

UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE CIENCIAS



ÁREA DE CIENCIA DE LA COMPUTACIÓN
7° LABORATORIO - CC312

- **TÍTULO:** Utilice Ansible para respaldar y configurar un dispositivo
- **ALUMNO:**
JHONATAN POMA MARTINEZ 20182729F
- **PROFESORES:** YURI JAVIER.,CCOICCA PACASI

2023

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.

Resumen:

DEVASC-LABVM: Es una máquina virtual diseñada específicamente para el aprendizaje y la práctica de habilidades relacionadas con el examen de certificación Cisco DevNet. Proporciona un entorno preconfigurado con herramientas y recursos para realizar laboratorios y prácticas relacionadas con el desarrollo de aplicaciones, automatización de redes y otros temas cubiertos en el examen.

CSR1000v: Es una versión virtualizada del **router Cisco ISR 1000 Series**. Es un router virtualizado que se ejecuta como una máquina virtual y permite emular funciones de enrutamiento y servicios de red en un entorno virtual. El CSR1000v se utiliza principalmente para casos de uso como el despliegue de enrutamiento y servicios de red en entornos de nube, laboratorios de prueba y desarrollo, simulación de redes y otras aplicaciones relacionadas con la infraestructura de red.

ANSIBLE: Ansible es una herramienta de automatización de TI (tecnología de la información), que se utiliza para administrar, configurar sistemas informáticos y redes. Permite a los administradores de sistemas automatizar tareas repetitivas, como el aprovisionamiento de servidores, la configuración de software y la implementación de aplicaciones.

Ansible utiliza un enfoque basado en lenguaje de programación para describir y automatizar la configuración de sistemas. Utiliza un lenguaje declarativo llamado YAML

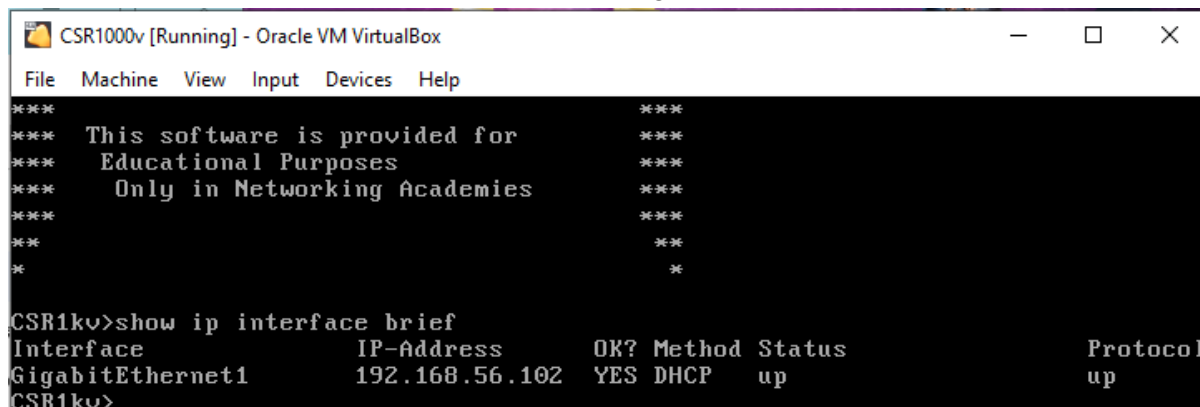
PLAYBOOK de ANSIBLE: Un playbook es un archivo en formato YAML que contiene una serie de instrucciones y tareas que Ansible debe ejecutar en un conjunto de sistemas o servidores.

Un playbook de Ansible describe el estado deseado del sistema y las acciones que deben realizarse para alcanzar ese estado. Puede contener tareas como la instalación de paquetes, la configuración de archivos de configuración, la gestión de servicios, la copia de archivos, la ejecución de comandos y cualquier otra acción necesaria para configurar y administrar los sistemas de forma automatizada.

Los playbooks de Ansible lo ejecutaremos mediante el comando **`ansible-playbook`**, que toma como entrada el archivo YAML del playbook y lo ejecuta en los sistemas especificados en un inventario. Ansible se encarga de conectarse a los sistemas remotos a través de SSH y ejecutar las tareas definidas en el playbook.

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

Iniciamos el router virtual CSR1000v, al ingresar el comando “show ip interface brief” nos mostrará un resumen de la interfaces IP configuradas.



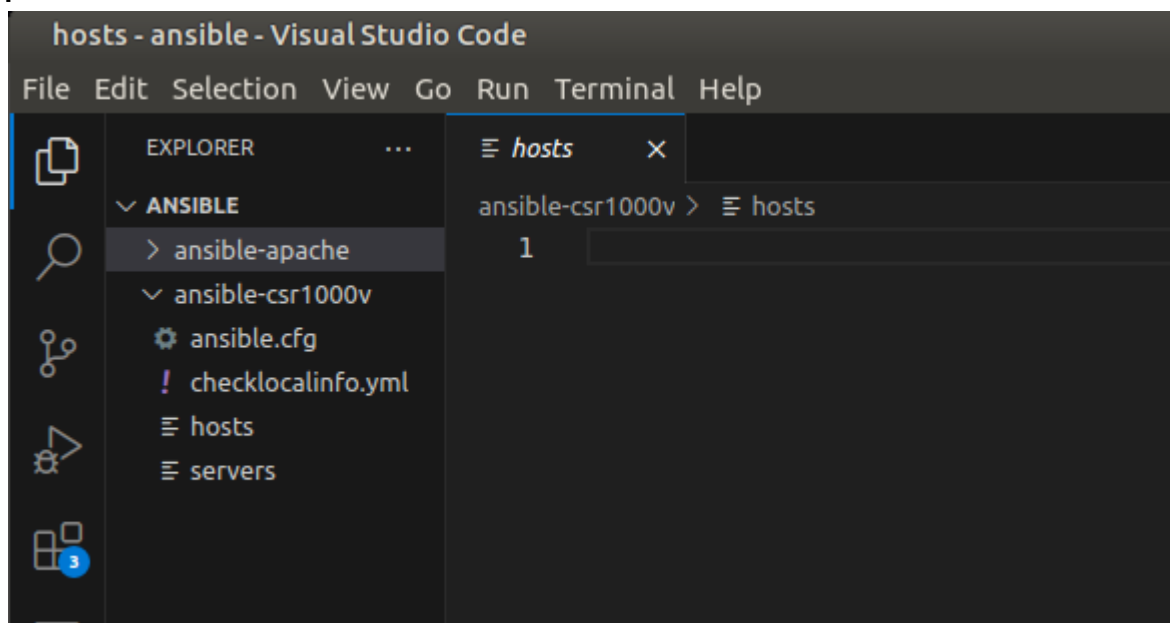
```
CSR1000v [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

****
**** This software is provided for ****
**** Educational Purposes ****
**** Only in Networking Academies ****
****
****
**
*

CSR1kv>show ip interface brief
Interface          IP-Address      OK? Method Status          Protocol
GigabitEthernet1   192.168.56.102  YES DHCP    up              up
CSR1kv>
```

PARTE 2: Configurar Ansible.

paso 1 :abrimos en vscode el archivo **ansible-csr1000v**



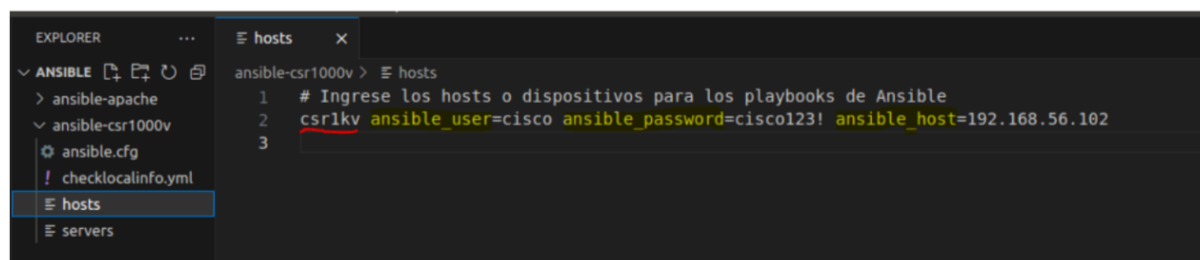
```
hosts - ansible - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
  ANSIBLE
    > ansible-apache
    > ansible-csr1000v
      ansible.cfg
      ! checklocalinfo.yml
      hosts
      servers

ansible-csr1000v > hosts
1
```

paso2 :Editamos el fichero de inventario de Ansible.

Para ello ya hemos inicializado nuestro CSR1kv, y ahi obtendremos la IP para poder conectarnos y eso lo escribimos en este script.



```
EXPLORER
  ANSIBLE
    > ansible-apache
    > ansible-csr1000v
      ansible.cfg
      ! checklocalinfo.yml
      hosts
      servers

ansible-csr1000v > hosts
1 # Ingrese los hosts o dispositivos para los playbooks de Ansible
2 csr1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.102
3
```

paso3: Muestra la versión de Ansible y la ubicación predeterminada ansible.cfg.

```
devasc@labvm: ~/labs/devnet-src/ansible
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cd ..
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]
               [--sftp-extra-args SFTP_EXTRA_ARGS]
               [--scp-extra-args SCP_EXTRA_ARGS]
               [--ssh-extra-args SSH_EXTRA_ARGS] [-C] [--syntax-check] [-D]
               [-e EXTRA_VARS] [--vault-id VAULT_IDS]
               [--ask-vault-pass | --vault-password-file VAULT_PASSWORD_FILES]
               [-f FORKS] [-M MODULE_PATH] [--playbook-dir BASEDIR]
               [-a MODULE_ARGS] [-m MODULE_NAME]
               pattern
ansible: error: the following arguments are required: pattern
devasc@labvm:~/labs/devnet-src/ansible$
```

seleccionamos para ver la versión de ansible,

```
devasc@labvm:~/labs/devnet-src/ansible$ ansible --version
ansible 2.9.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/devasc/.ansible/plugins/modules', '/usr/
share/ansible/plugins/modules']
  ansible python module location = /home/devasc/.local/lib/python3.8/site-packages
/ansible
  executable location = /home/devasc/.local/bin/ansible
  python version = 3.8.2 (default, Apr 27 2020, 15:53:34) [GCC 9.3.0]
devasc@labvm:~/labs/devnet-src/ansible$
```

paso4: Mostramos el archivo ansible.cfg predeterminado.

Ansible utiliza el archivo **ansible.cfg** para establecer ciertos valores predeterminados. Estos valores se pueden modificar. Se destacan las entradas que estarán en su archivo ansible.cfg para este laboratorio.

Observaremos que Ansible muestra que el archivo de hosts de inventario que utilizará de forma predeterminada es **/etc/ansible/hosts**.

```
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...

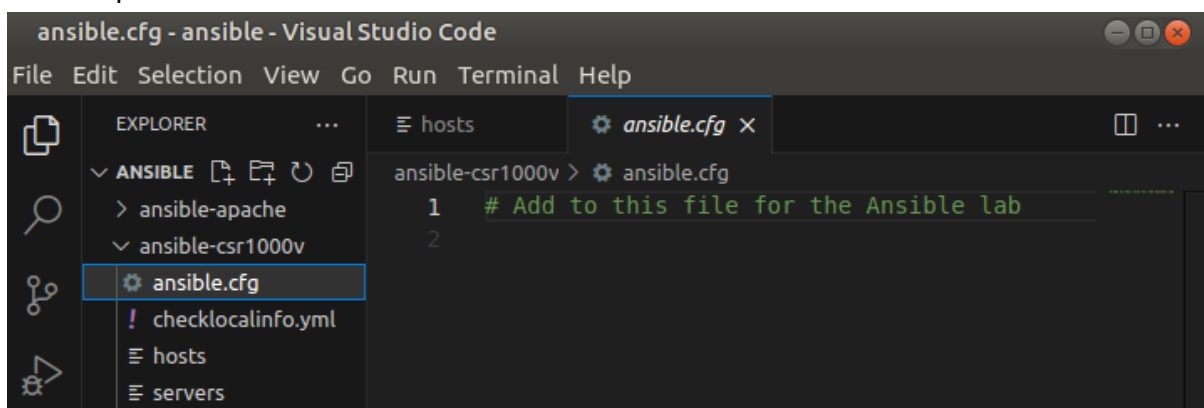
#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass  = True
--More--
```

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

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**. Mostramos el archivo “**cat ansible.cfg**” para ver que está vacío, excepto por un comentario.

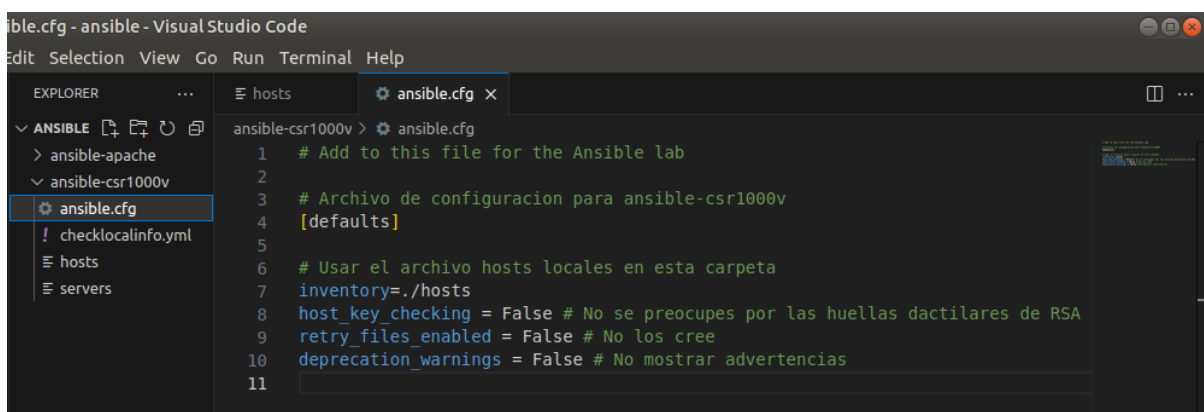
```
devasc@labvm: ~/labs/devnet-src/ansible/ansible-csr1000v
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/ansible$ cd ansible-csr1000v/
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible --version
ansible 2.9.9
  config file = /home/devasc/labs/devnet-src/ansible/ansible-csr1000v/ansible.cfg
  configured module search path = ['/home/devasc/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/devasc/.local/lib/python3.8/site-packages/ansible
  executable location = /home/devasc/.local/bin/ansible
  python version = 3.8.2 (default, Apr 27 2020, 15:53:34) [GCC 9.3.0]
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
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$
```

también podemos abrirlo mediante Visual Code.



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**.



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:

- **inventory=./hosts** : su archivo de inventario es el archivo de hosts del directorio actual.

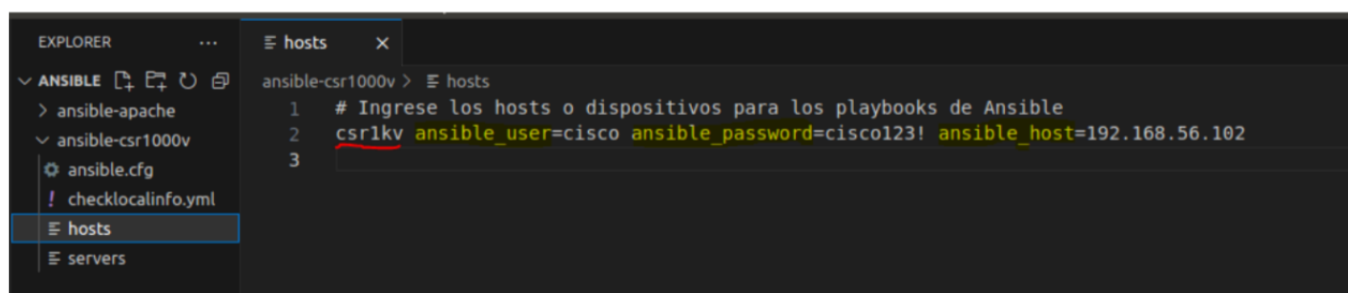
- **host_key_checking = False** - El entorno de desarrollo local no tiene las claves SSH configuradas. Ha establecido el conjunto `host_key_checking` en `False`, que es el valor predeterminado. En una red de producción, `host_key_checking` se establecería en `True`.
- **retry_files_enabled = False** - 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 necesitaremos **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.



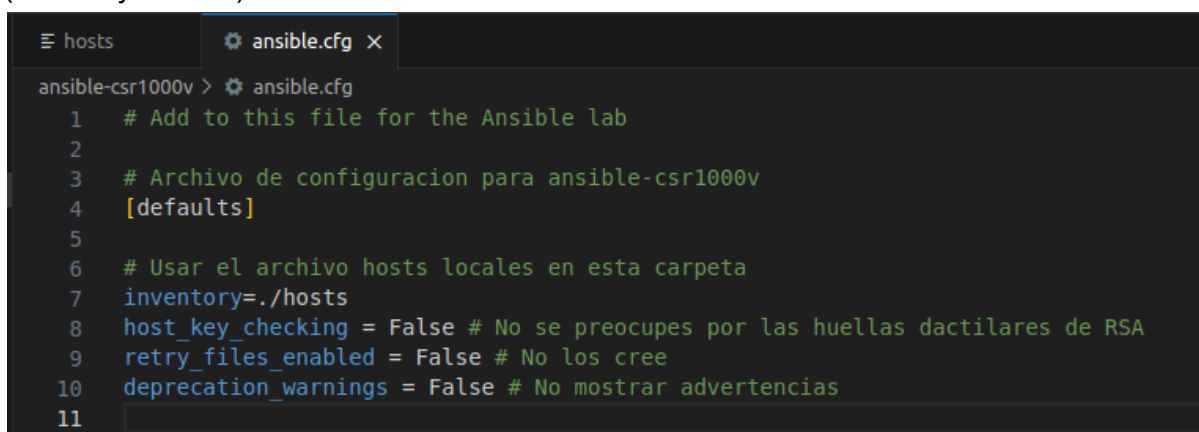
```

EXPLORER
├── ansible-apache
├── ansible-csr1000v
│   ├── ansible.cfg
│   ├── checklocalinfo.yml
│   └── hosts
└── servers

ansible-csr1000v > hosts
1 # Ingrese los hosts o dispositivos para los playbooks de Ansible
2 csr1kv ansible_user=cisco ansible_password=cisco123! ansible_host=192.168.56.102
3

```

Ha **editado el archivo ansible.cfg** para utilizar el archivo hosts local como archivo de inventario (`inventory=./hosts`).



```

hosts  ansible.cfg
ansible-csr1000v > ansible.cfg
1 # Add to this file for the Ansible lab
2
3 # Archivo de configuracion para ansible-csr1000v
4 [defaults]
5
6 # Usar el archivo hosts locales en esta carpeta
7 inventory=./hosts
8 host_key_checking = False # No se preocupe por las huellas dactilares de RSA
9 retry_files_enabled = False # No los cree
10 deprecation_warnings = False # No mostrar advertencias
11

```

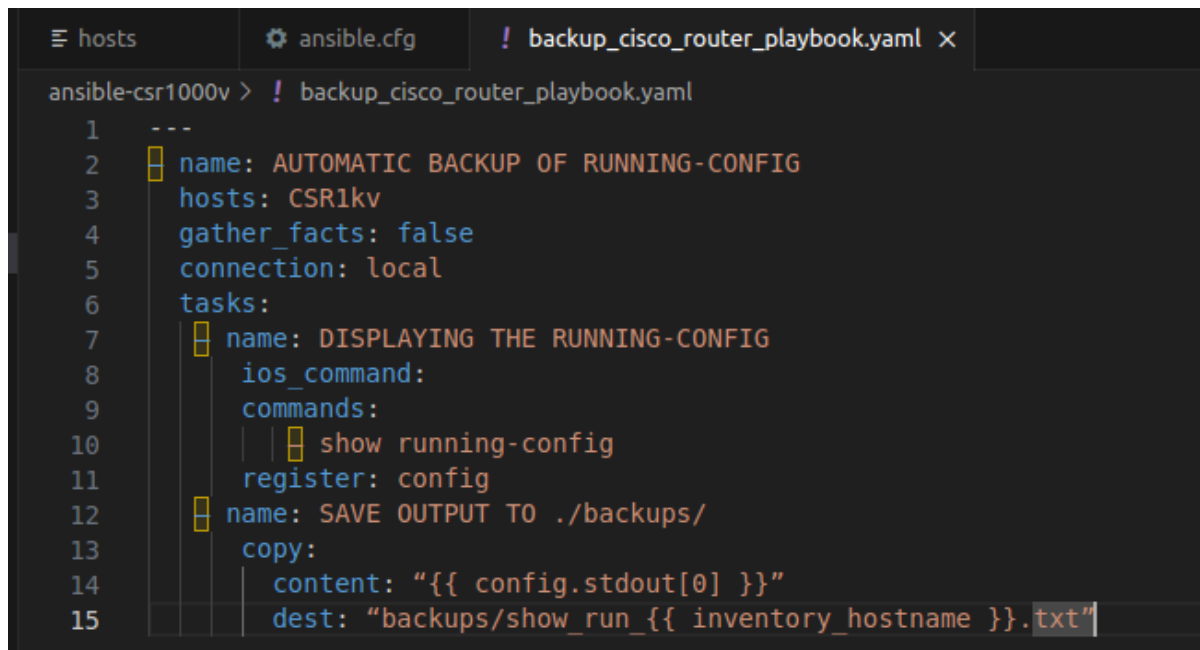
En la siguiente Parte, crearemos un playbook para decirle a Ansible qué hacer.

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

Paso 1: Crear un playbook Ansible.

El playbook de Ansible es un archivo YAML. Asegúramos 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



```
ansible-csr1000v > ! backup_cisco_router_playbook.yaml
1  ---
2  name: AUTOMATIC BACKUP OF RUNNING-CONFIG
3  hosts: CSR1kv
4  gather_facts: false
5  connection: local
6  tasks:
7    name: DISPLAYING THE RUNNING-CONFIG
8    ios_command:
9      commands:
10       - show running-config
11      register: config
12    name: SAVE OUTPUT TO ./backups/
13    copy:
14      content: "{{ config.stdout[0] }}"
15      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`)

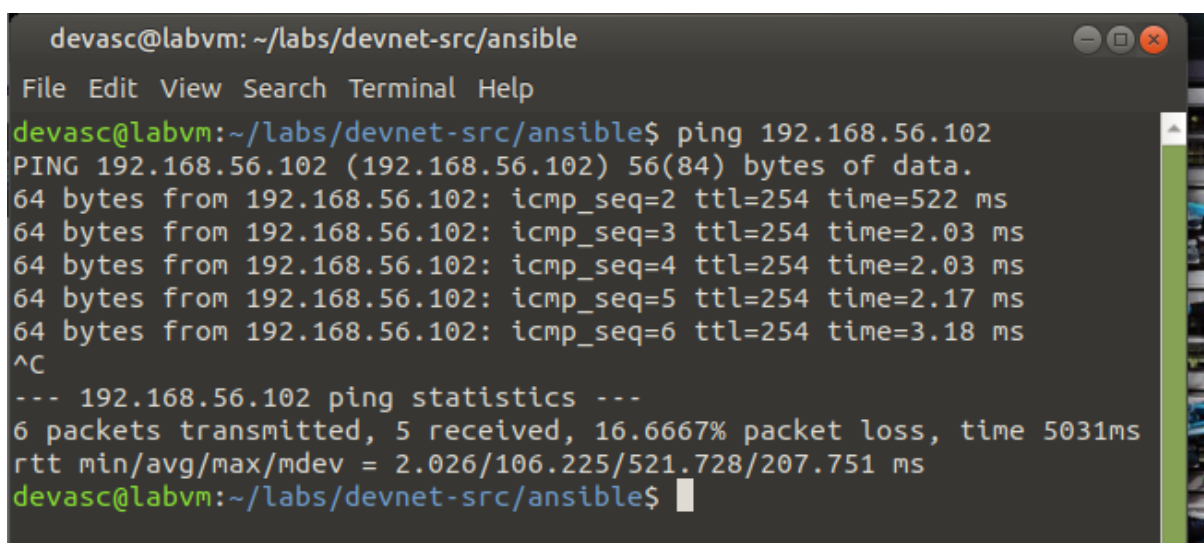
- **commands:** - Este parámetro está asociado con el módulo `ios_command`. Se utiliza para enumerar comandos IOS en el playbook que se van a enviar al dispositivo IOS remoto. Se devuelve el resultado resultante del comando.
- **- show running-config** - Este es el comando Cisco IOS enviado usando el 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:
 1. **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 la tarea 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.
 2. **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, iniciamos la máquina virtual CSR1000v. Hacemos ping para verificar que podamos acceder a él.

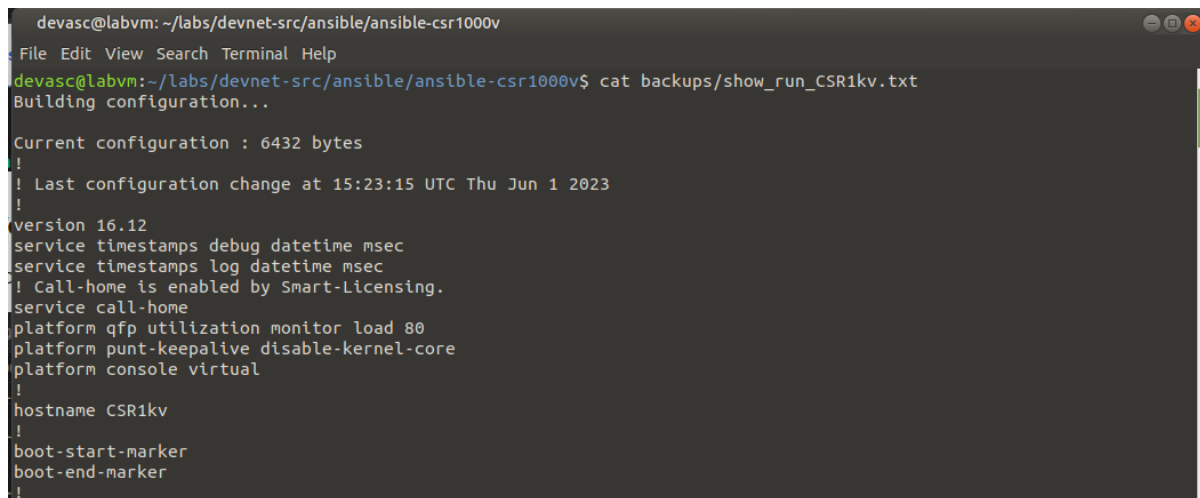


```
devasc@labvm: ~/labs/devnet-src/ansible
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/ansible$ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data:
64 bytes from 192.168.56.102: icmp_seq=2 ttl=254 time=522 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=254 time=2.03 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=254 time=2.03 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=254 time=2.17 ms
64 bytes from 192.168.56.102: icmp_seq=6 ttl=254 time=3.18 ms
^C
--- 192.168.56.102 ping statistics ---
6 packets transmitted, 5 received, 16.6667% packet loss, time 5031ms
rtt min/avg/max/mdev = 2.026/106.225/521.728/207.751 ms
devasc@labvm:~/labs/devnet-src/ansible$
```


b. Cree el directorio backups. Como indicamos en la última línea de nuestro playbook, este es el directorio donde se almacenará el archivo de configuración de **copia de seguridad**. Los playbooks de Ansible lo ejecutaremos mediante el comando ``ansible-playbook`` + “**nombre archivo YAML**”, y veremos que se ejecutan la tareas y en una ellas realiza la copia de seguridad.



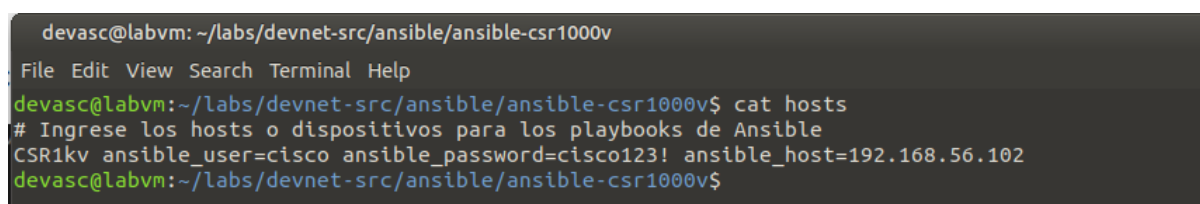
Verificamos que si se creo el archivo de copia de seguridad en la carpeta backups.



PARTE 4: Usar Ansible para configurar un dispositivo.

Crearemos otro playbook de Ansible para configurar el direccionamiento IPv6 en el router CSR1000v.

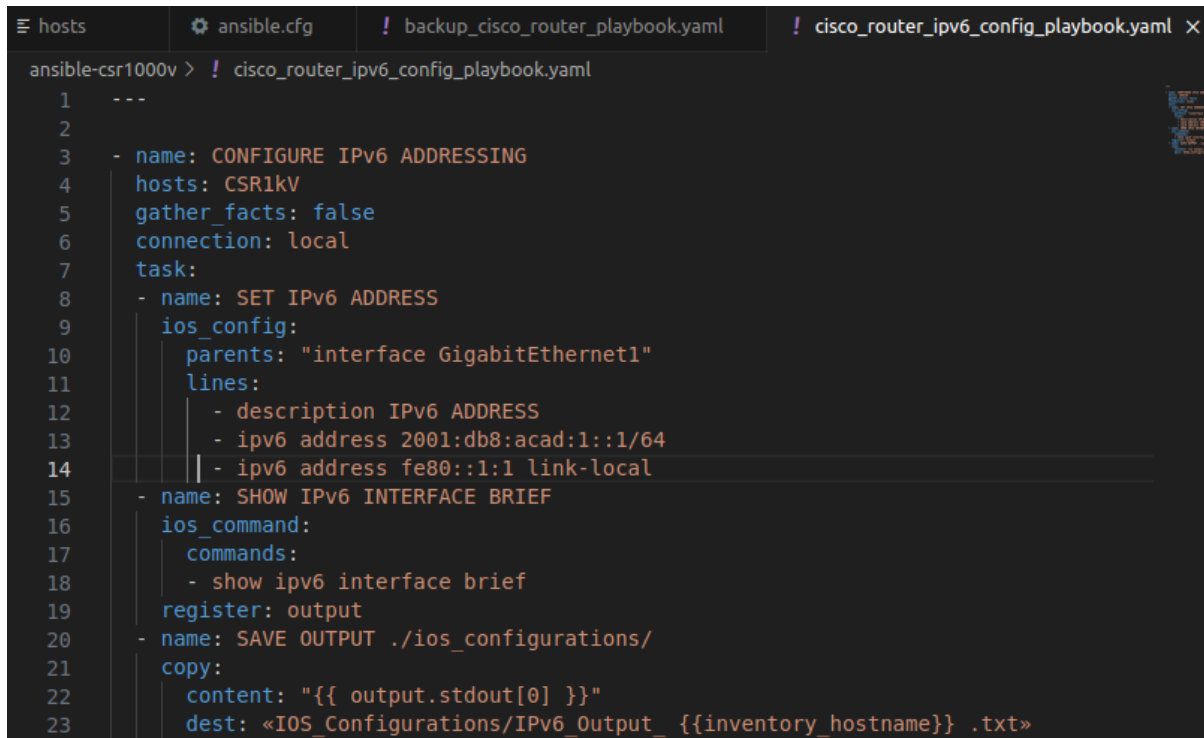
PASO 1: Vemos el archivo de inventario de hosts.



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.



```
1  ---
2
3  - name: CONFIGURE IPv6 ADDRESSING
4    hosts: CSR1kv
5    gather_facts: false
6    connection: local
7    task:
8      - name: SET IPv6 ADDRESS
9        ios_config:
10         parents: "interface GigabitEthernet1"
11         lines:
12           - description IPv6 ADDRESS
13           - ipv6 address 2001:db8:acad:1::1/64
14           - ipv6 address fe80::1:1 link-local
15      - name: SHOW IPv6 INTERFACE BRIEF
16        ios_command:
17         commands:
18           - show ipv6 interface brief
19         register: output
20      - name: SAVE OUTPUT ./ios_configurations/
21        copy:
22         content: "{{ output.stdout[0] }}"
23         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(TASK) 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: Ejecutamos el playbook para configurar el direccionamiento IPv6 en CSR1000v

```
devasc@labvm: ~/labs/devnet-src/ansible/ansible-csr1000v
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook -v cisco_router_ipv6_config_playbook.yaml
Using /home/devasc/labs/devnet-src/ansible/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] => {"stdout": [{"GigabitEthernet1 [up/up]\n FE80::1:1\n 2001:DB8:ACAD:1::1\n\n", "stdout_lines": [{"GigabitEthernet1 [up/up]", " FE80::1:1", " 2001:DB8:ACAD:1::1"}]}

TASK [SAVE OUTPUT ./ios_configurations/] *****
changed: [CSR1kv] => {"changed": true, "checksum": "60784fbaae4bd825b7d4f121c450effe529b553c", "dest": "ios_configurations/IPv6_output_CSR1kv.txt", "gid": 900, "group": "devasc", "md5sum": "56e879f15e6e776cf131cec5abfc1886", "mode": "0664", "owner": "devasc", "size": 67, "src": "/home/devasc/.ansible/tmp/ansible-tmp-1685643699.0458863-8943-242054421103032/source", "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$
```

paso 5: Verificamos que se haya creado el archivo



Conclusiones y observaciones:

- El router virtual CSR1000v no me funcionaba en el virtual box version 7 y buscando informacion si funciona en la version 6.1, probablemente debido a una incompatibilidad entre la version7 de virtual box y la imagen de CSR1000v.
- Desde el momento en que podamos hacer ping en los hosts a través de Ansible, ya podemos empezar a automatizar el entorno. Comenzamos con tareas pequeñas.
- Ansible nos permite automatizar tareas de configuración y copias de seguridad en dispositivos de red y servidores, con lo que reducimos la carga de trabajo manual y minimizamos los errores humanos
- Creamos un playbook que se encargara de mostrar la configuraciones en ejecucion del dispositivo CSR1kv y tambien realizar un archivo respaldo en la carpeta que creamos “backups”.
- El otro playbook que creamos se encargara de configurar direcciones IPv6 en la interfaz GigabitEthernet1 de un dispositivo CSR1kv, nos mostrara información de la interfaz IPv6 y tambien guardara la salida en un archivo que creamos.

- Ansible utilizara estos playbooks, lo que le permitira reutilizar y compartir configuraciones y tareas comunes entre diferentes dispositivos.