Série d'Exercices Pratiques : Ansible - Déploiement d'un Serveur Web

Ce parcours d'exercices progressifs a pour objectif de vous faire maîtriser les concepts clés d'Ansible, en partant de commandes simples jusqu'à l'organisation avancée avec les Rôles et la gestion des secrets.

Scénario de Base : Le Serveur Web Nginx

Nous allons déployer et configurer un serveur web simple (Nginx) sur un ou plusieurs serveurs cibles.

Prérequis :

1. Ansible installé sur la machine de contrôle.
2. Au moins une machine cible Linux accessible via SSH avec un utilisateur ayant les droits sudo (e.g., web01).

Exercice 1 : Prise en Main et Inventaire (Inventory & Ad-Hoc)

Objectifs

- Créer le fichier d'inventaire.
- Tester la connectivité Ansible de base.
- Exécuter une commande simple à distance.

Étapes

1. **Créez un fichier inventory.ini** et définissez un groupe appelé [webservers]. Ajoutez-y l'adresse IP (ou le nom d'hôte) de votre machine cible (web01).
   [webservers]
   web01 ansible_host=votre_ip_ou_hostname

2. Vérifiez la connectivité en utilisant la commande ad-hoc ping sur le groupe webservers.
   `ansible webservers -m ping`
3. Vérifiez l'espace disque sur toutes les machines du groupe webservers en utilisant le module shell.
   `ansible webservers -m shell -a "df -h"`

Exercice 2 : Le Premier Playbook (Basic Tasks)

Objectifs

- Écrire un Playbook.
- Utiliser les modules apt (ou yum) et service.
- S'assurer que le service est démarré et activé au démarrage.

Étapes

1. **Créez un Playbook nommé deploy_nginx.yml**.
2. Ajoutez un play ciblant le groupe webservers.
3. Ajoutez une première tâche pour **installer le paquet nginx** en utilisant le module apt (ou yum si vous êtes sur Red Hat/CentOS). Assurez-vous que state: present.
4. Ajoutez une deuxième tâche pour **démarrer le service nginx** et le configurer pour qu'il se lance au boot (en utilisant le module ansible.builtin.service avec state: started et enabled: yes).

Exécution

`ansible-playbook deploy_nginx.yml`
Vérifiez que Nginx est bien accessible sur votre machine cible.

Exercice 3 : Configuration Dynamique (Variables & Templates)

Objectifs

- Définir et utiliser des variables.
- Déployer un fichier de configuration dynamique en utilisant le module template (Jinja2).

Étapes

1. **Créez un dossier vars** et ajoutez un fichier vars/main.yml. Définissez-y la variable message_accueil: "Bienvenue sur le Serveur Web IaC!".
2. **Créez le dossier templates** et un fichier templates/index.html.j2 (le template Jinja2).
3. Dans ce template, utilisez la variable définie :
   ```
   <!-- templates/index.html.j2 -->
   <h1>{{ message_accueil }}</h1>
   <p>Ce serveur est configuré par Ansible.</p>
   ```

4. **Modifiez votre Playbook deploy_nginx.yml :**
   - Chargez le fichier vars/main.yml.
   - Ajoutez une nouvelle tâche qui utilise le module template pour copier index.html.j2 vers l'emplacement par défaut de la page d'accueil Nginx (souvent /var/www/html/index.html).

Exécution

`ansible-playbook deploy_nginx.yml`
Vérifiez que la page d'accueil du serveur Nginx affiche votre message personnalisé.

Exercice 4 : Gestionnaires de Changement (Handlers)

Objectifs

- Comprendre le principe d'idempotence et d'optimisation.
- Utiliser un handler pour redémarrer le service uniquement en cas de besoin.

Étapes

1. **Modifiez votre Playbook deploy_nginx.yml pour y inclure une section handlers**.
2. Définissez un handler nommé restart nginx qui utilise le module service pour redémarrer Nginx.
3. Dans la tâche de déploiement du template (Exercice 3), ajoutez la directive notify: restart nginx.
4. **Supprimez la tâche de démarrage du service** que vous aviez à la fin du Playbook (car nous utiliserons désormais le Handler).

Tests d'Idempotence

1. Exécutez le Playbook une première fois : Le Template est copié, le Handler est notifié, Nginx redémarre.
2. Exécutez le Playbook une deuxième fois sans changer le Template : La tâche de Template signale ok (inchangé), le Handler n'est **pas** notifié, Nginx ne redémarre pas.
3. Modifiez le message_accueil dans vars/main.yml. Exécutez le Playbook : La tâche de Template signale changed, le Handler est notifié, Nginx redémarre.

Exercice 5 : Structuration en Rôle (Roles)

Objectifs

- Organiser l'ensemble de votre configuration Nginx sous forme de Rôle réutilisable.

Étapes

1. Créez la structure de base du Rôle webserver :
   `ansible-galaxy init webserver`
2. **Migrez les fichiers et tâches :**
   - Déplacez le contenu de vars/main.yml dans webserver/vars/main.yml (ou webserver/defaults/main.yml si vous souhaitez que les valeurs soient facilement surchargeables).
   - Déplacez le contenu du dossier templates dans webserver/templates.
   - Déplacez les tâches (install, deploy_template) de votre Playbook vers webserver/tasks/main.yml.
   - Déplacez le handler vers webserver/handlers/main.yml.
3. **Simplifiez votre Playbook deploy_nginx.yml :** Il doit maintenant être très court et faire uniquement référence au Rôle.
   ```
   # deploy_nginx.yml
   ```

```
- name: Déploiement du serveur web avec Role
  hosts: webservers
  roles:
    - webserver
```

Exécution

`ansible-playbook deploy_nginx.yml`
Le résultat doit être identique à l'exercice précédent, mais le code est désormais modulaire et organisé.

Exercice 6 : Sécurité et Secrets (Ansible Vault)

Objectifs

- Sécuriser une donnée sensible (un mot de passe ou une clé d'API) en utilisant Ansible Vault.

Scénario

Nous ajoutons un utilisateur administrateur avec un mot de passe chiffré.

Étapes

1. Créez un fichier de secrets chiffré (qui sera chargé par le rôle) :
   `ansible-vault create webserver/vars/secrets.yml`
   Entrez un mot de passe pour le Vault et ajoutez le contenu suivant :
   ```
   # webserver/vars/secrets.yml (chiffré)
   admin_password: "motdepasseTRESsecretpourlutilisateur"
   ```

2. **Modifiez webserver/tasks/main.yml** pour inclure le fichier secrets.yml.
   ```
   # Dans webserver/tasks/main.yml, en tête du fichier
   - name: Inclure les variables secrètes
     ansible.builtin.include_vars:
       file: secrets.yml
   ```

3. **Ajoutez une tâche** pour créer un utilisateur admin_iac avec le mot de passe chiffré (vous utiliserez le module user avec l'option password, nécessitant le mot de passe haché ou un module spécifique pour le hachage).

Exécution

Lancez le Playbook en fournissant le mot de passe du Vault.
`ansible-playbook deploy_nginx.yml --ask-vault-pass`
Vérifiez dans les logs que le mot de passe chiffré a été utilisé pour la création de l'utilisateur.