Cobarrubias Joshua B. CPE 028-CPE41S1

Final Case Study | Network Automation and Programmability

Objectives:

Part 1: Design a laboratory activity that discusses the three network topics excluding basic configuration, IP address, and show commands regarding network automation or network programmability.

Part 2: Use pyATS to test your network.

Part 3: Submit a laboratory activity documentation and video presentation of the FINAL CASE STUDY. Make sure that the CAMERA is ON when recording your video presentation.

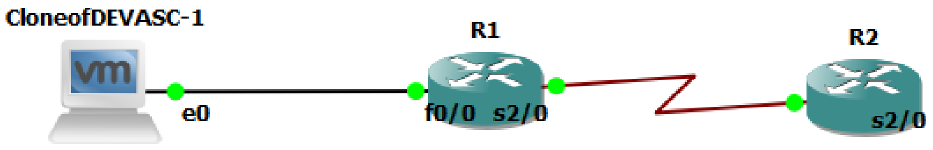
Part 4: Create a GitHub repository of the FINAL CASE STUDY. Make sure to submit all codes, documentation, and video representation.

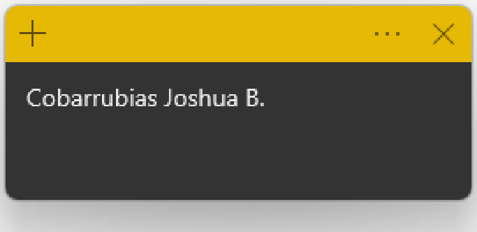
Part 5: Submit the link of your GitHub repository

Intended Learning Outcome (ILOs):

1) Designing a basic topology
2) Implementation of ACL and OSPF using ansible
3) Setting a backup configuration using ansible
4) Using Pyats to test the network
5) Sending the files to GitHub repository

Topology:





Cobarrubias Joshua B.

Required Resources:

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

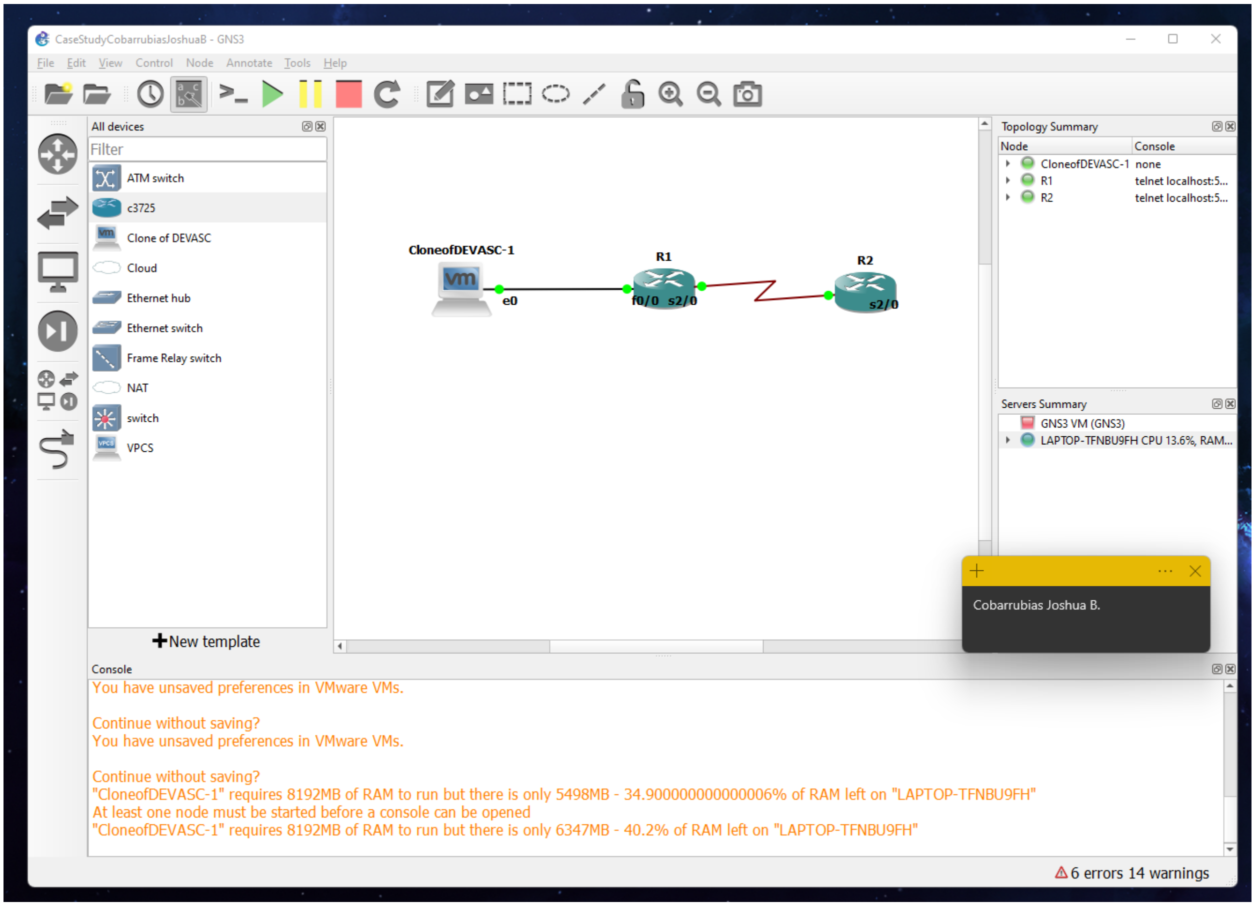
Addressing Table:

Device	Interface	IP Address	Subnet Masks
R1	S2/0	10.0.10.1	255.255.255.252
	F0/0	192.168.1.1	255.255.255.0
R2	S2/0	10.0.10.2	255.255.252.252
PC1	F0/0	192.168.1.2	255.255.255.252

Instructions

Part 1: Launch the DEVASC VM and GNS3

In this part, you launch two VMS. GNS3 and DEVASC.



Topology Summary

Node	Console
CloneofDEVASC-1	none
R1	telnet localhost:5...
R2	telnet localhost:5...

Servers Summary

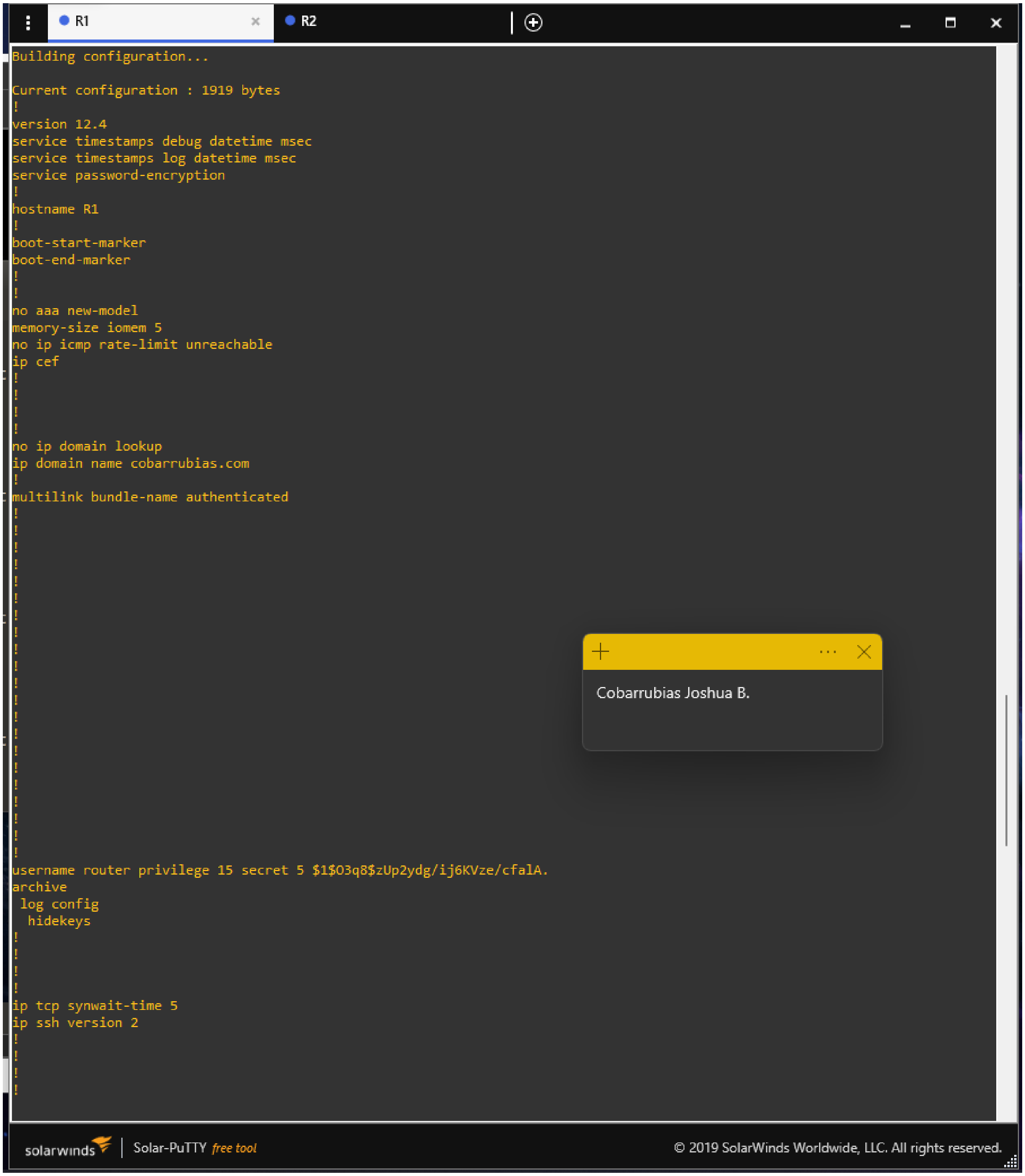
GNS3 VM (GNS3)
LAPTOP-TFNBU9FH CPU 13.6%, RAM...

Console

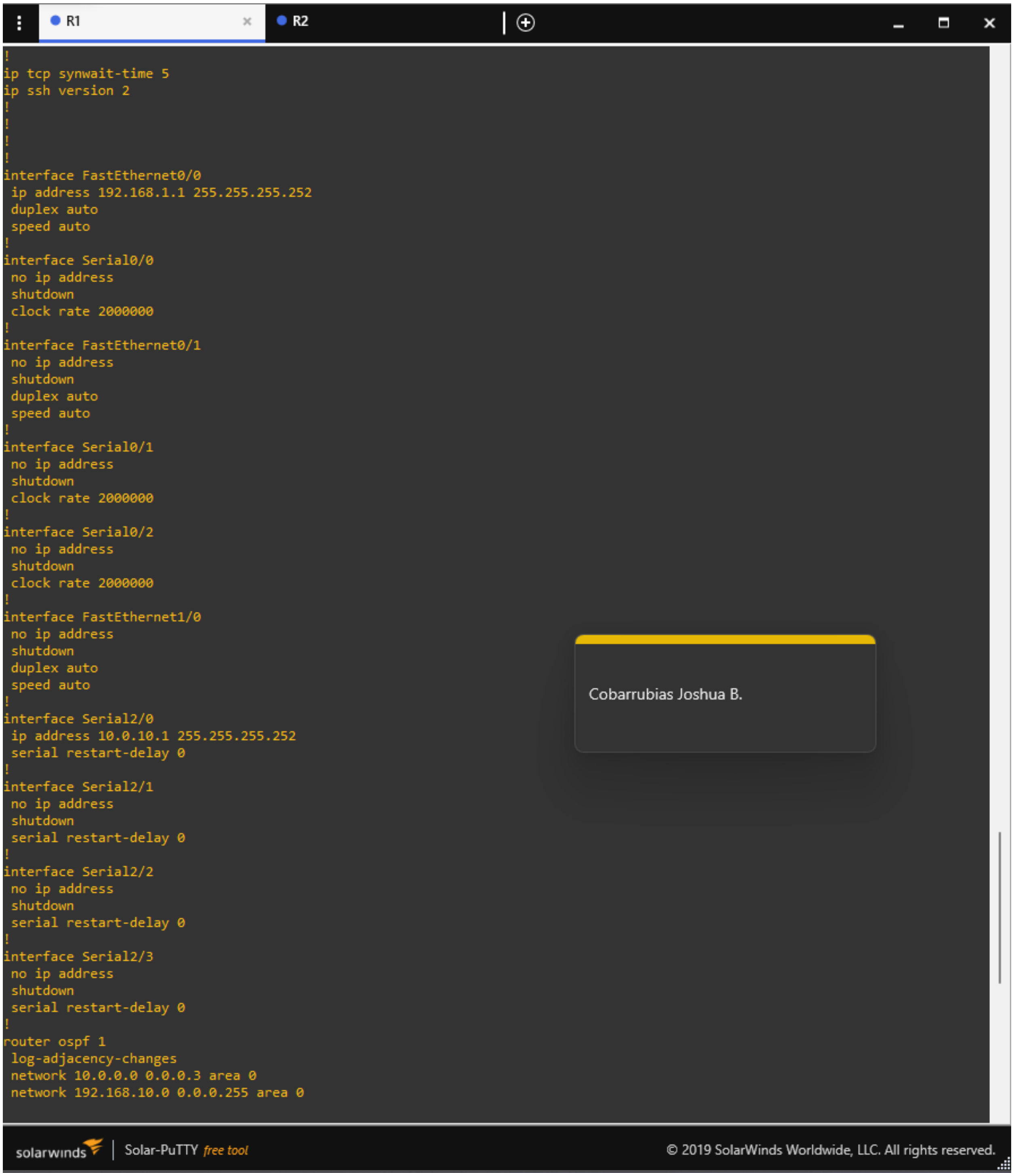
```
You have unsaved preferences in VMware VMs.

Continue without saving?
You have unsaved preferences in VMware VMs.

Continue without saving?
"CloneofDEVASC-1" requires 8192MB of RAM to run but there is only 5498MB - 34.900000000000006% of RAM left on "LAPTOP-TFNBU9FH"
At least one node must be started before a console can be opened
"CloneofDEVASC-1" requires 8192MB of RAM to run but there is only 6347MB - 40.2% of RAM left on "LAPTOP-TFNBU9FH"
```

⚠ 6 errors 14 warnings

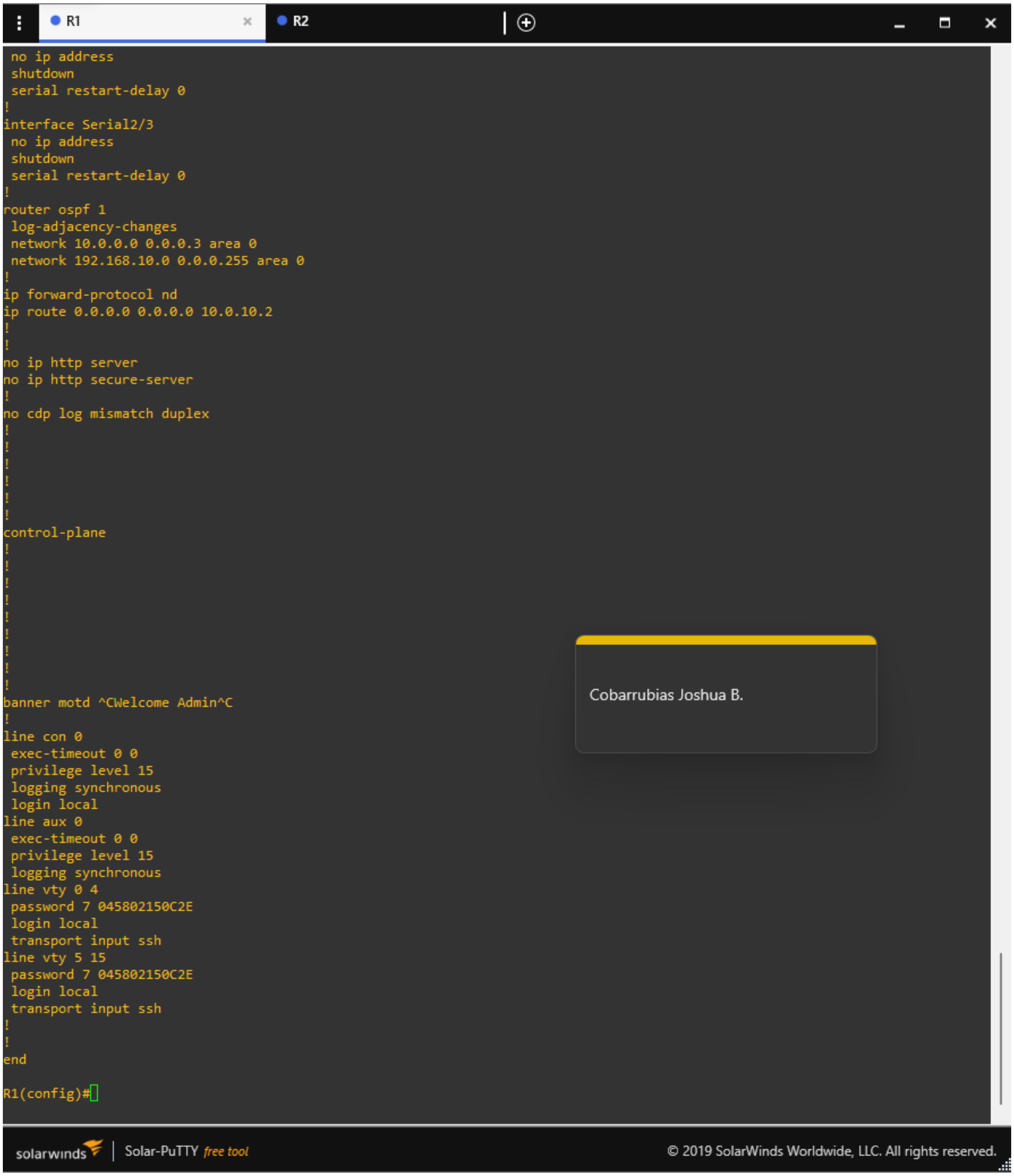
Connect and apply basic configuration on both routers and check if the DEVASC pc VM is connected on R1 and R2.



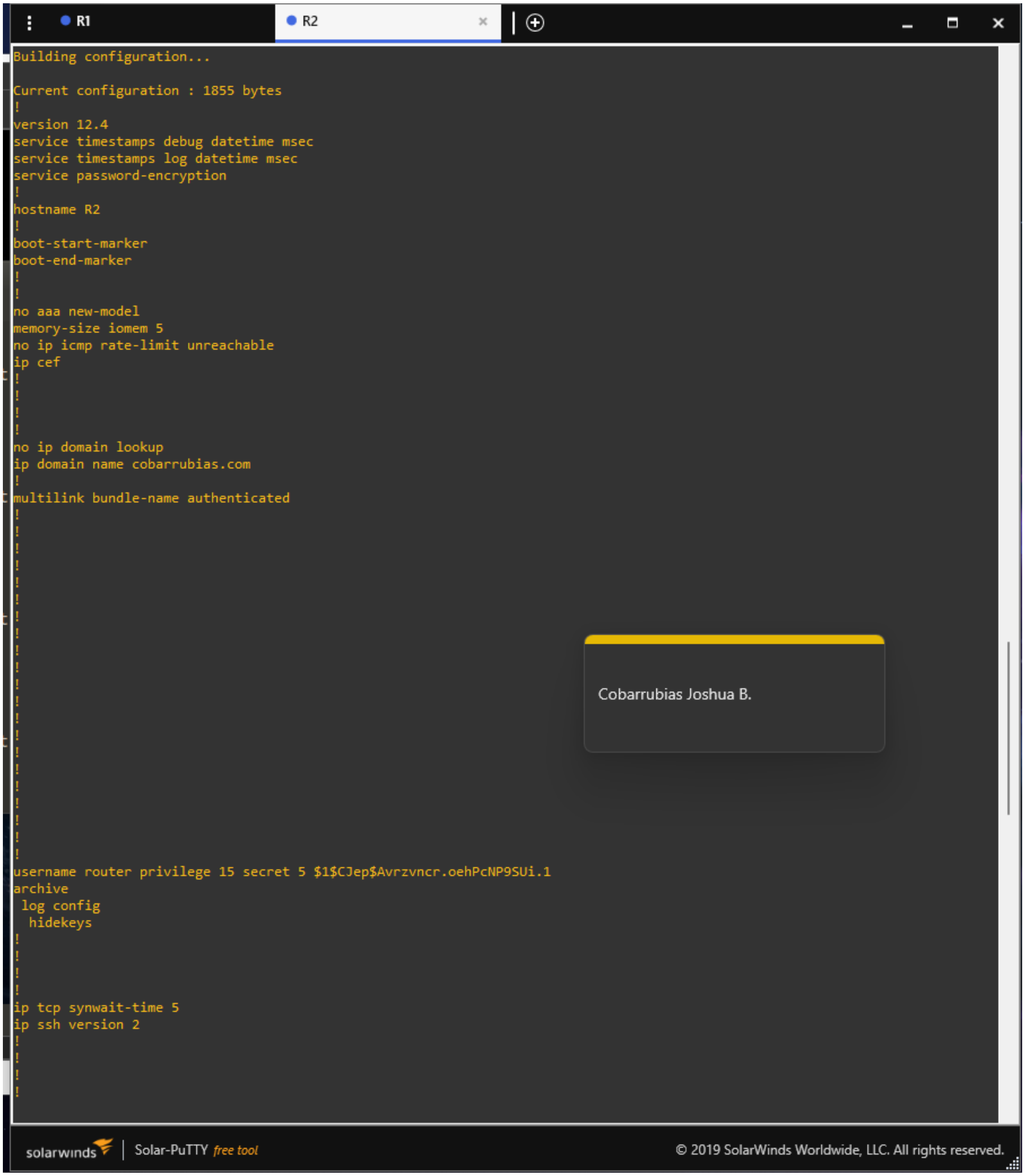
```
Building configuration...

Current configuration : 1919 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
no ip domain lookup
ip domain name cobarrubias.com
!
multilink bundle-name authenticated
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
username router privilege 15 secret 5 $1$O3q8$zUp2ydg/ij6KVze/cfalA.
archive
 log config
  hidekeys
!
!
!
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
```



```
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.252
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 shutdown
 clock rate 2000000
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial0/1
 no ip address
 shutdown
 clock rate 2000000
!
interface Serial0/2
 no ip address
 shutdown
 clock rate 2000000
!
interface FastEthernet1/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial2/0
 ip address 10.0.10.1 255.255.255.252
 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
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
```

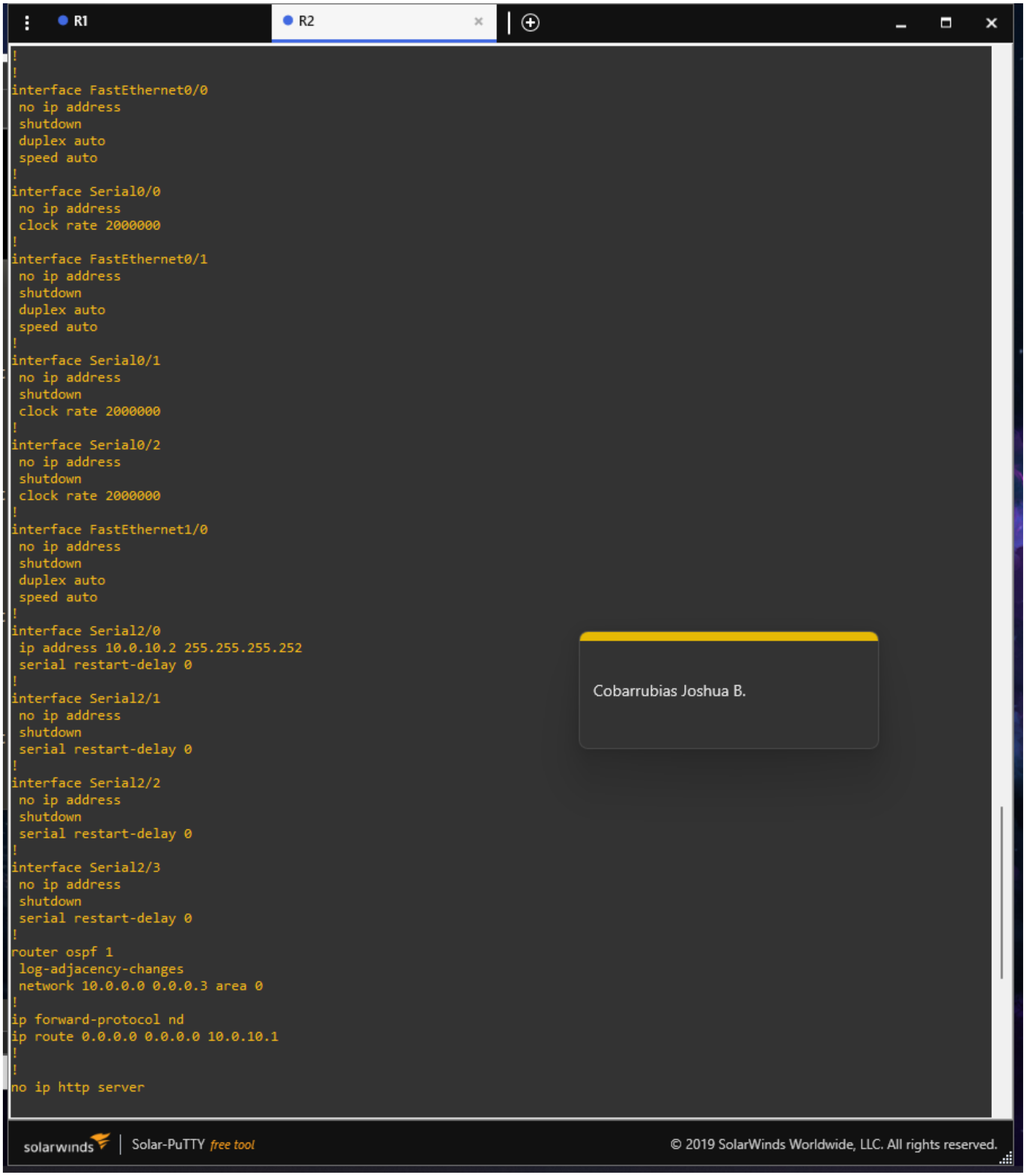


```
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/3
 no ip address
 shutdown
 serial restart-delay 0
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.3 area 0
 network 192.168.10.0 0.0.0.255 area 0
!
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 10.0.10.2
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
!
!
!
!
control-plane
!
!
!
!
!
!
!
!
!
banner motd ^CWelcome Admin^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
 password 7 045802150C2E
 login local
 transport input ssh
line vty 5 15
 password 7 045802150C2E
 login local
 transport input ssh
!
!
end

R1(config)#
```

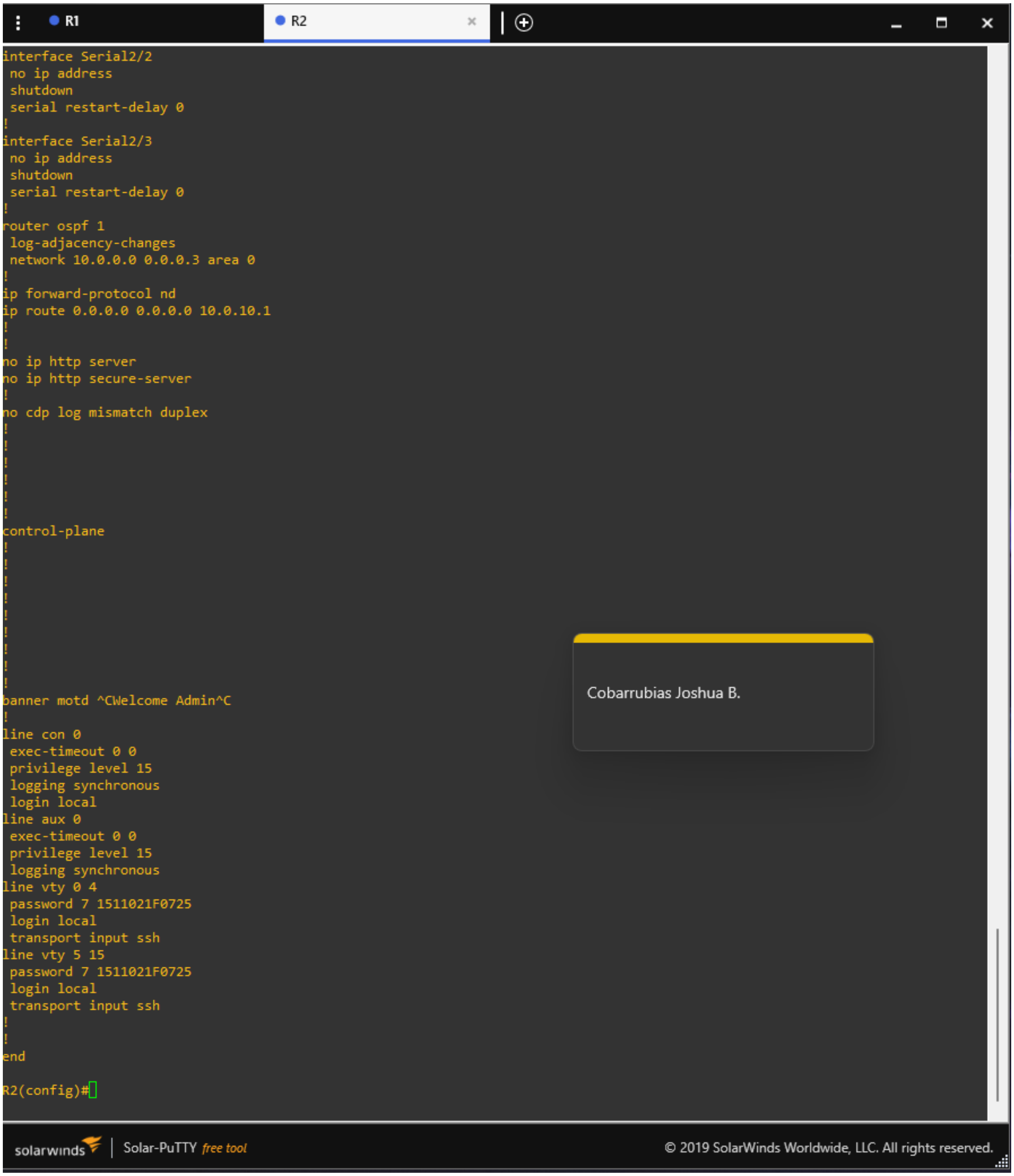


```
Building configuration...

Current configuration : 1855 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
no ip domain lookup
ip domain name cobarrubias.com
!
multilink bundle-name authenticated
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
!
username router privilege 15 secret 5 $1$CJep$Avrzvncr.oehPcNP9SUi.1
archive
 log config
  hidekeys
!
!
!
!
ip tcp synwait-time 5
ip ssh version 2
!
!
!
!
```

Cobarrubias Joshua B.



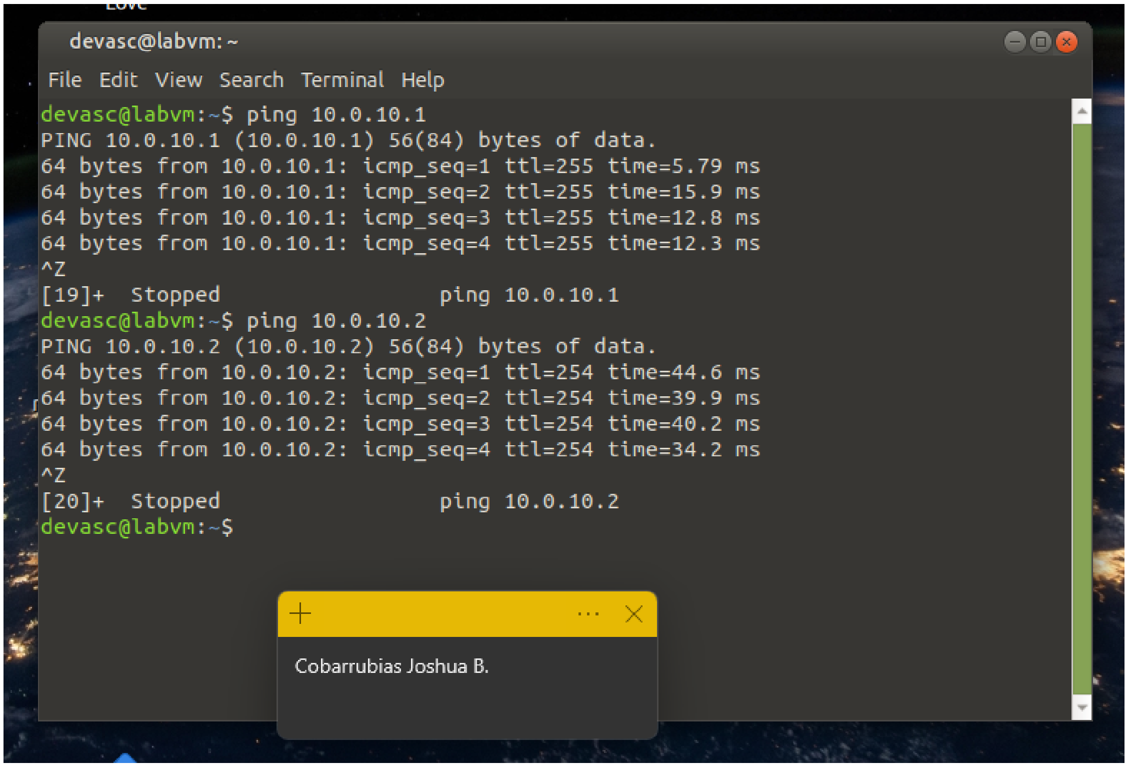
```
!
!
interface FastEthernet0/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 clock rate 2000000
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial0/1
 no ip address
 shutdown
 clock rate 2000000
!
interface Serial0/2
 no ip address
 shutdown
 clock rate 2000000
!
interface FastEthernet1/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
interface Serial2/0
 ip address 10.0.10.2 255.255.255.252
 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
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.3 area 0
!
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 10.0.10.1
!
!
no ip http server
```



R1
R2
Cobarrubias Joshua B.
solarwinds | Solar-PuTTY free tool
© 2019 SolarWinds Worldwide, LLC. All rights reserved.

```
interface Serial2/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/3
 no ip address
 shutdown
 serial restart-delay 0
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.3 area 0
!
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 10.0.10.1
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
!
!
!
!
control-plane
!
!
!
!
!
!
!
!
!
banner motd ^CWelcome Admin^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
 password 7 1511021F0725
 login local
 transport input ssh
line vty 5 15
 password 7 1511021F0725
 login local
 transport input ssh
!
!
end

R2(config)#
```

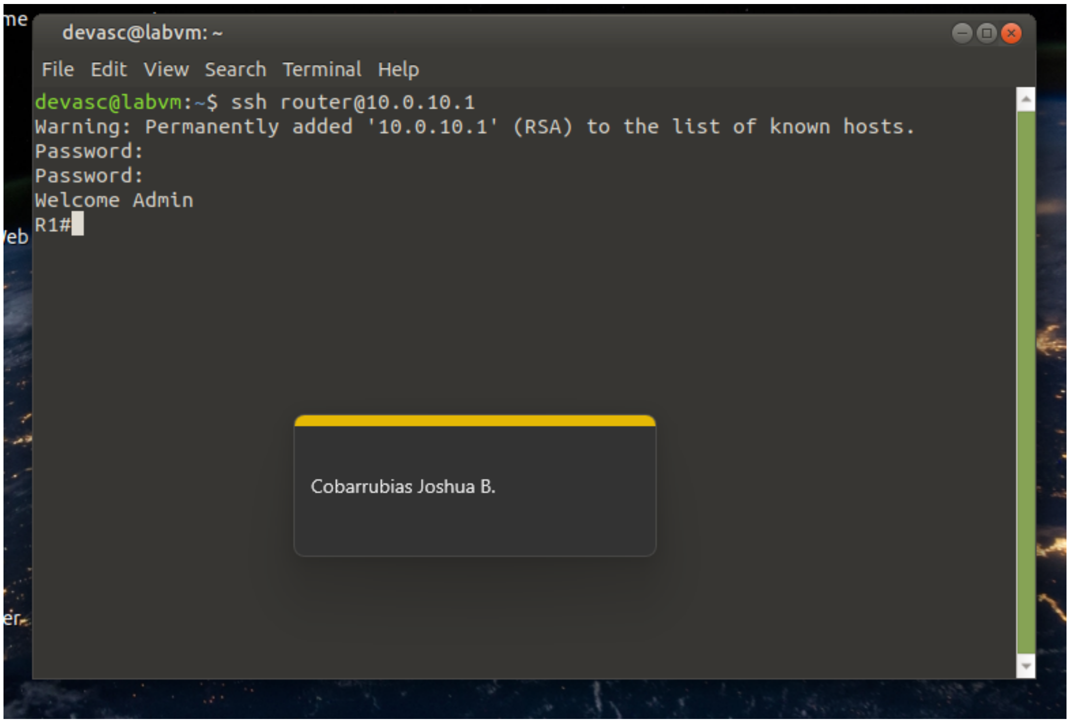
After setting up basic configuration and connecting R1 and R2 and DEVASC VM. We need to check if we could ping R1 and R2 and access them on the SSH.



```
devasc@labvm:~$ ping 10.0.10.1
PING 10.0.10.1 (10.0.10.1) 56(84) bytes of data.
64 bytes from 10.0.10.1: icmp_seq=1 ttl=255 time=5.79 ms
64 bytes from 10.0.10.1: icmp_seq=2 ttl=255 time=15.9 ms
64 bytes from 10.0.10.1: icmp_seq=3 ttl=255 time=12.8 ms
64 bytes from 10.0.10.1: icmp_seq=4 ttl=255 time=12.3 ms
^Z
[19]+  Stopped                 ping 10.0.10.1
devasc@labvm:~$ ping 10.0.10.2
PING 10.0.10.2 (10.0.10.2) 56(84) bytes of data.
64 bytes from 10.0.10.2: icmp_seq=1 ttl=254 time=44.6 ms
64 bytes from 10.0.10.2: icmp_seq=2 ttl=254 time=39.9 ms
64 bytes from 10.0.10.2: icmp_seq=3 ttl=254 time=40.2 ms
64 bytes from 10.0.10.2: icmp_seq=4 ttl=254 time=34.2 ms
^Z
[20]+  Stopped                 ping 10.0.10.2
devasc@labvm:~$
```

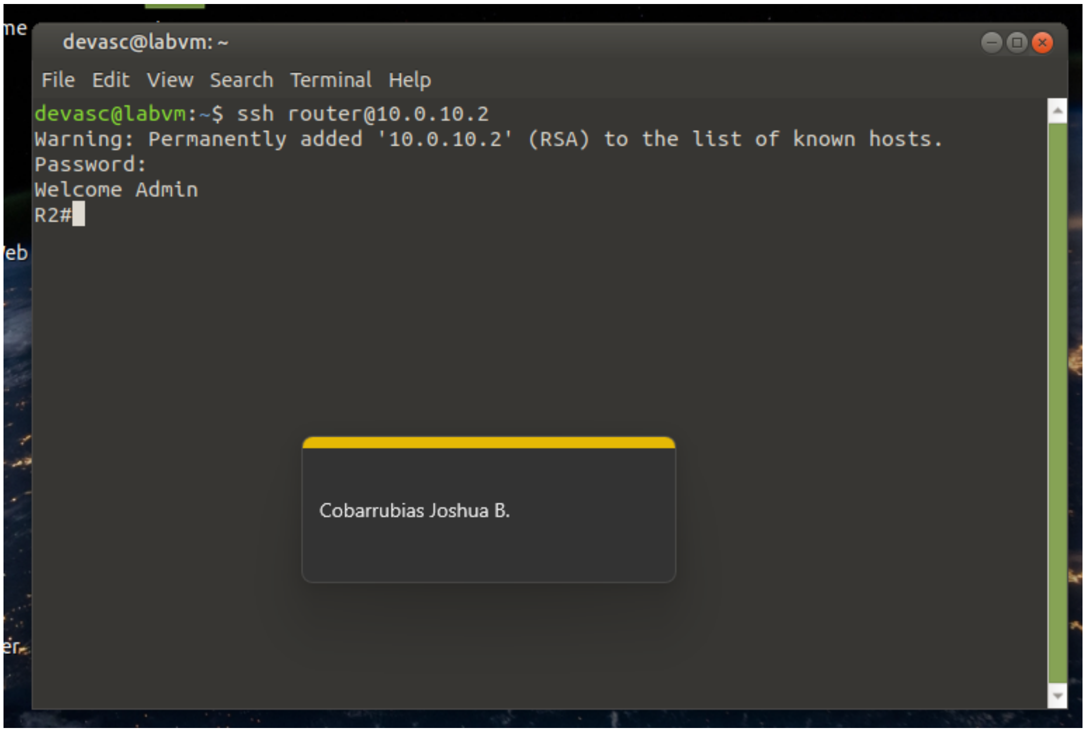
Cobarrubias Joshua B.

Then will be checking if the SSH of R1 and R2 is accessible and fully operational.



```
devasc@labvm:~$ ssh router@10.0.10.1
Warning: Permanently added '10.0.10.1' (RSA) to the list of known hosts.
Password:
Password:
Welcome Admin
R1#
```

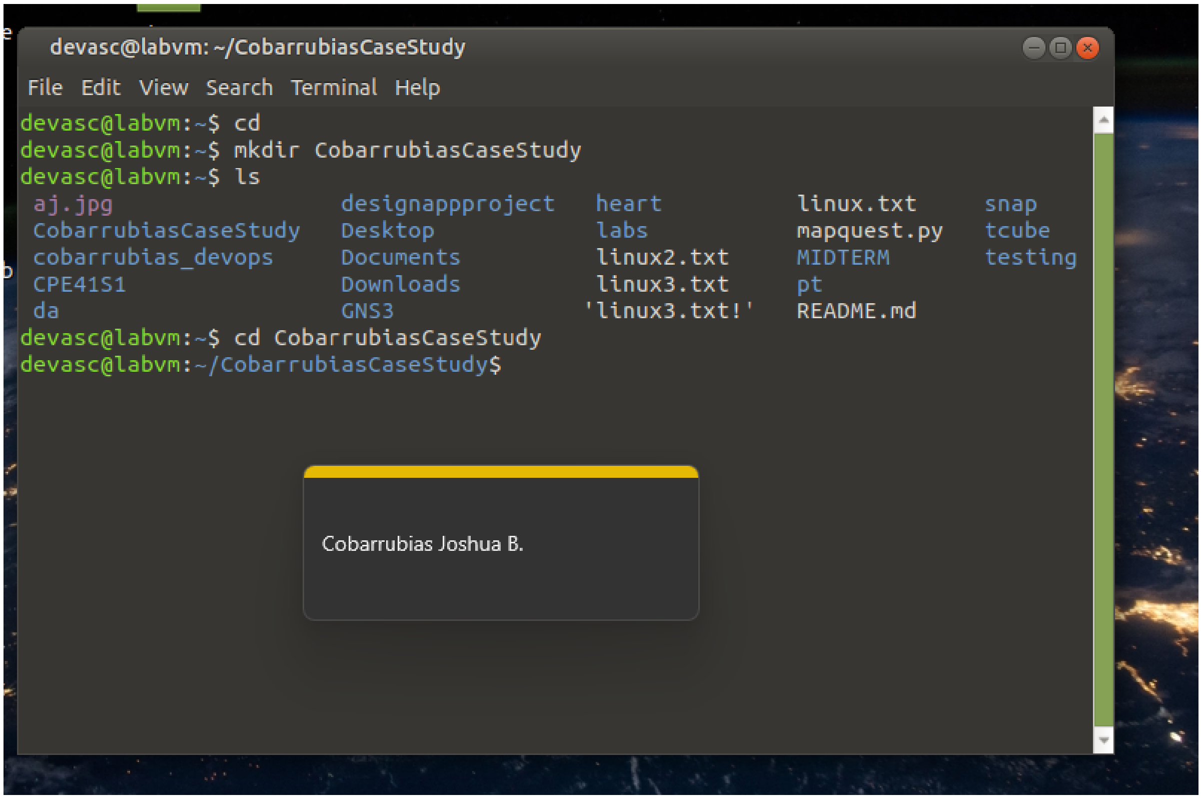
Cobarrubias Joshua B.



```
devasc@labvm:~$ ssh router@10.0.10.2
Warning: Permanently added '10.0.10.2' (RSA) to the list of known hosts.
Password:
Welcome Admin
R2#
```

Cobarrubias Joshua B.

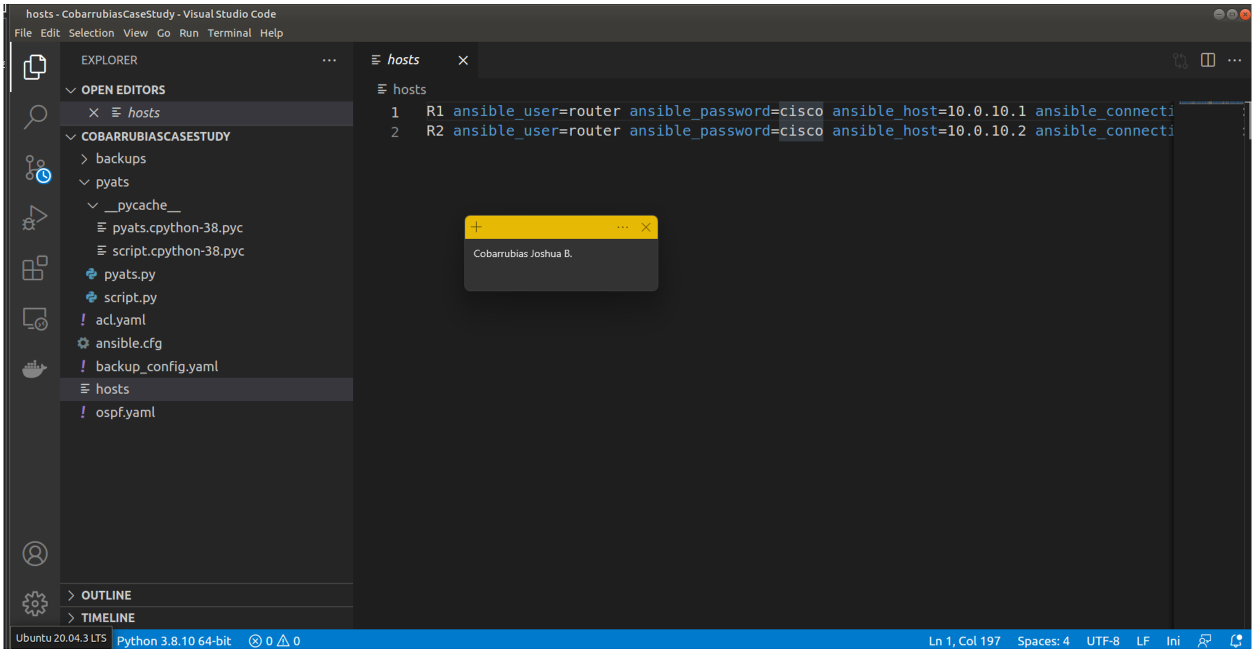
After successfully accessing both routers using SSH we need to go to our DEVASC and create a project directory.


devasc@labvm: ~/CobarrubiasCaseStudy
File Edit View Search Terminal Help

```
devasc@labvm:~$ cd
devasc@labvm:~$ mkdir CobarrubiasCaseStudy
devasc@labvm:~$ ls
 aj.jpg                 designappproject   heart          linux.txt     snap
 CobarrubiasCaseStudy   Desktop            labs           mapquest.py   tcube
 cobarrubias_devops     Documents          linux2.txt     MIDTERM       testing
 CPE41S1                Downloads          linux3.txt     pt
 da                     GNS3              'linux3.txt!'   README.md
devasc@labvm:~$ cd CobarrubiasCaseStudy
devasc@labvm:~/CobarrubiasCaseStudy$
```

Cobarrubias Joshua B.

Then we need to open VSCode to open the project file using the directory we made CobarrubiasCaseStudy. Then will be creating a host file containing the ansible username and password on host file.



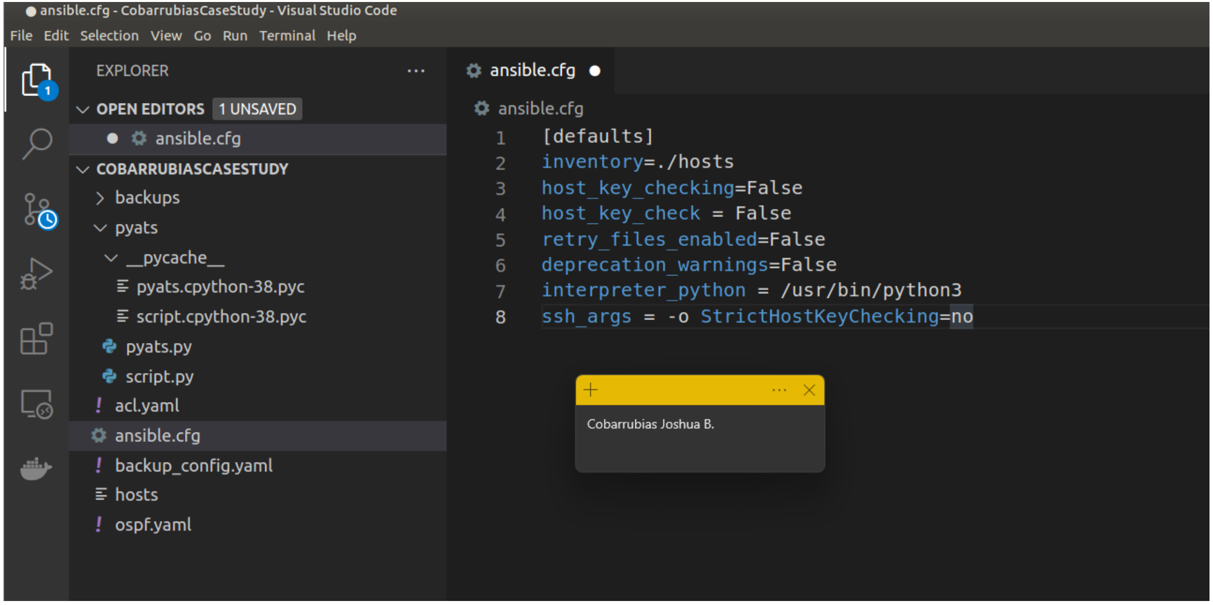
```
R1 ansible_user=router ansible_password=cisco ansible_host=10.0.10.1 ansible_connecti
R2 ansible_user=router ansible_password=cisco ansible_host=10.0.10.2 ansible_connecti
```

This will be the codes for the hosts file.

R1 ansible_user=router ansible_password=cisco ansible_host=10.0.10.1 ansible_connection=network_cli ansible_network_os=ios ansible_become=yes ansible_become_method=enable ansible_become_pass=cisco

R2 ansible_user=router ansible_password=cisco ansible_host=10.0.10.2 ansible_connection=network_cli ansible_network_os=ios ansible_become=yes ansible_become_method=enable ansible_become_pass=cisco

After creating an hosts file, we need to create an ansible configuration file on the directory naming ansible.cfg.



```
[defaults]
inventory=./hosts
host_key_checking=False
host_key_check = False
retry_files_enabled=False
deprecation_warnings=False
interpreter_python = /usr/bin/python3
ssh_args = -o StrictHostKeyChecking=no
```

This will be the codes for the ansible.cfg

[defaults]

inventory=./hosts

host_key_checking=False

host_key_check = False

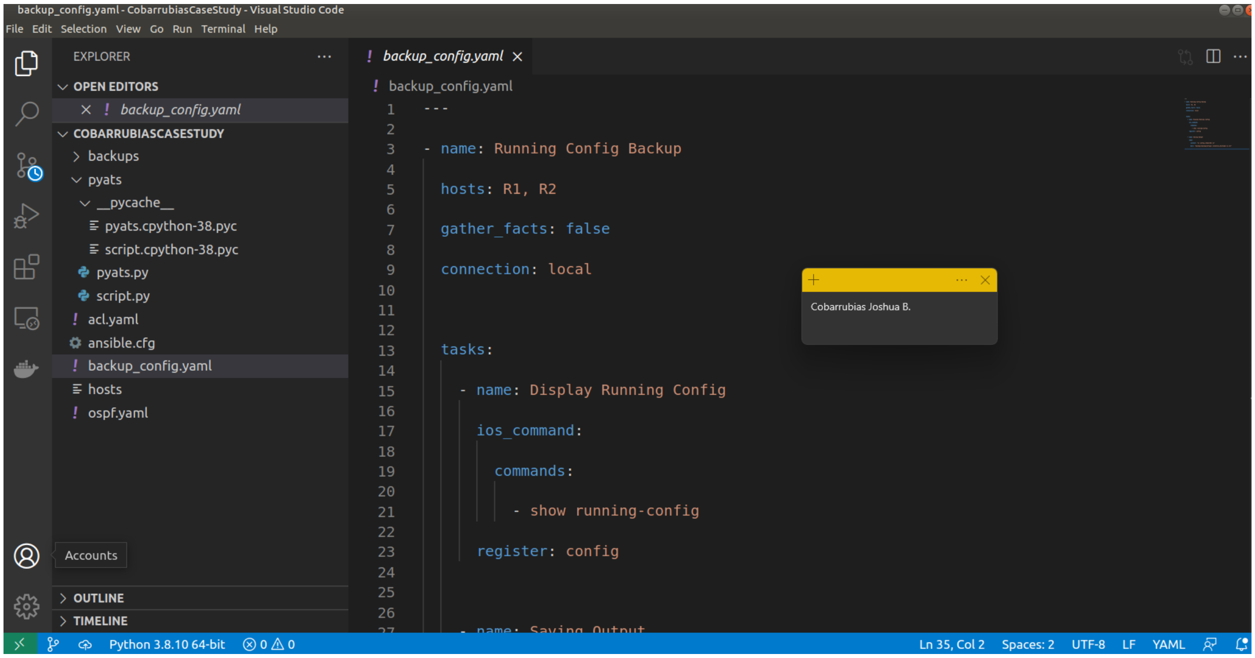
retry_files_enabled=False

deprecation_warnings=False

interpreter_python = /usr/bin/python3

ssh_args = -o StrictHostKeyChecking=no

We will now create our yaml file naming backup_config.yaml this will be the backup of our configuration on our R1 and R2.

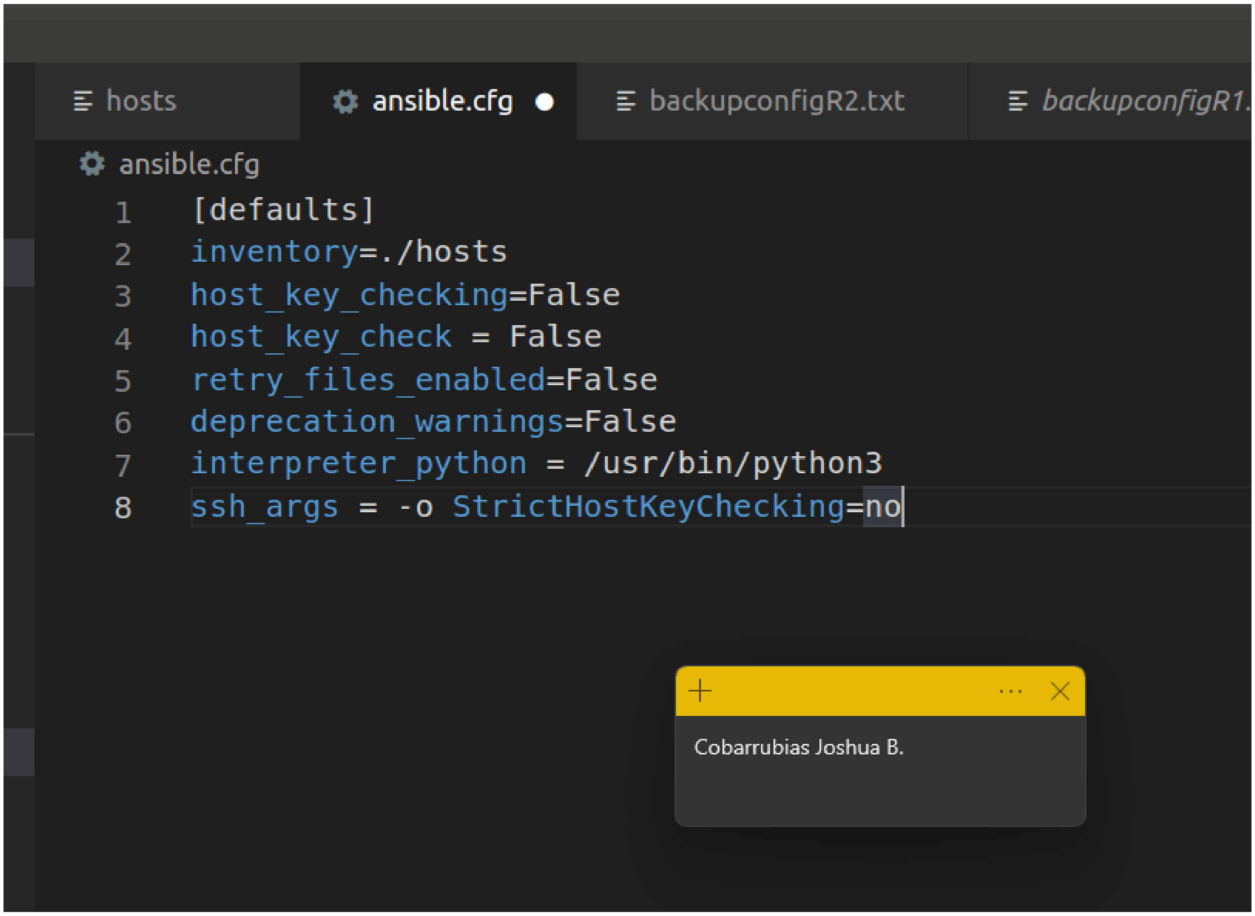


```
---

- 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
```

This will be the code for 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"
```

We will now create our ansible configure file will be naming it ansible.cfg



```
[defaults]
inventory=./hosts
host_key_checking=False
host_key_check = False
retry_files_enabled=False
deprecation_warnings=False
interpreter_python = /usr/bin/python3
ssh_args = -o StrictHostKeyChecking=no
```

This will be the code for the ansible.cfg

[defaults]

inventory=./hosts

host_key_checking=False

host_key_check = False

retry_files_enabled=False

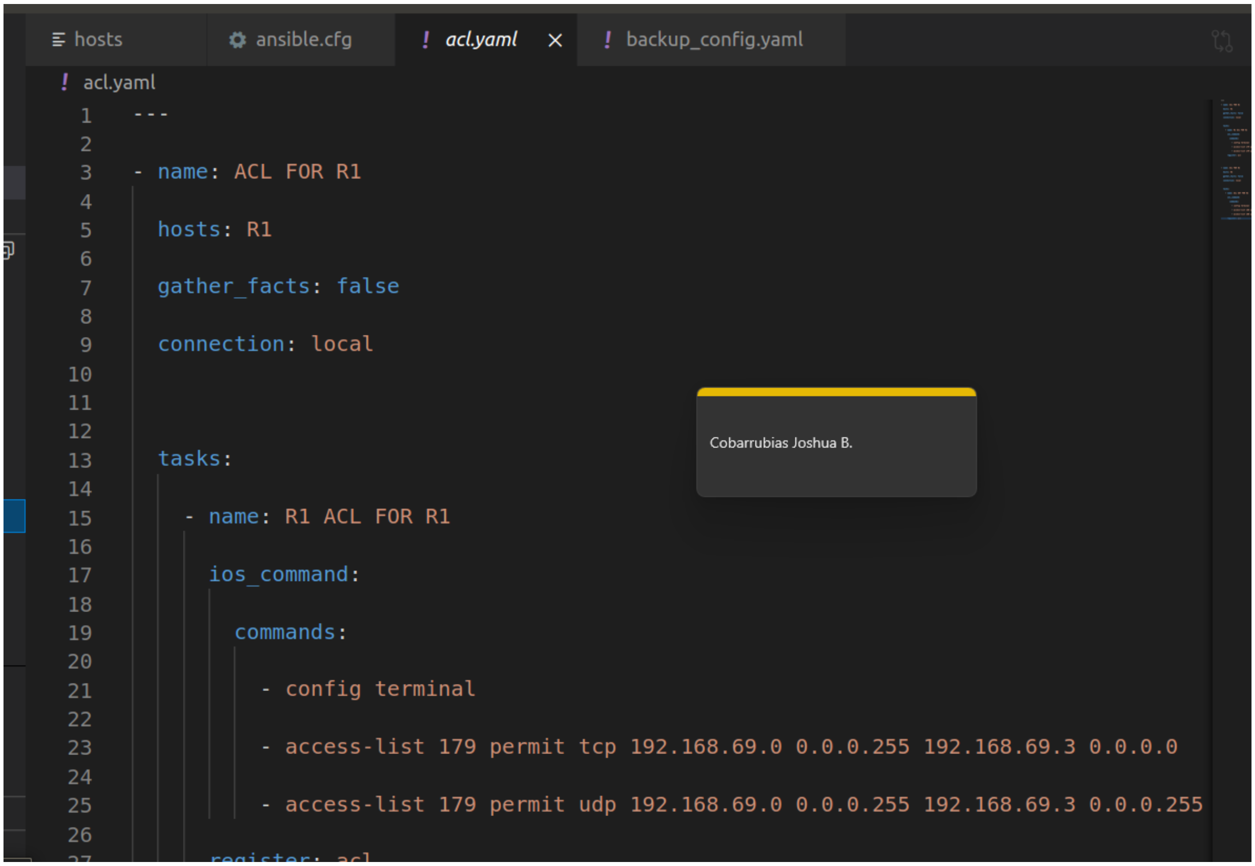
deprecation_warnings=False

interpreter_python = /usr/bin/python3

# ssh arguments to use

ssh_args = -o StrictHostKeyChecking=no

After creating the ansible.cfg we will now be creating our ACL configuration and naming it acl.yaml

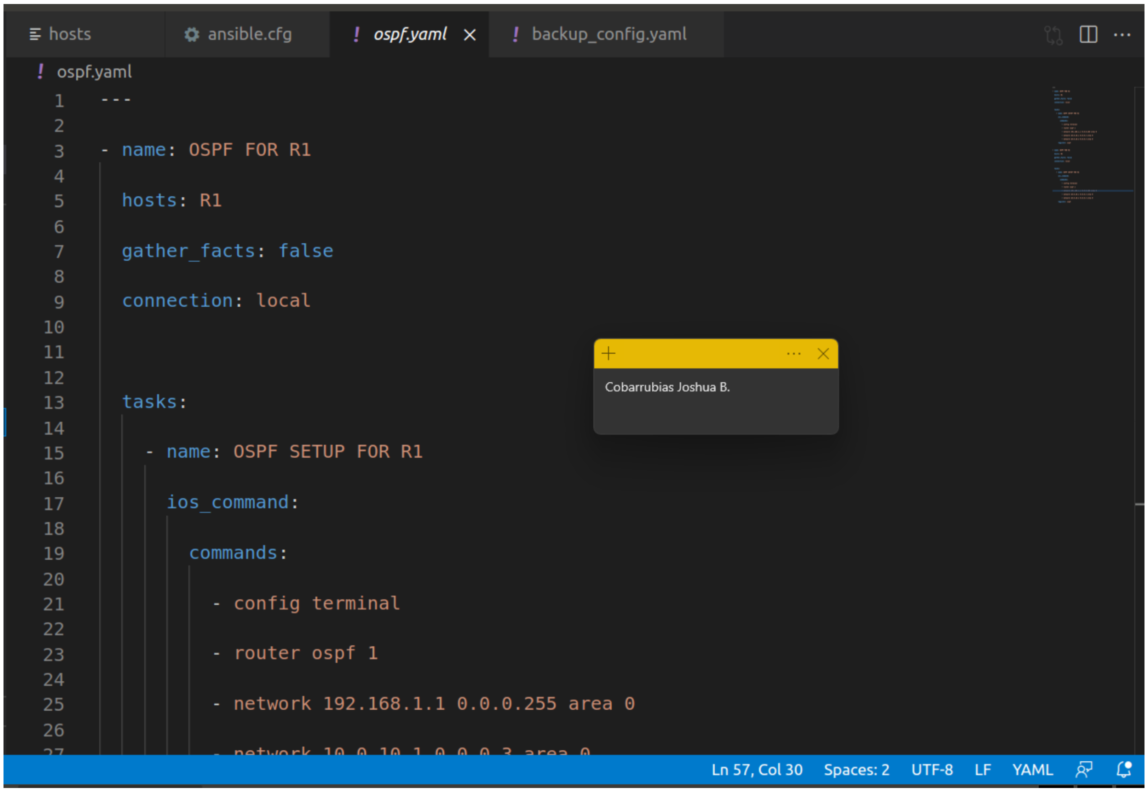


```
---

- name: ACL FOR R1

  hosts: R1

  gather_facts: false

  connection: local


  tasks:

    - name: R1 ACL FOR R1

      ios_command:

        commands:

          - config terminal

          - access-list 179 permit tcp 192.168.69.0 0.0.0.255 192.168.69.3 0.0.0.0

          - access-list 179 permit udp 192.168.69.0 0.0.0.255 192.168.69.3 0.0.0.255

      register: acl
```

This will be the code for the acl.yaml

```
---
- name: ACL FOR R1
  hosts: R1
  gather_facts: false
  connection: local
  tasks:
    - name: R1 ACL FOR R1
      ios_command:
        commands:
          - config terminal
          - access-list 179 permit tcp 192.168.69.0 0.0.0.255 192.168.69.3 0.0.0.0
          - access-list 179 permit udp 192.168.69.0 0.0.0.255 192.168.69.3 0.0.0.255
      register: acl
- name: ACL FOR R2
  hosts: R2
  gather_facts: false
  connection: local
  tasks:
    - name: ACL SET FOR R2
      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
```

And for the last we will be creating our OSPF configuration and will be naming it ospf.yaml

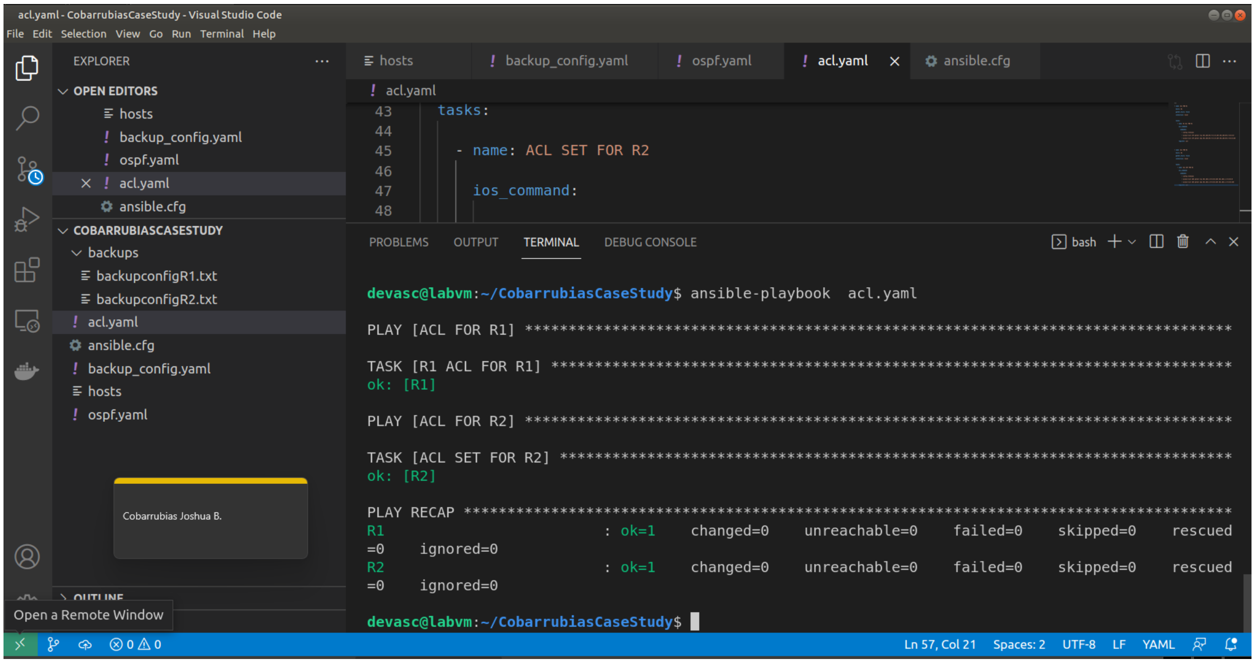


```yaml
---

- name: OSPF FOR R1

  hosts: R1

  gather_facts: false

  connection: local



  tasks:

    - name: OSPF SETUP FOR R1

      ios_command:

        commands:

          - config terminal

          - router ospf 1

          - network 192.168.1.1 0.0.0.255 area 0

          - network 10.0.10.1 0.0.0.3 area 0
```

This will be the code for the ospf.yaml

```
---
- name: OSPF FOR R1
  hosts: R1
  gather_facts: false
  connection: local
  tasks:
    - name: OSPF SETUP FOR R1
      ios_command:
        commands:
          - config terminal
          - router ospf 1
          - network 192.168.1.1 0.0.0.255 area 0
          - network 10.0.10.1 0.0.0.3 area 0
          - network 10.0.10.2 0.0.0.3 area 0
      register: ospf
- name: OSPF FOR R2
  hosts: R2
  gather_facts: false
  connection: local
  tasks:
    - name: OSPF SETUP FOR R2
      ios_command:
        commands:
          - config terminal
          - router ospf 1
          - network 192.168.1.1 0.0.0.255 area 0
          - network 10.0.10.1 0.0.0.3 area 0
          - network 10.0.10.2 0.0.0.3 area 0
      register: ospf
```

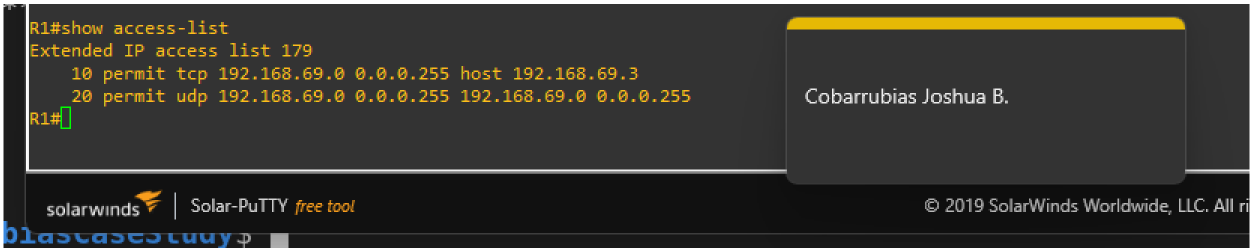
After creating all the files will be running it using ansible-playbook. The first thing will run is the acl.yaml we need to go to terminal and write ansible-playbook acl.yaml



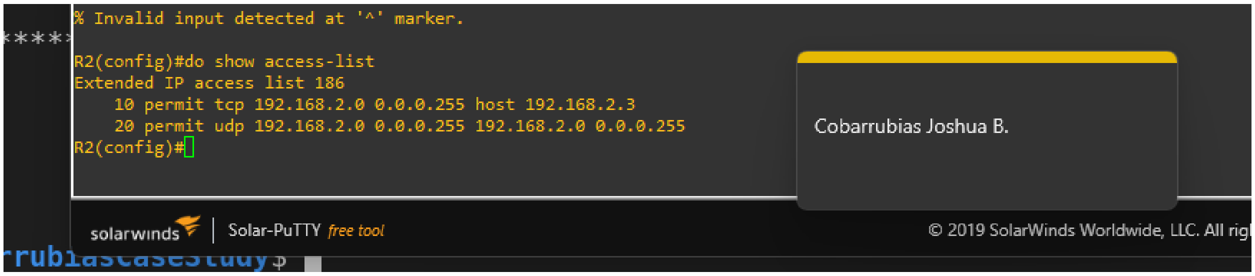
```
tasks:

  - name: ACL SET FOR R2

    ios_command:
```

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

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

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

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

TASK [ACL SET FOR R2] **************************************************************************
ok: [R2]

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

devasc@labvm:~/CobarrubiasCaseStudy$ 
```

To check if the ACL is implemented we need to go to GNS3 console of R1,R2 and type show access-lists



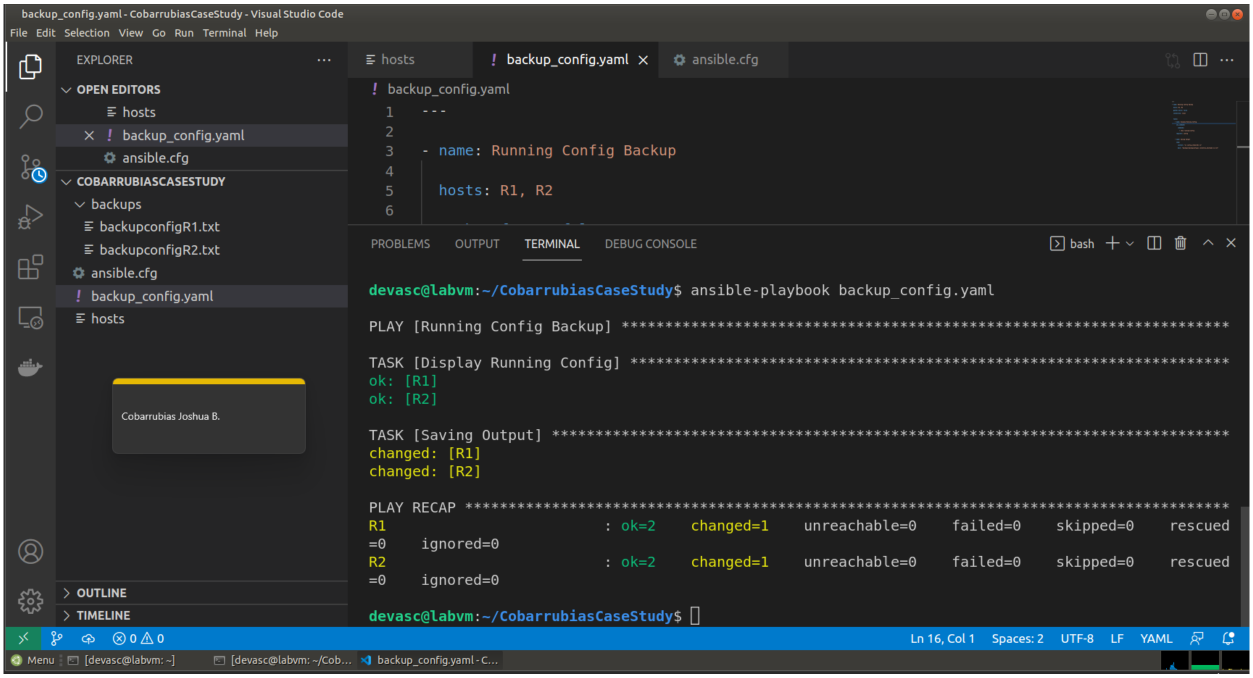
```
R1#show access-list
Extended IP access list 179
    10 permit tcp 192.168.69.0 0.0.0.255 host 192.168.69.3
    20 permit udp 192.168.69.0 0.0.0.255 192.168.69.0 0.0.0.255
R1#
```



```
% Invalid input detected at '^' marker.

R2(config)#do show access-list
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)#
```

As you can see the ACL was fully configured using ansible.

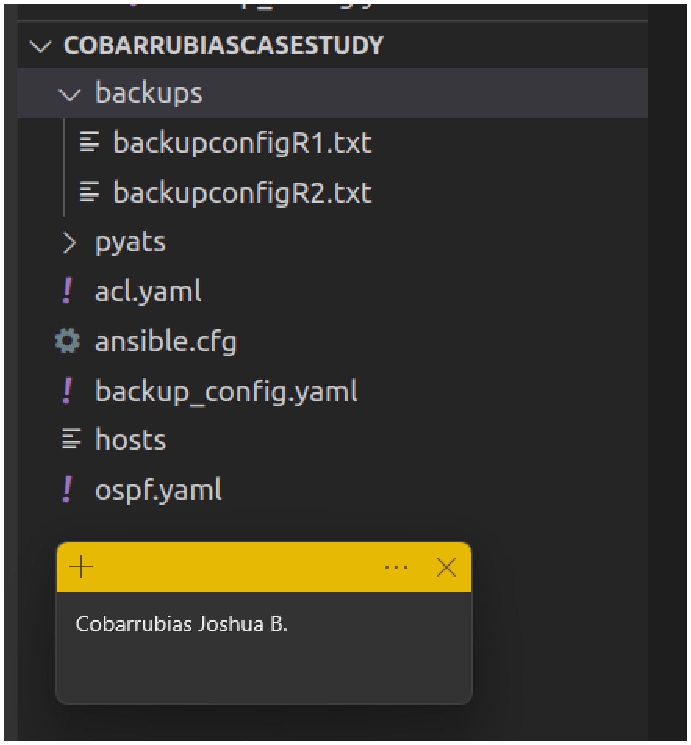
The next thing will run is our backup_config.yaml using ansible by using this code on our terminal on the VSCode ansible-playbook backup_config.yaml



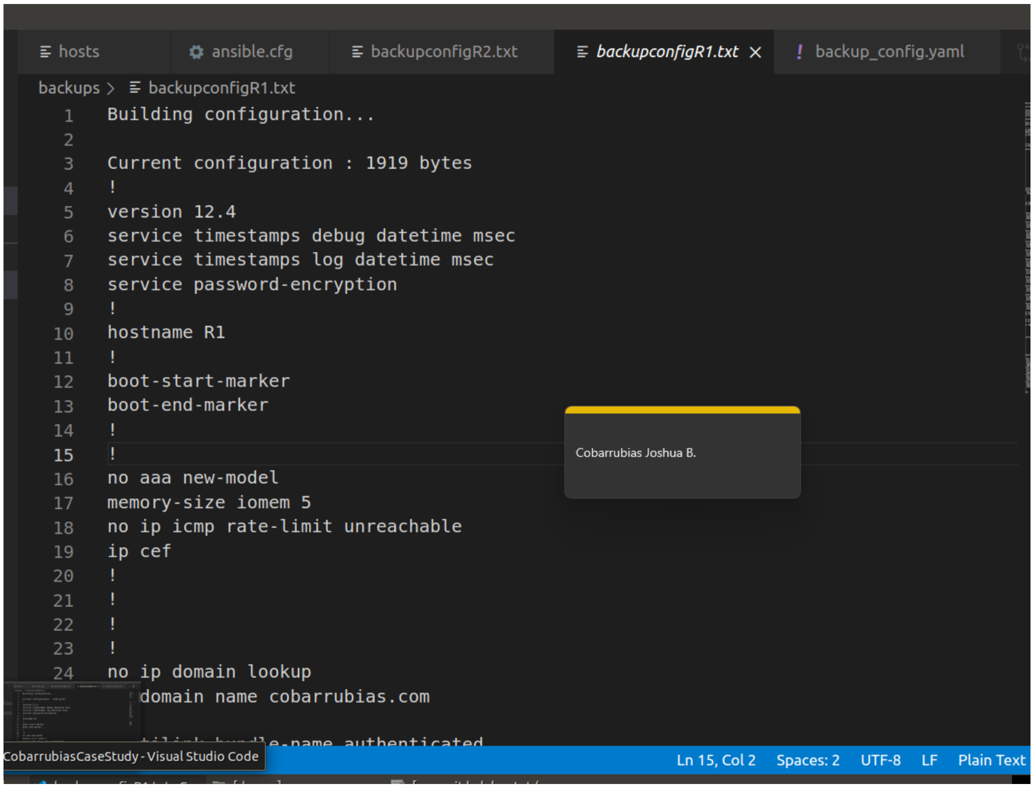
```yaml
---

- name: Running Config Backup

  hosts: R1, R2

```

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

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

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

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

PLAY RECAP *****************************************************************************************
R1                         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued
=0    ignored=0
R2                         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued
=0    ignored=0

devasc@labvm:~/CobarrubiasCaseStudy$ 
```

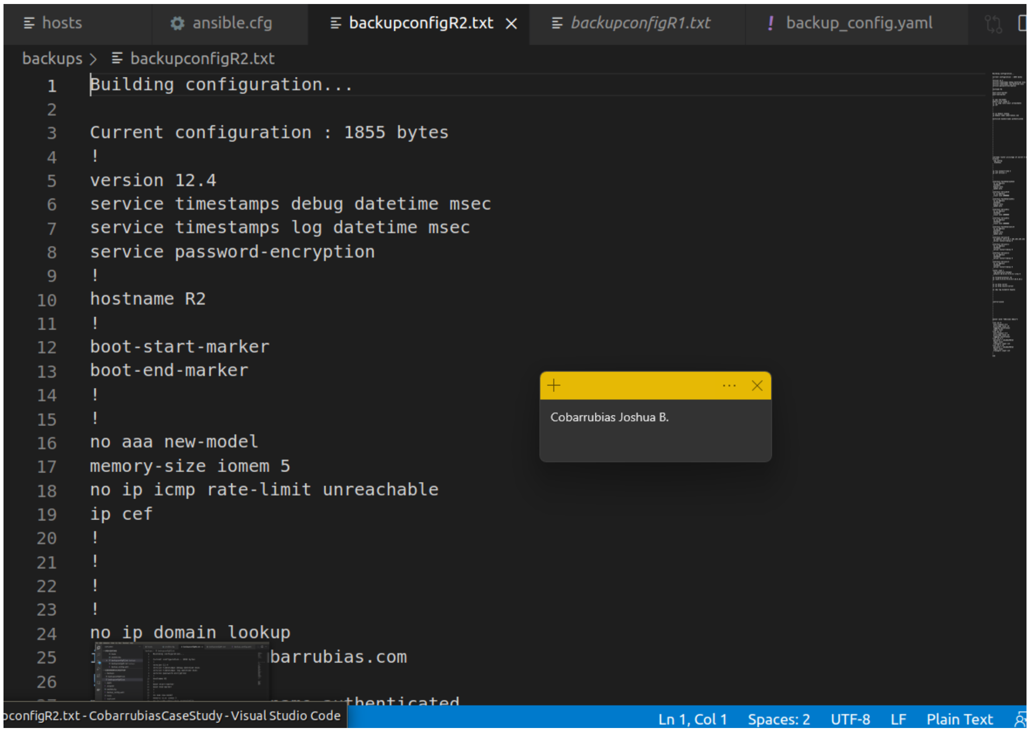
Then to check if the backup_config.yaml was successfully using ansible we will check to backups folder if it contains our backups_config by txt file.



And will be checking if the backupconfigR1.txt file has our configuration same to backupconfigR2.txt

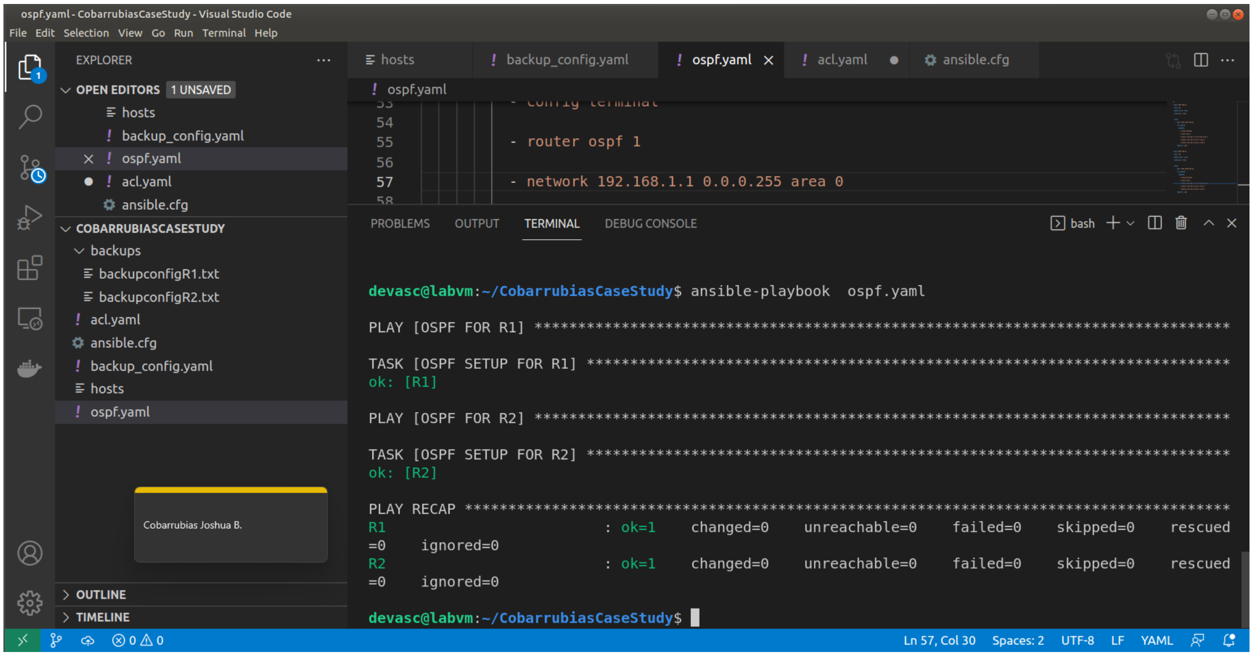


```
Building configuration...

Current configuration : 1919 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
no ip domain lookup
   domain name cobarrubias.com

   tilink bundle-name authenticated
```



```
Building configuration...

Current configuration : 1855 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
memory-size iomem 5
no ip icmp rate-limit unreachable
ip cef
!
!
!
!
no ip domain lookup
i            barrubias.com
!
                   thenticated
```

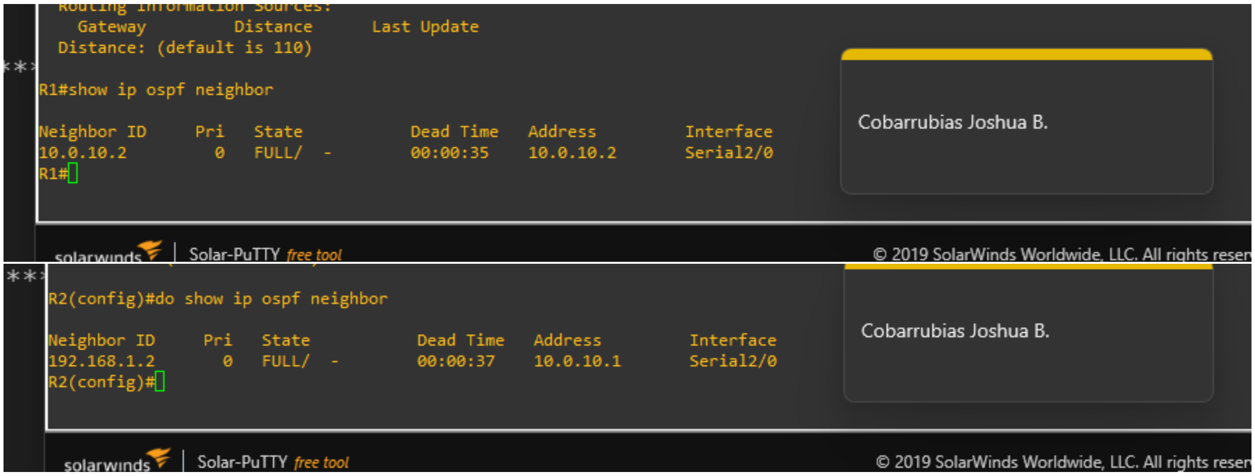
After creating the ACL and backing up the configuration of the routers R1 and R2. We will now create our OSPF on our routers R1 and R2. So, we already made the files for OSPF we will now only test it. By going to the terminal on VSCode and inputting ansible-playbook ospf.yaml



```
        - config terminal

        - router ospf 1

        - network 192.168.1.1 0.0.0.255 area 0
```

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

PLAY [OSPF FOR R1] ****************************************************************************

TASK [OSPF SETUP FOR R1] **********************************************************************
ok: [R1]

PLAY [OSPF FOR R2] ****************************************************************************

TASK [OSPF SETUP FOR R2] **********************************************************************
ok: [R2]

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

devasc@labvm:~/CobarrubiasCaseStudy$ 
```

So, as you can see it was successful using ansible now to check if OSPF was successfully implemented we need to go to GNS3 and go to R1 and R2 console and type in show ip ospf neighbor.

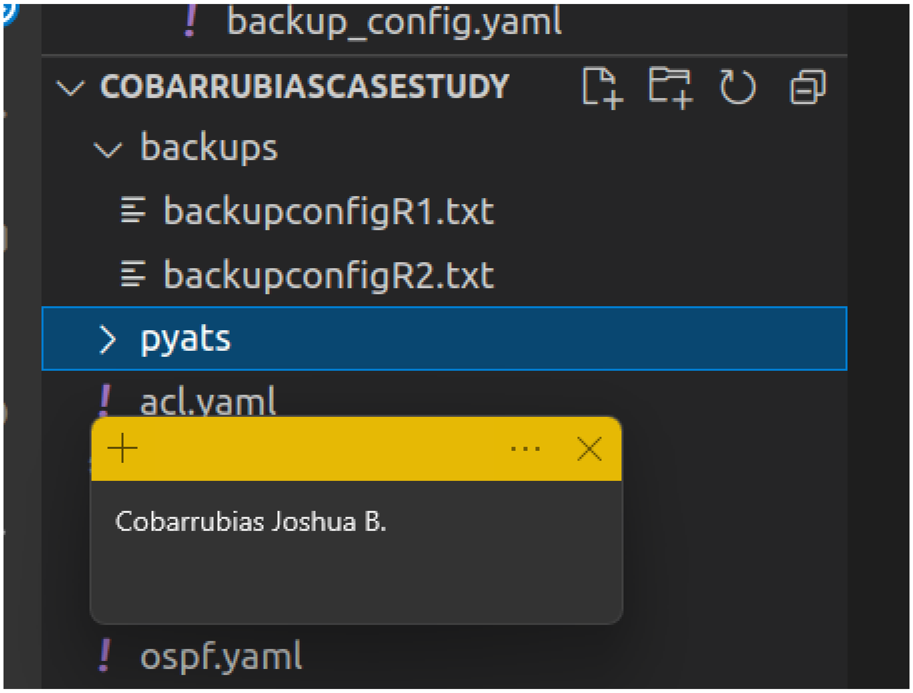

Cobarrubias Joshua B.
solarwinds | Solar-PuTTY free tool
© 2019 SolarWinds Worldwide, LLC. All rights reser
Cobarrubias Joshua B.
solarwinds | Solar-PuTTY free tool
© 2019 SolarWinds Worldwide, LLC. All rights reser

```
  Routing Information Sources:
    Gateway         Distance      Last Update
  Distance: (default is 110)

R1#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.0.10.2         0   FULL/  -        00:00:35    10.0.10.2       Serial2/0
R1#
```

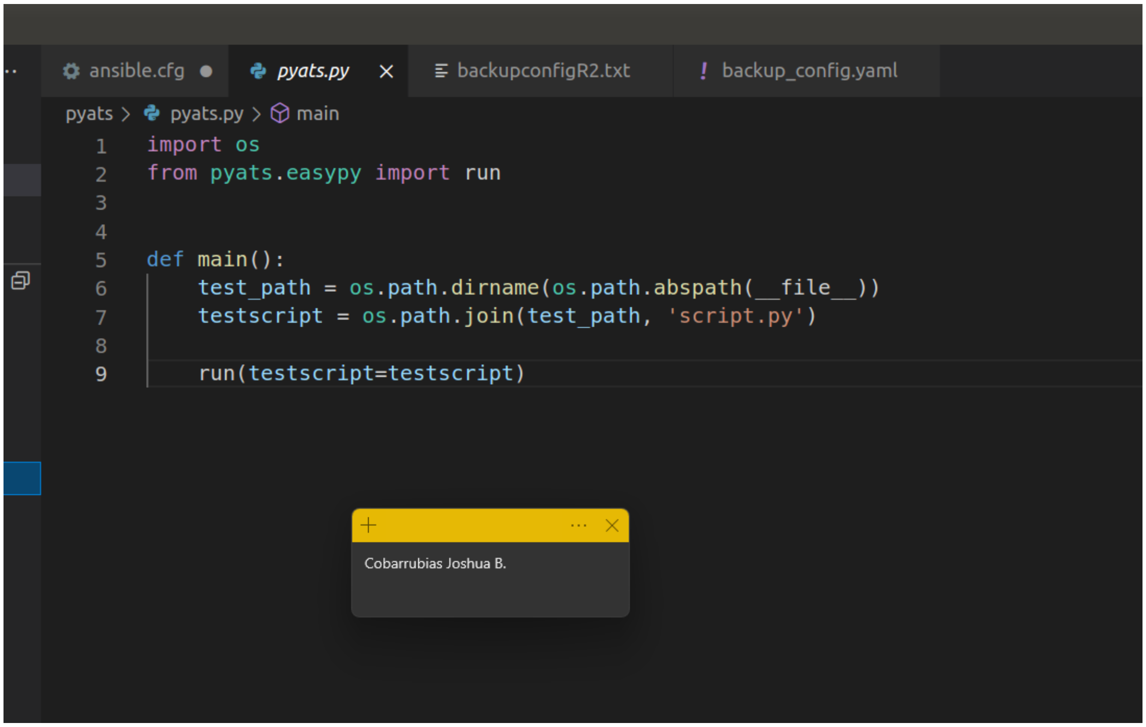
```
R2(config)#do show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
192.168.1.2       0   FULL/  -        00:00:37    10.0.10.1       Serial2/0
R2(config)#
```

So, as you can see OSPF was implemented on both routers R1 and R2 because now it has a neighbor ID of an OSPF.

Now for the final we will now create our pyats by creating a script to test our network. First, we need to create an pyats folder inside our directory.



After creating a folder to our directory, we need to create a pyats.py for testing our network.

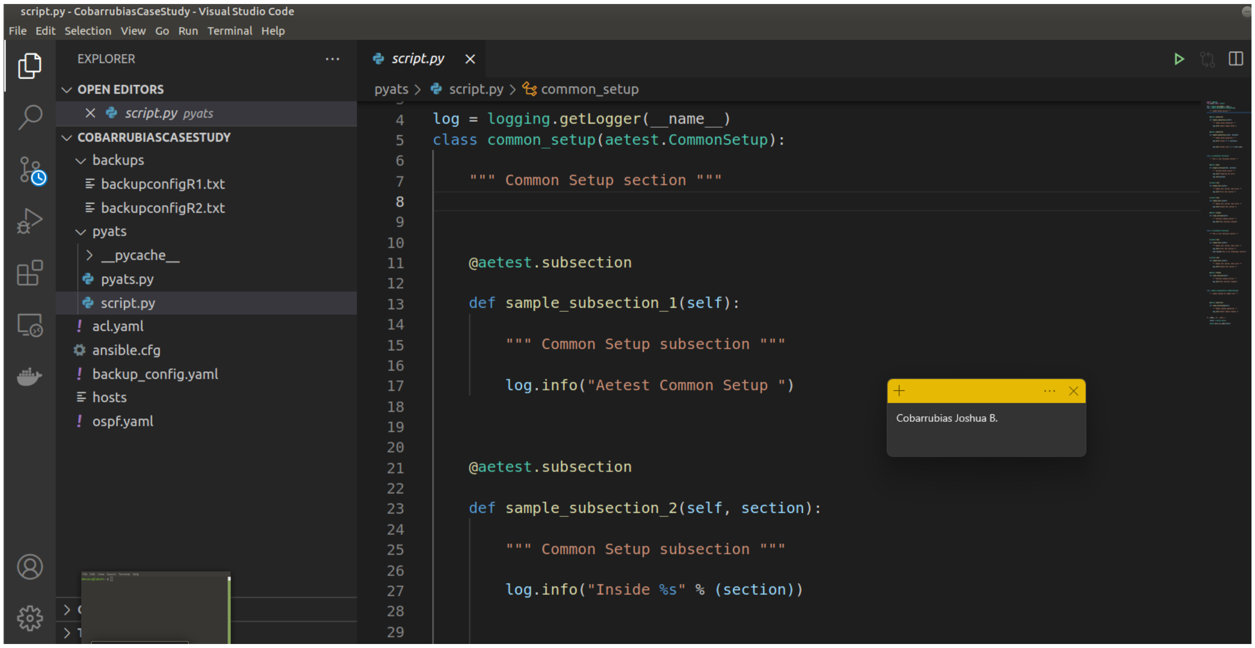


```
import os
from pyats.easypy import run


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

    run(testscript=testscript)
```

This will be the code for pyats.py

```
import os

from pyats.easy py import run

def main():

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

    testscript = os.path.join(test_path, 'script.py')

    run(testscript=testscript)
```

After creating the pyats.py we will need to create another python file naming it script.py



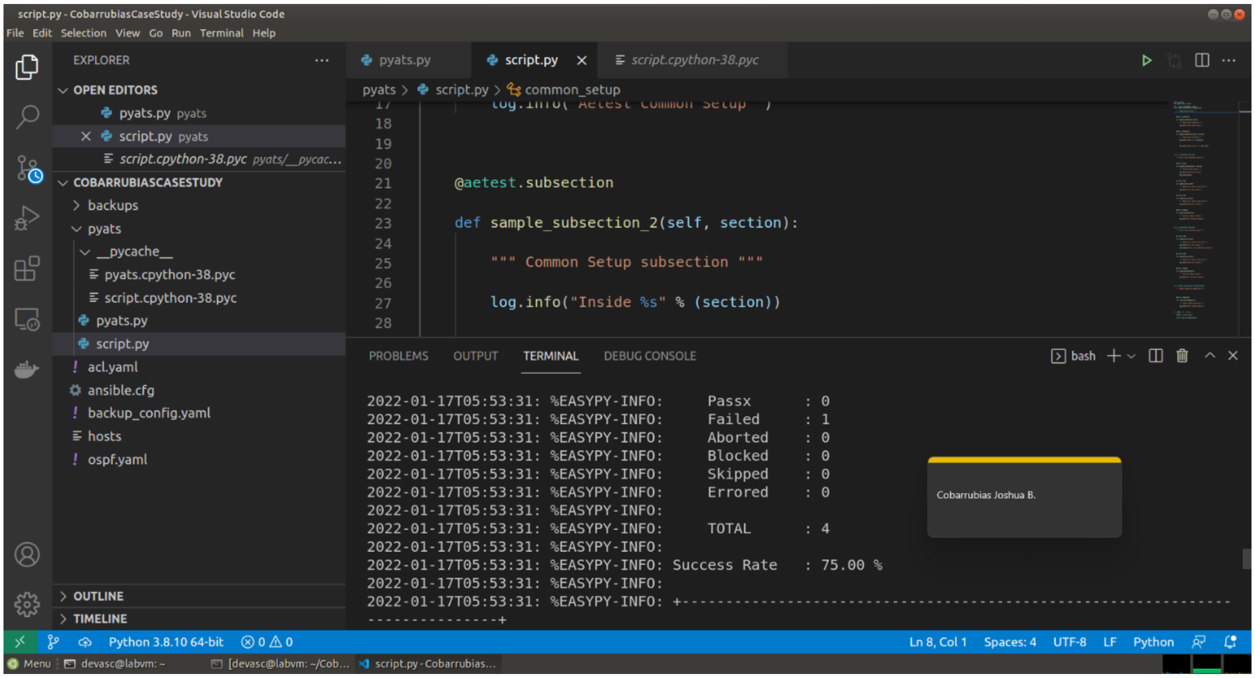
```
log = logging.getLogger(__name__)
class common_setup(aetest.CommonSetup):

    """ 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))

```

This will be the code for script.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)
```

Now after coding the script.py we will now test the scripty using python that would run a basic test. The command will be pyats run job/pyats_job.py

pyats run job pyats/pyats.py

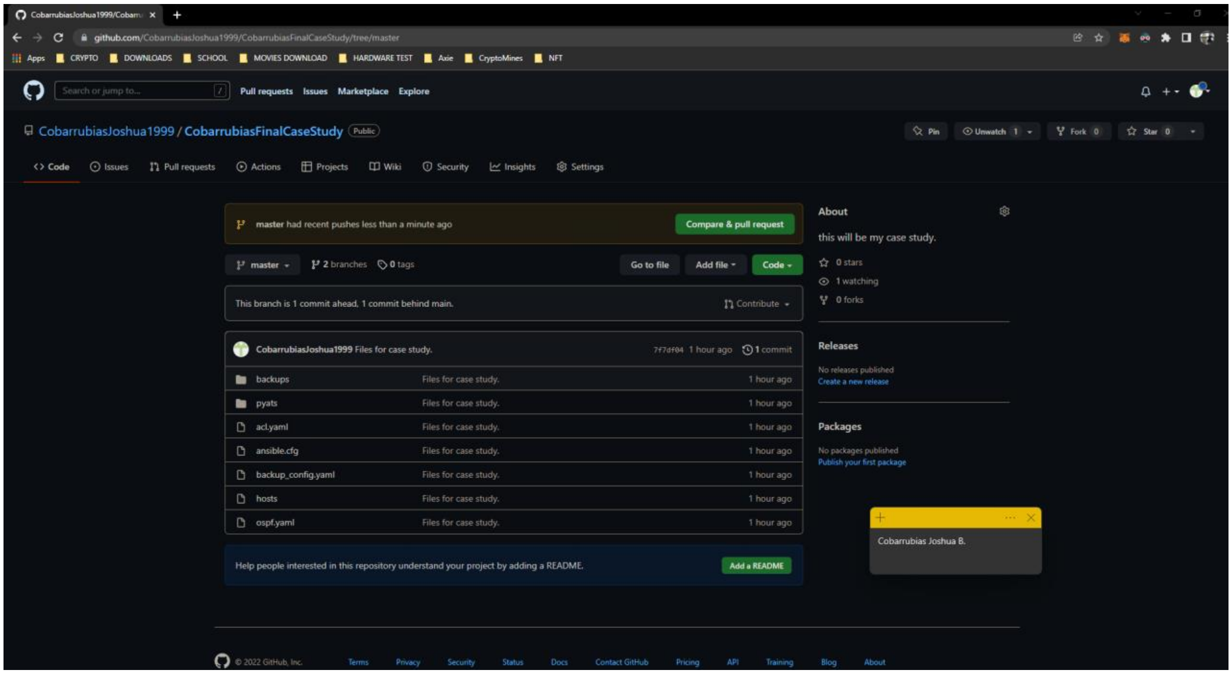


```
        log.info("Aetest Common Setup")


    @aetest.subsection

    def sample_subsection_2(self, section):

        """ Common Setup subsection """

        log.info("Inside %s" % (section))
```

```
2022-01-17T05:53:31: %EASYPY-INFO:     Passx      : 0
2022-01-17T05:53:31: %EASYPY-INFO:     Failed     : 1
2022-01-17T05:53:31: %EASYPY-INFO:     Aborted    : 0
2022-01-17T05:53:31: %EASYPY-INFO:     Blocked    : 0
2022-01-17T05:53:31: %EASYPY-INFO:     Skipped    : 0
2022-01-17T05:53:31: %EASYPY-INFO:     Errored    : 0
2022-01-17T05:53:31: %EASYPY-INFO:
2022-01-17T05:53:31: %EASYPY-INFO:     TOTAL      : 4
2022-01-17T05:53:31: %EASYPY-INFO:
2022-01-17T05:53:31: %EASYPY-INFO: Success Rate   : 75.00 %
2022-01-17T05:53:31: %EASYPY-INFO:
2022-01-17T05:53:31: %EASYPY-INFO: +------------------------------------------------------------------------------
---------------+
```

So as you can see it was a success of testing the network using our job and script by pyATS.

Now we will now create a repository to our GitHub and commit and push our work for our case study.



master had recent pushes less than a minute ago

This branch is 1 commit ahead, 1 commit behind main.

CobarrubiasJoshua1999 Files for case study. 7f7df04 1 hour ago 1 commit

backups	Files for case study.	1 hour ago
pyats	Files for case study.	1 hour ago
acl.yaml	Files for case study.	1 hour ago
ansible.cfg	Files for case study.	1 hour ago
backup_config.yaml	Files for case study.	1 hour ago
hosts	Files for case study.	1 hour ago
ospf.yaml	Files for case study.	1 hour ago

Help people interested in this repository understand your project by adding a README.

this will be my case study.

No releases published

No packages published

Conclusion:

In conclusion this case study shows us how to utilize an ansible for configuration backups, ACL implementation, and other tasks. OSPF is used on the network's router. The use of the fundamentals of ansible and understanding its significance is beneficial. Were automating the commands we use to configure our routers. Knowing how to put those into actions. Manually configuring network topics and using ansible will go a long way toward saving time when configuring a network. Finally, the laboratory exercise teaches us how to use pyATS and genie to test the network functionality. As well as whether the parameters are appropriately applied. And I also realized that all our activities or laboratory activities was helpful I based all my work in this case study on my past laboratories and implemented it here in my case study. I learned a lot in this class and will surely explore more so I can deeper my knowledge on network automation.

Links:

GitHub: https://github.com/CobarrubiasJoshua1999/CobarrubiasFinalCaseStudy.git

Honor Pledge for Graded Assignment

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