Name: Dalmacio, Aaron Christopher V.	Course: CPE 028
Section: CPE41S1	Date: 01 / 17 / 2022

Final Case Study | Network Automation and Programmability

Topology

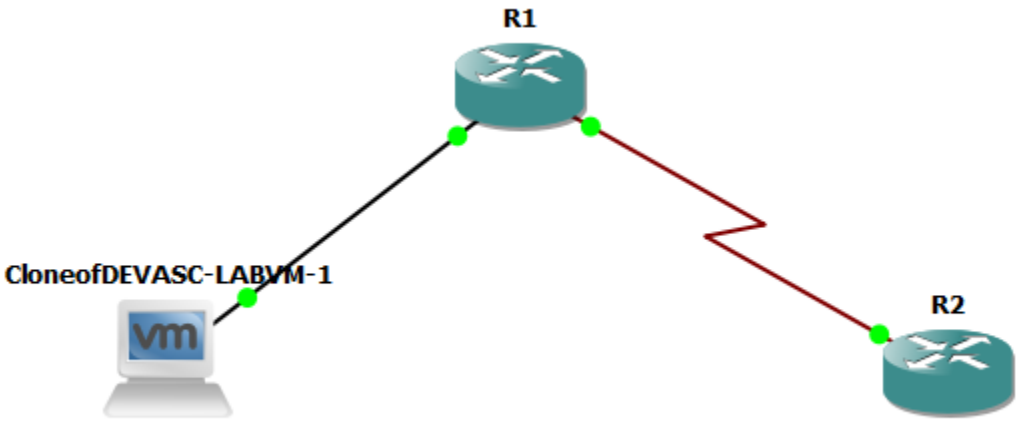


Figure 1. Network Topology

Addressing Table

Device	Interface	IP Address	Subnet Masks
R1	Serial 2/0	10.0.0.1	255.255.255.252
	FastEthernet0/0	192.168.10.1	255.255.255.0
R2	Serial 2/0	10.0.0.2	255.255.255.252
PC1	Ethernet0	192.168.10.2	255.255.255.0

Objectives

The objective of this activity is to design a laboratory activity that discusses the three network topics excluding basic configuration, IP address, and show commands regarding network automation or network programmability. Another thing is that we should use pyATS to test the network.

Required Resources

- 1 PC with operating system of your choice
- Virtual Box or VMWare
- DEVASC Virtual Machine
- GNS3

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.

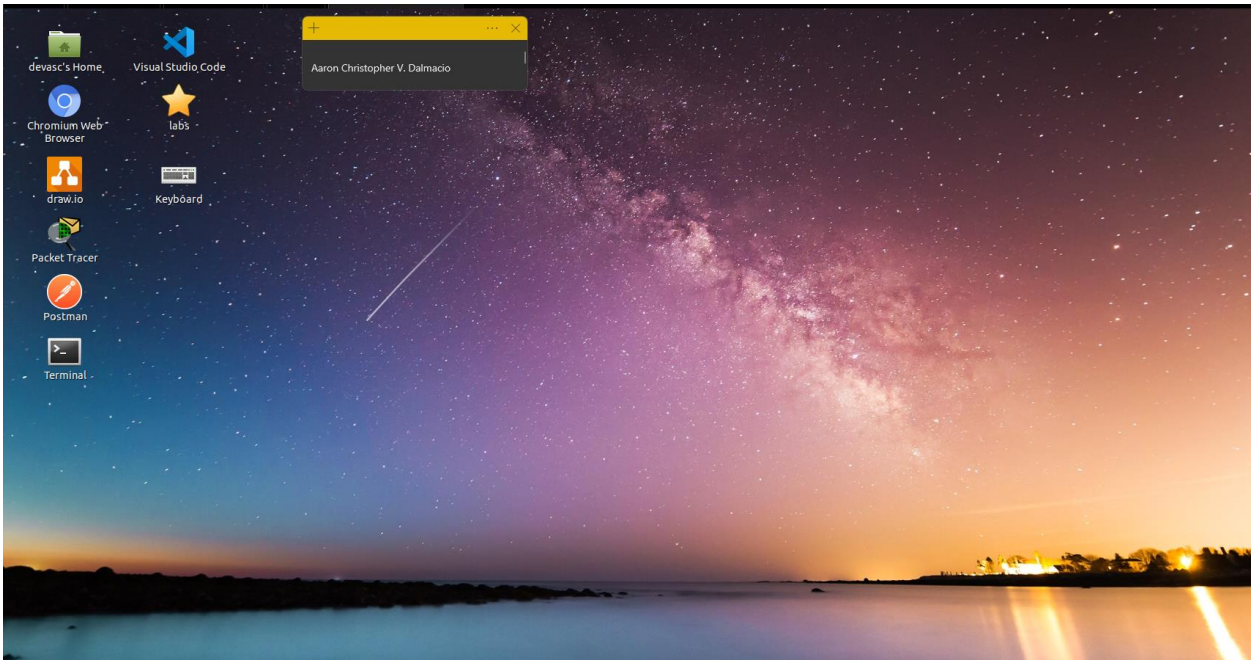


Figure 2. Open the DEVASC VM

Part 2: Launch the GNS3

If you do not have GNS3, then you must install it now. If you have already downloaded the GNS3, launch the GNS3 Software now

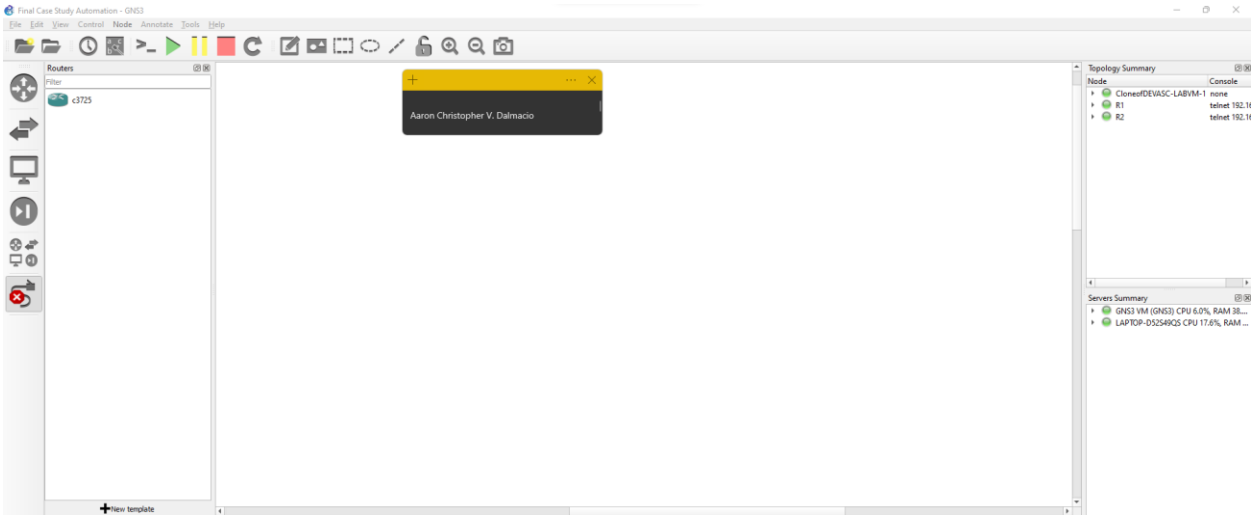


Figure 3. Open the GNS3

Step 1: Create a new Project

On the GNS3 Software, click on File that could be found on the upper left of the software then click on New blank project. A pop up window will come up and you could now add a new project. Enter the name of the project and find your desired location. After entering the name and location for the project, press OK then the project is now created.

Step 2: Install the CISCO IOS images needed for the switches and routers.

Search and download images that are needed for you to do your topology in GNS3. Having CISCO IOS images would help on building Cisco networks in GNS3.

Step 3: Create the topology

After downloading the specific routers and VMs that are needed, you can now create your topology. Connect all of the switches, routers, and computers using the wirings available.

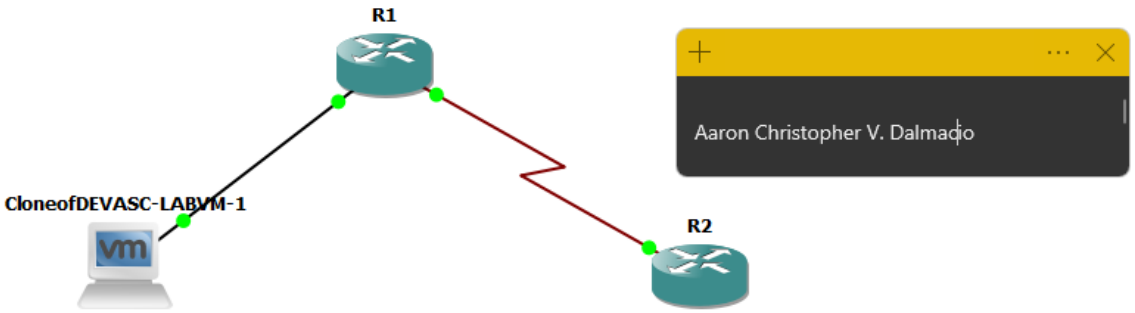


Figure 4. Creating the Topology on the GNS3

Step 4: Apply basic configurations on the routers and switches.

In the routers you should apply basic configurations such as its hostname, password, banner, and ip address of each interface.

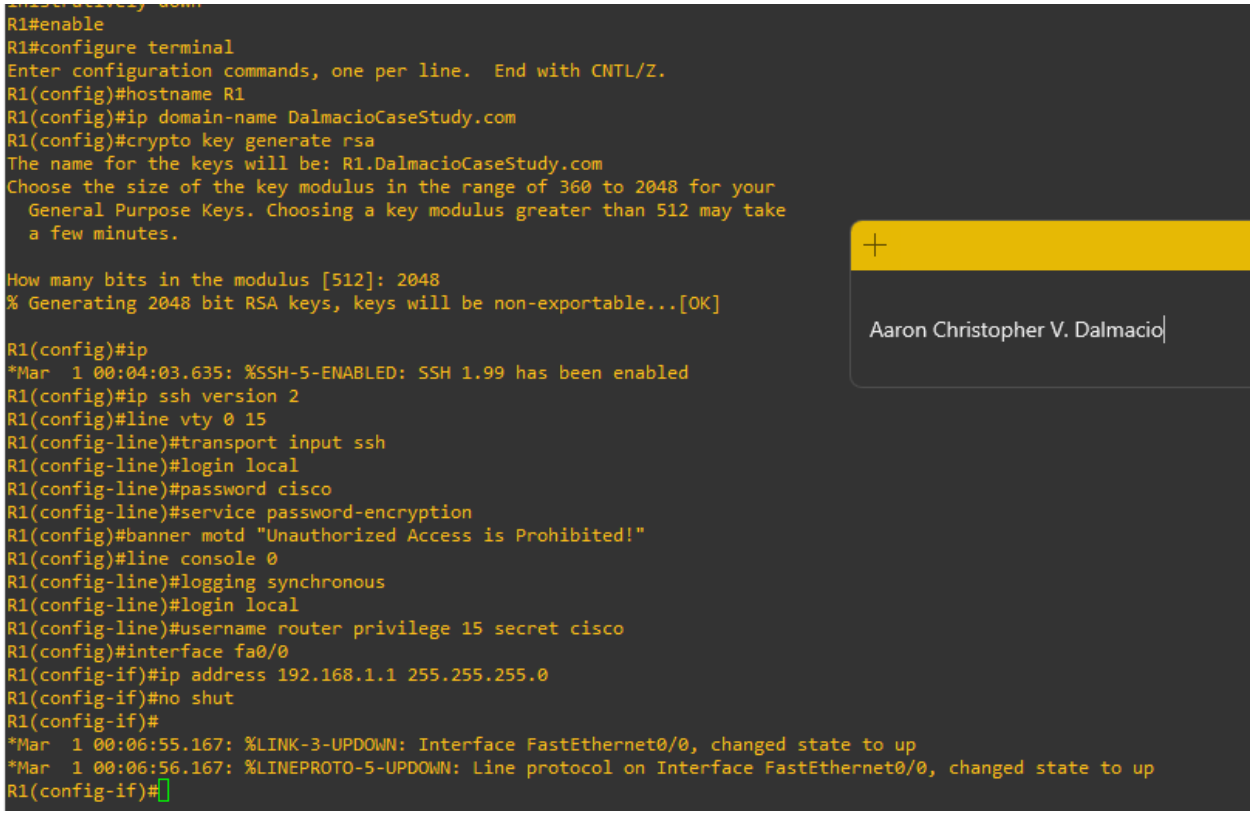


Figure 5. Applying Basic Configuration and Address on Router 1

Figure 6. Applying Basic Configuration and Address on Router 2

Step 6: Ping all of the Device and Access using SSH

In this step, you should be able to ping the routers and access it using SSH.

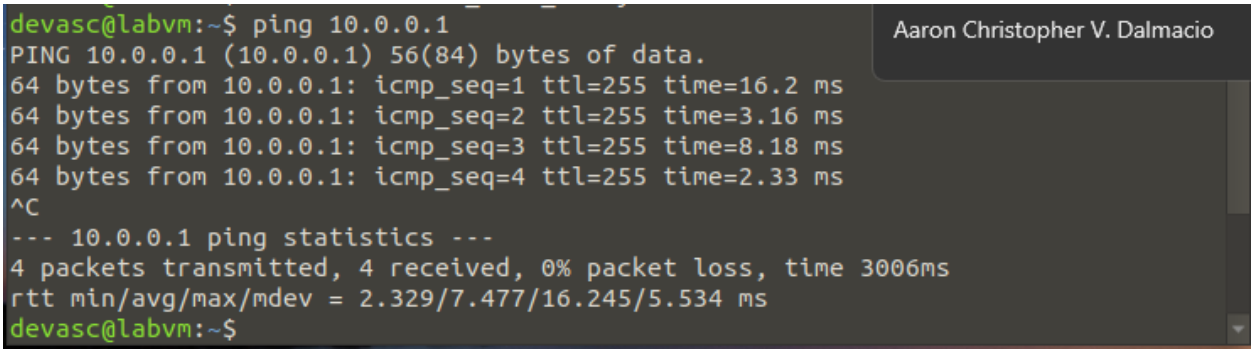


Figure 7. Ping 10.0.0.1 in the Terminal

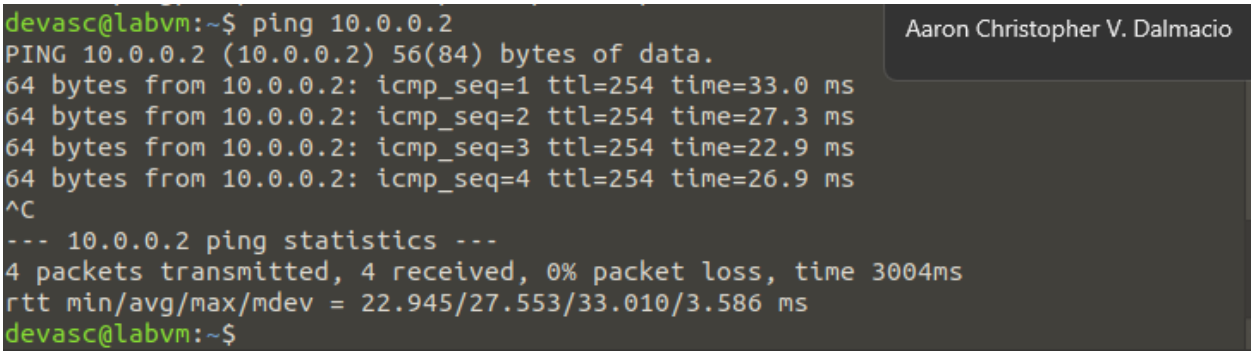


Figure 8. Ping 10.0.0.2 in the Terminal

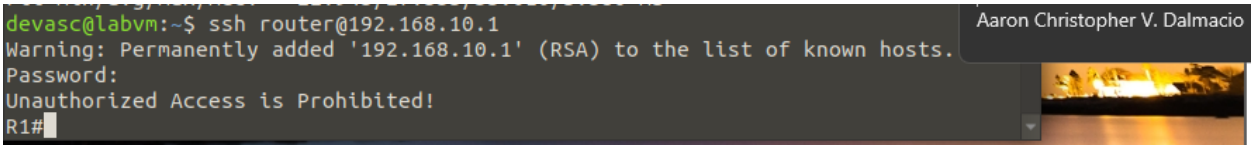


Figure 9. Ping 10.0.0.2 in the Terminal

Part 3: Using the Devasc Machine to Apply and Test Automation

Step 1: Create a new Directory

Step 1 shows the creation of your directory using the terminal of the DEVASC VM. Enter your preferred name for your directory.

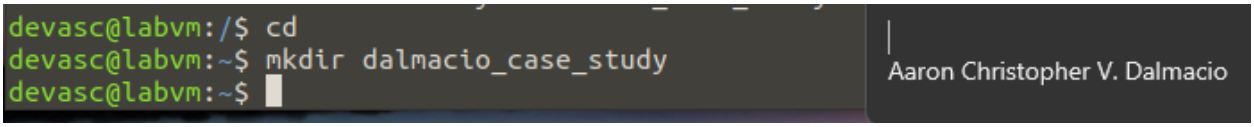


Figure 10. Creating a New Directory named dalmacio_case_study

Step 2: Create Hosts File

In step 2, you must create the hosts file.

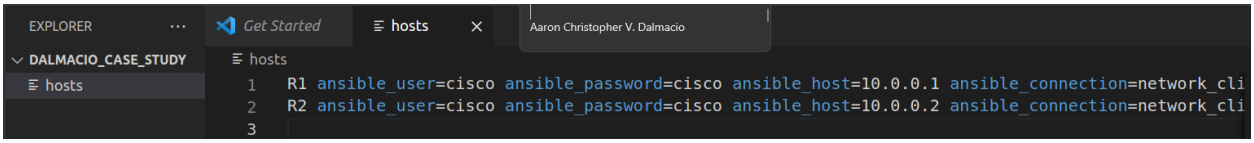


Figure 11. Creating hosts file

Enter the following codes on the hosts file:

R1 ansible_user=cisco ansible_password=cisco ansible_host=10.0.0.1 ansible_connection=network_cli ansible_network_os=ios ansible_become=yes ansible_become_method=enable ansible_become_pass=cisco

R2 ansible_user=cisco ansible_password=cisco ansible_host=10.0.0.2 ansible_connection=network_cli ansible_network_os=ios ansible_become=yes ansible_become_method=enable ansible_become_pass=cisco

Step 3: Create the Ansible configuration file.

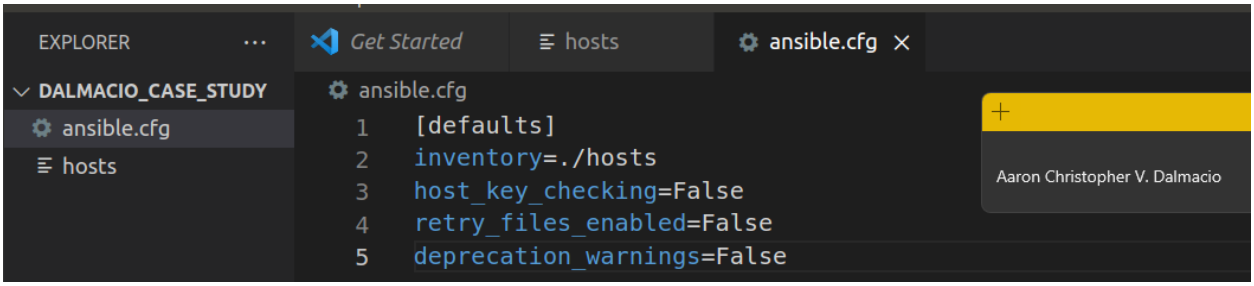


Figure 12. Creating the Ansible.cfg file

Enter the following codes on the ansible.cfg:

[defaults]

inventory=./hosts

host_key_checking=False

retry_files_enabled=False

deprecation_warnings=False

Step 4: Creating a backup yaml file named backup_config.yaml

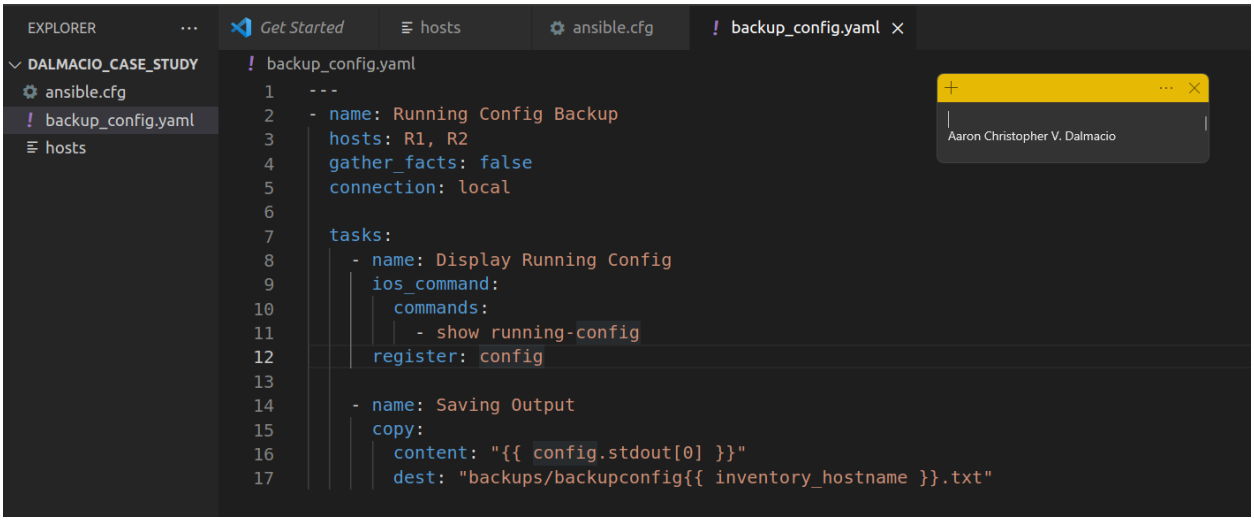


Figure 13. Creating the Backup_config.yaml

- name: Running Config Backup

hosts: R1, R2

```
gather_facts: false
connection: local

tasks:
  - name: Display Running Config
    ios_command:
      commands:
        - show running-config
    register: config

  - name: Saving Output
    copy:
      content: "{{ config.stdout[0] }}"
      dest: "backups/backupconfig{{ inventory_hostname }}.txt"
```

Step 5: Creating a ospf yaml file named ospf_config.yaml

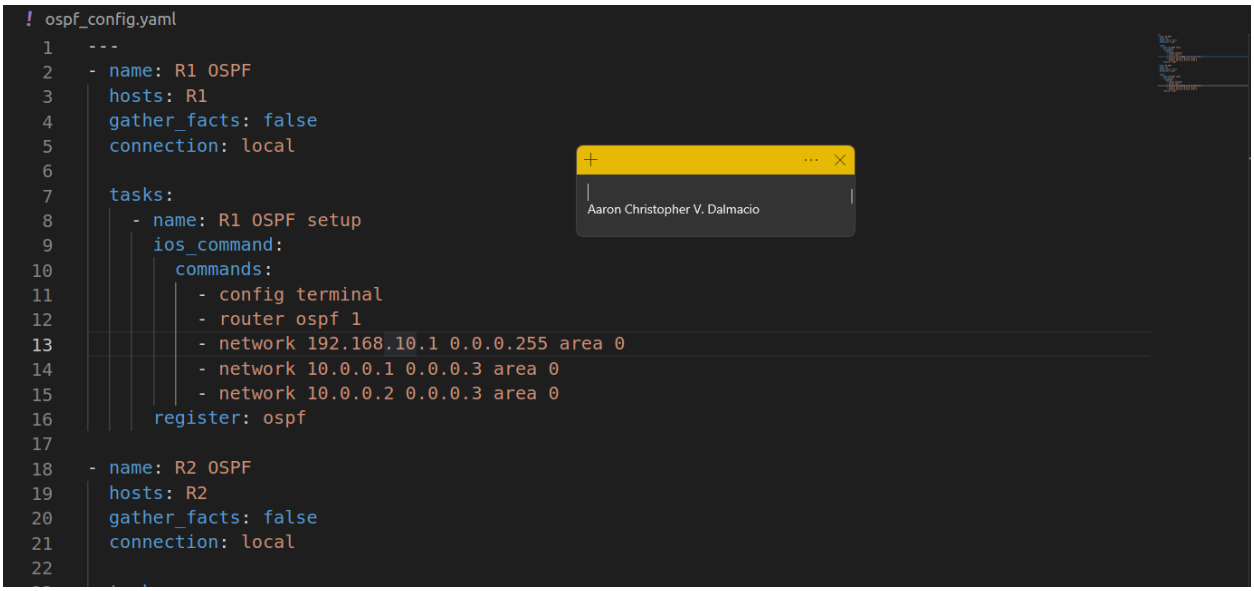


Figure 14. Creating the ospf_config.yaml (Part 1)

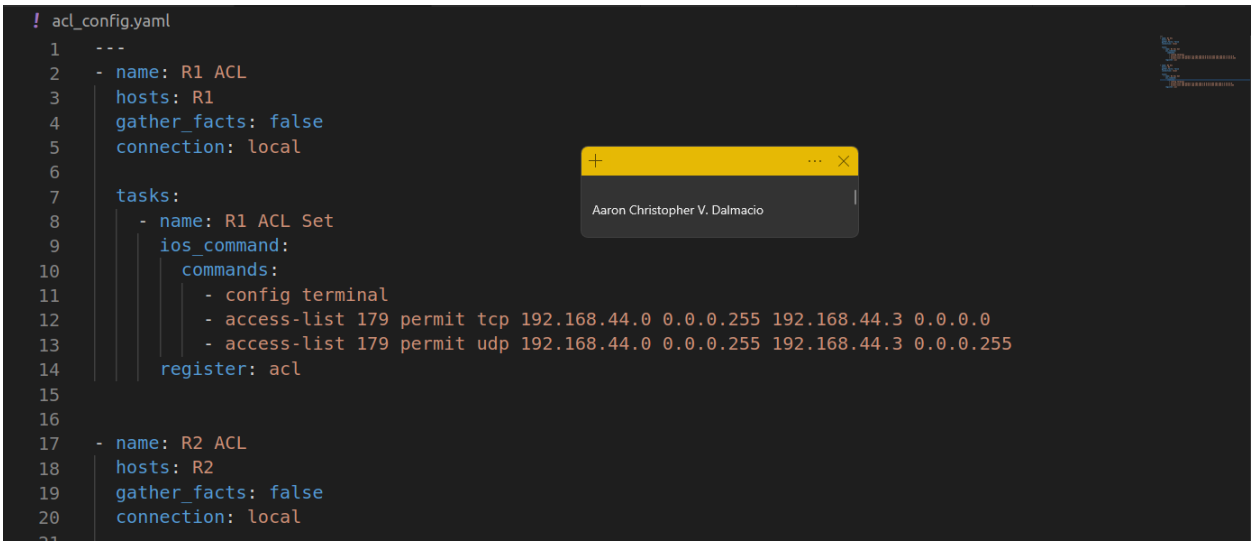
Figure 15. Creating the ospf_config.yaml (Part 2)

Enter the following codes on the ospf_config.yaml:

```
---
- name: R1 OSPF
  hosts: R1
  gather_facts: false
  connection: local

  tasks:
    - name: R1 OSPF setup
      ios_command:
        commands:
          - config terminal
          - router ospf 1
          - network 192.168.10.1 0.0.0.255 area 0
          - network 10.0.0.1 0.0.0.3 area 0
          - network 10.0.0.2 0.0.0.3 area 0
      register: ospf

- name: R2 OSPF
  hosts: R2
  gather_facts: false
  connection: local

  tasks:
```

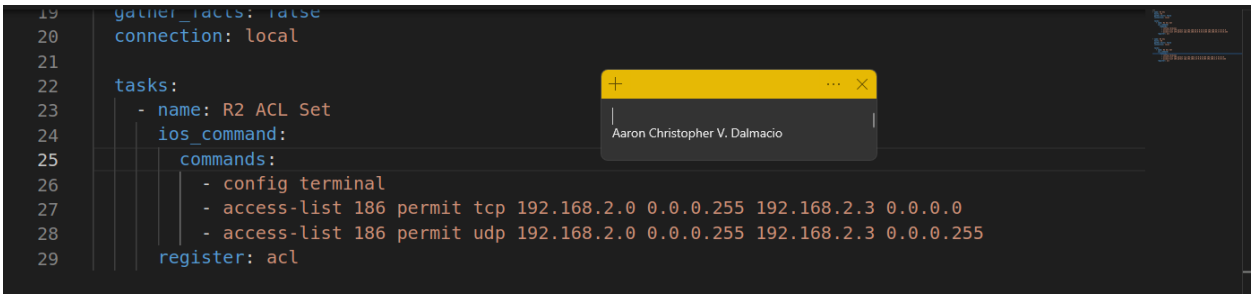
```
- name: R2 OSPF setup
  ios_command:
    commands:
      - config terminal
      - router ospf 1
      - network 192.168.10.1 0.0.0.255 area 0
      - network 10.0.0.1 0.0.0.3 area 0
      - network 10.0.0.2 0.0.0.3 area 0
    register: ospf
```

Step 6: Creating an acl yaml file named acl_config.yaml




```
---
- name: R1 ACL
  hosts: R1
  gather_facts: false
  connection: local

  tasks:
    - name: R1 ACL Set
      ios_command:
        commands:
          - config terminal
          - access-list 179 permit tcp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.0
          - access-list 179 permit udp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.255
      register: acl


- name: R2 ACL
  hosts: R2
  gather_facts: false
  connection: local
```

Figure 16. Creating the acl_config.yaml (Part 1)




```
  connection: local

  tasks:
    - name: R2 ACL Set
      ios_command:
        commands:
          - config terminal
          - access-list 186 permit tcp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.0
          - access-list 186 permit udp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.255
      register: acl
```

Figure 16. Creating the acl_config.yaml (Part 2)

Enter the following codes on the acl_config.yaml:

```
- name: R1 ACL
  hosts: R1
  gather_facts: false
  connection: local

  tasks:
    - name: R1 ACL Set
      ios_command:
        commands:
          - config terminal
          - access-list 179 permit tcp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.0
          - access-list 179 permit udp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.255
      register: acl


- name: R2 ACL
  hosts: R2
  gather_facts: false
  connection: local

  tasks:
    - name: R2 ACL Set
      ios_command:
        commands:
          - config terminal
          - access-list 186 permit tcp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.0
          - access-list 186 permit udp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.255
      register: acl
```

Step 7: Checking the result of Ansible Playbook backup_config.yaml

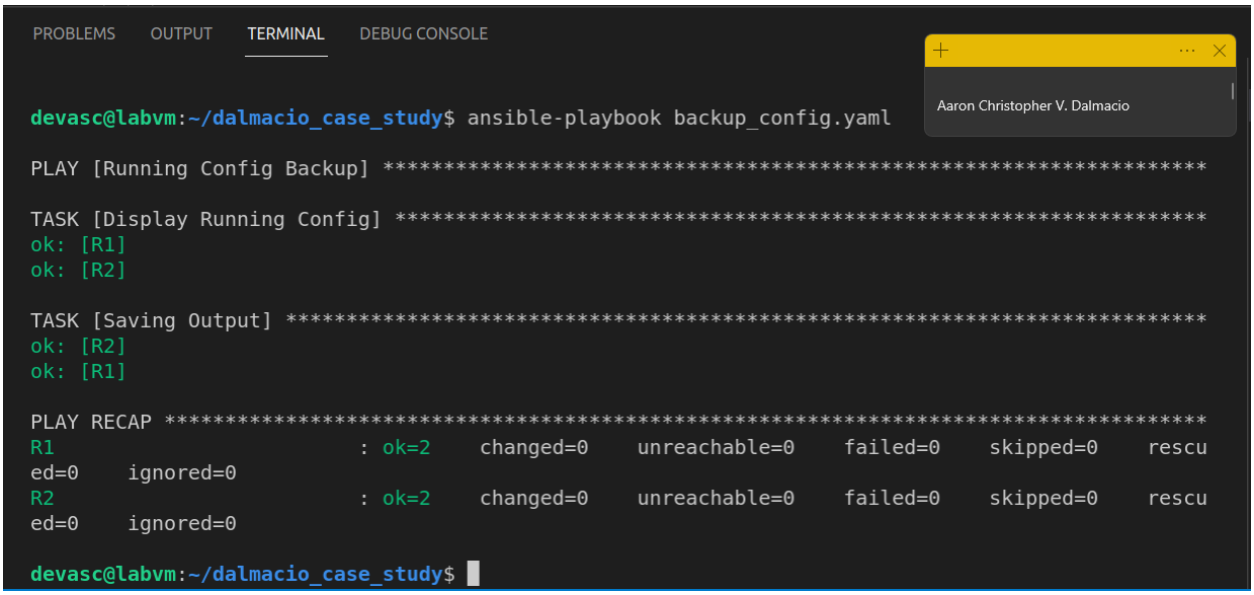


Figure 17. Checking the result of backup_config.yaml

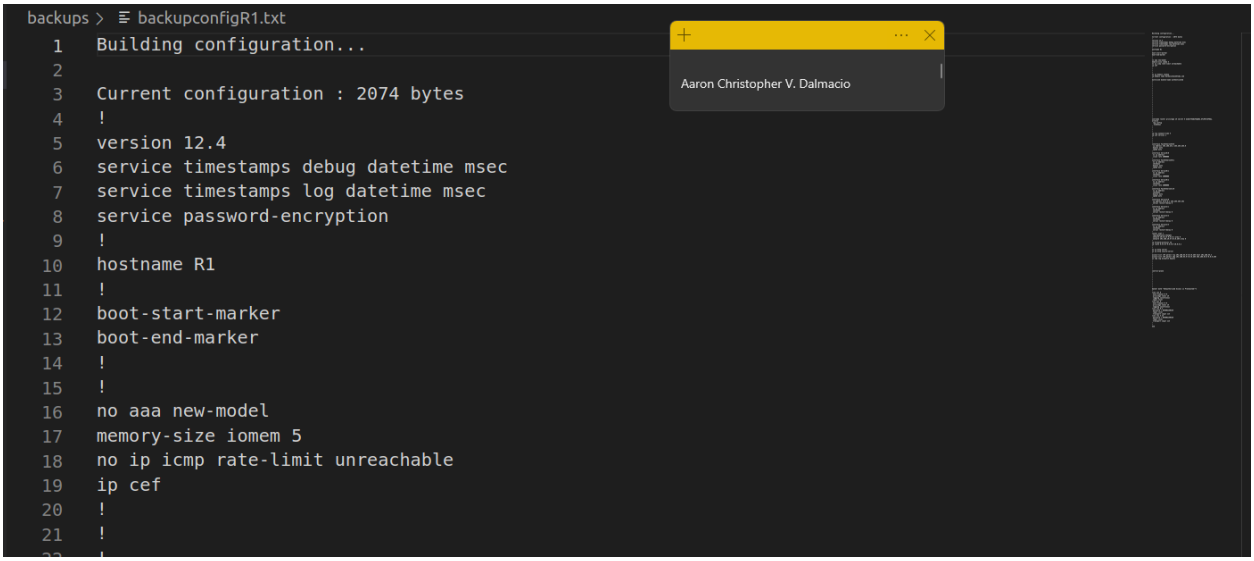


Figure 18. Checking the BackupconfigR1.txt

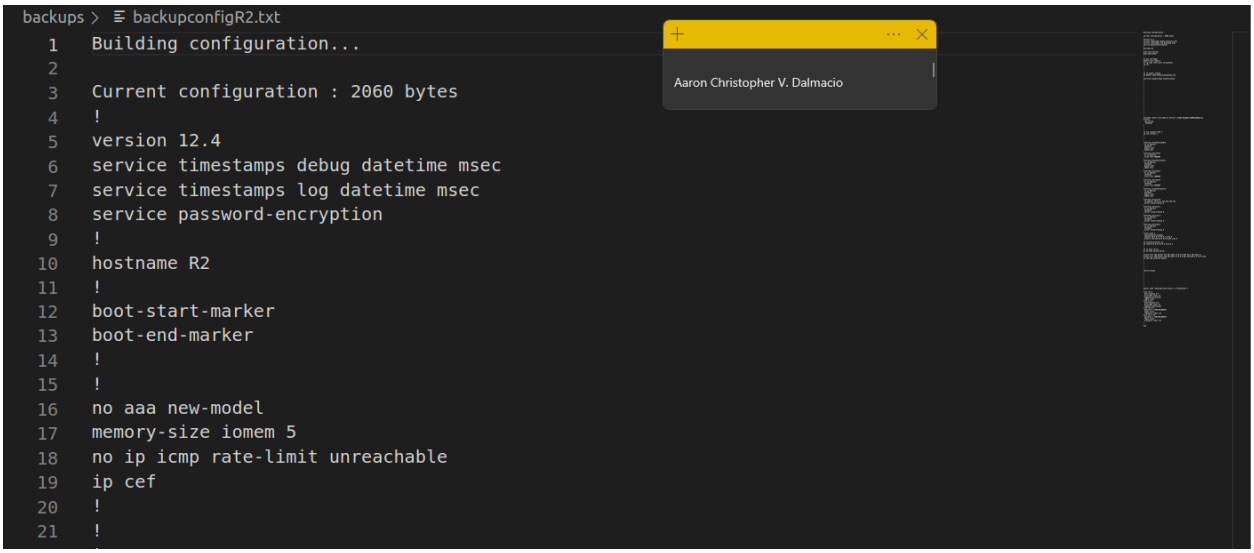


Figure 19. Checking the BackupconfigR2.txt

Step 8: Checking the result of Ansible Playbook ospf_config.yaml

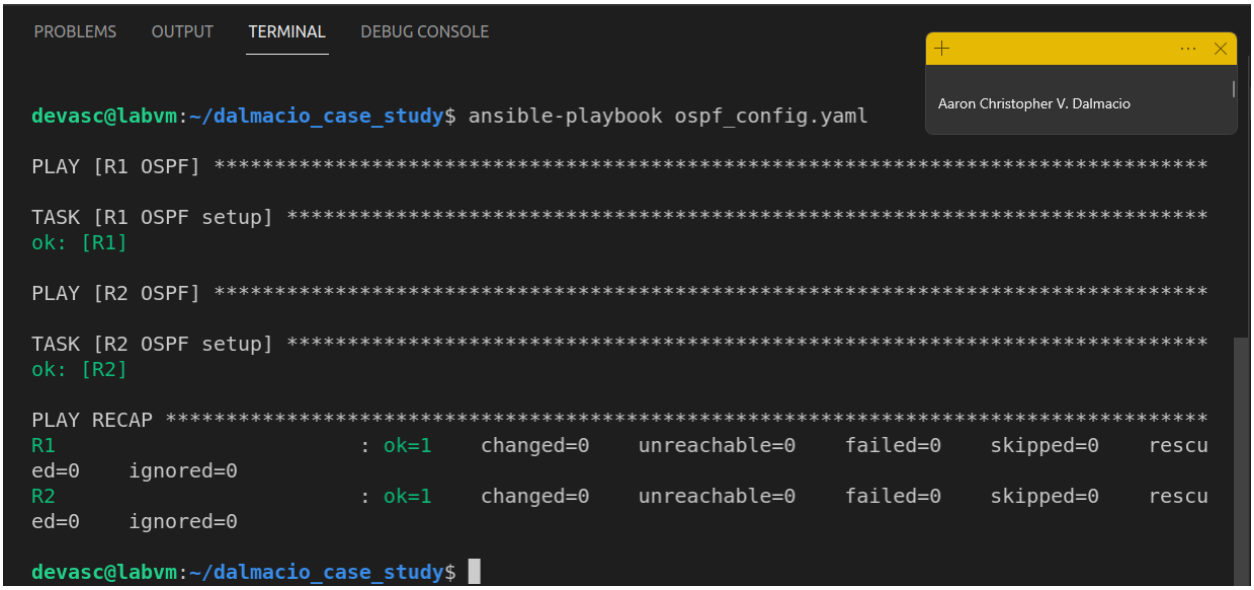


Figure 20. Checking the result of ospf_config.yaml

Step 9. Checking the OSPF neighbors on the Routers

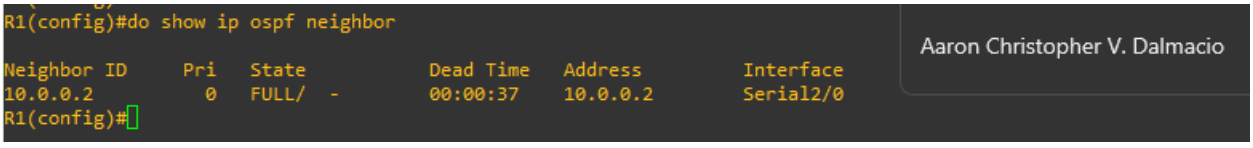


Figure 21. Checking the ospf neighbor of R1

```
R2(config)#do show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.10.1      0   FULL/  -        00:00:33    10.0.0.1        Serial2/0
R2(config)#
```

Aaron Christopher V. Dalmacio

Figure 22. Checking the ospf neighbor of R2

Step 10: Checking the result of Ansible Playbook acl_config.yaml

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

devasc@labvm:~/dalmacio_case_study$ ansible-playbook acl_config.yaml

PLAY [R1 ACL] ******************************************************************************

TASK [R1 ACL Set] **************************************************************************
ok: [R1]

PLAY [R2 ACL] ******************************************************************************

TASK [R1 ACL Set] **************************************************************************
ok: [R2]

PLAY RECAP *********************************************************************************
R1                         : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0
R2                         : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescu
ed=0    ignored=0

devasc@labvm:~/dalmacio_case_study$
```

Aaron Christopher V. Dalmacio

Figure 23. Checking the result of acl_config.yaml

Step 11: Checking the current configuration of the Routers

```
R1(config)#do show access-lists
Extended IP access list 179
    10 permit tcp 192.168.44.0 0.0.0.255 host 192.168.44.3
    20 permit udp 192.168.44.0 0.0.0.255 192.168.44.0 0.0.0.255
R1(config)#
```

Aaron Christopher V. Dalmacio

Figure 24. Showing the Implemented ACL for R1

```
R2(config)#do show access-lists
Extended IP access list 186
    10 permit tcp 192.168.2.0 0.0.0.255 host 192.168.2.3
    20 permit udp 192.168.2.0 0.0.0.255 192.168.2.0 0.0.0.255
R2(config)#
```

Aaron Christopher V. Dalmacio

Figure 25. Showing the Implemented ACL for R2

Step 12: Creating the py file for the pyAts, to test the network

Figure 26. Creating the pyats.py file

Step 13: Creating the script for the pyats file.

Figure 27. Creating the script for the pyats file (Part 1)

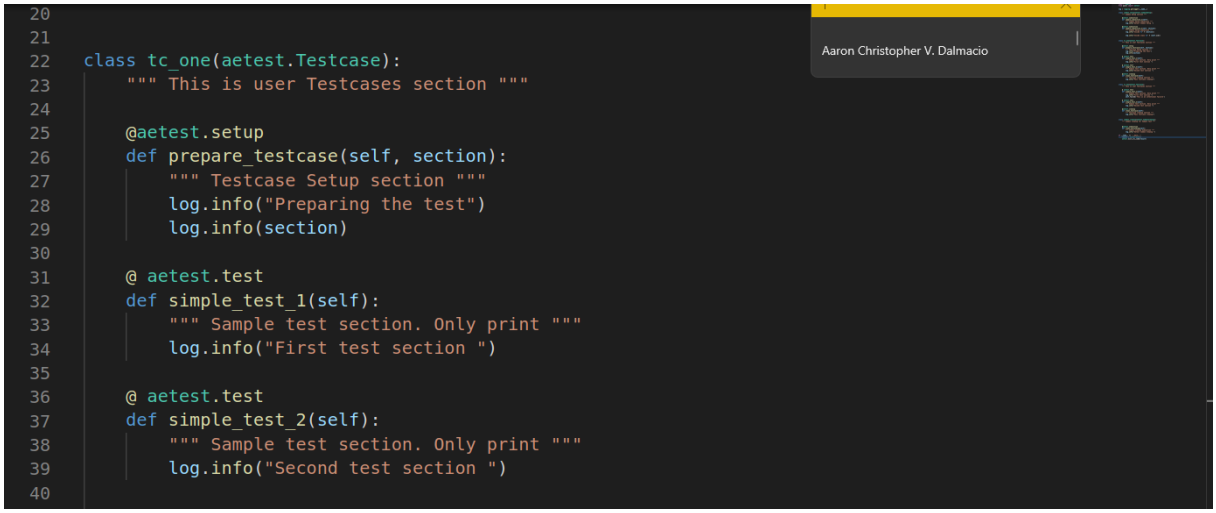


Figure 28. Creating the script for the pyats file (Part 2)

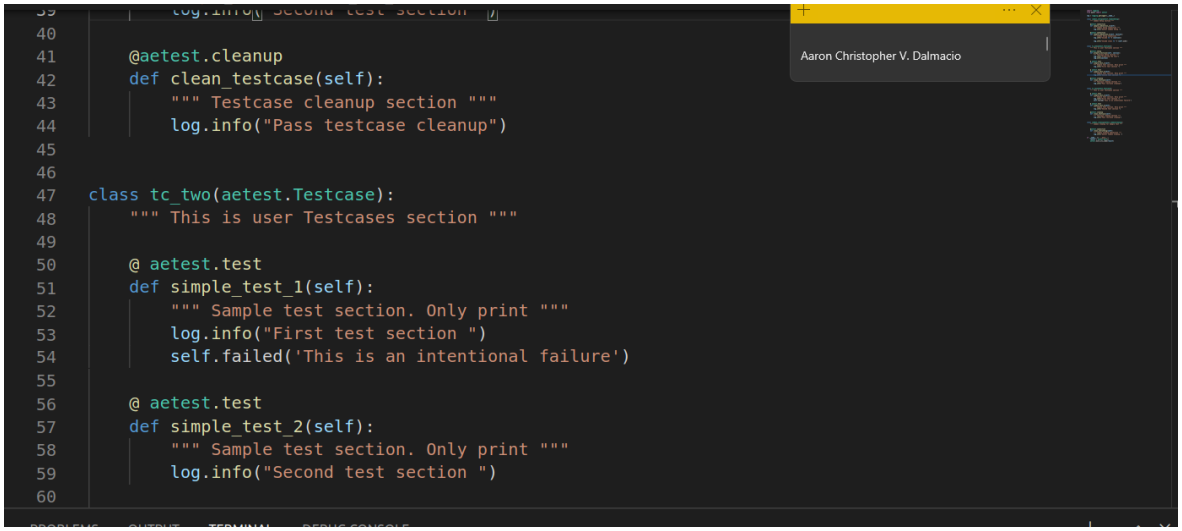


Figure 29. Creating the script for the pyats file (Part 3)

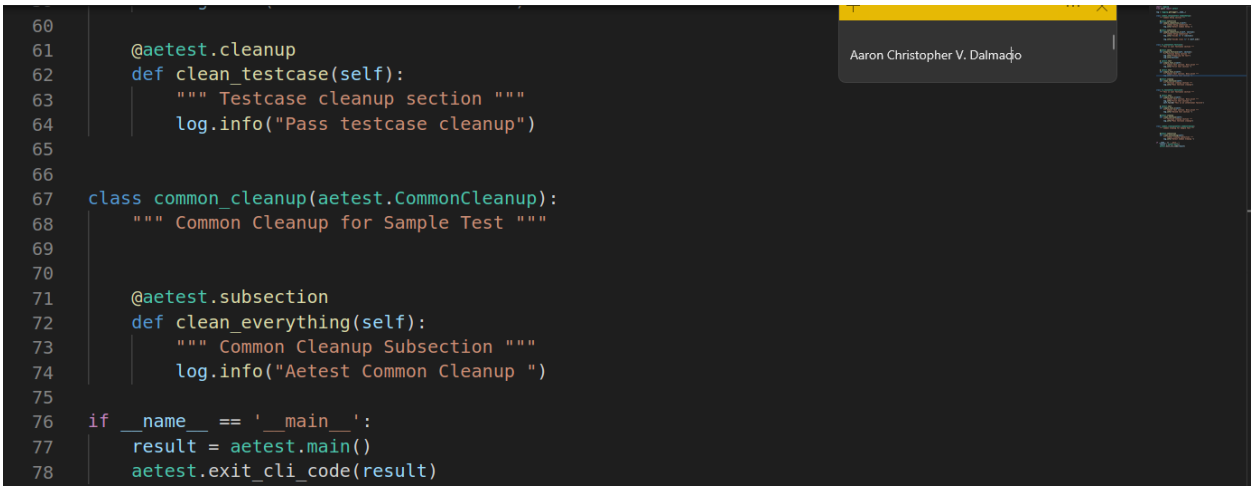


Figure 30. Creating the script for the pyats file (Part 4)

Enter the following code for the script of the Pyats python file

```
import logging
from pyats import aetest

log = logging.getLogger(__name__)

class common_setup(aetest.CommonSetup):
    """ Common Setup section """

    @aetest.subsection
```

```
    def sample_subsection_1(self):
        """ Common Setup subsection """
        log.info("Aetest Common Setup ")

    @aetest.subsection
    def sample_subsection_2(self, section):
        """ Common Setup subsection """
        log.info("Inside %s" % (section))

        log.info("Inside class %s" % (self.uid))


class tc_one(aetest.Testcase):
    """ This is user Testcases section """

    @aetest.setup
    def prepare_testcase(self, section):
        """ Testcase Setup section """
        log.info("Preparing the test")
        log.info(section)

    @ aetest.test
    def simple_test_1(self):
        """ Sample test section. Only print """
        log.info("First test section ")

    @ aetest.test
    def simple_test_2(self):
        """ Sample test section. Only print """
        log.info("Second test section ")

    @aetest.cleanup
    def clean_testcase(self):
        """ Testcase cleanup section """
        log.info("Pass testcase cleanup")
```

```
class tc_two(aetest.Testcase):
    """ This is user Testcases section """

    @ aetest.test
    def simple_test_1(self):
        """ Sample test section. Only print """
        log.info("First test section ")
        self.failed('This is an intentional failure')

    @ aetest.test
    def simple_test_2(self):
        """ Sample test section. Only print """
        log.info("Second test section ")

    @aetest.cleanup
    def clean_testcase(self):
        """ Testcase cleanup section """
        log.info("Pass testcase cleanup")


class common_cleanup(aetest.CommonCleanup):
    """ Common Cleanup for Sample Test """


    @aetest.subsection
    def clean_everything(self):
        """ Common Cleanup Subsection """
        log.info("Aetest Common Cleanup ")

if __name__ == '__main__':
    result = aetest.main()
    aetest.exit_cli_code(result)
```

Step 14: Testing the network pyATS

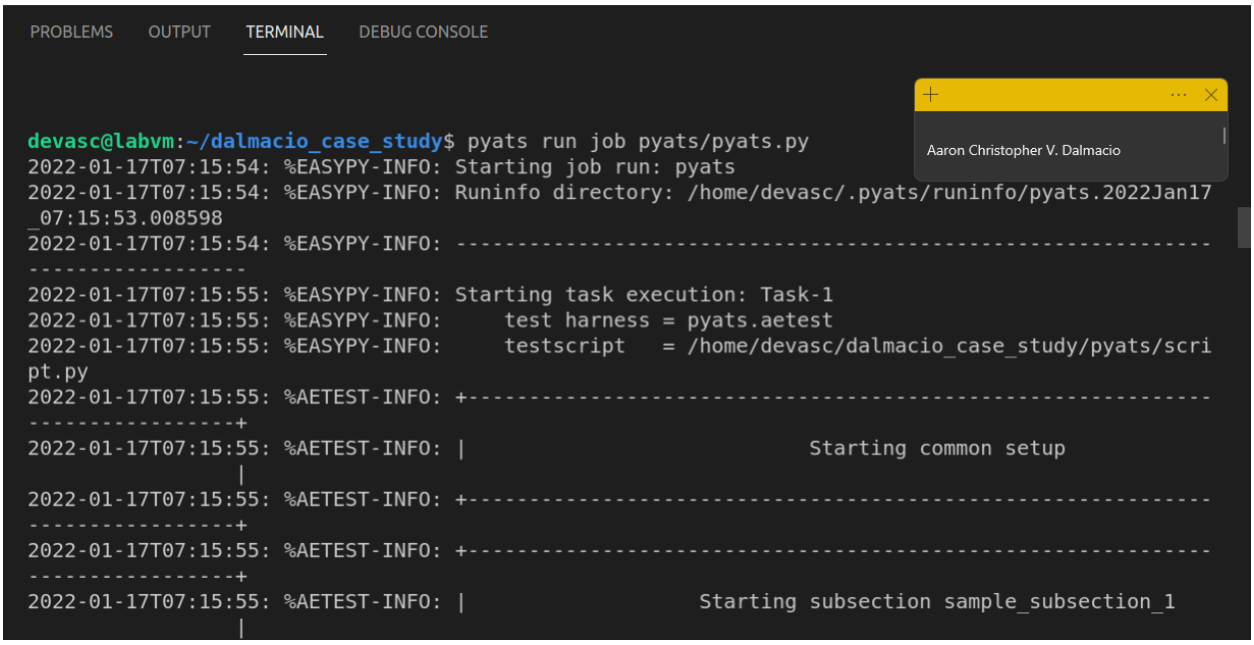


Figure 31. Using the pyATS run job pyats/pyats.py to test the network

Part 4. Using Github to place your Files

In this part, we would be going to create a new repository in our Github account. This would be the repository for all of the files we have created all throughout the project.

Step 1: Creating Github Repo

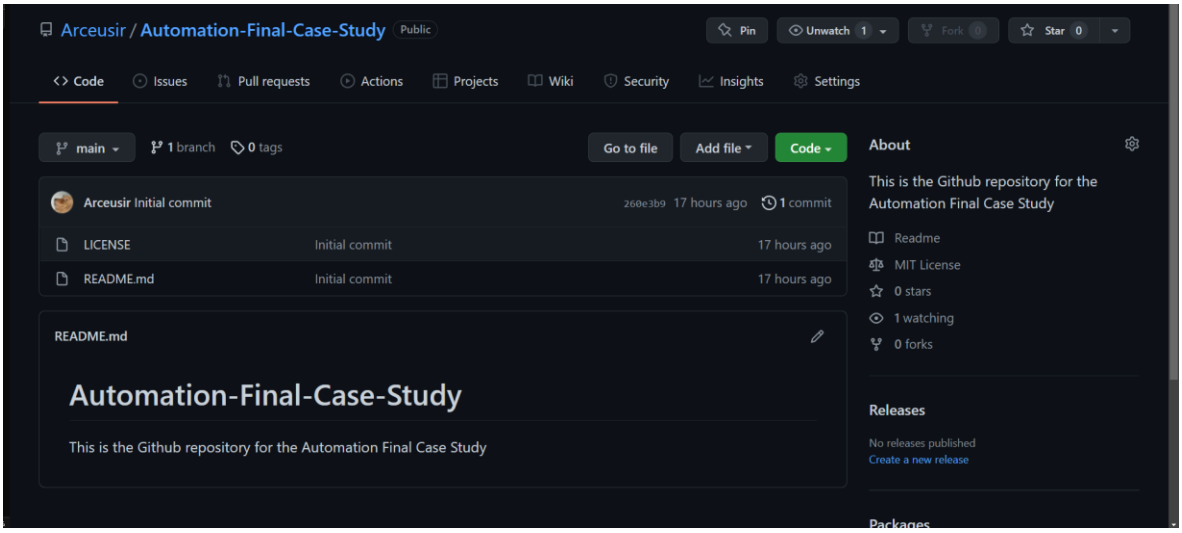


Figure 31. Creating a Github Repository named “Automation Final Case Study”

Step 2: Add files in Github

Use the code below in the terminal

- git init
- git remote add origin https://github.com/Arceusir/Automation-Final-Case-Study.git
- git add -A
- git commit -m “Final Case Study”
- git push -u origin master
- enter username and password
- Check the repository

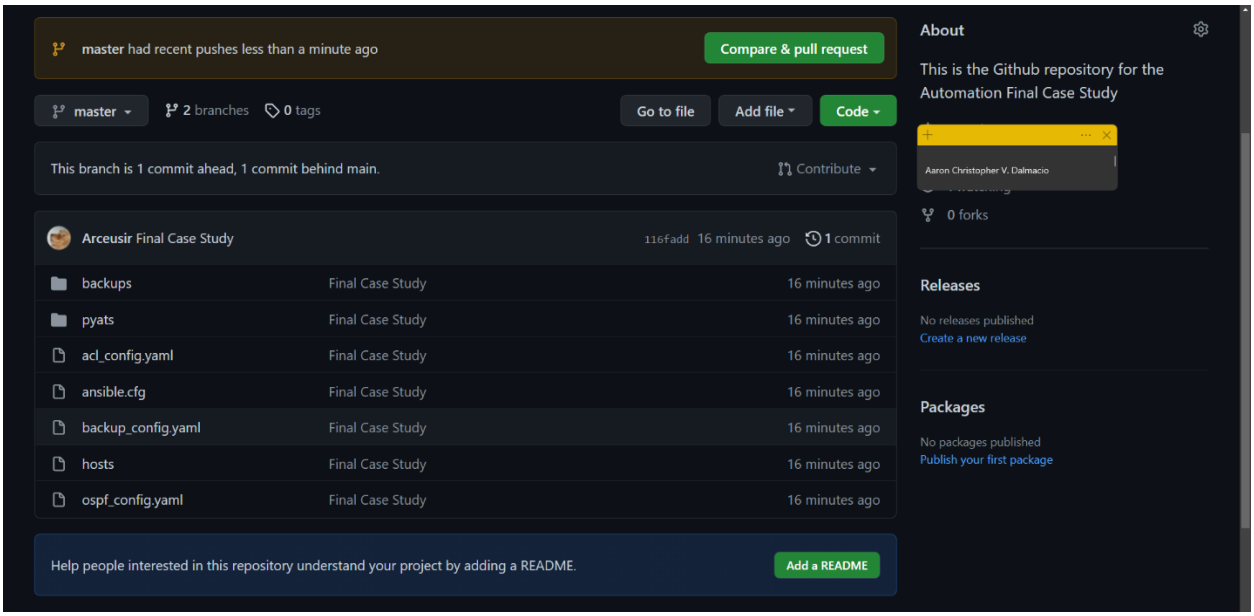


Figure 33. Checking the repository if the files have been pushed

Github Link: https://github.com/Arceusir/Automation-Final-Case-Study.git

Google Drive Link for Video:
https://drive.google.com/drive/folders/1FHDQYfICKBy5rD2thJ562VdT6rxL4HfK?usp=sharing

“I affirm that I have not given or received any unauthorized help on this assignment, and that this work is my own."