

## **PREMIERE PARTIE :**

### **INTRODUCTION**

De nos jours, nous voyons bel et bien comment le monde informatique ne cesse de se développer.

Dans le cadre de notre projet, nous n'allons uniquement pas parler de la problématique majeure mais nous allons également proposer une piste de solution afin de répondre au besoin rencontré lors d'une mise-à-niveau. Il est important de savoir qu'actuellement la sécurité demeure un élément majeur permettant le bon fonctionnement d'un système informatique. Effectuer une mise-à-jour depuis internet pourrait engendrer une faille de sécurité et compromettre le bon fonctionnement d'une structure informatique. Afin d'avoir le contrôle total de différentes mises-à-jour, il est serait plus judicieux de mettre en place un référentiel local pouvant contenir tous les données nécessaires et ce, en fonction du besoin. Cela renforce non seulement la fiabilité et l'origine de données (paquets, images, etc...) présents dans le référentiel local mais limite les risques de compromission du système au premier niveau.

## DEUXIEME PARTIE :

### DIMENSIONNEMENT, OS ET IMPORTANCE

Dans le cadre de cette deuxième partie, nous n'allons pas seulement lister les différents prérequis pour les différents OS de notre infrastructure mais aussi donner l'importance des outils à déployer.

ADRESSE RESEAU : 192.168.9.1					
ID	MACHINES CLIENTES (A) / SERVEURS (B)	OS / Version	ADRESSE IP	DIMENSIONNEMENT (RAM + CPU + TAILLE DISQUE)	IMPORTANCE
1	Windows serveur (B)	Windows server 19	192.168.9.130	RAM : 2GO / CPU : 2 / DISQUE: 60GB	AD et création d'utilisateur LDAP qui sera intégrer à AWX ( authentification sécurisée)
2	AWX (B)	ubuntu-22.04.1-desktop-amd64	192.168.9.109	RAM : 12GO / CPU : 2 / DISQUE: 40GB	Planification et supervision des tâches
3	GITEA (B)	ubuntu-22.04.1-desktop-amd64	192.168.9.110	RAM : 4GO / CPU : 2 / DISQUE: 25GB	Versionning de nos configurations, stockage de nos différents fichiers et synchronisation avec AWX
4	MARC : REFERENTIEL LOCAL (B)	ubuntu-22.04.1-desktop-amd64	192.168.9.120	RAM : 6GO / CPU : 2 / DISQUE: 35GB	Gestionnaire centrale de nos mises-à-jour
5	ANSIBLE (B)	ubuntu-22.04.1-desktop-amd64	192.168.9.115	RAM : 6GO / CPU : 2 / DISQUE: 25GB	Machine test pour pousser les configurations sur les machins clientes
6	CLIENT 1 (A)	ubuntu-22.04.1-desktop-amd64	192.168.9.132	RAM : 2GO / CPU : 2 / DISQUE: 20GB	Machine test receptionnant les playbook ansible , intégration dans AWX
7	CLIENT 2 (A)	ubuntu-22.04.1-desktop-amd64	192.168.9.133	RAM : 2GO / CPU : 2 / DISQUE: 20GB	Machine test receptionnant les playbook ansible , intégration dans AWX
CAPACITE TECHNIQUE DE LA MACHINE					
HOTE:					
-----					
----					
1. MEMOIRE RAM : 24 GO ( Fréquence : 3200 MHZ)					
2. DISQUE : SSD					
3. TAILLE : 1 TO					
4. CPU : intel core I3					
© Ngandu Jean-Marc 2025					

Image 1 : image sur le dimensionnement des équipements de notre infrastructure

Au niveau de la page suivante, nous allons décrire brièvement les différents outils utilisés avec capture d'écran.

## 1. AWX

Comme nous le savons bien, mettre à jour un système demande non seulement une certaine flexibilité, mais exige aussi l'utilisation de certains outils qui pourraient non seulement faciliter la tâche mais aussi suivre correctement les différents logs, l'évolution de chaque pc client intégré lors du processus de mise-à-jour, d'exécution de différents Playbook, etc. Dans le cadre de notre projet nous allons utiliser AWX qui est un outil open source et qui sera déployer avec Kubernetes (cluster MINIKUBE) afin de planifier et synchroniser nos différentes tâches.

Prérequis techniques :

1. Linux (gestion de permissions, déplacement dans le répertoire, SSH, partage de fichier en réseau, gestion d'utilisateurs et groupes)
2. Docker
3. Les bases de kubernetes (installation minikube, commande administrative, etc...)
4. Configuration de l'adresse IP
5. Dimensionnement de la machine
6. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795

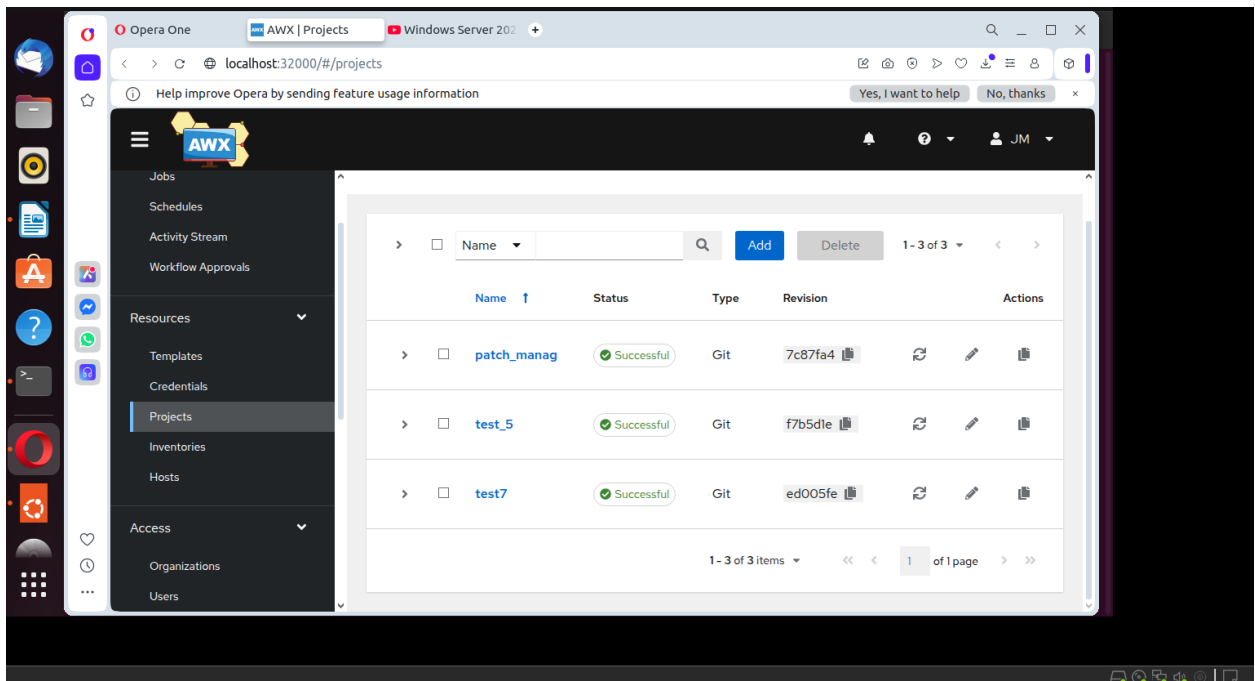


Image 2 : présentation de l'interface graphique AWX via l'onglet projet

## 2. GITEA

Suite à l'évolution constante de différentes technologies et de certaines configurations dans notre monde actuel, il serait primordiale de savoir quand, comment et où stocker les différents projets (configuration, différentes versions etc...) sur ce, nous allons utiliser Gitea qui est une version de Git en mode graphique afin de permettre le versionning de nos différents configurations, stocker nos différents projets. Il sera déployer grâce à docker et sera intégré à AWX.

Prérequis techniques:

1. Docker
2. Linux (gestion de permissions, configuration de fichiers, SSH, gestion de ports, gestion d'utilisateurs et groupes, etc...)
3. Git
4. NGINX ou TREAIFIK pour le reverse proxy
5. Ansible pour configurer et partager les configurations en réseau
6. Vs code
7. Configuration de l'adresse IP
8. Dimensionnement de la machine
9. Génération des certificats avec OpenSSL et création du domaine
10. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795

Afin de mieux voir le fichier d'installation de gitea, veuillez cliquer sur le lien github ci-après :

[https://github.com/jeanmarctsh/gitea\\_deploy/tree/gitea](https://github.com/jeanmarctsh/gitea_deploy/tree/gitea)

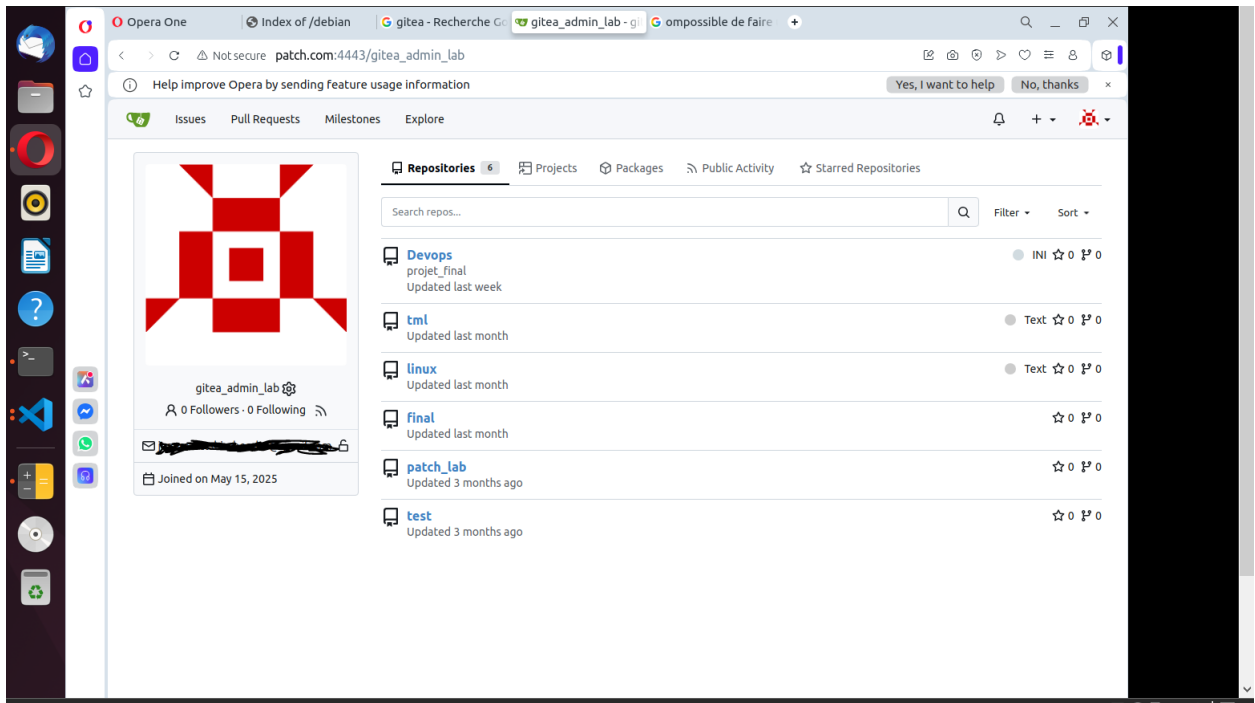


Image 3 : présentation de l'interface web Gitea avec différents repository

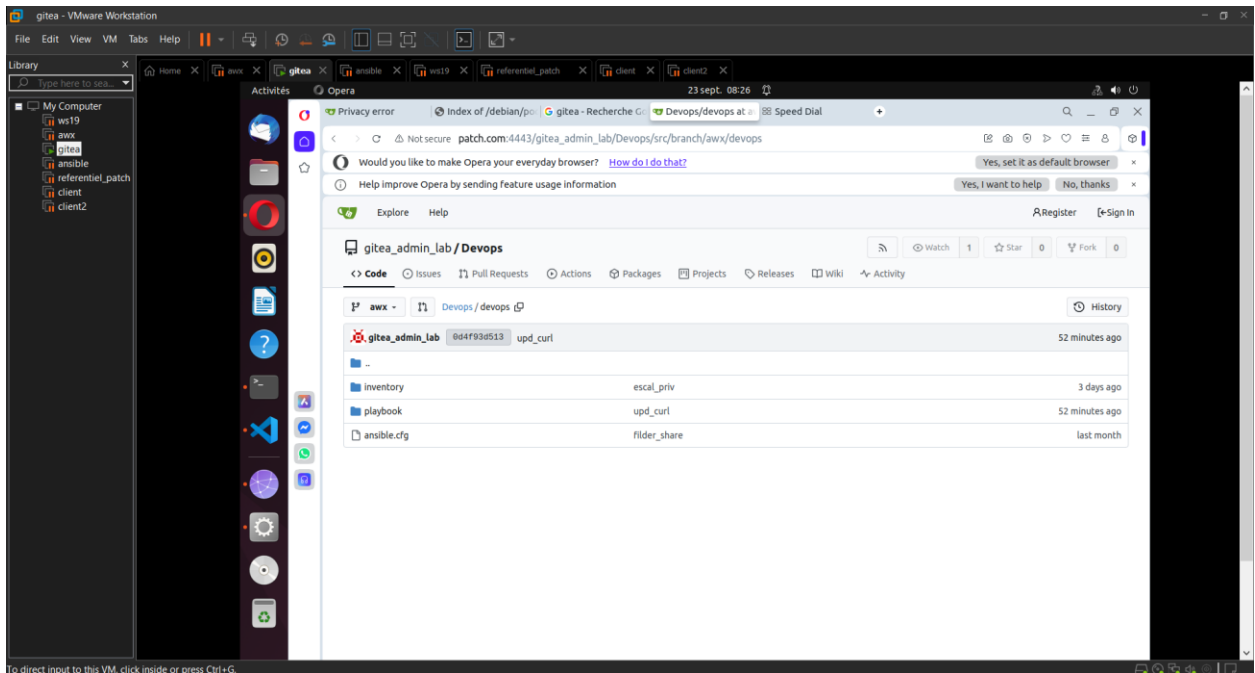


Image 4 : Présentation de l'arborescence de notre projet dans l'interface Gitea

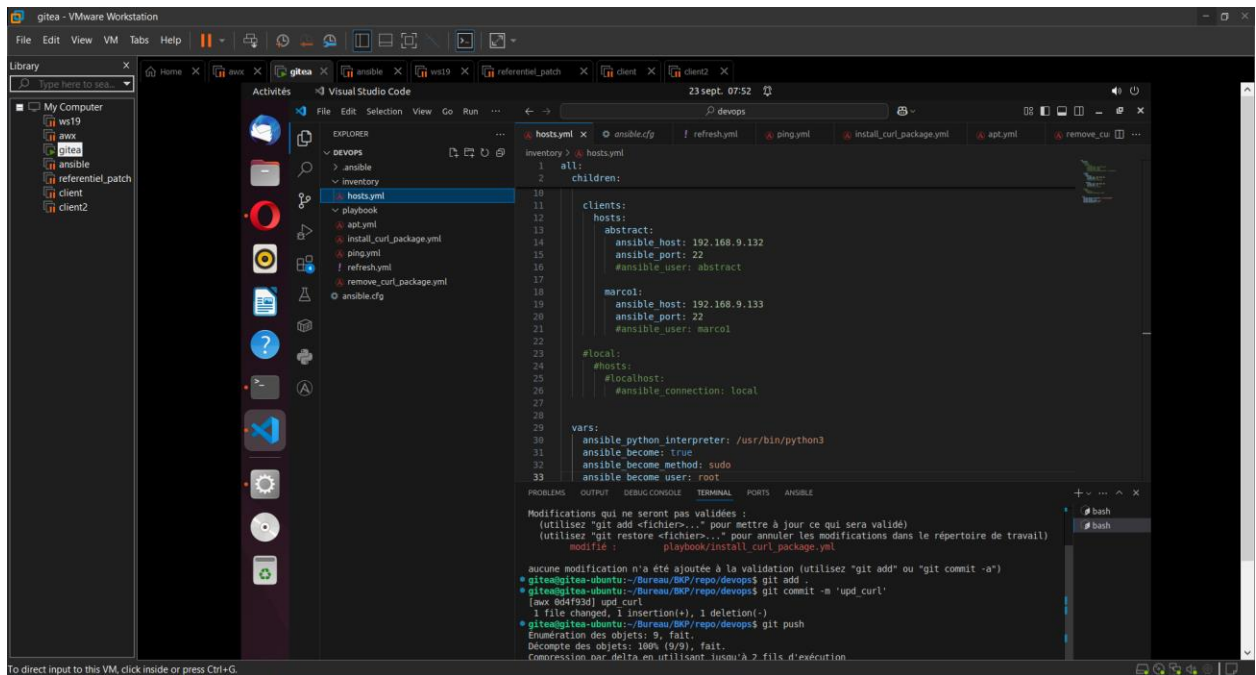


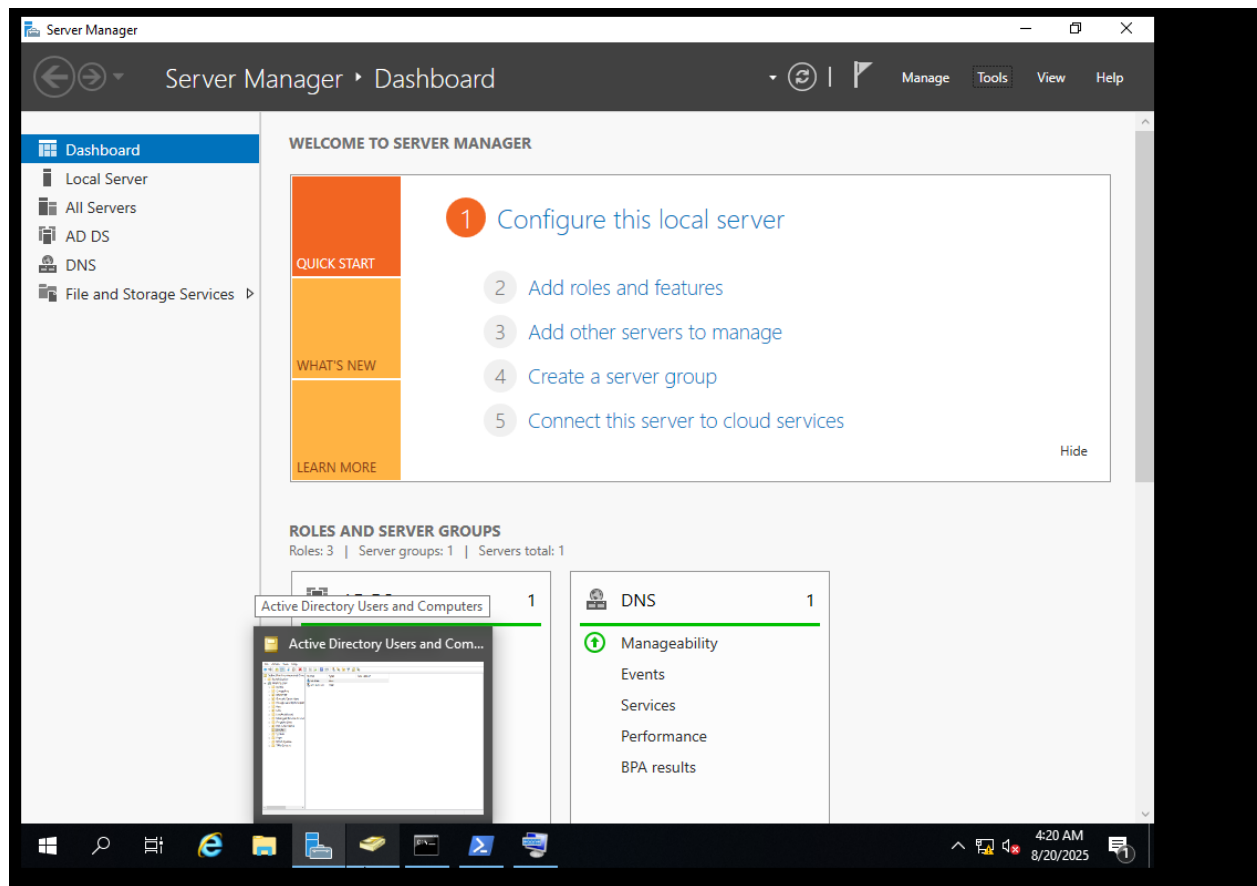
Image 5 : Présentation de l'inventaire Ansible au format yaml

### 3. WINDOWS SERVER 19

Afin d'assurer une sécurité pour les utilisateurs LDAP qui seront créés puis intégrer dans AWX nous allons utiliser Windows server 19 pour la gestion d'authentification, gestion de groupes et d'utilisateurs. Mais aussi pour la gestion de différentes permissions.

Prérequis techniques :

1. Installer Windows server 19
2. Configuration du nom du domaine et de l'adresse IP
3. Configuration d'Active Directory
4. Création de groupes, utilisateurs, unité d'organisation
5. Gestion de permissions
6. Dimensionnement de la machine
7. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795



*Image 6 : Présentation du Dashboard de Windows server 19*

#### 4. ANSIBLE

De nos jours, exécuter les tâches manuellement (configuration) engendre souvent certaines erreurs et lorsque nous faisons face à de centaines d'équipements, les maintenir manuellement devient très compliqué à la longue et prend plus de temps, c'est la raison pour laquelle il est important de savoir quoi automatisé. Il faudra procéder de la manière ci-après :

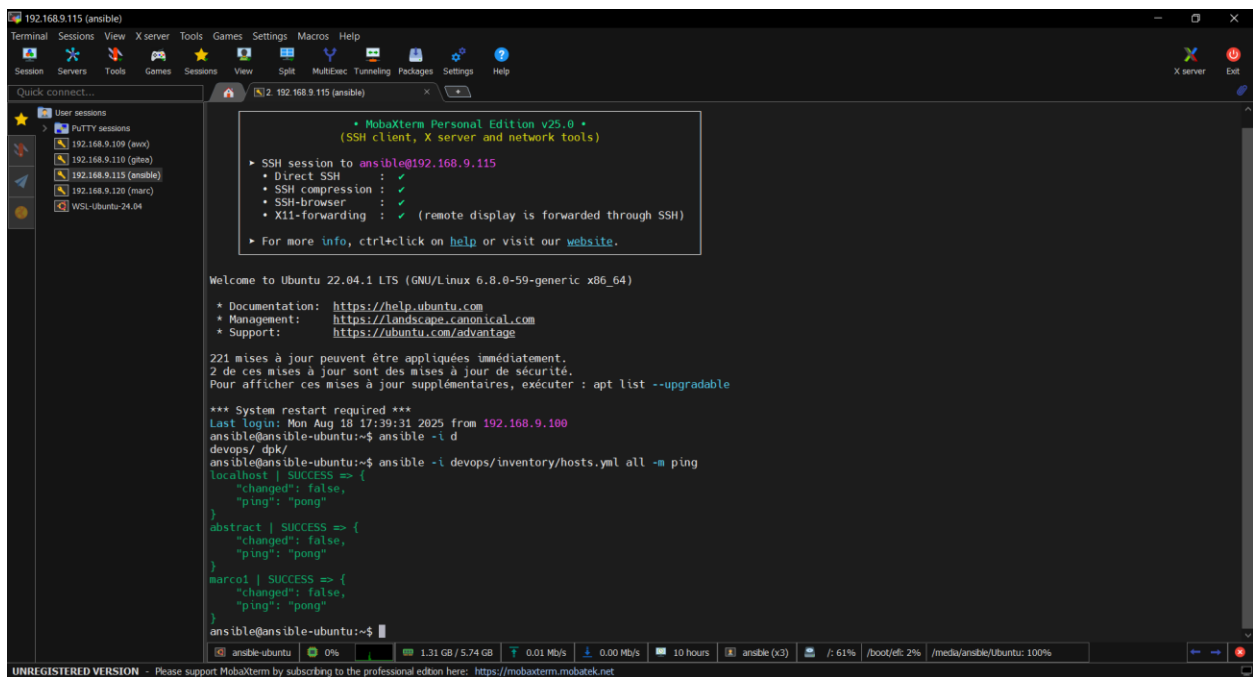
1. Installer ansible à partir du lien officiel
2. Créer le fichier de configuration à partir du lien officiel également
3. Créer dossier pour configurer le fichier d'inventaire (format INI ou format YAML)
4. Créer un dossier pour le Playbook
5. Installer SSH et créer les paires de clé (plus rapide et sécurisées)
6. Copier la clé publique vers les hôtes cibles

Lien officiel d'ansible :

<https://docs.ansible.com>

## Prérequis techniques :

1. Linux (gestion de permissions, déplacement dans le répertoire, commandes administratives, SSH, partage de fichier en réseau, compression du fichier ou du dossier, gestion d'utilisateurs et groupes, etc...)
2. Configuration de l'adresse IP
3. Dimensionnement de la machine
4. Git
5. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795



*Image 7: Test de connectivité d'ansible vers les machines clientes*



```

ansible@ansible-ubuntu:~$ ansible -i devops/inventory/hosts.yml all -m ping
localhost | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
abstract | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
marcol | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
ansible@ansible-ubuntu:~$ ansible-playbook -i devops/inventory/hosts.yml devops/playbook/apt.yml --syntax-check
playbook: devops/playbook/apt.yml
ansible@ansible-ubuntu:~$ ansible-playbook -i devops/inventory/hosts.yml devops/playbook/apt.yml --check -K
BECOME password:

PLAY [update] *****
TASK [Gathering Facts] *****
ok: [abstract]
fatal: [marcol]: FAILED! => {"msg": "Incorrect sudo password"}
TASK [apt] *****
changed: [abstract]
TASK [print msg] *****
ok: [abstract] => {
  "msg": "okokok"
}
PLAY RECAP *****
abstract      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
marcol        : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
ansible@ansible-ubuntu:~$

```

Image 8: check de vérification pour la syntaxe du playbook sur l'hôte abstract

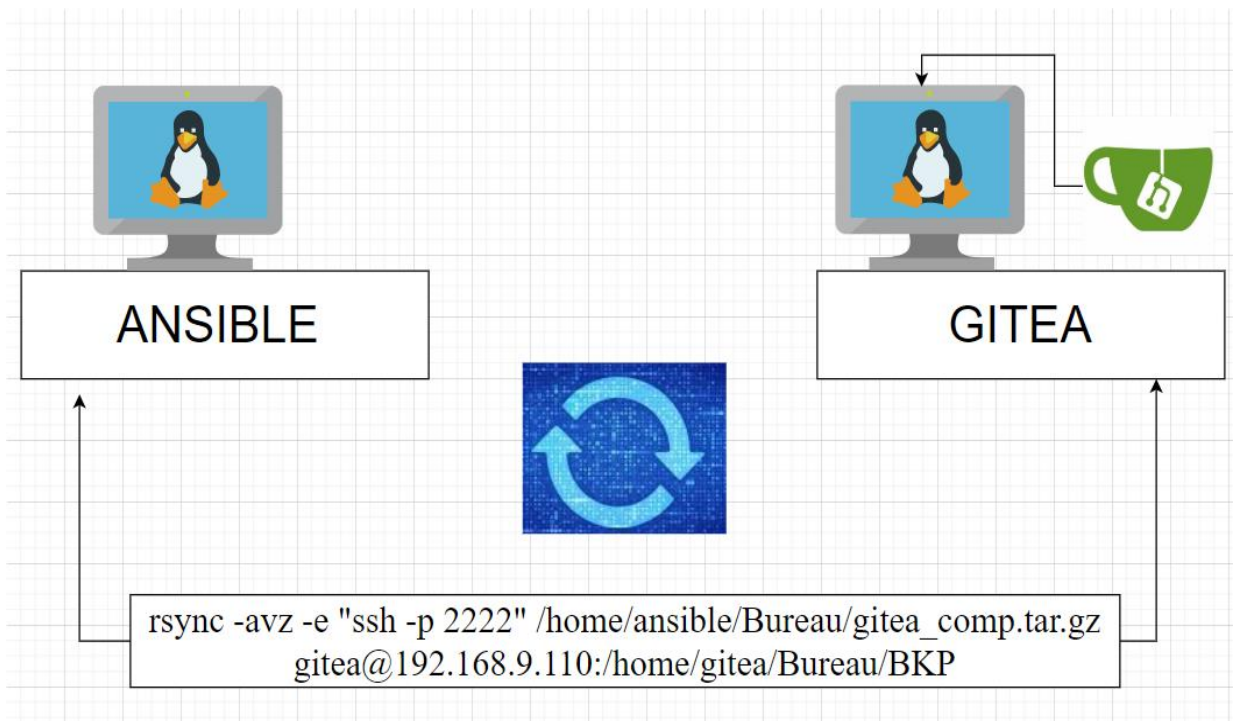


Image 9 : Partage du fichier entre ansible et Gitea



## 5. CLIENT 1 ET CLIENT 2

Les deux ordinateurs serviront de machines clientes. C'est sur ces deux machines que les différentes configurations seront poussées, que ce soit avec Ansible en mode console ou ansible en mode graphique (précisément avec AWX).

Prérequis techniques :

1. Configuration de l'adresse IP
2. Linux (SSH, commandes administratives, gestion d'utilisateurs et groupes)
3. Dimensionnement de la machine
4. Utilisateur de service
5. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795

ID	ORDINATEUR	USERNAME	IP	UTILISATEUR DE SERVICE
1.	Client1	abstract	192.168.9.132	ansi
2.	Client2	marc	192.168.9.133	ansi

Tableau 1 : Description des machines clientes

## 6. REFERENTIEL LOCAL

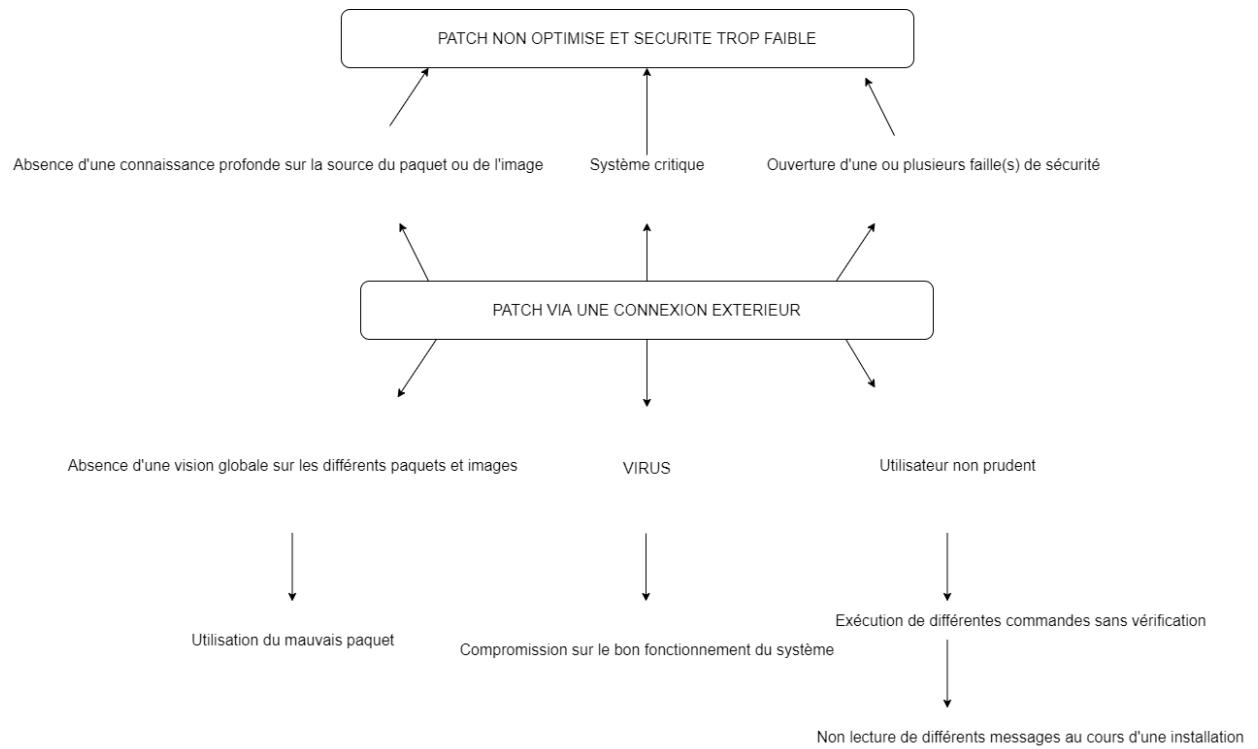
Mettre à jour un système n'a jamais été une tâche aisée, de nombreuses personnes pensent qu'il suffit de lancer une simple commande afin d'avoir toutes les mises-à-jour nécessaires, ce qui n'est malheureusement pas le cas. Comme nous l'avons bien dit ci-haut cela engendre généralement une faille de sécurité au sein d'un système informatique. Raison pour laquelle il est fortement recommandé de mettre en œuvre un référentiel local afin d'avoir une vision, un contrôle total de différents paquets ou images. Et cela permet de faire les différentes mises-à-jour en interne de façon plus structurée et contrôlée. Les hôtes pourront joindre le dépôt local afin d'effectuer toutes les mises-à-jour nécessaires sans se connecter sur internet. Et cela renforce une certaine sécurité de base.

## Prérequis techniques :

1. Dimensionnement de la machine
2. Configuration de l'adresse IP
3. Scripting basique en bash
4. Apache2 (configuration de fichiers, règle de filtrage, etc)
5. Génération des certificats SSL et création de domaine local
6. Connaissance sur la génération de la clé GPG
7. Connaissance sur le fonctionnement du logiciel reprepo
8. Exportation de dépôts .deb dans le référentiel local
9. Linux (Gestion de permissions, navigation dans les différents répertoires, gestion de paquets, vérification de dépendances, vérification de la source officielle, compression et décompression de certains fichiers, gestion d'utilisateurs et groupes)
10. Savoir prioriser le dépôt pour les mises-à-jour locale
11. Connaissance du répertoire /home/username/etc/apt
12. Hyperviseur de type2 : VMware-workstation-full-17.5.0-22583795

Sur ce, nous allons faire une présentation graphique de la problématique majeure de notre projet, afin de dégager les grandes lignes dans le but de mieux solutionner notre problème. Afin de mieux contextualiser cela, nous allons utiliser la notion :

1. Arbre à problème pour dégager la problématique majeur et ses différentes branches
2. Arbre à solution pour proposer les différentes pistes de solutions



© Ngandu Jean-Marc 2025

*Image 12 : Présentation de l'arbre à problème de notre problématique*

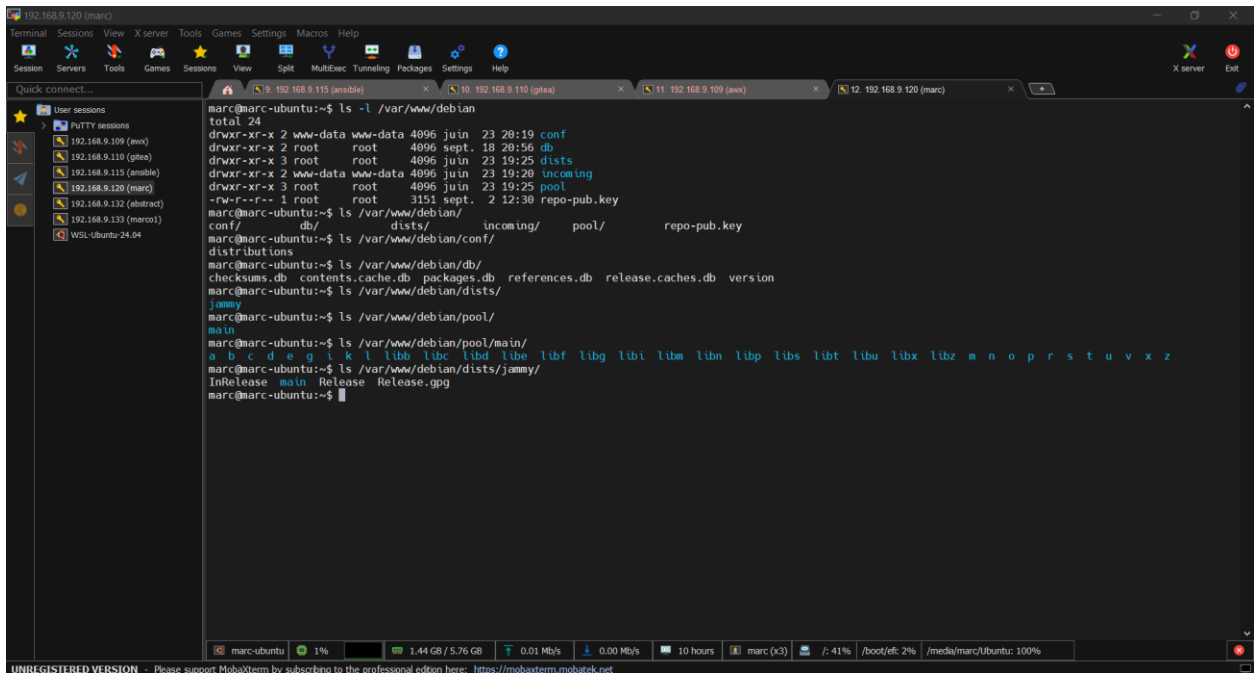


Image 13 : Présentation de composants de notre référentiel local

Cette image représente l'arborescence générale de notre dépôt local déployé à l'aide l'outil REPREPO.

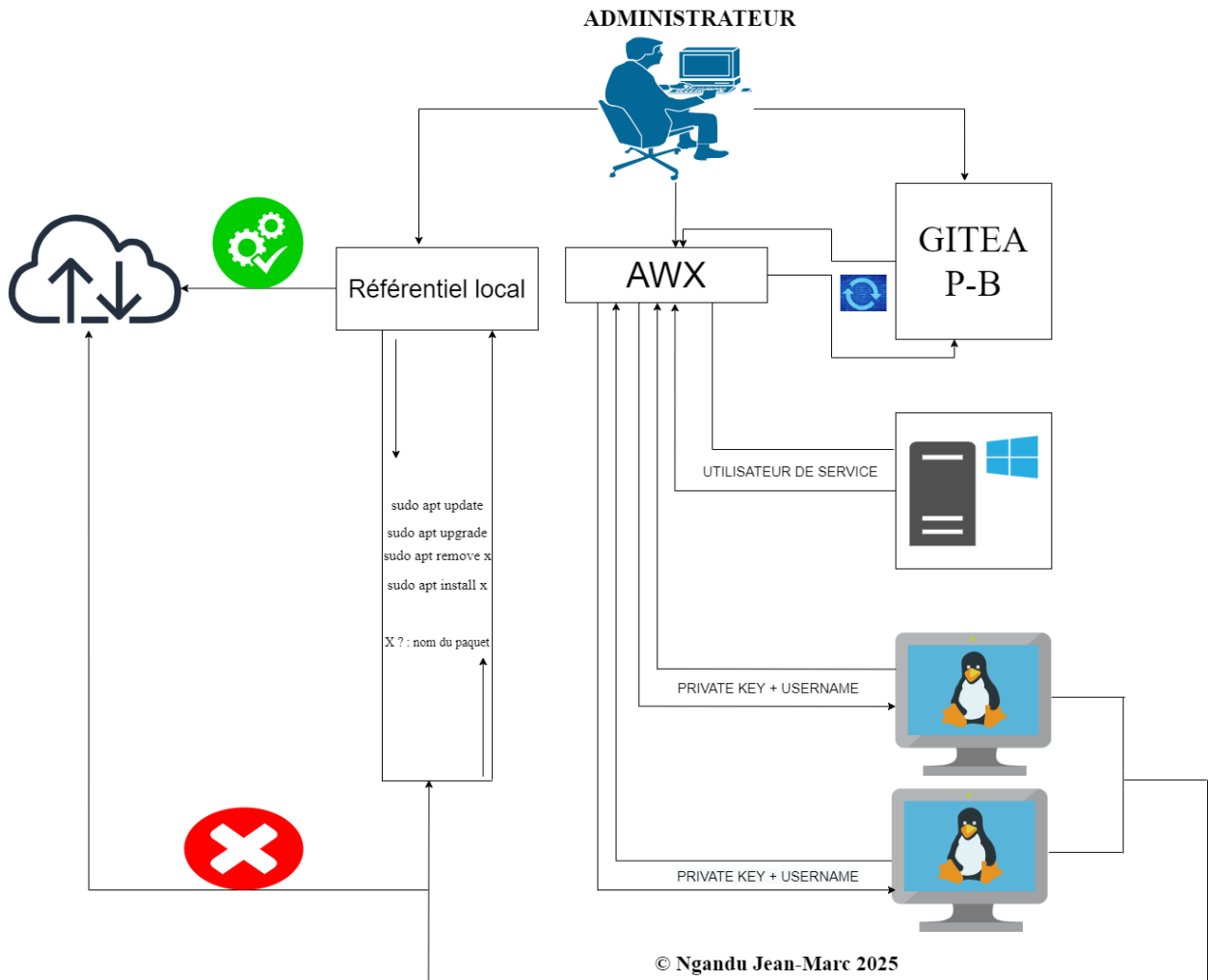


Image 14 : Workflow exacte de notre architecture général

Compte tenu de différentes spécificités de notre hôte local, nous avons misé sur l'utilisation des VMS de manière progressive afin de maximiser les ressources de la machine hôte telles que (RAM, CPU, carte graphique, etc...) et par la suite partager les différents fichiers de configuration vers les VMS cible.

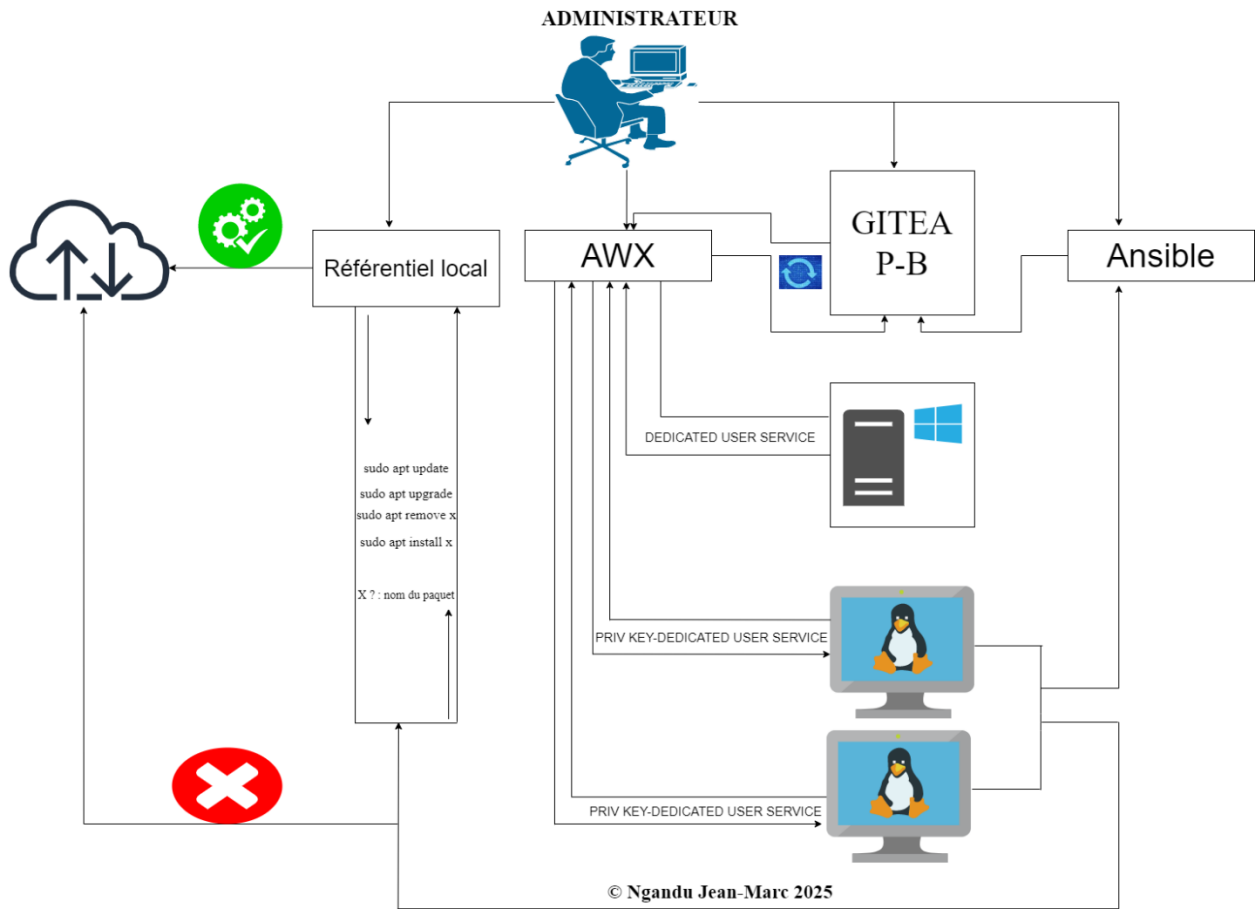


Image 15 : Workflow général de notre architecture général

Ici on voit bien une VM linux ansible qui gère deux machines clientes et transfère le fichier au serveur Gitea à la fin (après vérification et validation)



## **TROISIEME PARTIE :**

### **DESCRIPTION GENERALE DE LA SOLUTION AYANT ETE DEPLOYEE**

Cette troisième partie décrit le fonctionnement global de notre architecture. Afin de bien visualiser cela, veuillez cliquer sur le lien ci-dessous :

[\*\*Lien ici github \(format vidéo\)\*\*](#)

## CONCLUSION

En guise de conclusion,

Sachant que nous vivons dans un monde hyper-connecté il devient de plus en plus nécessaire de gérer son infrastructure informatique de manière plus subtile et flexible afin de limiter toute situation pouvant entraîner le dysfonctionnement d'un système infrastructure. Étant donné que le risque zéro n'existe pas. Il serait plus raisonnable de mettre en œuvre un système informatique solide afin non seulement d'avoir une vision globale et détaillée de différents paquets, images, etc... mais aussi de maximiser le délai d'exécution de différentes tâches lors de différentes mises-à-jour, il revient donc aux entreprises de faire un choix judicieux permettant de les protéger en interne tout comme en externe et de sensibiliser les différents employés de ne point exécuter une simple commande, cliquer sur un lien non maîtrisable, etc... Effectuer ce geste constitue un risque de sécurité qui ne se voit pas du coup. La sécurité étant un élément majeur dans le monde actuel, l'adoption de bonnes pratiques constitue un bon départ de sécurité.

## Table of Contents

<b>PREMIERE PARTIE :</b>	1
<b>INTRODUCTION</b>	1
<b>DEUXIEME PARTIE :</b>	2
<b>DIMENSIONNEMENT, OS ET IMPORTANCE</b>	2
<b>1. AWX</b>	3
Prérequis techniques :	3
<b>2. GITEA</b>	4
Prérequis techniques:	4
<b>3. WINDOWS SERVER 19</b>	6
Prérequis techniques :	6
<b>4. ANSIBLE</b>	7
Prérequis techniques :	8
<b>5. CLIENT 1 ET CLIENT 2</b>	11
Prérequis techniques :	11
<b>6. REFERENTIEL LOCAL</b>	11
Prérequis techniques :	12
<b>TROISIEME PARTIE :</b>	17
<b>DESCRIPTION GENERALE DE LA SOLUTION AYANT ETE DEPLOYEE</b>	17
<b>CONCLUSION</b>	18