

Mise en place du serveur Ansible

Le serveur Ansible, c'est quoi ?

Le serveur Ansible est le nœud de contrôle : la seule machine qui détient le dépôt, la clé SSH d'administration et le mot de passe du Vault. Les VM clientes n'ont rien d'installé en plus que SSH et Python.

Élément	Rôle
Environnement virtuel Python	Isole la version d'Ansible du système
Clé SSH <code>ed25519</code>	Authentifie le serveur Ansible auprès des VM
Fichier <code>.vault_pass</code>	Contient le mot de passe de déchiffrement du Vault

Dépôt Git → ansible-playbook → SSH (réseau d'administration) → VM client

Note — Le serveur Ansible est le chemin des machines : il se connecte aux VM de façon automatique. Les interventions humaines, elles, passent par le bastion Teleport (voir le livre [Teleport](#) et la page de présentation du projet).

Concrètement, la version d'Ansible fournie par `apt` sous Debian 12 est ancienne. L'installation se fait donc dans un environnement virtuel avec `pip`, ce qui permet de figer la version et de la mettre à jour sans toucher au système.

Prérequis

1. **Une VM Debian 12** dans le réseau d'administration (`10.99.0.2`), avec 2 Go de mémoire.
2. **Un utilisateur non-root** avec `sudo` : la suite est exécutée avec cet utilisateur.
3. **Un accès SSH** aux VM clientes par la clé publique, avec l'utilisateur `deploy` (créé lors du provisioning).

Étape 1 : mettre le système à jour et installer les outils

```
# Mise à jour de l'index des paquets puis des paquets installés.
sudo apt update && sudo apt full-upgrade -y

# python3-venv : création d'environnements virtuels
# python3-pip   : installateur Python
# git           : gestion du dépôt
# jq, curl      : utiles pour interroger des API (voir la page Authentik)
sudo apt install -y python3-venv python3-pip git jq curl
```

Étape 2 : créer l'environnement virtuel et installer Ansible

```
# Création du dépôt de travail. Tout le projet vivra ici.
mkdir -p ~/nebuleuse && cd ~/nebuleuse
git init

# Création de l'environnement virtuel dans un dossier caché du répertoire personnel.
python3 -m venv ~/.venv/ansible

# Activation : le prompt se préfixe de (ansible), et "pip" / "ansible" pointent
# désormais vers l'environnement virtuel.
source ~/.venv/ansible/bin/activate
```

Le fichier `requirements.txt` fige la version d'Ansible utilisée par le projet.

```
# requirements.txt
# ansible-core : le moteur d'Ansible, sans les collections de modules (installées à part).
# La plage de versions évite une montée de version majeure involontaire.
ansible-core>=2.17,<2.19

# jmespath : requis par le filtre "json_query", utile pour exploiter les réponses d'API.
```

```
jmespath
```

```
pip install -r requirements.txt
```

```
ansible --version # vérification : affiche la version et le chemin de l'interpréteur Python
```

Les collections de modules sont déclarées dans un second fichier, indépendant de la version du moteur.

```
# requirements.yml
collections:
  # community.docker : modules docker_compose_v2, docker_network...
  - name: community.docker
  # community.general : modules divers (ufw, etc.) utilisés par certains rôles
  - name: community.general
  # ansible.posix : modules système (sysctl, authorized_key...)
  - name: ansible.posix
```

```
ansible-galaxy collection install -r requirements.yml
```

Étape 3 : créer la clé SSH d'administration

```
# -t ed25519 : algorithme moderne, clés courtes et rapides
# -f          : chemin du fichier de clé
# -C          : commentaire pour reconnaître la clé dans les authorized_keys
# La clé n'a volontairement pas de passphrase : elle est utilisée par des playbooks
# non interactifs. En contrepartie, le fichier doit rester en mode 600 et le serveur
# Ansible n'est accessible que depuis le réseau d'administration.
ssh-keygen -t ed25519 -f ~/.ssh/nebuleuse_ed25519 -C "ansible@nebuleuse"
```

La clé publique (`~/.ssh/nebuleuse_ed25519.pub`) est celle que le processus de provisioning dépose dans `~deploy/.ssh/authorized_keys` de chaque VM.

Étape 4 : le mot de passe du Vault

```
# Génère 32 caractères aléatoires et les écrit dans .vault_pass.
openssl rand -base64 24 > .vault_pass

# Lecture/écriture pour le propriétaire uniquement.
chmod 600 .vault_pass
```

Le fichier `.gitignore` empêche que le mot de passe du Vault parte dans le dépôt.

```
# .gitignore
# Ne jamais versionner le mot de passe du Vault ni la clé privée.
.vault_pass
*.pem
*_ed25519
```

Note — Le mot de passe du Vault doit aussi être conservé hors du serveur Ansible (gestionnaire de mots de passe, coffre physique). Sans lui, les fichiers chiffrés sont définitivement illisibles.

Étape 5 : le fichier de configuration `ansible.cfg`

```
# ansible.cfg
[defaults]
# Inventaire utilisé par défaut : évite de passer -i à chaque commande.
inventory = inventory/hosts.yml

# Dossier des rôles du projet.
roles_path = roles

# Utilisateur SSH sur les VM clientes.
remote_user = deploy

# Clé privée utilisée pour la connexion.
private_key_file = ~/.ssh/nebuleuse_ed25519

# Fichier contenant le mot de passe du Vault (créé à l'étape 4).
# Doit exister : Ansible échoue au démarrage si le fichier est introuvable.
vault_password_file = .vault_pass
```

```
# Nombre de machines traitées en parallèle.
forks = 10

# Détection automatique de l'interpréteur Python sur les cibles, sans avertissement.
interpreter_python = auto_silent

[privilege_escalation]
# Les tâches s'exécutent en root via sudo : l'utilisateur "deploy" a un sudo sans mot de
passe.
become = True
become_method = sudo

[ssh_connection]
# Pipelining : réduit le nombre d'opérations SSH par tâche, donc accélère les playbooks.
pipelining = True

# ControlMaster : réutilise une connexion SSH ouverte pendant 60 s.
# accept-new : accepte la clé d'hôte d'une VM inconnue à la première connexion,
# puis refuse tout changement ultérieur (protection contre l'usurpation).
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=accept-new
```

Étape 6 : un premier inventaire

```
# inventory/hosts.yml
all:
  children:
    clients:                                # groupe regroupant toutes les VM clientes
      hosts:
        client-alpha:
          # Adresse du réseau d'administration, pas l'adresse publique.
          ansible_host: 10.99.0.21
        client-beta:
          ansible_host: 10.99.0.22
```

Vérification

```
# Affiche l'arbre de l'inventaire : les deux clients doivent apparaître sous "clients".
ansible-inventory --graph

# Test de connectivité SSH et de Python (ce n'est pas un ping réseau).
ansible clients -m ansible.builtin.ping
```

Résultat attendu pour chaque client :

```
client-alpha | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Pièges

1. **vault_password_file introuvable** : le message « The vault password file ... was not found » apparaît dès la première commande. Il suffit de créer `.vault_pass` (étape 4) avant tout.
2. **Permission denied (publickey)** : la clé publique n'est pas dans les `authorized_keys` de l'utilisateur `deploy`, ou la clé privée n'est pas en mode 600.
3. **Mauvais ansible** : si l'environnement virtuel n'est pas activé, la commande `ansible` appelle la version système. Le préfixe `(ansible)` du prompt sert de contrôle.
4. **Pas de changed sur un ping** : normal, ce module est en lecture seule.

Page suivante : **Inventaire et variables.**

Revision #2

Created 2026-10-15 07:04:46 UTC by clement-derouet

Updated 2026-10-15 07:17:26 UTC by clement-derouet