Ansible - Installation et configuration

Ansible est une plate-forme logicielle libre pour la configuration et la gestion des ordinateurs qui combine le déploiement de logiciels et services, l'exécution de tâches ad-hoc, et la gestion de configuration.

De plus, Ansible ne nécessite pas d'installation de client sur les serveurs ciblés, la communication entre le serveur et les nœuds est rendue possible par connexion SSH sécurisée.

Nous allons déployer 4 VMs Centos7 avec n'importe que hyperviseur à votre disposition l'une sera le Management Node et les autres les Managed Nodes.

```
+-----------+----------------+-------------------+
| Container | Container Port |     Host Port     |
+-----------+----------------+-------------------+
| master01  |     22:2220    |    10.20.30.10    |
+-----------+----------------+-------------------+
|   host01  |     22:2221    |    10.20.30.11    |
+-----------+----------------+-------------------+
|   host02  |     22:2222    |    10.20.30.12    |
+-----------+----------------+-------------------+
|   host03  |     22:2223    |    10.20.30.13    |
+-----------+----------------+-------------------+
```

1. Préparation du Lab

1.1 Prérequis (master01)

Actuellement, Ansible peut être exécuté à partir de n'importe quelle machine sur laquelle Python 2 (version 2.7) ou Python 3 (version 3.5 et supérieures) est installé (l'interpréteur python est installé par défaut sur les machines Linux). Cela inclut Red Hat, Debian, CentOS, macOS, n'importe lequel des BSD, etc. Cependant, Windows n'est pas pris en charge pour le nœud de contrôle.

Clônez la VM Centos7 en y ajoutant une deuxième interface host-only.

Paramétrez le hostname

```
[root@localhost ~]# hostnamectl set-hostname master01
```

Configurez l'adresse IP : 10.20.30.10/24

```
[root@master01 ~]# nmcli c modify eth0 ipv4.method manual
[root@master01 ~]# nmcli c modify eth0 ipv4.addresses 10.20.30.10/24
[root@master01 ~]# nmcli c down eth0; nmcli c up eth0
```

Ajoutez le contenu suivant à /etc/hosts

```
[root@master01 ~]# cat /etc/hosts
...
10.20.30.10            master01
10.20.30.11            host01
10.20.30.12            host02
10.20.30.13            host03
```

1.2 Installation de Ansible sur master01

L'installation se tient sur une ligne de code :

```
[root@master01 ~]# yum -y install ansible
```

1.3 Les nœuds esclaves

Refaites les étapes de configuration du hostname, l'adresse IP et le fichier /etc/hosts précédentes en tenant compte de chaque hôte.

Sur les nœuds esclaves, vous avez besoin d'un moyen de communication SSH, le port SSH doit donc être ouvert. Vous avez également besoin de Python 2 (version 2.6 ou ultérieure) ou Python 3 (version 3.5 ou ultérieure).

Vérifiez que python est installé, sinon installez-le :

```
[root@host01 ~]# yum list installed python
```

Nous disposons maintenant de tous les logiciels nécessaires pour administrer nos serveurs via Ansible.

2. Créer un utilisateur normal (toutes les VMs)

Ceci est important car nous utiliserons cet utilisateur pour effectuer toutes les tâches liées à ansibles. Pour les besoins de ce Lab, je vais créer un utilisateur "ansible".

```
[root@master01 ~]# useradd ansible
[root@master01 ~]# echo ansible:password | chpasswd
```

Répétez la même chose **sur les nœuds gérés**, c'est-à-dire créez le même utilisateur sur tous vos hôtes gérés:

```
[root@host01 ~]# useradd ansible && echo ansible:password | chpasswd
```

3. Créer et distribuer des clés SSH aux nœuds gérés (master01)

Nous devons maintenant activer la connexion sans mot de passe entre notre nœud de contrôle et tous les hôtes gérés. Nous allons donc configurer la connexion basée sur la paire de clés à l'aide de ssh-keygen.

Connectez-vous ou basculez vers l'utilisateur «ansible» et exécutez-le *ssh-keygen* dans le format ci-dessous.

```
[root@master01 ~]# su - ansible
[ansible@master01 ~]$
```

La première étape consistant à générer la clé publique qui permettra la communication entre le serveur et les hôtes :

```
[ansible@master01 ~]$ ssh-keygen -t rsa -b 1024 -N '' -C "master key"
Generating publc/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id rsa.pub.
The key fingerprint is:
SHA256:xxIyEgjqZPe+XHoqUjHySqvz2y9NGsKQQVlraw8+Hjo master key
The key's randomart image is:
+---[RSA 1024]----+
|oooo.            |
|o.. ..           |
|.= +. o .        |
|*.ooo. o o       |
| +o+o.  S o      |
| .=o= .  o       |
|. += B .         |
|.Eoo* =.         |
|o+=ooB+          |
+----[SHA256]-----+
```

Cela créera une paire de clés publique et privée dans le répertoire personnel sous ~/.ssh/. Maintenant, copiez la clé publique sur les hôtes cible à gérer. Nous l'utilisons *ssh-copy-id* :

```
[ansible@master01 ~]$ ssh-copy-id ansible@master01
[ansible@master01 ~]$ ssh-copy-id ansible@host01
[ansible@master01 ~]$ ssh-copy-id ansible@host02
[ansible@master01 ~]$ ssh-copy-id ansible@host03
```

Test à effectuer :

```
[ansible@master01 ~]$ ssh host01
[ansible@host01 ~]$ exit
```

Nous avons donc pu nous connecter à notre hôte géré host01 sans aucun mot de passe, faites de même avec les autres hôtes.

4. Configurer l'escalade de privilèges à l'aide de sudo

Étant donné que notre utilisateur ansible aurait besoin d'une élévation de privilèges, nous créerons une nouvelle règle pour l'utilisateur ansible utilisant un nouveau fichier sous */etc/sudoers.d*

```
[root@master01 ~]# echo "ansible ALL=(ALL) NOPASSWD: ALL" >
/etc/sudoers.d/ansible
```

Ajoutez la même règle sur tous vos hôtes gérés

```
[root@host01 ~]# echo "ansible ALL=(ALL) NOPASSWD: ALL" >
/etc/sudoers.d/ansible
```

5. Ansible en mode ad hoc

Après avoir installé ansible, configurons ansible pour exécuter des commandes ad hoc.

Nous allons créer une «base» de projet sous notre répertoire personnel:

```
[ansible@master01 ~]$ mkdir base && cd base
```

5.1 Créer et configurer ansible.cfg

Le fichier de configuration pour Ansible peut exister dans plusieurs emplacements différents, nous allons créer un fichier *ansible.cfg* sous le projet base dans lequel nous identifions comment connecter des hôtes distants.

```
[ansible@master01 base]$ vim ansible.cfg
[defaults]
remote_user = ansible
ask_pass = false
host_key_checking = false
inventory = inventory.ini
log_path = base.log
```

5.2 Créer un fichier d'inventaire statique

L'inventaire contient une liste de noms d'hôte ou d'adresses IP. Bien que vous deviez éviter d'utiliser des adresses IP dans l'inventaire. Nous avons créé trois groupes d'hôtes :

```
[ansible@master01 base]$ vi inventory.ini
[masters_nodes]
master01

[hosts_nodes]
host0[1:3] ansible_user=ansible

[lab_nodes:children]
masters_nodes
hosts_nodes
```

Pour répertorier les hôtes correspondants à l'aide de notre fichier d'inventaire, utilisez la commande ci-dessous. Cela n'exécutera aucune commande sur les nœuds d'inventaire:

```
[ansible@master01 base]$ ansible all --list-hosts
  hosts (4):
    master01
    host01
    host02
    host03
```

5.3 Exécution de commandes ad hoc

Les commandes ad hoc dans Ansible sont utilisées pour effectuer des tâches ou des opérations qui sont nécessaires sur une base ad hoc, ou une seule fois, en fonction de l'exigence.

Ansible utilise plein de modules pour réaliser les taches nécessaires à l'administration des hôtes, l'option -m est suivie du nom du module désiré.

La plupart des modules sont **idempotents**, ce qui signifie qu'ils peuvent être exécutés plusieurs fois en toute sécurité, et si le système est déjà dans le bon état, ils ne feront rien.

Essayez un ping ansible sur les hôtes. Le ping ansible indique s'il peut se connecter ou non aux hôtes.

```
[ansible@master01 base]$ ansible host01 -m ping
host01 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

Essayez un ping Ansible sur tous les serveurs

```
[ansible@master01 base]$ ansible all -m ping
```

Dans l'exemple ci-dessous, nous vérifierons la mémoire disponible sur nos hôtes gérés à l'aide de la commande *free -m*. Pour lancer nos modules ou commandes sur un groupe d'hôtes, il suffit de remplacer l'option all par le nom de notre groupe *hosts_nodes* :

```
[ansible@master01 base]$ ansible hosts_nodes -m shell -a "free -m"
host03 | CHANGED | rc=0 >>
              total        used        free      shared  buff/cache   available
Mem:           3931         938        1543          34        1449        2695
Swap:          2047           0        2047
…
```

Vous pouvez aussi exécuter directement des commandes Linux sur Ansible, en utilisant tout simplement l'option -a. Par exemple :

```
[ansible@master01 base]$ ansible all -a "whereis python"
```

Certaines commandes ou modules peuvent nécessiter une élévation de privilèges. Par exemple si vous tentez d'exécuter la commande ci-dessous depuis le compte ansible vous obtiendrez une erreur Permission denied :

```
[ansible@master01 base]$ ansible host01 -a "grep ansible /etc/shadow"
host01 | FAILED | rc=2 >>
grep: /etc/shadow: Permission non accordéenon-zero return code
```

Dans le cas où vous auriez besoin d'une élévation de privilèges pour l'exécution d'une commande, utilisez alors l'option *--become* (cette option exploite les outils existants d'élévation de privilèges comme *sudo*) :

```
[ansible@master01 base]$ ansible host01 -a "grep ansible /etc/shadow" --become
host01 | CHANGED | rc=0 >>
ansible:$6$cGUmJ/.2w8qcDX$Jm
```

Tous les modules sont accessibles sur la documentation d'Ansible ou avec la commande

```
$ ansible-doc --list
```

5.4 Les modules avec les options

Certains modules ansible nécessitent des paramètres supplémentaires pour une utilisation efficiente du module en question.

Avec le module *copy*, nous allons essayer d'ajouter du contenu dans un fichier sur *host01*

```
[ansible@master01 base]$ ansible host01 -m copy -a "content='Hello, Je suis
Elies Jebri' dest=~/hello.txt"
host01 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
...
```

Un autre exemple où on souhaite créer un fichier vide nommé *test.txt* dans le dossier */home/ansible/* et ne lui fournir que des droits de lecture (0644) :

```
[ansible@master01 base]$ ansible all -m file -a
"path=/home/ansible/test.txt state=touch mode=0644"
host03 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "dest": "/home/ansible/test.txt",
    "gid": 1001,
    "group": "ansible",
    "mode": "0644",
    "owner": "ansible",
    "size": 0,
    "state": "file",
    "uid": 1001
}
…
```

La commande ci-dessous utilisera le module *lineinfile* pour créer le fichier *sudoers* pour Ansible et ajouter la ligne pour les autorisations *sudo* de l'utilisateur Ansible.

```
$ ansible host04 -m lineinfile -a "dest=/etc/sudoers.d/ansible \
line='ansible ALL=(ALL) NOPASSWD: ALL' create=yes state=present" \
-u root --ask-pass
```

5.5 Parallélisme

Par défaut, Ansible utilise seulement 5 processus simultanés. Cependant, si vous avez plus de hôtes que la valeur définie par défaut pour le nombre de processus, Ansible prendra ainsi un peu plus de temps pour traiter tous vos hôtes.

```
[ansible@master01 base]$ ansible-config list | grep -i -A3 DEFAULT_FORKS
DEFAULT_FORKS:
  default: 5
  description: Maximum number of forks Ansible will use to execute tasks on
  target hosts.
```

Vous pouvez augmenter le nombre de processus avec l'option -f. Dans cet exemple je vais utiliser 9 processus simultanés pour lancer le module ping :

```
[ansible@master01 base]$ ansible all -m ping -f 8
```

Si vous avez la puissance de traitement disponible et que vous souhaitez utiliser plus de forks, vous pouvez définir le nombre dans *ansible.cfg*:

```
[defaults]
forks = 30
```

5.6 Recueillir des informations (Facts) sur vos hôtes

Les Facts sont des propriétés système qui sont collectées par Ansible lors de son exécution sur un système distant. Les Facts contiennent des détails utiles tels que le stockage, la version (mineur et majeur) de l'OS, configuration du réseau sur un système cible. Ils peuvent être exportés vers un fichier en tant que type de rapport système, ou ils peuvent être utilisés pendant l'exécution d'un playbook Ansible pour conditionner l'exécution de vos tâches.

Voici la commande pour voir tous les Facts :

```
[ansible@master01 base]$ ansible all -m setup
```

Vous pouvez également filtrer la sortie pour afficher uniquement certains Facts en utilisant le paramètre *filter*. Dans cet exemple je vais récupérer des informations réseau de mes hosts :

```
[ansible@master01 base]$ ansible all -m setup -a 'filter=*ip*'
host02 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "10.20.30.12",
            "172.20.10.194"
        ],
        "ansible_all_ipv6_addresses": [],
        "ansible_default_ipv4": {
            "address": "172.20.10.194",
            "alias": "eth0",
            "broadcast": "172.20.10.255",
            "gateway": "172.20.10.2",
            "interface": "eth0",
            "macaddress": "00:0c:29:cb:4d:05",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "172.20.10.0",
```

```
            "type": "ether"
        },
        "ansible_default_ipv6": {},
        "ansible_fips": false,
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
...
```

5.7 Installation d'un service avec Ansible

Afin de ne pas être obligé de rajouter --become à chaque commande nous allons modifier le fichier *ansible.cfg* comme suit :

```
[ansible@master01 base]$ vim ansible.cfg
[defaults]
remote_user = ansible
host_key_checking = false
inventory = inventory.ini
log_path = base.log

[privilege escalation]
become = true
become method = sudo
become_user = root
become_ask_pass = false
```

Nous allons maintenant utiliser le module *yum* sur le nœud host02, le service à installer est nginx, s'il est déjà présent, Ansible ne fera rien, sinon, il installera le service. Nous aurons ainsi la commande suivante :

```
[ansible@master01 base]$ ansible host02 -m yum -a "name=nginx
state=present"
host02 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "changes": {
        "installed": [
            "nginx"
        ]
    },
...
```

5.8 Démarrage du service avec Ansible

Je demande à Ansible d'appliquer au nœud host02 le démarrage du service nginx :

```
[ansible@master01 base]$ ansible host02 -m service -a "name=nginx
state=started enabled=yes"

[ansible@master01 base]$ curl host02
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Welcome to CentOS</title>
```

5.9 Arrêter le service

```
[ansible@master01 base]$ ansible host02 -m systemd -a "name=nginx
state=stopped"
```

5.10 Retirer Nginx

```
[ansible@master01 base]$ ansible host02 -m yum -a "name=nginx state=absent"
```

5.11 Module raw

Il arrive parfois que Python ne soit pas installé sur un hôte; dans ce cas, vous pouvez utiliser le module: *raw*, qui permet de passer des commandes Ansible sans utiliser Python :

```
$ ansible -i inventory.ini -m raw -a "yum install -y python2" host03 --user
root --ask-pass
```

Le module *raw* est un ancien module, aujourd'hui il est préférable d'utiliser les modules *shell* ou *command.*

Dans la plupart des cas d'utilisation, les deux modules mènent au même objectif. Voici les principales différences entre ces modules.

- Avec le module Command, la commande sera exécutée sans passer par un shell. Par conséquent, certaines variables comme $HOME ne sont pas disponibles. Et aussi des opérations de streaming comme < , > , | et & ne fonctionnera pas.
- Le module Shell exécute une commande via un shell, par défaut /bin/sh. Cela peut être modifié avec l'option exécutable . La tuyauterie et la redirection sont disponibles à cet effet.
- Le module de commande est plus sécurisé, car il ne sera pas affecté par l'environnement de l'utilisateur.