Ansible : outil d'automatisation

Contributeurs

(SISR2-2016) Anthony Varin, Alexandre Saligny

Mise a jour

par Valentin Pepin, Benjamin Hohn (SISR2-2020)

Ansible est un outil d'automatisation de tâches (déploiement de logiciels et mise à jours)

Prérequis

Outils:

- Machine sous **Debian** avec le paquet **ssh** installé pour pouvoir gérer l'établissement d'une connexion distante récurrente
- Cibles avec ssh accessible en root

Remarque:

- la version de **Ansible** décrite dans la procédure est la version **1.7**. Certains éléments sont à modifier pour des versions supérieures.

Procédures

Installation de Ansible

1. Ajouter le dépôt correspondant et la clé pour ***apt*** :

```
echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main"
>> /etc/apt/sources.list
apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
93C4A3FD7BB9C367
```

2. Mettre à jour APT :

```
apt update
```

3. Installation du paquet **Ansible**

```
 apt install ansible
```

Configuration des Hôtes

Pour enregistrer les différente machine cible (hôte) on modifie le fichier **/etc/ansible/hosts** en ajoutant **a la fin** du fichier les adresse IP ou les FQDN des hôtes, rassemblés par [groupes] :

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
...
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

#green.example.com
...

#db-[99:101]-node.example.com

[nom du groupe]
ip machine
ip machine
```

Une machine peut apparaître dans plusieurs [groupes].

Échanges de clés SSH

Pour pouvoir entrer en contact avec les hôtes cible, on établira une connexion récurrente grâce à SSH :

1. Créer une clé privée/publique sur le serveur Ansible

   ```
   ssh-keygen –t {rsa|dsa} [-b <nombre_bits>]
   ```

   - choisir le type de chiffrement de la clé : RSA ou DSA
   - <nombre_bits> est une puissance de 2 précisant la complexité de la clé (1024, 2048, etc)
2. Echanger la clé publique avec les hôtes cible :

   ```
   ssh-copy-id -i <chemin/clé_publique> <compte>@<IP_ou_FQDN_cible>
   ```

   - Pour choisir un numero de port on peut rajouter -p <numero de port> après ssh-copy-id
   - la clé publique générée par *ssh-keygen* se trouve par défaut dans ***/root/.ssh/*** sous le nom ***id_rsa.pub***.
   - Si vous arrivez pas à accéder au serveur cible faite ***nano /etc/ssh/sshd_config*** Chercher la ligne ***PermitRootLogin*** puis dé-commenter mettre ***yes*** a la fin de la ligne, puis réessayer.

Test de connexion

Pour tester la connexion entre ansible et les machines cibles on utilise la commande:

```
ansible all -m ping -u <nom du compte utilisé par le ssh>
```

Si la communication est établie, on obtient:

```
ipmachine | success >> {
    "changed": false,
    "ping": "pong"
}
```

Playbook

Création d'un script

Il faut créer un dossier **playbook** dans le dossier **/etc/ansible** pour pouvoir stocker un script

Structure d'un script

Crée un fichier YAML **<nom-du-fichier>.yml**

```
- hosts: <Groupe, ip, nom>
  become: yes
  become_method: sudo

  tasks:
   - name: mise a jour des serveurs
     apt: update_cache=yes
```

Création fichier crypte

Dans cette version, il faudra :

- disposer l'éditeur de texte **vim** (à installer éventuellement)
- le mot de passe pour basculer en *sudo* sur la machine distante –> soit *mpsudo* cette valeur
- le mot de passe pour décrypter le fichier sécurisé –> soit *mpcrypt* cette valeur

Procédure : 1. Créer un fichier crypté stockant le mot de passe mpsudo<script .sh>:

```
ansible-vault create <nom du fichier.sh>
```

Le mot de passe mpcrypt vous sera demandé pour la création de ce fichier.

2. Enregistrer ce mot de passe mpcrypt dans un fichier .txt avec droit d'écriture et de lecture seulement pour le compte root.

3. Déployer le script

```
ansible-playbook <nom du script.yml> -u root --vault-password-file=<nom du fichier txt contenant le mot de passe crypt>
```

Déploiement de script crypte

```
ansible-playbook <nom du script> -u root --ask-sudo-pass
```

Automatisation pour l'exécution de plusieurs *playbook*

Le script suivant permet d'exécuter une liste de *playbook* à la suite

```
#script nommé autoPlaybooks.sh
ansible-playbook $1 --vault-password-file=<nom du fichier contenant le mot de passe du fichier crypté>
ansible-playbook $2 --vault-password-file=<nom du fichier contenant le mot de passe du fichier crypté>
...
ansible-playbook $x --vault-password-file=<nom du fichier contenant le mot de passe du fichier crypté>
```

<script .sh> Pour exécuter le script *autoPlaybooks.sh*:

```
./autoPlaybooks.sh <playbook1.yml> <playbook2.yml> ...
```