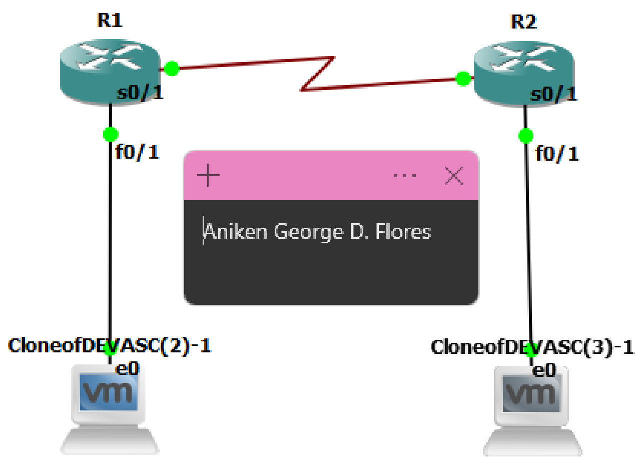
Name: Aniken George Flores **Date: 1/17/2022**

Final Case Study | Network Automation and Programmability



Network Topology

Network Addressing Table

Device	Interface	IP Address	Subnet mask
R1	F0/1	192.168.10.62	255.255.255.192
	S0/1	11.25.1.2	255.255.255.252
R2	F0/1	192.168.20.62	255.255.255.192
	S0/1	11.25.1.3	255.255.255.252
PC1	E0	192.168.10.61	255.255.255.252
PC2	E0	192.168.20.61	255.255.255.252

Required Resources

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

Instructions:

Part 1: Launch the GNS3

Step 1: Create new project

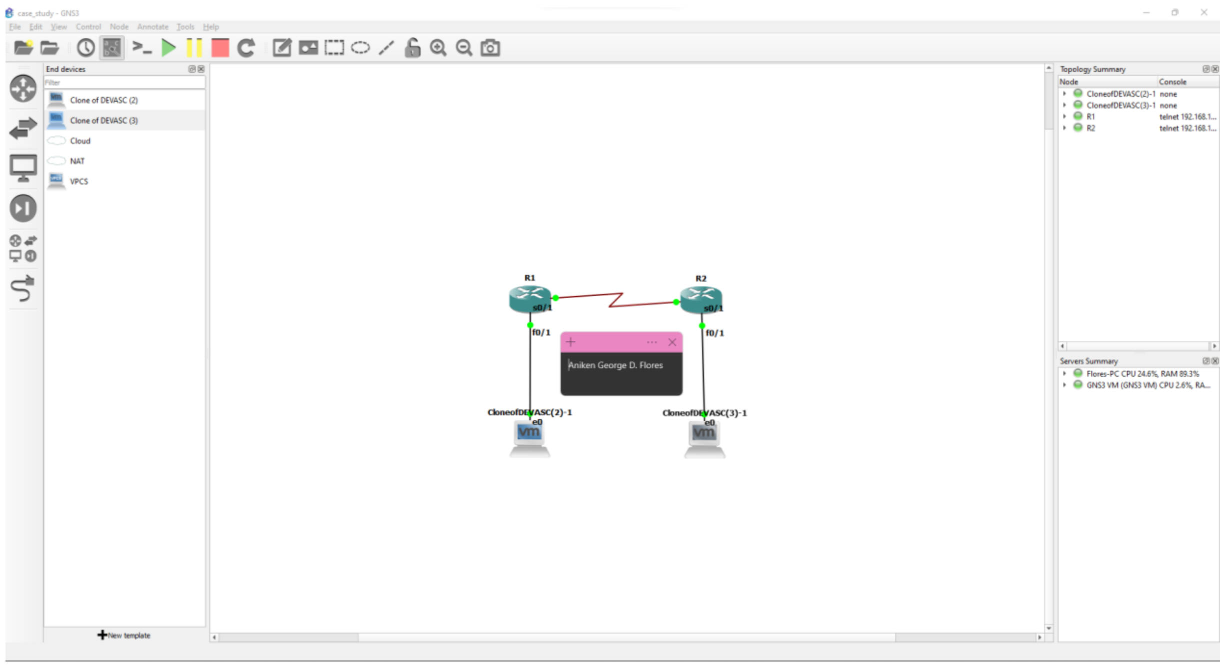
To create a new file, click on the file tab on the upper left portion of the window then click create new blank project.

Step 2: Install the CISCO IOS image for the router

Download the router image you need for this activity.

Step 3: Create the topology

Follow the topology shown in the image and connect each device.



Node	Console
CloneofDEVASC(2)-1	none
CloneofDEVASC(3)-1	none
R1	telnet 192.168.1...
R2	telnet 192.168.1...

Flores-PC CPU 24.6%, RAM 89.3%
GNS3 VM (GNS3 VM) CPU 2.6%, RA...

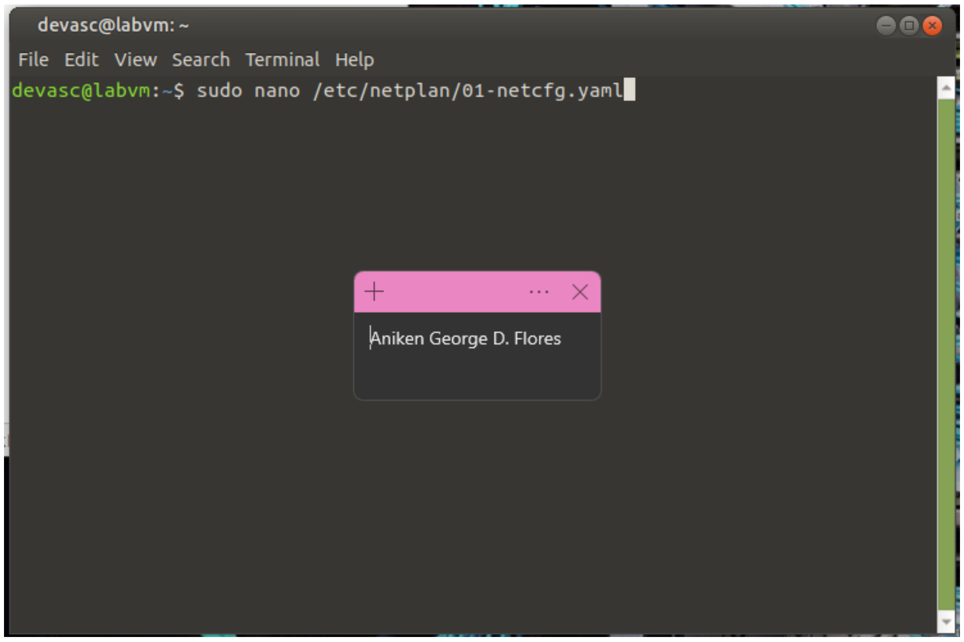
Step 4: Apply basic configuration to the routers

Apply the basic configurations to the routers by following the network addressing table.

Step 5: configure the netplan of both pc

Issue these commands in the terminal.

```
devasc@labvm:~$ sudo nano /etc/netplan/01-netcfg.yaml
```

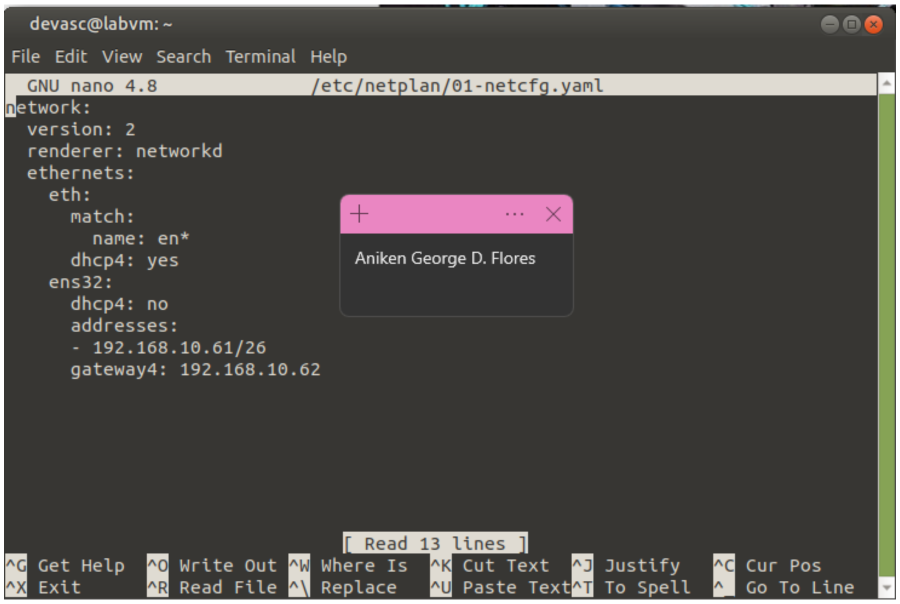


```
devasc@labvm:~$ sudo nano /etc/netplan/01-netcfg.yaml
```

Aniken George D. Flores

Once you are in the nano type this:

```
network:
  version: 2
  renderer: networkd
  ethernets:
   eth:
       match:
           name: en*
       dhcp4: yes
   ens32:
     dhcp4: no
     addresses:
     - [IP Address here]/[Suffix]
     gateway4: [Default gateway]
```



```
devasc@labvm: ~
File  Edit  View  Search  Terminal  Help
  GNU nano 4.8                 /etc/netplan/01-netcfg.yaml
network:
  version: 2
  renderer: networkd
  ethernets:
    eth:
      match:
        name: en*
      dhcp4: yes
    ens32:
      dhcp4: no
      addresses:
      - 192.168.10.61/26
      gateway4: 192.168.10.62

                                [ Read 13 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

Aniken George D. Flores

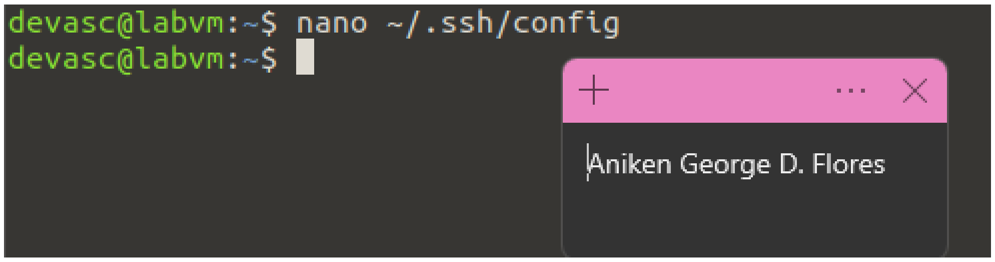
After configuring type in “sudo netlan apply” command to save the configurations. Do this for both PC’s.

Step 6: Access the router via SSH

To access the routers by ssh let us add some configurations to the ssh config.

Type these commands to the terminal:

$ nano ~/.ssh/config



```
devasc@labvm:~$ nano ~/.ssh/config
devasc@labvm:~$ 
```

Aniken George D. Flores

In the config file type this:

Host *

Port 22

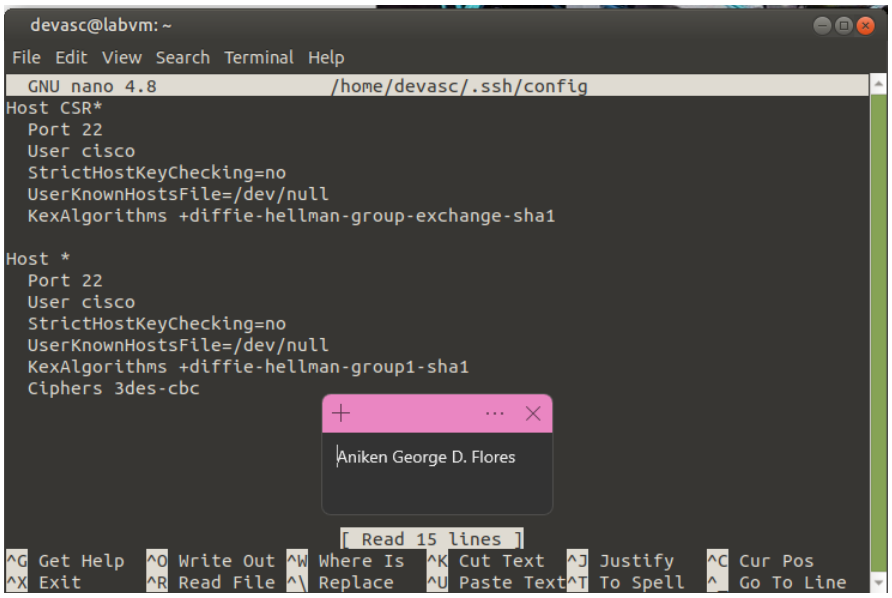
User cisco

StrictHostKeyChecking=no

UserKnownHostsFile=/dev/null

KexAlgorithms +diffie-hellman-group1-sha1

Ciphers 3des-cbc



devasc@labvm: ~

File Edit View Search Terminal Help

```
  GNU nano 4.8                  /home/devasc/.ssh/config
Host CSR*
  Port 22
  User cisco
  StrictHostKeyChecking=no
  UserKnownHostsFile=/dev/null
  KexAlgorithms +diffie-hellman-group-exchange-sha1

Host *
  Port 22
  User cisco
  StrictHostKeyChecking=no
  UserKnownHostsFile=/dev/null
  KexAlgorithms +diffie-hellman-group1-sha1
  Ciphers 3des-cbc

                           [ Read 15 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

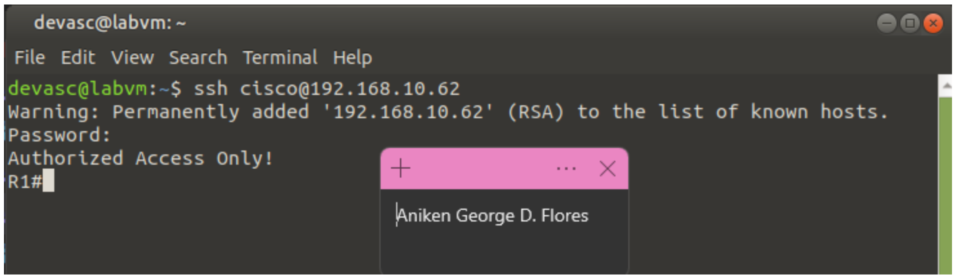
Aniken George D. Flores

Then hit ctrl+x to exit.

Step 7: Accessing the SSH

In the terminal make sure you can access the routers by typing this commands.

ssh cisco@192.168.10.62



devasc@labvm: ~

File Edit View Search Terminal Help

```
devasc@labvm:~$ ssh cisco@192.168.10.62
Warning: Permanently added '192.168.10.62' (RSA) to the list of known hosts.
Password:
Authorized Access Only!
R1#
```

Aniken George D. Flores

If you can access it you are now ready for the next part.

Part 2: Applying test automation

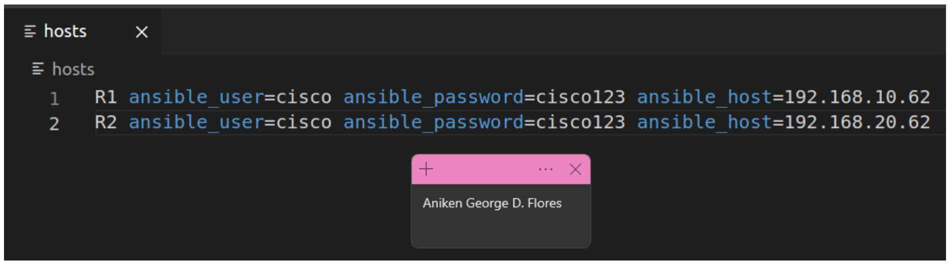
Step 1: Open Visual Studio Code and add a new directory.

Step 2: Create Hosts file

In this file type in this codes:

R1 ansible_user=cisco ansible_password=cisco123 ansible_host=192.168.10.62

R2 ansible_user=cisco ansible_password=cisco123 ansible_host=192.168.20.62

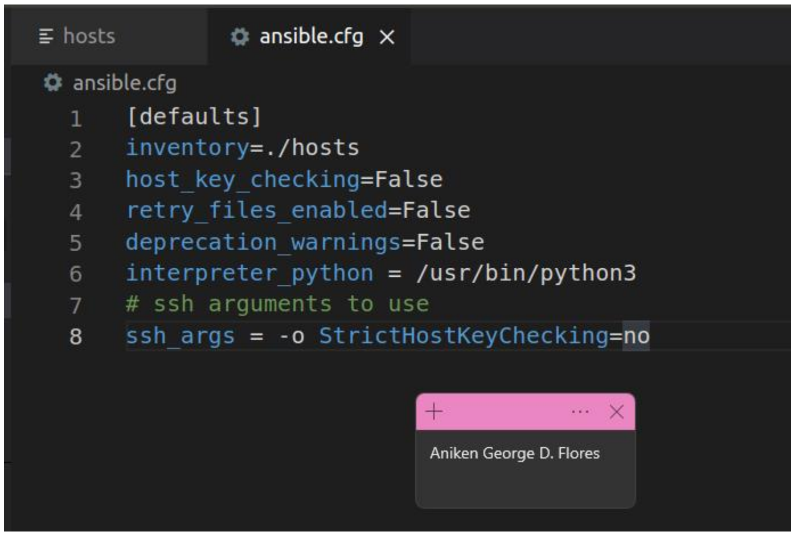


```
R1 ansible_user=cisco ansible_password=cisco123 ansible_host=192.168.10.62
R2 ansible_user=cisco ansible_password=cisco123 ansible_host=192.168.20.62
```

Step 3: Create Ansible configuration file

In this configuration file type in this codes:

```
[defaults]
inventory=./hosts
host_key_checking=False
retry_files_enabled=False
deprecation_warnings=False
interpreter_python = /usr/bin/python3
# ssh arguments to use
ssh_args = -o StrictHostKeyChecking=no
```

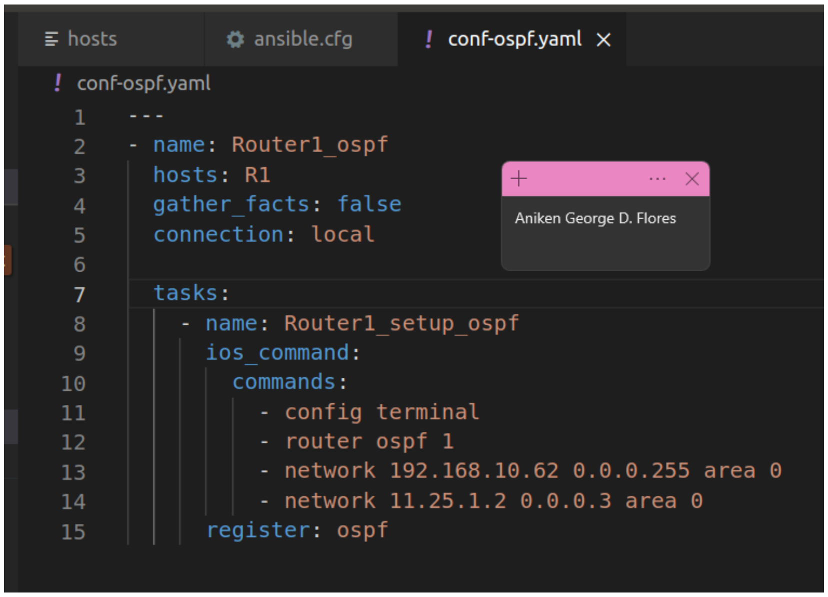


```
[defaults]
inventory=./hosts
host_key_checking=False
retry_files_enabled=False
deprecation_warnings=False
interpreter_python = /usr/bin/python3
# ssh arguments to use
ssh_args = -o StrictHostKeyChecking=no
```

Step 4: Configuring the OSPF

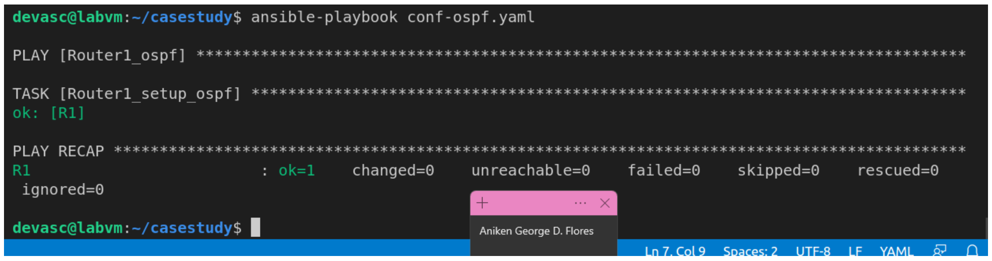
First Create the yaml file and name it conf_ospf.yaml. In this file type in this codes:

```
 ---
- name: Router1_ospf
  hosts: R1
gather_facts: false
connection: local

tasks:
 - name: Router1_setup_ospf
  ios_command:
commands:
   - config terminal
   - router ospf 1
       - network 192.168.10.62 0.0.0.255 area 0
       - network 11.25.1.2 0.0.0.3 area 0
    register: ospf
```



```yaml
---
- name: Router1_ospf
  hosts: R1
  gather_facts: false
  connection: local

  tasks:
    - name: Router1_setup_ospf
      ios_command:
        commands:
          - config terminal
          - router ospf 1
          - network 192.168.10.62 0.0.0.255 area 0
          - network 11.25.1.2 0.0.0.3 area 0
      register: ospf
```

To run type in the command "ansible-playbook conf-ospf.yaml" in the terminal



```
devasc@labvm:~/casestudy$ ansible-playbook conf-ospf.yaml

PLAY [Router1_ospf] ****************************************************************************

TASK [Router1_setup_ospf] **********************************************************************
ok: [R1]

PLAY RECAP *************************************************************************************
R1                         : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
 ignored=0

devasc@labvm:~/casestudy$ 
```

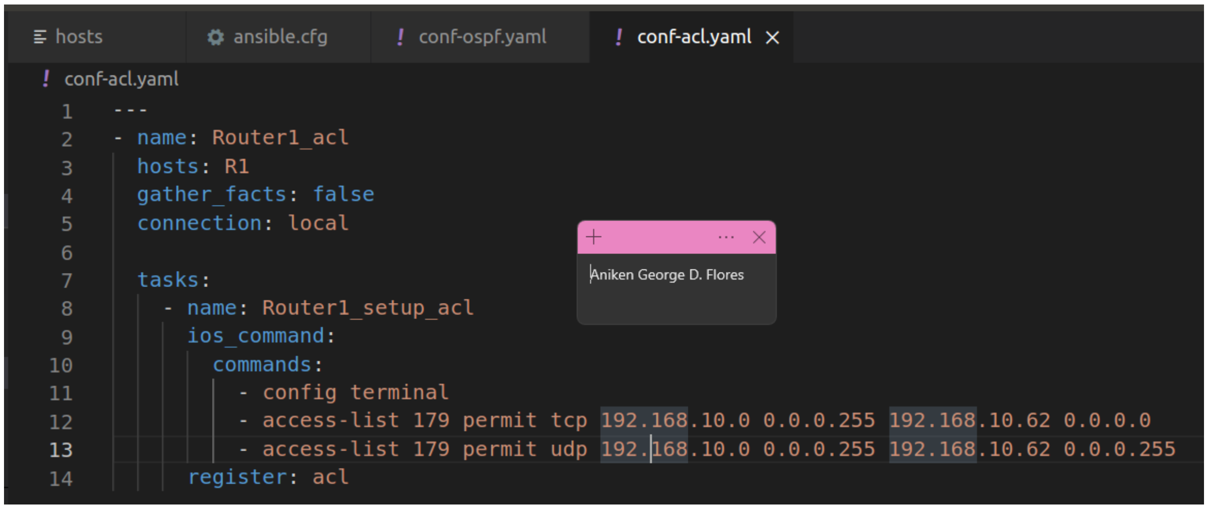
Do this for both PC's just change the addresses and router names.

Step 5: Creating the ACL configuration

Create the ACL configuration file, name it conf-acl.yaml. Enter these commands in the configuration files.

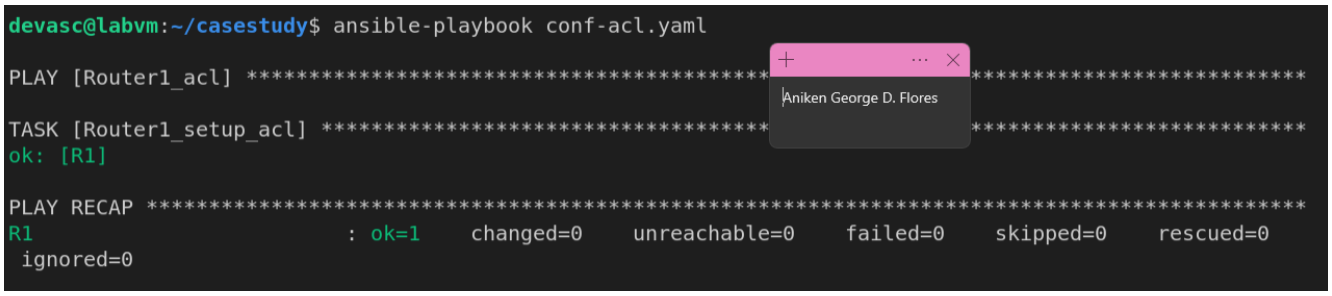
```
 ---
- name: Router1_acl
 hosts: R1
 gather_facts: false
connection: local
```

```
tasks:
  - name: Router1_setup_acl
 ios_command:
 commands:
 - config terminal
  - access-list 179 permit tcp 192.168.10.0 0.0.0.255 192.168.10.62 0.0.0.0
- access-list 179 permit udp 192.168.10.0 0.0.0.255 192.168.10.62 0.0.0.255
register: acl
```



```
conf-acl.yaml
---
- name: Router1_acl
  hosts: R1
  gather_facts: false
  connection: local

  tasks:
    - name: Router1_setup_acl
      ios_command:
        commands:
          - config terminal
          - access-list 179 permit tcp 192.168.10.0 0.0.0.255 192.168.10.62 0.0.0.0
          - access-list 179 permit udp 192.168.10.0 0.0.0.255 192.168.10.62 0.0.0.255
      register: acl
```

To run this, enter the command ansible-playbook conf-acl.yaml.



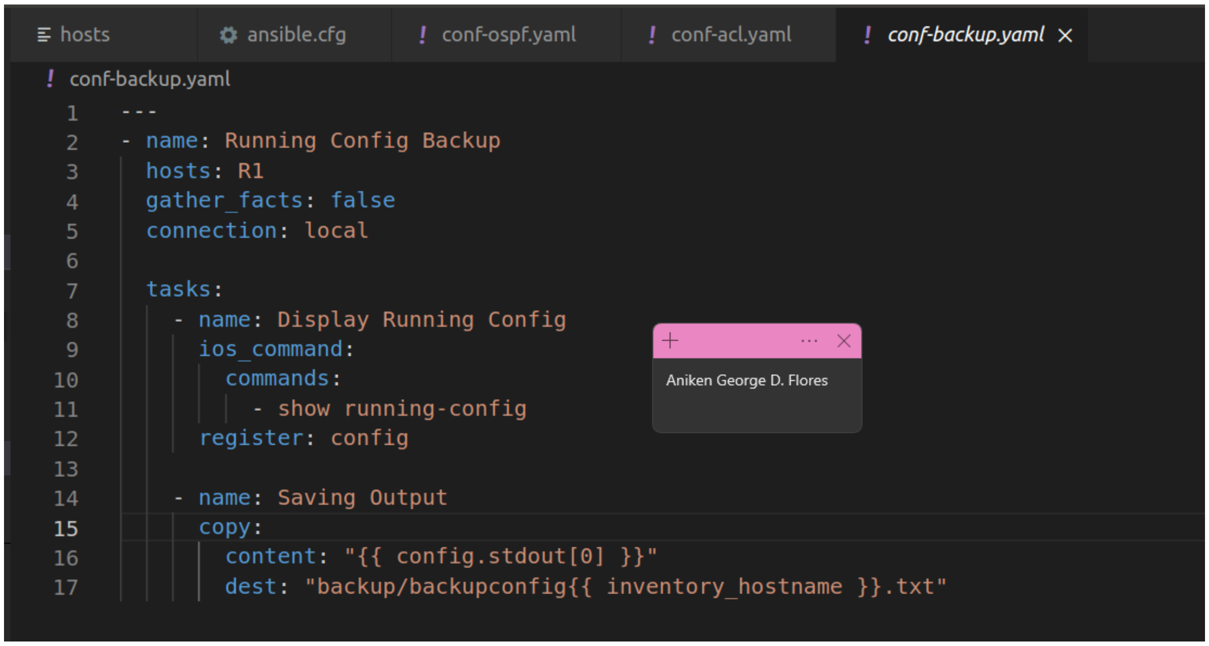
```
devasc@labvm:~/casestudy$ ansible-playbook conf-acl.yaml

PLAY [Router1_acl] ******************************************************************************

TASK [Router1_setup_acl] ************************************************************************
ok: [R1]

PLAY RECAP **************************************************************************************
R1                         : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
 ignored=0
```

Do this for both PC’s.

Step 6: Creating the backup config

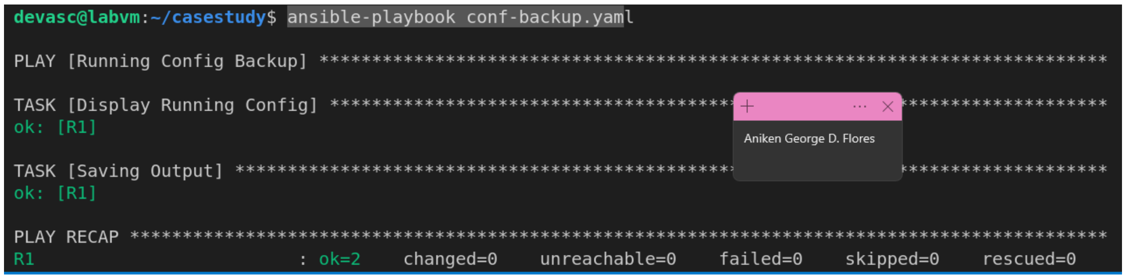
First create a folder named backup. Create a file named conf-backup.yaml and type in these codes.

```
---
- name: Running Config Backup
hosts: R1
 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: "backup/backupconfig{{ inventory_hostname }}.txt"
```



```yaml
conf-backup.yaml
---
- name: Running Config Backup
  hosts: R1
  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: "backup/backupconfig{{ inventory_hostname }}.txt"
```

To run these commands, enter this code in the terminal “ansible-playbook conf-backup.yam”.



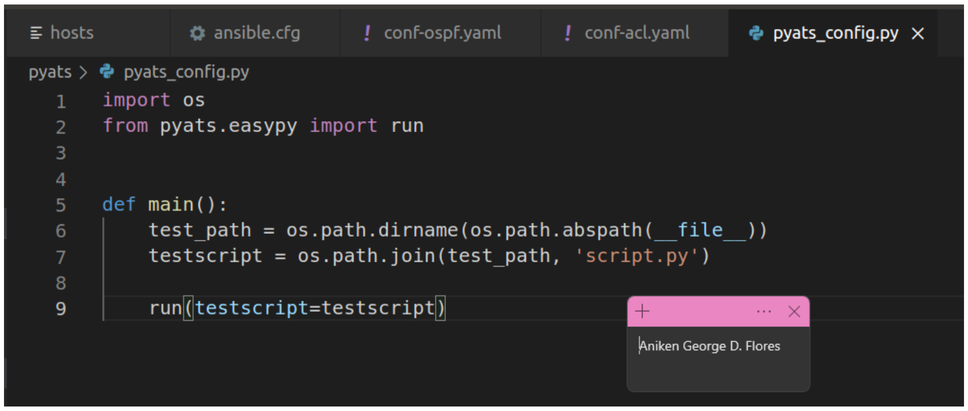
```
devasc@labvm:~/casestudy$ ansible-playbook conf-backup.yaml

PLAY [Running Config Backup] ***************************************************************

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

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

PLAY RECAP *********************************************************************************
R1                         : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
```

Do this for both PC’s.

Step 7: Creating the py file for the pyAts.

Enter the following commands in the pyats_config.py:

```
import os
from pyats.easypy import run


def main():
        test_path = os.path.dirname(os.path.abspath(__file__))
        testscript = os.path.join(test_path, 'script.py')


        run(testscript=testscript)
```

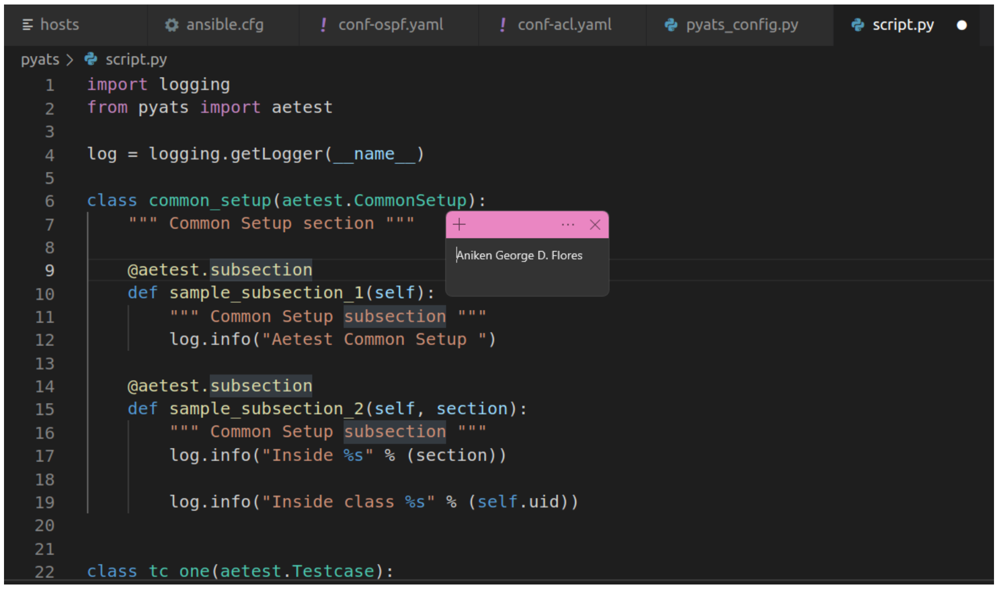


```python
import os
from pyats.easypy import run


def main():
    test_path = os.path.dirname(os.path.abspath(__file__))
    testscript = os.path.join(test_path, 'script.py')

    run(testscript=testscript)
```

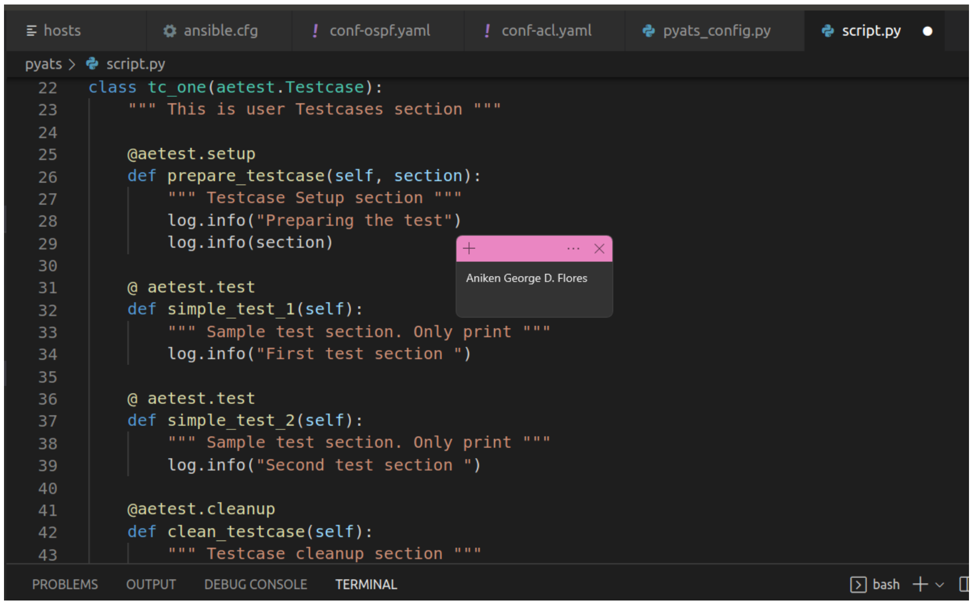
Step 8: Creating the scipt.py

```
                    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))
```

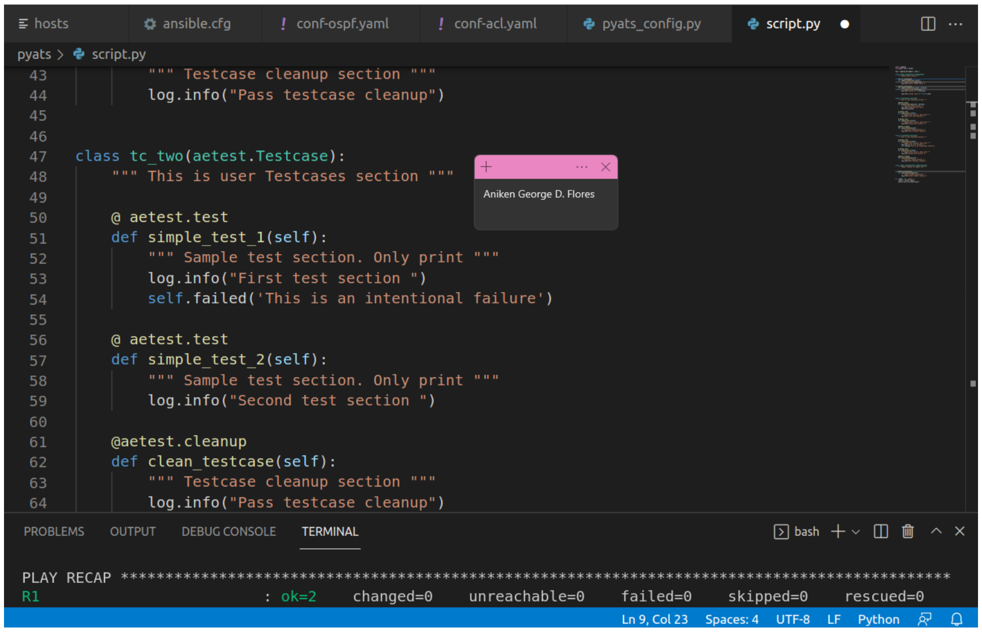
```
class tc_one(aetest.Testcase):
    """ This is user Testcases section """

    @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)
```



```python
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):
```



```python
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 """
```



hosts | ansible.cfg | conf-ospf.yaml | conf-acl.yaml | pyats_config.py | script.py

pyats > script.py

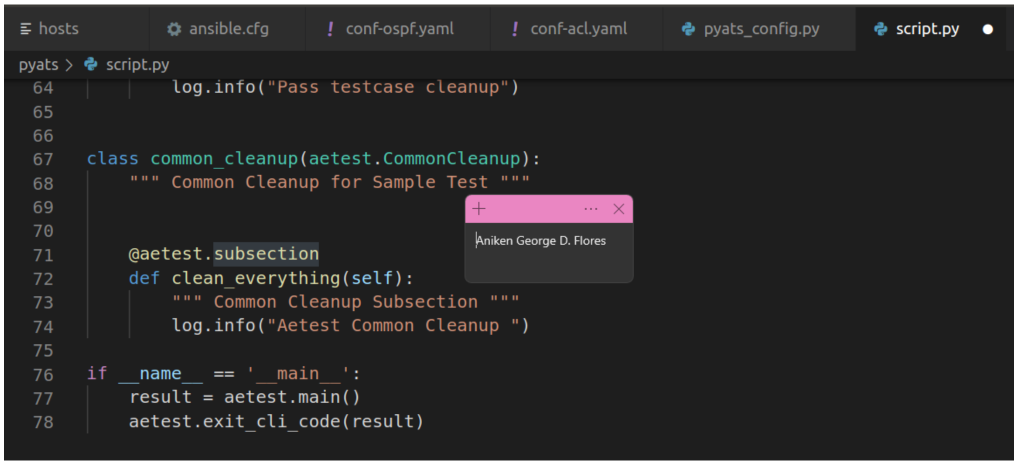
```
        """ Testcase cleanup section """
        log.info("Pass testcase cleanup")


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")
```

Aniken George D. Flores

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PLAY RECAP *****************************************************************************************
R1                          : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
```

Ln 9, Col 23 Spaces: 4 UTF-8 LF Python



hosts | ansible.cfg | conf-ospf.yaml | conf-acl.yaml | pyats_config.py | script.py

pyats > script.py

```
        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)
```

Aniken George D. Flores

To run the script enter these command pyats run job pyats/pyats_config.py. Do this for both PC's.

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