TD 2 : Ansible

Objectif : Le but de ce TD est de bien comprendre les notions de base d'Ansible, à savoir les inventaires, playbook et les rôles.

Installation :

La première à chose à faire ça serait d'installer Ansible sur une VM qui tourne sur linux. Il existe trois façons pour installer Ansible : via le binaire, via pip ou bien directement avec le paquet de la distribution. On va privilégier une installation via pip depuis un environnement virtuel, pour cela il faut utiliser les commandes suivantes :

```
sudo apt install python3-pip
python3 -m venv myenv
source myenv/bin/activate
pip install ansible
```

Une installation avec le paquet de la distribution risque de vous donner une version non plus supportée d'Ansible. Vous devez avoir une version supérieure à 2.15.

Exercice 1 : Ansible CLI

Il est possible d'utiliser Ansible en ligne de commande pour des fins de test ou de collecte d'information. Dans cet exercice vous allez découvrir la commande ansible qui se présente comme suit :

```
ansible -i "node2," all -m ping -k
```

L'option `-i` permet de préciser l'inventaire (les machines cibles). Ici on va utiliser juste une chaine avec le nom de la machine cible (node1 dans cet exemple). Pour cela il faut avoir au préalable une résolution de nom avec @IP de cette machine.
L'argument suivant permet de préciser le groupe cible. Ici `all` c'est un mot clé qui désigne la racine (càd toutes les machines qui sont renseignées dans l'inventaire).
L'option `-m` permet de préciser le module à appliquer, ici il s'agit du module ping.
Il existe une multitude de modules Ansible. La liste complète est disponible ici : https://docs.ansible.com/ansible/latest/collections/index_module.html
Pour chaque module vous avez une description des arguments et options qu'il supporte avec des exemples à l'appui. L'option -k (au minuscule) permet de demander le mdp administrateur pour se connecter à la machine cible.

Pour plus d'information sur la commande `ansible` consulter le help : `ansible --help`

1. Tester la ligne de code précédente (`ansible -i "node2," all -m ping`), vous obtenez quoi résultat ?

Dans la suite de l'exercice vous allez manipuler les modules : command, shell, copy, fetch et debug.

2. Préparer une 2e VM (Debian 11 ou plus) qui jouera le rôle de machine cible.
3. Consulter la documentation pour le module command permettant de lancer des commandes sur la machine cible puis tester le avec les commandes : **uptime ; ls ; pwd**.
4. Utiliser l'option **-e** de la commande ansible pour déclarer une variable d'environnement sur la machine cible. La syntaxe est -e "`var1=Bonjour`". Puis utiliser le module **debug** afin de visualiser le contenu de cette variable depuis la machine cible.
5. Grâce au module **copy** copier un fichier présent sur votre serveur vers la machine cible.
6. Grâce au module **fetch** récupérer un fichier présent sur votre machine cible pour le mettre sur le répertoire /tmp de votre serveur.
7. Avec le module **shell** lancer la commande qui permet de compter le nombre de processus qui tourne sur la machine cible.

Exercice 2 : Inventory

Un des éléments qu'il faut maitriser dans le monde Ansible c'est la définition des inventaires. Il s'agit de l'endroit où on doit définir les machines cibles, éventuellement les groupper avec des relations entre les groupes et voir définir des variables.
Cette structuration peut être formaté avec le format ini ou bien YAML.
Voici un exemple :

Soit la structuration suivante pour définir des groupes (parent x), des sous-groupes (enfant x) et les hosts (srv x) :

* un groupe parent1
* groupes enfants : enfant1 et enfant2
* "sous" enfant de enfant2 : enfant3
* enfant1 = srv1 et srv2
* enfant2 = srv3
* parent1 = srv4
* enfant3= srv5

Une syntaxe ini donnera :

```
[parent1]
srv4
[enfant1]
srv1
srv2
[enfant2]
srv3
[enfant3]
srv5
[parent1:children]
enfant1
```

```
enfant2
[enfant2:children]
enfant3
```

Par contre une syntaxe YAML donnera :

```
all:
  children:
    parent1:
      hosts:
        srv4:
      children:
        enfant1:
          hosts:
            srv1:
            srv2:
        enfant2:
          hosts:
            srv3:
        children
          enfant3:
            hosts:
              srv5:
```

Les mots clés `all, children et hosts` sont automatiquement reconnu par Ansible.

Il est possible d'utiliser des patterns afin de simplifier l'écriture :

```
```
all:
  children:
    parent1:
      parent2:
      hosts:
        srv4:
      children:
        enfant1:
          hosts:
            srv[1:2]:
        enfant2:
          hosts:
            srv3:
          children
            enfant3:
              hosts:
                srv5:
    parent2:
      hosts:
        srv[6:10]:
```

Dans un inventaire il est possible de déclarer des variables. À savoir qu'il y a deux types de variables : celles relatives aux groupes et celles relatives aux hosts. Pour les définir, soit on va les préciser directement dans le fichier inventaire ou bien on va les préciser dans des fichiers spécifiques dans les dossiers group_vars et host_vars. Ces dossiers doivent se situer au même niveau de votre fichier inventaire et doivent respecter ces noms (attendu par Ansible).
```

Exemple de déclaration directement dans le fichier inventaire :

```
```
all:
  children:
    parent1:
      parent2:
      hosts:
        srv4:
          var1: "variable pour le srv4"
      children:
        enfant1:
          hosts:
            srv[1:2]:
          vars:
            var1: "variable pour le groupe enfant1"
        enfant2:
          hosts:
            srv3:
              var1: "variable pour le srv3"
          children
            enfant3:
              hosts:
                srv5:
```

Exemple de structuration avec les dossiers `group_vars` et `host_vars` :

```
```
├── group_vars
│   ├── parent1.yml
│   ├── parent2.yml
│   ├── enfant1.yml
│   ├── enfant2.yml
│   └── enfant3.yml
├── host_vars
│   ├── srv4.yml
│   └── srv3.yml
└── inventory.yml
```

À noter que les noms des fichiers pour déclarer les variables doivent porter le même nom que celui des groupes ou bien des hosts déclarés dans l'inventaire.

1. Mettez à jour votre fichier /etc/hosts pour avoir la résolution de nom des nœuds node1 jusqu'à node10 avec @IP de la machine cible.
2. Créer un inventaire avec la syntax YAML qui permet de définir :
    a. Un groupe 'webserver' pour les serveurs web nginx. Ce groupe contient les nœuds 2 et 3.
    b. Un groupe 'dbserver' pour bases de données. Ce groupe contient les nœuds 4 et 5.
    c. Un groupe 'appdocker' pour les applications docker. Ce groupe contient les nœuds 6 et 7.
    d. Un groupe 'common' qui regroupe les trois groupes précédents.
    e. Un groupe 'nocommon' qui représentent les nœuds réseaux de 8 à 10.
3. Afin de tester votre inventaire, lancer la commande suivante :

```
ansible -i inventory.yml all -e "var1=Bonjour" -m debug -a
'msg={{ var1 }}'
```

Vous devez avoir le même message qui s'affiche pour l'ensemble des nœuds.

4. Maintenant vous allez rajouter des variables au niveau de votre inventaire.
    a. Déclarer une variable pour le groupe webserver qui a comme valeur « Ici c'est le groupe des serveurs web »
    b. Déclarer une variable pour le groupe dbserver qui a comme valeur « Ici c'est le groupe des serveurs de base de données »
    c. Déclarer une variable pour le node 5 qui a comme valeur « Ici c'est le neoud 5 »
    d. Déclarer une variable pour le groupe common qui a comme valeur « Ici c'est le groupe common »
    e. Déclarer une variable pour le groupe nocommon qui a comme valeur « Ici c'est le groupe nocommon »
5. Pour tester votre travail lancer la commande précédente sans l'option -e. Vérifier la concordance des valeurs affichées avec les emplacements des nœuds. Quelle valeur est affichée pour les nœuds 6 et 7 ?
6. Pour garder un fichier inventaire le plus simple possible, vous allez déclarer les variables dans des fichiers spécifiques à placer dans les dossiers `group_vars` et `host_vars`. Vous suffixez à chaque fois le contenu des variables par 'version 2' afin de constater la différence. Une fois c'est fait relancer la commande ansible, qu'est-ce que vous remarquez ?
7. Vous l'avez bien compris ! les variables déclarées dans les `group_vars` et `host_vars` surchargent celles déclarées dans l'inventaire. Nettoyez votre inventaire en retirant les lignes de déclarations de variables.
8. Vous allez modifier votre inventaire afin de rajouter le groupe 'monitoring' qui contient le nœud 7 et les groupes webserver et dbserver.
9. Utiliser la commande suivante afin de visualiser sur le terminal la structuration de votre inventaire : `ansible-inventory -i inventory.yml --list –yaml` Vérifier bien le contenu affiché des variables et si ça correspond à votre déclaration.
10. Tester également la commande : `ansible-inventory -i inventory.yml –graph –vars`
11. Et pour finir vous allez utiliser un plugin qui permet de produire un schéma à partir de votre inventaire. Pour cela installer les packages suivants :

```
pip3 install ansible-inventory-grapher
sudo apt install graphviz graphicsmagick-imagemagick-compat
```

Ensuite lancer la commande suivante :

```
ansible-inventory-grapher -i inventory.yml all | dot -Tpng |
display png:-
```

Vous allez déposer l'image sur moodle.

Exercice 3 : Playbook et modules

Dans cet exercice vous allez écrire votre premier playbook. Cet élément clé d'Ansible, permet de faire le lien entre l'inventaire (machines cibles) et les actions à effectuer (qui sont généralement déclarées dans les rôles). Dans un souci de

simplification, un playbook peut également contenir des actions à réaliser sur les machines cibles, ainsi que la déclaration des variables. Voici un exemple :

```
```
- name: Mon Playbook
  hosts: all #on peut mettre un nom de groupe ou de host
  become_user: yes
  tasks:
  - name: je debug
    debug:
      msg: "La valeur de la variable est {{ var1 }}"
```

Dans cet exemple :

- `all` signifie que l'ensemble des machines et groupes déclarés dans l'inventaire sont concernés par les actions qui vont suivre.
- `become_user` : les actions vont être exécutés en tant que root
- `tasks` : mot clé pour désigner les tâches à accomplir à travers l'appel à des modules
- `debug` : il s'agit du module `debug` vu précédemment

Pour lancer une playbook il faut utiliser la commande :

```
sudo ansible-playbook -i my-inventory.yml my-playbook.yml
```

Afin que votre machine Ansible arrive à interroger vos machines cibles, il faut préciser les @IP, les logins et mdp qu'elle doit utiliser. Pour cela il faut ajouter ces variables dans votre inventaire ou bien dans les fichiers propres à chaque machine.

```
Ansible_user : **********
ansible_password : ***********
ansible_host : A.B.C.D
```

Dans cet exercice vous allez utiliser l'inventaire que vous avez écrit dans l'exercice 2. Comme vous avez qu'une seul machine cible, vous pouvez déclarer les variables `ansible_user, ansible_password et ansible_host` dans le fichier all.yml (dans le dossier group_vars).

1. Faite tourner votre playbook avec l'exemple donné précédemment en utilisant l'inventaire de l'exercice 2. Vous obtenez quoi comme résultat ?
2. À présent vous allez explorer le module file, pour cela écrivez un 2e playbook qui permet (vous tester votre playbook pour chaque question) :

    a. Créer un répertoire. Quel est le propriétaire du dossier ?

    b. Créer un répertoire avec comme propriétaire l'utilisateur root
    c. Modifier le groupe du répertoire créé pour que ça soit root et les droits pour que ça soit 755.
    d. Créer les répertoires 1/2/3/4 dans le dossier /tmp dans une seule tâche
    e. Créer un fichier dans le répertoire 4
```

f. Créer un lien symbolique vers le répertoire 4 que vous nommée symlink
g. Supprimer le fichier créé précédemment.

3. Vous pouvez rejouer votre playbook étape par étape en rajoutant l'option –step dans votre commande.

Exercice 4 : les boucles with_items

Dans cet exercice vous allez voir comment vous pouvez faire des boucles avec Ansible. Voici un exemple où plusieurs sous répertoires sont créés dans un répertoire donné :

```
--
- name: Mon premier Playbook
  hosts: all
  become: yes
  tasks:
  - name: création des sous-répertoires avec with
    file:
      path: /tmp/TD_R511/{{item}}
      state: directory
    with_items:
      - t1
      - t2
      - t3
```

La tâche suivante permet de faire des listes composées :

```
- name: création de fichiers et dossiers (liste composée)
    file:
      path: /tmp/TD_R511/{{ item.dir }}/{{ item.file }}
      state: touch
        #recurse: yes
    with_items:
    - { dir: "t1", file: "fichierA"}
    - { dir: "t2", file: "fichierB"}
```

Ces exemples s'appuient sur une boucle avec with_item, mais il faut savoir que d'autres itérateurs existent avec Ansible. Vous pouvez consulter la documentation suivante pour plus d'information :
https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_loops.html

1. Créer un playbook qui permet de copier une liste de fichier vers votre serveur cible.