Exercice 11 - Utiliser Ansible pour sauvegarder et configurer un périphérique

Objectifs

Partie 1 : Lancer la machine virtuelle DEVASC et la machine virtuelle CSR1000v

Partie 2 : Configurer Ansible

Partie 3 : Utiliser Ansible pour sauvegarder une configuration

Partie 4 : Utiliser Ansible pour configurer un périphérique

Contexte/scénario

Dans ce TP, vous explorerez les principes fondamentaux de l'utilisation d'Ansible pour automatiser certaines tâches de gestion de périphériques de base. Tout d'abord, vous allez configurer Ansible dans votre VM DEVASC. Ensuite, vous allez utiliser Ansible pour vous connecter au CSR1000v et sauvegarder sa configuration. Enfin, vous allez configurer le CSR1000v avec l'adressage IPv6.

Ressources requises

- 1 PC avec système d'exploitation de votre choix
- VirtualBox ou VMWare
- Machine virtuelle DEVASC
- Machine virtuelle CSR1000v

Instructions

Partie 1 : Lancez les machines virtuelles DEVASC et CSR1000V

Si vous n'avez pas encore terminé le **TP - Installez la machine virtuelle DEVASC-LAB et CSR1000v**, faites-le maintenant. Si vous avez déjà terminé ce laboratoire, lancez les machines virtuelles DEVASC et CSR1000v maintenant.

Partie 2 : Configurer Ansible

Dans cette partie, vous allez configurer Ansible pour qu'il s'exécute à partir d'un répertoire spécifique.

Étape 1: Ouvrez le répertoire Ansible dans VS Code.

a. Ouvrez **VS Code** (il est déjà installé dans la VM DEVASC).

b. Cliquez sur **File > Open Folder...** et accédez au dossier **/labs/devnet-src/ansible/ansible-csr1000v**.

c. Cliquez sur **OK**.

d. Le sous-répertoire du laboratoire est maintenant chargé dans le volet VS Code **EXPLORER** pour votre commodité.

Étape 2: Modifiez le fichier de stock Ansible.

Ansible utilise un fichier d'inventaire appelé **hosts** qui contient des informations sur l'appareil utilisées par les playbooks Ansible. L'emplacement par défaut du fichier d'inventaire Ansible est **/etc/ansible/hosts** comme spécifié dans le fichier **ansible.cfg** par défaut dans le même répertoire **/etc/ansible**. Ces fichiers par défaut

sont utilisés lorsque Ansible est exécuté globalement. Cependant, dans ce laboratoire, vous allez exécuter Ansible à partir du répertoire **ansible-csr1000v**. Par conséquent, vous avez besoin d' **hôtes** séparés et de fichiers **ansible.cfg** pour chaque laboratoire.

Remarque : Les termes **fichier hosts** et **fichier d'inventaire** sont synonymes et seront utilisés de façon interchangeable dans les travaux pratiques d'Ansible.

Le fichier d'inventaire Ansible définit les périphériques et groupes d'appareils utilisés par le playbook Ansible. Le fichier peut être dans l'un des nombreux formats, y compris YAML et INI, en fonction de votre environnement Ansible. Le fichier d'inventaire peut répertorier les périphériques par adresse IP ou par nom de domaine complet (FQDN), et peut également inclure des paramètres spécifiques à l'hôte.

a. Ouvrez le fichier **hosts** dans le répertoire **ansible-csr1000v**.

b. Ajoutez les lignes suivantes au fichier **hosts** et enregistrez.

```
# Enter the hosts or devices for Ansible playbooks
CSR1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.101
```

Après le commentaire (#), le fichier **hosts** commence par un alias, **CSR1kv**. Un alias est utilisé depuis le playbook Ansible pour référencer un appareil. Après l'alias, le fichier **hosts** spécifie trois variables qui seront utilisées par le playbook Ansible pour accéder au périphérique. Il s'agit des informations d'identification SSH dont Ansible a besoin pour accéder en toute sécurité à la machine virtuelle CSR1000v.

- **ansible_user** est une variable contenant le nom d'utilisateur utilisé pour se connecter au périphérique distant. Sans cela, l'utilisateur qui exécute l'ansible-playbook serait utilisé.
- **ansible_password** est une variable contenant le mot de passe correspondant à **ansible_user**. Si elle n'est pas spécifiée, la clé SSH sera utilisée.
- **ansible_host** est une variable contenant l'adresse IP ou le nom de domaine complet du périphérique. N'oubliez pas d'ajuster l'adresse IP à votre CSR1KV.

Étape 3: Affiche la version Ansible et l'emplacement ansible.cfg par défaut.

a. Pour voir où Ansible stocke le fichier ansible.cfg par défaut, ouvrez une fenêtre de terminal et naviguez jusqu'au répertoire parent ansible.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cd ..
devasc@labvm:~/labs/devnet-src/ansible$
```

b. Tapez **ansible** pour obtenir la liste des commandes ansible. Notez l'option **--version**.

```
devasc@labvm:~/labs/devnet-src/ansible$ ansible
usage: ansible [-h] [--version] [-v] [-b] [--become-method BECOME_METHOD]
               [--become-user BECOME_USER] [-K] [-i INVENTORY] [--list-hosts]
               [-l SUBSET] [-P POLL_INTERVAL] [-B SECONDS] [-o] [-t TREE] [-k]
               [--private-key PRIVATE_KEY_FILE] [-u REMOTE_USER]
               [-c CONNECTION] [-T TIMEOUT]
               [--ssh-common-args SSH_COMMON_ARGS]
<output omitted>
devasc@labvm:~/labs/devnet-src/ansible$
```

c. Utilisez la commande **ansible --version** pour afficher les informations sur la version. Notez que ce TP utilise la version 2.9.6. Ansible inclut certains fichiers par défaut, y compris un fichier de configuration par défaut, **ansible.cfg**.

```
devasc@labvm:~/labs/devnet-src/ansible$ ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
```

```
  configured module search path = ['/home/devasc/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.8.2 (default, Apr 27 2020, 15:53:34) [GCC 9.3.0]
devasc@labvm:~/labs/devnet-src/ansible$
```

Étape 4: Affiche le fichier ansible.cfg par défaut.

Le fichier **ansible.cfg** est utilisé par Ansible pour définir certaines valeurs par défaut. Ces valeurs peuvent être modifiées.

À l'aide du chemin par défaut affiché dans la commande **ansible --version**, affichez le fichier de configuration par défaut. Notez que c'est un dossier très long. Vous pouvez diriger la sortie de la commande **cat** vers **more** de sorte qu'elle affiche une page à la fois ou utiliser **more** directement au lieu de **cat**. En surbrillance sont les entrées qui seront dans votre fichier **ansible.cfg** pour ce laboratoire.

```
devasc@labvm:~/labs/devnet-src/ansible$ cat /etc/ansible/ansible.cfg | more
# config file for ansible -- https://ansible.com/
# ===============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

#inventory = /etc/ansible/hosts
<output omitted>

# uncomment this to disable SSH key host checking
#host_key_checking = False
<output omitted>

# retry files
# When a playbook fails a .retry file can be created that will be placed in ~/
# You can enable this feature by setting retry_files_enabled to True
# and you can change the location of the files by setting retry_files_save_path

#retry_files_enabled = False
<output omitted>
```

Notez qu'Ansible indique que le fichier des **hôtes** d'inventaire qu'il utilisera par défaut est **/etc/ansible/hosts**. Lors d'une étape précédente, vous avez modifié le fichier des **hôtes** d'inventaire dans le **répertoire ansible-csr1000v**. Dans l'étape suivante, vous allez modifier un nouveau **fichier ansible.cfg** qui utilise le fichier d'inventaire des **hôtes** que vous avez créé.

Étape 5: Modifiez l'emplacement du fichier ansible.cfg.

a. Ansible utilisera le fichier de configuration situé dans **/etc/ansible/ansible.cfg**, sauf s'il y a un fichier **ansible.cfg** dans le répertoire courant. Revenez au répertoire ansible-csr1000v. Il y a déjà un fichier réservé **ansible.cfg** dans ce répertoire. Afficher l'emplacement actuel de **ansible.cfg** à l'aide de la commande **ansible --version**.

```
devasc@labvm:~/labs/devnet-src/ansible$ cd ansible-csr1000v/
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible --version
ansible 2.9.6
  config file = /home/devasc/labs/devnet-src/ansible/ansible-csr1000v/ansible.cfg
<output omitted>
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

b. Affichez le fichier pour voir qu'il est vide, sauf pour un commentaire. Vous allez modifier ce fichier à l'étape suivante.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cat ansible.cfg
# Add to this file for the Ansible lab
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

Étape 6: Modifiez le fichier ansible.cfg.

Maintenant, vous devez modifier votre fichier **/ansible-csr1000v/ansible.cfg** pour inclure l'emplacement de votre fichier d'inventaire **hosts**. Rappelez-vous que le fichier de configuration par défaut dans **/etc/ansible/ansible.cfg** utilise le fichier d'inventaire dans /etc/ansible/hosts.

a. Ouvrez le fichier **/ansible-csr1000v/ansible.cfg** dans VS Code.

b. Vous pouvez supprimer le commentaire. Ajoutez les lignes suivantes au fichier et enregistrez-le.

```
# config file for ansible-csr1000v
[defaults]
# Use local hosts file in this folder
inventory=./hosts
host_key_checking = False # Don't worry about RSA Fingerprints
retry_files_enabled = False # Do not create them
deprecation_warnings = False # Do not show warnings
```

Comme Python, le # est utilisé pour les commentaires dans le fichier **ansible.cfg**. Si l'entrée fait référence à un nom de fichier tel que **inventory=./hosts** le commentaire ne peut pas venir après l'entrée. Ansible traite le # et le commentaire qui suit comme faisant partie du nom de fichier. Par conséquent, dans ces cas, le commentaire # doit être sur une ligne distincte. Cependant, les variables peuvent avoir un commentaire sur la même ligne que celle affichée pour **host_key_check** et **retry_files_enabled**.

Le fichier **ansible.cfg** indique à Ansible où trouver le fichier d'inventaire et définit certains paramètres par défaut. Les informations que vous avez entrées dans votre fichier **ansible.cfg** sont les suivantes :

- **`inventaire=./hosts`** - Votre fichier d'inventaire est le fichier **hosts** dans le répertoire courant.
- **`host_key_check = False`** - L'environnement de développement local ne dispose pas de clés SSH configurées. Vous avez défini l'option **host_key_check** sur **False**, qui est la valeur par défaut. Dans un réseau de production, **host_key_check** serait défini sur **True**.
- **`retry_files_enabled = False`** - Lorsqu'Ansible rencontre des problèmes pour exécuter des playbooks pour un hôte, il affiche le nom de l'hôte dans un fichier du répertoire courant se terminant par**.retry**. Pour éviter l'encombrement, il est courant de désactiver ce paramètre.

- **`deprecation_warnings=false`** - Les avertissements d'obsolescence indiquent l'utilisation de fonctionnalités héritées qui doivent être supprimées dans une future version d'Ansible. Vous avez désactivé cet avertissement.

Étape 7: RÉSUMÉ : Vos fichiers de configuration Ansible.

Dans cette partie, vous avez configuré Ansible pour qu'il s'exécute dans le répertoire **ansible-csr1000v**. Par défaut, Ansible utilise des fichiers dans le répertoire **/etc/ansible**. Le fichier /etc/ansible/**ansible.cfg** par défaut indique que le fichier d'inventaire par défaut est **/etc/ansible/hosts**.

Cependant, dans ce laboratoire, vous avez besoin d'un fichier hosts et d'un fichier ansible.cfg dans votre répertoire ansible-csr1000v.

- Vous avez modifié le fichier **hosts** pour contenir les informations de connexion et d'adresse IP pour le routeur CSR1000v
- Vous avez modifié le fichier **ansible.cfg** pour utiliser le fichier hôte local comme fichier d'inventaire (**inventory=./hôtes**).

Dans la partie suivante, vous allez créer un playbook pour dire à Ansible ce qu'il faut faire.

Partie 3 : Utiliser Ansible pour sauvegarder une configuration

Dans cette partie, vous allez créer un playbook Ansible qui automatisera le processus de sauvegarde de la configuration du CSR1000v. Les playbooks sont au centre d'Ansible. Lorsque vous souhaitez qu'Ansible récupère des informations ou effectue une action sur un appareil ou un groupe d'appareils, vous exécutez un playbook pour effectuer le travail.

Un playbook Ansible est un fichier YAML contenant un ou plusieurs plays. Chaque play est un ensemble de tâches.

- Un **play** est un ensemble de tâches correspondant à un appareil ou un groupe d'appareils.
- Une **tâche** est une action unique qui fait référence à un **module** à exécuter avec tous les arguments et actions en entrée. Ces tâches peuvent être simples ou complexes selon le besoin d'autorisations, l'ordre d'exécution des tâches, etc.

Un playbook peut également contenir des **rôles**. Un rôle est un mécanisme permettant de diviser un playbook en plusieurs composants ou fichiers, de simplifier le playbook et de le rendre plus facile à réutiliser. Par exemple, le rôle **commun** est utilisé pour stocker les tâches qui peuvent être utilisées dans tous vos playbooks. Les rôles dépassent le cadre de ce laboratoire.

Le playbook Ansible YAML comprend **des objets**, des **listes** et des **modules**.

- Un objet YAML est une ou plusieurs paires de valeurs clés. Les paires de valeurs clés sont séparées par un deux-points sans l'utilisation de guillemets, par exemple **hosts: CSR1kv**.
- Un objet peut contenir d'autres objets tels qu'une liste. YAML utilise des listes ou des tableaux. Un tirait "-" est utilisé pour chaque élément de la liste.
- Ansible est livré avec un certain nombre de modules (appelés bibliothèque de modules) qui peuvent être exécutés directement sur des hôtes distants ou via des playbooks. Par exemple, le module **ios_command** utilisé pour envoyer des commandes à un périphérique IOS et renvoyer les résultats. Chaque tâche se compose généralement d'un ou de plusieurs modules Ansible.

Vous exécutez un playbook Ansible à l'aide de la commande **ansible-playbook**, par exemple :

```
ansible-playbook backup_cisco_router_playbook.yaml -i hosts
```

La commande **ansible-playbook** utilise des paramètres pour spécifier :

- Le playbook que vous voulez exécuter (**backup_cisco_router_playbook.yaml**)

- Le fichier d'inventaire et son emplacement (**-i hosts**). Ce paramètre est nécessaire si vous n'avez pas de fichier **ansible.cfg** qui change son emplacement par défaut.

Étape 1: Créez votre playbook Ansible.

Le playbook Ansible est un fichier YAML. Assurez-vous d'utiliser l'indentation YAML appropriée. Chaque espace et chaque tiret sont importants. Vous risquez de perdre une certaine mise en forme si vous copiez et collez le code dans ce laboratoire.

a. Dans VS Code, créez un nouveau fichier dans le répertoire **ansible-csr1000v** avec le nom suivant : **backup_cisco_router_playbook.yaml**

b. Ajoutez les informations suivantes au fichier.

```
---
- name: AUTOMATIC BACKUP OF RUNNING-CONFIG
  hosts: CSR1kv
  gather_facts: false
  connection: local

  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"
```

Étape 2: Examinez votre playbook Ansible.

Le playbook que vous avez créé contient un play avec deux tâches. Voici une explication de votre playbook :

- --- Ceci est au début de chaque fichier YAML, ce qui indique à YAML qu'il s'agit d'un document distinct. Chaque fichier peut contenir plusieurs documents séparés par ---
- **`name : BACKUP AUTOMATIQUE DE RUNNING-CONFIG`** - Ceci est le nom du **play**.
- **`hosts : CSR1kv`** - Ceci est l'alias du fichier **hosts** précédemment configuré. En faisant référence à cet alias dans votre playbook, le playbook peut utiliser tous les paramètres associés à cette entrée de fichier d'inventaire qui inclut 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 sur les serveurs pour l'automatisation des tâches. Ce n'est pas nécessaire lorsque vous travaillez avec des périphériques réseau.
- **`connection: local`** - Spécifie que vous n'utilisez pas SSH, par conséquent la connexion est locale.
- **`tasks:`** - Ce mot-clé indique une ou plusieurs tâches à effectuer.

La première tâche consiste à afficher le running-config.

- `- name: DISPLAYING THE RUNNING-CONFIG` - Name of the task.

- **`ios_command:`** - Ceci est un **module** Ansible qui est utilisé pour envoyer des commandes à un périphérique IOS et retourner les résultats lus depuis le périphérique. Cependant, il ne prend pas en charge les commandes de configuration. Le module **ios_config** est utilisé à cet effet, comme vous le verrez dans la prochaine partie de ce TP.

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

- **`command :`** - Ce paramètre est associé au module **ios_command**. Il est utilisé pour répertorier les commandes IOS dans le playbook qui doivent être envoyées au périphérique IOS distant. La sortie résultante de la commande est renvoyée.

- **`- show running-config`** - Ceci est la commande Cisco IOS envoyée en utilisant le module **ios_command**.

- **`register: config`** - Ansible inclut les registres utilisés pour capturer la sortie d'une tâche vers une variable. Cette entrée 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 :

- **`- name: SAVE OUTPUT TO ./backups/`** - Name of the task

- **`copy :`** - Ceci est un **module** Ansible utilisé pour copier des fichiers vers un emplacement distant. Deux paramètres sont associés à ce module :

  - **`content: "`**`{{ config.stdout[0] }}`**`"`** - La valeur spécifiée pour ce paramètre est les données stockées dans la variable de **configuration**, la variable de registre Ansible utilisée dans la tâche précédente. La sortie standard (**stdout**) est le descripteur de fichier par défaut où un processus peut écrire une sortie utilisée dans des systèmes d'exploitation de type Unix, tels que Linux et Mac OS X.

  - **`dest: "backups/show_run_{{ inventory_hostname }}.txt"`** - Ceci est le chemin d'accès et le nom du fichier vers lequel le fichier doit être copié. La variable **inventory_hostname** est une "variable magique" Ansible qui reçoit automatiquement le nom d'hôte tel que configuré dans le fichier **hosts** . Dans votre cas, rappelez-vous qu'il s'agit de **CSR1kV**. Ce paramètre génère un fichier **show_run_CSR1kv.txt** stocké dans le répertoire **backups**. Le fichier contiendra la sortie de la commande **show running-config** . Vous allez créer le répertoire **backups** à l'étape suivante.

Étape 3: Exécutez le Playbook de sauvegarde Ansible.

a. Dans la partie 1, vous avez démarré la machine virtuelle CSR1000v. Ping pour vérifier que vous pouvez y accéder.

```
devasc@labvm:~/labs/devnet-src/ansible$ ping 192.168.56.101 -c 3
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=63 time=0.913 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=63 time=0.875 ms
^C
--- 192.168.56.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0,875/0,894/0,913/0,019 ms
devasc@labvm:~/labs/devnet-src/ansible$
```

b. Créez le répertoire **backups**. Comme indiqué dans la dernière ligne de votre playbook, il s'agit du répertoire où le fichier de configuration de sauvegarde sera stocké.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ mkdir backups
```

c. Vous pouvez maintenant exécuter le playbook Ansible à l'aide de la commande **ansible-playbook** :

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook
backup_cisco_router_playbook.yaml

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

TASK [DISPLAYING THE RUNNING-CONFIG] ********************************************
ok: [CSR1kv]

TASK [SAVE OUTPUT TO backups/] **************************************************
changed: [CSR1kv]

PLAY RECAP **********************************************************************
CSR1kv : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

Remarque: Dans de nombreux exemples, vous verrez le livre de jeu fonctionner avec l'option **-i** *inventory-filename*. Par exemple :

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook
backup_cisco_router_playbook.yaml -i hosts
```

Cette option indique à Ansible l'emplacement et le nom du fichier d'inventaire, la liste des appareils que le playbook utilisera. Cette option n'est pas nécessaire car vous avez configuré le nom et l'emplacement du fichier d'inventaire dans votre fichier local **ansible.cfg** : inventory=./hosts. Vous pouvez utiliser l'option **-i** *inventory-filename* pour remplacer les informations contenues dans le fichier **ansible.cfg** .

Le **PLAY RECAP** doit afficher **ok=2 changed=1** indiquant une exécution réussie du playbook.

Si votre playbook Ansible échoue, voici quelques éléments à vérifier dans votre playbook :

- Assurez-vous que vos **hôtes** et les fichiers **ansible.cfg** sont corrects.
- Assurez-vous que l'indentation YAML est correcte.
- Assurez-vous que votre commande IOS est correcte.
- Vérifiez toute la syntaxe du playbook Ansible.
- Vérifiez que vous pouvez effectuer un ping sur le CSR1000v.

Si vous continuez à rencontrer des problèmes :

- Essayez de taper une ligne à la fois et d'exécuter le playbook à chaque fois.
- Comparez votre fichier avec le playbook des solutions dans le répertoire **ansible_solutions**.

Étape 4: Vérifiez que le fichier de sauvegarde a été créé.

Dans VS Code, ouvrez le dossier **backups** et ouvrez le fichier **show_run_CSR1kv.txt**. Vous pouvez également utiliser la fenêtre du terminal pour ouvrir le fichier avec **cat backups/show_run_CSR1kv.txt**. Vous disposez maintenant d'une sauvegarde de la configuration CSR1000v.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v/backups$ cat
show_run_CSR1kv.txt
Building configuration...

Current configuration : 4004 bytes
!
```

```
! Last configuration change at 23:57:14 UTC Sun May 17 2020
!
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
platform console virtual
!
hostname CSR1kv
!
<output omitted>
```

Partie 4 : Utiliser Ansible pour configurer un périphérique

Dans cette partie, vous allez créer un autre playbook Ansible pour configurer l'adressage IPv6 sur le routeur CSR1000v.

Étape 1: Affichez votre fichier d'inventaire des hôtes.

a. Réexaminez votre fichier d'inventaire des **hôtes**. Pour rappel, ce fichier contient l'alias **CSR1kv** et trois variables d'inventaire pour le nom d'utilisateur, le mot de passe et l'adresse IP de l'hôte. Le playbook de cette partie utilisera également ce fichier et le fichier **ansible.cfg** que vous avez créé au début du TP.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cat hosts
# Enter the hosts or devices for Ansible playbooks
CSR1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.101
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

Étape 2: Créez un nouveau playbook.

a. Dans VS Code, créez un nouveau fichier dans le répertoire **ansible-csr1000v** avec le nom suivant: **cisco_router_ipv6_config_playbook.yaml**

b. Ajoutez les informations suivantes au dossier. Assurez-vous d'utiliser l'indentation YAML appropriée. Chaque espace et chaque tiret sont importants. Vous risquez de perdre une certaine mise en forme si vous copiez et collez.

```
---
- name: CONFIGURE IPv6 ADDRESSING
  hosts: CSR1kv
  gather_facts: false
  connection: local

  tasks:
   - name: SET IPv6 ADDRESS
     ios_config:
       parents: "interface GigabitEthernet1"
       lines:
         - description IPv6 ADDRESS
         - ipv6 address 2001:db8:acad:1::1/64
         - ipv6 address fe80::1:1 link-local
```

```
- name: SHOW IPv6 INTERFACE BRIEF
  ios_command:
    commands:
      - show ipv6 interface brief
  register: output

- name: SAVE OUTPUT ios_configurations/
  copy:
    content: "{{ output.stdout[0] }}"
    dest: "ios_configurations/IPv6_output_{{ inventory_hostname }}.txt"
```

Étape 3: Examinez votre playbook Ansible.

Une grande partie de ce playbook est similaire au playbook que vous avez créé dans la partie précédente. La principale différence est la première tâche SET IPv6 ADRESSE.

Voici une brève description des éléments de la tâche :

- **`ios_config :`** - Ceci est un module Ansible utilisé pour configurer un périphérique IOS. Vous pouvez utiliser la commande **ansible-doc ios_config** pour afficher les détails des paramètres **parents** et **lignes** utilisés dans ce playbook.
- **`parents : "interface GigabiteThernet1"`** - Ce paramètre indique le mode de configuration de l'interface IOS.
- **`lines :`** - Un ensemble ordonné de commandes IOS est configuré dans cette section, spécifiant les informations d'adressage IPv6 pour l'interface GigabiteThernet1.

Le reste du playbook est similaire aux tâches de la partie précédente. La deuxième tâche utilise le module **ios_command** et la commande **show ipv6 interface brief** pour afficher la sortie et l'envoyer à la **sortie** enregistrée.

La dernière tâche enregistre les informations contenues dans la **sortie** enregistrée dans un fichier **IPv6_output_CSR1kv.txt** du sous-répertoire **ios_configurations**.

Étape 4: Exécutez le playbook Ansible pour configurer l'adressage IPv6 sur la machine virtuelle CSR1000v.

a. Dans la partie 1, vous avez démarré la machine virtuelle CSR1000v. Ping pour vérifier que vous pouvez y accéder. Entrez **Ctrl+ C** pour annuler le ping.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=63 time=0.913 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=63 time=0.875 ms
^C
--- 192.168.56.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0,875/0,894/0,913/0,019 ms
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

b. Créez le répertoire **ios_configurations**. Comme indiqué dans la dernière ligne de votre playbook, il s'agit du répertoire où la sortie de la commande **show ipv6 interface brief** sera stockée.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ mkdir
ios_configurations
```

c. Vous pouvez maintenant exécuter le playbook Ansible à l'aide de la commande **ansible-playbook**. L'option **-v** verbeuse peut être utilisée pour afficher les tâches effectuées dans le playbook.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook -v
cisco_router_ipv6_config_playbook.yaml
Using /home/devasc/labs/ansible-csr1000v/ansible.cfg as config file

PLAY [CONFIGURE IPv6 ADDRESSING] ***********************************************

TASK [SET IPv6 ADDRESS] ********************************************************
changed: [CSR1kv] => {"ansible_facts": {"discovered_interpreter_python":
"/usr/bin/python3"}, "banners": {}, "changed": true, "commands": ["interface
GigabitEthernet1", "description IPv6 ADDRESS", "ipv6 address 2001:db8:acad:1::1/64",
"ipv6 address fe80::1:1 link-local"], "updates": ["interface GigabitEthernet1",
"description IPv6 ADDRESS", "ipv6 address 2001:db8:acad:1::1/64", "ipv6 address
fe80::1:1 link-local"]}

TASK [SHOW IPv6 INTERFACE BRIEF] ***********************************************
ok: [CSR1kv] => {"changed": false, "stdout": ["GigabitEthernet1 [up/up]\n FE80::1:1\n
2001:DB8:ACAD:1::1"], "stdout_lines": [["GigabitEthernet1 [up/up]", " FE80::1:1", "
2001:DB8:ACAD:1::1"]]}

TASK [SAVE OUTPUT ./ios_configurations/] ***************************************
ok: [CSR1kv] => {"changed": false, "checksum":
"60784fbaae4bd825b7d4f121c450effe529b553c", "dest":
"ios_configurations/IPv6_output_CSR1kv.txt", "gid": 900, "group": "devasc", "mode":
"0664", "owner": "devasc", "path": "ios_configurations/IPv6_output_CSR1kv.txt",
"size": 67, "state": "file", "uid": 900}

PLAY RECAP *********************************************************************
CSR1kv : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

La première fois que vous lancez le playbook, le **PLAY RECAP** doit afficher **ok=3 changed= 2** et **failed = 0** indiquant une exécution réussie. Ces valeurs peuvent être différentes si vous exécutez à nouveau le playbook.

Étape 5: Vérifiez que le fichier de la sortie a été créé.

Dans VS Code, ouvrez le dossier **ios_configurations** et cliquez sur le fichier **IPv6_output_CSR1kv.txt**. Vous pouvez également utiliser la fenêtre du terminal pour afficher le fichier avec **cat ios_configurations/IPv6_output_CSR1kv.txt**. Vous disposez maintenant d'une sauvegarde de la configuration CSR1000v.

```
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cat
ios_configurations/IPv6_output_CSR1kv.txt
GigabitEthernet1 [up/up]
    FE80::1:1
    2001:DB8:ACAD:1::1
devasc@labvm:~/labs/ansible-csr1000v/ios_configurations$
```