Lab Guidé sur les Rôles Ansible avec des Exemples

Introduction

Ansible est un outil de gestion de configuration simple et puissant qui permet d'automatiser la gestion de serveurs. Les **rôles Ansible** permettent d'organiser et de structurer les playbooks pour les rendre modulaires et réutilisables. Ce lab vous guide à travers la création et l'utilisation de rôles pour configurer un environnement sur trois hôtes gérés (**host01**, **host02**, et **host03**) depuis un master (**master01**).

Pré-requis

- Ansible est installé sur **master01**.
- Les hôtes à gérer (**host01**, **host02**, **host03**) sont accessibles via SSH avec des clés configurées.
- Python est installé sur les hôtes.
- Inventaire initial :

```
# /etc/ansible/hosts
[web_servers]
host01 ansible_user=root

[db_servers]
host02 ansible_user=root
host03 ansible_user=root
```

Étape 1 : Créer la structure d'un rôle

1. Créez un projet de test :

```
mkdir -p /home/ansible-lab/roles
cd /home/ansible-lab/roles
ansible-galaxy init nginx_setup
```

Cette commande crée la structure suivante :

```
nginx_setup/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Étape 2 : Configurer le rôle

2.1 Définir les variables par défaut

- Modifiez defaults/main.yml pour ajouter des variables :

```
nginx_version: "1.20.2"
nginx_port: 80
```

2.2 Ajouter des tâches

- Éditez tasks/main.yml pour définir les tâches du rôle :

```
---
- name: Installer les pré-requis
  ansible.builtin.yum:
    name: epel-release
    state: present

- name: Installer NGINX
  ansible.builtin.yum:
    name: "nginx-{{ nginx_version }}"
    state: present

- name: Démarrer et activer NGINX
  ansible.builtin.service:
    name: nginx
    state: started
    enabled: true

- name: Configurer le fichier de configuration
  ansible.builtin.template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    notify: Restart NGINX
```

2.3 Créer un fichier de modèle

- Ajoutez un fichier templates/nginx.conf.j2 :

```
user nginx;
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    server {
        listen {{ nginx_port }};
        server_name localhost;

        location / {
            root /usr/share/nginx/html;
            index index.html;
        }
    }
}
```

2.4 Définir les handlers

- Modifiez handlers/main.yml pour gérer les redémarrages :

```
---
- name: Restart NGINX
  ansible.builtin.service:
    name: nginx
    state: restarted
```

Étape 3 : Utiliser le rôle dans un playbook

- Créez un fichier site.yml pour inclure le rôle :

```
---
- name: Déployer NGINX sur les serveurs web
  hosts: web_servers
  roles:
    - nginx_setup
```

Étape 4 : Tester le rôle

1. Exécutez le playbook :

```
ansible-playbook -i /etc/ansible/hosts /home/ansible-lab/site.yml
```

2. Vérifiez si NGINX est correctement installé :

```
curl http://host01
```

Étape 5 : Étendre avec un autre rôle

- Créez un deuxième rôle pour configurer MariaDB :

```
ansible-galaxy init mariadb_setup
```

- Ajoutez les tâches pour installer et configurer MariaDB dans mariadb_setup/tasks/main.yml :

```
---
- name: Installer MariaDB
  ansible.builtin.yum:
    name: mariadb-server
    state: present

- name: Démarrer et activer MariaDB
  ansible.builtin.service:
    name: mariadb
    state: started
    enabled: true
```

- Modifiez site.yml pour inclure les deux rôles :

```
---
- name: Déployer NGINX sur les serveurs web
  hosts: web_servers
  roles:
    - nginx_setup

- name: Configurer MariaDB sur les serveurs DB
  hosts: db_servers
  roles:
    - mariadb_setup
```