Série d'Exercices Avancés : Ansible - Load Balancing Dynamique

Ce parcours d'exercices se concentre sur l'exploitation des **facts** (informations collectées sur les hôtes) et la puissance des **templates Jinja2** pour réaliser une configuration dynamique de répartiteur de charge.

Scénario Évolutif : Répartiteur de Charge Frontal

Nous allons :

1. Déployer 3 serveurs web (web01, web02, web03) dont la page d'accueil affiche leur nom d'hôte et leur IP.
2. Déployer un serveur frontal unique (lb01) qui fera du Round Robin vers ces 3 serveurs web.
3. Automatiser la liste des serveurs cibles pour que le Load Balancer se mette à jour sans modification manuelle du code.

Prérequis :

- Les exercices de la première série (Rôles, Playbooks) sont maîtrisés.
- Quatre machines cibles accessibles via SSH (nommées par exemple web01, web02, web03, et lb01).
- Un rôle webserver (créé lors de l'Exercice 5) prêt à être réutilisé.

Exercice 1 : Environnement Multi-Nœuds et Collecte des Facts

Objectifs

- Définir un inventaire multi-groupes.
- S'assurer que les "Facts" sont collectés sur tous les hôtes cibles pour connaître leurs adresses IP.

Étapes

1. **Modifiez votre fichier inventory.ini** pour inclure le Load Balancer et les trois serveurs Web :
   ```
   # inventory.ini
   [webservers]
   web01 ansible_host=adresse_ip_web_1
   web02 ansible_host=adresse_ip_web_2
   web03 ansible_host=adresse_ip_web_3
   ```

```
[loadbalancers]
lb01 ansible_host=adresse_ip_lb

[all_servers:children]
webservers
loadbalancers
```

2. Testez la collecte de Facts en affichant l'adresse IP de web01 depuis la machine de contrôle (vous devriez voir une liste très longue d'informations collectées).
   $$ \text{ansible web01 -m setup -a "filter=ansible_default_ipv4" } $$

Exercice 2 : Templating Avancé (Personnalisation par Nœud)

Objectifs

- Utiliser les "Facts" Ansible pour personnaliser le contenu sur chaque nœud.
- Réexécuter le rôle webserver sur les trois hôtes.

Étapes

1. **Modifiez le template Jinja2 existant** (webserver/templates/index.html.j2 ou équivalent) pour qu'il affiche dynamiquement le nom d'hôte et l'adresse IP de la machine :
   ```
   <!-- webserver/templates/index.html.j2 -->
   <!DOCTYPE html>
   <html lang="fr">
   <head><title>Serveur Web Ansible</title></head>
   <body>
     <h1>Bienvenue sur {{ message_accueil }}</h1>
     <p>Ce contenu est servi par :</p>
     <ul>
       <li>Nom du Serveur (Hostname) : <strong>{{ inventory_hostname }}</strong></li>
       <li>Adresse IP : <strong>{{ ansible_default_ipv4.address }}</strong></li>
     </ul>
   </body>
   </html>
   ```
   - **Note :** inventory_hostname est une variable Ansible standard. ansible_default_ipv4.address est un "Fact" collecté.
2. Exécutez le Playbook qui cible le rôle webserver (il devrait maintenant cibler webservers).
   $$ \text{ansible-playbook deploy_nginx.yml} $$
3. **Vérifiez** en accédant aux adresses IP de web01, web02 et web03 que chacun affiche correctement ses propres informations.

Exercice 3 : Déploiement du Load Balancer (Configuration Statique)

Objectifs

- Créer un nouveau rôle pour le Load Balancer.
- Déployer Nginx sur le serveur frontal (lb01) et le configurer manuellement pour le Round Robin.

Étapes

1. Créez un nouveau rôle nommé loadbalancer :
   $$ \text{ansible-galaxy init loadbalancer} $$
2. **Créez un Playbook deploy_lb.yml** qui exécute le rôle loadbalancer uniquement sur le groupe loadbalancers.
   ```
   # deploy_lb.yml
   - name: Déploiement du Load Balancer
     hosts: loadbalancers
     become: yes
     roles:
       - loadbalancer
   ```

3. **Dans loadbalancer/templates/nginx.conf.j2, créez le template Nginx pour la répartition de charge :**
   - Incluez la section upstream pointant statiquement vers les adresses IP de web01, web02 et web03 (utilisez les adresses réelles pour le moment).

   ```
   # Exemple de contenu à mettre dans loadbalancer/templates/nginx.conf.j2
   upstream backend_servers {
       # Attention : ces IPs sont codées en dur pour cet exercice !
       server IP_DE_WEB01:80;
       server IP_DE_WEB02:80;
       server IP_DE_WEB03:80;
   }

   server {
       listen 80;
       location / {
           proxy_pass http://backend_servers;
       }
   }
   # ... autres configurations Nginx (installation, service, handler)
   ```

4. **Complétez le rôle loadbalancer** avec les tâches nécessaires pour installer Nginx, copier le template nginx.conf.j2 (vers /etc/nginx/sites-available/default par exemple), et notifier le

redémarrage de Nginx.

Exécution

$$ \text{ansible-playbook deploy_lb.yml} $$
Vérifiez que l'accès à l'IP du Load Balancer (lb01) affiche les pages de web01, web02, web03 en alternance (Round Robin).

Exercice 4 : Configuration Dynamique du Load Balancer (Magie Jinja2)

Objectifs

- Rendre la configuration upstream entièrement dynamique.
- Utiliser les variables groups et hostvars pour itérer sur les membres d'un groupe et récupérer leurs Facts.

Étapes

1. **Modifiez le template Nginx du Load Balancer** (loadbalancer/templates/nginx.conf.j2) pour remplacer la liste d'IP codées en dur par une boucle Jinja2 :
   ```
   # loadbalancer/templates/nginx.conf.j2 (Version dynamique)
   upstream backend_servers {
      {% for host in groups['webservers'] %}
      server {{ hostvars[host]['ansible_default_ipv4']['address'] }}:80;
      {% endfor %}
   }

   server {
      listen 80;
      location / {
         proxy_pass http://backend_servers;
      }
   }
   ```
   - **Explication :**
     - groups['webservers'] est la liste des noms d'hôtes du groupe webservers.
     - hostvars[host] est un dictionnaire qui contient tous les Facts (y compris l'IP) de l'hôte spécifié.
2. **Exécutez le Playbook deploy_lb.yml**.

Vérification

1. Vérifiez le fichier /etc/nginx/sites-available/default sur le serveur lb01 pour confirmer que la boucle Jinja2 a généré la liste correcte des adresses IP.
2. Si vous ajoutez un cinquième serveur (web04) à l'inventaire dans le groupe [webservers],

la simple réexécution du Playbook deploy_lb.yml devrait automatiquement le faire apparaître dans la configuration du Load Balancer.

Exercice 5 : Bonus - Provisioning et Configuration (Intégration IaC)

Objectifs

- Combiner les rôles et les Playbooks pour un déploiement complet en une seule commande.

Étapes

1. **Créez un Playbook maître full_deployment.yml** qui orchestre toutes les étapes.
2. Ce Playbook doit contenir deux Plays :
    - **Play 1 :** Cibler le groupe webservers et exécuter le rôle webserver.
    - **Play 2 :** Cibler le groupe loadbalancers et exécuter le rôle loadbalancer.

Exécution

$$ \text{ansible-playbook full_deployment.yml} $$
Une seule commande met à jour tous les serveurs Web et le Load Balancer, prouvant la puissance de l'orchestration complète avec Ansible.