TP 5 – Les conditions, Handlers et boucles

Objectif

Les trois fonctionnalités fondamentales d'Ansible sont les suivantes:

- Conditions
- Gestionnaires
- Boucles

Les conditions

Ansible peut utiliser des conditions pour exécuter des tâches ou des plays lorsque certaines conditions sont remplies.

Pour implémenter un conditionnel, l'instruction `when` doit être utilisée, suivie de la condition à tester. La condition est exprimée en utilisant l'un des opérateurs disponibles ci-dessous:

\==	Compare deux objets pour l'égalité.
! =	Compare deux objets pour l'inégalité.
>	true si le côté gauche est supérieur au côté droit.
> =	true si le côté gauche est supérieur ou égal au côté droit.
<	true si le côté gauche est plus bas que le côté droit.
< =	true si le côté gauche est inférieur ou égal au côté droit.

Nous allons installer un serveur FTP, mais uniquement sur les hôtes qui se trouvent dans le groupe d'inventaire "ftpserver".

- Pour ce faire, modifiez d'abord l'inventaire pour ajouter le groupe ftpserver et placez `node2` dedans.
- Créez ensuite le fichier `ftpserver.yml` sur votre hôte de contrôle ansible dans le répertoire `~/ansible-files/`:

```
---
- name: Install vsftpd on ftpservers
  hosts: all
  become: yes
  tasks:
  - name: Install FTP server when host in ftpserver group
    apt:
      name: vsftpd
      state: latest
    when: inventory_hostname in groups["ftpserver"]
```

- Exécutez-le et examinez la sortie.

Les handlers

Parfois, lorsqu'une tâche apporte une modification au système, une ou plusieurs tâches supplémentaires peuvent devoir être exécutées. Par exemple, une modification du fichier de configuration d'un service peut alors nécessiter le redémarrage du service pour que la configuration modifiée prenne effet.

Ici, les handlers d'Ansible entrent en jeu. Les handlers peuvent être considérés comme des tâches inactives qui ne sont déclenchées que lorsqu'elles sont explicitement invoquées à l'aide de l'instruction "notify".

Ecrivez un Playbook qui:

- gère les fichier de configuration d'Apache `/etc/apache2/` sur tous les hôtes du groupe web.
- redémarre Apache lorsque le fichier a changé

- Vous avez d'abord besoin du fichier qu'Ansible déploiera, prenez simplement celui de node1.

```
scp vagrant@192.168.201.11:/etc/apache2/ports.conf ~/ansible-files/files/
```

- Ensuite, créez le Playbook `apache2_conf.yml`. Assurez-vous que vous êtes dans le répertoire `~/ansible-files`.

```
---
- name: manage httpd.conf
  hosts: web
  become: yes
  tasks:
  - name: Copy Apache ports configuration file
    copy:
      src: ports.conf
      dest: /etc/apache2
    notify:
      - restart_apache
  handlers:
  - name: restart_apache
    service:
      name: apache2
      state: restarted
```

- La section “notifier” n’appelle le handler que lorsque la tâche de copie modifie réellement le fichier. De cette façon, le service est redémarré uniquement si nécessaire - et non à chaque exécution du playbook.

- La section “handlers” définit une tâche qui n’est exécutée que sur notification.

- Exécutez le Playbook. Nous n’avons encore rien modifié dans le fichier, donc il ne devrait pas y avoir de lignes “changed” dans la sortie et bien sûr le handler n’a pas dû se déclencher.

- Maintenant, changez la ligne `Listen 80` dans `ports.conf` en :

```
Listen 8080
```

- Exécutez à nouveau le Playbook. Maintenant, la sortie d’Ansible devrait être beaucoup plus intéressante:

  - `ports.conf` a dû être copié

  - Le handler a dû redémarrer Apache

```
http://192.168.201.11:8080/index.html
```

  Apache devrait maintenant écouter sur le port 8080.

- N’hésitez pas à modifier à nouveau le fichier ports.conf et à exécuter le playbook.

Les boucles simples

- Pour montrer la fonction des boucles, vous allez générer trois nouveaux utilisateurs sur `node2`. Pour cela, créez le fichier `loop_users.yml` dans `~/ansible-files` sur votre nœud de contrôle. Utilisez le module `user` pour générer les comptes utilisateurs.

```
---
- name: Ensure users
  hosts: node2
  become: yes
  tasks:
  - name: Ensure three users are present
    user:
      name: "{{ item }}"
      state: present
    loop:
    - dev_user
    - qa_user
    - prod_user
```

- Les noms ne sont pas fournis directement au module utilisateur. Au lieu de cela, il n'y a qu'une variable appelée `{{item}}` pour le paramètre `name`.
- Le mot clé `loop` répertorie les noms d'utilisateur réels. Ceux-ci remplacent le `{{item}}` pendant l'exécution réelle du playbook.

- Pendant l'exécution, la tâche n'est répertoriée qu'une seule fois, mais trois modifications sont répertoriées en dessous.

Les boucles complexes

Les boucles peuvent également se trouver sur des listes. Imaginez que les utilisateurs doivent être affectés à différents groupes supplémentaires:

```
    - username: dev_user
      groups: ftp
    - username: qa_user
      groups: ftp
    - username: prod_user
      groups: www-data
```

Le module `user` a le paramètre optionnel `groups` pour l'assigner à un groupe.

- Réécréez le playbook pour créer les utilisateurs avec des droits d'utilisateur supplémentaires:

```
---
- name: Ensure users
  hosts: node2
  become: yes

  tasks:
  - name: Ensure three users are present
    user:
      name: "{{ item.username }}"
      state: present
      groups: "{{ item.groups }}"
    loop:
    - { username: 'dev_user', groups: 'ftp' }
    - { username: 'qa_user', groups: 'ftp' }
    - { username: 'prod_user', groups: 'www-data' }
```

- Vérifiez la sortie.
- Vérifiez que l'utilisateur `dev_user a bien été créé sur node2 :`

```
ansible node2 -m command -a "id dev_user"
```