Cisco Networking Academy® Mind Wide Open™

Práctica de laboratorio 7: Utilice Ansible para respaldar y configurar un dispositivo

Versión en inglés:

https://www.ccna6rs.com/7-4-7-lab-use-ansible-to-back-up-and-configure-a-device-answers/

Objetivos

Parte 1: Iniciar la Máquina Virtual DEVASC y la Máquina Virtual CSR1000v.

Parte 2: Configurar Ansible.

Parte 3: Usar Ansible para realizar una copia de seguridad de una configuración.

Parte 4: Usar Ansible para configurar un dispositivo.

Aspectos básicos/Situación

En este laboratorio, explorará los fundamentos de cómo utilizar Ansible para automatizar algunas tareas básicas de administración de dispositivos. Primero, configure Ansible en su máquina virtual de DEVASC. A continuación, utilice Ansible para conectarse al CSR1000v y realice una copia de seguridad de su configuración. Finalmente, configurar el CSR1000v con direccionamiento IPv6.

Recursos necesarios

- 1 Computadora con sistema operativo de su elección
- Virtual Box o VMWare
- Máquina virtual (Virtual Machine) DEVASC
- Máquina virtual CSR1000v

Instrucciones

Parte 1: Iniciar las VM DEVASC y las VM CSR1000v

Si no se ha completado el laboratorio **- Instalar el Entorno de Laboratorio de la Máquina Virtual**, hacerlo ahora. Si se ha completado ya, inicie la máquina virtual DEVASC.

Si aún no ha completado el **laboratorio - Instalar la máquina virtual DEVASC-LAB y CSR1000v**, hágalo ahora. Si ya ha completado ese laboratorio, inicie la máquina virtual CSR1000v ahora.

Parte 2: Configurar Ansible

En esta parte, configurará Ansible para que se ejecute desde un directorio específico.

Paso 1: Abrir el directorio Ansible en VS (Visual Studio) Code.

a. Abra **VS (Visual Studio) Code**.

b. Haga clic en **Archivo > Abrir carpeta...** y vaya a la carpeta **/labs/devnet-src/ansible**.

c. Haga clic en **Aceptar**.

d. Los dos subdirectorios para los laboratorios de Ansible ahora se cargan en el panel VS Code **EXPLORER** para su comodidad. En este laboratorio, trabajará con el directorio **ansible-csr1000v**.

Paso 2: Editar el fichero de inventario de Ansible.

Ansible utiliza un archivo de inventario denominado **hosts** que contiene información del dispositivo utilizada por los libros de reproducción de Ansible. La ubicación predeterminada del archivo de inventario de Ansible es **/etc/ansible/hosts**, tal como se especifica en el **ansible.cfg** predeterminado en el mismo directorio **/etc/ansible**. Estos archivos predeterminados se utilizan cuando Ansible se ejecuta globalmente. Sin embargo, en este laboratorio ejecutará Ansible desde el directorio **ansible-csr1000v**. Por lo tanto, necesita **hosts** separados y archivos **ansible.cfg** para cada laboratorio.

Nota: Los términos **fichero de hosts** y **fichero de inventario** son sinónimos y se utilizarán indistintamente en los laboratorios de Ansible.

El archivo de inventario de Ansible define los dispositivos y grupos de dispositivos que utiliza el playbook de Ansible. El archivo puede estar en uno de los muchos formatos, incluidos YAML e INI, dependiendo de su entorno Ansible. El archivo de inventario puede enumerar los dispositivos por dirección IP o nombre de dominio completo (FQDN), y también puede incluir parámetros específicos del host.

a. Abra el archivo **hosts** en el directorio **ansible-csr1000v**.

b. Agregue las siguientes líneas al archivo **hosts** y guárdelas.

```
# Ingrese los hosts o dispositivos para los playbooks de Ansible
csr1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.101
```

Después del comentario (#), el archivo **hosts** comienza con un alias, **CSr1kv**. Se utiliza un alias desde el manual de Ansible para hacer referencia a un dispositivo. Después del alias, el archivo **hosts** especifica tres variables que usará el playbook de Ansible para acceder al dispositivo. Estas son las credenciales SSH que Ansible necesita para acceder de forma segura a la máquina virtual CSR1000v.

- **ansible_user** es una variable que contiene el nombre de usuario utilizado para conectarse al dispositivo remoto. Sin esto, se usaría el usuario que está ejecutando el ansible-playbook.
- **ansible_password** es una variable que contiene la contraseña correspondiente para **ansible_user**. Si no se especifica, se usará la clave SSH.
- **ansible_host** es una variable que contiene la dirección IP o el FQDN del dispositivo.

Paso 3: Muestra la versión de Ansible y la ubicación predeterminada ansible.cfg.

a. Para ver dónde almacena Ansible el archivo ansible.cfg predeterminado, abra una ventana de terminal y navegue por un directorio hasta el directorio padre ansible.

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

b. Escriba **ansible** para obtener una lista de los comandos ansible. Observe la opción **—version**.

```
devasc @labvm: ~/labs/devnet-src/ansible$ ansible
uso: ansible [-h] [—version] [-v] [-b] [—become-método BECOME_METHOD] [—
               become-user BECOME_USER] [-K] [-i INVENTARIO] [—list-hosts]
               [-l SUBJUEGO] [-P POLL_INTERVALO] [-B SEGUNDOS] [-o] [-t ÁRBOL] [-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. Utilice el comando **ansible —version** para mostrar la información de la versión. Observe que este laboratorio está utilizando la versión 2.9.6. Ansible incluye ciertos archivos predeterminados, incluido un archivo de configuración predeterminado, **ansible.cfg**.

```
devasc @labvm: ~/labs/devnet-src/ansible$ ansible —version
ansible 2.9.6
  archivo de configuración = /etc/ansible/ansible.cfg
  ruta de búsqueda del módulo configurado = ['/home/devasc/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ubicación del módulo python ansible = /usr/lib/python3/dist-packages/ansible
  ubicación ejecutable = /usr/bin/ansible
  versión de python = 3.8.2 (predeterminado, 27 abr 2020, 15:53:34) [GCC 9.3.0]
devasc @labvm: ~/labs/devnet-src/ansible$
```

Paso 4: Mostrar el archivo ansible.cfg predeterminado.

Ansible utiliza el archivo **ansible.cfg** para establecer ciertos valores predeterminados. Estos valores se pueden modificar.

Con la ruta predeterminada mostrada en el comando **ansible —version**, muestre el archivo de configuración por defecto. Observe que este es un archivo muy largo. Puede canalizar la salida del comando **cat** a **more** para que muestre una página a la vez. Se destacan las entradas que estarán en su archivo **ansible.cfg** para este laboratorio.

```
devasc @labvm: ~/labs/devnet-src/ansible$ cat /etc/ansible/ansible.cfg | más
# archivo de configuración para ansible — https://ansible.com/
# ===============================================

# casi todos los parámetros pueden ser anulados en ansible playbook
# o con banderas de línea de comando. ansible leerá ANSIBLE_CONFIG,
# ansible.cfg en el directorio de trabajo actual, .ansible.cfg en
# el directorio de inicio o /etc/ansible/ansible.cfg, lo que sea
# encuentra primero

(Predeterminado: )

# algunos valores predeterminados básicos...

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

# descomentar esto para deshabilitar la comprobación de host de clave SSH
#host_key_checking = False
<output omitted>

# reintentar archivos
# Cuando un playbook falla, se puede crear un archivo .retry que se colocará en ~/
# Puede habilitar esta función estableciendo retry_files_enabled en True
```

```
# y puede cambiar la ubicación de los archivos estableciendo retry_files_save_path

#retry_files_enabled = False
<output omitted>
```

Observe que Ansible muestra que el archivo de **hosts** de inventario que utilizará de forma predeterminada es **/etc/ansible/hosts**. En un paso anterior, editamos el archivo de **hosts** de inventario en el directorio **ansible-csr1000v**. En el siguiente paso editaremos un nuevo **archivo ansible.cfg** que utiliza el archivo de inventario de **hosts** que creó.

Paso 5: Cambie la ubicación del archivo ansible.cfg.

a. Ansible usará el archivo de configuración ubicado en **/etc/ansible/ansible.cfg** a menos que haya un archivo **ansible.cfg** en el directorio actual. Vuelva al directorio ansible-csr1000v. Ya hay un archivo **ansible.cfg** marcador de posición en este directorio. Muestra la ubicación actual de **ansible.cfg** con el comando **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
  archivo de configuración = /home/devasc/labs/devnet-src/ansible/ansible-csr1000v/ansible.cfg
<output omitted>
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

b. Muestra el archivo para ver que está vacío, excepto por un comentario. Editará este archivo en el siguiente paso.

```
devasc @labvm: ~/labs/devnet-src/ansible/ansible-csr1000v$ cat ansible.cfg
# Añadir a este archivo para el laboratorio de Ansible
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

Paso 6: Editar el archivo ansible.cfg.

Ahora, debe editar el archivo **/ansible-csr1000v/ansible.cfg** para incluir la ubicación del archivo de inventario de **hosts**. Recuerde que el archivo de configuración predeterminado en **/etc/ansible/ansible.cfg** utiliza el archivo de inventario en /etc/ansible/hosts.

a. Abra el archivo **/ansible-csr1000v/ansible.cfg** en VS Code.

b. Puede eliminar el comentario. Agregue las siguientes líneas al archivo y guárdelo.

```
# archivo de configuración para ansible-csr1000v
(Predeterminado: )
# Usar el archivo hosts locales en esta carpeta
inventario=./hosts
host_key_checking = False # No se preocupes por las huellas dactilares de RSA
retry_files_enabled = False # No los cree
deprecation_warnings = False # No mostrar advertencias
```

Al igual que Python, el # se usa para comentarios dentro del archivo **ansible.cfg**. Si la entrada hace referencia a un nombre de archivo como **inventory=./hosts** el comentario no puede venir después de la entrada. Ansible trata el # y el comentario que sigue como parte del nombre de archivo. Por lo tanto, en estos casos, el comentario # debe estar en una línea separada. Sin embargo, las variables pueden tener un comentario en la misma línea que se muestra para **host_key_checking** y **retry_files_enabled**.

El archivo **ansible.cfg** indica a Ansible dónde encontrar el archivo de inventario y establece ciertos parámetros predeterminados. La información que ingresó en su archivo **ansible.cfg** es:

- **`inventario=./hosts`** : su archivo de inventario es el archivo de **hosts** del directorio actual.
- **`host_key_checking = False`** - El entorno de desarrollo local no tiene las claves SSH configuradas. Ha establecido el conjunto **host_key_checking** en **False**, que es el valor predeterminado. En una red de producción, **host_key_checking** se establecería en **True**.
- **`retry_files_enabled = False`** - Cuando Ansible tiene problemas para ejecutar libros de reproducción para un host, generará el nombre del host en un archivo en el directorio actual que termina en**.retry**. Para evitar el desorden, es común deshabilitar esta configuración.
- **`deprecation_warnings=false`** : las advertencias de obsolescencia indican el uso de características heredadas que están programadas para su eliminación en una versión futura de Ansible. Ha desactivado esta advertencia.

Paso 7: RESUMEN: Sus archivos de configuración de Ansible.

En esta parte, ha configurado Ansible para que se ejecute en el directorio **ansible-csr1000v**. De forma predeterminada, Ansible utiliza archivos en el directorio **/etc/ansible**. El archivo /etc/ansible/**ansible.cfg** predeterminado indica que el archivo de inventario predeterminado es **/etc/ansible/hosts**.

Sin embargo, en este laboratorio necesita un archivo hosts y un archivo ansible.cfg en su directorio ansible-csr1000v.

- Ha editado el archivo **hosts** para que contenga información de inicio de sesión y dirección IP para el router CSR1000v
- Ha editado el archivo **ansible.cfg** para utilizar el archivo hosts local como archivo de inventario (**inventory=./hosts**).

En la siguiente Parte, creará un playbook para decirle a Ansible qué hacer.

Parte 3: Usar Ansible para realizar una copia de seguridad de una configuración

En esta parte, creará playbook de Ansible que automatizará el proceso de copia de seguridad de la configuración del CSR1000v. Los playbooks están en el centro de Ansible. Cuando desee que Ansible obtenga información o realice una acción en un dispositivo o grupo de dispositivos, ejecute un playbook para realizar el trabajo.

Un playbook de Ansible es un archivo YAML que contiene una o más reproducciones. Cada obra es una colección de tareas.

- Un **juego** es un conjunto de tareas que coincidan con un dispositivo o grupo de dispositivos.
- Una **tarea** es una acción única que hace referencia a un **módulo** para ejecutarse junto con los argumentos y acciones de entrada. Estas tareas pueden ser simples o complejas dependiendo de la necesidad de permisos, el orden de ejecución de las tareas, etc.

Un playbook también puede contener **roles**. Un rol es un mecanismo para dividir un playbook en varios componentes o archivos, simplificando el playbook y facilitando su reutilización. Por ejemplo, el rol **común** se utiliza para almacenar tareas que se pueden usar en todos los playbooks. Los roles están fuera del alcance de este laboratorio.

El manual de Ansible YAML incluye **objetos**, **listas** y **módulos**.

- Un objeto YAML es uno o más pares clave-valor. Los pares clave-valor están separados por dos puntos sin el uso de comillas, por ejemplo **hosts: CSR1kv.**
- Un objeto puede contener otros objetos, como una lista. YAML utiliza listas o matrices. Se utiliza un guión "-" para cada elemento de la lista.
- Ansible se envía con una serie de módulos (llamados biblioteca de módulos) que se pueden ejecutar directamente en hosts remotos o a través de playbooks. Un ejemplo es el módulo **ios_command** utilizado

para enviar comandos a un dispositivo IOS y devolver los resultados. Por lo general, cada tarea consta de uno o más módulos de Ansible.

Ejecute un playbook de Ansible utilizando el comando **ansible-playbook**, por ejemplo:

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

El comando **ansible-playbook** utiliza parámetros para especificar:

- El playbook que desea ejecutar (**backup_cisco_router_playbook.yaml**)
- El fichero de inventario y su ubicación (**-i hosts**).

Paso 1: Crear un playbook Ansible.

El playbook de Ansible es un archivo YAML. Asegúrese de utilizar la sangría YAML adecuada. Cada espacio y guión es significativo. Puede perder algo de formato si copia y pega el código en este laboratorio.

a. En VS Code, cree un nuevo archivo en el directorio **ansible-csr1000v** con el siguiente nombre: **backup_cisco_router_playbook.yaml**

1) Agregue la siguiente información al archivo.

```
---
- name: AUTOMATIC BACKUP OF RUNNING-CONFIG
  hosts: CSR1kV
  gather_facts: false
  connection: local

  tareas:
   - name: DISPLAYING THE RUNNING-CONFIG
     ios_command:
       a continuación:
         - show running-config
     register: config

   - name: SAVE OUTPUT TO ./backups/
     copy:
       content: "{{ config.stdout[0] }}"
       dest: «backups/show_run_ {{inventory_hostname}} .txt»
```

Paso 2: Examinar su playbook de Ansible.

El playbook que ha creado contiene una obra con dos tareas. La siguiente es una explicación de su playbook:

- – Esto está al principio de cada archivo YAML, lo que indica a YAML que se trata de un documento separado. Cada archivo puede contener varios documentos separados por –
- **`name: Copia de seguridad automática de ejecución`** - Este es el nombre de la obra.
- **`hosts: CSr1kv`** - Este es el alias del archivo **hosts** previamente configurado. Al referirse a este alias en su playbook, el libro de jugadas puede utilizar todos los parámetros asociados a esta entrada del archivo de inventario, que incluye el nombre de usuario, la contraseña y la dirección IP del dispositivo.
- **`gather_facts: false`** - Ansible fue diseñado originalmente para trabajar con servidores Linux, copiando módulos Python a los servidores para la automatización de tareas. Esto no es necesario cuando se trabaja con dispositivos de red.

- **`connection: local`** : especifica que no está utilizando SSH, por lo tanto, la conexión es local.
- **`tasks:`** - Esta palabra clave indica una o más tareas que se van a realizar.

La primera tarea es mostrar el running-config.

- `- name: VISUALIZACION DE LA CONFIGURACION EJECUCION` - Nombre de la tarea.
- **`ios_command:`** - Este es un **módulo** Ansible que se utiliza para enviar comandos a un dispositivo IOS y devolver los resultados leídos desde el dispositivo. Sin embargo, no admite comandos de configuración. El módulo **ios_config** se utiliza para este propósito, como verá en la siguiente parte de este laboratorio.

  Nota: En el terminal Linux, puede usar el *comando ansible-doc* module_name *para ver las páginas de manual de cualquier* **módulo** y los parámetros asociados a ese módulo. (por ejemplo, **ansible-doc ios_command**)
- **`comandos:`** - Este parámetro está asociado con el módulo **ios_command**. Se utiliza para enumerar comandos IOS en el playbook que se van a enviar al dispositivo IOS remoto. Se devuelve el resultado resultante del comando.
- **`- show running-config`** - Este es el comando Cisco IOS enviado usando el módulo **ios_command**.
- **`register: config`** - Ansible incluye registros utilizados para capturar el resultado de una tarea a una variable. Esta entrada especifica que la salida del comando **show running-config** anterior se almacenará en la **configuración** de la variable.

La segunda tarea es guardar la salida:

- **`- name: SAVE OUTPUT TO./backups/`** - Nombre de la tarea
- **`copy:`** - Este es un **módulo** Ansible utilizado para copiar archivos en una ubicación remota. Hay dos parámetros asociados con este módulo:
  - **`content:`** `«{{config.stdout [0]}}»` - El valor especificado para este parámetro son los datos almacenados en la variable de **configuración**, la variable de registro Ansible utilizada en latarea anterior. La salida estándar (**stdout**) es el descriptor de archivo predeterminado donde un proceso puede escribir la salida utilizada en sistemas operativos tipo UNIX, como Linux y Mac OSX.
  - **`dest: «backups/show_run_ {{inventory_hostname}} .txt»`** - Esta es la ruta y el nombre del archivo donde se debe copiar el archivo. La variable **inventory_hostname** es una "variable mágica" de Ansible que recibe automáticamente el nombre de host tal y como se configura en el archivo **hosts** . En su caso, recuerde que esto es **CSR1kV**. Este parámetro da como resultado un archivo **show_run_CSR1kv.txt** almacenado en el directorio de **copias** de seguridad. El archivo contendrá la salida del comando **show running-config** . Creará el directorio de **backups** de seguridad en el siguiente paso.

Paso 3: Ejecute el playbook copia de seguridad de Ansible.

a. En la parte 1, inició la máquina virtual CSR1000v. Haga ping para verificar que hay acceso acceder a él. Escriba **Ctrl+ C** para anular el ping.

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

b. Cree el directorio **backups**. Como se indica en la última línea de su playbook, este es el directorio donde se almacenará el archivo de configuración de copia de seguridad.

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

c. Ahora puede ejecutar el playbook de Ansible usando el comando **ansible-playbook** :

```
devasc @labvm: ~/labs/devnet-src/ansible$ 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/] ************************************************
cambiado: [CSr1kV]

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

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

Nota: En muchos ejemplos verá la ejecución del playbook usando la **opción -i** *inventory-file name*. Por ejemplo:

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

Esta opción indica a Ansible la ubicación y el nombre del archivo de inventario, la lista de dispositivos que utilizará el playbook. Esta opción no es necesaria porque configuró el nombre y la ubicación del archivo de inventario en el archivo **ansible.cfg** local: inventory=./hosts. Puede utilizar la opción **-i** *inventory-filename* para anular la información del archivo **ansible.cfg** .

El **PLAY RECAP** debe mostrar **ok=2 changed=1** indicando una ejecución exitosa del playbook.

Si su playbook de Ansible falla, algunas de las cosas que se deben revisar en él son:

- Asegurarse de que sus **hosts** y archivos **ansible.cfg** sean correctos.
- Asegurarse de que la sangría YAML es correcta.
- Asegurarse de que el comando IOS sea correcto.
- Comprobar toda la sintáxis del playbook de Ansible.
- Verificar que puede hacer ping al CSR1000v.

Si sigue teniendo problemas:

- Intentar escribir una línea a la vez y ejecutar el playbook cada vez.
- Comparar su archivo con el playbook de soluciones en el directorio **ansible_solutions**.

Paso 4: Verificar que se ha creado el archivo de copia de seguridad.

En VS Code, abra la carpeta de **copias** de seguridad y abra el archivo **show_run_CSR1kv.txt**. También puede utilizar la ventana de terminal con "cat" en el archivo con **cat backups/show_run_CSR1kv.txt** Ahora tiene una copia de seguridad de la configuración CSR1000v.

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

Parte 4: Usar Ansible para configurar un dispositivo

En esta parte, creará otro playbook de Ansible para configurar el direccionamiento IPv6 en el router CSR1000v.

Paso 1: Ver el archivo de inventario de hosts.

a. Vuelva a examinar el archivo de inventario de **hosts**. Como recordatorio, este archivo contiene el alias **CSr1kv** y tres variables de inventario para el nombre de usuario, la contraseña y la dirección IP del host. El playbook de esta parte también utilizará este archivo y el archivo **ansible.cfg** que creó al principio del laboratorio.

```
devasc @labvm: ~/labs/devnet-src/ansible$ hosts cat
# Ingresar los hosts o dispositivos para los manuales de Ansible
csr1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.101
devasc @labvm: ~/labs/devnet-src/ansible$
```

Paso 2: Crear un nuevo playbook.

a. En VS Code, cree un nuevo archivo en el directorio **ansible-csr1000v** con el siguiente nombre: **cisco_router_ipv6_config_playbook.yaml**

b. Agregue la siguiente información al archivo. Asegúrese de utilizar la sangría YAML adecuada. Cada espacio y guión es significativo. Puede perder algo de formato si copia y pega.

```
---
- name: CONFIGURE IPv6 ADDRESSING
  hosts: CSR1kV
  gather_facts: false
  connection: local

  tareas:
   - 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:
    a continuación:
      - show ipv6 interface brief
  register: output

- name: SAVE OUTPUT ./ios_configurations/
  copy:
    content: "{{ output.stdout[0] }}"
    dest: «IOS_Configurations/IPv6_Output_ {{inventory_hostname}} .txt»
```

Paso 3: Examinar su playbook de Ansible.

Gran parte de este playbook es similar al que creó en la parte anterior. La principal diferencia es la primera tarea CONFIGURAR DIRECCIÓN IPv6..

A continuación, se presenta una breve descripción de los elementos de la tarea:

- **`ios_config:`** - Este es un módulo Ansible utilizado para configurar un dispositivo IOS. Puede utilizar el comando **ansible-doc ios_config** para ver los detalles de los parámetros de **líneas** y **padres** utilizados en playbook.
- **`parents: "interface Gigabitethernet1"`** - Este parámetro indica el modo de configuración de la interfaz IOS.
- **`lines:`** - En esta sección se configura un conjunto ordenado de comandos IOS, especificando la información de direccionamiento IPv6 para la interfaz GigabitetherNet1.

El resto del playbook es similar a las tareas de la parte anterior. La segunda tarea utiliza el módulo **ios_command** y el comando **show ipv6 interface brief** para mostrar el resultado y enviarlo a la **salida** del registro.

La última tarea guarda la información de la **salida** del registro en un archivo **IPv6_output_CSR1kv.txt** en el subdirectorio **ios_configurations**.

Paso 4: Ejecutar el playbook de Ansible para configurar el direccionamiento IPv6 en la máquina virtual CSR1000v.

a. En la parte 1, inició la máquina virtual CSR1000v. Haga ping para verificar que se puede acceder a él. Escriba **Ctrl+ C** para anular el ping.

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

b. Cree el directorio **ios_configurations**. Como se indica en la última línea de su playbook, este es el directorio donde se almacenará la salida del comando **show ipv6 interface brief**.

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

c. Ahora puede ejecutar el playbook de Ansible usando el comando **ansible-playbook**. La opción **-v** verbose se puede utilizar para mostrar las tareas que se están realizando en el playbook.

```
devasc @labvm: ~/labs/devnet-src/ansible$ ansible-playbook -v
cisco_router_ipv6_config_playbook.yaml
Usar /home/devasc/labs/ansible-csr1000v/ansible.cfg como archivo de configuración

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$
```

La primera vez que ejecute el playbook, el **PLAY RECAP** debe mostrar **ok=3 changed=2** y **failed=0** indicando una ejecución exitosa. Estos valores pueden ser diferentes si vuelve a ejecutar el playbook.

Paso 5: Verificar que se ha creado el archivo de la salida.

En VS Code, abra la carpeta **ios_configurations** y haga clic en el archivo **IPv6_output_CSR1kv.txt**. También puede utilizar la ventana de terminal para ver el archivo con **cat ios_configurations/IPv6_output_CSR1kv.txt**. Ahora tiene una copia de seguridad de la configuración 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$
```