

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

## Final Case Study | Network Automation and Programmability

### Topology

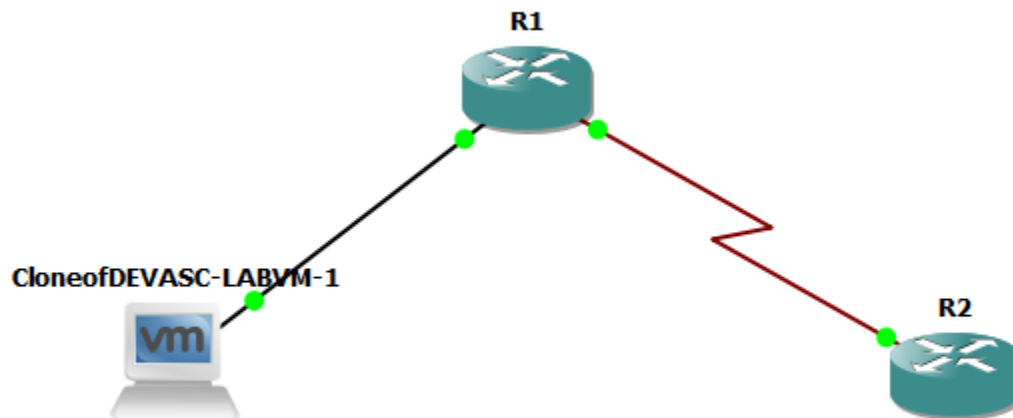


Figure 1. Network Topology

### 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.

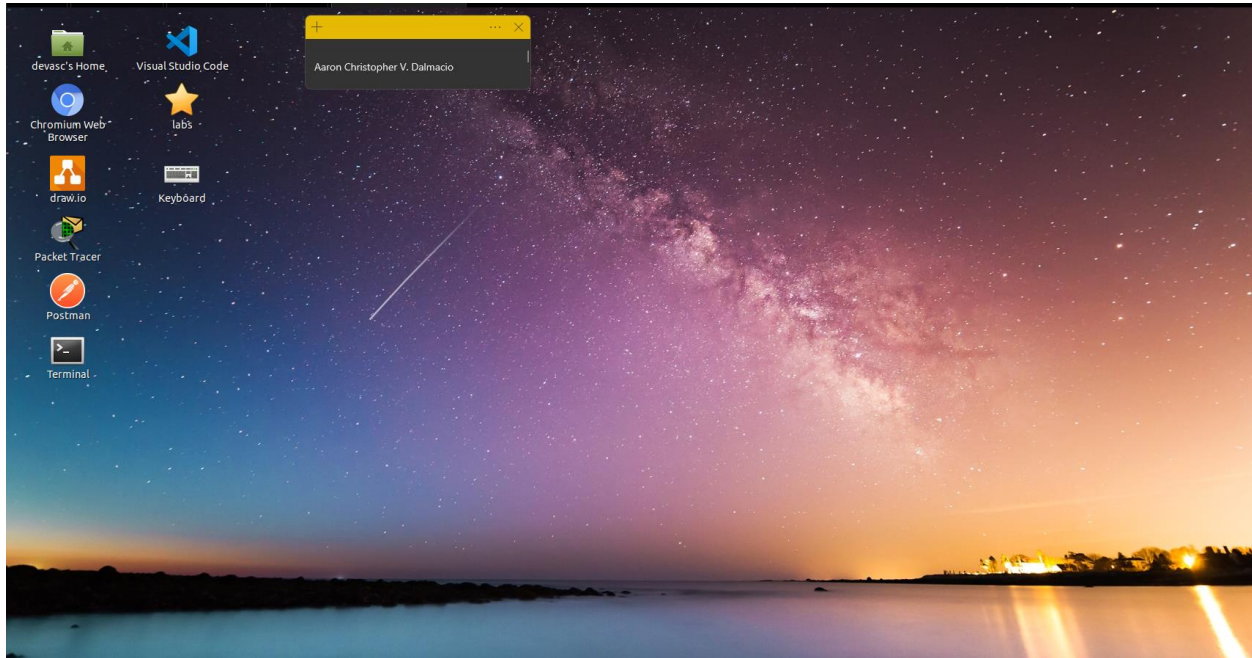


Figure 2. Open the DEVASC VM

### 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

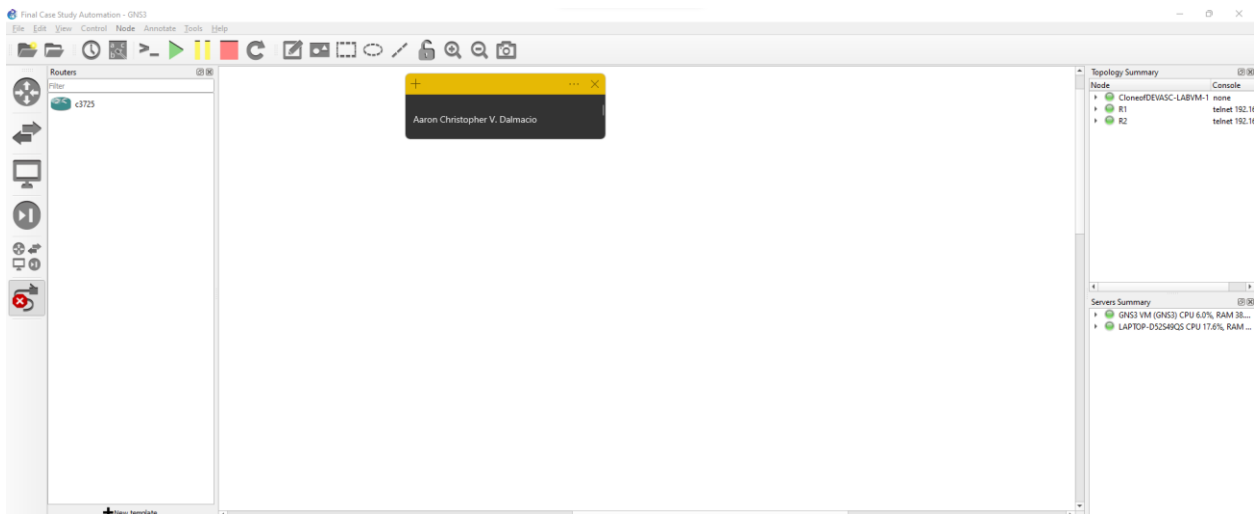


Figure 3. Open the GNS3

### Step 1: Create a new Project

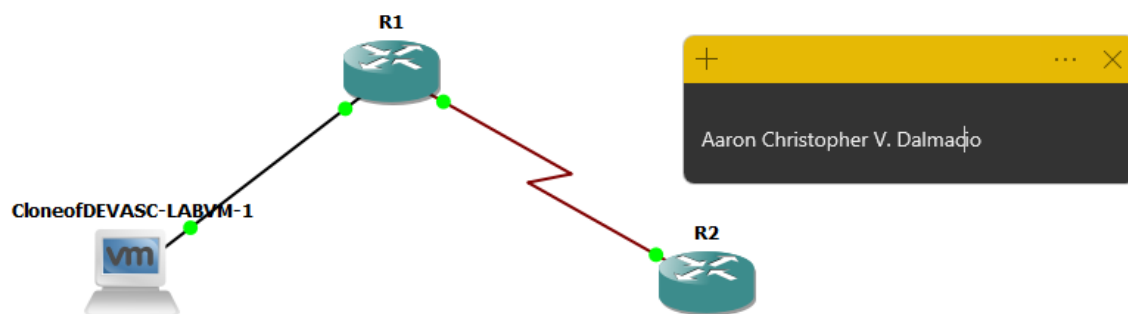
On the GNS3 Software, click on File that could be found on the upper left of the software then click on New blank project. A pop up window will come up and you could now add a new project. Enter the name of the project and find your desired location. After entering the name and location for the project,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

```

R1#enable
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#hostname R1
R1(config)#ip domain-name DalmacioCaseStudy.com
R1(config)#crypto key generate rsa
The name for the keys will be: R1.DalmacioCaseStudy.com
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#ip
*Mar  1 00:04:03.635: %SSH-5-ENABLED: SSH 1.99 has been enabled
R1(config)#ip ssh version 2
R1(config)#line vty 0 15
R1(config-line)#transport input ssh
R1(config-line)#login local
R1(config-line)#password cisco
R1(config-line)#service password-encryption
R1(config)#banner motd "Unauthorized Access is Prohibited!"
R1(config)#line console 0
R1(config-line)#logging synchronous
R1(config-line)#login local
R1(config-line)#username router privilege 15 secret cisco
R1(config)#interface fa0/0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#
*Mar  1 00:06:55.167: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar  1 00:06:56.167: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#

```

+

Aaron Christopher V. Dalmacio

**Figure 5. Applying Basic Configuration and Address on Router 1**

```

R2#enable
R2#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#hostname R2
R2(config)#ip domain-name DalmacioCaseStudy.com
R2(config)#crypto key generate rsa
The name for the keys will be: R2.DalmacioCaseStudy.com
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...[OK]

R2(config)#
*Mar  1 00:07:42.507: %SSH-5-ENABLED: SSH 1.99 has been enabled
R2(config)#ip ssh version 2
R2(config)#line vty 0 15
R2(config-line)#transport input ssh
R2(config-line)#login local
R2(config-line)#password cisco
R2(config-line)#service password-encryption
R2(config)#banner motd "Unauthorized Access is Prohibited!"
R2(config)#line console 0
R2(config-line)#logging synchronous
R2(config-line)#login local
R2(config-line)#username router privilege 15 secret cisco
R2(config)#int s0/0
R2(config-if)#ip address 10.0.0.2 255.255.255.252
R2(config-if)#no shut
R2(config-if)#
*Mar  1 00:10:42.167: %LINK-3-UPDOWN: Interface Serial0/0, changed state to up
R2(config-if)#
*Mar  1 00:10:43.171: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
R2(config-if)#

```

+

Aaron Christopher V. Dalmacio

**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.

```

devasc@labvm:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=255 time=16.2 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=255 time=3.16 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=255 time=8.18 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=255 time=2.33 ms
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 2.329/7.477/16.245/5.534 ms
devasc@labvm:~$

```

Aaron Christopher V. Dalmacio

**Figure 7. Ping 10.0.0.1 in the Terminal**

```

devasc@labvm:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=254 time=33.0 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=254 time=27.3 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=254 time=22.9 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=254 time=26.9 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 22.945/27.553/33.010/3.586 ms
devasc@labvm:~$

```

Aaron Christopher V. Dalmacio

Figure 8. Ping 10.0.0.2 in the Terminal

```

devasc@labvm:~$ ssh router@192.168.10.1
Warning: Permanently added '192.168.10.1' (RSA) to the list of known hosts.
Password:
Unauthorized Access is Prohibited!
R1#

```

Aaron Christopher V. Dalmacio

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.

```

devasc@labvm:/$ cd
devasc@labvm:~$ mkdir dalmacio_case_study
devasc@labvm:~$

```

Aaron Christopher V. Dalmacio

Figure 10. Creating a New Directory named dalmacio\_case\_study

### Step 2: Create Hosts File

In step 2, you must create the hosts file.

```

EXPLORER  ...  Get Started  hosts  x  Aaron Christopher V. Dalmacio
└─ DALMACIO_CASE_STUDY
  └─ hosts
      1 R1 ansible_user=cisco ansible_password=cisco ansible_host=10.0.0.1 ansible_connection=network_cli
      2 R2 ansible_user=cisco ansible_password=cisco ansible_host=10.0.0.2 ansible_connection=network_cli
      3

```

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.

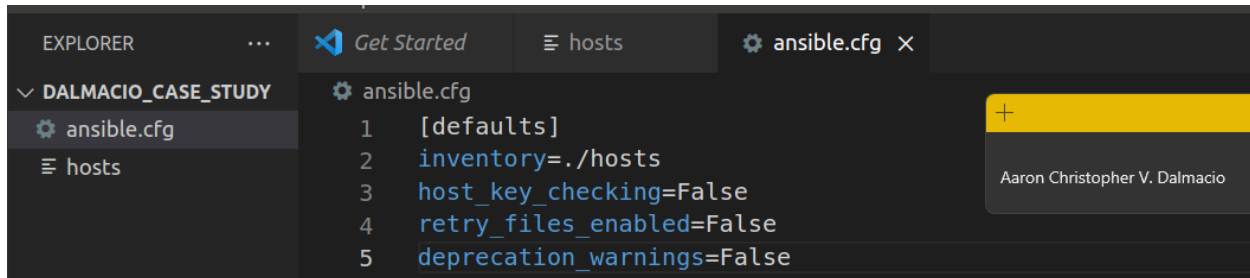


Figure 12. Creating the Ansible.cfg file

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

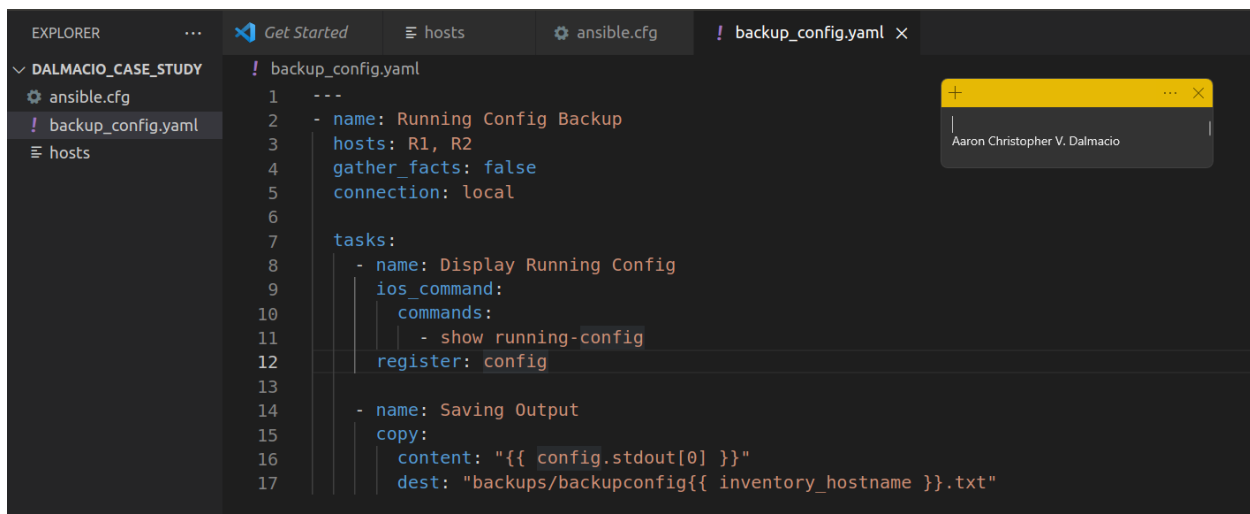


Figure 13. Creating the Backup\_config.yaml

```
---
- 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

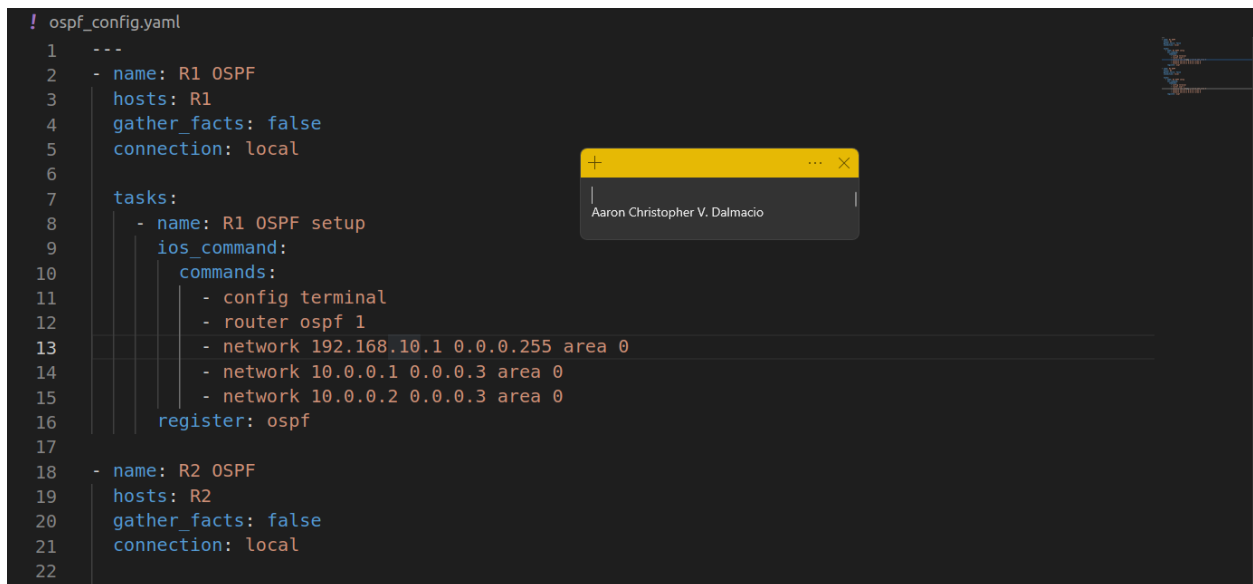
- name: Saving Output

copy:

content: "{{ config.stdout[0] }}"

dest: "backups/backupconfig{{ inventory\_hostname }}.txt"

### Step 5: Creating a ospf yaml file named ospf\_config.yaml



```
! ospf_config.yaml
1 ---
2 - name: R1 OSPF
3   hosts: R1
4   gather_facts: false
5   connection: local
6
7   tasks:
8     - name: R1 OSPF setup
9       ios_command:
10         commands:
11           - config terminal
12           - router ospf 1
13           - network 192.168.10.1 0.0.0.255 area 0
14           - network 10.0.0.1 0.0.0.3 area 0
15           - network 10.0.0.2 0.0.0.3 area 0
16         register: ospf
17
18 - name: R2 OSPF
19   hosts: R2
20   gather_facts: false
21   connection: local
22
```

Figure 14. Creating the ospf\_config.yaml (Part 1)



```
17
18 - name: R2 OSPF
19   hosts: R2
20   gather_facts: false
21   connection: local
22
23   tasks:
24     - name: R2 OSPF setup
25       ios_command:
26         commands:
27           - config terminal
28           - router ospf 1
29           - network 192.168.10.1 0.0.0.255 area 0
30           - network 10.0.0.1 0.0.0.3 area 0
31           - network 10.0.0.2 0.0.0.3 area 0
32       register: ospf
```

**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

```
! acl_config.yaml
1  ---
2  - name: R1 ACL
3    hosts: R1
4    gather_facts: false
5    connection: local
6
7    tasks:
8      - name: R1 ACL Set
9        ios_command:
10          commands:
11            - config terminal
12            - access-list 179 permit tcp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.0
13            - access-list 179 permit udp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.255
14          register: acl
15
16
17 - name: R2 ACL
18   hosts: R2
19   gather_facts: false
20   connection: local
21
```

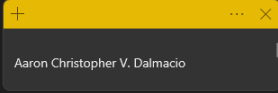


Figure 16. Creating the acl\_config.yaml (Part 1)

```
19   gather_facts: false
20   connection: local
21
22   tasks:
23     - name: R2 ACL Set
24       ios_command:
25         commands:
26           - config terminal
27           - access-list 186 permit tcp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.0
28           - access-list 186 permit udp 192.168.2.0 0.0.0.255 192.168.2.3 0.0.0.255
29         register: acl
```

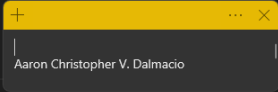


Figure 16. Creating the acl\_config.yaml (Part 2)

Enter the following codes on the acl\_config.yaml:

---

- name: R1 ACL  
hosts: R1  
gather\_facts: false  
connection: local

tasks:

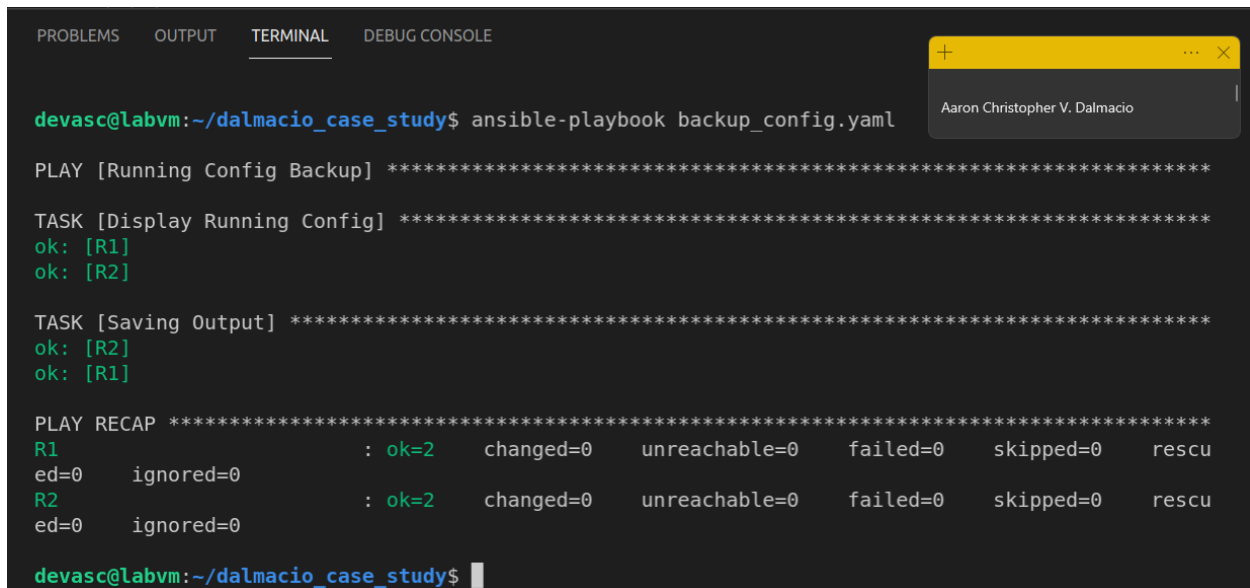
- name: R1 ACL Set  
ios\_command:  
  commands:  
    - config terminal  
    - access-list 179 permit tcp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.0  
    - access-list 179 permit udp 192.168.44.0 0.0.0.255 192.168.44.3 0.0.0.255  
register: acl

- 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**



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

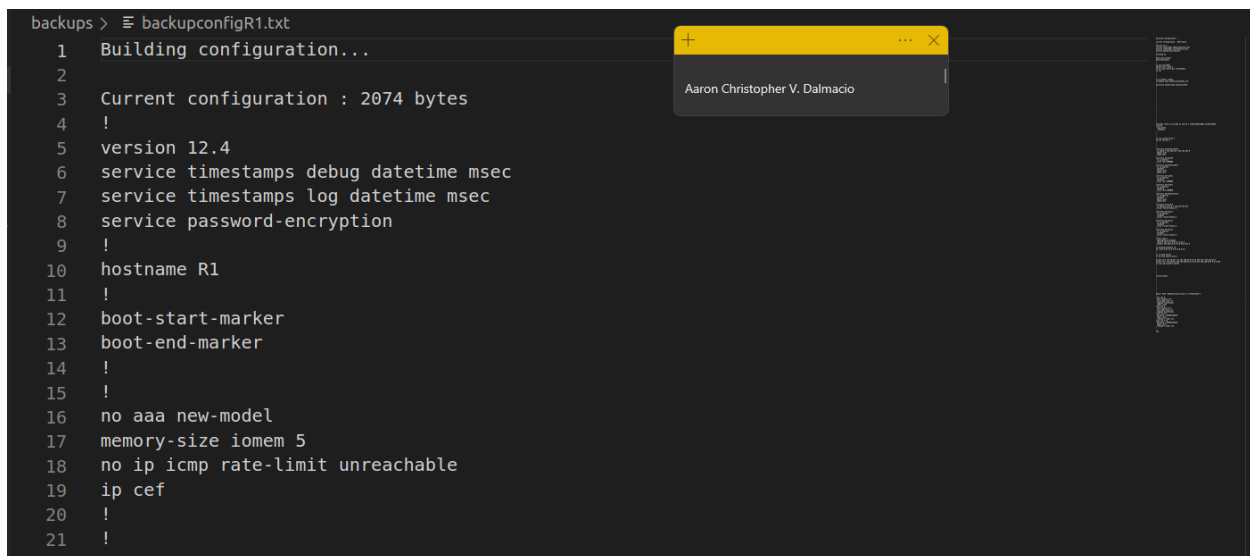
PLAY [Running Config Backup] *****

TASK [Display Running Config] *****
ok: [R1]
ok: [R2]

TASK [Saving Output] *****
ok: [R2]
ok: [R1]

PLAY RECAP *****
R1                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescue=0
R2                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescue=0
devasc@labvm:~/dalmacio_case_study$
```

Figure 17. Checking the result of backup\_config.yaml



```
backups > backupconfigR1.txt
1 Building configuration...
2
3 Current configuration : 2074 bytes
4 !
5 version 12.4
6 service timestamps debug datetime msec
7 service timestamps log datetime msec
8 service password-encryption
9 !
10 hostname R1
11 !
12 boot-start-marker
13 boot-end-marker
14 !
15 !
16 no aaa new-model
17 memory-size iomem 5
18 no ip icmp rate-limit unreachable
19 ip cef
20 !
21 !
22
```

Figure 18. Checking the BackupconfigR1.txt

```
backups > backupconfigR2.txt
1 Building configuration...
2
3 Current configuration : 2060 bytes
4 !
5 version 12.4
6 service timestamps debug datetime msec
7 service timestamps log datetime msec
8 service password-encryption
9 !
10 hostname R2
11 !
12 boot-start-marker
13 boot-end-marker
14 !
15 !
16 no aaa new-model
17 memory-size iomem 5
18 no ip icmp rate-limit unreachable
19 ip cef
20 !
21 !
```

Figure 19. Checking the BackupconfigR2.txt

## Step 8: Checking the result of Ansible Playbook ospf\_config.yaml

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

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

PLAY [R1 OSPF] *****

TASK [R1 OSPF setup] *****
ok: [R1]

PLAY [R2 OSPF] *****

TASK [R2 OSPF setup] *****
ok: [R2]

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

devasc@labvm:~/dalmacio_case_study$
```

Figure 20. Checking the result of ospf\_config.yaml

## Step 9. Checking the OSPF neighbors on the Routers

```
R1(config)#do show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address        Interface
10.0.0.2        0     FULL/ -         00:00:37   10.0.0.2       Serial2/0
R1(config)#
```

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 *****
```

	ok=1	changed=0	unreachable=0	failed=0	skipped=0	rescu
R1	ed=0	ignored=0				
R2	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

```
pyats > pyats.py > ...
1 import os
2 from pyats.easypy import run
3
4 def main():
5     test_path = os.path.dirname(os.path.abspath(__file__))
6     testscript = os.path.join(test_path, 'script.py')
7
8     run(testscript=testscript)
```

Figure 26. Creating the pyats.py file

### Step 13: Creating the script for the pyats file.

```
pyats > script.py > ...
1 import logging
2 from pyats import aetest
3
4 log = logging.getLogger(__name__)
5
6 class common_setup(aetest.CommonSetup):
7     """ Common Setup section """
8
9     @aetest.subsection
10    def sample_subsection_1(self):
11        """ Common Setup subsection """
12        log.info("Aetest Common Setup ")
13
14    @aetest.subsection
15    def sample_subsection_2(self, section):
16        """ Common Setup subsection """
17        log.info("Inside %s" % (section))
18
19        log.info("Inside class %s" % (self.uid))
20
```

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

```
20
21
22 class tc_one(aetest.Testcase):
23     """ This is user Testcases section """
24
25     @aetest.setup
26     def prepare_testcase(self, section):
27         """ Testcase Setup section """
28         log.info("Preparing the test")
29         log.info(section)
30
31     @aetest.test
32     def simple_test_1(self):
33         """ Sample test section. Only print """
34         log.info("First test section ")
35
36     @aetest.test
37     def simple_test_2(self):
38         """ Sample test section. Only print """
39         log.info("Second test section ")
40
```

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

```
39 |         log.info("Second test section ")
40 |
41 |     @aetest.cleanup
42 |     def clean_testcase(self):
43 |         """ Testcase cleanup section """
44 |         log.info("Pass testcase cleanup")
45 |
46 |
47 |     class tc_two(aetest.Testcase):
48 |         """ This is user Testcases section """
49 |
50 |         @aetest.test
51 |         def simple_test_1(self):
52 |             """ Sample test section. Only print """
53 |             log.info("First test section ")
54 |             self.failed('This is an intentional failure')
55 |
56 |         @aetest.test
57 |         def simple_test_2(self):
58 |             """ Sample test section. Only print """
59 |             log.info("Second test section ")
60 |
```

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

```
60 |
61 |     @aetest.cleanup
62 |     def clean_testcase(self):
63 |         """ Testcase cleanup section """
64 |         log.info("Pass testcase cleanup")
65 |
66 |
67 |     class common_cleanup(aetest.CommonCleanup):
68 |         """ Common Cleanup for Sample Test """
69 |
70 |
71 |         @aetest.subsection
72 |         def clean_everything(self):
73 |             """ Common Cleanup Subsection """
74 |             log.info("Aetest Common Cleanup ")
75 |
76 | if __name__ == '__main__':
77 |     result = aetest.main()
78 |     aetest.exit_cli_code(result)
```

**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")

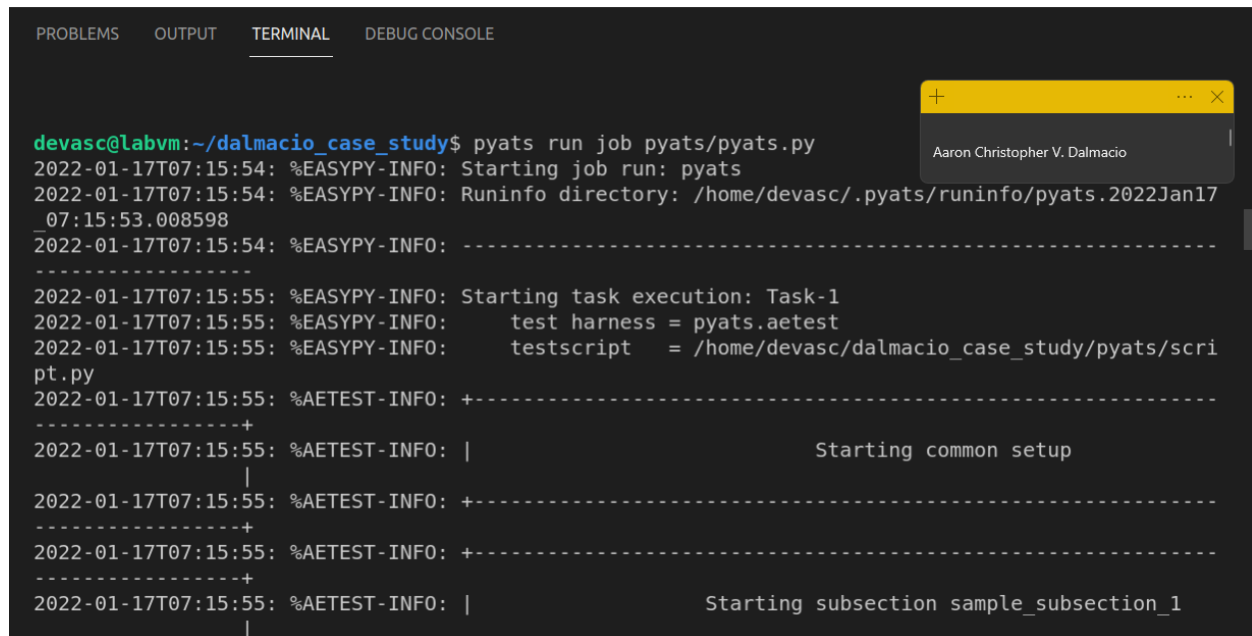
class common_cleanup(aetest.CommonCleanup):
    """ Common Cleanup for Sample Test """

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

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

```

## Step 14: Testing the network pyATS

A terminal window with a dark background and light-colored text. The terminal shows the execution of the command 'pyats run job pyats/pyats.py'. The output includes several informational messages from EASYPY and AETEST, indicating the start of a job run, the directory for runinfo, the start of task execution, and the test harness and testscript paths. The terminal also shows the start of a common setup and a subsection named 'sample\_subsection\_1'. A yellow window title bar is visible at the top right of the terminal window, displaying the name 'Aaron Christopher V. Dalmacio'.

```
devasc@labvm:~/dalmacio_case_study$ pyats run job pyats/pyats.py
2022-01-17T07:15:54: %EASYPY-INFO: Starting job run: pyats
2022-01-17T07:15:54: %EASYPY-INFO: Runinfo directory: /home/devasc/.pyats/runinfo/pyats.2022Jan17_07:15:53.008598
2022-01-17T07:15:54: %EASYPY-INFO: -----
2022-01-17T07:15:55: %EASYPY-INFO: Starting task execution: Task-1
2022-01-17T07:15:55: %EASYPY-INFO:      test harness = pyats.aetest
2022-01-17T07:15:55: %EASYPY-INFO:      testscript  = /home/devasc/dalmacio_case_study/pyats/script.py
2022-01-17T07:15:55: %AETEST-INFO: +-----
2022-01-17T07:15:55: %AETEST-INFO: |                                     Starting common setup
2022-01-17T07:15:55: %AETEST-INFO: +-----
2022-01-17T07:15:55: %AETEST-INFO: +-----
2022-01-17T07:15:55: %AETEST-INFO: |                                     Starting subsection sample_subsection_1
```

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

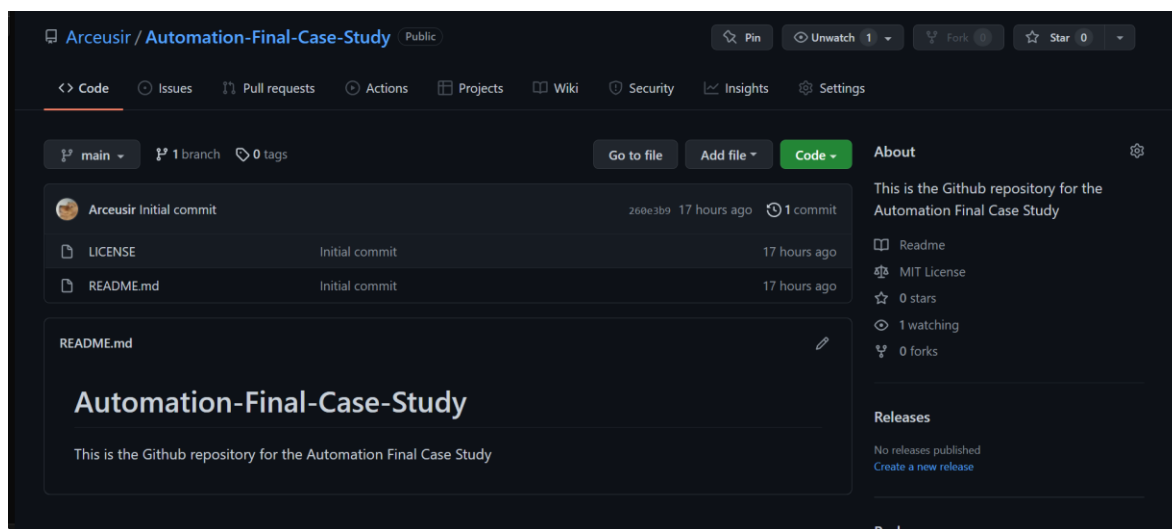
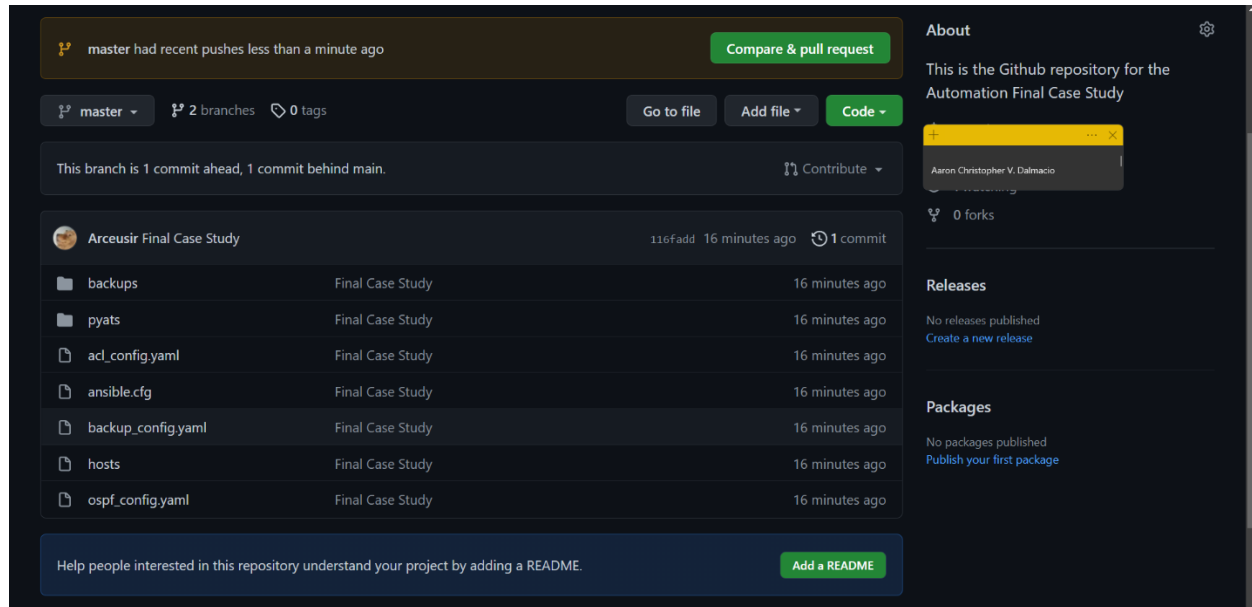


Figure 31. Creating a Github Repository named “Automation Final Case Study”

### 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."**