



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE

Applied Computing

Wales Institute of Science and Art (WISA)

Network Programmability (ACCB5029)

Lab 9: Automate backup and device configuration using Ansible.

ICONS Used in this document:



Do research and complete the task. Need referencing/ in-text citations. Shall be included as a part of your coursework.



Record your Reflection.

 **Networking**
CISCO™ **Academy**

Task 1:**Objectives**

1. Configure Ansible.
2. Backup device configuration using Ansible.
3. Configure a device using Ansible.

Task 2:**Objectives**

1. Research other available automation tools to setup and configure Infrastructure and applications.
2. As a Team, compare the differences between one tool (based on your selection) and Ansible. Your comparison must address its pros and cons, availability, scalability, effectiveness etc (not limited to)
3. Design and propose an application and automate using the selected tool. (or) use the existing device, configure the device, and enable backup using the selected tool.
4. Record your individual reflection.

Required Resources

Host computer with DEVASC VM.

CSR1000v VM

Note: You must have completed ALL the previous labs before starting this Lab.

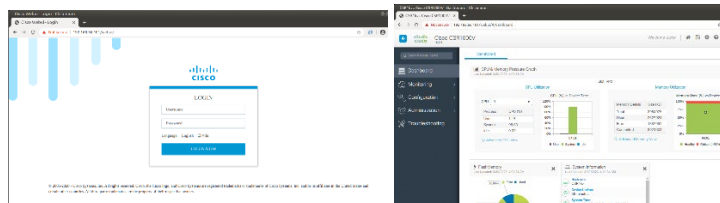
Instructions

1. Launch the DEVASC VM
2. Install CSR1000v VM.
3. Verify Connectivity
 - 3.1. From the CSR1000v VM, use the show ip interface brief command to verify the IPv4 address of the GigabitEthernet1 interface. The CSR1000v is most likely using the IPv4 address from the 192.168.x.y address space. Refer to your Advanced Networking Module: Which is the optimal command to verify the IP address in the interface?

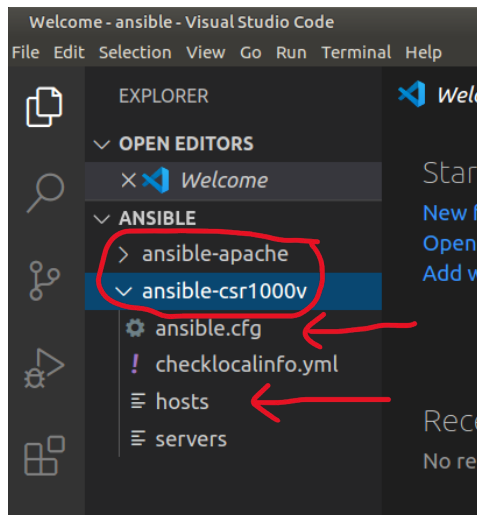
```
*Mar 30 04:38:30.544: %SYS-6-LOGOUT: User cisco has exited
.56.102)
**
**
*** Cisco Networking Academy ***
*** This software is provided for ***
*** Educational Purposes ***
*** Only in Networking Academies ***
**
**
CSR1k0>en
CSR1k0#sho ip int br
Interface IP-Address OK? Method Status
GigabitEthernet1 192.168.56.101 YES DHCP up
```

IP Address and Interface must be showing up. Else, Troubleshoot

- 3.2. On DEVASC VM, Ping the CSR1000v VM from the DEVASC VM.
devasc@labvm:~\$ ping 192.168.56.101
Ping must be successful. Else, Troubleshoot
- 3.3. On DEVASC VM, Connect to the CSR1000v with SSH from the DEVASC VM. Use the password cisco123! to authenticate. Notice that you are automatically in privileged EXEC mode. Enter exit to end the SSH session.
Password:
Connection must be successful. Else, Troubleshoot
- 3.4. From the DEVASC VM, access the CSR1000v WebUI (Web User Interface). Open a web browser on the DEVASC VM.
For the URL enter: <https://192.168.56.101>
Note: Be sure to use "https"
If your browser displays a warning similar to, "Your connection is not private": Click Advanced. Click Proceed to 192.168.56.101 (unsafe).
You will now see a LOGIN screen. Enter the Credentials and login:
If URL is not opening, Troubleshoot.



4. Configure Ansible
 - 4.1. Open the Ansible directory in VS Code.
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.



Observation:

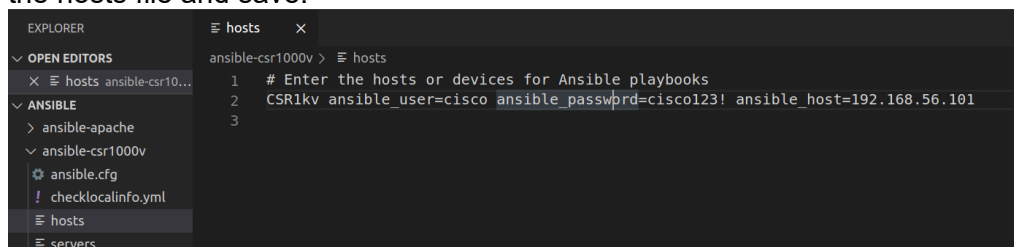
Ansible uses an inventory file called `hosts` that contains device information used by Ansible playbooks. The default location for the Ansible inventory file is `/etc/ansible/hosts` as specified in the default `ansible.cfg` in the same `/etc/ansible` directory.

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.

- 4.2. Open the `hosts` file in the `ansible-csr1000v` directory. Add the following lines to the `hosts` file and save.



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.

Observation:

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.

- 4.3. 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$ an
anacron          ansible-console  ansible-pull
animate          ansible-doc      ansible-test
ansible          ansible-galaxy   ansible-vault
ansible-config   ansible-inventory
ansible-connection ansible-playbook
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
```

- 4.3.1. Use the ansible --version command to display version information. Notice that this lab is using version 2.9.9. Ansible includes certain default files, including a default configuration file, ansible.cfg.

```
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$ 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
#
# uncomment this to disable SSH key host checking
#host_key_checking = False

# 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
#retry_files_save_path = ~/.ansible-retry
```

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.

- 4.4. 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.

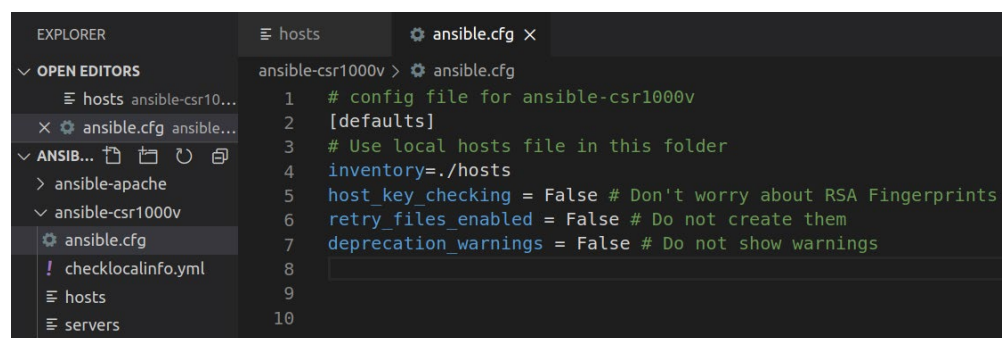
- 4.4.1. Display the current location of `ansible.cfg` with the `ansible --version` command.
- 4.4.2. Display the file to see that it is empty, except for a comment. You will edit this file in the next step.

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

- 4.4.3. Edit the `ansible.cfg` file. Open the `/ansible-csr1000v/ansible.cfg` file in VS Code.

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`.

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



Observation:

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.

- 4.5. **SUMMARY:** In this step, 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. So,

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

5. 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.

Observation: 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 hyphen “-” 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.

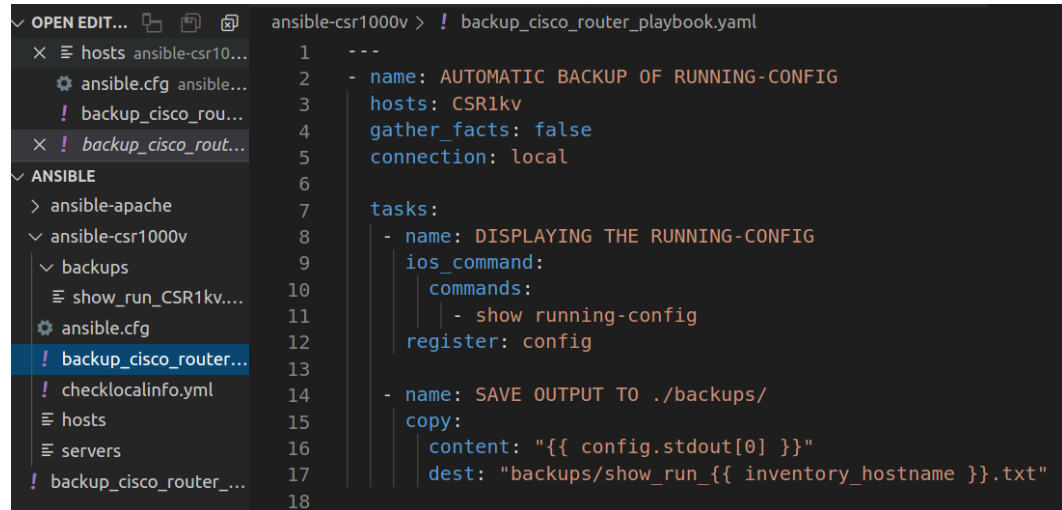
You run an Ansible playbook using the `ansible-playbook` command, for example:
`ansible-playbook backup_cisco_router_playbook.yaml -i hosts`

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).

- 5.1. Create your Ansible playbook: The Ansible playbook is a YAML file. In VS Code, create a new file in the ansible-csr1000v directory with the following name: backup_cisco_router_playbook.yaml and add the following information to the file.



```

1  ---
2  - name: AUTOMATIC BACKUP OF RUNNING-CONFIG
3    hosts: CSR1kv
4    gather_facts: false
5    connection: local
6
7  tasks:
8    - name: DISPLAYING THE RUNNING-CONFIG
9      ios_command:
10       commands:
11         - show running-config
12       register: config
13
14    - name: SAVE OUTPUT TO ./backups/
15      copy:
16       content: "{{ config.stdout[0] }}"
17       dest: "backups/show_run_{{ inventory_hostname }}.txt"
18

```

- 5.2. 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:

`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.

5.3. Run the Ansible backup Playbook.

5.3.1. Test the connectivity with your CSR Router using appropriate command.

5.3.2. 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.

5.3.3. Run the Ansible playbook using the `ansible-playbook` command:

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

```

devasc@labvn:~$ cd labs/devnet-src/ansible/ansible-csr1000v/
devasc@labvn:~/labs/devnet-src/ansible/ansible-csr1000v$ mkdir backups
devasc@labvn:~/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@labvn:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook backup_cisco_router_playbook.yaml -i hosts

PLAY [AUTOMATIC BACKUP OF RUNNING-CONFIG] *****
TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [CSR1kv]

TASK [SAVE OUTPUT TO ./backups/] *****
ok: [CSR1kv]

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

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

```

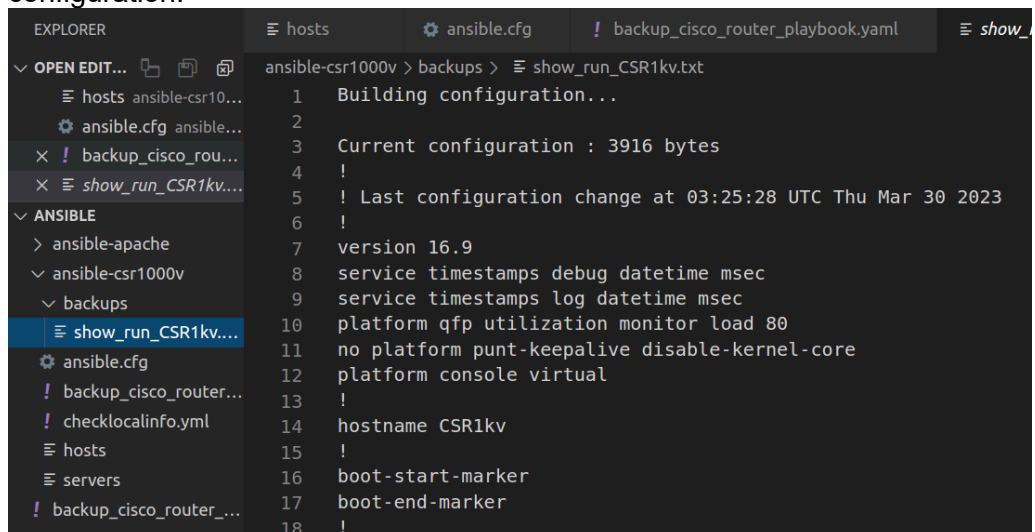
This option tells Ansible the location and name of the inventory file, the list of devices the playbook will use. This option is not necessary because you configured the inventory file name and location in your local `ansible.cfg` file: `inventory=./hosts`. You can use the `-i inventory-filename` option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

5.4. 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.



```

EXPLORER
  hosts
  ansible.cfg
  backup_cisco_router_playbook.yaml
  show_run_CSR1kv.txt
  ansible-csr1000v > backups > show_run_CSR1kv.txt
    1 Building configuration...
    2
    3 Current configuration : 3916 bytes
    4 !
    5 ! Last configuration change at 03:25:28 UTC Thu Mar 30 2023
    6 !
    7 version 16.9
    8 service timestamps debug datetime msec
    9 service timestamps log datetime msec
   10 platform qfp utilization monitor load 80
   11 no platform punt-keepalive disable-kernel-core
   12 platform console virtual
   13 !
   14 hostname CSR1kv
   15 !
   16 boot-start-marker
   17 boot-end-marker
   18 !

```

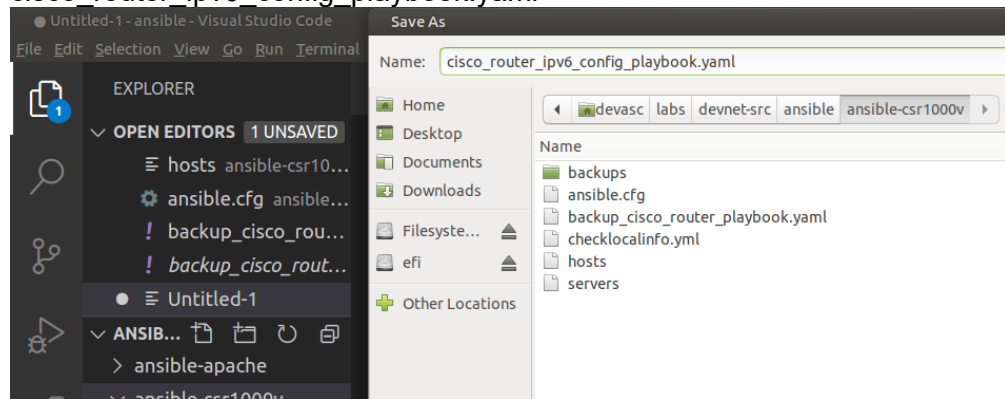
6. Use Ansible to Configure a Device

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

- 6.1. 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/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$
```

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



- 6.3. Add the following information to the file.

```
ansible-csr1000v > ! cisco_router_ipv6_config_playbook.yaml
1  ---
2  - name: CONFIGURE IPv6 ADDRESSING
3    hosts: CSR1kv
4    gather_facts: false
5    connection: local
6
7    tasks:
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
16      - name: SHOW IPv6 INTERFACE BRIEF
17        ios_command:
18         commands:
19           - show ipv6 interface brief
20         register: output
21
22      - name: SAVE OUTPUT ./ios_configurations/
23        copy:
24         content: "{{ output.stdout[0] }}"
25         dest: "ios_configurations/IPv6_output_{{ inventory_hostname }}.txt"
```

- 6.4. Examine your Ansible playbook: Much of this playbook is like the playbook you created in the previous part.

Observation:

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 like 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.

- 6.5. Run the Ansible playbook to configure IPv6 addressing on the CSR1000v VM. Before running the playbook, check your connectivity.

6.5.1. 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.

6.5.2. 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-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=255 time=1.34 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=255 time=1.50 ms
^C
192.168.56.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.342/1.422/1.583/0.108 ms
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ mkdir ios_configurations
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook -v cisco_router_ipv6_config_playbook.yaml
PLAY [CONFIGURE IPv6 ADDRESSING] *****
TASK [SET IPv6 ADDRESS] *****
changed: [CSR1kv] => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": true, "command": ["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]", "FE80::1::1", "2001:DB8:ACAD::1::1"}, {"GigabitEthernet1", "[up/up]", "FE80::1::1", "2001:DB8:ACAD::1::1"}]}
TASK [SAVE OUTPUT ./ios_configurations/] *****
changed: [CSR1kv] => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": true, "checksum": "60784fbaa4d0257d4f121c450effe5290553c", "dest": "ios_configurations/IPv6_output_CSR1kv.txt", "gid": 900, "group": "devasc", "md5sum": "56d79f15ee0776cf131c34bfc1880", "mode": "0664", "owner": "devasc", "size": 67, "src": "/home/devasc/ansible/tmp/ansible-tmp-160815110.9628327-4694-48567590297706/source", "state": "file", "uid": 900}
PLAY RECAP *****
          ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ cat ios_configurations/IPv6_output_CSR1kv.txt
GigabitEthernet1
FE80::1::1
2001:DB8:ACAD:1::1
CSR1kv#

```

Observation: 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.

- 6.6. 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.

```

CSR1kv>en
CSR1kv#sho ip v
CSR1kv#sho ipv6 int br
GigabitEthernet1      [up/up]
FE80::1::1
2001:DB8:ACAD:1::1
CSR1kv#

```

Task 2:**Objectives**

1. Research other available automation tools to setup and configure Infrastructure and applications.
2. As a Team, compare the differences between one tool (based on your selection) and Ansible. Your comparison must address its pros and cons, availability, scalability, effectiveness etc (not limited to)
3. Design and propose an application and automate using the selected tool. (or) use the existing device, configure the device, and enable backup using the selected tool.
4. Record your individual reflection.

End of document.