Création d'un Role Ansible

ANSIBLE

Kévin MOUKIN
M2 Informatique
2018-2019

Contents

1 Sudo

Lorsque l'on exporte une clé ssh afin de se connecter automatiquement à une machine distante pour utiliser les commandes sudo, il est nécessaire de saisir le mot de passe sudo. Il est possible d'ajouter la clé public du client shh à la machine cible en temps que root pour permettre l'utilisation des commande sudo sans mot de passe, Cependant cette technique n'est pas conventionnelle. Ou encore il est possible de changer le groupe de l'utilisateur en le métant sudo.

2 Ansible

2.1 Playbook

Un playbook est un script écrit en YAML qui permet de configurer un serveur. Il décrit les tâches que Ansible doit acomplir sur le seveur.

2.2 Role

Un role est un playbook structurer pour répondre à une utilisation plus large et réutilisable.

3 Configuration du serveur Ansible

3.1 Création clé ssh

On crée une clé ssh afin de pouvoir se connecter sans mot de passe à notre serveur cible.

```
ssh-keygen -t rsa
```

Puis on copie la clé fraichement créée sur notre serveur cible.

```
ssh-copy-id -i ~.ssh/key.pub user@host
```

3.2 Installation

Pour installer Ansible on utlise la commande.

```
sudo apt-get install ansible
```

3.3 Configuration Hosts

Le fichier Hosts permet de renseigner toutes les machines qu' Ansible doit gérer, il n'est pas possible de déployer sur une machine si son nom ou son adresse ip n'est pas renseigné. Il faut donc renseigner l'adresse ip de la machine que l'on va utiliser pour recevoir le déploiement de Ansible.

```
/etc/ansible/hosts
```

Une fois ceci fait on teste la communication d'ansible avec les serveurs cibles.

```
ansible all -m ping
```

4 Création d'un Role

4.1 Playbook

Nous avons commencé par créer un playbook que l'on a appellé script.yml. Dans ce playbook nous avons renseigné les clients grace à hosts, le nom d'utilisateur ssh grace à remote_user, le nom de l'utilisateur sur la machine distante avec become_user , la méthode utilisé avec become_method et le type connexion avec connection.

```
- hosts: all
  remote_user: username
  become: true
  become_user: root
  become_method: su
  connection: ssh
```

Cette partie permet la connection avec les clients.

Nous avons ensuite ajouté les dépots grace a apt_repository qui nécéssite l'instation du paquet python_apt sur client.

```
 - name: Add  Repo
   apt_repository:
       repo: deb address_repository version_repository
       state: present
```

Certain dépot nécéssite une clé de valition il donc parfois obligatoire de l'ajouter avant d'ajouter le depot.

```
- name : Add DotDeb Key
  apt_key:
      url: address_key
```

Une fois ceci fait on déploie le script en utilisant la commande:

```
ansible-playbook script.yml -K
```

L'attribut -K permet de saisir de le mot de passe afin de pouvoir installer les dépots.

4.2 Role

On a créer un role que l'on a nommé repository avec la commande `ansible-galaxy init -p roles/nom_du_role`

Nous avons modifié notre script YAML en lui iniquant uniquement le role a appeller

```
roles:
        - role: repository
          become: true
```

Afin de rendre notre script réutilisable nous avons rajouter dans le fichier */etc/ansible/hosts* des variables pour le nom de l'utilisateur, le type de connexion et la méthode, auparevent nous avions directement ses informations dans notre Playbook.

```
Host ansible_user=user_ssh ansible_become_user=root  ansible_become_method=su ansible_conn
```

Dans le dossier *roles/repository* on retrouve plusieurs dossiers c'est le dossier *tasks* qui nous intéresse c'est là que l'on a mis nos différentes tâches.

Dans *roles/repository/tasks/main.yml* nous avons listé nos différentes tâches.

Afin de différiencier les sytèmes Ubuntu et Debian nous avons utilisé l'attribut *ansible_ distribution*

```
  when: ansible_distribution == "Nom_de_la_distribution"
```

A l'aide cet attribut nous avons pu determiner la distribution et appliquer un script spécifique à chaque distributions.