Galaxy Swiss Bourdin

Sommaire

I. Introduction

1. Contexte

Le laboratoire Galaxy Swiss Bourdin (GSB), issu de la fusion entre Galaxy et Swiss Bourdin, est devenu un leader mondial en 2009. Basé à Paris, GSB a choisi la France pour améliorer le suivi de ses activités de visite médicale, tout en ayant son siège social à Philadelphie, aux États-Unis. J'interviens en tant qu'administrateur système et réseau au sein de ce groupe.

2. Besoin

Le laboratoire Galaxy Swiss Bourdin (GSB) souhaite mettre en place un outil d'automatisation et d'industrialisation afin de mieux gérer la configuration et le déploiement de ses infrastructures. Cet outil devra permettre :

- **L'automatisation des déploiements :**
  Éviter la répétition manuelle de tâches d'installation ou de mise à jour, tout en garantissant la cohérence des configurations à travers l'ensemble du parc serveur.
- **La gestion centralisée :**
  Centraliser la définition des rôles et des tâches, afin de standardiser et d'uniformiser le déploiement des différents services (base de données, serveurs web, portails captifs, etc.).
- **La maintenance et évolutivité simplifiées :**
  Faciliter l'application rapide de mises à jour ou de correctifs de sécurité, grâce à une structure modulaire et scalable qui s'intègre dans l'environnement existant de GSB.

En somme, GSB requiert un outil qui assure à la fois la fiabilité, la simplicité et la rapidité des opérations, tout en respectant les politiques internes de sécurité et de conformité déjà en vigueur. L'objectif est d'éliminer les processus manuels répétitifs, de réduire les risques d'erreur et d'améliorer la visibilité globale sur l'infrastructure.

II. Solution

1. Choix de la technologie

Pour pouvoir automatiser et industrialiser la configuration de nos serveurs, l'une des solutions consiste à mettre en place un outil d'automatisation. Grâce à ce type de plateforme, nous bénéficions d'une vue centralisée sur les tâches d'installation et de maintenance, et nous pouvons être avertis rapidement lorsqu'un dysfonctionnement ou une incohérence de configuration survient sur un serveur ou un équipement informatique.

L'industrialisation des serveurs est le processus qui permet d'établir une configuration standard, de déployer celle-ci de manière fiable et répétable, et d'assurer la maintenance continue des serveurs physiques ou virtuels.

2. Analyse comparative

Je souhaite choisir une solution d'automatisation et de gestion de configuration qui réponde précisément aux besoins de mon infrastructure. Pour cela, je me suis intéressé(e) aux outils les plus répandus et les plus éprouvés, qu'ils reposent sur un principe d'exécution agentless (sans agent) ou qu'ils requièrent un Master/Agent. Mon objectif est de bénéficier d'une plateforme qui offre une vue centralisée sur les tâches d'installation et de maintenance, tout en assurant une détection rapide des éventuels dysfonctionnements ou incohérences de configuration.

Cette analyse va ainsi comparer plusieurs solutions, en tenant compte de la facilité de mise en œuvre, de la communauté et de la richesse de l'écosystème, de la performance, ainsi que du coût et de la scalabilité. Les principaux points forts et les limites de chacune seront détaillés dans le tableau suivant, afin de m'aider à déterminer laquelle convient le mieux à mon environnement et à mes priorités.

Critère	Ansible	Puppet	Chef	SaltStack
Facilité d'utilisation	☆☆☆☆☆ (5/5)	☆☆☆ (3/5)	☆☆☆ (3/5)	☆☆☆☆ (4/5)
	YAML clair, agentless	Utilise DSL Ruby, nécessite un Master/Agent	DSL Ruby, architecture Master/Agent	Syntaxe proche de Python, plus simple qu'un DSL
Communauté et écosystème	☆☆☆☆☆ (5/5)	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)	☆☆☆ (3/5)
	Très vaste (Galaxy, docs, etc.)	Communauté solide et historique	Bonne communauté, surtout en entreprise	Moins populaire, communauté plus restreinte
Performance	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)	☆☆☆☆☆ (5/5)
	Performant, surtout en push	Stable pour configurations continues	Bonne performance, mais dépend du Chef Server	Très rapide, gestion d'événements en temps réel
Modèle d'architecture	Push (sans agent)	Pull (Master/Agent)	Pull (Chef Server)	Mixte (push/pull)
	S'exécute via SSH	Contrôle en continu	Orienté recettes (recipes)	Possibilité de Master/Minion
Configuration as code	Basé sur YAML (playbooks, rôles)	Basé sur manifestes (.pp) en DSL Ruby	Basé sur recipes en DSL Ruby	Basé sur states/pillars (YAML/Jinja)
Scalabilité	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)	☆☆☆☆ (4/5)
	S'adapte bien à de nombreux serveurs	Conçu pour de grands environnements	Capable de gérer des infrastructures étendues	Bonne gestion des environnements distribués
Coût	☆☆☆☆☆ (5/5)	☆☆☆ (3/5)	☆☆☆ (3/5)	☆☆☆☆ (4/5)
	Gratuit (open source)	Version open source, options Enterprise payantes	Version open source et Enterprise (Chef Automate)	Gratuit (open source), avec modules payants éventuels

Au terme de cette analyse, j'opte pour Ansible : sa simplicité d'utilisation, son modèle agentless et sa communauté dynamique répondent parfaitement à mes besoins. De plus, la clarté de sa syntaxe facilite l'automatisation et la maintenance de mon infrastructure, faisant d'Ansible le choix évident pour industrialiser les configurations.

III. Ansible

1. Qu'est-ce qu'Ansible ?

Ansible est un outil d'automatisation et de gestion de configuration agentless, c'est-à-dire qu'il ne nécessite pas d'installer de programme spécifique sur les machines cibles. Il repose sur un langage de description clair (YAML) pour définir les actions à effectuer, comme l'installation de paquets ou la configuration de services. Sa mise en œuvre est simple, grâce à une architecture basée sur le **push** (via SSH) et une communauté très active qui propose de nombreux rôles prêts à l'emploi. Ansible permet de déployer rapidement des environnements homogènes, de maintenir la cohérence des configurations et de réduire considérablement les tâches manuelles répétitives.

2. Origines d'Ansible

Ansible a été créé en 2012 par Michael DeHaan, qui souhaitait rendre l'automatisation des infrastructures plus accessible. Inspiré par des solutions comme Puppet ou Chef, il a choisi de développer un outil plus léger et plus simple à prendre en main grâce à :

- **La syntaxe YAML** : pour décrire les configurations de façon lisible.
- **L'architecture agentless** : pour simplifier l'installation et la maintenance des serveurs.

3. Les avantages d'Ansible

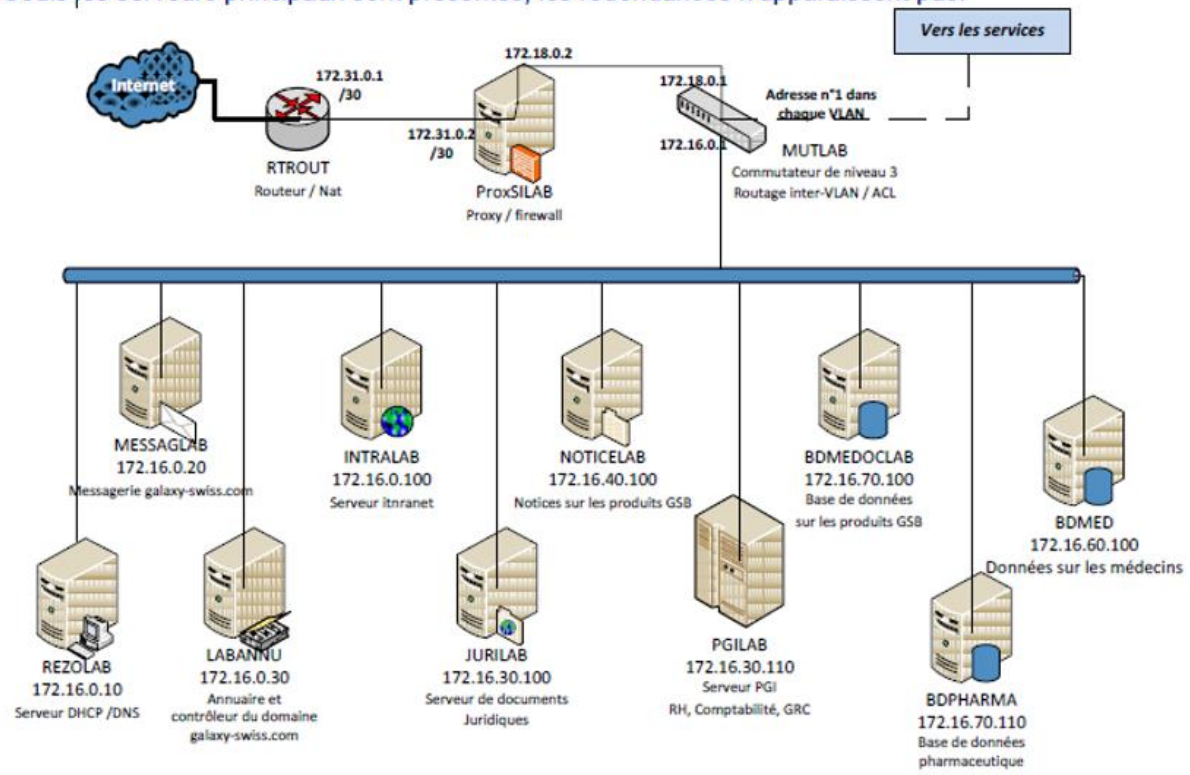
- **Simplicité et efficacité** : La syntaxe YAML et l'absence d'agent à installer permettent de démarrer rapidement.
- **Idempotence** : Les mêmes instructions peuvent être réexécutées sans risque de modifier un système déjà conforme, réduisant ainsi les erreurs.
- **Flexibilité** : Qu'il s'agisse d'environnements physiques, virtuels ou cloud, Ansible s'adapte facilement. Ses modules et sa capacité d'orchestration en font un choix idéal pour les architectures modernes et distribuées.

IV. Infrastructure

1. Schéma réseau actuel

Salle serveur et connexion internet

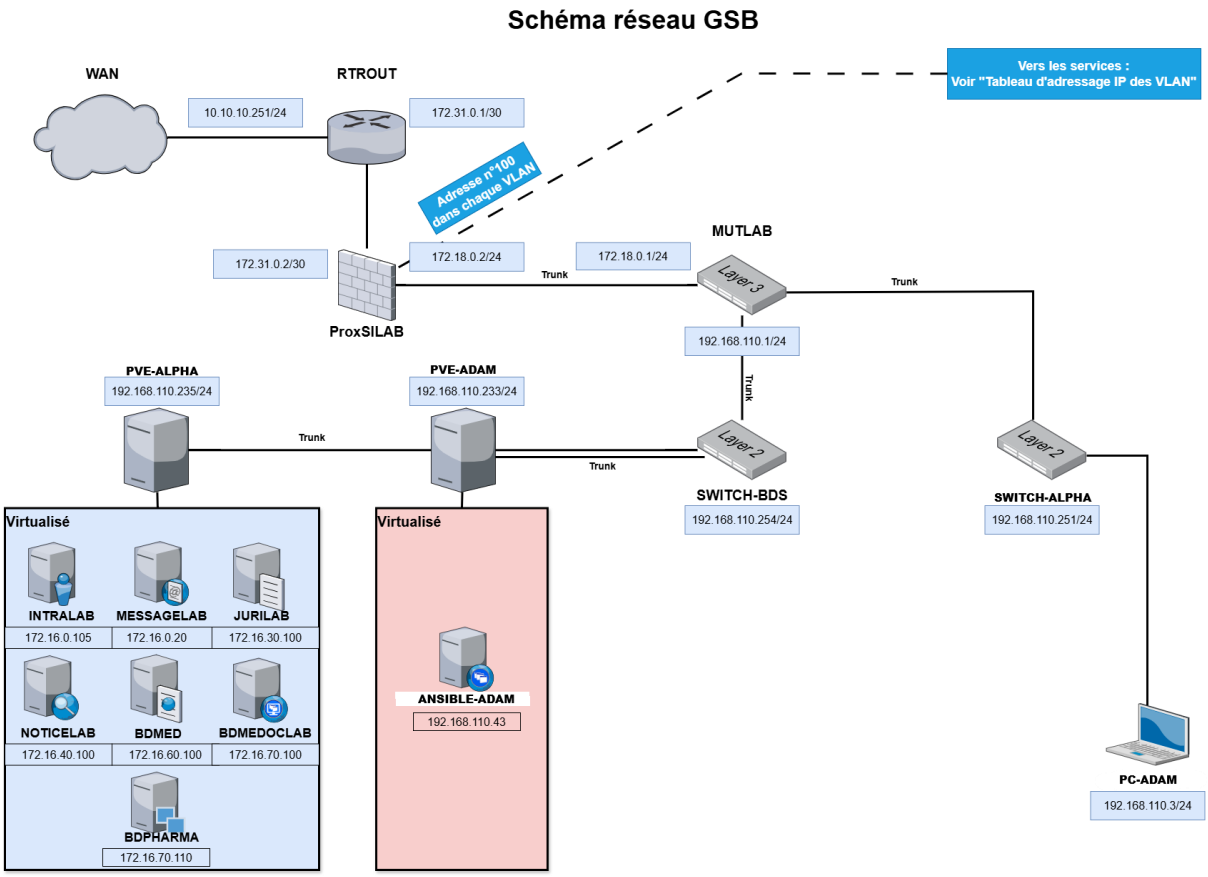
L'organisation des serveurs est la suivante. Il n'est pas précisé si les serveurs sont virtualisés ou non. Seuls les serveurs principaux sont présentés, les redondances n'apparaissent pas.



2. Tableau d'Adressage IP des VLAN

VLAN	Service(s)	Passerelle VLAN
110	Réseau & Système	192.168.110.0/24
20	Direction / DSI	192.168.20.0/24
30	RH / Compta / Juridique / Secrétariat	192.168.30.0/24
40	Communication / Rédaction	192.168.40.0/24
50	Développement	192.168.50.0/24
60	Commercial	192.168.60.0/24
70	Labo-Recherche	192.168.70.0/24
100	Accueil	192.168.100.0/24
150	Visiteurs	192.168.150.0/24
200	Démonstration	192.168.200.0/24
300	Serveurs	172.16.0.0/17
400	Sortie	172.18.0.0/30

3. Schéma réseau de la réalisation professionnelle



4. Matériel à disposition

Dans la mise en place de ma réalisation professionnelle, voici le matériel mis à ma disposition par le groupe GSB :

- **Cisco Catalyst 3560G (MUTLAB)**
- **Cisco Catalyst 3750G (SW-RS-ALPHA)**
- **Cisco Catalyst 2960-S (SWITCH-BDS)**
- **Un routeur Cisco (RTROUT)**
- **Un routeur/pare-feu pfSense (ProxSilab)**
- **Hyperviseur de type 1 (PVE-ALPHA)**
- **Hyperviseur de type 1 (PVE-ADAM)**
- **Un point d'accès Wi-Fi**

V. Mise en place d'Ansible

1. Installation

1. Configuration de ma VM DEB13-ADAM

Sur une VM Debian 13 existante sur mon environnement, je créé un utilisateur ansible

'adduser ansible'

```
adam@DEB13-MASTER:~$ sudo adduser ansible
[sudo] Mot de passe de adam :
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour ansible
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Is the information correct? [Y/n]
```

J'ajoute cet utilisateur au groupe sudo

'usermod -aG sudo ansible'

```
adam@DEB13-MASTER:~$ sudo usermod -aG sudo ansible
adam@DEB13-MASTER:~$ su ansible
Mot de passe :
ansible@DEB13-MASTER:/home/adam$ sudo whoami
[sudo] Mot de passe de ansible :
root
```

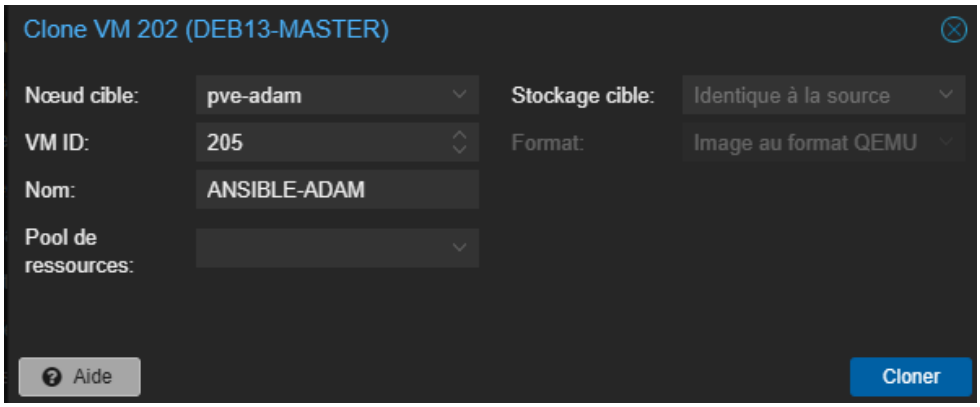
Afin de permettre à l'utilisateur ansible d'utiliser sudo sans mot de passe, j'ajoute la ligne suivante au fichier '/etc/sudoers'

ansible ALL=(ALL) NOPASSWD :ALL

```
adam@DEB13-MASTER:~$ sudo nano /etc/sudoers
  GNU nano 8.4                                          /etc/sudoers *
# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d

# Ansible
ansible ALL=(ALL) NOPASSWD:ALL
```

2. Clonage de la VM

Je clone la VM DEB13-ADAM dans mon environnement Proxmox. Cette étape me permet de disposer d'un clone fonctionnel prêt pour l'installation d'Ansible.



3. Renommage la VM

Après avoir dupliqué la VM, je lui attribue un nouveau nom pour la différencier de la VM d'origine. Pour ce faire, j'édite les fichier '/etc/hosts' et '/etc/hostname' :

sudo nano /etc/hosts

```
adam@DEB13-MASTER:~$ sudo nano /etc/hosts
  GNU nano 8.4                                                  /etc/hosts *
127.0.0.1       localhost
127.0.1.1       ANSIBLE-ADAM.gsb.lan    ANSIBLE-ADAM

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

sudo nano /etc/hostname

```
adam@DEB13-MASTER:~$ sudo nano /etc/hostname
  GNU nano 8.4
ANSIBLE-ADAM
```

4. Configuration d'une adresse IP fixe

Pour que ma nouvelle VM ait une adresse IP fixe, je modifie la configuration réseau. J'ajuste les paramètres réseau dans le fichier `/etc/network/interfaces` :

sudo nano /etc/network/interfaces

```
  GNU nano 8.4                                          /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug ens18
iface ens18 inet static
        address 192.168.110.60
        gateway 255.255.255.0
        dns-nameserver 192.168.110.101 8.8.8.8 1.1.1.1
```

5. Génération de la clé SSH et échange de clé

Pour permettre à ma VM de communiquer avec mes machines debian sans mot de passe, je génère une clé SSH en utilisant la commande suivante :

ssh-keygen -t rsa

```
ansible@ANSIBLE-ADAM:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_ed25519):
Created directory '/home/ansible/.ssh'.
Enter passphrase for "/home/ansible/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_ed25519
Your public key has been saved in /home/ansible/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:oR483NqABCSg6hsiLQZwL/roWgVBm0x+DRiuRf91+Ns ansible@ANSIBLE-ADAM
The key's randomart image is:
+--[ED25519 256]--+
|+oOo.            |
|.B.* o .         |
|o X + o +        |
|o+ = = * .       |
|+ . + O S        |
|oo o . * o       |
|=+o   o o E      |
|+=o              |
|=o.              |
+----[SHA256]-----+
```

Ensuite, j'ajoute la clé publique de ma VM sur ma machine DEB13-ADAM, en utilisant la commande suivante pour le transfert :

ssh-copy-id ansible@192.168.110.42

```
ansible@ANSIBLE-ADAM:~$ ssh-copy-id ansible@192.168.110.54
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_ed25519.pub"
The authenticity of host '192.168.110.54 (192.168.110.54)' can't be established.
ED25519 key fingerprint is SHA256:bhgFnHFWffgfGFtIPZFV0BIChnTj6kQ5Uw+SV2it+nY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@192.168.110.54's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@192.168.110.54'"
and check to make sure that only the key(s) you wanted were added.

ansible@ANSIBLE-ADAM:~$
```

Enfin, je vérifie le bon fonctionnement de la clé en me connectant en SSH à DEB13-ADAM depuis ANSIBLE-ADAM :

ssh ansible@192.168.110.42

```
ansible@ANSIBLE-ADAM:~$ ssh ansible@192.168.110.54
Linux DEB13-MASTER 6.12.63+deb13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.12.63-1 (2025-12-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jan 19 11:08:34 2026 from 192.168.110.60
ansible@DEB13-MASTER:~$ La connexion SSH sans mot de passe a bien fonctionné !
```

6. Installation d'Ansible

Une fois que la communication SSH est prête, je procède à l'installation d'Ansible sur la VM. Afin d'obtenir la version la plus récente, je suis les recommandations d'installation sur la documentation d'Ansible :

For a more recent version, Debian users can use the Ubuntu PPA according to the following table:

Debian		Ubuntu	UBUNTU_CODENAME
Debian 13 (Trixie)	->	Ubuntu 24.04 (Noble)	`noble`
Debian 12 (Bookworm)	->	Ubuntu 22.04 (Jammy)	`jammy`
Debian 11 (Bullseye)	->	Ubuntu 20.04 (Focal)	`focal`
Debian 10 (Buster)	->	Ubuntu 18.04 (Bionic)	`bionic`

The following example assumes that you already have `wget` and `gpg` installed.

Add the repository and install Ansible. Set `UBUNTU_CODENAME=...` based on the table above (we use `jammy` in this example):

```
$ UBUNTU_CODENAME=jammy
$ wget -O- "https://keyserver.ubuntu.com/pks/lookup?fingerprint=on&op=get&search=0x6125E2A8C77F2818FB7BD15B93C4A3FD7BB9C367" | sudo gpg --dearmour -o
/usr/share/keyrings/ansible-archive-keyring.gpg
$ echo "deb [signed-by=/usr/share/keyrings/ansible-archive-keyring.gpg] http://ppa.launchpad.net/ansible/ansible/ubuntu $UBUNTU_CODENAME main" | sudo
tee /etc/apt/sources.list.d/ansible.list
$ sudo apt update && sudo apt install ansible
```

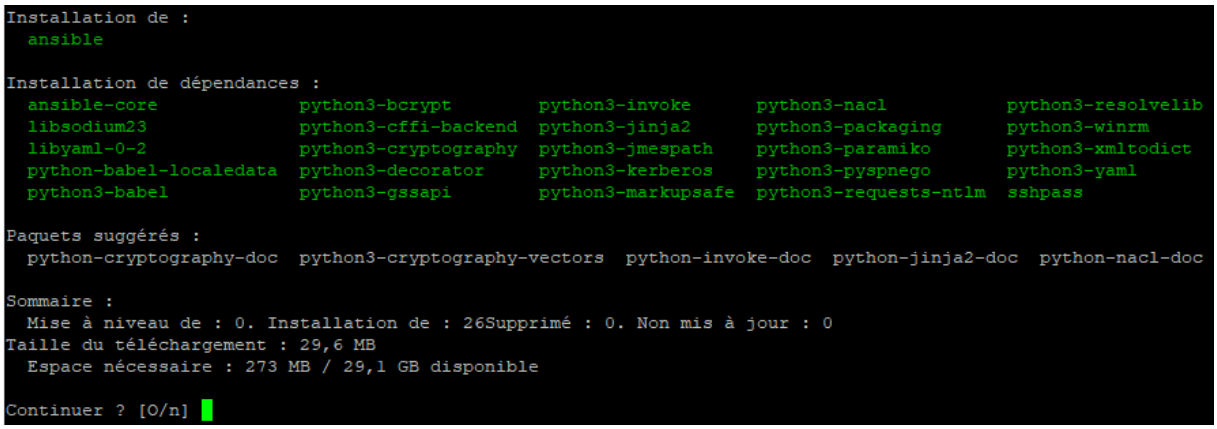
J’utilise les commandes suivantes afin de compléter mon installation :

UBUNTU_CODENAME=noble

wget -O- "https://keyserver.ubuntu.com/pks/lookup?fingerprint=on&op=get&search=0x6125E2A8C77F2818FB7BD15B93C4A3FD7BB9C367" | sudo gpg --dearmour -o /usr/share/keyrings/ansible-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/ansible-archive-keyring.gpg] http://ppa.launchpad.net/ansible/ansible/ubuntu $UBUNTU_CODENAME main" | sudo tee /etc/apt/sources.list.d/ansible.list

sudo apt update && sudo apt install ansible



L’installation est maintenant terminée, je passe à la configuration.

2. Configuration

1. Création du fichier d'inventaire Ansible

Je crée le fichier `~/ansible/inventory/hosts`, dans lequel j'ajoute les machines présente sur le réseau GSB :

Ce fichier `hosts` contient les machines sur lesquelles j'exécuterai mes commandes avec Ansible.

```
  GNU nano 8.4                                    /etc/ansible/inventory/hosts
[gsb]

INTRALAB        ansible_host=172.16.0.105
BDMED           ansible_host=172.16.60.100
BDMEDOCLAB      ansible_host=172.16.70.100
BDPHARMA        ansible_host=172.16.70.110
MESSAGELAB      ansible_host=172.16.0.20
JURILAB         ansible_host=172.16.30.100
NOTICELAB       ansible_host=172.16.40.100

[test]

TEST1   ansible_host=192.168.110.51
TEST2   ansible_host=192.168.110.52
TEST3   ansible_host=192.168.110.58
```

2. Création du fichier config Ansible

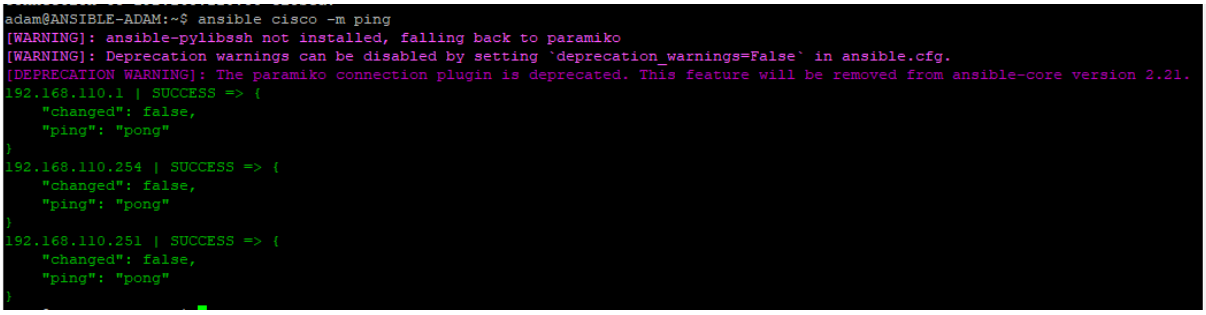
Je crée le fichier `~/ansible/ansible.cfg`, ce fichier contient des informations essentielles au fonctionnement d'ansible, tel que l'emplacement du fichier hosts, des rôles ainsi que de la clé SSH utilisé pour se connecter aux machines du parc :

```
[defaults]
inventory = /etc/ansible/inventory/hosts
roles_path = /etc/ansible/roles
remote_user = ansible
private_key_file = /home/ansible/.ssh/id_rsa
host_key_checking = False
interpreter_python = auto_silent
[ssh_connection]
ssh_args = -o KexAlgorithms=+diffie-hellman-group14-sha1,diffie-hellman-group1-sha1 -o HostKeyAlgorithms=+ssh-rsa -o Ciphers=+aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
```

3. Test

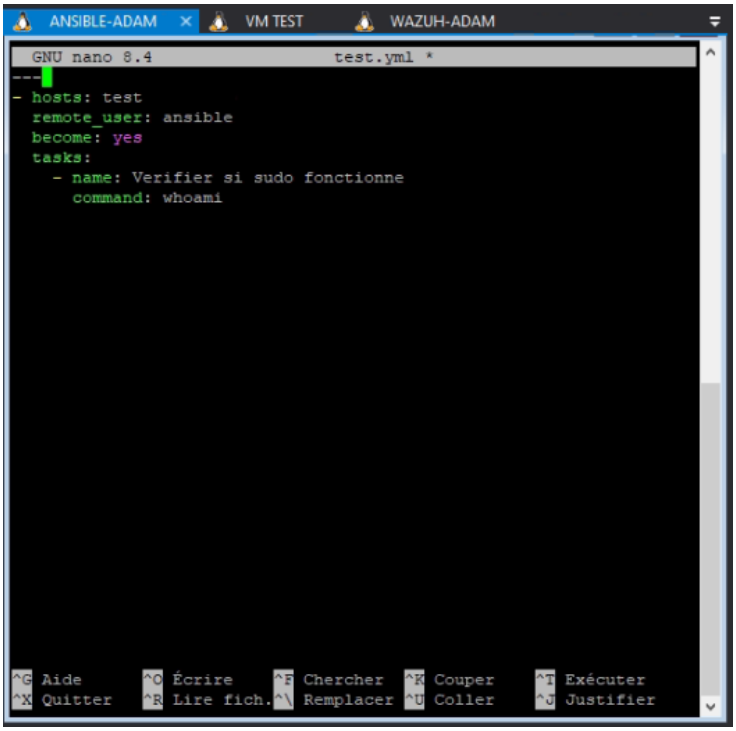
1. Ping des machines

Afin de confirmer que les machines sont bien joignables depuis Ansible, j'utilise la commande 'ansible test -m ping' :



2. Création d'un playbook de test

Pour tester l'exécution d'un playbook, je crée un playbook nommé `test.yml`. Ce fichier contient une simple commande `whoami` pour vérifier que la connexion est fonctionnelle. Voici le contenu de mon fichier `test.yaml` :

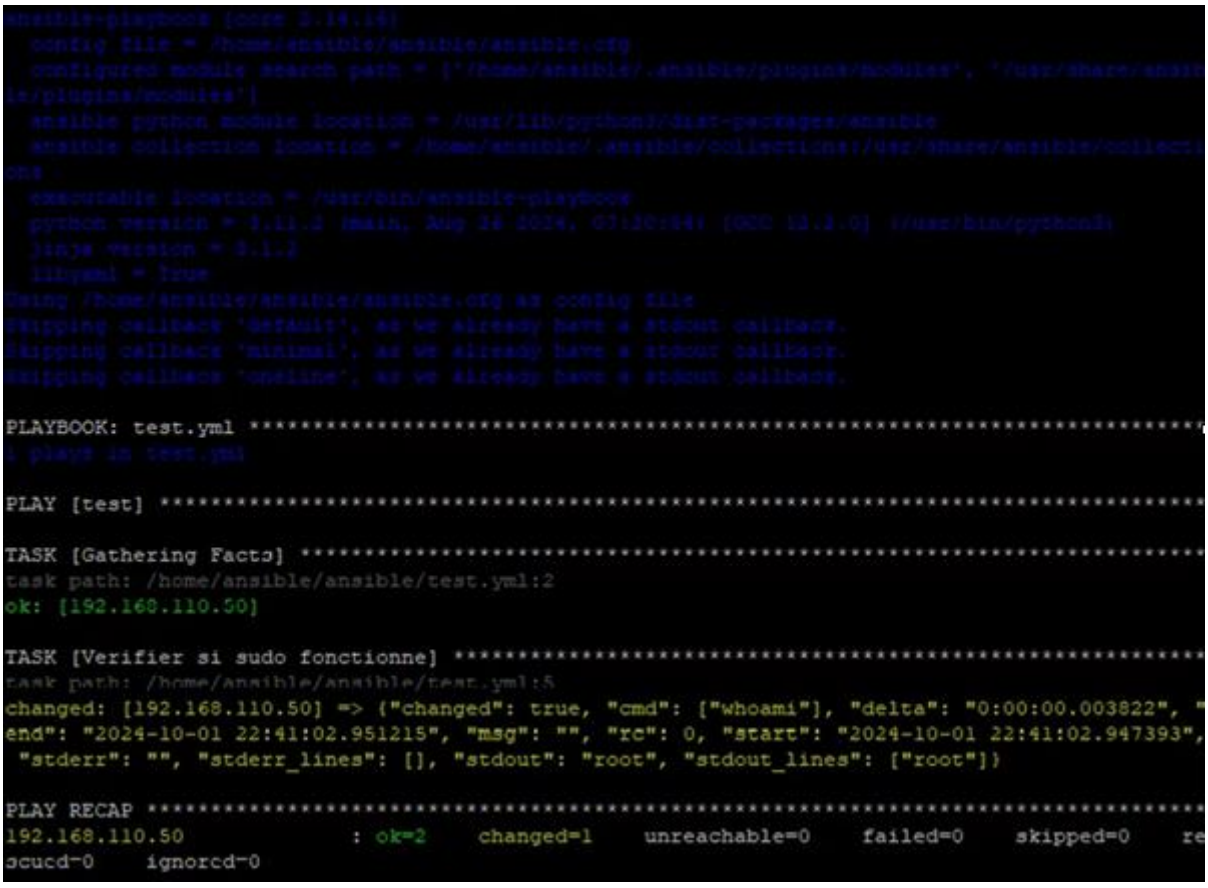


3. Exécution du playbook

Enfin, je lance le playbook pour vérifier la connexion et l'exécution de la commande :

ansible-playbook test.yaml

Ansible exécute la commande `whoami` sur les machines du groupe `test`. Si tout fonctionne correctement, je devrais voir apparaître le nom d'utilisateur utilisé pour la connexion, confirmant que la configuration est réussie.



VI. Évolution possible

1. Installation de WordPress

En intégrant un rôle dédié à l'installation et à la configuration de WordPress, GSB peut automatiser la mise en place de ces applications web sur des serveurs Linux. Cela inclut l'installation des dépendances nécessaires (PHP, MySQL, Apache), la création de bases de données, ainsi que la configuration initiale de WordPress.

Intérêts pour GSB :

- Réduire les interventions manuelles et les risques d'erreur.
- Uniformiser les déploiements sur plusieurs environnements (test, production).
- Simplifier la maintenance et les mises à jour de ces applications critiques.

VII. Conclusion

L'adoption d'Ansible au sein du laboratoire Galaxy Swiss Bourdin (GSB) s'avère être une décision stratégique, permettant d'automatiser et d'industrialiser de nombreux processus critiques : déploiements de serveurs, configuration de logiciels, supervision et maintenance.

Cette approche agentless réduit la complexité et facilite la prise en main, tandis que l'écriture de tâches sous forme de playbooks en YAML offre une lisibilité et une standardisation accrues. Avec un écosystème large et dynamique, Ansible apporte une réponse à la fois fiable et flexible aux besoins variés de GSB, tout en ouvrant la voie à des évolutions futures (installation de nouvelles applications, gestion des postes Windows, sauvegardes réseau, etc.).

En définitive, Ansible se positionne comme un levier essentiel pour gagner en efficacité, renforcer la sécurité et améliorer la qualité de service au sein de l'infrastructure GSB.