/24) dans le jargon Ansible, ces serveurs sont appelés ***nodes*** ;
- Un serveur de contrôle (192.168.17.5/24), appelé ***node manager***. C'est le serveur sur lequel seront installés les outils Ansible et depuis lequel, les opérations de configuration seront lancées à distance sur les nodes.

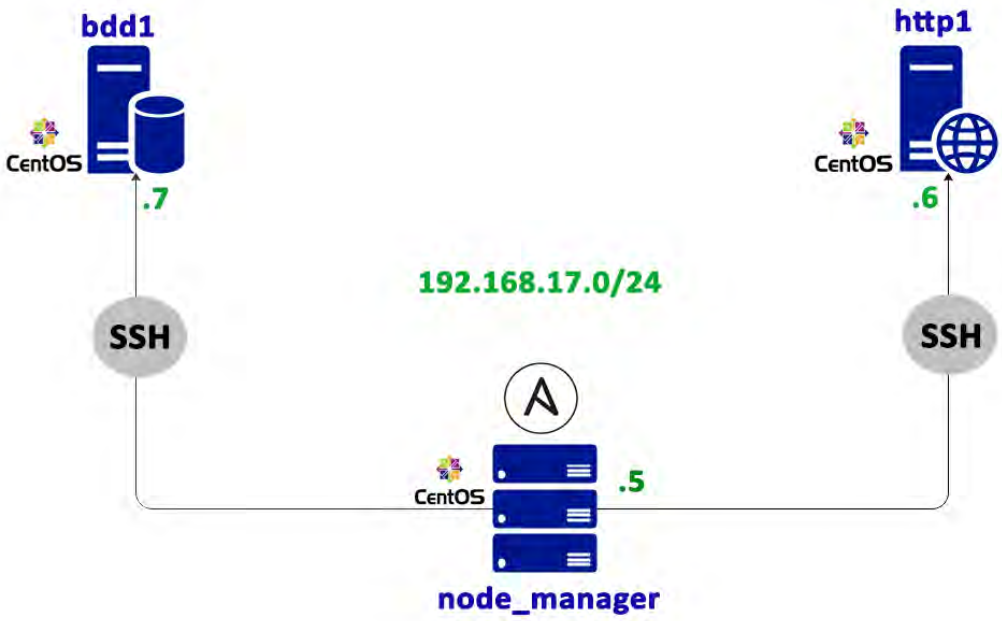

Figure 14 : Architecture de production

➢ Node

Un node (ou managed node, ou host) est un poste connecté au node manager en SSH, et sur lequel Ansible viendra pousser les tâches d'automatisation. Ansible n'est pas installé sur les nodes.

➢ Node manager

Un node manager, ou control node, est un poste qui contrôle les nodes grâce à sa connexion SSH. Il a Ansible d'installé pour leur pousser les tâches d'automatisation grâce aux commandes *ansible* et *ansible-playbook*

✓ **Création d'utilisateur simple**

Pour ne pas travailler en root sur le node manager (ce n'est vraiment pas recommandé, le compte root peut tout faire sans aucune limite) on va donc créer un simple utilisateur *user-ansible* :

```
[root@localhost ~]# adduser user-ansible
[root@localhost ~]# su - user-ansible
```

```
[root@192 ~]# passwd user-ansible
Changing password for user user-ansible.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Figure 15 : Création utilisateur

Pour faciliter le travail on crée le même utilisateur (*user-ansible*) sur les nodes.

Sur le node manager on va donner les droits sudo à *user-ansible*

```
[user-ansible@localhost ~]$ sudo usermod -aG wheel user-ansible

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for user-ansible:
```

Figure 16 : Attribution des droits sudo à user-ansible

✓ **Mise en place des dépôts EPEL**

Qu'est-ce que les dépôts EPEL ? Il s'agit d'un acronyme pour « Extra Packages for Enterprise Linux ». C'est un groupe de volontaires issu de la communauté autour de la distribution Linux Fedora, qui crée, maintient et gère un ensemble de paquets logiciels supplémentaires à destination de plusieurs distributions incluant, entre autres, Red Hat Enterprise Linux (RHEL), CentOS et Scientific Linux.

Pour installer un tel dépôt, il suffit d'exécuter la commande suivante :

```
[user-ansible@localhost ~]$ sudo yum -y install epel-release.noarch
Last metadata expiration check: 0:04:51 ago on Sun 20 Dec 2020 11:04:11 AM PST.
Dependencies resolved.
================================================================================
 Package              Architecture    Version           Repository       Size
================================================================================
Installing:
 epel-release         noarch          8-8.el8           extras           23 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 23 k
Installed size: 32 k
Downloading Packages:
epel-release-8-8.el8.noarch.rpm                 117 kB/s |  23 kB     00:00
--------------------------------------------------------------------------------
Total                                            13 kB/s |  23 kB     00:01
Running transaction check
Transaction check succeeded.
```

Figure 17 : Installation des dépôts EPEL

- ✓ **Installation d'Ansible**

Le travail préliminaire ayant été réalisé, nous avons maintenant un serveur proprement configuré et prêt à être utilisé pour mettre en place notre intranet. Nous allons passer à la toute dernière étape préliminaire : l'installation d'Ansible.

Une fois celle-ci effectuée, nous utiliserons l'outil pour le reste de nos opérations d'installation et de configuration.

Quelle que soit la source des paquets logiciels associés à Ansible (EPEL, ou un autre dépôt accessible), l'installation de l'outil sur un système CentOS se limite à un appel au gestionnaire de paquets *yum* :

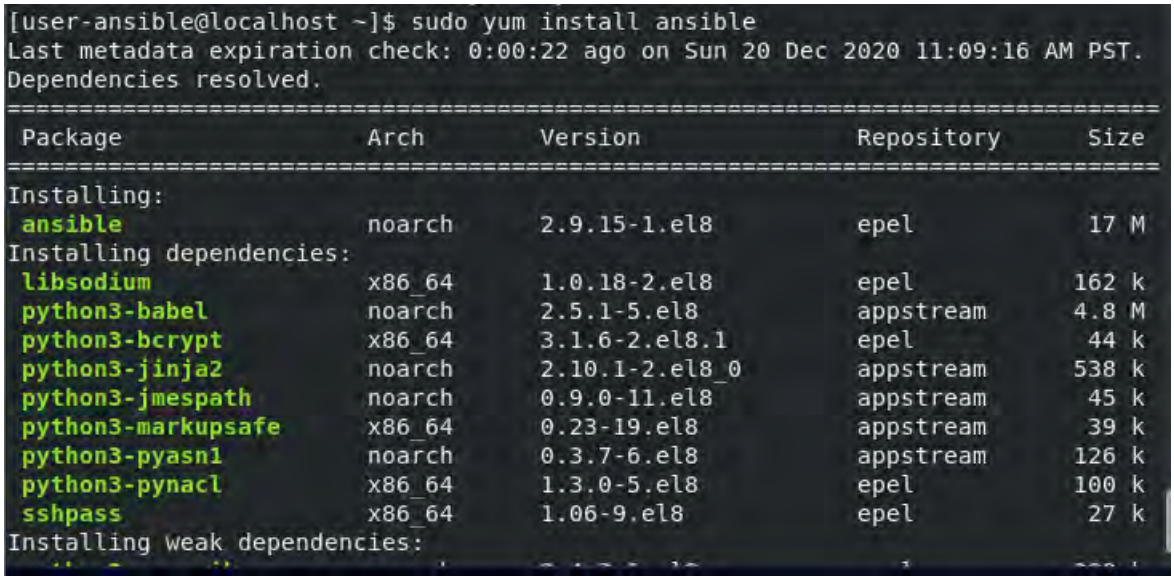

Figure 18 : Installation de Ansible

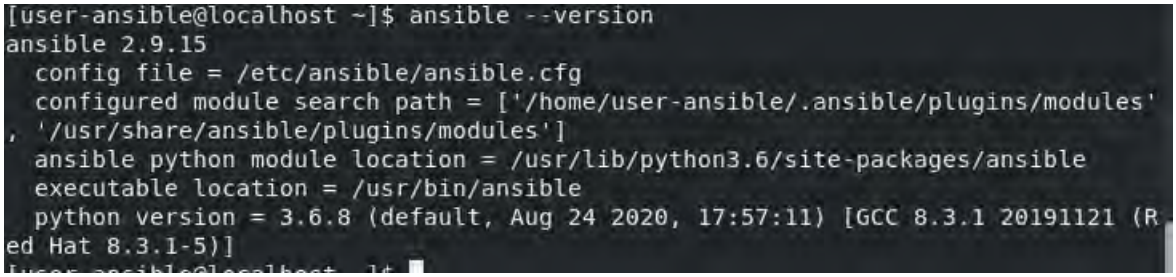

Figure 19 : Vérification de l'installation de Ansible

- ✓ **Préparation de la communication avec les nodes**

On va ajouter dans le fichier */etc/hosts* sur le node manager l'enregistrement des 2 nodes :

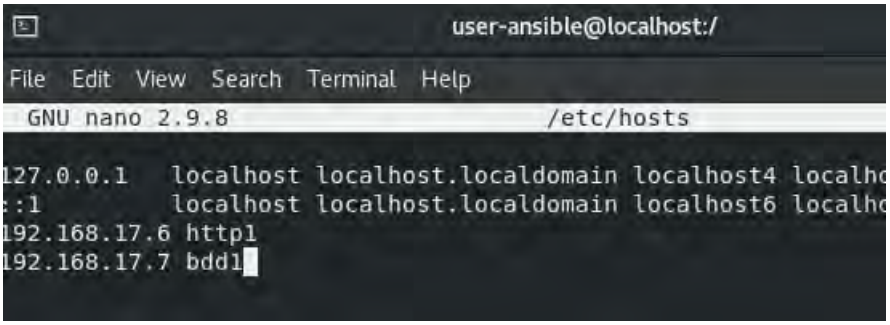

Figure 20 : Ajout des nodes dans le fichier /etc/hosts sur le node manager

Pour fonctionner, Ansible a besoin d'un fichier inventaire. Ce fichier contient la liste des nodes. On va donc enregistrer le nom des nodes dans ce fichier.

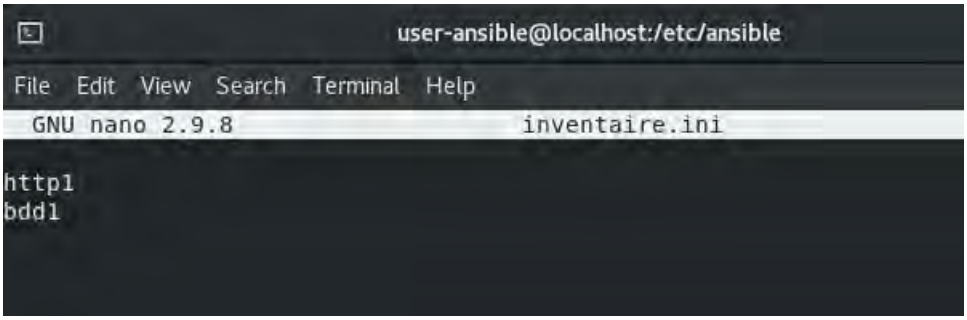

Figure 21 : Ajout du nom des nodes dans le fichier inventaire.ini

Pour se connecter en SSH, il est recommandé d'utiliser une paire de clés plutôt que d'utiliser un mot de passe. La communication SSH est établie sur la base de clés SSH ; cette pratique est conseillée, car elle permet un niveau d'authentification beaucoup plus sûr que l'authentification par mot de passe.

On va maintenant créer une paire de clés SSH de type ecdsa pour l'utilisateur user-ansible.

NB : Elliptic Curve Digital Signature Algorithm (ECDSA) est un algorithme de signature numérique qu'il est conseillé d'utiliser aujourd'hui pour avoir un niveau de sécurité satisfaisant.

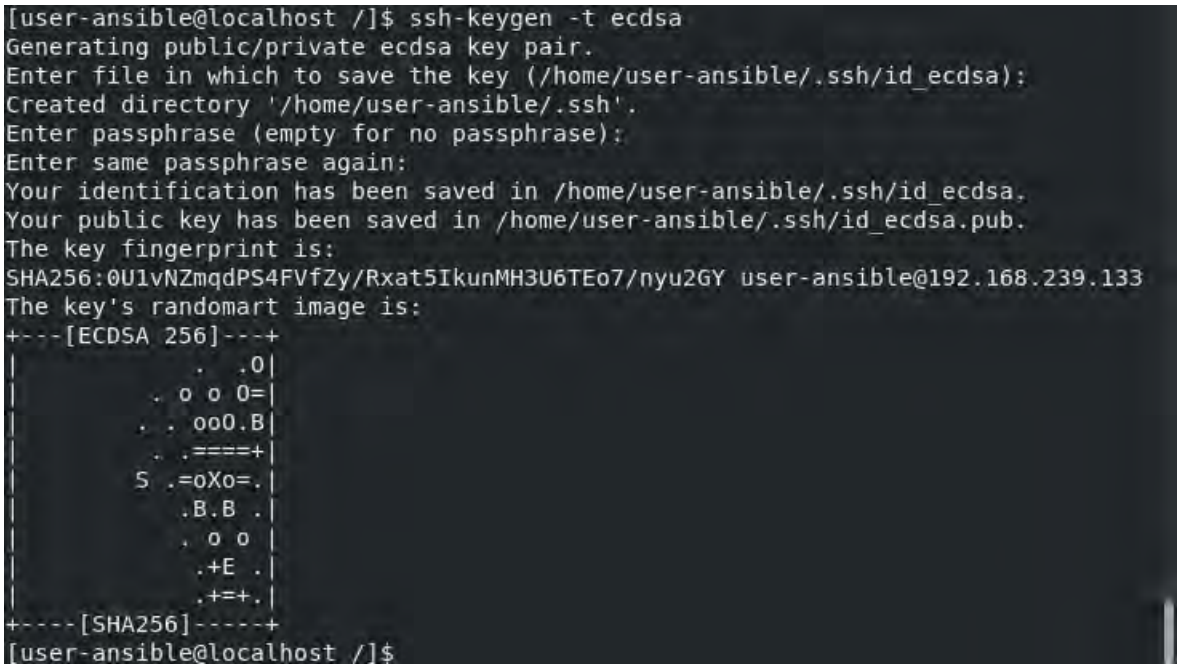

Figure 22 : Génération des clés ECDSA

On va utiliser le module *authorized_key* pour enregistrer la clé publique de l'utilisateur user-ansible sur tous les nodes. Ainsi, on pourra se connecter aux nodes sans saisir de mot de passe SSH.

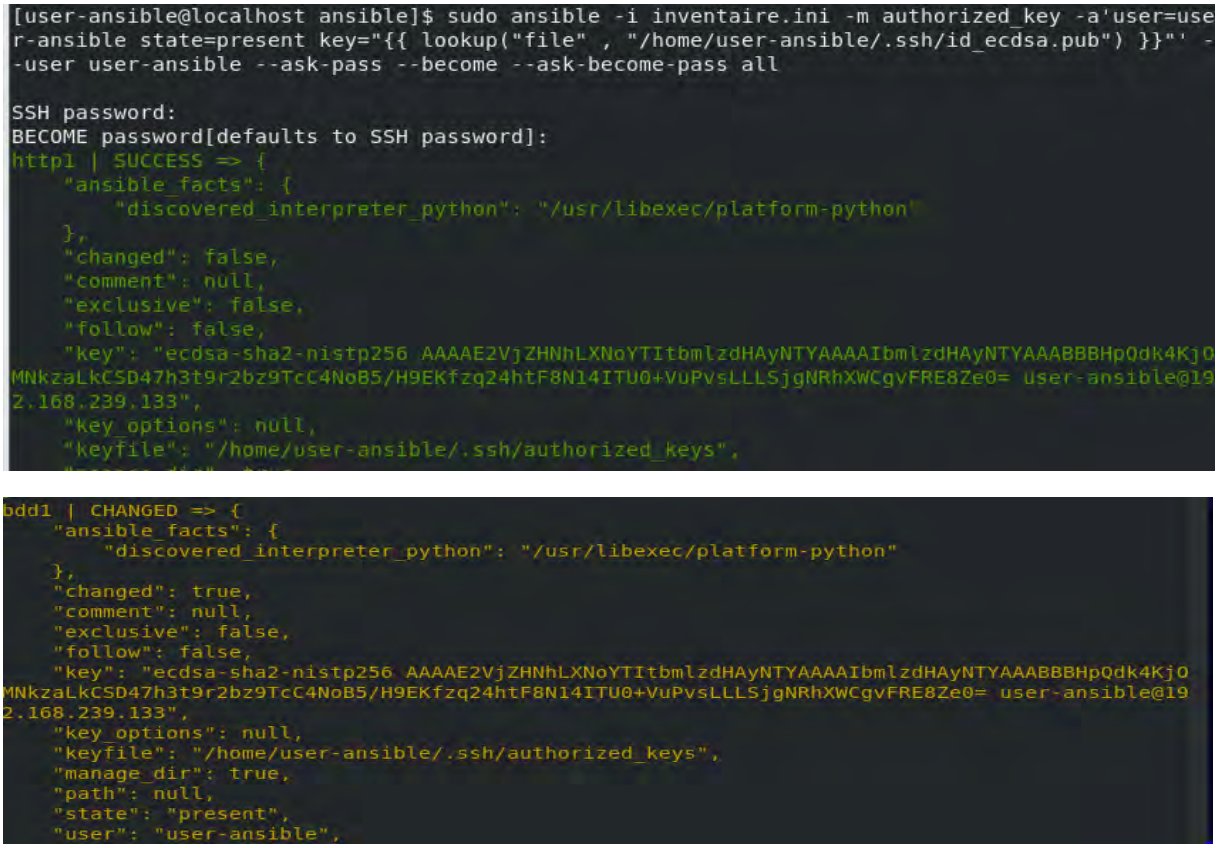

Figure 23 : Ajout de la clé publique ecdsa sur les nodes

- -m authorized_key : module authorized_key (ajoute ou supprime les clés SSH pour des utilisateurs) **;**
- user=user-ansible : l'utilisateur concerné est user-ansible ;
- state=present : indique d'ajouter le fichier ;
- key="{{ lookup("file", "/home/user-ansible/.ssh/id_ecdsa.pub") }}" : utilise la commande lookup pour rechercher le fichier concerné.

Après cela on peut tester de faire un ping ssh sur le node http1

Figure 24 : Test de ping

5.2. Mise en place de l'intranet à l'aide de Ansible

✓ **Structuration du déploiement avec les roles de Ansible**

Pour l'installation de MediaWiki on suivre six (6) étapes :

- Installer un serveur web Apache sur le premier serveur (http1) ;
- Installer PHP également sur le premier serveur (http1) ;
- Installer une base de données MariaDB sur le deuxième serveur (bdd1) ;
- Télécharger les fichiers sources de MediaWiki et les mettre sur le serveur web (http1) ;
- Configurer le serveur web (http1) pour pointer vers l'URL http://http1/mediawiki;
- Finaliser l'installation de MediaWiki via le script d'installation sur http1.

Pour ce faire on va créer cinq (5) rôles :

- Un rôle pour installer Apache : apache ;
- Un rôle pour installer MariaDB : mariadb ;
- Un rôle pour configurer Apache pour MediaWiki : confapache ;
- Un rôle pour configurer MariaDB pour MediaWiki : confdb ;
- Un rôle qui contiendra les variables globales : commun.

Par défaut, Ansible va chercher les rôles dans le répertoire "roles" qui est placé dans le répertoire de travail : celui dans lequel les commandes Ansible sont lancées.

Pour avoir une arborescence complète, on peut utiliser la commande *ansible-galaxy*.

NB : Ansible Galaxy fait référence au site web de Galaxy 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).

On va maintenant créer le rôle apache en utilisant *ansible-galaxy*

```
[user-ansible@192 roles]$ sudo ansible-galaxy init apache
[sudo] password for user-ansible:
- Role apache was created successfully
[user-ansible@192 roles]$
```

Figure 25 : Création du rôle apache

On va essayer de visualiser l'arborescence du rôle apache

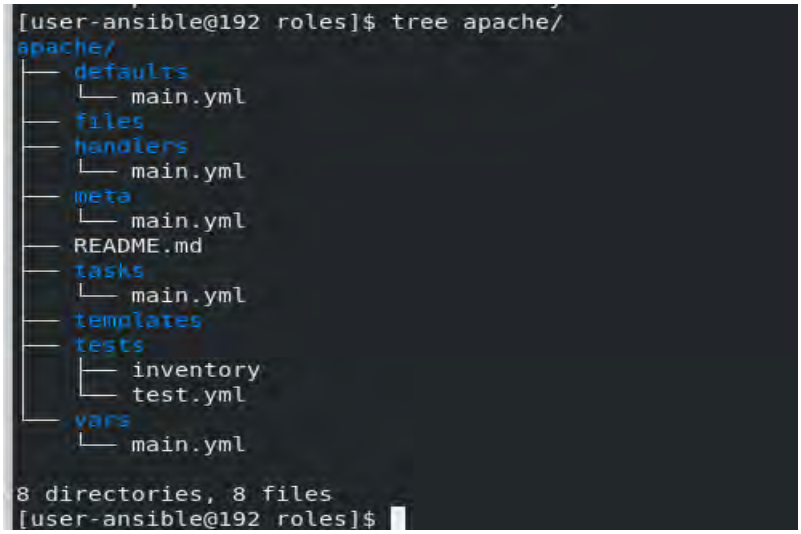

Figure 26 : L'arborescence du rôle apache

- 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).

✓ **Création de l'arborescence des roles MediaWiki**

On va créer, un à un, la structure des cinq (5) rôles pour le déploiement de MediaWiki.

La structure du rôle Apache est déjà créé.

Créons le rôle mariadb

Pour le rôle mariadb, on a besoin d'installer des paquets logiciels sur Linux et démarrer un service.

On n'a donc pas besoin d'utiliser ansible-galaxy pour le rôle mariadb, un seul fichier de configuration est nécessaire ici :

```
[user-ansible@192 roles]$ sudo mkdir -p mariadb/tasks/
[sudo] password for user-ansible:
[user-ansible@192 roles]$ sudo touch  mariadb/tasks/main.yml
[user-ansible@192 roles]$
```

Figure 27 : Création de l'arborescence du rôle mariadb

Le fichier main.yml contiendra les tâches à exécuter pour installer MariaDB.

Comme les opérations de configuration et les variables concernent spécifiquement le déploiement de MediaWiki, les rôles de configuration et commun vont être placés spécifiquement dans un répertoire *mediawiki*.

Nous créons donc un répertoire :

```
[user-ansible@192 roles]$ sudo mkdir mediawiki
[user-ansible@192 roles]$
```

Figure 28 : Création du répertoire mediawiki

Créons le rôle commun

Le rôle commun contiendra des variables partagées entre les rôles confapache et confdb. Ainsi, au lieu de définir les mêmes variables à deux endroits différents, il est préférable de créer un rôle commun, et d'utiliser une dépendance avec les autres rôles.

Pour cela, vous avez besoin du répertoire defaults et du fichiers main.yml qui contiendront les variables globales :

```
[user-ansible@192 roles]$ sudo mkdir -p mediawiki/commun/defaults
[user-ansible@192 roles]$ sudo touch mediawiki/commun/defaults/main.yml
[user-ansible@192 roles]$
```

Figure 29 : Création de l'arborescence du role commun

Créons le role confdb

Pour le rôle confdb, on a besoin de créer une base de données et d'attribuer des droits sur cette base.

Et nous créerons une dépendance avec le rôle commun pour partager des variables globales.

Pour cela, on a besoin des répertoires meta et tasks :

```
[user-ansible@192 roles]$ sudo mkdir -p mediawiki/confdb/meta mediawiki/confdb/t
asks
[user-ansible@192 roles]$
```

Figure 30 : Création de l'arborescence du rôle confdb

et des fichiers main.yml qui contiendront les actions :

```
[user-ansible@192 roles]$ sudo touch  mediawiki/confdb/meta/main.yml
[user-ansible@192 roles]$ sudo touch  mediawiki/confdb/tasks/main.yml
```

Figure 31 : Ajout des fichiers main.yml dans l'arborescence du role confdb

Créons le rôle confapache

On va partager des variables globales avec le rôle confdb, alors on ajoutera une dépendance avec le rôle commun.

On va également créer le répertoire d'installation de MediaWiki, télécharger les fichiers MediaWiki sur le site officiel, lancer le script d'installation et mettre à jour la base de données.

Pour cela, on a besoin des répertoires meta et tasks :

```
[user-ansible@192 roles]$ sudo mkdir -p mediawiki/confapache/meta mediawiki/conf
apache/tasks
[user-ansible@192 roles]$
```

Figure 32 : Création de l'arborescence du rôle confapache

Et des fichiers main.yml qui contiendront les actions :

```
[user-ansible@192 roles]$ sudo touch mediawiki/confapache/meta/main.yml mediawik
i/confapache/tasks/main.yml
[user-ansible@192 roles]$
```

Figure 33 : Ajout des fichiers main.yml dans l'arborescence du role confapache

- ✓ **Amélioration du fichier inventaire.ini en vue du déploiement**

Toujours dans l'esprit de modularité, on va modifier le fichier inventaire.ini pour séparer les deux nodes dans deux groupes distincts.

Ce qui permettra par la suite de s'adresser à un groupe plutôt qu'aux nodes directement. De cette façon, si on veut ajouter un node, on aura simplement à ajouter le node dans le bon groupe dans le fichier inventaire, et à rejouer nos scripts sans rien changer d'autre.

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                    inventaire.ini

[apache]
http1
[bd]
bdd1
```

Figure 34 : Modification du fichier inventaire.ini

Voici le travail à réaliser pour nos fichiers YAML

1. Le rôle **apache** :
    - un fichier **tasks/main.yml** contient les actions pour installer Apache, un appel à un fichier de configuration pour installer PHP et une notification pour redémarrer Apache ;
    - un fichier **tasks/php7-install.yml** contient les actions pour installer PHP ;
    - un fichier **handler/main.yml** contient les actions pour redémarrer le service Apache.
2. Le rôle **mariadb** :
    - un fichier **tasks/main.yml** contient les actions pour installer MariaDB.
3. Le rôle **commun** de MediaWiki :
    - un fichier **defaults/main.yml** contient les variables d'installation qui seront utilisées dans les rôles suivants.
4. Le rôle **confapache** de MediaWiki :
    - un fichier **meta/main.yml** contient la dépendance avec le rôle commun ;
    - un fichier **tasks/main.yml** contient les actions pour configurer Apache pour MediaWiki.
5. Le rôle **confdb** de MediaWiki :
    - un fichier **meta/main.yml** contient la dépendance avec le rôle commun ;
    - un fichier **tasks/main.yml** contient les actions pour configurer MariaDB pour MediaWiki.

Les rôles peuvent être indépendants ou dépendants les uns des autres. Par exemple, les rôles apache, mariadb et commun sont indépendants et peuvent être utilisés séparément. Par contre, les rôles confapache et confdb dépendent du rôle commun.

✓ **Construisons nos fichiers YAML**

Un fichier de configuration YAML contenu dans les rôles peut contenir une liste de tâches ou une liste de variables.

Les fichiers YAML commencent toujours par 3 tirets (---). Ensuite, on a les différentes tâches successives qui commencent par un (1) tiret (-) et le nom de la tâche. Chaque tâche utilise un

module avec ses arguments ou ses options. Les arguments ou les options sont décalés à la ligne de 2 espaces.

Pour construire un fichier, il faut chercher dans la documentation Ansible quel module utiliser.

Pour installer Apache, il faut utiliser un gestionnaire de paquets. Sur CentOS, nous utiliserons Yum.

Création du fichier YAML pour le rôle apache

Voici le fichier tasks/main.yml qui contient les actions d'installation d'Apache et de PHP.

Figure 35 : Contenu du fichier d'installation d'apache

Voyons en détail chacune des tâches décrites dans ce fichier :

1. La première tâche**, "Installation apache** " va installer le serveur Apache avec le module "**yum**". Le **name: "httpd"** indique le paquet concerné et le **state: "present"** spécifie qu'il faut l'installer.
2. La deuxième tâche, " **Activation service apache** " va activer le service Apache avec le module "**service**". Le **name: "httpd"** indique le service concerné, le **state: "started"** indique que le service sera démarré et le **enabled: yes** indique que le service sera activé.

3. La troisième tâche, "**Install PHP**" inclut un fichier de configuration externe pour installer PHP. La tâche fait appel avec l'option "**include**" au fichier **php-install.yml** qui est placé dans le répertoire **tasks** à coté de main.yml. La condition **when** avec le **filtre** php_install Voyons en détail chacune des tâches décrites dans ce fichier :
4. La première tâche, "apache installation" va installer le serveur Apache avec le module "yum". Le name: "httpd" indique le paquet concerné et le state: "present" spécifie qu'il faut l'installer.
5. La deuxième tâche, "apache service activation" va activer le service Apache avec le module "service". Le name: "httpd" indique le service concerné, le state: "started" indique que le service sera démarré et le enabled: yes indique que le service sera activé.
6. La troisième tâche, "install PHP" inclut un fichier de configuration externe pour installer PHP. La tâche fait appel avec l'option "include" au fichier php7-install.yml qui est placé dans le répertoire tasks à coté de main.yml. La condition when avec le filtre (php_install|default(False)|bool) permettent de conditionner l'installation de PHP.
7. install|default(False)|bool) permettent de conditionner l'installation de PHP.

NB : La variable php_install est un booléen par défaut à false. Cette variable va permettre d'indiquer si PHP doit être installé ou non. Cette variable sera initialisée à l'appel du rôle.

On va ajouter un fichier pour l'installation spécifique de PHP (php-install.yml) afin de séparer l'installation d'Apache et l'installation de PHP.

```
[user-ansible@192 roles]$ sudo nano apache/tasks/php-install.yml
```

Figure 36 : Création du fichier d'installation PHP

```
 GNU nano 2.9.8              apache/tasks/php-install.yml              Modified

---
#1. Installation du depot EPEL(Extra Packages for Entreprise Linux)
- name: "epel activation"
  yum:
name: "epel-release"
state: present

#2. Installation depot REMI pr beneficier du paquet php7
- name: "remi repo activation"
  yum:
name: "https://rpms.remirepo.net/entreprise/remi-release-7.rpm"
state: present

#3. Installattion PHP et ses extensions
- name: "Install PHP"
  yum:
name: "php,php-mysql,php-xml,php-mbstring,php-mcrypt,php-gd,php-intl"
state: latest
enablerepo: "remi-php70"

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

```
name: "php,php-mysql,php-xml,php-mbstring,php-mcrypt,php-gd,php-intl"
state: latest
enablerepo: "remi-php70"
  changed_when: yes
  notify: [ "apache restart" ]
```

Figure 37 : Contenu du fichier d'installation de PHP

Voyons en détail chacune des tâches décrites dans ce fichier :

1. La première tâche, "**epel activation**", va activer le dépôt **epel** avec le module "**yum**". Le **name: "epel-release"** indique le dépôt concerné et le **state: present** spécifie qu'il faut l'installer.
   Sur **CentOS**, tous les logiciels ne sont pas disponibles par défaut. C'est le cas pour **PHP7** ; il faut donc ajouter le dépôt **EPEL** qui va contenir le paquet **PHP7**.
2. La deuxième tâche, "**remi repo activation**", va installer le paquet remi-release-7.rpm avec le module "**yum**". Le nom indique le service concerné ; le **state: present** indique que le paquet sera installé.
3. Sous **CentOS**, les paquets logiciels sont sous la forme **RPM**. Pour installer **PHP7**, nous avons besoin du paquet **remi-release-7.rpm** qui est dans le dépôt **REMI** qui dépend du dépôt **EPEL**.

4. La troisième tâche, "**Install PHP**", installe les paquets php7 avec le module "**yum**". Et les options :
    - le **name** indique l'ensemble des paquets à installer ;
    - **state: latest** indique qu'il faut installer les dernières versions disponibles des paquets ;
    - **enablerepo: "remi-php70"** indique d'utiliser le dépôt **remi-php70** pour les installer ;
    - **changed_when** : force le changement d'état, c'est-à-dire qu'avec cette condition à **yes**, l'exécution de la tâche provoquera un changement ;
    - **notify: ["apache restart"]** indique que si la tâche change d'état (et uniquement si la tâche a engendré un changement), **notify** fait appel au **handler** "apache restart" pour relancer le service Apache.

NB : Ansible est idempotent, cela signifie qu'une opération a le même effet, qu'on l'applique une ou plusieurs fois. Si l'état demandé dans l'action est conforme à ce qui est demandé, alors Ansible ne fait rien. L'idempotence est possible car Ansible gère 3 états d'exécution pour une tâche : ***skipping***, ***ok*** ou ***changed***.

Créons le script handler pour relancer un service

Les handlers, ou script handlers, sont communément utilisés pour regrouper des actions récurrentes, comme des redémarrages de services, par exemple.

Les handlers sont exécutés uniquement si une tâche les appelle à travers un notify, une notification déclenchée par la tâche. Le notify se déclenche si l'état de la tâche est changed.

C'est plutôt utile pour redémarrer un service si et seulement si l'état du serveur a changé, ce qui évite des interruptions de service inutiles.

Dans le fichier roles/apache/handlers/main.yml :

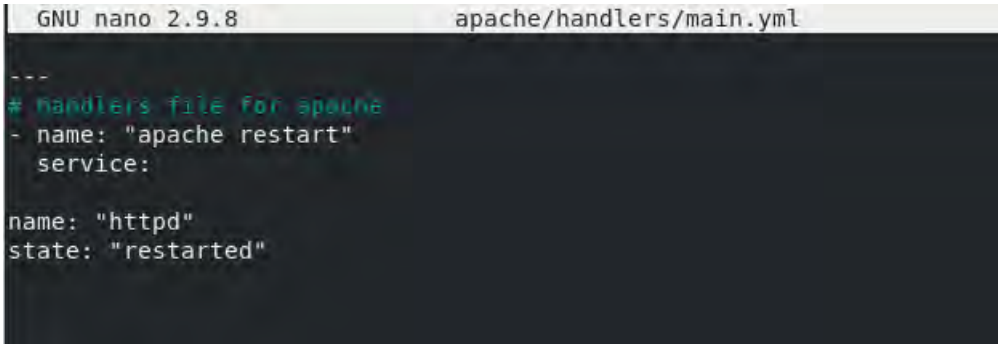

Figure 38 : Contenu du fichier pour relancer le service apache

Le module service est utilisé pour relancer le service **httpd** avec l'option **state: "restarted"**.

Création du fichier YAML pour le rôle mariadb :

Figure 39 : Modification du fichier d'installation de mariadb

Figure 40 : Contenu du fichier d'installation de mariadb

La première tâche "**installation serveur mariadb**" va installer **mariadb** serveur et son extension **Python** avec le module "**yum**". Le **name: "mariadb-server,MySQL-python"** indique les paquets concernés et **state: "installed"** indique que les paquets doivent être installés.

La deuxième tâche "**Demarre le service mariadb** " va activer et démarrer le service MariaDB avec le module "**service**". Le **name: "mariadb"** indique le service concerné, **state: "started"** indique de démarrer le service et **enabled:yes**, de l'activer.

Création du fichier YAML pour le rôle commun :

Les variables globales vont être utilisées dans les rôles confapache et confdb. Elles vont donc être définies dans le rôle commun, dans le répertoire defaults et dans le fichier main.yml.

Dans ce fichier il y aura une variable pour stocker le mot de passe de la base de données. Et il n'est vraiment pas recommandé de mettre des mots de passe en clair dans les fichiers de configuration.

Dans ce projet on utilise **Ansible2021** comme mot de passe. Pour chiffrer le mot de passe qui sera utilisé dans le fichier de variables, on utiliser la commande suivante :

```
[user-ansible@192 roles]$ sudo ansible-vault encrypt_string 'foobar' --name 'med
iawiki_db_password'
New Vault password:
Confirm New Vault password:
mediawiki_db_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          3461636465616135343431343233383837306632656562336532376131626139666439
3737353930
          3233393034323733636232353362643363626165643735340a3166373465633636336 6
6235323863
          3838313565346366373531646265376633633763636535666335393937396637336262
3335326533
          6665663939343033320a6334306431333938316334616363346366313432303064643 5
6566393063
          6465
Encryption successful
[user-ansible@192 roles]$
```

Figure 41 : Chiffrement du mot de passe pour la base de données

```
[user-ansible@192 roles]$ sudo nano mediawiki/commun/defaults/main.yml
```

Figure 42 : Modification du fichier de variables

```
  GNU nano 2.9.8          mediawiki/commun/defaults/main.yml          Modified

---
# Nom de la bdd
mediawiki_db_name: "mediawiki"

# username et mot de passe

mediawiki_db_user: "mediawiki"
mediawiki_db_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          3461636465616135343431343233383837306632656562336532376131626139666643$
          3233393034323733636232353362643363626165643735340a3166373465633636336$
          3838313565346366373531646265376633633763636535666335393937396637336262$
          6665663939343033320a6334306431333938316334616363346366313432303064643$
          6465

# nom admin et mot de passe
mediawiki_admin_user: "admin"
mediawiki_admin_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
```

```
# nom admin et mot de passe
mediawiki_admin_user: "admin"
mediawiki_admin_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          346163646561613534343134323338383730663265656233653237613162613966643$
          323339303432373363623235336264336362616564373534 0a316637346563363633 6$
          383831356534636637353164626537663363376363653566633539393739663733626$
          666566393934303332 0a6334306431333938316334616363346366313432303064643$
          6465

#Nom du mediawiki et son titre
mediawiki_name: "mediawiki"
mediawiki_title: "TEST"
```

```
#Nom du mediawiki et son titre
mediawiki_name: "mediawiki"
mediawiki_title: "TEST"

#Repertoire d'installation de mediawiki
mediawiki_directory: "/var/www/html/{{mediawiki_name}}"

#Repertoire de maintenace
mediawiki_maintenance_directory: "{{mediawiki_name}}/maintenance"

#Definir le premier node du groupe mariadb
mediawiki_db_host: "{{groups.db.0}}"
```

```
#URL des sources mediawiki
mediawiki_archive_url: "https://releases.wikimedia.org/mediawiki/1.31/mediawiki$
```

Figure 43 : Contenu du fichier de variables

Au format YAML, les variables sont définies par un couple “clé: valeur” séparé par deux points (:).

On peut voir également dans ce fichier, que deux valeurs sont utilisées : {{mediawiki_name}} et {{mediawiki_directory}}, qui font respectivement appel aux deux variables mediawiki_name et mediawiki_directory contenus dans le fichier.

Création du fichier YAML pour le role confdb :

Pour que le rôle **confdb** puisse utiliser les **variables** enregistrées dans le rôle **commun**, il faut déclarer cette **dépendance** dans le répertoire **meta** :

```
[user-ansible@192 roles]$ nano mediawiki/confdb/meta/main.yml
```

Figure 44 : Modification du fichier de dépendance pour le role confdb

```
 GNU nano 2.9.8            mediawiki/confdb/meta/main.yml

---
dependencies:
  -role: "mediawiki/commun"
```

Figure 45 : Contenu du fichier de dépendance pour le role confdb

On va maintenant écrire le code pour configurer MediaWiki

```
[user-ansible@192 roles]$ sudo nano mediawiki/mariadb/tasks/main.yml
```

Figure 46 : Modification du fichier de configuration de MediaWiki

```
#1. INstallation de la bdd mediawiki
- name: "mediawiki database"
  mysql_db:
name: "{{mediawiki_db_name}}
state: present

#2. Creation d'un acces utilisateur et privileges sur la bdd mediawiki
- name: "mediawiki user+privileges"
  mysql_user:
name: "{{mediawiki_db_user}}"
password: "{{mediawiki_db_password}}"
priv: "{{mediawiki_db_name}}.*:ALL"
host: ""{{item}}
state: present
  with_items: "{{groups.apache}}"
```

Figure 47 : Code du fichier de configuration de MediaWiki

1. La première tâche "**Installataion de la bdd mediawiki**" va créer la base de données avec le module "**mysql_db**". Et les options :
    - le **name: "{{mediawiki_db_name}}"** indique le nom de la base de données qui est récupéré de la variable **mediawiki_db_name** définie dans le rôle **commun** ;
    - le **state: present** indique de créer la base de données.
2. La **deuxième tâche** "**mediawiki user+privileges**" va créer un accès utilisateur et des privilèges associés sur la base de données pour tous les nodes avec le module "**mysql_user**". Et les options :
    - le **name: "{{mediawiki_db_user}}"** indique le nom d'utilisateur à créer ;
    - **password: "{{mediawiki_db_password}}"** indique le mot de passe à utiliser ;
    - **priv: "{{mediawiki_db_name}}.*:ALL"** indique les privilèges à donner à l'utilisateur ;

- **host: "{{item}}"** indique le node concerné par les accès ;
- **state: present** indique de créer l'utilisateur et ses accès.
- **with_items**: La boucle **with_items** va parcourir le **groupe Apache** et remplacer successivement la variable **item** par les noms des nodes présents dans le groupe ;
- "{{**groups.apache**}}" : cette variable "magique" est tirée directement du fichier inventaire.

Création du fichier YAML pour le role confapache :

Pour que le rôle confapache puisse utiliser les variables enregistrées dans le rôle commun, il faut déclarer cette dépendance dans le répertoire meta :

```
[user-ansible@192 roles]$ sudo nano mediawiki/confapache/meta/main.yml
```

Figure 48 : Modification du fichier de dépendance pour le role confapache

```
 GNU nano 2.9.8          mediawiki/confapache/meta/main.yml

dependencies:
  - role: "mediawiki/commun"
```

Figure 49 : Code du fichier de dépendance pour le role confapache

On va maintenant terminer en configurant les fichiers MediaWiki dans Apache

```
[user-ansible@192 roles]$ sudo nano mediawiki/confapache/tasks/main.yml
```

Figure 50 : Modification des fichiers de Configuration mediawiki

```
---

#1. Création du repertoire pour l'installation des fichiers Mediawiki
- name: "mediawiki directory"
  file:
path: "{{mediawiki_directory}}"
owner: "apache"
group: "apache"
state: directory

#2. Décompresse le fichier source archive Mediawiki et le formate sans extensio
- name: "uncompress mediawiki archive"
  unarchive:
src: "{{mediawiki_archive_url}}"
dest: "{{mediawiki_directory}}"
owner: "apache"
group: "apache"
remote_src: yes
# supprime mediawiki-1.xx.x/ du chemin
```

```
# Supprime mediawiki-1.XX.X/ du chemin
extra_opts: --transform=s/mediawiki-[0-9\.]*\///

#3. Exécute la tâche avec l'utilisateur apache, se place dans le répertoire de $
- name: "mediawiki configuration"
  become: yes
  become_user: "apache"
  args:
   creates: "{{mediawiki_directory}}/LocalSettings.php"
   chdir: "{{mediawiki_maintenance_directory}}"
  command:
```

```
  command:
php install.php --scriptpath /{{mediawiki_name}}
    --dbname mediawiki --lang fr
    --dbuser {{mediawiki_db_user}}
    --dbpass {{mediawiki_db_password}}
    --pass {{mediawiki_admin_password}}
    --dbserver {{mediawiki_db_host}}
    {{mediawiki_title}} {{mediawiki_admin_user}}
  run_once: yes
  delegate_to: "{{item}}"
  with_items: "{{groups.apache}}"
```

```
#4. Exécute la tâche avec l'utilisateur apache, se place dans le répertoire de $
- name: "mediawiki db update"
  become: yes
  become_user: "apache"
  command:
php update.php --quick
  args:
chdir: "{{mediawiki_maintenance_directory}}"
  # La mise à jour à besoin d'être lancée une seule fois
```

```
  # La mise à jour à besoin d'être lancée une seule fois
  run_once: yes
  register: resultat
  changed_when: "' ...done.' in resultat.stdout"
```

Figure 51 : Code des fichiers de Configuration mediawiki

Voyons le détail de ces tâches :

La première tâche, "**mediawiki directory**", va créer le répertoire dans lequel les fichiers MediaWiki seront placés avec le module "**file**". Et les options :

- **path: "{{mediawiki_directory}}"** indique le chemin du répertoire à créer ;
- **owner: "apache"** indique le propriétaire du répertoire ;
- **group: "apache"** indique le groupe du répertoire ;
- **state: directory** indique de créer un répertoire.

La deuxième tâche, "**mediawiki directory**", va décompresser le fichier source archive MediaWiki et le renommer avec le module "**unarchive**". Et les options :

- **src: "{{mediawiki_archive_url}}"** indique l'adresse de l'archive à télécharger ;
- **dest: "{{mediawiki_directory}}"** indique le chemin où doit être décompressée l'archive ;
- **owner: "apache"** indique le propriétaire des fichiers ;
- **group: "apache"** indique le groupe des fichiers ;
- **remote_src: yes** indique que la source est externe ;
- **extra_opts: --transform=s/mediawiki-[0-9\.]*\/// ** indique de renommer le répertoire avec le nom *mediawiki*.

La troisième tâche, "**mediawiki configuration**", va configurer MediaWiki à l'aide d'une commande PHP en utilisant le module command. Et les arguments et options :

- **become: yes** indique d'utiliser un autre utilisateur ;
- **become_user**: "apache" indique d'utiliser le compte Apache pour faire les actions ;
- **args:** indique les arguments à utiliser ;
- **creates: "{{mediawiki_directory}}/LocalSettings.php"** indique de ne rien faire si le fichier LocalSettings.php existe déjà :
    - **chdir: "{{mediawiki_maintenance_directory}}"** indique de se placer dans le répertoire Maintenance,
    - **command:** indique d'utiliser le module command pour lancer le script install.php avec les options de configuration ;
- **run_once: yes** indique de l'exécuter une seule fois ;
- **delegate_to: "{{item}}"** indique de déléguer l'action à un item de la liste ;
- **with_items: "{{groups.apache}}"** indique le nom du node du groupe Apache défini dans l'inventaire et le place dans Item.

La quatrième tâche, "**mediawiki db update**", va mettre à jour la base de données pour finir la configuration de MediaWiki avec le module command. Et les options :

- **become: yes** indique de prendre l'identité d'un autre utilisateur ;
- **become_user: "apache"** indique d'utiliser le compte Apache ;

- **command:** indique de lancer le script update.php à l'aide de la commande PHP et l'option rapide ;
- **args:** indique de se placer dans le répertoire de maintenance ;
- **run_once: yes** indique de faire l'action une seule fois ;
- **register: resultat** indique de stocker le résultat dans la variable result ;
- **changed_when: "' ...done.' in resultat.stdout"** indique que le changement est effectif quand l'action retourne done.

Quelques notions particulières de ce fichier sont à préciser :

- **args** : la section **args** va mettre d'indiquer à la tâche les prérequis à l'exécution des actions ;
- **run_once** : l'action sera exécutée une seule fois ;
- **delegate_to** : la tâche est déléguée au premier node du groupe Apache. Couplée avec **run_once**, l'action sera exécutée sur un seul node, ce qui évitera des corruptions de base, si les actions sont jouées sur deux nodes en même temps ;
- **register** : va stocker le résultat de l'action dans la variable **resultat**. Couplé à **changed_when**, l'état de la tâche sera indiqué **changed,** si l'action retourne **done**.

Dans la suite, on va créer trois (3) playbooks :

1. Un pour **installer Apache** ;
2. Un pour **installer MariaDB** ;
3. Un pour **configurer MediaWiki**.

L'ordre d'exécution de ces playbooks est important, car les rôles de configuration dépendent des rôles d'installation. Les serveurs **Apache** et **MariaDB** doivent être **installés avant** que la **configuration** de MediaWiki puisse être lancée. De la même façon, la **configuration** de MediaWiki sur le serveur **MariaDB** doit être effectuée **avant** la configuration du serveur state: present.

✓ **Création du playbook pour installer Apache**

L'installation d'Apache va consister à lancer le rôle apache et à définir la variable qui permettra d'indiquer si PHP doit être installé ou pas.

Créons le playbook install-apache.yml

```
[user-ansible@192 ansible]$ nano install-apache.yml
```

Figure 52 : Création du playbook d'installation d'Apache

```
 GNU nano 2.9.8                  install-apache.yml

---
- name: "Installation apache"
  hosts: http1
  roles:
    - role: "apache"
      php_install: yes
```

Figure 53 : Code du playbook d'installation d'Apache

Le playbook est un fichier **YAML**. Il est structuré de la façon suivante :

- Il commence par **3 tirets** ;
- Ensuite, il peut y avoir un **bloc général**, constitué d'un **en-tête** composé du **nom des nodes ou du groupe concerné**, de **variables**, **d'options**, etc.
- Puis, vous trouvez des **blocs spécifiques** qui définissent les **jeux d'instruction** (ou ***play*** en anglais). Chaque jeu d'instruction est composé du **nom du jeu** (avec un tiret au début), et de façon **optionnelle**, d'un **en-tête** et d'une **section** qui peut prendre plusieurs formes comme des **tasks**, des **rôles**, des **handlers**… en fonction de l'action demandée.
- Dans le cas du playbook install-apache.yml ci-dessus, il y a deux (2) jeux d'instruction.
- Chaque **ligne** à l'intérieur d'un bloc est **indentée**, et chaque début de ligne est décalé de **deux espaces**.

Comme on peut le voir, la principale différence entre un **playbook** et un fichier de configuration de type **main.yml** est que le **playbook** contient une liste de jeux d'instructions (**plays**) et que le **main.yml** contient une liste de tâches (**tasks**) ou de **variables**.

Dans **install-apache.yml,** il n'y a pas de bloc général. Par contre, il y a un bloc spécifique pour définir le jeu d'instructions qui consiste à lancer le rôle apache et à définir la valeur de la variable **php_install** (ce qui déclenchera l'installation de PHP). Il y a également un en-tête qui indique sur quel node il faut lancer les actions.

Le jeu d'instructions est défini de la façon suivante :

- - name: "Installation apache" indique le nom du jeu d'instructions ;
- hosts: http1 indique le node concerné ;
- roles: indique une section rôles ;
- - role: "apache" indique le rôle à lancer ;
- php_install: yes indique la valeur de la variable php_install.

On va maintenant lancer le playbook pour installer apache sur **http1** avec la commande *ansible-playbook*

Vu qu'on a publié la clé publique sur les nodes, pour faire du déploiement sur les nodes il faut préciser la clé privée dans le fichier de configuration d'Ansible **ansible.cfg :**

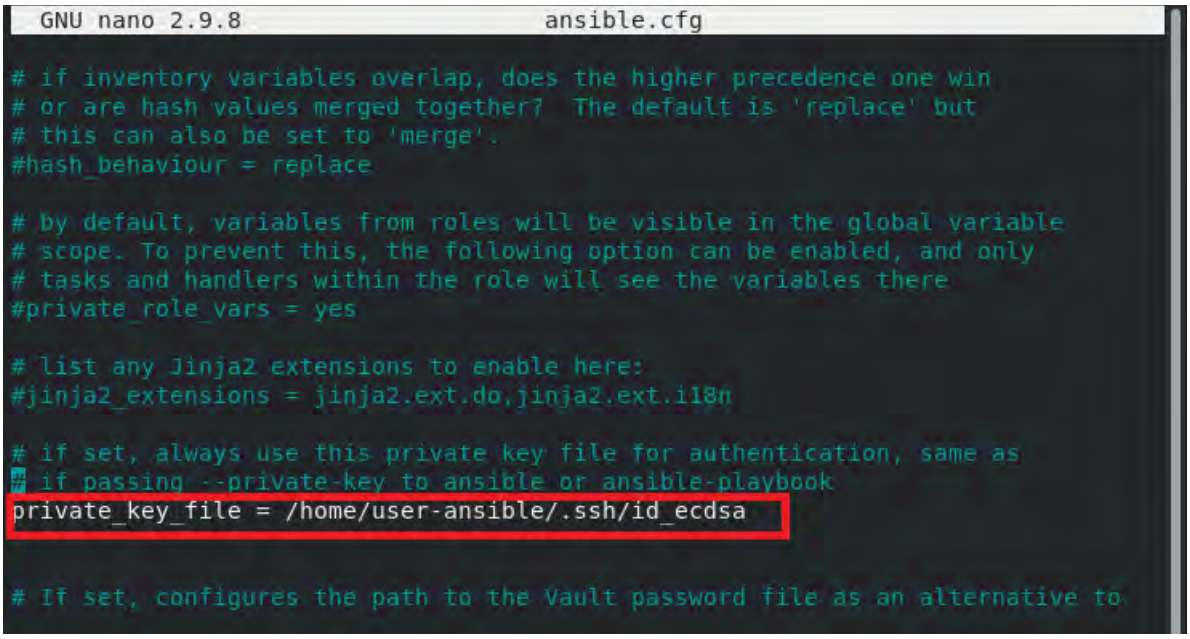

Figure 54 : Enregistrement de la clé privée dans le fichier de configuration d'Ansible

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

```
[user-ansible@localhost ansible]$ sudo ansible-playbook -i inventaire.ini --user user-
ansible --become --ask-become-pass install-apache.yml
[sudo] password for user-ansible:
BECOME password:

PLAY [Installation apache] *****************************************************

TASK [Gathering Facts] *********************************************************
ok: [http1]

TASK [Installation apache] *****************************************************
ok: [http1]

TASK [Activation service apache] ***********************************************
ok: [http1]

TASK [apache : epel activation] ************************************************
ok: [http1]

TASK [apache : Install PHP] ****************************************************
changed: [http1]

PLAY RECAP *********************************************************************
http1                      : ok=5    changed=1    unreachable=0    failed=0    skipped
=0    rescued=0    ignored=0

[user-ansible@localhost ansible]$
```

Figure 55 : Lancement du playbook pour installer apache

On peut constater que le callback donne la liste des tâches pour chaque action définie dans le rôle apache, avec leur état (changed ou ok). Ce qui signifie que toutes les actions ont provoqué un changement sur le node http1.

La dernière ligne du callback indique le récapitulatif par node, de l'exécution des tâches selon les quatre états possibles (ok, changed, unreachable, failed).

Il est temps de vérifier sur le node http1, si Apache et PHP ont bien été installés :

```
[user-ansible@192 tasks]$ ssh user-ansible@http1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Feb  6 05:50:02 2021 from 192.168.17.5
[user-ansible@192 ~]$ php --version
PHP 7.2.24 (cli) (built: Oct 22 2019 08:28:36) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
[user-ansible@192 ~]$
```

Figure 56 : Vérification de l'installation de PHP

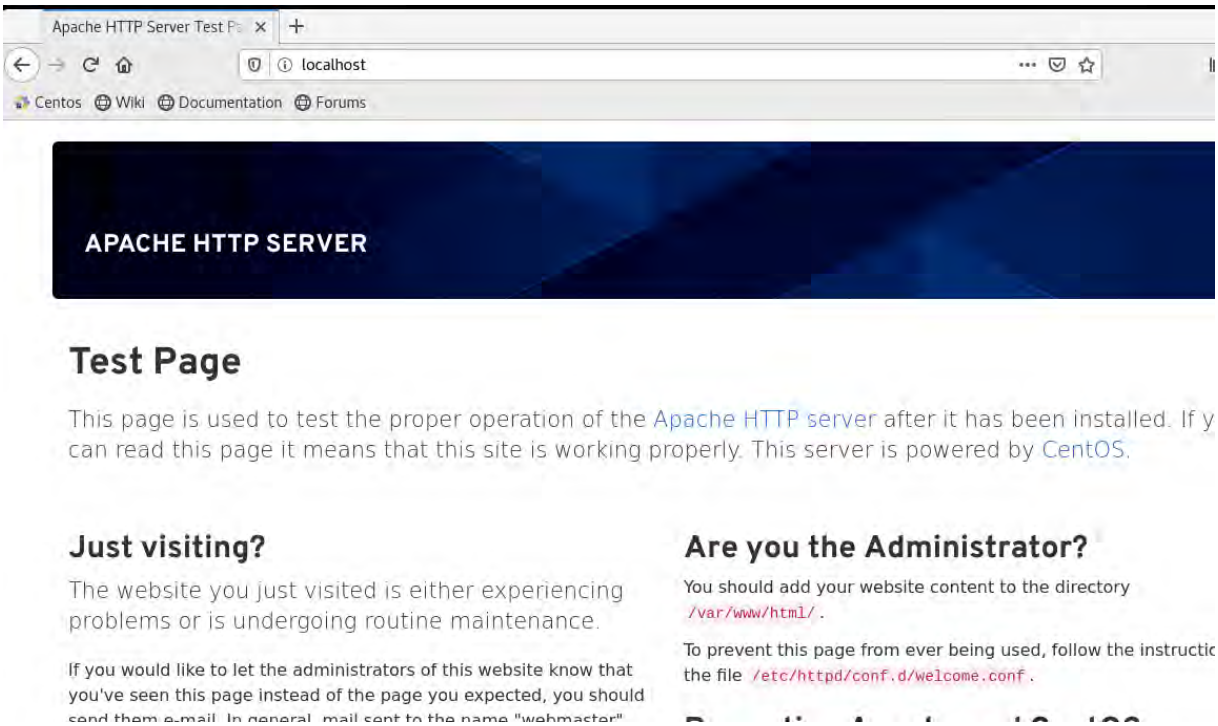

Figure 57 : Vérification de l'installation de Apache

✓ **Création du playbook pour installer MariaDB :**

L'installation de MariaDB va simplement consister à lancer le rôle mariadb.

Créons le playbook install-mariadb.yml :

```
[user-ansible@localhost ansible]$ sudo nano install-mariadb.yml
```

Figure 58 : Création du playbook d'installation de MariaDB

```
  GNU nano 2.9.8          /etc/ansible/install-mariadb.yml

---
- name: "Installation MariaDB"
  hosts: bdd1
  gather_facts: no
  roles:
    - role: mediawiki/mariadb
```

Figure 59 : Contenu du playbook d'installation de MariaDB

Le playbook contient une seule tâche, qui définit le nom de **l'host** concerné par l'action, l'option **gather_facts** est désactivée et le **rôle** à lancer est "mariadb".

NB : L'option **gather_facts** permet de collecter des informations sur les nodes distants.

Passons à l'exécution du playbook

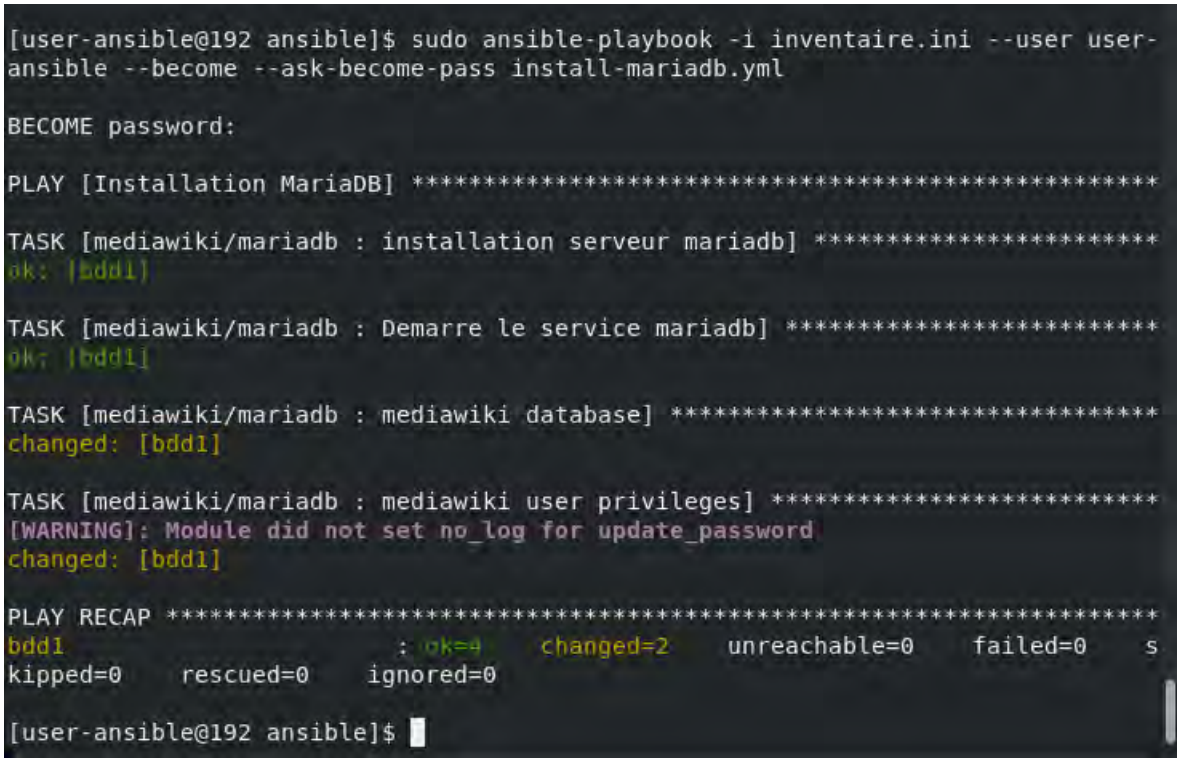

Figure 60 : Exécution du playbook d'installation de MariaDB

Même chose que pour le playbook précédent, les différentes étapes d'exécution du rôle **mariadb** sont listées, et un résumé est donné en fin de liste avec l'état global de l'exécution. Deux tâches ont été exécutées, elles ont provoqué un changement sur le node bdd1.

Pour le vérifier, on se connecte sur le node **bdd1** et interrogez le **statut** du **service MariaDB** avec la commande suivante :

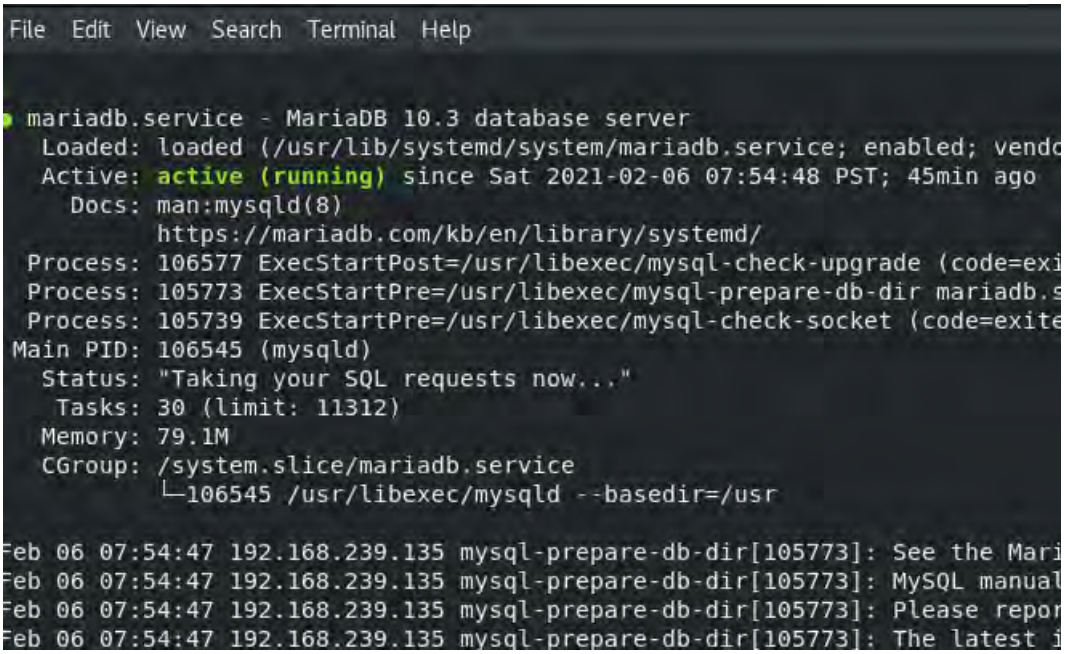

Figure 61 : Vérification de l'installation de MariaDB

- ✓ **Création du playbook pour installer MediaWiki :**

La configuration de MediaWiki va consister à lancer les rôles **confapache** et **confdb**.

Pour cela, nous allons créer un playbook **install-mediawiki.yml :**

```
[user-ansible@192 ansible]$ sudo nano install-mediawiki.yml
```

Figure 62 : Création du playbook pour installer MediaWiki

```
 GNU nano 2.9.8                 install-mediawiki.yml

---
- name: "MediaWiki db configuration"
  hosts: db
  gather_facts: no
  tags: [ "mariadb", "mysql" ]
  roles:
    - role: "mediawiki/confdb"

- name: "MediaWiki apache configuration"
  hosts: apache
  gather_facts: no
  tags: "apache"
  roles:
    - role: "mediawiki/confapache"
```

Figure 63 : Contenu du playbook pour installer MediaWiki

Le **playbook** contient deux tâches, une pour chaque rôle. Il est important de commencer par la configuration de **MariaDB** car la suite en dépend. Pour configurer

MediaWiki sur Apache, la base de données doit déjà être créée. Toutes ces instructions sont contenues dans le guide d'installation de MediaWiki.

Pour chaque tâche est défini le **host** concerné par les actions, le **gather_facts** est désactivé et des **tags** ont été ajoutés aux tâches.

NB : Lorsque le playbook est volumineux, il peut s'avérer utile de pouvoir exécuter une partie spécifique, plutôt que de tout exécuter d'un coup. Pour cela, Ansible prend en charge un attribut « tags ». Le ou les tags à exécuter pourront être indiqués dans les options de la commande ansible-playbook

Maintenant lançons le playbook :

```
PLAY [MediaWiki db configuration] ***********************************************

TASK [mediawiki/confdb : mediawiki database] ************************************
changed: [bdd1]

TASK [mediawiki/confdb : mediawiki user+privileges] *****************************
changed: [bdd1] => (item=http1)

PLAY [MediaWiki apache configuration] *******************************************

TASK [mediawiki/confapache : mediawiki directory] *******************************
changed: [http1]

TASK [mediawiki/confapache : uncompress mediawiki archive] **********************
changed: [http1]

TASK [mediawiki/confapache : mediawiki configuration] ***************************
changed: [http1 -> http1] => (item=http1)


TASK [mediawiki/confapache : mediawiki db update] *******************************
changed: [http1]

PLAY RECAP **********************************************************************
bdd1 : ok=2 changed=2 unreachable=0 failed=0
http1 : ok=4 changed=4 unreachable=0 failed=0
```

Figure 64 : Exécution du playbook de configuration de MediaWiki

Sur le node http1 on va essayer de se connecter à localhost :

Figure 65 : Page d'accueil de MediaWiki

5.3. Amélioration de l'infrastructure

✓ **Mise en place d'un motd**

La première amélioration que nous allons mettre en place peut sembler cosmétique, mais elle est en fait assez importante. Nous allons en effet modifier le fichier **/etc/motd** du serveur afin que, lorsque l'on se connecte sur le système, un message spécifique à ce système soit affiché. Ceci est une bonne pratique à respecter.

Il est rare qu'une personne maintienne un seul système et l'on se retrouve donc souvent connecté à différents serveurs au même moment. Il suffit de rappeler maladroitement la mauvaise commande de connexion pour se retrouver à exécuter la bonne opération... sur la mauvaise machine ! Disposer d'un motd qui rappelle, à la connexion, où l'on se trouve peut ainsi éviter bien des malheurs...

Nous allons donc modifier notre motd pour qu'il affiche le nom du serveur.

Nous allons aussi en profiter pour rajouter quelques informations à l'aide de celles à la disposition d'Ansible (« facts ») lors de la génération du fichier telles que le nom complet du système (FQDN ou « fully qualified domain named » et les adresses IP du système.

En effet, si Ansible sera toujours en mesure de déterminer le nom du système, ainsi que son FQDN, il n'est pas impossible que le système ne dispose pas d'adresses IPv4 ou IPv6. Il va

donc falloir générer cette partie du contenu du fichier motd en fonction des capacités du système.

Afin d'accomplir ceci, nous allons introduire des structures de contrôle issues de la syntaxe du moteur de génération de fichier (Jinja2). Il s'agit en fait, de simples tests conditionnels (instruction if) et d'un mécanisme d'itération (boucle for).

Voici donc le contenu de notre fichier modèle **motd.j2** :

```
  GNU nano 2.9.8                    templates/motd.j2

# templates/motd.j2
Serveur: {{ ansible_hostname }}
FQDN: {{ ansible_fqdn }}
{% if ansible_all_ipv4_addresses is not none %}
Adresses IPv4:
  {% for ip in ansible_all_ipv4_addresses %}
- {{ ip }}
  {% endfor %}
{% endif %}
{% if ansible_all_ipv6_addresses is not none %}
Adresses IPv6:
  {% for ip in ansible_all_ipv6_addresses %}
* {{ ip }}
  {% endfor %}
{% endif %}
```

Figure 66 : Contenu du fichier modèle pour le motd

Créons un playbook pour notre modèle motd :

```
[user-ansible@192 ansible]$ sudo nano motd.yml
```

Figure 67 : Création du playbook pour le modèle motd

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                    motd.yml

---
- name: \Installation du serveur intranet\
  hosts: localhost
  gather_facts: true
  vars:
  tasks:
    - name: \Configurer motd pour ce serveur\
      template:
        src: templates/motd.j2
        dest: /etc/motd
```

Figure 68 : Contenu du playbook pour le fichier modèle motd

```
[user-ansible@192 ansible]$ ansible-playbook --user user-ansible --become --ask-
become-pass motd.yml
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [\Installation du serveur intranet\] ***********************************

TASK [Gathering Facts] ******************************************************
ok: [localhost]

TASK [\Configurer motd pour ce serveur\] ************************************
changed: [localhost]

PLAY RECAP ******************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figure 69 : Exécution de motd.yml

Voici le rendu de ce fichier sur notre serveur :

```
[user-ansible@192 ansible]$ cat /etc/motd
# templates/motd.j2
Serveur: serveurAnsibleTdsi
FQDN: serveurAnsibleTdsi
Adresses IPv4:
  - 192.168.17.5
  - 192.168.239.133
  Adresses IPv6:
  * fe80::79b7:9e86:1000:5661
  * fe80::5748:b35c:8a77:bcec


[user-ansible@192 ansible]$
```

Figure 70 : Rendu du fichier motd

- ✓ **Personnalisation du .bashrc**

Dans la même logique que la mise en place du motd, nous allons aussi déployer une configuration spécifique du terminal (à l'aide du fichier invisible .bashrc) afin de personnaliser l'affichage du prompt sur le serveur. Cette amélioration a la même fonction que la précédente : assurer que l'on soit bien conscient d'être sur ce serveur et pas une autre machine lors de l'exécution de la commande. La stratégie de personnalisation retenue est relativement simple : nous allons configurer le terminal afin que le nom de l'utilisateur et du système soit d'une couleur différente. Comme c'est un serveur en production, nous allons donc utiliser la couleur rouge qui retiendra certainement l'œil.

Afin de mettre ceci en place, nous allons ajouter les lignes suivantes au fichier /root/.bashrc fourni par défaut :

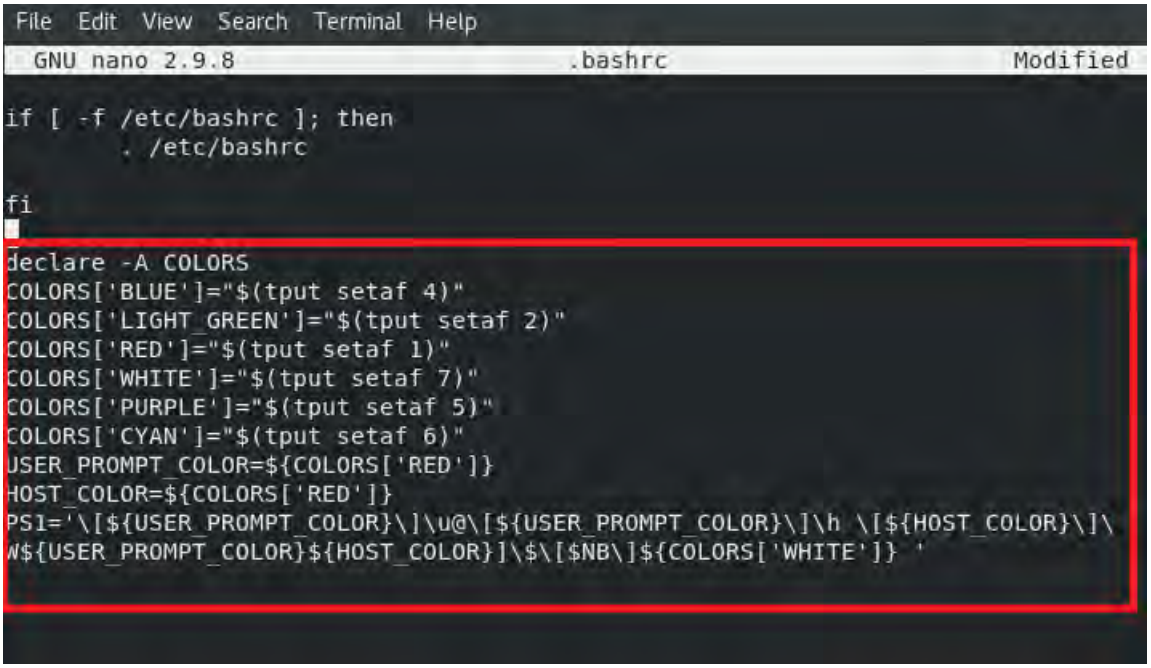

Figure 71 : Modification du fichier .bashrc

Pour faciliter la lecture de la configuration, on commence par définir un tableau associatif qui, comme son nom l'indique, va associer les appels à la commande tput (qui change la couleur du caractère imprimé à l'écran) au nom de la couleur.

Ensuite, on définit les variables USER_PROMPT_COLOR et HOST_COLOR qui vont nous permettre de configurer le contenu de la variable spéciale que le terminal utilise pour définir le prompt utilisateur.

On peut maintenant vérifier en se connectant avec root :

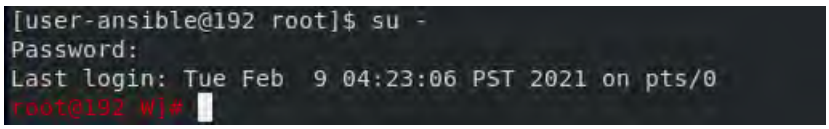

Figure 72 : Vérification de la personnalisation .bashrc

- ✓ **Mise en place du service SSH**

Jusqu'à maintenant, nous sommes partis du principe que nous avons un accès direct sur le serveur et que, après son installation, nous avons, à l'aide d'un clavier, d'un écran et éventuellement d'une souris, effectué les opérations nécessaires dessus. Afin de nous faciliter la maintenance et d'éviter d'avoir à physiquement accéder à la machine dès qu'une modification est nécessaire, nous allons mettre en place un accès à distance à notre serveur. À cette fin, nous allons donc installer et configurer un serveur SSH.

```
  GNU nano 2.9.8                    install-ssh.yml

---
- name: \Installation du serveur intranet\
  hosts: localhost
  gather_facts: false
  vars:
  tasks:
    - name: \Installer le serveur SSH\
      yum:
        name:
          - openssh-server
        state: latest
    - name: \Démarrer le serveur SSHD\
      service:
        name: sshd
        enabled: yes
        state: started
```

Figure 73 : Contenu du playbook d'installation de SSH

```
[user-ansible@192 ansible]$ sudo ansible-playbook --user user-ansible --become -
-ask-become-pass install-ssh.yml

BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [\Installation du serveur intranet\] ***********************************

TASK [\Installer le serveur SSH\] *******************************************
ok: [localhost]

TASK [\Démarrer le serveur SSHD\] *******************************************
ok: [localhost]

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

Figure 74 : Exécution du playbook d'installation de SSH

A partir du node bdd1 on va se connecter sur notre serveur via SSH :

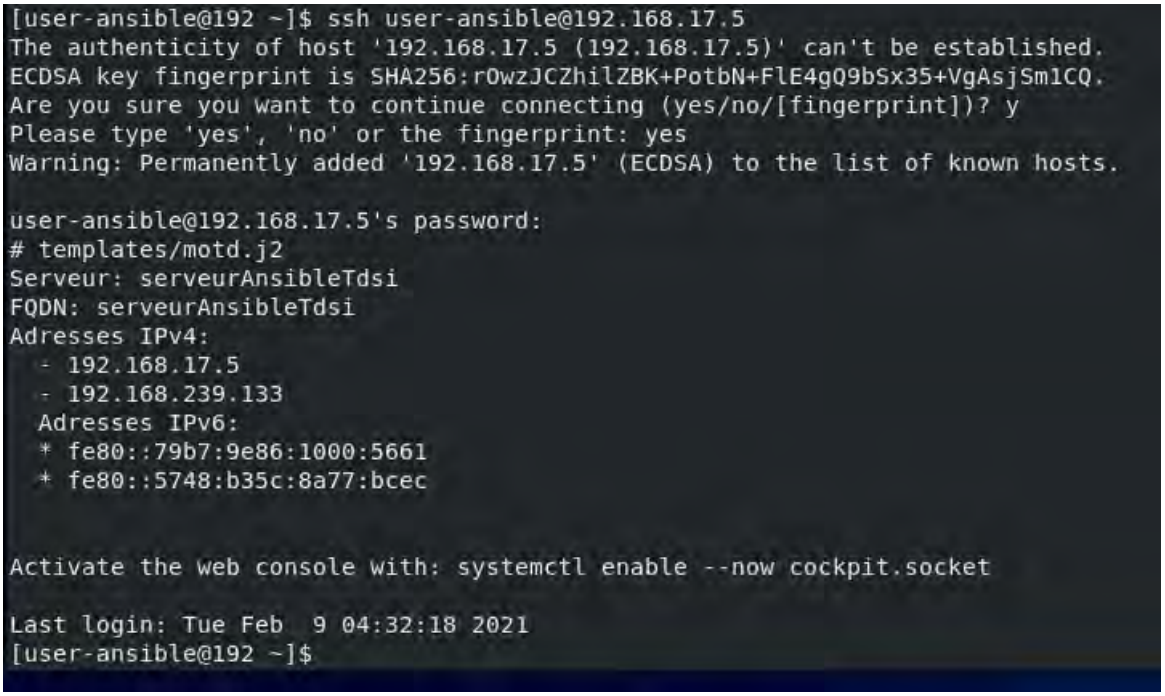

Figure 75 : Vérification de connexion SSH

✓ **Configuration des serveurs NTP**

Lorsqu'on est responsable d'un système informatique, l'une des pires situations qui puissent arriver est que l'horloge du système soit inexacte. Ceci a de nombreuses conséquences sur la bonne exécution du système. La plus simple de ces conséquences étant que les tâches planifiées (à l'aide de crontab, que nous utiliserons plus loin) ne s'exécutent pas au « bon moment ». Bref, il est essentiel qu'un serveur soit bien à l'heure et c'est pour cette raison qu'il existe une notion de « serveur de temps » utilisant NTP (« Network Time Protocol ») afin de régulièrement vérifier que les systèmes sont tous bien synchronisés entre eux.

```
 GNU nano 2.9.8                    install-ntp.yml

---
- name: \Installation du serveur intranet\
  hosts: localhost
  gather_facts: no
  vars:
  tasks:
    - name: \Installer ntpd\
      yum:
        name:
          - chrony
        state: latest
    - name: \Démarrer le serveur ntpd\
      service:
        name: chronyd
        enabled: yes
        state: started
```

Figure 76 : Contenu du playbook d'installation NTP

```
[user-ansible@192 ansible]$ sudo ansible-playbook --user user-ansible --become -
-ask-become-pass install-ntp.yml
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [\Installation du serveur intranet\] ***********************************

TASK [\Installer ntpd\] *****************************************************
ok: [localhost]

TASK [\Démarrer le serveur ntpd\] *******************************************
ok: [localhost]

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

Figure 77 : Exécution du playbook d'installation des serveurs NTP

On peut vérifier l'installation :

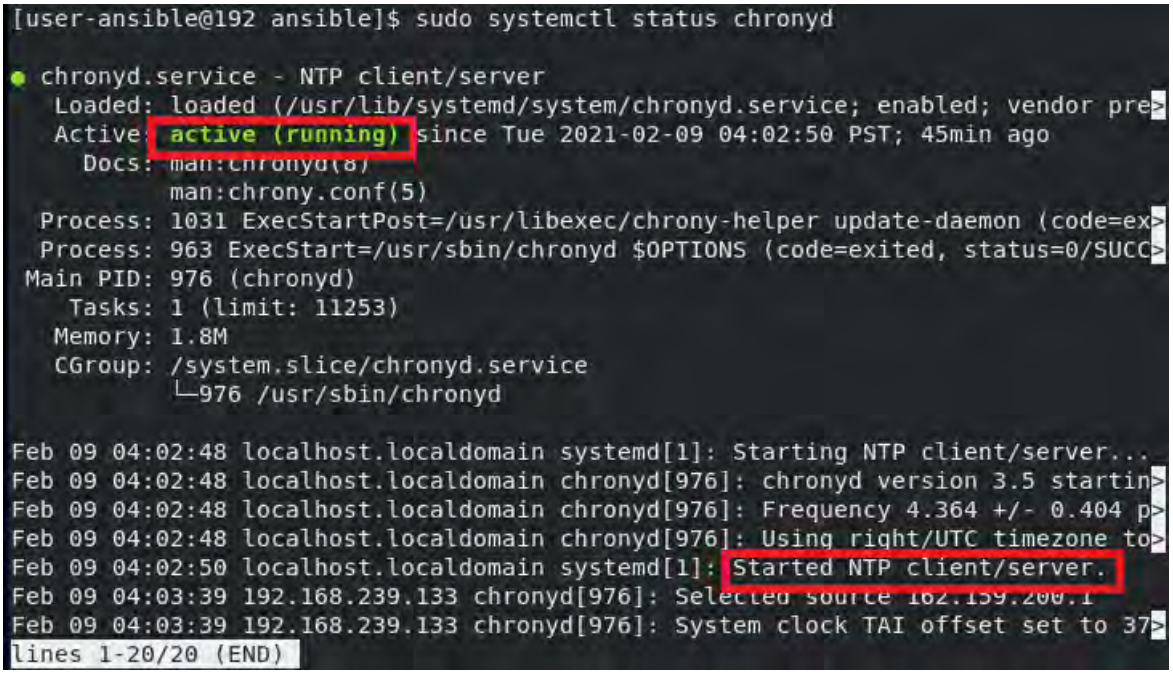

Figure 78 : Vérification de l''installation des serveurs NTP

On va maintenant vérifier si les serveurs NTP nous fournissent l'heure :

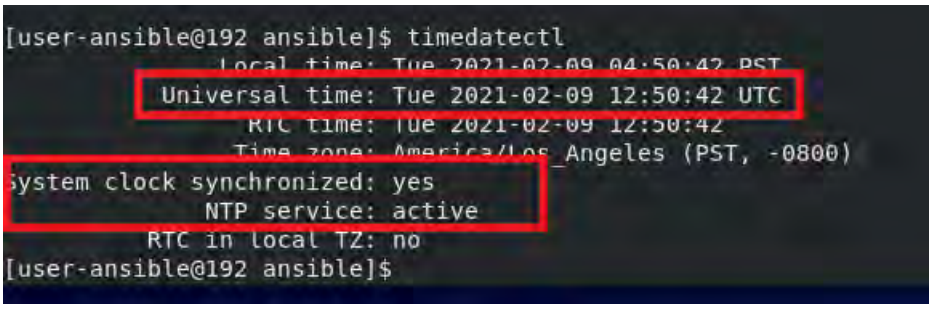

Figure 79 : Vérification de l'heure

✓ **Planification de l'exécution de taches à l'aide de CRON**

Pour nous faciliter la maintenance de ce serveur, nous allons maintenant automatiser un certain nombre de tâches récurrentes. Pour ce faire, nous allons demander au service Cron, un outil justement dédié à l'exécution de tâches planifiées, d'exécuter (au bon moment) les tâches suivantes :

■ Démarrage tous les jours du service apache ;

■ effectuer, une fois par semaine, une sauvegarde de l'ensemble de la base de données MediaWiki.

On va créer un petit playbook pour démarrer le service apache :

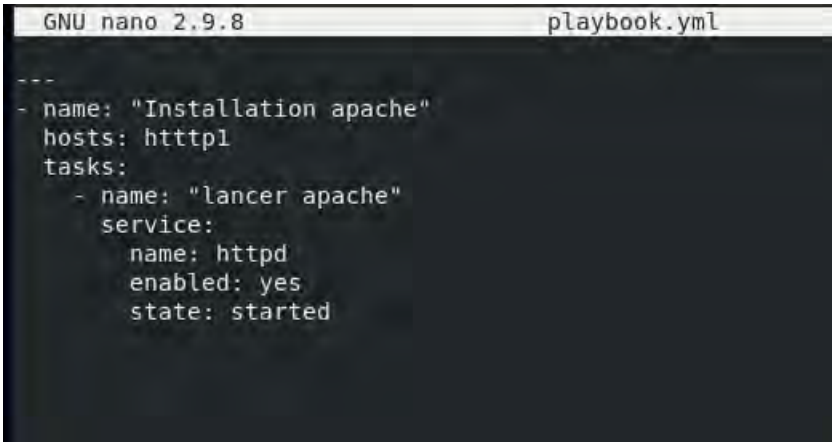

Figure 80 : playbook de démarrage du service apache

Passons au fichier pour automatiser cette tâche à l'aide de cron :

Figure 81 : playbook pour démarrer automatiquement apache

Figure 82 : Exécution du playbook d'automatisation de apapche

Quelques remarques sur l'extrait de « playbook » proposé ci-dessus :

- L'approche retenue ici est d'utiliser l'attribut special_time du module cron, qui se traduira par l'instruction **@daily** dans la table de tâches de Cron (ce qui laissera la liberté à Cron de décider quand exécuter cette tâche) ;
- Enfin, notons bien l'utilisation de l'option -C avec la commande ansible-playbook, qui garantit qu'Ansible n'exécute aucune opération sur le système, il se contentera, à la place, de simplement vérifier que la configuration du serveur est conforme (ou non) aux attentes.

Essayons de sauvegarder les données de MediaWiki.

En faisant la sauvegarde sans rendre indisponible pour un moment MediaWiki, on s'expose au risque de voir des modifications de pages qui ne sont pas sauvegardées ou, pire, de se retrouver avec un état incohérent à la reconstruction du wiki à partir de cette sauvegarde.

MediaWiki permet d'effectuer ceci de manière très simple, en définissant une variable au sein de son fichier de configuration LocalSettings.php :

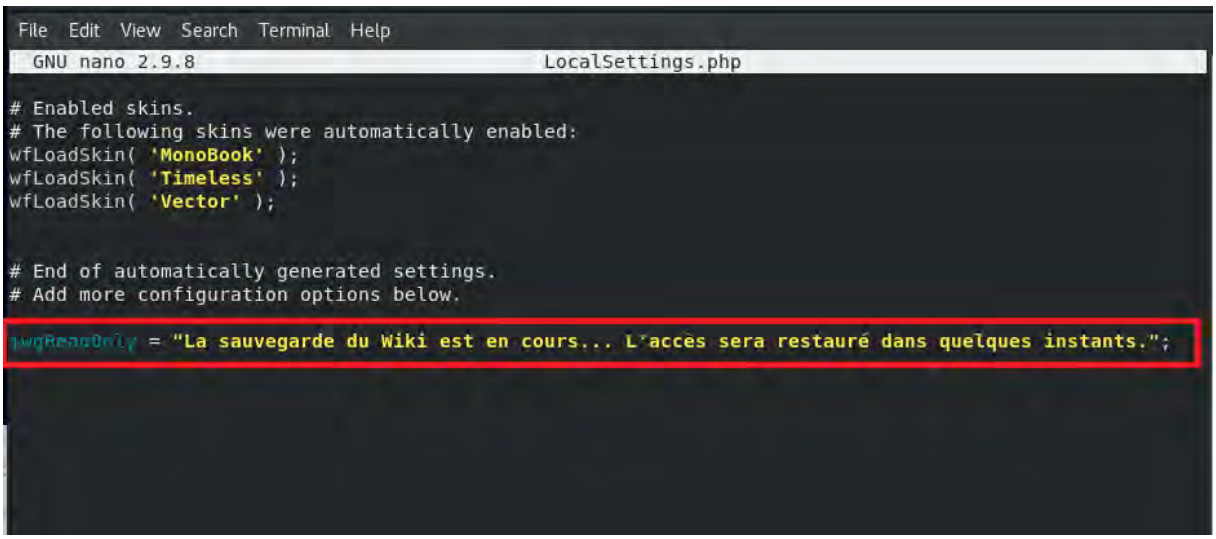

Figure 83 : Ajout de la variable dans le fichier de configuration MediaWiki

On écrit un script pour la sauvegarde :

```
  GNU nano 2.9.8                    backup-mediawiki-db.sh

#!/bin/bash
set -eo pipefail
readonly DB_HOST='localhost'
readonly DB_BACK_USER='root'
readonly DB_BACKUP_ARCHIVE=\/backup/mediawiki-db-$(date +%m-%d-%Y).gz\
readonly WIKI_MAINTENANCE_MODE=\\$wgReadOnly = 'Dumping Database
be restored shortly';\
readonly PATH_TO_LOCAL_SETTINGS='/var/www/html/w/LocalSettings.php'
mediawiki_maintenance_mode() {
  local pathToLocalSettings=${1:-${PATH_TO_LOCAL_SETTINGS}}
  sed -i -e \\$a${WIKI_MAINTENANCE_MODE}\ \${pathToLocalSettings}\}
mediawiki_normal_mode() {
  local pathToLocalSettings=${1:-${PATH_TO_LOCAL_SETTINGS}}
  set -i -e '/wgReadOnly/d' \${pathToLocalSettings}\}
trap mediawiki_normal_mode EXIT
mediawiki_maintenance_mode '/var/www/html/w/LocalSettings.php'
mysqldump -h\${DB_HOST}\ -u\${DB_BACK_USER}\ | gzip > \${DB_BACKUP_ARCHIVE}\
```

Figure 84 : Script de sauvegarde de la base de données

Nous pouvons maintenant utiliser Ansible afin de mettre place cette sauvegarde automatique sous forme de tâche planifiée et gérée par Cron :

```
  GNU nano 2.9.8                    backup-db.yml

---
- name: \Installation du serveur intranet\
  hosts: htttp1
  gather_facts: true
  vars:
    - path_to_db_backup_script: /root/backup-mediawiki-db.sh
  tasks:
    - name: \Installation des outils nécessaires\
      yum:
        name: gzip
        state: installed
    - name: \Installation du script de sauvegarde.\
      copy:
        src: /root/backup-mediawiki-db.sh
        dest: "{{ path_to_db_backup_script }}"
    - name: \Sauvegarde la base de données MediaWiki toutes les semaines\
      cron:
        special_time: weekly
        job: "{{ path_to_db_backup_script }}"

                          [ Read 19 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
```

Figure 85 : playbook de sauvegarde automatique de la base de données

```
[user-ansible@192 ansible]$ sudo ansible-playbook -i inventaire.ini --user user-ansi
ble --become --ask-become-pass backup-db.yml

BECOME password:

PLAY [\Installation du serveur intranet\] ***********************************

TASK [Gathering Facts] ******************************************************
ok: [htttp1]

TASK [\Installation des outils nécessaires\] ********************************
ok: [htttp1]

TASK [\Installation du script de sauvegarde.\] ******************************
changed: [htttp1]

TASK [\Sauvegarde la base de données MediaWiki toutes les semaines\] **********
[DEPRECATION WARNING]: The 'name' parameter will be required in future
releases.. This feature will be removed in version 2.12. Deprecation warnings
can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [htttp1]

PLAY RECAP ******************************************************************
htttp1                     : ok=4    changed=2    unreachable=0    failed=0    skipp
ed=0    rescued=0    ignored=0
```

Figure 86 : Exécution playbook de sauvegarde automatique de la base de données

5.4. Sécurisation du serveur

✓ **Configuration du pare-feu**

En fait, nous sommes partis du principe que le pare-feu était simplement désactivé, ce qui n'est pas un scénario improbable dans le contexte d'un serveur utilisé uniquement en interne. Cependant, nous allons maintenant nous y intéresser et utiliser le module **firewalld:** d'Ansible pour le configurer, car tout serveur interne n'est jamais qu'à un ou deux « rebonds » d'Internet et du monde extérieur. Notons néanmoins que, dans le contexte de notre serveur intranet, l'apport de ce pare-feu est assez minime.

Sa fonction ici sera surtout de réduire les chances d'attaques de type déni de service en refusant immédiatement des paquets à destination de services qui ne s'exécutent simplement pas sur la machine. Par essence, une vulnérabilité vient généralement d'un service qui s'exécute sur le système. Or, la plupart de temps, ces services doivent justement être accessibles et donc le port associé ouvert, ce qui rend de facto, la soi-disant protection fournie par le pare-feu nulle et non avenue (à moins de mettre en place un contrôle de flux élaboré à l'aide de cet outil, bien évidemment, mais ceci requiert une maîtrise avancée de l'outil). Les seuls services qui doivent être accessibles à distance sur le serveur sont Apache et le serveur SSH. Ceci dit, dans la prochaine section, nous allons mettre en place HTTPS au sein de la

configuration de Apache. Configurons donc directement le pare-feu afin de n'autoriser que l'accès aux flux HTTPS et HTTP :

```
File  Edit  View  Search  Terminal  Help
  GNU nano 2.9.8                    conf-firewall.yml

---
- name: \Installation du serveur intranet\
  hosts: localhost
  gather_facts: false
  tasks:
    - name: start firewalld
      service:
        name: firewalld
        state: started
        enabled: yes
    - firewalld:
        service: "{{ item }}"
        permanent: yes
        state: enabled
      loop:
        - https
        - http
        - ssh
```

Figure 87 : Configuration du pare-feu

```
[user-ansible@192 ansible]$ sudo ansible-playbook --user user-ansible --become --ask
-become-pass conf-firewall.yml
BECOME password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [\Installation du serveur intranet\] ***************************************

TASK [start firewalld] **********************************************************
ok: [localhost]

TASK [firewalld] ****************************************************************
changed: [localhost] => (item=https)
changed: [localhost] => (item=http)
ok: [localhost] => (item=ssh)

PLAY RECAP **********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipp
ed=0    rescued=0    ignored=0
```

Figure 88 : Exécution de la configuration du pare-feu

- ✓ **Sécurisation de Apache**

Au bout du compte, la seule application réellement exposée sur notre serveur est le serveur web, soit Apache. C'est donc ce flux de données HTTP (entre Apache et ses clients) dont nous devons assurer la sécurité et la confidentialité.

- Configuration de Apache pour utiliser HTTPS

De nos jours, l'utilisation de la version chiffrée du protocole HTTP est devenue systématique, si ce n'est à des fins d'assurer la confidentialité des échanges. C'est un peu regrettable en termes d'impact sur l'environnement (le chiffrement et le déchiffrement augmentent, à coup sûr, la consommation d'énergie des systèmes, bien souvent pour rendre inaccessible une information de toute manière publique). Nous allons donc mettre en place ce protocole chiffré, ce qui requiert l'installation d'un certificat sur le serveur. Afin de rester simples et didactiques, nous allons utiliser un certificat signé manuellement que nous allons créer nous-mêmes.

Afin de générer ce certificat, nous allons utiliser la commande openssl :

```
[user-ansible@192 mediawiki]$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl
/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
[sudo] password for user-ansible:
Generating a RSA private key
.............+++++
.+++++
writing new private key to '/etc/ssl/private/apache-selfsigned.key'
req: Can't open "/etc/ssl/private/apache-selfsigned.key" for writing, No such file or directory
[user-ansible@192 mediawiki]$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl
/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
Generating a RSA private key
.....................................................................+++++
...+++++
writing new private key to '/etc/ssl/apache-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:SN
State or Province Name (full name) []:Dakar
Locality Name (eg, city) [Default City]:Dakar
Organization Name (eg, company) [Default Company Ltd]:TDSI
Organizational Unit Name (eg, section) []:TDSI
Common Name (eg, your name or your server's hostname) []:tdsi.com
Email Address []:cherifkasse0@gmail.com
[user-ansible@192 mediawiki]$
```

Figure 89 : Génération de certificats

Ceci fait, modifions le fichier de configuration pour y ajouter l'utilisation de SSL, mais aussi changer le port par défaut (443 au lieu de 80) :

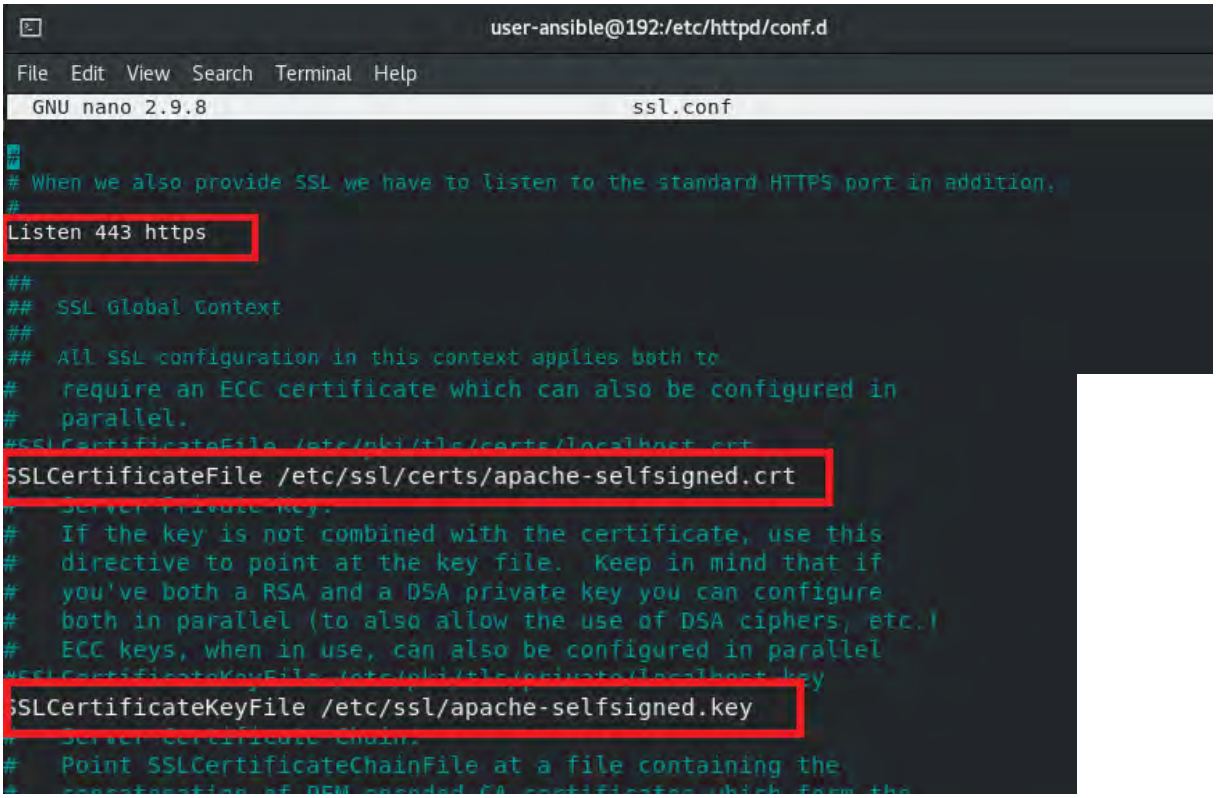

Figure 90 : Modification du fichier ssl.conf

Vérifions cela en essayant de se connecter à un site utilisant HTTPS :

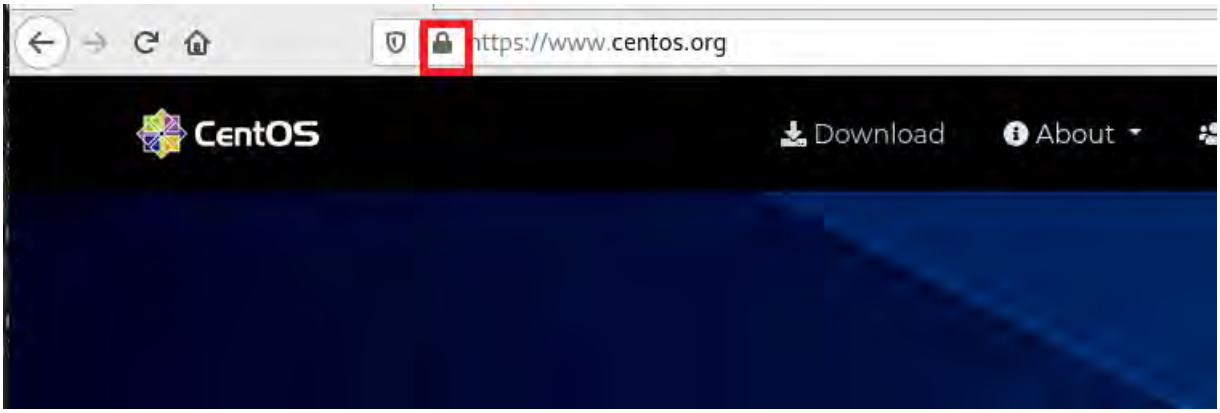

Figure 91 : Vérification de configuration HTTPS

Commençons par limiter le nombre de protocoles supportés par Apapche. Dans sa configuration par défaut, le serveur HTTP a déjà pris soin de supprimer les protocoles SSLv2 et SSLv3. Néanmoins, les protocoles TLSv1.0 et v1.1 qui sont de plus en plus abandonnés par la plupart des navigateurs sont encore autorisés.

En outre, comme il s'agit d'un intranet, nous sommes en mesure de savoir quels navigateurs seront installés sur les postes utilisateurs des employés de la PME et de nous assurer ainsi que seuls les protocoles choisis seront utilisés. Conformément à notre stratégie, nous interdisons

des protocoles grâce à une connaissance et une maîtrise de notre infrastructure, et non, « par défaut ». Ainsi, nous sommes en mesure de restreindre le nombre de protocoles disponibles lors d'un échange entre Apache et ses clients à la liste suivante :

```
ssl_protocols TLSv1.2 TLSv1.3
```

Figure 92 : Limitation de protocoles utilisé par Apache

Par défaut, Apache laisse le client déterminer la suite de chiffrement qui sera utilisée lors de l'échange. Ceci signifie malheureusement que si le navigateur client sélectionne une suite de chiffrement peu robuste, ou même compromise, la confidentialité de la communication ne sera pas garantie.

Afin d'éviter ce risque, nous allons configurer Apache pour qu'il impose la suite de chiffrement lors de la connexion d'un nouveau client :

```
ssl_prefer_server_ciphers on
```

Figure 93 : Imposition du chiffrement

- ✓ **Sécurisation du serveur avec SELinux**

SELinux est un mécanisme de sécurité spécifique aux distributions de Red Hat (et dérivées, telles que CentOS). C'est une protection très appréciable, qui tend à limiter, autant que possible, la marge de manœuvre d'un acteur malveillant cherchant à compromettre le système. En quelques mots, SELinux s'assure que les activités des utilisateurs et des logiciels sur le système sont conformes aux attentes.

Si elles diffèrent de ces attentes (par exemple, un serveur web qui tente subitement de modifier le contenu de la partition de démarrage /boot), SELinux intervient et interdit l'opération. Ainsi, même si l'on a réussi à exploiter une faille d'un logiciel, on ne peut plus facilement s'en servir pour affecter le reste du système.

Nous allons créer un playbook dont l'exécution va assurer que SELinux est bien activé :

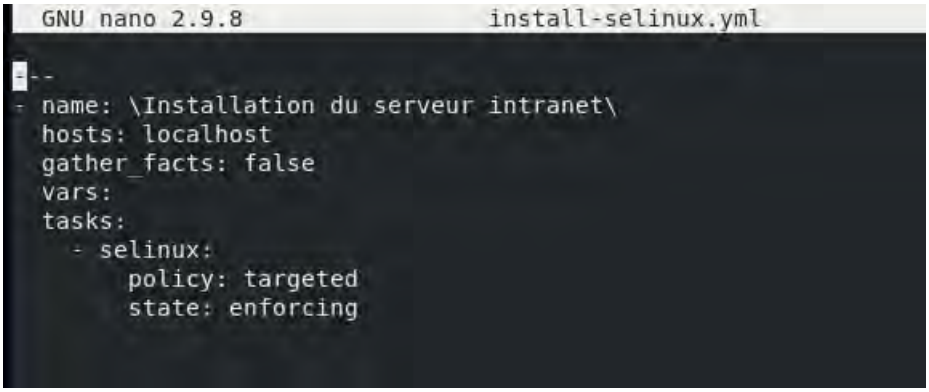

Figure 94 : Activation de SELinux

Lançons le playbook :

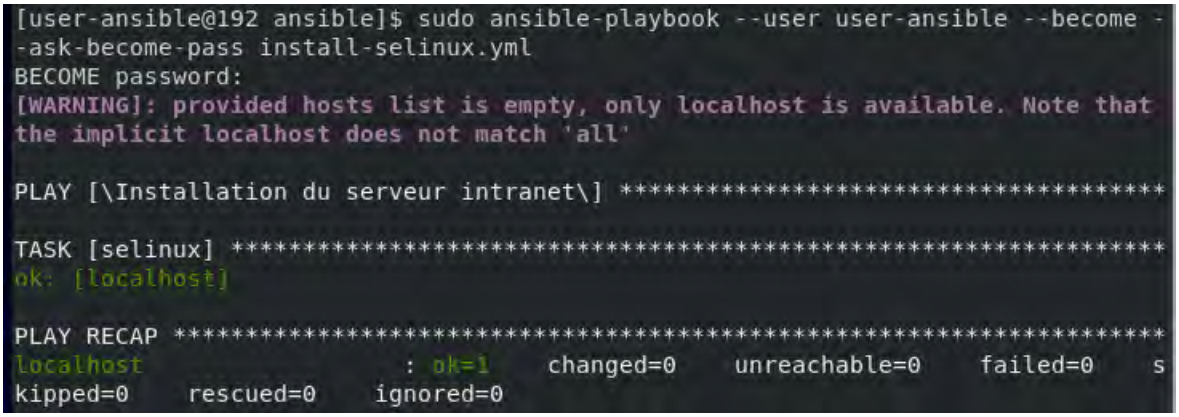

Figure 95 : Lancement du playbook d'activation de SELinux

CONCLUSION GENERALE ET PERSPECTIVES

Dans ce rapport nous avons été amené à présenter le déploiement d'un intranet Linux dans une PME avec Ansible et CentOS.

Après avoir parlé des généralités sur les réseaux, et avoir fait une synthèse sur ce qu'est un intranet, nous avons abordé l'automatisation avec l'outil Ansible pour les déploiements automatisés et les configurations systèmes ainsi que la sécurisation de notre intranet surtout avec la configuration de pare-feu et l'utilisation de SELinux.

La réalisation de ce projet a été très enrichissante en nous familiarisant avec le monde du réseau, de la sécurité, des systèmes Linux et surtout de l'outil Ansible.

Enfin malgré certaines contraintes, les objectifs fixés ont été atteints.

Comme perspectives, nous allons essayer de mettre en place un playbook Ansible permettant de détecter de manière automatique certains types d'attaques ou des comportements malveillants, de s'habituer avec l'outil Ansible Tower (Interface graphique et API REST pour Ansible).

BIBLIOGRAPHIE

1. etude-et-mis-en-place-dun-reseau-intranet-sous-linux , récupéré sur https://www.rapport-gratuit.com/etude-et-mis-en-place-dun-reseau-intranet-sous-linux/

2. Deployez-un-intranet-linux-pour-votre-pme, récupéré sur https://boutique.ed-diamond.com/module/dmdpdfstamper/read?id_order_detail=106862

WEBOGRAPHIE

1. https://www.mediawiki.org/wiki/MediaWiki , 12-02-2021
2. https://mariadb.org/ , 02-02-2021
3. https://httpd.apache.org/ , 02-02-2021
4. https://openclassrooms.com/fr/courses/2035796-utilisez-ansible-pour-automatiser-vos-taches-de-configuration/6371875-preparez-la-communication-avec-les-nodes,14-02-2021
5. https://devopssec.fr/article/avantages-fonctionnement-ansible , 28-01-2021
6. http://blog.projixi-europe.com/2017/04/05/english-open-source-automatisation-tools/#.YByNXuj7TDd , 25-01-2021
7. https://www.veritis.com/blog/chef-vs-puppet-vs-ansible-comparison-of-devops-management-tools/ , 10-02-2021
8. http://blog.projixi-europe.com/2017/04/05/english-open-source-automatisation-tools/#.YByFYej7TDc , 31-01-2021
9. https://africa.visiativ.com/les-avantages-d-un-intranet/ , 07-02-2021
10. https://www.vtx.ch/fr/blog/pourquoi-travaillons-nous-avec-la-distribution-linux-centos#:~:text=tout%20d'abord%2C%20ce%20syst%C3%A8me,de%20ce%20syst%C3%A8me%20sont%20applicatives , 07-02-2021
11. https://www.appvizer.fr/magazine/collaboration/reseau-social-interne/intranet-entreprise
12. https://techtoc.tv/event/138/cloud--entreprise-collaborative-et-gouvernance/intranet-s--et-reseaux-sociaux-d-entreprise/l-intranet-dans-les-pme-pmi--quelles-solutions-privilegier ,15-02-201
13. https://www.vmware.com/fr/topics/glossary/content/it-automation.html
14. https://www.redhat.com/en/blog/adventures-ansible-challenges-solutions-and%E2%80%A6-subversion ,12-02-201
15. https://www.memoireonline.com/02/12/5299/tude-du-projet-d-implementation-d-un-intranet-collaboratif-dans-une-entreprise-multi-sites-sous-l.html ,08-02-2021
16. https://www.anthedesign.fr/erp-intranet/intranet-entreprise/ ,29-01-2021
17. https://devopssec.fr/article/introduction-cours-complet-ansible , 04-02-2021
18. https://www.redhat.com/fr/topics/automation/whats-it-automation , 31-12-2020