Implémentation d’un orchestrateur simplifié d’instructions distantes (type Ansible maison)

Contexte

Dans un contexte d’automatisation DevOps, la nécessité d’orchestrer des actions techniques sur des serveurs distants se généralise. Ce projet consiste à concevoir un outil de type Ansible maison, capable d'interpréter un fichier de tâches (au format YAML) et d’appliquer séquentiellement des actions sur un ou plusieurs serveurs distants en SSH.

Objectifs

- Créer un orchestrateur d’automatisation léger.

- Implémenter des connexions SSH sécurisées.

- Exécuter plusieurs types de modules (commandes, paquets, services, configurations).

- Définir les tâches dans un format structuré (YAML).

- Produire un code Python propre et conforme à la PEP8 (via black).

Fonctionnalités développées

1. Chargement des fichiers

todos.yaml : contient la séquence des tâches à exécuter.

inventory.yaml : liste des hôtes avec informations de connexion (IP, port, login, mot de passe ou clé SSH).

2. Connexion SSH

Nom d’utilisateur + mot de passe

Authentification simple.

*Utilisation de la bibliothèque paramiko pour l’établissement de la connexion SSH.*

3. Modules exécutables

Chaque tâche est associée à un module et des params. Les modules gérés sont :

command : exécution de commandes shell.

apt : installation ou suppression de paquets Debian.

sysctl : configuration de paramètres noyau.

service : démarrage ou arrêt de services.

copy : copie de fichier distant/local.

4. Affichage & journalisation

Utilisation du module logging pour afficher les étapes de traitement.

Résultat affiché pour chaque tâche, avec état d’exécution (stdout).

5. Conformité PEP8

L’outil black a été utilisé pour formater le code source.

6. Lancement script

Ton script principal utilise Click pour exécuter les tâches. Voici comment un opérateur doit l'utiliser dans un terminal :

python execute\_tasks.py -f todos.yaml -i inventory.yaml

### Explication des options :

* -f : chemin vers le fichier YAML décrivant les **tâches à exécuter** (ex. installer nginx, copier un fichier, lancer un service).
* -i : chemin vers le fichier YAML décrivant l’**inventaire des serveurs** cibles (adresses IP, ports, identifiants SSH, mot de passe ou clé privée).

✅ Format de sortie envisagé

À la fin de l’exécution, chaque tâche produit un résumé clair, structuré ainsi :

2025-07-03 10:30:21 - main - INFO - Connexion à webserver (192.168.1.10:22)

[webserver] ✅ Paquet 'nginx-common' installé

[webserver] ✅ Service 'nginx' démarré

[webserver] ✅ Paramètre système 'net.core.somaxconn' = 8192 (non permanent)

[webserver] ✅ Fichier /app/dummy.txt copié vers ./dummy.txt

[webserver] ✅ Template nginx appliqué avec les variables : listen\_port=8000, server\_name=\_

[webserver] ✅ Commande personnalisée exécutée : echo "test" > /tmp/dummy-file

2025-07-03 10:30:30 - main - INFO - Fin de traitement pour webserver

Résumé :

✔️ webserver : 6 tâches réussies / 6

Ces messages sont lisibles en langage courant (pas de jargon technique inutile), permettant à un utilisateur de comprendre l’état du système à l’issue du traitement.