# Lab - Use Ansible to Back Up and Configure a Device

## Objectives

**Part 1: Launch the DEVASC VM and CSR1000v VM**

**Part 2: Configure Ansible**

**Part 3: Use Ansible to Back up a Configuration**

**Part 4: Use Ansible to Configure a Device**

## Background / Scenario

In this lab, you will explore the fundamentals of how to use Ansible to automate some basic device management task. First, you will configure Ansible in your DEVASC VM. Next, you will use Ansible to connect to the CSR1000v and back up its configuration. Finally, you will configure the CSR1000v with IPv6 addressing.

## Required Resources

- 1 PC with operating system of your choice
- Virtual Box or VMWare
- DEVASC Virtual Machine
- CSR1000v Virtual Machine

## Instructions

## Part 1: Launch the DEVASC VM and CSR1000v VMs

If you have not already completed the **Lab - Install the Virtual Machine Lab Environment**, do so now. If you have already completed that lab, launch the DEVASC VM now.

If you have not already completed the **Lab - Install the DEVASC-LAB and CSR1000v VM**, do so now. If you have already completed that lab, launch the CSR1000v VM now.

## Part 2: Configure Ansible

In this part, you will configure Ansible to run from a specific directory.

### Step 1: Open the Ansible directory in VS Code.

a. Open **VS Code**.

b. Click **File > Open Folder...** and navigate to the **/labs/devnet-src/ansible** folder.

c. Click **OK**.

d. The two subdirectories for the Ansible labs are now loaded in the VS Code **EXPLORER** pane for your convenience. In this lab, you will work with the **ansible-csr1000v** directory.

### Step 2: Edit the Ansible inventory file.

Ansible uses an inventory file called **hosts** that contains device information used by Ansible playbooks. The default location for the Ansible inventory file is **/etc/ansible/hosts** as specified in the default **ansible.cfg** in the same **/etc/ansible** directory. These default files are used when Ansible is run globally. However, in this lab you will run Ansible from the **ansible-csr1000v** directory. Therefore, you need separate **hosts** and **ansible.cfg** files for each lab.

**Note**: The terms **hosts file** and **inventory file** are synonymous and will be used interchangeably throughout the Ansible labs.

The Ansible inventory file defines the devices and groups of devices that are used by the Ansible playbook. The file can be in one of many formats, including YAML and INI, depending on your Ansible environment. The inventory file can list devices by IP address or fully qualified domain name (FQDN), and may include host specific parameters as well.

a. Open the **hosts** file in the **ansible-csr1000v** directory.

b. Add the following lines to the **hosts** file and save.

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

After the comment (#), the **hosts** file begins with an alias, **CSR1kv**. An alias is used from within the Ansible playbook to reference a device. After the alias, the **hosts** file specifies three variables that will be used by the Ansible playbook to access the device. These are the SSH credentials Ansible needs to securely access the CSR1000v VM.

- **ansible_user** is a variable containing the username used to connect to the remote device. Without this, the user that is running the ansible-playbook would be used.
- **ansible_password** is a variable containing the corresponding password for **ansible_user**. If not specified, the SSH key would be used.
- **ansible_host** is a variable containing the IP address or FQDN of the device.

## Step 3: Display the Ansible version and default ansible.cfg location.

a. To see where Ansible stores the default ansible.cfg file, open a terminal window and navigate up one directory to the ansible parent directory.

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

b. Type **ansible** to get a list of the ansible commands. Notice the **--version** option.

```
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. Use the **ansible --version** command to display version information. Notice that this lab is using version 2.9.6. Ansible includes certain default files, including a default configuration file, **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$
```

## Step 4: Display the default ansible.cfg file.

The **ansible.cfg** file is used by Ansible to set certain default values. These values can be modified.

Using the default path displayed in the **ansible --version** command, display the default configuration file. Notice this is a very long file. You can pipe the output of the **cat** command to **more** so that it displays one page at a time. Highlighted are the entries that will be in your **ansible.cfg** file for this lab.

```
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>
```

Notice that Ansible shows that the inventory **hosts** file it will use by default is **/etc/ansible/hosts**. In a previous step, you edited the inventory **hosts** file in the **ansible-csr1000v directory**. In the next step you will edit a new **ansible.cfg file** which uses the **hosts** inventory file that you created.

## Step 5: Change the location of the ansible.cfg file.

a. Ansible will use the config file located in **/etc/ansible/ansible.cfg** unless there is an **ansible.cfg** file in the current directory. Change back to the ansible-csr1000v directory. There is already a placeholder **ansible.cfg** file in this directory. Display the current location of **ansible.cfg** with the **ansible --version** command.

```
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. Display the file to see that it is empty, except for a comment. You will edit this file in the next step.

```
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$
```

### Step 6: Edit the ansible.cfg file.

Now, you need to edit your **/ansible-csr1000v/ansible.cfg** file to include the location of your **hosts** inventory file. Remember, the default config file in **/etc/ansible/ansible.cfg** uses the inventory file in /etc/ansible/hosts.

a. Open the **/ansible-csr1000v/ansible.cfg** file in VS Code.

b. You can remove the comment. Add the following lines to the file and save it.

```
# 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
```

Like Python, the # is used for comments within the **ansible.cfg** file. If the entry refers to a file name such as **inventory=./hosts** the comment cannot come after the entry. Ansible treats the # and the comment that follows as part of the filename. Therefore, in these cases, the # comment must be on a separate line. However, variables can have a comment on the same line as shown for **host_key_checking** and **retry_files_enabled**.

The **ansible.cfg** file tells Ansible where to find the inventory file and sets certain default parameters. The information you entered in your **ansible.cfg** file is:

- **`inventory=./hosts`** - Your inventory file is the **hosts** file in the current directory.
- **`host_key_checking = False`** - The local development environment does not have SSH keys set up. You have set the **host_key_checking** set to **False**, which is the default. In a production network, **host_key_checking** would be set to **True**.
- **`retry_files_enabled = False`** - When Ansible has problems running playbooks for a host, it will output the name of the host into a file in the current directory ending in **.retry**. To prevent clutter, it is common to disable this setting.
- **`deprecation_warnings=False`** - Deprecation warnings indicate usage of legacy features that are slated for removal in a future release of Ansible. You have disabled this warning.

### Step 7: SUMMARY: Your Ansible configuration files.

In this Part, you configured Ansible to run in the **ansible-csr1000v** directory. By default, Ansible uses files in the **/etc/ansible** directory. The default /etc/ansible/**ansible.cfg** file indicates that the default inventory file is **/etc/ansible/hosts**.

However, in this lab you need a hosts file and ansible.cfg file in your ansible-csr1000v directory.

- You edited the **hosts** file to contain login and IP address information for the CSR1000v router
- You edited the **ansible.cfg** file to use the local hosts file as the inventory file (**inventory=./hosts**).

In the next Part, you will create a playbook to tell Ansible what to do.

## Part 3: Use Ansible to Back up a Configuration

In this Part, you will create an Ansible playbook that will automate the process of backing up the configuration of the CSR1000v. Playbooks are at the center of Ansible. When you want Ansible to get information or perform an action on a device or group of devices, you run a playbook to get the job done.

An Ansible playbook is a YAML file containing one or more plays. Each play is a collection of tasks.

- A **play** is a matching set of tasks to a device or group of devices.
- A **task** is a single action that references a **module** to run along with any input arguments and actions. These tasks can be simple or complex depending on the need for permissions, the order to run the tasks, and so on.

A playbook may also contain **roles**. A role is a mechanism for breaking a playbook into multiple components or files, simplifying the playbook and making it easier to reuse. For example, the **common** role is used to store tasks that can be used across all of your playbooks. Roles are beyond the scope of this lab.

The Ansible YAML playbook includes **objects**, **lists** and **modules**.

- A YAML object is one or more key value pairs. Key value pairs are separated by a colon without the use of quotation marks, for example **hosts: CSR1kv**.
- An object can contain other objects such as a list. YAML uses lists or arrays. A hypen “**-**“ is used for each element in the list.
- Ansible ships with a number of modules (called the module library) that can be executed directly on remote hosts or through playbooks. An example is the **ios_command** module used to send commands to an IOS device and return the results. Each task typically consists of one or more Ansible modules.

You run an Ansible playbook using the **ansible-playbook** command, for example:

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

The **ansible-playbook** command uses parameters to specify:

- The playbook you want to run (**backup_cisco_router_playbook.yaml**)
- The inventory file and its location (**-i hosts**).

### Step 1: Create your Ansible playbook.

The Ansible playbook is a YAML file. Make sure you use the proper YAML indentation. Every space and dash is significant. You may lose some formatting if you copy and paste the code in this lab.

a. In VS Code, create a new file in the **ansible-csr1000v** directory with the following name: **backup_cisco_router_playbook.yaml**

b. Add the following information to the file.

```
---
- 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"
```

## Step 2: Examine your Ansible playbook.

The playbook you have created contains one play with two tasks. The following is an explanation of your playbook:

- `---` This is at the beginning of every YAML file, which indicates to YAML that this is a separate document. Each file may contain multiple documents separated by `---`
- **`name: AUTOMATIC BACKUP OF RUNNING-CONFIG`** - This is the name of the play.
- **`hosts: CSR1kv`** - This is the alias of the previously configured **hosts** file. By referring to this alias in your playbook, the playbook can use all the parameters associated with this inventory file entry which includes the username, password, and IP address of the device.
- **`gather_facts: false`** - Ansible was originally designed to work with Linux servers, copying Python modules to the servers for task automation. This is not necessary when working with networking devices.
- **`connection: local`** - Specifies that you are not using SSH, therefore the connection is local.
- **`tasks:`** - This keyword indicates one or more tasks to be performed.

The first task is to display the running-config.

- `- name: DISPLAYING THE RUNNING-CONFIG` - Name of the task.
- **`ios_command:`** - This is an Ansible **module** that is used to send commands to an IOS device and return the results read from the device. However, it does not support configuration commands. The **ios_config** module is used for this purpose, as you will see in the next Part of this lab.

  **Note**: In the Linux terminal, you can use the **ansible-doc** *module_name* command to view the manual pages for any **module** and the parameters associated with that module. (e.g. **ansible-doc ios_command**)
- **`commands:`** - This parameter is associated with the **ios_command** module. It is used to list IOS commands in the playbook that are to be sent to the remote IOS device. The resulting output from the command is returned.
- **`- show running-config`** - This is the Cisco IOS command sent using the **ios_command** module.
- **`register: config`** - Ansible includes registers used to capture output of a task to a variable. This entry specifies that the output from the previous **show running-config** command will be stored in the variable **config**.

The second task is to save the output:

- **`- name: SAVE OUTPUT TO ./backups/`** - Name of the task
- **`copy:`** - This is an Ansible **module** used to copy files to a remote location. There are two parameters associated with this module:
  - o **`content:`** `"{{ config.stdout[0] }}"` - The specified value for this parameter is the data stored in the **config** variable, the Ansible register variable used in the previous task. Standard

output (**stdout**) is the default file descriptor where a process can write output used in Unix-like operating systems, such as Linux and Mac OS X.

- `dest: "backups/show_run_{{ inventory_hostname }}.txt"` **-** This is the path and file name to where the file should be copied. The **inventory_hostname** variable is an Ansible "magic variable" that automatically receives the hostname as configured in the **hosts** file. In your case, recall that this is **CSR1kv**. This parameter results in a file **show_run_CSR1kv.txt** stored in the **backups** directory. The file will contain the output of the **show running-config** command. You will create the **backups** directory in the next step.

## Step 3: Run the Ansible backup Playbook.

a. In Part 1, you started the CSR1000v VM. Ping it to verify you can access it. Enter **Ctrl+ C** to abort the 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. Create the **backups** directory. As indicated in the last line of your playbook, this is the directory where the backup configuration file will be stored.

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

c. Now you can run the Ansible playbook using the **ansible-playbook** command:

```
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/] **************************************************
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$
```

**Note**: In many examples you will see the playbook run using the **-i** *inventory-filename* option. For example:

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

This option tells Ansible the location and name of the inventory file, the list of devices the playbook will use. This option is not necessary because you configured the inventory file name and location in your local **ansible.cfg** file: inventory=./hosts. You can use the **-i** *inventory-filename* option to override the information in the **ansible.cfg** file.

The **PLAY RECAP** should display **ok=2 changed=1** indicating a successful playbook execution.

If your Ansible playbook fails, some of the things to check in your playbook are:

- Make sure your **hosts** and **ansible.cfg** files are correct.
- Make sure the YAML indentation is correct.
- Make sure your IOS command is correct.
- Check all the Ansible playbook syntax.
- Verify you can ping the CSR1000v.

If you continue to have problems:

- Try typing one line at a time and running the playbook each time.
- Compare your file with the solutions playbook in the **ansible_solutions** directory.

### Step 4: Verify the backup file has been created.

In VS Code, open the **backups** folder and open the **show_run_CSR1kv.txt** file. You can also use the terminal window to cat the file with **cat backups/show_run_CSR1kv.txt**. You now have a backup of the CSR1000v configuration.

```
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>
```

## Part 4: Use Ansible to Configure a Device

In this Part, you will create another Ansible playbook to configure IPv6 addressing on the CSR1000v router.

### Step 1: View your hosts inventory file.

a. Re-examine your **hosts** inventory file. As a reminder, this file contains the alias **CSR1kv** and three inventory variables for the username, password, and host IP address. The playbook for this Part will also use this file and the **ansible.cfg** file you created early in the lab.

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

### Step 2: Create a new playbook.

a. In VS Code, create a new file in the **ansible-csr1000v** directory with the following name: **cisco_router_ipv6_config_playbook.yaml**

b. Add the following information to the file. Make sure you use the proper YAML indentation. Every space and dash is significant. You may lose some formatting if you copy and paste.

```
---
- 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"
```

### Step 3: Examine your Ansible playbook.

Much of this playbook is similar to the playbook you created in the previous Part. The main difference is the first task SET IPv6 ADDRESS.

The following is a brief description of the items in the task:

- **`ios_config:`** - This is an Ansible module used to configure an IOS device. You can use the **ansible-doc ios_config** command to see the details for the **parents** and **lines** parameters used in this playbook.
- **`parents: "interface GigabitEthernet1"`** - This parameter indicates the IOS interface configuration mode.
- **`lines:`** - An ordered set of IOS commands are configured in this section, specifying the IPv6 addressing information for the GigabitEthernet1 interface.

The rest of the playbook is similar to the tasks in the previous Part. The second task uses the **ios_command** module and the command **show ipv6 interface brief** to display output and send it to the register **output**.

The last task saves the information in the register **output** to a file **IPv6_output_CSR1kv.txt** in the **ios_configurations** subdirectory.

## Step 4: Run the Ansible playbook to configure IPv6 addressing on the CSR1000v VM.

a. In Part 1, you started the CSR1000v VM. Ping it to verify you can access it. Enter **Ctrl+ C** to abort the 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. Create the directory **ios_configurations**. As indicated in the last line of your playbook, this is the directory where the output for the **show ipv6 interface brief** command will be stored.

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

c. Now you can run the Ansible playbook using the **ansible-playbook** command. The **-v** verbose option can be used to display the tasks being performed in the playbook.

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

The first time you run the playbook, the **PLAY RECAP** should display **ok=3 changed=2** and **failed=0** indicating a successful execution. These values may be different if you run the playbook again.

### Step 5: Verify the file of the output has been created.

In VS Code, open the **ios_configurations** folder and click the **IPv6_output_CSR1kv.txt** file. You can also use the terminal window to view the file with **cat ios_configurations/IPv6_output_CSR1kv.txt**. You now have a backup of the CSR1000v configuration.

```
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$
```