TP 2 : Ansible

Objectif : Le but de ce TP est de maitriser les notions de base d'Ansible

I. Premier pas avec Ansible

Ansible est un outil de gestion de configuration et d'automatisation pour les systèmes informatiques. Il utilise un modèle de description de la configuration, appelé "**playbook**", pour décrire les tâches à accomplir sur les différents ordinateurs et les différentes étapes à suivre pour atteindre un état cible pour le système. Les playbooks peuvent être exécutés localement ou à distance sur des hôtes distants via SSH.

EXERCICE 1

Dans cet exercice l'objectif est de déployer un serveur web nginx sur un serveur distant. Vous allez utiliser une 2e VM linux qui va jouer le rôle de serveur web. Sur cette VM créer le fichier /`etc/sudoers.d/username` et ajouter dedans la ligne suivante :

```
username ALL=(ALL) NOPASSWD: ALL
```

avec `username` est un nom d'utilisateur valide sur la VM linux client.

Étapes à suivre :

1- Créer un fichier d'inventaire avec les informations du serveur distant. Par exemple :

```
[servers]
VM1 ansible_host=@IP ansible_user=USER ansible_password=PASSWORD
```

Vous pouvez bien sûr utiliser la notation YAML.

2- Tester la possibilité de lancer des commandes sur votre 2e VM avec le module ping par exemple : ansible -i "VM2," all -m ping
Si vous avez des erreurs d'accès en SSH avec ansible adopter plutôt la solution suivante : vous allez configurer l'authentification par clé SSH plutôt que d'utiliser des mots de passe. Une fois les clés SSH configurées, vous n'aurez plus besoin de gérer les invites de mot de passe. Voici comment procéder :
   - Générer une clé SSH : `ssh-keygen -t rsa -b 4096`
   - Copiez la clé SSH sur l'hôte distant : `ssh-copy-id user@node1`
   - Exécutez maintenant la commande Ansible sans l'indicateur -k (car aucun mot de passe ne sera nécessaire) :

3- Créer un playbook nommé deploy_nginx.yml pour déployer le serveur web nginx. Le playbook doit contenir les tâches suivantes :
   - Installer nginx
   - Démarrer le service nginx
   - Ajouter un fichier index.html avec un message d'accueil personnalisé. Pour cela vous utiliser le module 'copy' avec les paramètres 'dest' et 'content'

4- Exécuter le playbook avec la commande suivante :

```
ansible-playbook -i inventory deploy_nginx.yml
```

Avec '`inventory`' votre fichier qui contient la liste des machines cibles.
Cela va déployer nginx sur les serveurs distants en utilisant les tâches définies dans le playbook.

5- Tester que le service nginx tourne bien sur votre 2e VM et que l'accès à la page principale (http :127.0.0.1) donne bien votre message d'accueil personnalisé.
6- Arrêter le service nginx avant de passer à l'exercice suivant

EXERCICE 2

Cet exercice a pour objectif de vous permettre de pratiquer la création d'inventaires, de playbooks et de rôles, ainsi que la configuration de services tels qu'Apache et MariaDB (MySQL). Vous allez également tester votre configuration et affiner vos compétences en matière de débogage.

Pour cet exercice vous aurai besoin de consulter la doc suivante concernant les modules **service**, **mysql_db** et **mysql_user** :

- https://docs.ansible.com/ansible/2.9/modules/service_module.html
- https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_db_module.html
- https://docs.ansible.com/ansible/latest/collections/community/mysql/mysql_user_module.html#ansible-collections-community-mysql-mysql-user-module

1. Créez un inventaire avec deux groupes, un pour les serveurs web et un pour les serveurs de bases de données. Chaque groupe doit avoir au moins une machine.

- Pour simplifier le travail, vous allez utiliser une seule VM en tant que serveur web et DB.

2. Créez un rôle pour l'installation et la configuration d'Apache. Ce rôle doit :

   - Installer Apache
   - Créer un fichier de configuration personnalisé pour Apache (vous êtes libre sur cette partie)
   - Redémarrer le service Apache après la modification de la configuration

3. Créez un rôle pour la configuration de MariaDB. Ce rôle doit :
   - Installer MariaDB et python3-pymysql (package permettant à ansible de manipuler mysql)
   - Créer une base de données nommée mydb_2024
   - Créer un utilisateur avec des privilèges sur cette base de données
   - Redémarrer le service MariaDB après la modification de la configuration

- N'oublier pas de rajouter la ligne : **login_unix_socket: /var/run/mysqld/mysqld.sock** permettant à Ansible d'utiliser l'authentification existante pour root. Cette option spécifie le chemin du socket Unix à utiliser pour se connecter à la base de données. Par défaut, MariaDB utilise le socket /var/run/mysqld/mysqld.sock.

4. Créez un playbook qui va mettre à jour les packages avec la commande "**apt update**" puis installer Apache et MariaDB sur les serveurs web et de base de données respectivement et cela grâce aux rôles créés précédemment.

5. Utilisez votre playbook pour configurer les serveurs web et de base de données.

6. Testez votre configuration :

   - En accédant à la page d'accueil d'Apache sur la machine du groupe des serveurs web.
   - En se connectant à la BD avec l'utilisateur que vous avez créé et vérifier que votre base de données est bien présente.

EXERCICE 3

L'objectif de cet exercice est d'utiliser Ansible pour déployer un conteneur Docker avec une application Web.

Prérequis : une machine virtuelle où un serveur avec Ansible et Docker installés.

Étapes :

1. Créer un fichier Dockerfile qui met à jour les packages, installe apache2, copie un fichier HTML (index.html avec un contenu de votre choix) dans le conteneur et démarre le serveur Apache. Vous faite en sorte de démarre Apache en mode **foreground** pour maintenir le conteneur actif tant qu'Apache est en cours d'exécution.
2. Créer un playbook Ansible qui :
    a. Installe les packages **pip** et **docker** sur la machine cible,
    b. Démarre le service docker
    c. Copie les fichiers Dockerfile et index.html sur la machine cible.
    d. Construit l'image docker à partir du dockerfile avec le nom myapache_NM (vos initiales)
    e. Démarre un conteneur avec le nom myapache_container_NM et l'expose sur le port 8080.
3. Exécuter le playbook Ansible pour déployer le conteneur Docker et l'application Web.
4. Vérifier que l'application Web est accessible en accédant à l'adresse IP au port sur lequel l'application est exécutée.
5. Modifier votre playbook pour arrêter puis supprimer tous les conteneurs présents sur la machine cible et également supprimer toutes les images.
6. Si vous relancer votre playbook et dans le cas où aucun conteneur n'est active et aucune image n'est présente, vous allez avoir une erreur. Pour éviter cela, utiliser la clause **when, register et changed_when**

EXERCICE 4 : UTILISATION D'ANSIBLE POUR LA SAUVEGARDE DE CONFIGURATION

Dans cette partie, vous allez créer un playbook Ansible qui automatisera le processus de sauvegarde de la configuration des routeurs qui sont sur EVE-NG. Les playbooks sont au centre d'Ansible. Lorsque vous souhaitez qu'Ansible obtienne des informations ou effectuer une action sur un appareil ou un groupe d'appareils, vous exécutez un playbook pour faire le travail.

Ansible est livré avec un certain nombre de modules (appelés bibliothèque de modules) qui peuvent être exécutés directement sur les hôtes distants ou via des playbooks. Un exemple est le module **ios_command** utilisé pour envoyer des commandes à un périphérique IOS et renvoyer les résultats. Chaque tâche consiste généralement en un ou plusieurs modules Ansible.

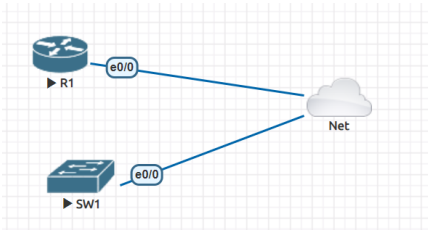
Pour la suite du TP vous mettez l'ensemble de vos fichiers de configuration dans un nouveau répertoire (**TP2-exe4)**.

Avant de continuer, il faut installer **python3-paramiko** et **ansible-pylibssh** sur la VM qui jouera le rôle de serveur Ansible. Aussi, pour accéder en ssh depuis votre VM Ansible aux clients il faut modifier le fichier **/etc/ssh/ssh_config** en rajoutant la ligne suivante :

kexalgorithms diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1

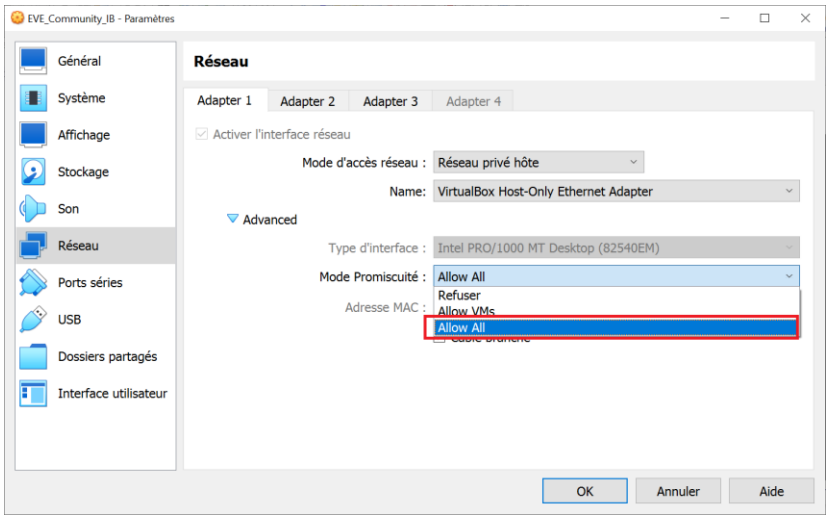
PARTIE 1

1) Préparation des routeurs sur eve-ng :



Commencer tout d'abord par préparer la maquette suivante sur eve-ng. Grâce au Cloud0, vous allez connecter votre VM qui joue le rôle de serveur Ansible à eve-ng. Vérifier bien que vous arrivez à faire un ping entre un équipement sur votre maquette et votre VM Ansible.

A noter que pour le mode promiscuité fonctionne, il faut autoriser tout au niveau de la VM eve-ng :



Sur le routeur et le switch vous allez faire le nécessaire pour configurer un accès ssh. Pour tester l'accès depuis votre VM ansible, vous allez utiliser la syntaxe suivante : **ssh user@IP -oHostKeyAlgorithms=+ssh-rsa**

2) Vous devez créer votre fichier inventaire pour lister les équipements cibles. Pour des raisons de clarté, vous allez utiliser une syntaxe en YAML étant donné que plusieurs paramètres seront nécessaires pour configurer l'accès. Voici le modèle à suivre :

```
---
router:
  hosts:
    R1:
      ansible_host: @IP
      ansible_network_os: cisco.ios.ios
      ansible_user: your_user
      ansible_password: your-pw
      ansible_become: yes
      ansible_become_method: enable
      ansible_become_password: enable-pw
```

3) Créez votre playbook Ansible :

   a. Créez un nouveau fichier dans votre répertoire avec le nom suivant : backup_router.yaml
   b. Ajoutez les informations suivantes au fichier :

```
---
- name: AUTOMATIC BACKUP OF RUNNING-CONFIG
  hosts: router
  gather_facts: false
  connection: ansible.netcommon.network_cli
  tasks:
    - name: DISPLAYING THE RUNNING-CONFIG
      ios_command:
        commands:
            - show running-config
      register: config
    - name: SAVE OUTPUT TO ./backups/
      copy:
        content: "{{ config.stdout[0] }}"
        dest: "backups/show_run_{{ inventory_hostname }}.txt"
```

Le playbook que vous avez créé contient un jeu de deux tâches. Voici une explication du contenu de ce playbook :

- --- C'est au début de chaque fichier YAML, ce qui indique à YAML qu'il s'agit d'un document séparé. Chaque fichier peut contenir plusieurs documents séparés par ---
- name : AUTOMATIC BACKUP OF RUNNING-CONFIG - C'est le nom du jeu d'instruction.
- **hosts** : router - Il s'agit de l'alias du fichier d'hôtes précédemment configuré. En se référant à cet alias dans votre playbook, le playbook peut utiliser tous les paramètres associés à cette entrée de fichier d'inventaire qui comprend le nom d'utilisateur, le mot de passe et l'adresse IP de l'appareil.
- **gather_facts** : false - Ansible a été conçu à l'origine pour fonctionner avec des serveurs Linux, en copiant des modules Python aux serveurs pour l'automatisation des tâches. Ceci n'est pas nécessaire lorsque vous travaillez avec des périphériques réseau.
- **connection**: ansible.netcommon.network_cli - Spécifie que vous utilisez SSH pour se connecter à l'équipement distant.
- **Tasks** : Ce mot-clé indique une ou plusieurs tâches à effectuer.

La première tâche consiste à afficher le fichier running-config.

- name : AFFICHAGE DU RUNNING-CONFIG - Nom de la tâche.
- **ios_command** : Il s'agit d'un module Ansible utilisé pour envoyer des commandes à un périphérique IOS et renvoie les résultats lus depuis l'appareil. Cependant, il ne prend pas en charge les commandes de configuration. Le module ios_config est utilisé à cette fin, comme vous le verrez dans la partie suivante de ce TP.

Remarque : Dans le terminal Linux, vous pouvez utiliser la commande **`ansible-doc module_name`** pour afficher les pages de manuel pour n'importe quel module et les paramètres associés à ce module. (par exemple `ansible-doc ios_command`)

- **commandes** : Ce paramètre est associé au module ios_command. Il est utilisé pour lister les commandes IOS au niveau du playbook qui doivent être envoyées au périphérique IOS distant. La sortie résultante de la commande est renvoyée.
- **show running-config** : vous avez l'habitude !
- **registre : config** - Ansible inclut des registres utilisés pour capturer la sortie d'une tâche vers une variable. Cette ligne spécifie que la sortie de la commande show running-config précédente sera stockée dans la variable config.

La deuxième tâche consiste à enregistrer la sortie :

- - nom : SAVE OUTPUT TO ./backups/ - Nom de la tâche
- **copy** : Il s'agit d'un module Ansible utilisé pour copier des fichiers vers un emplacement distant. Il y a deux paramètres associés à ce module :

- **content : "{{ config.stdout[0] }}"** - La valeur spécifiée pour ce paramètre est la donnée stockée dans la variable config, utilisée dans la tâche précédente. Standard output (stdout) est le descripteur de fichier par défaut où un processus peut écrire une sortie comme c'est utilisé dans les systèmes Unix.
- **dest : "backups/show_run_{{ inventory_hostname }}.txt"** - Il s'agit du chemin et du nom fichier où le résultat doit être copié. La variable **inventory_hostname** est une variable "magic' d'Ansible qui reçoit automatiquement le nom d'hôte tel qu'il est configuré dans le fichier hosts (fichier inventaire). Donc si le routeur est intitulé 'R1' dans l'inventaire, le paramètre génère un fichier 'show_run_R1.txt' stocké dans le répertoire 'backups'. Le fichier contiendra la sortie de la commande show running-config.

4) Exécutez votre Playbook de sauvegarde Ansible.
   a- Créez le répertoire des sauvegardes 'backups'. Comme indiqué dans la dernière ligne de votre playbook, il s'agit du répertoire où sera stocké le fichier de configuration de sauvegarde
   b- Lancer votre playbook.

Vous devez avoir un affichage qui ressemble à :

```
PLAY [AUTOMATIC BACKUP OF RUNNING-CONFIG]
************************************************************

TASK [DISPLAYING THE RUNNING-CONFIG]
************************************************************************
ok: [R1]

TASK [SAVE OUTPUT TO ./backups/]
************************************************************************
changed: [R1]

PLAY RECAP
*************************************************************************************
***********
R1                   : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

Vous pouvez ignorer les messages 'WARNING'
   c- Vérifiez que le fichier de sauvegarde a été créé.

PARTIE 2

5) Utiliser Ansible pour configurer un appareil :
   Dans cette partie, vous allez créer un autre playbook Ansible pour configurer l'adressage IPv4 sur le routeur R1.
   a- Créez une copie de votre playbook vers un nouveau fichier avec le nom suivant : router_ipv4_config.yaml
   b- Dans votre nouveau playbook vous allez ajouter la tâche suivante permettant de configurer l'interface ethernet 0/2 sur le routeur :

```yaml
tasks:
  - name: Config_ipv4_interface
    ios_config:
      parents: "interface Ethernet0/2"
      lines:
        - description IPv4 ADDRESS
        - interface e0/2
        - ip address 10.10.10.1 255.255.255.0
        - no shut
```

Assurez-vous de bien respecter l'indentation YAML.

- **`ios_config`** : est un module Ansible utilisé pour configurer un périphérique IOS. Vous pouvez utiliser la commande **`ansible-doc ios_config`** pour voir les détails des paramètres parents et lignes utilisés dans ce playbook.
- **parents** : "interface Ethernet0/0" - Ce paramètre indique le mode de configuration de l'interface IOS.
- **lignes** : un ensemble ordonné de commandes IOS qui est configuré dans cette section

c- Vous remplacez la commande d'affichage 'show running-config' par la commande permettant d'avoir un affichage résumé de l'état des interfaces.
d- Vous modifier l'emplacement de sauvegarde du résultat de la commande pour que ça soit dans le dossier **ios_show** (n'oublier pas de créer ce dossier s'il n'existe pas avec une tâche) avec le nom du fichier show_ip_intr_br_{nom du routeur}.txt
e- Lancer votre playbook et vérifier que la configuration a bien été effectué sur le routeur
f- Vérifier le résultat de la commande `show ip inter br` à partir du fichier présent dans 'ios_show'

6) Créer un autre playbook afin d'administrer le switch en :
   a. Changeant le nom vers SW_NM (vos initiales)
   b. Créer le VLAN 200
   c. Mettre l'interface e0/1 et e0/2 dans ce VLAN
7) Tester votre playbook sur le switch et vérifier que la configuration a bien eu lieu.

- Pour tester l'accès ssh vers votre switch il faut utiliser **ssh user@IP -oHostKeyAlgorithms=+ssh-rsa -c aes128-cbc**