Lab 1 – Installation

Configuration requise

Machine de contrôle

Vous pouvez exécuter **Ansible** sur n'importe quelle machine sur laquelle Python 2.6 ou 2.7 est installé (Windows n'est pas pris en charge pour la machine de contrôle).

Ansible prend en charge RedHat, Debian, CentOS, OS X, tous les BSD.

Nœuds clients

Les machines clientes doivent au moins avoir Python 2 (version 2.6 ou ultérieure) ou Python 3 (version 3.5 ou ultérieure)

Si SELinux est activé sur les nœuds distants, vous devrez installer le package `libselinux-python` sur les nœuds avant d'utiliser toute fonction liée à `copy` / `file` / `template` dans Ansible

Environnement

Nom d'hôte	Adresse IP	OS	Objectif
master.dev	192.168.60.100	CentOS 7 / Ubuntu 18.04	contrôle
app1.dev	192.168.60.4	CentOS 7	Nœud géré 1
app2.dev	192.168.60.5	CentOS 7	Nœud géré 2
db.dev	192.168.60.6	CentOS 7	Nœud géré 3

Authentification SSH

Comme indiqué précédemment, Ansible utilise *OpenSSH* pour la communication à distance. Ansible prend en charge l'authentification sans **mot de passe** et par **mot de passe** pour exécuter des commandes sur les nœuds gérés.

Authentification par mot de passe

L'authentification par mot de passe peut être utilisée si nécessaire en fournissant l'option **--ask-pass**. Cette option nécessite **sshpass** sur la machine de contrôle.

```
### CentOS 7 / RHEL 7 et Fedora ###
```

```
yum install -y sshpass
```

```
### Ubuntu 18.04 / 16.04 & Debian 9 ###
```

```
sudo apt-get update
sudo apt-get install -y sshpass
```

Nom d'utilisateur du serveur Ansible = Nom d'utilisateur du nœud géré racine = `vagrant`

Testez l'accès aux noeuds gérés à partir du noeud *master*:

```
ssh vagrant@192.168.60.4
vagrant@192.168.60.4's password:
```

Tapez le mot de passe de l'utilisateur *vagrant*: `vagrant`

Vous serez directement transféré sur la machine app1, en remarquant votre nouveau prompt:

```
Last login: Sat Nov 13 09:15:15 2021 from 10.0.2.2
[vagrant@app1 ~]$
```

Pour quitter la machine *app1* et revenir sur votre machine *master*, utilisez la commande `exit`, ou la combinaison des touches **Ctrl-D**.

Faites la même procédure pour les autres noeuds *app2* et *db*.

Remarque :

Selon la configuration par défaut de votre système Centos, parfois l'accès en SSH clair est désactivé, vous aurez le message d'erreur suivant :

```
...

Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Pour autoriser l'accès par le pair login/mot de passe, éditez le fichier de configuration "**/etc/ssh/sshd_config**"

```
[vagrant@app1 ~]$ vi /etc/ssh/sshd_config
```

Puis change la directive **PasswordAuthentication** en **"Ye"**

```
...

# To disable tunneled clear text passwords, change to no here!

#PasswordAuthentication yes

#PermitEmptyPasswords no

PasswordAuthentication yes
```

Authentification par clé SSH (*authentification sans mot de passe*)

Lorsqu'il s'agit d'authentification **ssh**, par défaut, il utilise des clés ssh (authentification sans mot de passe) pour s'authentifier auprès de la machine distante.

SSH Configurer l'accès par paire de clès RSA

L'authentification par clé est le plus sécurisé de plusieurs modes d'authentification utilisables avec OpenSSH, tels que le mot de passe simple et les tickets Kerberos. L'authentification par clé présente plusieurs avantages par rapport à l'authentification par mot de passe, par exemple, les valeurs de clé sont nettement plus difficiles à forcer ou à deviner que les mots de passe simples, à condition que la longueur de la clé soit suffisante. Les autres méthodes d'authentification ne sont utilisées que dans des situations très spécifiques.

SSH peut utiliser des clés "RSA" (Rivest-Shamir-Adleman) ou "DSA" ("Digital Signature Algorithm").

L'authentification par clé utilise deux clés, une clé "publique" que tout le monde est autorisé à voir et une autre clé "privée" que seul le propriétaire est autorisé à voir. Pour communiquer en toute sécurité à l'aide de l'authentification par clé, il faut créer une paire de clés, stocker en toute sécurité la clé privée sur l'ordinateur à partir duquel on veut se connecter et stocker la clé publique sur l'ordinateur auquel on veut se connecter.

L'utilisation de connexions basées sur des clés avec ssh est généralement considérée comme plus sécurisée que l'utilisation de connexions par mot de passe simples.

Génération de clés RSA

La première étape consiste à créer un ensemble de clés RSA à utiliser dans l'authentification.

Cela devrait être fait sur le *master* à partir duquel vous allez gérer les autres noeuds.

Pour créer vos clés SSH publiques et privées sur la ligne de commande :

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

Vous serez invité à indiquer un emplacement pour enregistrer les clés et une phrase secrète pour les clés. Cette phrase secrète protégera votre clé privée pendant qu'elle est stockée sur le disque dur :

```
[vagrant@master ~]$ ssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagrant/.ssh/id_rsa.
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:tcrUXS2cOMwGMZWQ0xyUE4hAZAo56bBQKGFxci+Uq80 vagrant@master.dev
The key's randomart image is:
+---[RSA 2048]----+
|.B+*.o=. .=X==   |
|* Oo.o  . +=*o o |
|o+ oo.    ..*.= .|
|. ...    o + o . |
|        o        |
+----[SHA256]-----+
```

Toutes nos félicitations! Vous avez maintenant un jeu de clés. Il est maintenant temps de faire en sorte que vos systèmes vous permettent de vous connecter avec eux.

Votre phrase secrète (passphrase) de clé SSH est uniquement utilisée pour protéger votre clé privée contre les voleurs. Elle n'est jamais transmise sur Internet et la force de votre clé n'a rien à voir avec la force de votre phrase secrète.

Niveau de cryptage de clé

Remarque : la valeur par défaut est une clé de 2048 bits. Vous pouvez augmenter cela à 4096 bits avec l'indicateur -b (Augmenter les bits rend plus difficile le craquage de la clé par des méthodes de force brute).

```
ssh-keygen -t rsa -b 4096
```

Transférer la clé publique vers l'hôte géré

La clé que vous devez transférer à l'hôte est la clé publique. Si vous pouvez vous connecter à un ordinateur via SSH à l'aide d'un mot de passe, vous pouvez transférer votre clé RSA en procédant comme suit depuis votre propre ordinateur :

```
ssh-copy-id <nom d'utilisateur>@<hôte>
```

Où `<username>` et `<host>` doivent être remplacés par votre nom d'utilisateur et le nom de l'ordinateur vers lequel vous transférez votre clé.

```
[vagrant@master ~]$ ssh-copy-id vagrant@192.168.60.4

/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/vagrant/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
it is to install the new keys
vagrant@192.168.60.4's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'vagrant@192.168.60.4'"
and check to make sure that only the key(s) you wanted were added.
```

Une autre alternative consiste à copier le fichier de clé publique sur le serveur et à le concaténer manuellement dans le fichier `~/.ssh/authorized_keys`.

Tester l'accès SSH sans mot de passe

Vous pouvez vous assurer que cela a fonctionné en faisant :

```
ssh <nom d'utilisateur>@<hôte>
```

Une fois que vous avez configuré la communication sans mot de passe, vérifiez-la.

```
[vagrant@master ~]$ ssh 192.168.60.4
Last login: Sat Nov 13 09:35:39 2021 from 192.168.60.1
[vagrant@app1 ~]$
```

Vous devriez maintenant pouvoir vous connecter à la machine distante sans mot de passe.

Installer Ansible sur CentOS 7 / RHEL 7 / Ubuntu 18.04 / 20.04 et Debian 9

Configuration de la machine de contrôle

1. Pour installer Ansible, nous devrons ***activer le référentiel EPEL sur CentOS 7 / RHEL7*** .

```
### CentOS 7 ###

yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

### RHEL 7 ###

subscription-manager repos --enable rhel-7-server-ansible-2.6-rpms

### Ubuntu 18.04 / Ubuntu 20.04 ###

sudo apt-get update
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update

### Debian 9 ###

sudo apt-get install dirmngr
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
93C4A3FD7BB9C367
echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main" | sudo tee -a
/etc/apt/sources.list.d/ansible.list
sudo apt-get update
```

2. Installez Ansible.

```
### CentOS 7 / RHEL 7 & Fedora 28 ###

yum install -y ansible

### Ubuntu 18.04 / 20.04 & Debian 9 ###

sudo apt-get install -y ansible
```

Une fois Ansible installé, vérifiez la version d'Ansible en exécutant la commande ci-dessous.

```
ansible --version
```

Output:

```
ansible 2.9.25

config file = /etc/ansible/ansible.cfg

configured module search path =
[u'/home/vagrant/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']

ansible python module location =
/usr/lib/python2.7/site-packages/ansible

executable location = /usr/bin/ansible

python version = 2.7.5 (default, Apr 2 2020, 13:16:51) [GCC 4.8.5
20150623 (Red Hat 4.8.5-39)]
```

Configurer les nœuds gérés

Les machines clientes doivent au moins avoir Python 2 (version 2.6 ou ultérieure) ou Python 3 (version 3.5 ou ultérieure).

```
### CentOS 7 / RHEL 7 & Fedora ###

yum install -y python

### Ubuntu 18.04 / 16.04 & Debian 9 ###

sudo apt-get install -y python
```

SELinux (CentOS / RHEL / Fedora)

Vérifiez l'état de SELinux: `sestatus`

Si SELinux est activé sur les nœuds gérés, vous devrez installer le package ci-dessous sur les nœuds avant d'utiliser les fonctionnalités `copy` / `file` / `template` dans Ansible.

```
yum install -y libselinux-python
```

- Pour désactiver temporairement SELinux sur CentOS 7, exécutez : `sudo setenforce 0`
- Modifiez le fichier */etc/selinux/config* et mettez `SELINUX` à `disabled`
- Vérifiez-le en exécutant le `sestatus` et `getenforce` à nouveau

Créer un inventaire Ansible

Modifiez (ou créez) le fichier **/etc/ansible/hosts** . Ce fichier contient l'inventaire des hôtes distantes auxquelles Ansible se connectera via SSH pour les gérer.

```
sudo vi /etc/ansible/hosts
```

Mettez un ou plusieurs systèmes distants et groupez-les. Ici, on ajoute les machines au groupe **demo_servers**.

Les groupes sont utilisés pour classer les systèmes pour un usage particulier. Si vous ne spécifiez aucun groupe, ils agiront comme des hôtes non groupés.

```
[demo_servers]

192.168.60.4
192.168.60.5
192.168.60.6
```

Premières commandes

Il est maintenant temps de vérifier tous nos nœuds en faisant simplement un ping depuis la machine de contrôle, pour ce faire, nous utiliserons la commande **ansible** avec les options **-m** (chargement de module) et **all** (tous les serveurs).

 # all servers - Fonctionne lorsque le nom d'utilisateur du serveur et du client est le même (sans mot de passe)

```
ansible all -m ping
```

 # all servers - "vagrant" est l'utilisateur du nœud géré (sans mot de passe)

```
ansible all -u vagrant -m ping
```

OU

 # Only demo_servers group - "vagrant" est l'utilisateur du nœud géré (sans mot de passe)

```
ansible demo_servers -u vagrant -m ping
```

OU

 # Si vous utilisez l'authentification par mot de passe

```
ansible -m ping all -u vagrant --ask-pass
```

Output:

```
192.168.60.4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
192.168.60.5 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
```

```
    },
    "changed": false,
    "ping": "pong"
}
192.168.60.6 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Dans l'exemple ci-dessus, nous avons utilisé le module ping avec la commande **ansible** pour envoyer un **ping** au groupe d'hôtes distants.

De la même manière, nous pouvons utiliser différents modules avec la commande **ansible** , vous pouvez trouver les modules disponibles :

ici https://docs.ansible.com/ansible/latest/user_guide/modules_intro.html.

Exercice

Publier la clé publique sur tous les nœuds gérés et tester que le ping de Ansible passe sans l'option `--ask-pass`