EUGENIO, Angel G.

CPE41S1

FINAL CASE STUDY

Addressing Table

Device	Interface	IP Address	Subnet Mask
Router1	e0/0	10.0.0.1	255.255.255.252
	e0/1	172.16.0.1	255.255.255.0
Router2	e0/0	10.0.0.2	255.255.255.252
	e0/1	172.16.1.1	255.255.255.0
PC1	NIC	172.16.0.2	255.255.255.0
PC2	NIC	172.16.1.129	255.255.255.0

ILO:

In this Laboratory Activity, the student must be able to gain more skills regarding Network Automation and Programmability.

Objectives:

- Design a Topology on GNS3
- SSH Configuration
- OSPF Configuration
- ACL Configuration
- Backup Router1 Configuration
- pyATS Testing

Required Resoursces

- 1 PC with operating system of your choice
- VirtualBox or VMware
- DEVASC Virtual Machine
- GNS3

Instructions

Part 1: Launch the GNS3

If you do not have GNS3, install it now. If you already have GSN3, launch it now.

Part 2: Create the topology on GNS3

Step 1: Create a New Project on GNS3

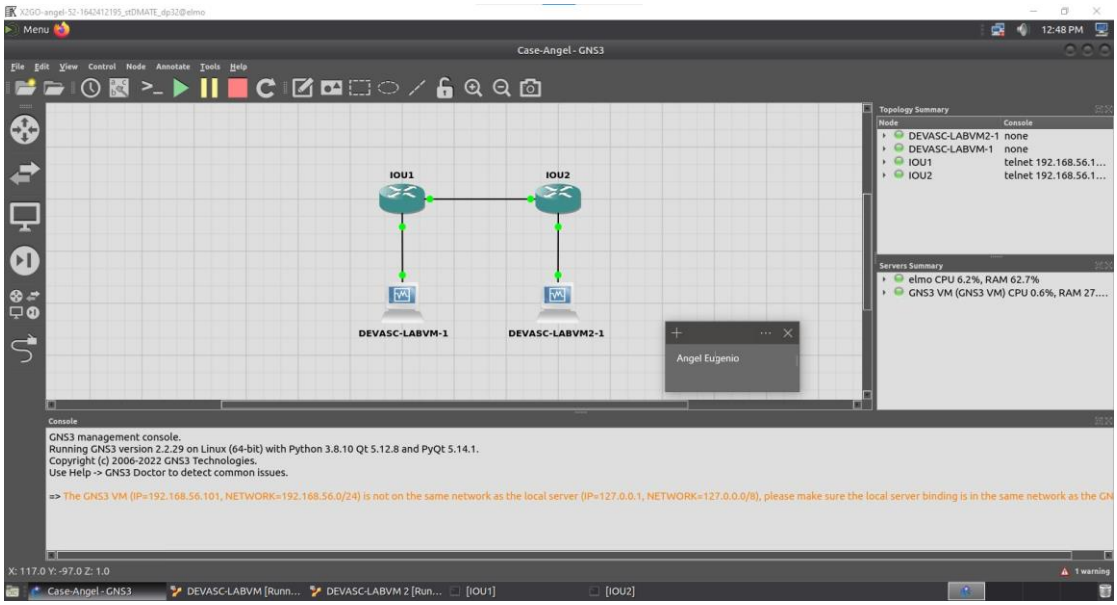
a) Create a New Project file and name it as Case_Study.

Step 2: Design a topology and implement it on your GNS3

a) Select Browse Routers and drag two routers on the workspace.
b) Select Browse End Devices and drag two DEVASC_LABVM on the workspace.
c) Select add a link and connect the routers and end devices on the workspace.

Step 3: Start all the Nodes

Click the Green Triangle to start all the nodes in the workspace.

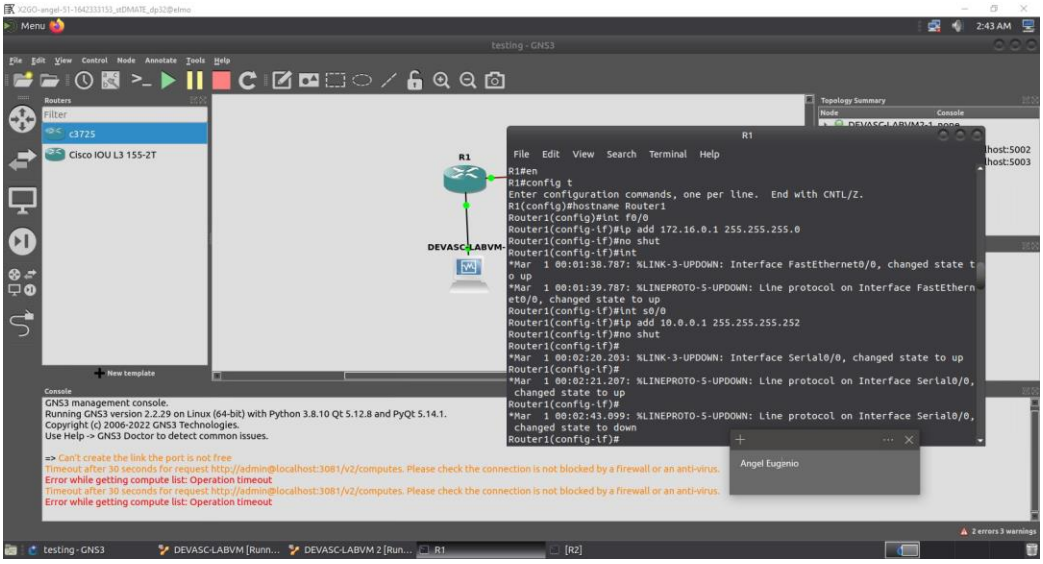


Part 3: SSH Configuration

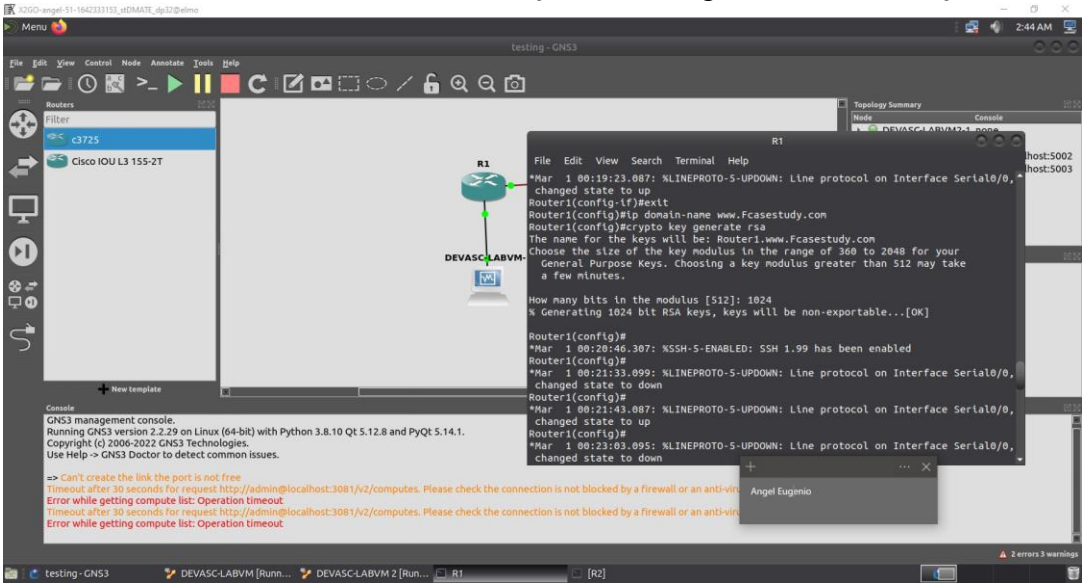
SSH Configuration is used to establish a secure connection within a network. It allows to encrypt and authenticate all connections.

Step 1: Implement Basic Configuration on the Routers

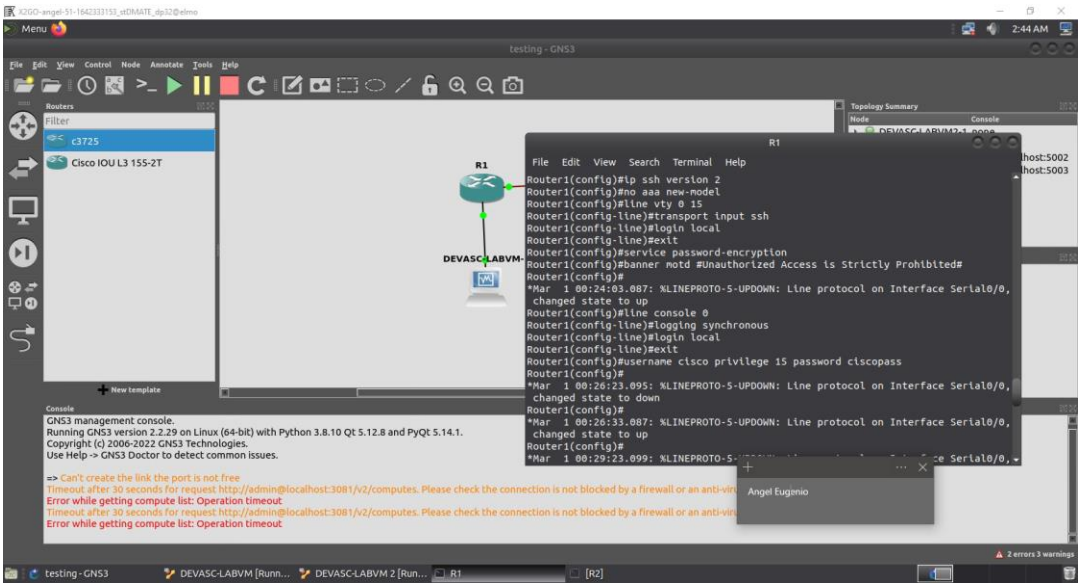
a) Configure the hostname and IP address of the first router as stated in the address table. The first router will be named as Router 1.



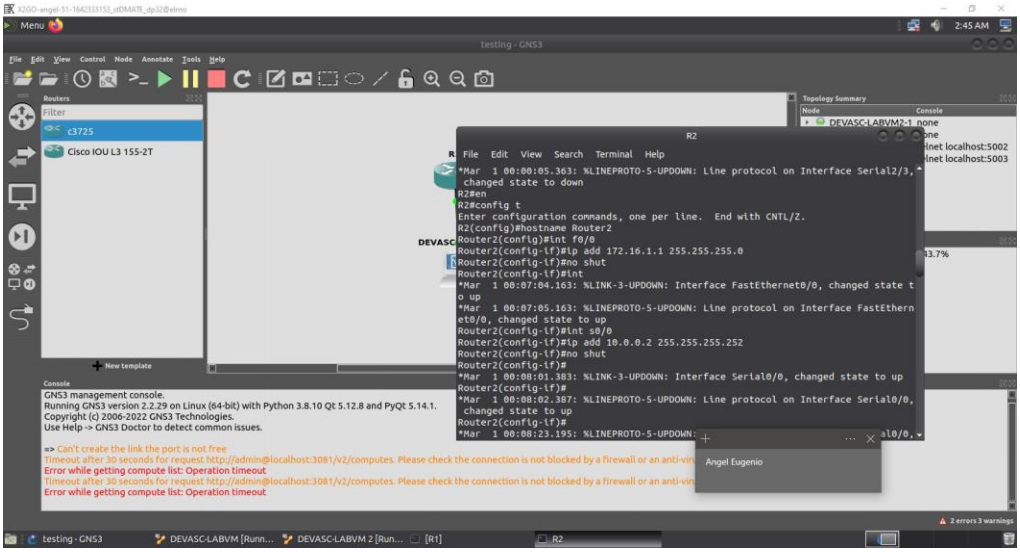
b) Set the domain name as ‘www.casestudy.com’ and generate RSA key.

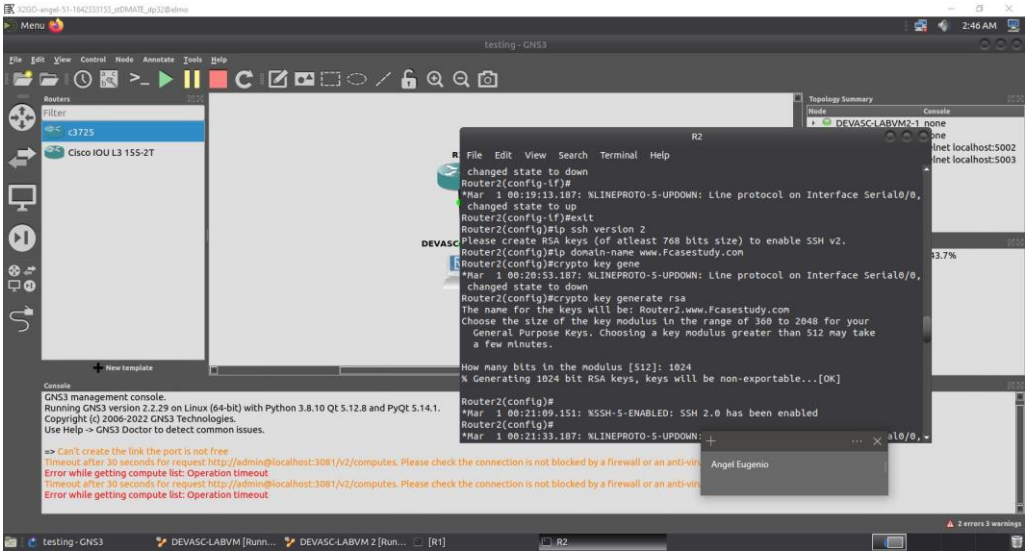


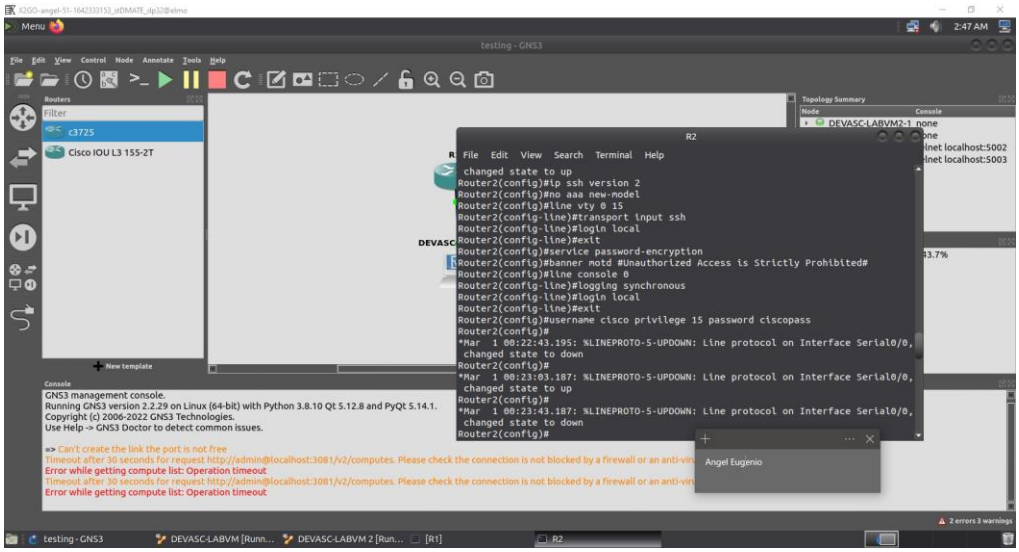
c) Configure SSH to establish a secure connection. Implement the ssh version 2. Use cisco as the username and ciscopass as the password.



d) Do the same configuration on the other router. This time set the hostname of the router as Router2.

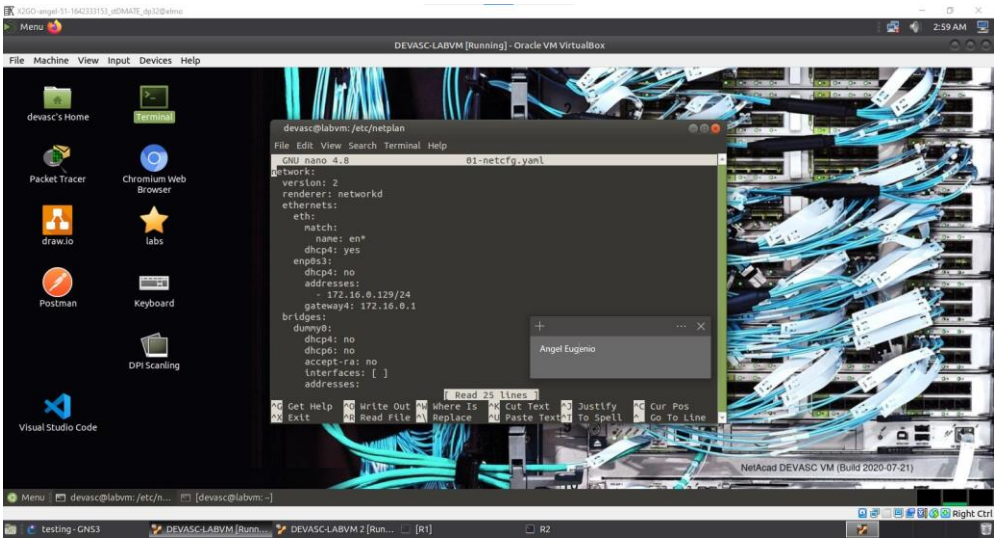




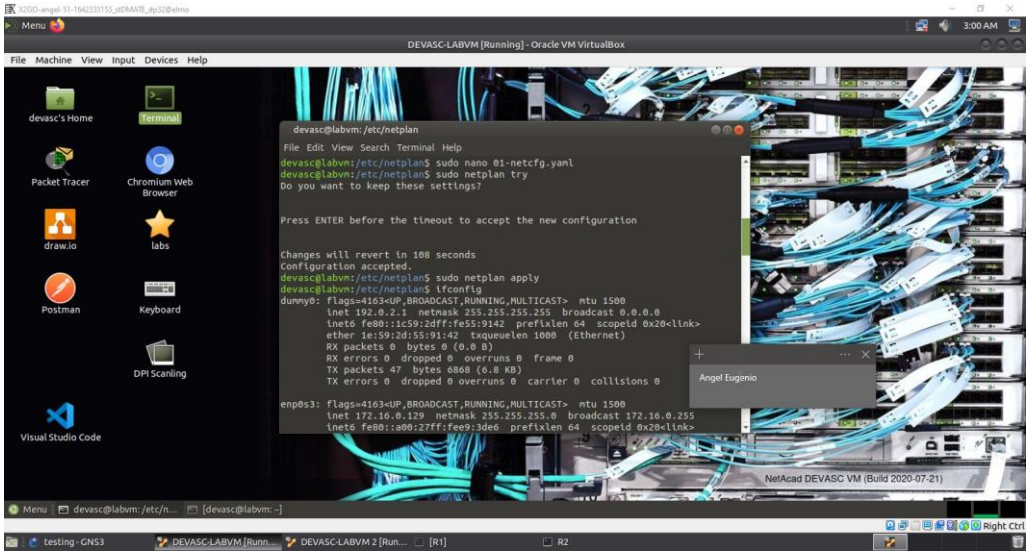


Step 2: Check the network interface of the end-user device.

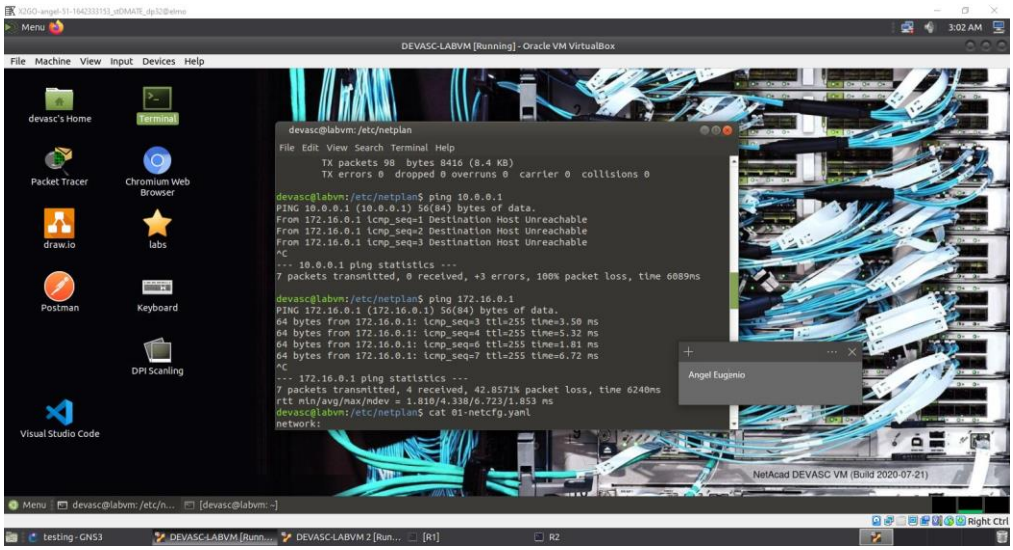
a) Change the directory and go to /etc/netplan/. Modify the 01-netcfg.yaml that can be found within the directory. Use **sudo nano 01-netcfg.yaml**.



b) After modifying the said file, apply the changes on the network by using the **sudo netplan apply** command. Then, check the network interface by applying the **ifconfig** command.

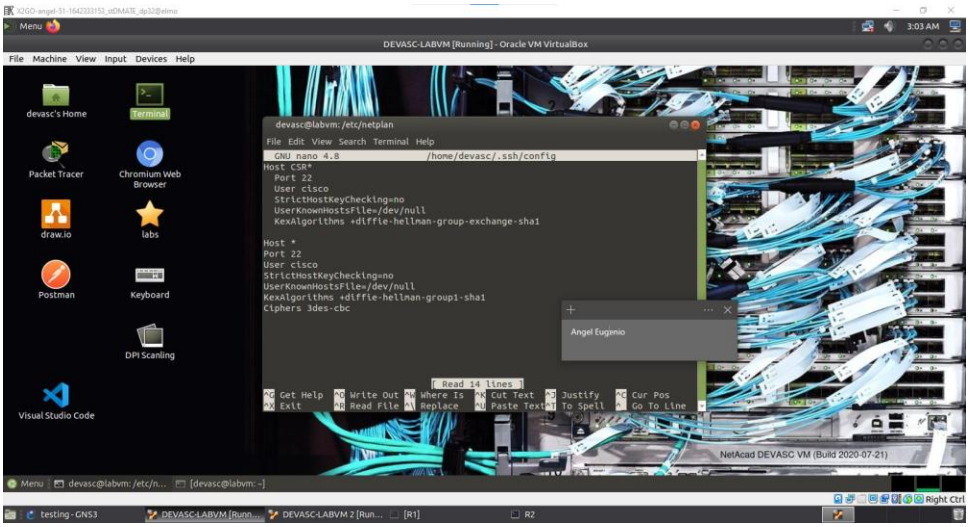


c) Verify that the IP addresses are reachable. Use the **ping** command.

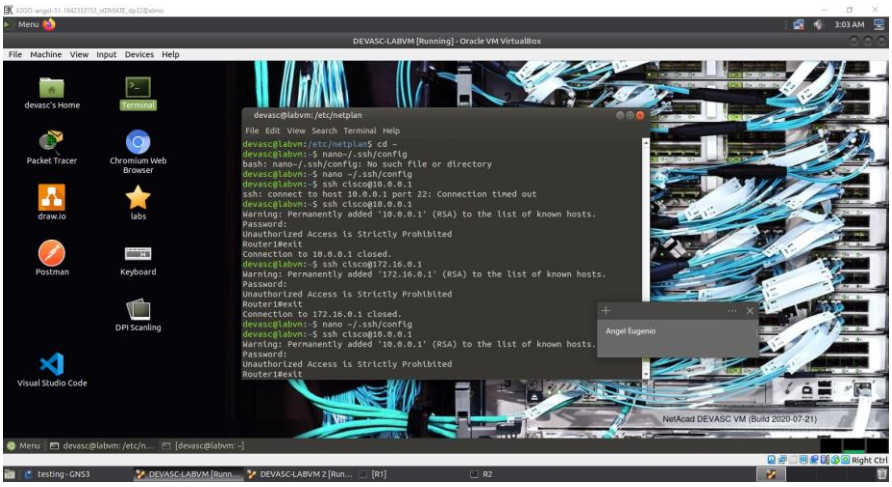


Step 3: Securely Access the Routers on the PC

a) Modify the Configuration of SSH by using **nano ~/.ssh/config** command.



b) Access the network via **ssh [username]@[address]** command. After accessing the network successfully, enter the **exit** command to exit.

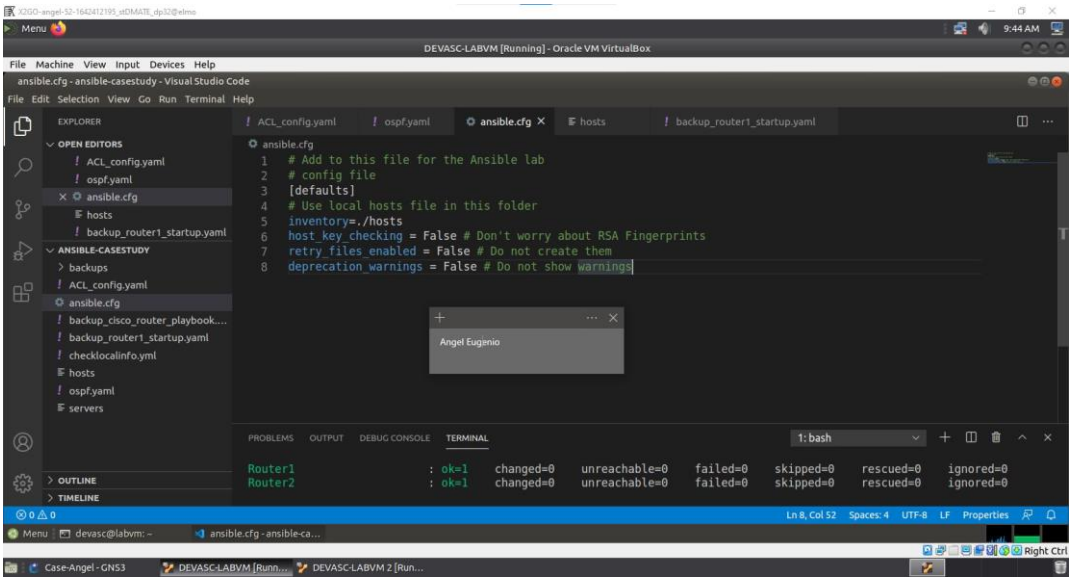


Part 4: OSPF Configuration

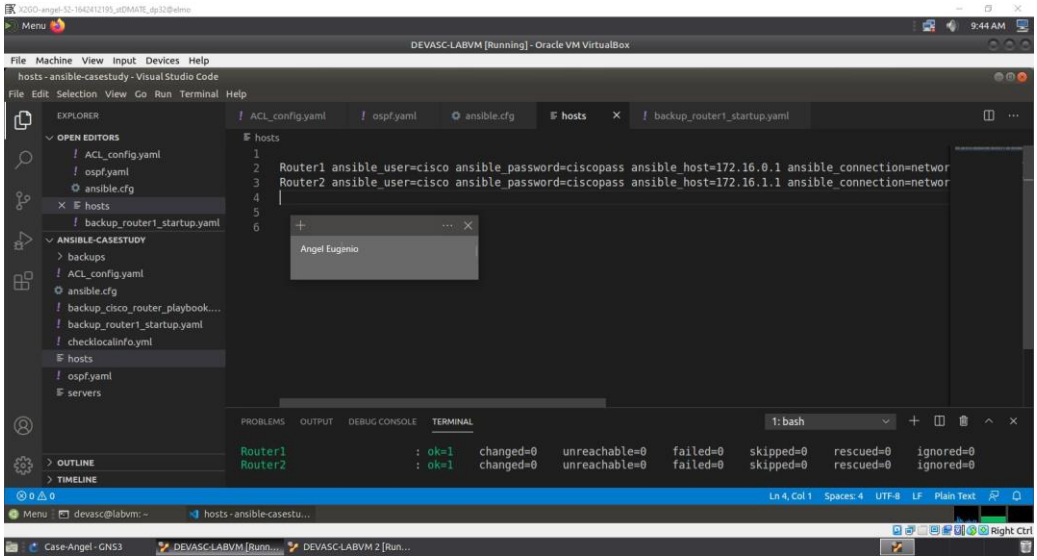
OSPF is established to allow routers to exchange information with each other. Through the given information, it can help the router to know shortest path through the network it belongs to.

Step 1: Edit the Ansible inventory file

a) Check the ansible.cfg file. It is responsible for telling the Ansible where to find the inventory file and for setting certain default parameters.



b) Check the hosts file. It is the one used by Ansible for containing the device information used by Ansible playbooks. Modify the information based on the one configured on the routers.



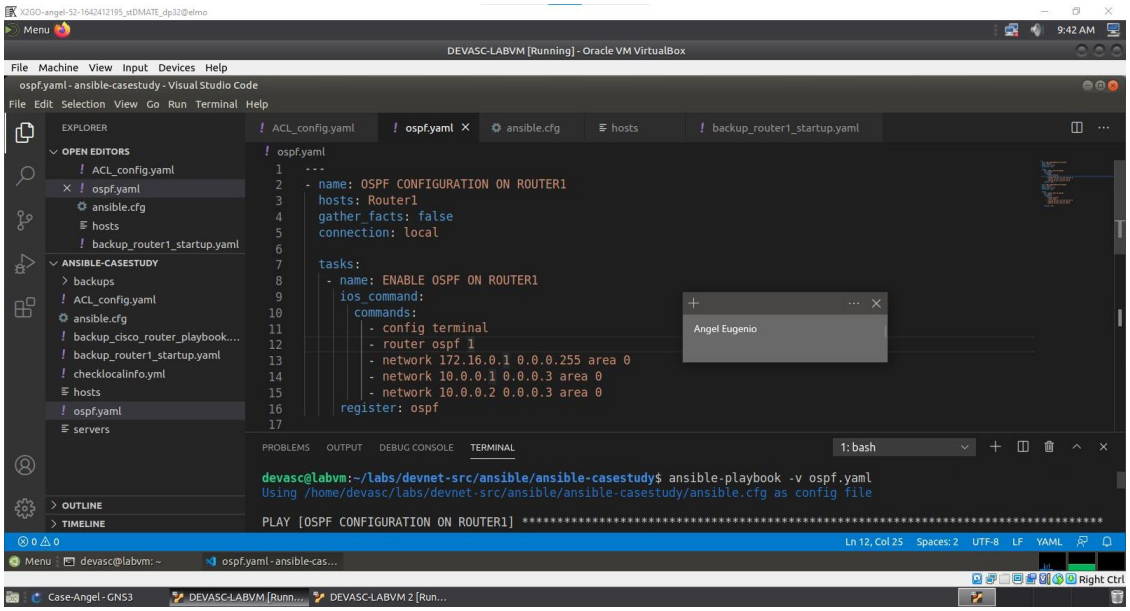
Step 2: Create Ansible Playbook for OSPF Configuration

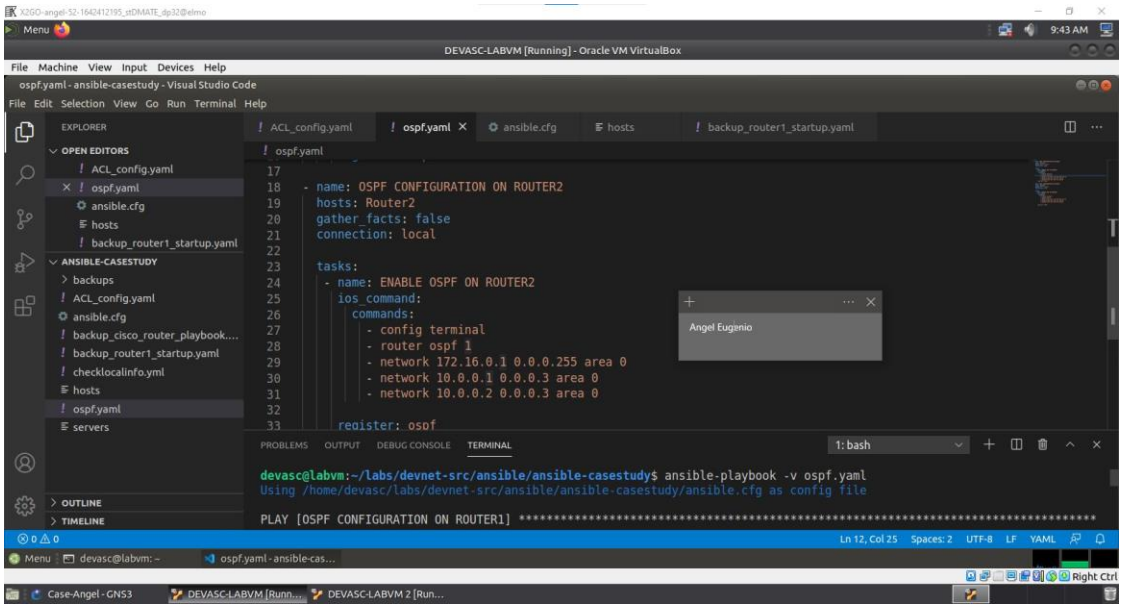
Create the YAML file that would enable OSPF on both routers. Copy the following code.

```
---
- name: OSPF CONFIGURATION ON ROUTER1
  hosts: Router1
  gather_facts: false
  connection: local

  tasks:
   - name: ENABLE OSPF ON ROUTER1
     ios_command:
       commands:
         - config terminal
         - router ospf 1
         - network 172.16.0.1 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: OSPF CONFIGURATION ON ROUTER2
  hosts: Router2
  gather_facts: false
  connection: local

  tasks:
   - name: ENABLE OSPF ON ROUTER2
     ios_command:
```

commands:
- config terminal
- router ospf 1
- network 172.16.0.1 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

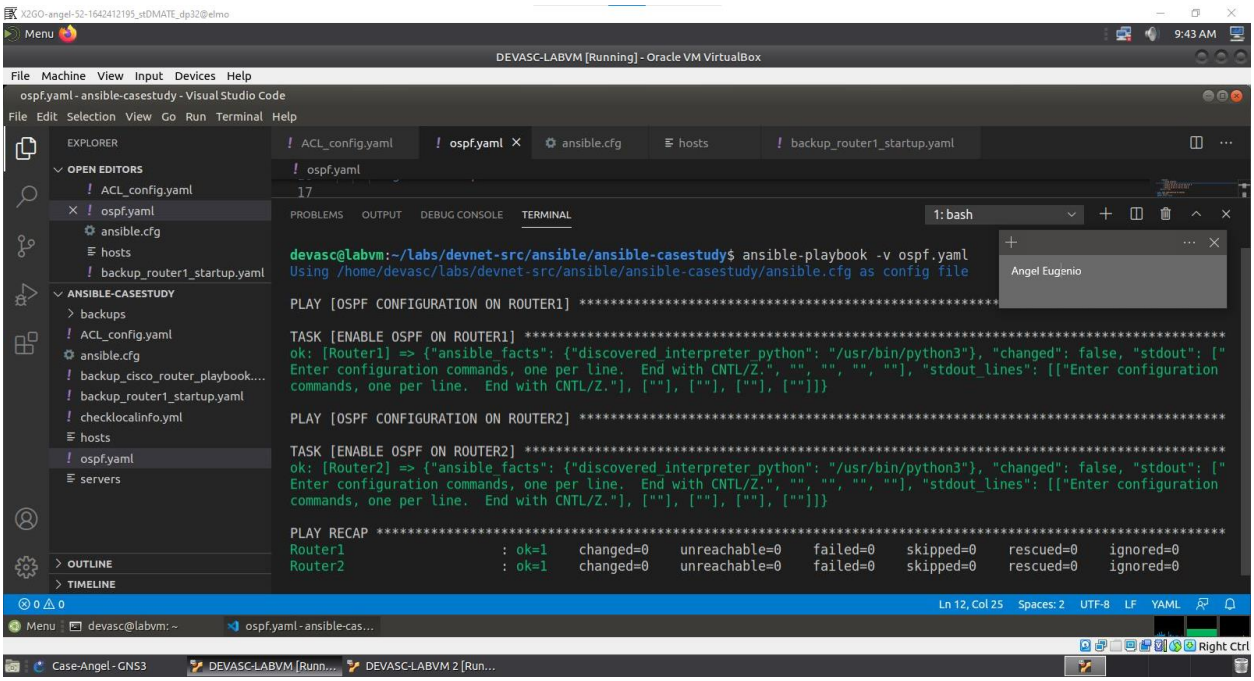
register: ospf





Step 3: Run the Ansible Playbook for OSPF Configuration

Run the created YAML file to apply the OSPF configuration. Use the **ansible-playbook -v [filename]** command.



PART 5: ACL CONFIGURATION

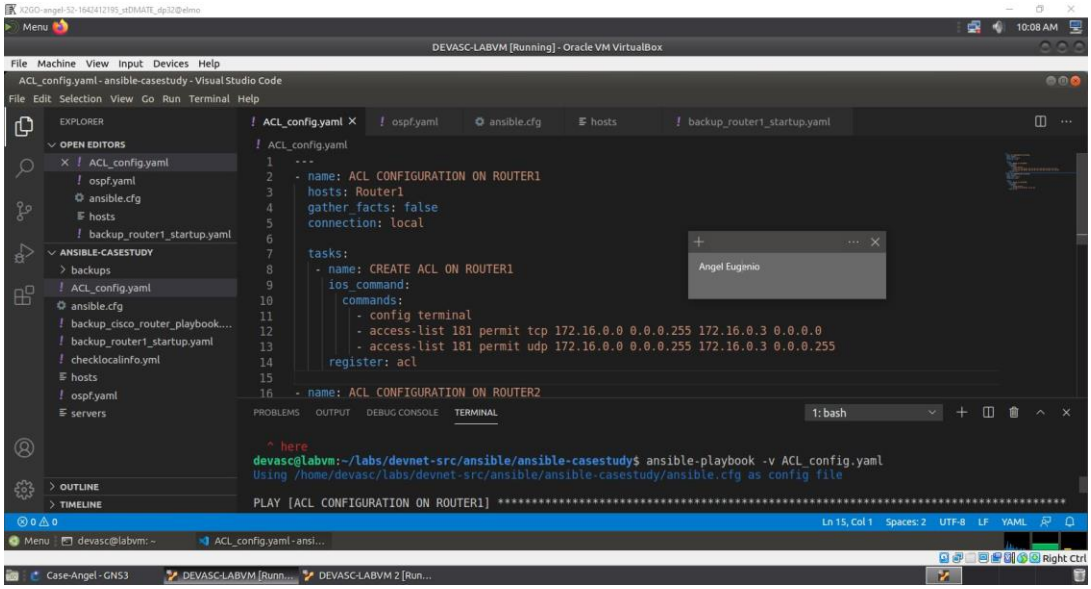
Access Control List or ACL is implemented to establish a set of rules that helps control a network regarding the traffic. It is like a gatekeeper that regulates the incoming and outgoing data packets based from the given rule.

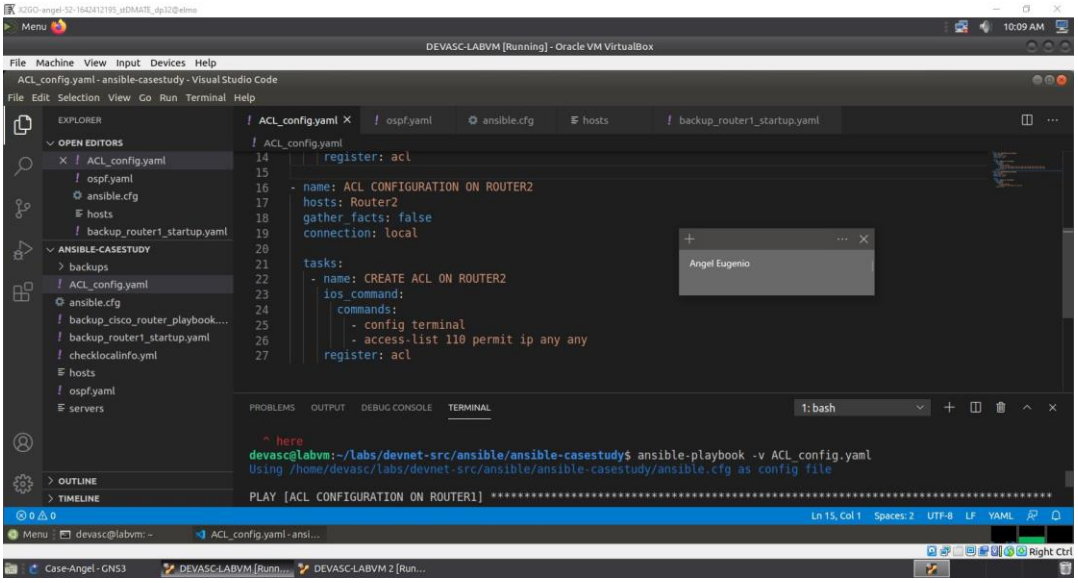
Step 1: Create Ansible Playbook for ACL Configuration

Create another YAML file that would enable ACL on both routers. Copy the following code.

```
---

- name: ACL CONFIGURATION ON ROUTER1

  hosts: Router1

  gather_facts: false

  connection: local


  tasks:
```

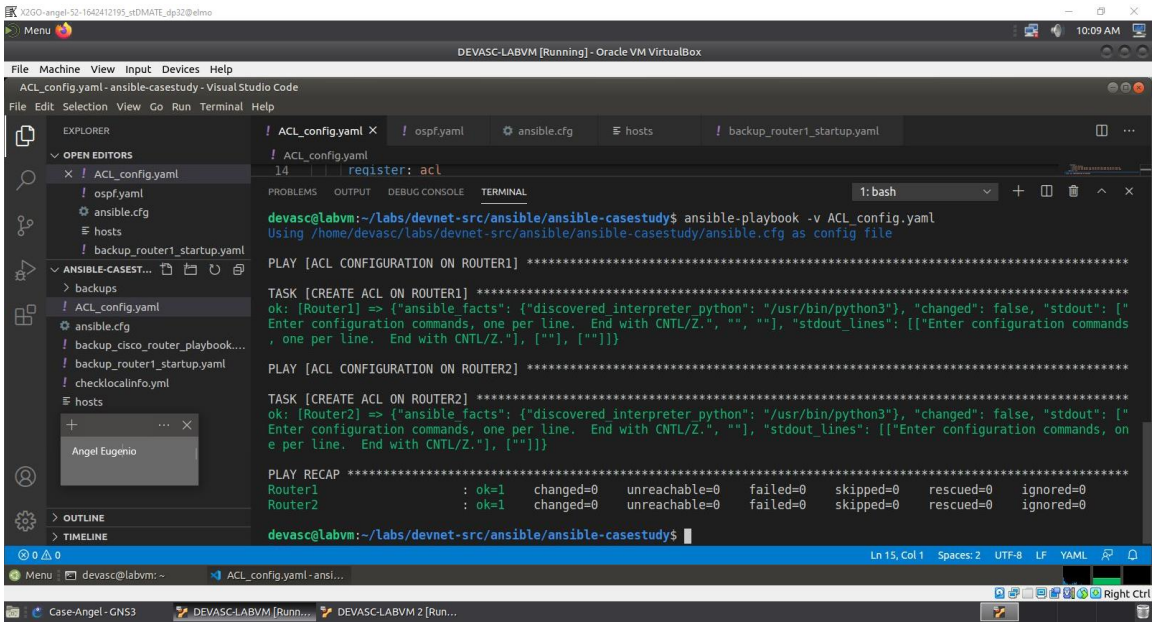
```
  - name: CREATE ACL ON ROUTER1
    ios_command:
      commands:
        - config terminal
        - access-list 181 permit tcp 172.16.0.0 0.0.0.255 172.16.0.3 0.0.0.0
        - access-list 181 permit udp 172.16.0.0 0.0.0.255 172.16.0.3 0.0.0.255
    register: acl

- name: ACL CONFIGURATION ON ROUTER2
  hosts: Router2
  gather_facts: false
  connection: local

  tasks:
   - name: CREATE ACL ON ROUTER2
     ios_command:
       commands:
         - config terminal
         - access-list 110 permit ip any any
     register: acl
```





Step 2: Run the Ansible Playbook for ACL Configuration

Run the created YAML file to apply the ACL configuration. Use the **ansible-playbook -v [filename]** command.



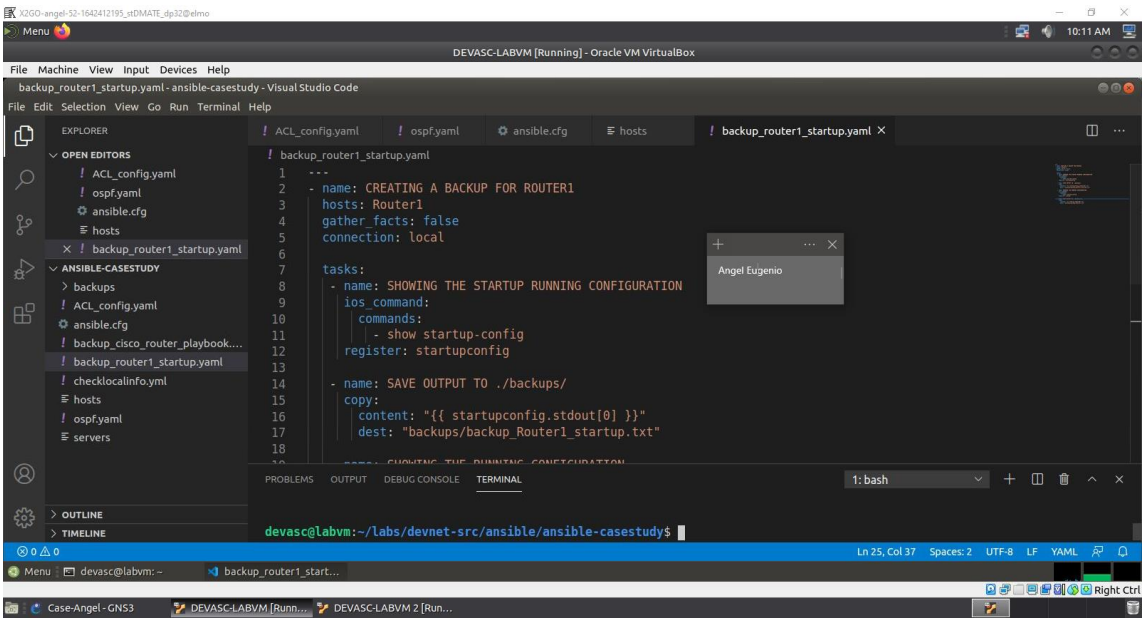
PART 6: BACKUP ROUTER 1

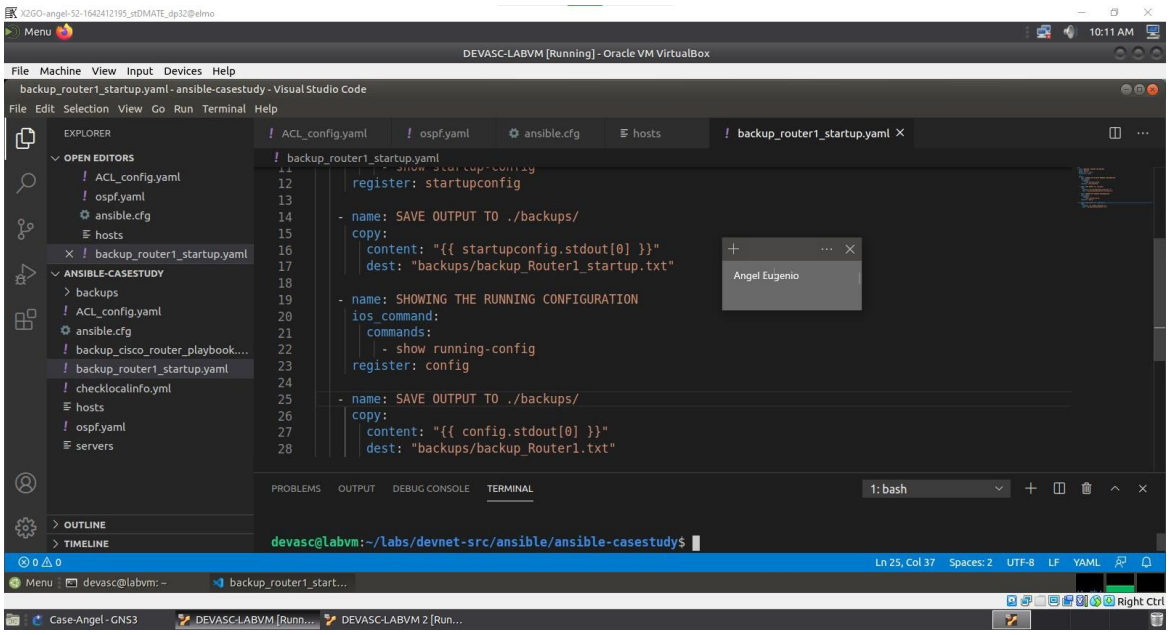
Step 1: Create Ansible Playbook for Saving the Configuration of Router 1

Create another YAML file that would enable saving the startup running configuration and running configuration of Router 1. Copy the following code.

```
---
- name: CREATING A BACKUP FOR ROUTER1
  hosts: Router1
  gather_facts: false
  connection: local


  tasks:
   - name: SHOWING THE STARTUP RUNNING CONFIGURATION
     ios_command:
       commands:
         - show startup-config
     register: startupconfig
```

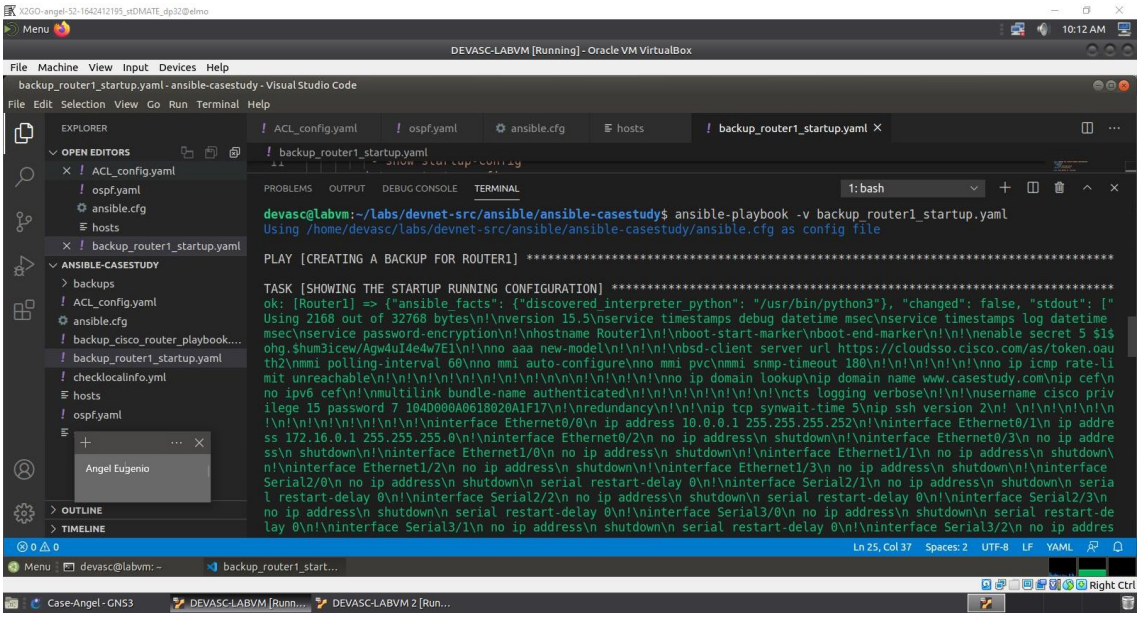
```
- name: SAVE OUTPUT TO ./backups/
  copy:
    content: "{{ startupconfig.stdout[0] }}"
    dest: "backups/backup_Router1_startup.txt"

- name: SHOWING THE RUNNING CONFIGURATION
  ios_command:
    commands:
      - show running-config
  register: config

- name: SAVE OUTPUT TO ./backups/
  copy:
    content: "{{ config.stdout[0] }}"
    dest: "backups/backup_Router1.txt"
```

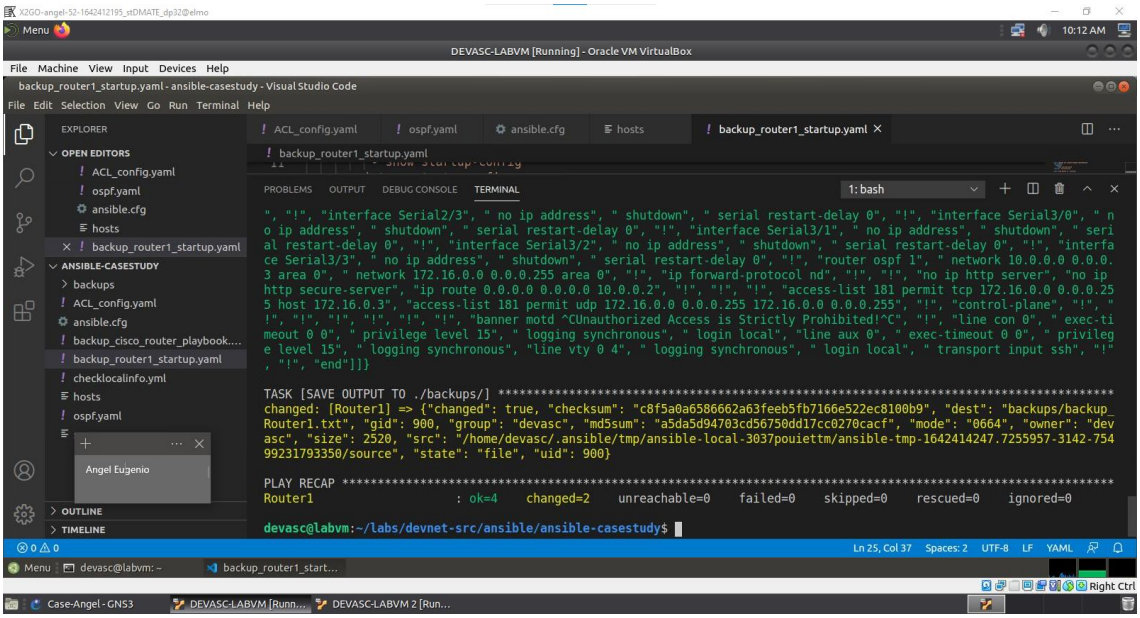




Step 2: Run the Ansible Playbook for Backup of Router1 Configuration

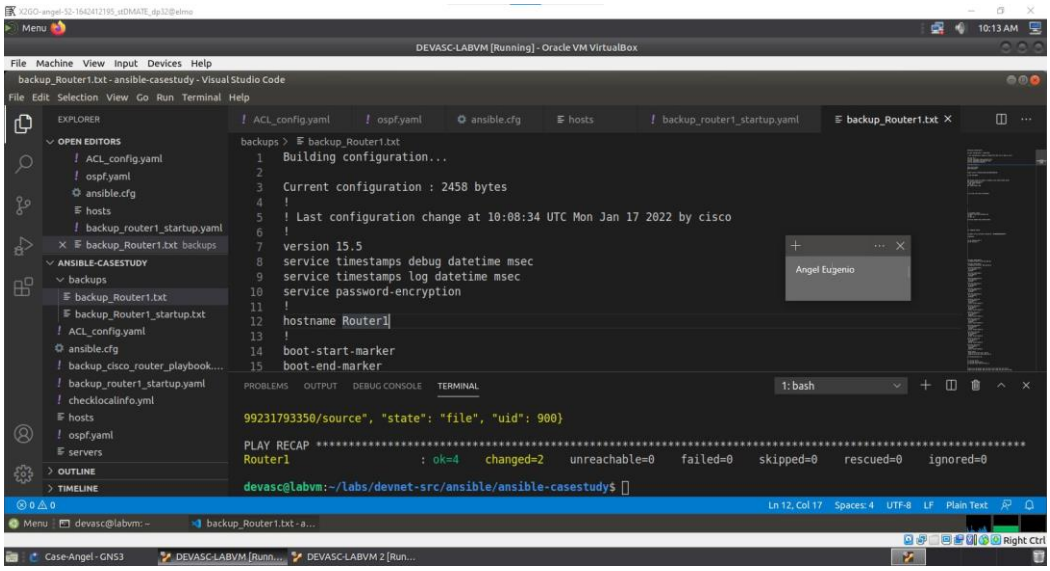
Run the created YAML file to save Router1 Configuration. Use the **ansible-playbook -v [filename]** command.





Step 3: Check the Save Output for Backup

Select the created backups folder and open the **backup_Router1.txt** and **backup_Router1_startup.txt**.



The content of backup_Router1.txt:

Building configuration...

Current configuration : 2458 bytes

```
!
! Last configuration change at 10:08:34 UTC Mon Jan 17 2022 by cisco
!
version 15.5
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname Router1
!
boot-start-marker
boot-end-marker
!
!
enable secret 5 $1$ohg.$hum3icew/Agw4uI4e4w7E1
!
no aaa new-model
!
!
!
bsd-client server url https://cloudsso.cisco.com/as/token.oauth2
mmi polling-interval 60
no mmi auto-configure
no mmi pvc
mmi snmp-timeout 180
!
!
!
```

!

!

no ip icmp rate-limit unreachable

!

!

!

!

!

!

!

!

!

!

!

!

no ip domain lookup

ip domain name www.casestudy.com

ip cef

no ipv6 cef

!

multilink bundle-name authenticated

!

!

!

!

!

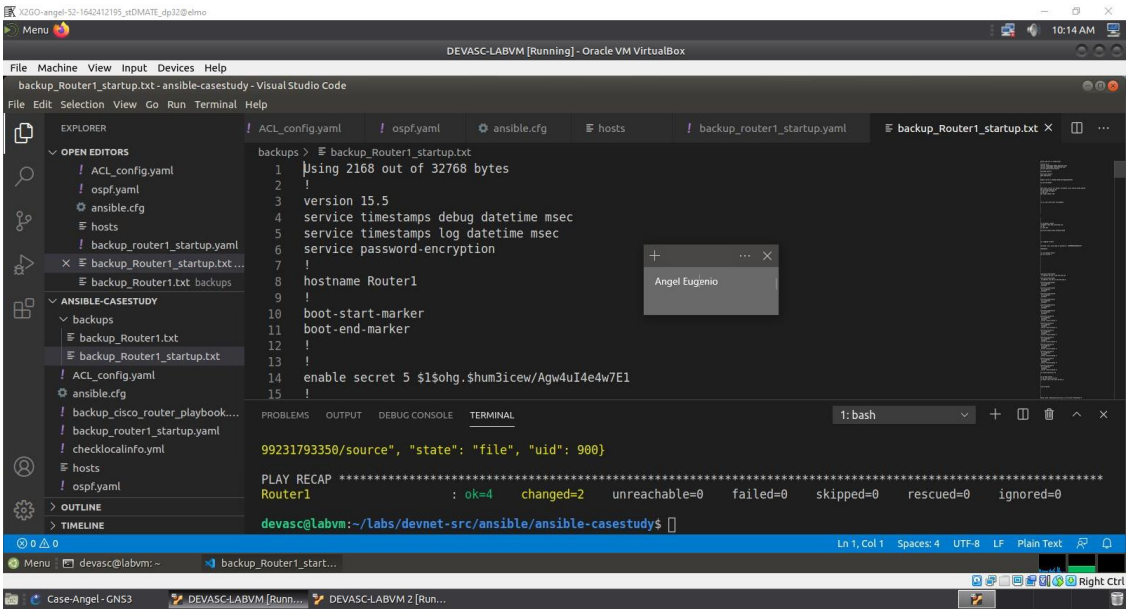
```
!
!
cts logging verbose
!
!
username cisco privilege 15 password 7 104D000A0618020A1F17
!
redundancy
!
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
interface Ethernet0/0
 ip address 10.0.0.1 255.255.255.252
!
```

```
interface Ethernet0/1
 ip address 172.16.0.1 255.255.255.0
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Ethernet1/1
 no ip address
 shutdown
!
interface Ethernet1/2
 no ip address
 shutdown
!
interface Ethernet1/3
 no ip address
 shutdown
!
interface Serial2/0
```

```
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/3
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/1
 no ip address
 shutdown
 serial restart-delay 0
```

```
!
interface Serial3/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/3
 no ip address
 shutdown
 serial restart-delay 0
!
router ospf 1
 network 10.0.0.0 0.0.0.3 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 10.0.0.2
!
!
!
access-list 181 permit tcp 172.16.0.0 0.0.0.255 host 172.16.0.3
access-list 181 permit udp 172.16.0.0 0.0.0.255 172.16.0.0 0.0.0.255
!
control-plane
```

```
!
!
!
!
!
!
!
banner motd ^CUnauthorized Access is Strictly Prohibited!^C
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
 login local
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 logging synchronous
 login local
 transport input ssh
!
!
end
```



The content of backup_Router1_startup.txt:

Using 2168 out of 32768 bytes

!

version 15.5

service timestamps debug datetime msec

service timestamps log datetime msec

service password-encryption

!

hostname Router1

!

boot-start-marker

boot-end-marker

!

!

enable secret 5 1ohg.$hum3icew/Agw4uI4e4w7E1

!

no aaa new-model

!

!

!

bsd-client server url https://cloudsso.cisco.com/as/token.oauth2

mmi polling-interval 60

no mmi auto-configure

no mmi pvc

mmi snmp-timeout 180

!

!

!

!

!

no ip icmp rate-limit unreachable

!

!

!

!

!

!

!

!

!

!

!

!

!

```
no ip domain lookup
ip domain name www.casestudy.com
ip cef
no ipv6 cef
!
multilink bundle-name authenticated
!
!
!
!
!
!
!
cts logging verbose
!
!
username cisco privilege 15 password 7 104D000A0618020A1F17
!
redundancy
!
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
!
```

```
!
!
!
!
!
!
!
!
interface Ethernet0/0
 ip address 10.0.0.1 255.255.255.252
!
interface Ethernet0/1
 ip address 172.16.0.1 255.255.255.0
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Ethernet1/1
 no ip address
```

```
 shutdown
!
interface Ethernet1/2
 no ip address
 shutdown
!
interface Ethernet1/3
 no ip address
 shutdown
!
interface Serial2/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/3
 no ip address
 shutdown
```

```
 serial restart-delay 0
!
interface Serial3/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/3
 no ip address
 shutdown
 serial restart-delay 0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
ip route 0.0.0.0 0.0.0.0 10.0.0.2
```

```
!
!
!
!
control-plane
!
!
!
!
!
!
!
banner motd ^CUnauthorized Access is Strictly Prohibited!^C
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
 login local
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 logging synchronous
 login local
 transport input ssh
!
```

!

end

PART 7: PYATS TESTING

The test declaration syntax for pyATS is created on popular Python unit-testing frameworks such as pytest. Here, it supports basic testing statements, like the assertion that a variable has a given value, and along with explicitly providing results via specific APIs.

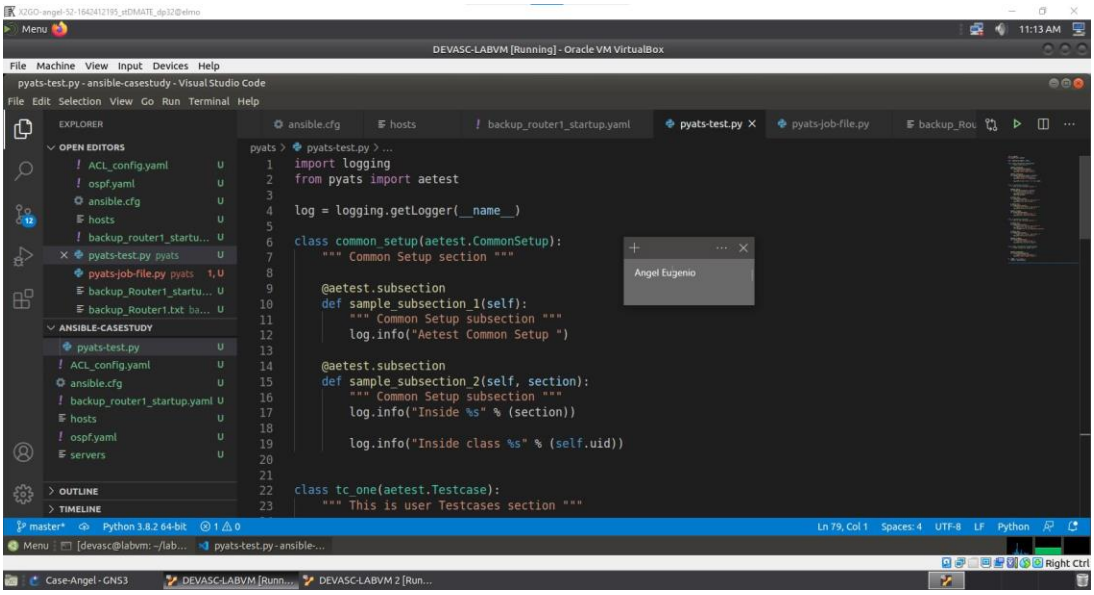
Step 1: Create a pyATS script where PyATS tests are declared.

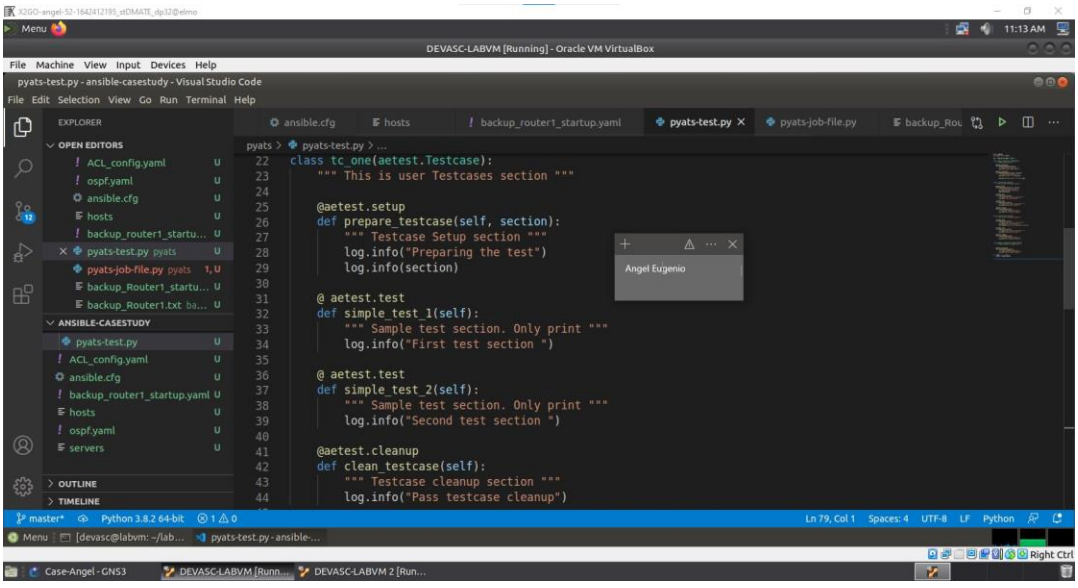
Create a new folder named as pyats. Inside the folder, create a python script which is titled as pyats-test.py. Copy the following code.

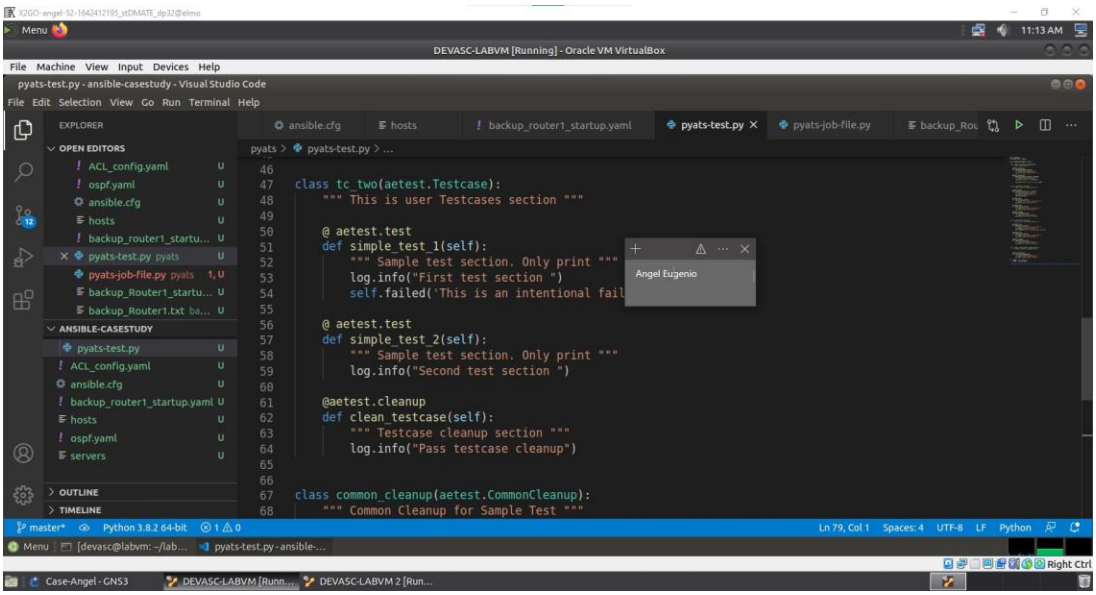
```
import logging
from pyats import aetest

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")
```

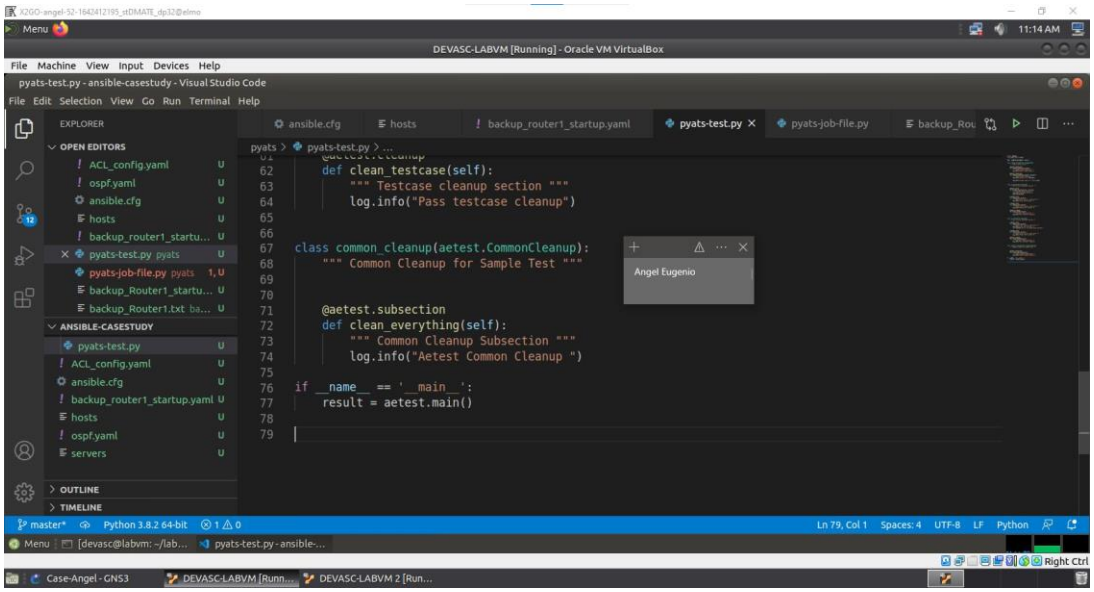
```
        log.info(section)

    @ 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()
```







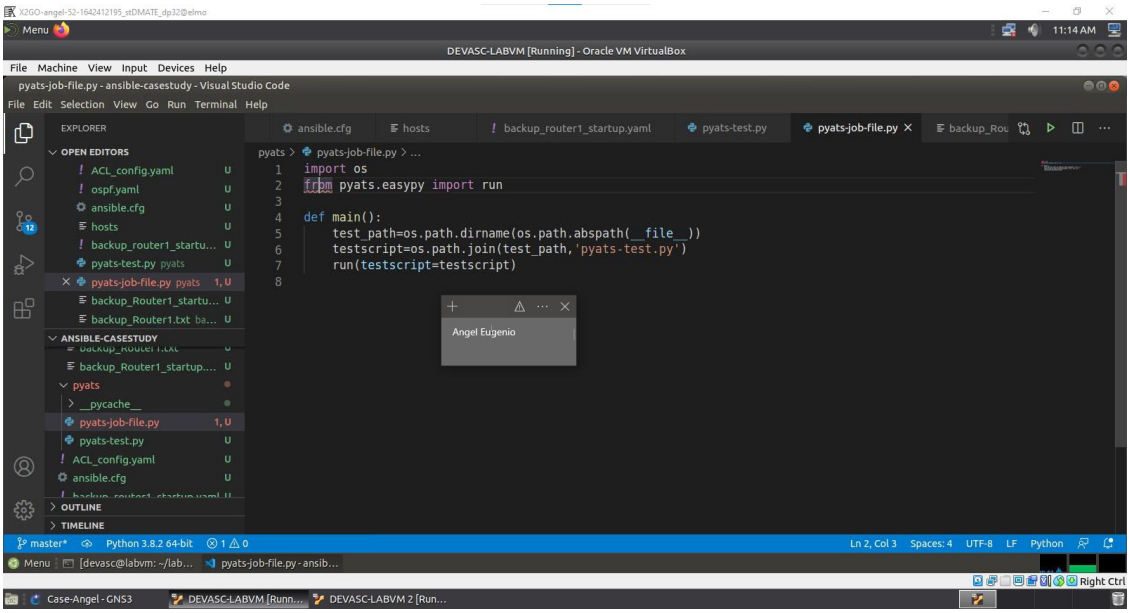


Step 2: Create a pyATS job for the compilation of the pyATS script and for running the script.

Create another python script unser the pyats folder. Name the file as pyats-job-file.py and copy the following code.

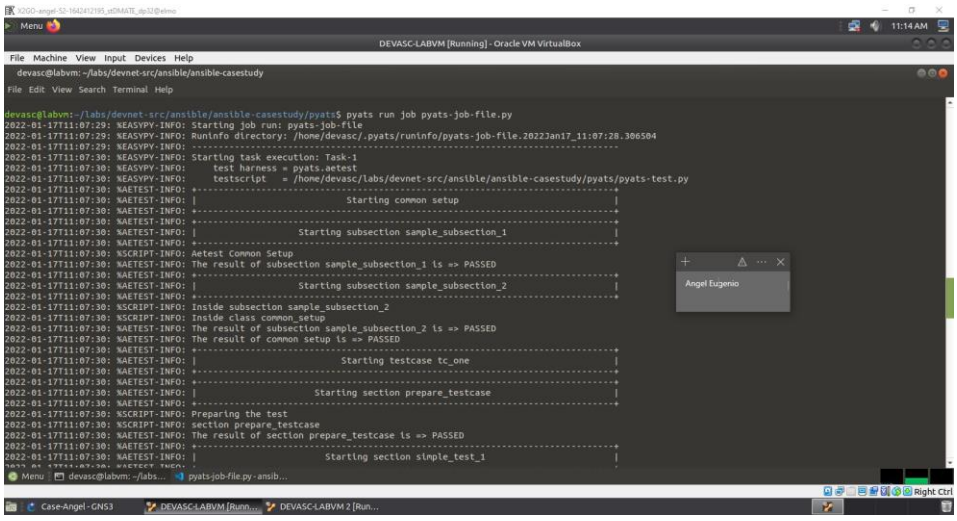
```
import os

from pyats.easypy import run

def main():

    test_path=os.path.dirname(os.path.abspath(__file__))
```

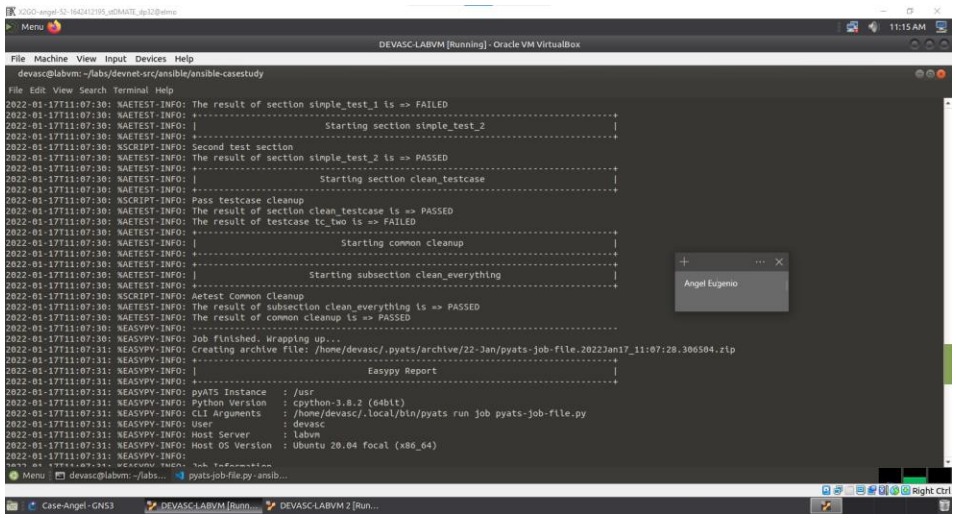
```
testscript=os.path.join(test_path,'pyats-test.py')
run(testscript=testscript)
```

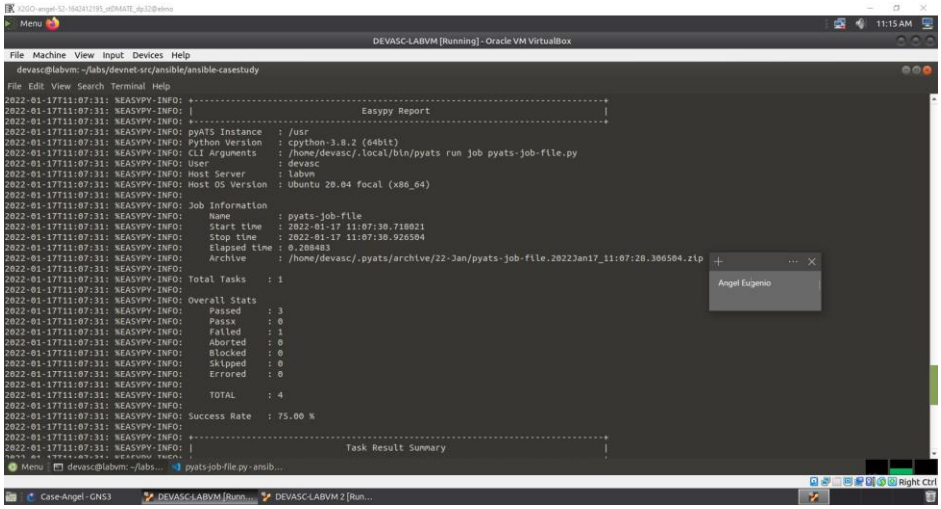


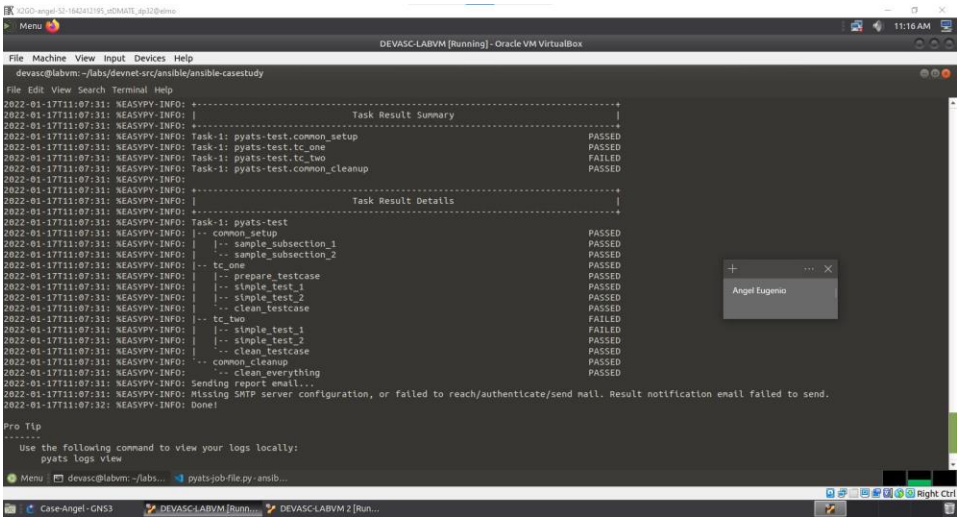
Step 3: Run the pyATS job.

Using the pyATS job and script files, run pyATS manually to invoke the basic test case. This will help verify if the pyATS job and script files work properly. Use the **pyats run job [filename]** command.




DEVASC-LABVM [Running] - Oracle VM VirtualBox
devasc@labvm: ~/labs/devnet-src/ansible/ansible-casestudy
2022-01-17T11:07:30: %AETEST-INFO: The result of section simple_test_1 is => FAILED
2022-01-17T11:07:30: %AETEST-INFO: | Starting section simple_test_2 |
2022-01-17T11:07:30: %SCRIPT-INFO: Second test section
2022-01-17T11:07:30: %AETEST-INFO: The result of section simple_test_2 is => PASSED
2022-01-17T11:07:30: %AETEST-INFO: | Starting section clean_testcase |
2022-01-17T11:07:30: %SCRIPT-INFO: Pass testcase cleanup
2022-01-17T11:07:30: %AETEST-INFO: The result of section clean_testcase is => PASSED
2022-01-17T11:07:30: %AETEST-INFO: The result of testcase tc_two is => FAILED
2022-01-17T11:07:30: %AETEST-INFO: | Starting common cleanup |
2022-01-17T11:07:30: %AETEST-INFO: | Starting subsection clean_everything |
2022-01-17T11:07:30: %SCRIPT-INFO: Aetest Common Cleanup
2022-01-17T11:07:30: %AETEST-INFO: The result of subsection clean_everything is => PASSED
2022-01-17T11:07:30: %AETEST-INFO: The result of common cleanup is => PASSED
2022-01-17T11:07:30: %EASYPY-INFO: Job finished. Wrapping up...
2022-01-17T11:07:31: %EASYPY-INFO: Creating archive file: /home/devasc/.pyats/archive/22-Jan/pyats-job-file.2022Jan17_11:07:28.306504.zip
2022-01-17T11:07:31: %EASYPY-INFO: | Easypy Report |
2022-01-17T11:07:31: %EASYPY-INFO: pyATS Instance : /usr
2022-01-17T11:07:31: %EASYPY-INFO: Python Version : cpython-3.8.2 (64bit)
2022-01-17T11:07:31: %EASYPY-INFO: CLI Arguments : /home/devasc/.local/bin/pyats run job pyats-job-file.py
2022-01-17T11:07:31: %EASYPY-INFO: User : devasc
2022-01-17T11:07:31: %EASYPY-INFO: Host Server : labvm
2022-01-17T11:07:31: %EASYPY-INFO: Host OS Version : Ubuntu 20.04 focal (x86_64)
Angel Eugenio


DEVASC-LABVM [Running] - Oracle VM VirtualBox
devasc@labvm: ~/labs/devnet-src/ansible/ansible-casestudy
2022-01-17T11:07:31: %EASYPY-INFO: | Easypy Report |
2022-01-17T11:07:31: %EASYPY-INFO: pyATS Instance : /usr
2022-01-17T11:07:31: %EASYPY-INFO: Python Version : cpython-3.8.2 (64bit)
2022-01-17T11:07:31: %EASYPY-INFO: CLI Arguments : /home/devasc/.local/bin/pyats run job pyats-job-file.py
2022-01-17T11:07:31: %EASYPY-INFO: User : devasc
2022-01-17T11:07:31: %EASYPY-INFO: Host Server : labvm
2022-01-17T11:07:31: %EASYPY-INFO: Host OS Version : Ubuntu 20.04 focal (x86_64)
2022-01-17T11:07:31: %EASYPY-INFO: Job Information
2022-01-17T11:07:31: %EASYPY-INFO: Name : pyats-job-file
2022-01-17T11:07:31: %EASYPY-INFO: Start time : 2022-01-17 11:07:30.718021
2022-01-17T11:07:31: %EASYPY-INFO: Stop time : 2022-01-17 11:07:30.926504
2022-01-17T11:07:31: %EASYPY-INFO: Elapsed time : 0.208483
2022-01-17T11:07:31: %EASYPY-INFO: Archive : /home/devasc/.pyats/archive/22-Jan/pyats-job-file.2022Jan17_11:07:28.306504.zip
2022-01-17T11:07:31: %EASYPY-INFO: Total Tasks : 1
2022-01-17T11:07:31: %EASYPY-INFO: Overall Stats
2022-01-17T11:07:31: %EASYPY-INFO: Passed : 3
2022-01-17T11:07:31: %EASYPY-INFO: Passx : 0
2022-01-17T11:07:31: %EASYPY-INFO: Failed : 1
2022-01-17T11:07:31: %EASYPY-INFO: Aborted : 0
2022-01-17T11:07:31: %EASYPY-INFO: Blocked : 0
2022-01-17T11:07:31: %EASYPY-INFO: Skipped : 0
2022-01-17T11:07:31: %EASYPY-INFO: Errored : 0
2022-01-17T11:07:31: %EASYPY-INFO: TOTAL : 4
2022-01-17T11:07:31: %EASYPY-INFO: Success Rate : 75.00 %
2022-01-17T11:07:31: %EASYPY-INFO: | Task Result Summary |
Angel Eugenio


DEVASC-LABVM [Running] - Oracle VM VirtualBox
devasc@labvm: ~/labs/devnet-src/ansible/ansible-casestudy
2022-01-17T11:07:31: %EASYPY-INFO: | Task Result Summary |
2022-01-17T11:07:31: %EASYPY-INFO: Task-1: pyats-test.common_setup PASSED
2022-01-17T11:07:31: %EASYPY-INFO: Task-1: pyats-test.tc_one PASSED
2022-01-17T11:07:31: %EASYPY-INFO: Task-1: pyats-test.tc_two FAILED
2022-01-17T11:07:31: %EASYPY-INFO: Task-1: pyats-test.common_cleanup PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | Task Result Details |
2022-01-17T11:07:31: %EASYPY-INFO: Task-1: pyats-test
2022-01-17T11:07:31: %EASYPY-INFO: |-- common_setup PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- sample_subsection_1 PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | `-- sample_subsection_2 PASSED
2022-01-17T11:07:31: %EASYPY-INFO: |-- tc_one PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- prepare_testcase PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- simple_test_1 PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- simple_test_2 PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | `-- clean_testcase PASSED
2022-01-17T11:07:31: %EASYPY-INFO: |-- tc_two FAILED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- simple_test_1 FAILED
2022-01-17T11:07:31: %EASYPY-INFO: | |-- simple_test_2 PASSED
2022-01-17T11:07:31: %EASYPY-INFO: | `-- clean_testcase PASSED
2022-01-17T11:07:31: %EASYPY-INFO: `-- common_cleanup PASSED
2022-01-17T11:07:31: %EASYPY-INFO: `-- clean_everything PASSED
2022-01-17T11:07:31: %EASYPY-INFO: Sending report email...
2022-01-17T11:07:31: %EASYPY-INFO: Missing SMTP server configuration, or failed to reach/authenticate/send mail. Result notification email failed to send.
2022-01-17T11:07:32: %EASYPY-INFO: Done!
Pro Tip
Use the following command to view your logs locally:
pyats logs view
Angel Eugenio

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