**Antigua, Jose Ricardo O.**

**CPE41S1**

**January 17, 2022**

**Engr. Alonica Villaneuva**

**Final Case Study | Network Automation and Programmability**

**Objectives**:

Part 1: Launch the DEVASC VM and CSR1kv VM

Part 2: Configure hosts, configuration, yaml

Part 3: Use ansible for network automation

Part 4: Use the pyATS in Ansible

**Background / Scenario**

In this lab, you will use ansible-playbooks that can automate the process. Ansible is an open-source IT automation platform that automates manual IT activities including provisioning, configuration management, application deployment, orchestration, and more. pyATS is an end-to-end testing ecosystem that focuses on data-driven and reusable testing and is built to support Agile, quick development iterations.

Required Resources

* 1 PC with operating system of your choice that can handle multiple Virtual Box or VMWare running simultaneously.
* DEVASC Virtual Machine
* CSR1kv Virtual Machine

**Instructions**

**Part 1: Launch the DEVASC VM**

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.

**Step 1: Launch the VMs**

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

**Step 2: Verify connectivity between the VMs.**

a.      In the CSR1kv VM, Ensure that the VM is loaded, press Enter to get a command prompt and then use **show ip interface brief** to verify that the IPv4 address is 192.168.56.106. If the address is different, make a note of it.

b.      Open a terminal in VS Code in the DEVASC VM.

c.      Ping the CSR1kv to verify connectivity. You should have already done this previously in the installation labs. If you are not able to ping, then revisit those labs listed above in Part 1a.

devasc@labvm:~$ **ping 192.168.56.104**

PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.

64 bytes from 192.168.56.104: icmp\_seq=2 ttl=254 time=1.74 ms

64 bytes from 192.168.56.104: icmp\_seq=3 ttl=254 time=1.70 ms

64 bytes from 192.168.56.104: icmp\_seq=4 ttl=254 time=1.98 ms

64 bytes from 192.168.56.104: icmp\_seq=5 ttl=254 time=1.09 ms

64 bytes from 192.168.56.104: icmp\_seq=6 ttl=254 time=1.74 ms

64 bytes from 192.168.56.104: icmp\_seq=7 ttl=254 time=1.80 ms

64 bytes from 192.168.56.104: icmp\_seq=8 ttl=254 time=1.71 ms

^C

--- 192.168.56.104 ping statistics ---

8 packets transmitted, 7 received, 12.5% packet loss, time 7023ms

rtt min/avg/max/mdev = 1.090/1.679/1.984/0.257 ms

**Part 2: Configure Ansible**

Step 1: Open a terminal in the DEVASC-LABVM

Step 2: Enable the SSH server

devasc@labvm:~$ **sudo systemctl start ssh**

**Step 3: 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 4: Edit the Ansible inventory file**

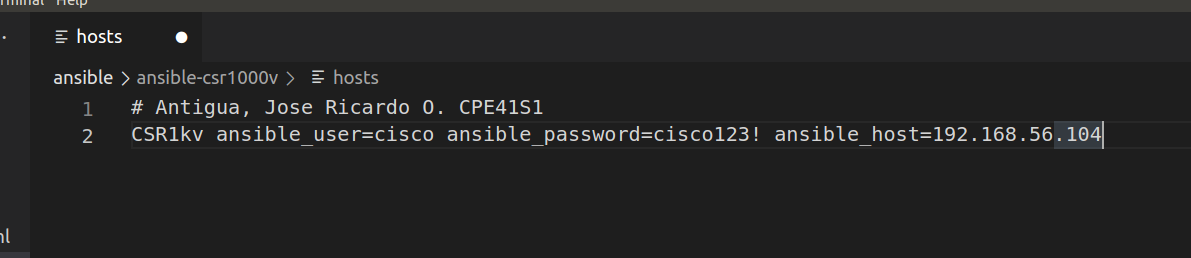
a. Open the **hosts** file in the ansible-csr1000v directory.

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

**Enter the hosts or devices for Ansible playbooks**

# Enter the hosts or devices for Ansible playbooks

CSR1kv ansible\_user=**cisco** ansible\_password=**cisco123**! ansible\_host=192.168.56.104



The hosts file begins with an alias, CSR1kv, after the remark (#). From within the program, an alias is utilized. To refer to a device, use an Ansible playbook. The hosts file specifies three variables that will be used after the alias. The Ansible playbook uses this to gain access to the device. Ansible requires the following SSH credentials. Access the CSR1000v VM in a secure manner.

Note: Make sure you change the host IP address to the IP address you took note of earlier.

**Step 5: 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.

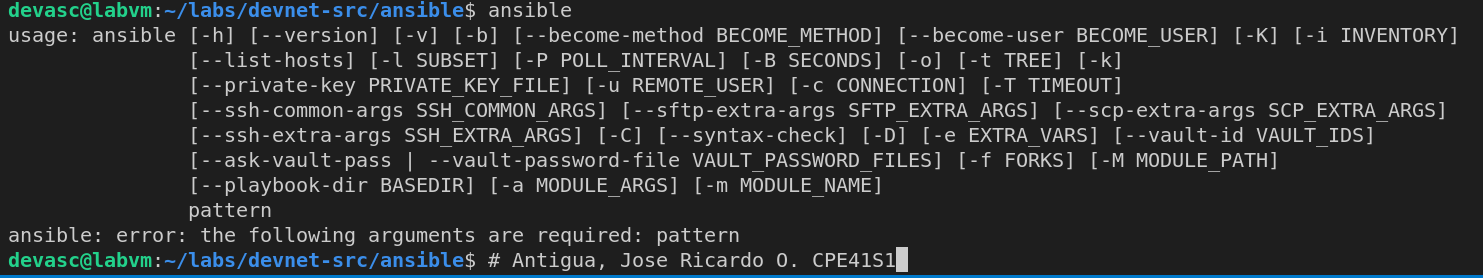
devasc@labvm:~/labs/devnet-src/ansible$ ansible

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

Output omitted:



B. Use the **ansible --version** command to display the version.

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

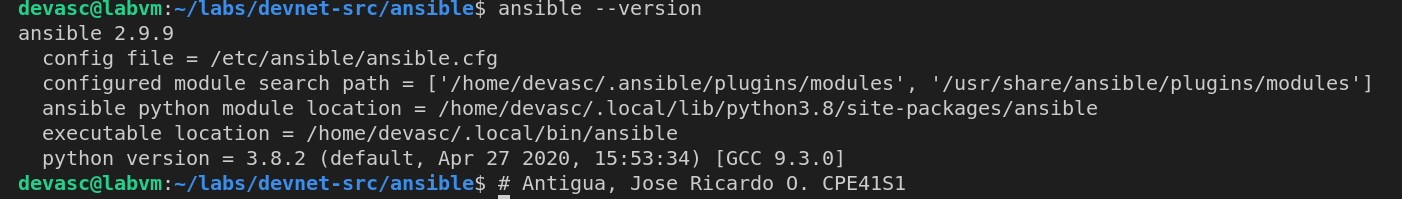
ansible 2.9.9

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



Notice that we are using the latest version of ansible which is ansible 2.9.9 during this laboratory. If you have a different version when performing this activity some minor changes might occur when proceeding. Read the changelogs in the ansible documentation if you see a problem when proceeding with the laboratory activity.

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

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]

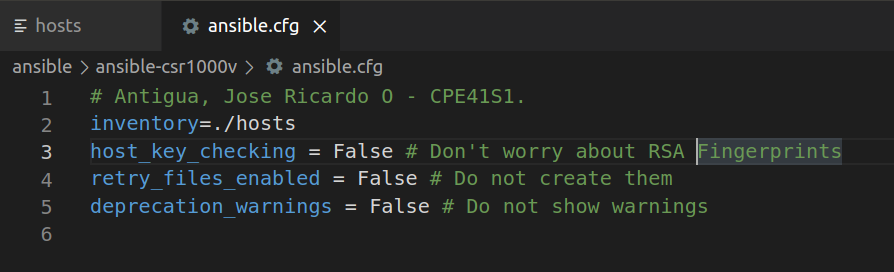
# Use local hosts file in this folder

**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



The ansible.cfg file tells Ansible where to find the inventory file and sets certain default parameters.

* Inventory - This points to the path of the inventory file which contains hostname or IP address details of managed nodes.
* host\_key\_checking - A host key is assigned for individual managed nodes
* retry\_files\_enabled - This is a parameter to enable the retry feature, allowing Ansible to create a .retry file whenever a playbook fails.
* deprecation\_warnings - Use of legacy features that are deprecated is indicated by deprecation warnings. In a future Ansible release, they'll be deprecated.

**Part 3: Use Ansible for Network Automation**

In this Part, you will create an Ansible playbook that will automate the process. 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.

**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.

a. In VS Code, create a new file in the ansible-csr1000v directory with the following name: case\_study\_playbook.yaml

b. Add the following information to the file.

---

- name: CASE STUDY

hosts: R1

gather\_facts: false

connection: local

tasks:

- name: Updating Port int gigabitethernet1

ios\_config:

lines:

- description - Configure by Ansible

parents: interface GigabitEthernet1

# OSPF

- name: Setting up Default OSPF

ios\_config:

lines:

- router-id 192.168.56.104

parents: router ospf 100

- name: Configuring NEW OSPF network

ios\_config:

lines:

- router-id 1.1.1.1

- network 192.168.56.0 0.0.0.255 area 0

parents: router ospf 1

- name: Ensuring SNMP strings are present

ios\_config:

commands:

- snmp-server community ansible-public RO

- snmp-server community ansible-private RW

- 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"

- name: Test Ping R1

ios\_ping:

dest: 192.168.56.104

- name: Test Ping R2

ios\_ping:

dest: 192.168.56.105

#Change ACL configuration and compare before and after configuration using

- name: collect configuration (before)

ios\_command:

commands:

- show run

register: result\_before

- name: Creating the ACL rules

ios\_config:

lines:

- access-list 110 deny icmp any any

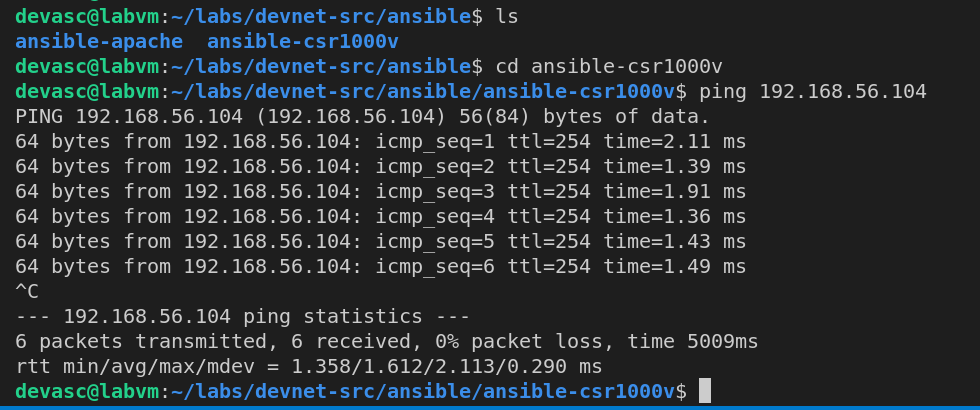
- access-list 110 permit ip any any

before: no access-list 110

match: exact

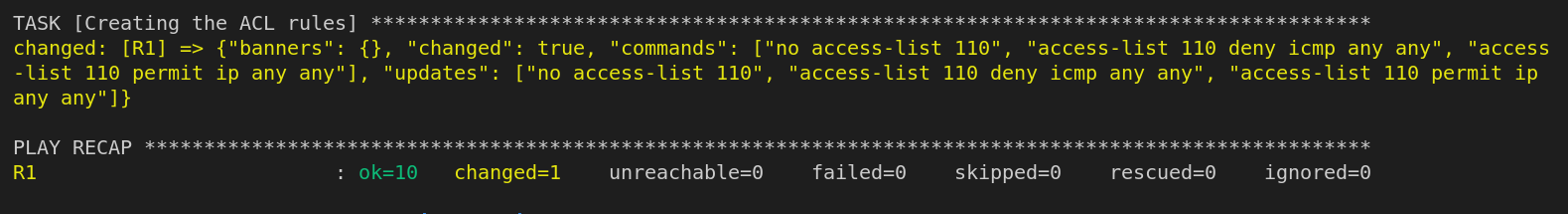
**Step 2: Run the Ansible playbook**

a. In Part 1, you started the CSR1000v VM. Ping it again to verify if your access it is still available. Enter Ctrl+ C to abort the ping.



b     Now you can run the Ansible-playbook command:

devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ **ansible-playbook case\_study\_playbook.yaml**



Check the results of running the playbook, when a failed value is returned, check the code again if some syntaxes are wrong. If other errors a still present such as misspelling, correct them and run the playbook again. Additionally, the playbook also includes a test where it pings the CSR VM hence, requiring the target VM to be open before running the playbook.

**Part 3: Testing Pyats in Ansible**

**Step 1:** In the existing code copy and paste the following to test the pyats.

 #Testing Pyats in Ansible

    - name: Testing pyats collect config (after)

      ios\_command:

        commands:

          - show run

      register: result\_after

    - name: Changing Ipv6 Address for Pyats 1 /64

      ios\_config:

        lines:

          - int gig 1

          - ipv6 address 2001:db8:acad:56::101/64

    - name: Changing Ipv6 Address for Pyats Link-Local

      ios\_config:

        lines:

          - int gig 1

          - ipv6 address fe80::56:1 link-local

  vars:

    exclude\_list:

      - (^Using.\*)

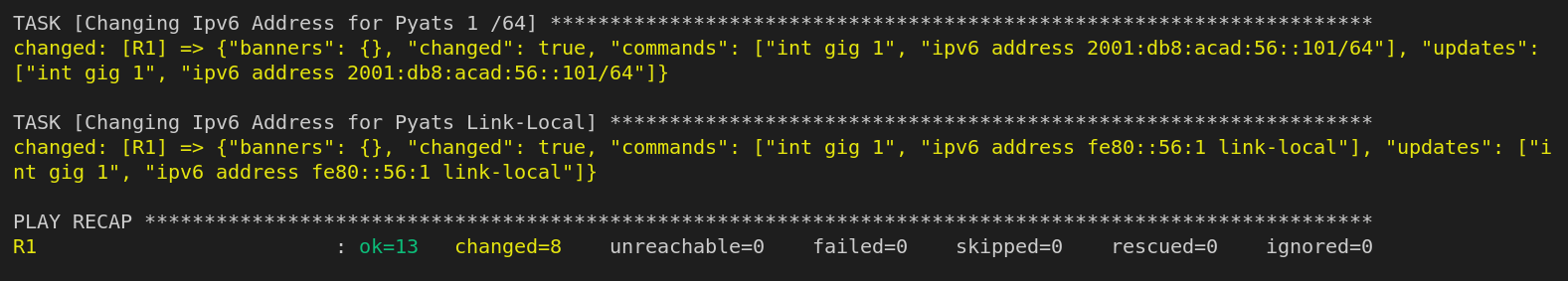
      - (Building.\*)

      - (Current.\*)

      - (crypto pki certificate chain.\*)

The task added is to test the pyATS in the ansible:

* command: the command to run on the device
* Ios\_config -module is used for this purpose. For dividing configuration into sections, utilize a basic block indent file syntax. This module provides a deterministic implementation for working with IOS configuration sections.
* lines - includes the configuration of the device.
* exclude\_list -The argument exclude means command lists which are excluded when comparing before and after configs. In the playbook example above, variable excluded\_list, which is defined as the play variable, is used.



Upon running the playbook with the added code for the first time, notice that the changed value also received new values, this presents that the playbook is utilizing new code sets in its playthrough.

**Step 2: Test Connectivity**

1. Test the connectivity and see if the connection of the network is working.

devasc@labvm: ~/labs/devnet-src/ansible/ansible-csr1000v$ **ansible-playbook case\_study\_playbook.yaml**

After running the command, the output should be similar below:

PLAY [CASE STUDY] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Updating Port int gigabitethernet1] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Setting up Default OSPF] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Configuring NEW OSPF network] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Ensuring SNMP strings are present] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [DISPLAYING THE RUNNING-CONFIG] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [SAVE OUTPUT to ./backups/] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [R1]

TASK [Test Ping R1] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Test Ping R2] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [collect configuration (before)] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Creating the ACL rules] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [R1]

TASK [Testing pyats collect config (after)] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [R1]

TASK [Changing Ipv6 Address for Pyats 1 /64] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [R1]

TASK [Changing Ipv6 Address for Pyats Link-Local] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [R1]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

R1 : ok=13 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

**Conclusion:**

I learned a lot about myself upon conducting this case study wherein my understanding of the course is truly challenged in both abilities to comprehend errors and debugging to the very core of my capacity to formulate network setups and their configuration. Additionally, I learned that I still have a lot to learn and relearn if I would even try to pursue further this field in engineering as I struggled very heavily with the problems presented to me along the way in developing this study. Ansible is truly a great tool for network activities.

**Video Link Simulation:**

https://drive.google.com/file/d/1Pz3oaCwGEBBAPbuNpPtHhbQ\_btTi6na1/view?usp=sharing

**Github Link:**

https://github.com/JRAntigua/Case\_Study\_Antigua