Introduction à Ansible

Par **Rémi Albertucci**

ANSIBLE

Table des matières

I Introduction

I.1 Un mot sur Ansible

Ansible est un outil d'automatisation open source qui n'installe rien sur les machines cibles. La communication se fait uniquement via SSH, avec les droits de l'utilisateur défini dans l'inventaire.

Ansible s'appuie sur trois concepts fondamentaux :

- L'inventaire : la liste des machines à gérer, organisées en groupes.
- Les modules : des unités d'action atomiques (installer un paquet, modifier un fichier, redémarrer un service…).
- Les playbooks : des fichiers YAML qui orchestrent l'exécution de modules sur des groupes d'hôtes.

I.2 Le but de ce guide

Ce guide a pour objectif de mettre en place Ansible sur une infrastructure existante sous Proxmox et d'automatiser les premières tâches d'administration :

- Générer une clé SSH dédiée et la distribuer sur les hôtes.
- Créer un inventaire structuré des machines.
- Vérifier la connectivité depuis le nœud de contrôle.
- Appliquer un durcissement SSH via un rôle Ansible.
- Poser les bases pour Fail2ban et les mises à jour automatisées.

II Préparation : générer une clé SSH

II.1 Générer la clé

On commence par générer une paire de clés ed25519 dédiée à Ansible.

L'algorithme ed25519 est à privilégier : plus rapide et plus sûr que RSA 2048. On ne met pas de passphrase pour permettre l'exécution non-interactive des playbooks.

```
ssh-keygen -t ed25519 -C "ansible" -f ~/.ssh/ansible_ed25519
```

II.2 Distribuer la clé

La commande ssh-copy-id ajoute la clé publique dans le fichier ~/.ssh/authorized_keys de l'utilisateur distant. On itère sur l'ensemble des hôtes qu’on voudra gérer :

```
for ip in 192.168.XXX.XXX 192.168.XXX.XXX 192.168.XXX.XXX \
          192.168.XXX.XXX 192.168.XXX.XXX 192.168.XXX.XXX 192.168.XXX.XXX; do
  echo "=== $ip ==="
  ssh-copy-id -i ~/.ssh/ansible_ed25519.pub <USER>@$ip
done
```

III Structure du projet

III.1 Arborescence

On crée l'arborescence du projet Ansible dans le répertoire de l'utilisateur :

```
mkdir -p ~/ansible/inventory
mkdir -p ~/ansible/playbooks
mkdir -p ~/ansible/roles
```

L'arborescence finale du projet est la suivante :

```
~/ansible/
├── ansible.cfg
├── inventory/
│   └── hosts.yml
├── playbooks/
│   └── security.yml
└── roles/
    ├── ssh_hardening/
    │   ├── tasks/
    │   │   └── main.yml
    │   └── handlers/
    │       └── main.yml
    ├── fail2ban/
    │   ├── tasks/
    │   └── handlers/
    ├── upgrade/
    │   ├── tasks/
    │   └── handlers/
    └── distupgrade/
        ├── tasks/
        └── handlers/
```

III.2 Fichier de configuration

Le fichier ansible.cfg évite de préciser l'inventaire et le chemin des rôles à chaque commande. Il doit être placé à la racine du projet :

```
# ~/ansible/ansible.cfg
[defaults]
roles_path = ~/ansible/roles
inventory  = ~/ansible/inventory/hosts.yml
```

> 💡 **Note :** *Ansible cherche ansible.cfg dans l'ordre suivant : répertoire courant, ~/.ansible.cfg, /etc/ansible/ansible.cfg. Toujours lancer les commandes depuis ~/ansible/ pour utiliser ce fichier.*

IV Inventaire

IV.1 Organisation des groupes

L'inventaire est structuré en trois groupes selon le niveau de gestion :

Groupe	Mode	Machines
exposed	Automatisé	reverse-proxy, debian-docker, debian-opnvpn
internal	Automatisé	debian-ffmpeg, debian-ffmpeg2, debian-torrenting, adguard
manual	Exclu	proxmox1, proxmox2, holocron, whiteferret, opnsense

Le groupe manual regroupe les machines gérées manuellement (hyperviseurs, pare-feu, NAS...). Il est systématiquement exclu des playbooks via le pattern all:!manual.

IV.2 hosts.yml

Créer le fichier ~/ansible/inventory/hosts.yml :

```
all:
  children:

    # Machines exposées ou critiques
    exposed:
      hosts:
        reverse-proxy:
          ansible_host: 192.168.XXX.XXX
        debian-docker:
          ansible_host: 192.168.XXX.XXX
        debian-opnvpn:
          ansible_host: 192.168.XXX.XXX

    # Machines internes
    internal:
      hosts:
        debian-ffmpeg:
          ansible_host: 192.168.XXX.XXX
        debian-ffmpeg2:
          ansible_host: 192.168.XXX.XXX
        debian-torrenting:
          ansible_host: 192.168.XXX.XXX
        adguard:
          ansible_host: 192.168.XXX.XXX

    # Gérés manuellement — exclus des playbooks
    manual:
      hosts:
        proxmox1:
          ansible_host: 192.168.XXX.XXX
        proxmox2:
          ansible_host: 192.168.XXX.XXX
        holocron:
          ansible_host: 192.168.XXX.XXX
        whiteferret:
          ansible_host: 192.168.XXX.XXX
        opnsense:
          ansible_host: 192.168.XXX.XXX

  vars:
    ansible_user: <USER>
    ansible_become: true
    ansible_become_method: sudo
    ansible_ssh_private_key_file: ~/.ssh/ansible_ed25519
```

V Test de connectivité

V.1 Ping Ansible

Le module ping d'Ansible vérifie que chaque hôte est joignable via SSH et que Python est disponible sur la machine cible (requis pour l'exécution des modules) :

```
ansible 'all:!manual' -m ping
```

Une réponse « SUCCESS » pour chaque hôte confirme que la clé SSH est en place et que la connexion fonctionne.

V.2 Résolution des erreurs de sudo

Il faut se connecter en su sur chaque machine concernée pour installer sudo et mettre un user en « nopasswd » pour que les commandes puissent s’éxecuter :

```
ssh <USER>@192.168.11.XXX
su -
apt install -y sudo
echo "<USER> ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
exit
```

Relancer le ping après correction pour valider :

```
ansible 'all:!manual' -m ping
```

V.3 Vérification des clés autorisées

Pour confirmer que la clé Ansible est bien présente dans « authorized_keys » sur chaque hôte, on lance :

```
ansible 'all:!manual' \
  -m command -a "cat /home/<USER>/.ssh/authorized_keys"
```

VI Roles

VI.1 Ssh hardening

Présentation

Ce rôle applique un ensemble de directives de durcissement sur le démon SSH de chaque machine gérée. Il modifie le fichier /etc/ssh/sshd_config via le module lineinfile, puis redémarre sshd uniquement si une modification a été effectuée.

Directive	Valeur	Effet
PermitRootLogin	no	Interdit la connexion directe en root
PasswordAuthentication	no	Force l'auth par clé SSH uniquement
X11Forwarding	no	Désactive le forwarding graphique
PermitEmptyPasswords	no	Bloque les comptes sans mot de passe
MaxAuthTries	3	Limite à 3 les tentatives d'auth
LoginGraceTime	20	Délai max d'authentification (secondes)

> ⚠️ **Attention :** S'assurer que la clé SSH est déployée sur tous les hôtes avant d'appliquer ce rôle. Une fois PasswordAuthentication no activé, la connexion par mot de passe sera définitivement bloquée.

Structure

```
mkdir -p ~/ansible/roles/ssh_hardening/tasks
mkdir -p ~/ansible/roles/ssh_hardening/handlers
```

Fichiers tasks et handlers

tasks/main.yml :

```
- name: Durcissement sshd_config
  lineinfile:
    path: /etc/ssh/sshd_config
    regexp: "{{ item.regexp }}"
    line: "{{ item.line }}"
    state: present
  loop:
    - { regexp: '^#?PermitRootLogin',         line: 'PermitRootLogin no' }
    - { regexp: '^#?PasswordAuthentication', line: 'PasswordAuthentication no' }
    - { regexp: '^#?X11Forwarding',           line: 'X11Forwarding no' }
    - { regexp: '^#?PermitEmptyPasswords',    line: 'PermitEmptyPasswords no' }
    - { regexp: '^#?MaxAuthTries',            line: 'MaxAuthTries 3' }
    - { regexp: '^#?LoginGraceTime',          line: 'LoginGraceTime 20' }
  notify: restart sshd
```

handlers/main.yml :

```
- name: restart sshd
  systemd:
    name: sshd
    state: restarted
```

VI.2 Fail2ban

Présentation

Fail2ban surveille les journaux système et bannit automatiquement les adresses IP qui génèrent trop d'échecs d'authentification. Il s'appuie sur iptables ou nftables pour bloquer les IP au niveau du pare-feu hôte.

Combiné au durcissement SSH (MaxAuthTries 3), il constitue une deuxième ligne de défense contre les attaques par force brute.

Structure

```
mkdir -p ~/ansible/roles/fail2ban/tasks
mkdir -p ~/ansible/roles/fail2ban/handlers
```

Fichiers

tasks/main.yml :

```
- name: Installer fail2ban
  apt:
    name: fail2ban
    state: present
    update_cache: true

- name: Déployer jail.local
  copy:
    dest: /etc/fail2ban/jail.local
    content: |
      [DEFAULT]
      bantime  = 1h
      findtime = 10m
      maxretry = 3

      [sshd]
      enabled = true
      port    = ssh
      logpath = %(sshd_log)s
      backend = %(sshd_backend)s
  notify: restart fail2ban

- name: Activer et démarrer fail2ban
  systemd:
    name: fail2ban
    enabled: true
    state: started
```

handlers/main.yml :

```
- name: restart fail2ban
  systemd:
    name: fail2ban
    state: restarted
```

> 💡 **Note :** *jail.local surcharge jail.conf sans le modifier, ce qui évite que les mises à jour de Fail2ban n'écrasent la configuration.*

VI.3 Mises à jour

Présentation

Deux rôles distincts couvrent les mises à jour système. Leur séparation permet d'appliquer la dist-upgrade de façon contrôlée, machine par machine, tandis que les mises à jour courantes peuvent être déclenchées plus librement.

Rôle	Commande équivalente	Usage
upgrade	apt upgrade	Mises à jour des paquets installés
distupgrade	apt dist-upgrade	Mise à jour de distribution (peut ajouter/supprimer des paquets)

Structure

```
mkdir -p ~/ansible/roles/upgrade/tasks
mkdir -p ~/ansible/roles/upgrade/handlers
mkdir -p ~/ansible/roles/distupgrade/tasks
```

```
mkdir -p ~/ansible/roles/distupgrade/handlers
```

Fichiers

roles/upgrade/tasks/main.yml :

```
- name: Mettre à jour le cache apt
  apt:
    update_cache: true
    cache_valid_time: 3600

- name: Mettre à jour les paquets installés
  apt:
    upgrade: safe
  notify: reboot si nécessaire
```

roles/distupgrade/tasks/main.yml :

```
- name: Mettre à jour le cache apt
  apt:
    update_cache: true

- name: Dist-upgrade
  apt:
    upgrade: dist
  notify: reboot si nécessaire
```

handlers/main.yml (commun aux deux rôles) :

```
- name: reboot si nécessaire
  reboot:
    msg: "Reboot après mise à jour"
    reboot_timeout: 120
  when: ansible_facts['os_family'] == 'Debian'
```

> ⚠️ **Attention :** Le handler reboot redémarre la machine si déclenché. Ne l'inclure dans un playbook de production qu'avec une fenêtre de maintenance planifiée.

VII Playbooks

VII.1 Création du playbook

Le playbook security.yml regroupe les rôles de sécurité à appliquer sur l'ensemble des hôtes gérés. On peut y ajouter fail2ban dès que le rôle est prêt :

```
# ~/ansible/playbooks/security.yml
- name: Sécurisation des hôtes
  hosts: all:!manual
  roles:
    - ssh_hardening
    - fail2ban
```

Pour les mises à jour, un playbook dédié permet de les déclencher indépendamment :

```
# ~/ansible/playbooks/update.yml
- name: Mise à jour des paquets
  hosts: all:!manual
  roles:
    - upgrade
```

VII.2 Lancement du playbook

Se placer dans le répertoire du projet pour que ansible.cfg soit pris en compte, puis lancer le playbook :

```
cd ~/ansible
ansible-playbook playbooks/security.yml
```

Ansible affiche un résumé PLAY RECAP en fin d'exécution pour chaque hôte :

Statut	Signification
ok	Tâche vérifiée, aucun changement nécessaire (état déjà conforme).
changed	Tâche exécutée, une modification a été apportée.
failed	La tâche a échoué. Consulter le message d'erreur détaillé.
skipped	La tâche a été ignorée (condition when non remplie).

> 💡 **Note :** *Ajouter --check pour simuler l'exécution sans modifier les hôtes (dry-run). Utile pour valider un playbook avant de l'appliquer en production.*

VIII Conclusion

Nous savons maintenant comment mettre en place une base Ansible fonctionnelle sur l'infrastructure.

Les fondations posées (inventaire, clé SSH dédiée, rôles modulaires) permettent d'étendre facilement l'automatisation à de nouvelles tâches.