ESPONILLA, JONNEL M.
CPE41S1

Final Case Study – Network Automation and Programmability

Objectives

Part 1: Setup the GNS3 and apply basic configurations on routers and virtual machines

Part 2: Verify the SSH connections from DEVASC to R1, R2, and Ubuntu VM

Part 3: Create the ansible configurations files for OSPF, ACL, and Apache

Part 4: Run and the ansible configurations files for OSPF, ACL, and Apache

Part 5: Verify the connection from the webserver (Ubuntu VM)

Part 6: Test the network using pyATS

Background / Scenario

In this laboratory activity, I have created a network on the GNS3 that tackles three networking topics such as the network communication through SSH, configuring the OSPF and ACL of routers through ansible, and configuring the web server using ansible. The DEVASC Virtual Machine will act as the main computer while the Ubuntu Virtual Machine will act as the web server for this setup.

Required Resources

- Host computer with at least 4 GB of RAM and free disk space
- Virtual Box or VMware with the following virtual machine installed:
  - DEVASC Virtual Machine
  - Ubuntu 20.04.3 LTS
- GNS3

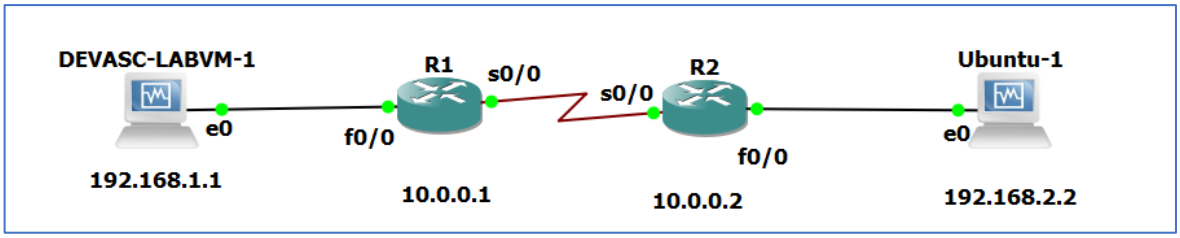
Instructions

Part 1: Setup the GNS3 and apply basic configurations on routers and virtual machines

Open the GNS3 and setup the network connection.

Step 1: Setup the network connection.

Below is the topology of the network connection used in this case study. I have used the DEVASC-LABVM as my main computer where I have deployed the configuration files and connected to other devices using SSH. The Ubuntu VM is used as a web server.



Step 2: Apply the basic configurations on the two routers.

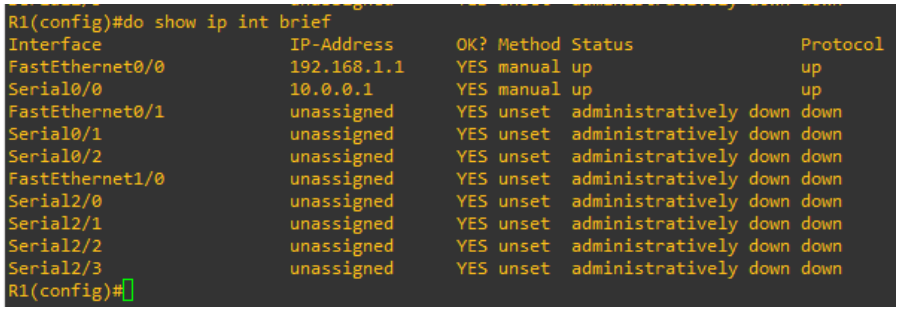
Apply basic configuration such as the username, password, secret, RSA, SSH, and IP addresses on each router.

The table below is the IP address of corresponding interfaces of each device.

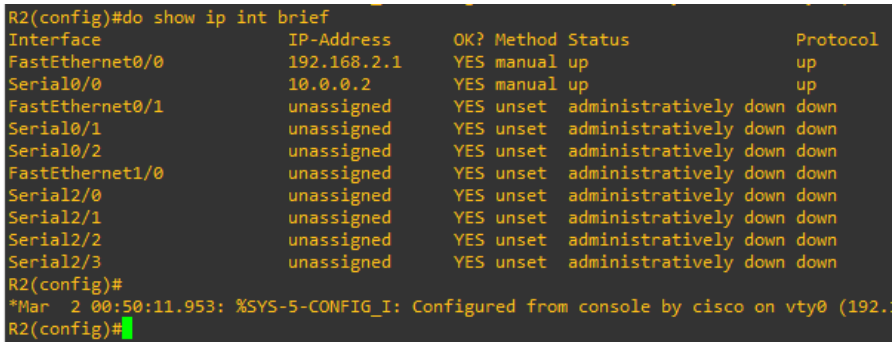
Devices	Interface	IP Address	Subnet Mask
DEVASC-LABVM	E0	192.168.1.1	255.255.255.0
Ubuntu	E0	192.168.2.2	255.255.255.0
R1	S0/0	10.0.0.1	255.255.255.252
	F0/0	192.168.1.1	255.255.255.0
R2	S0/0	10.0.0.2	255.255.255.252
	F0/0	192.168.2.1	255.255.255.0

Screenshot of IP configurations on devices

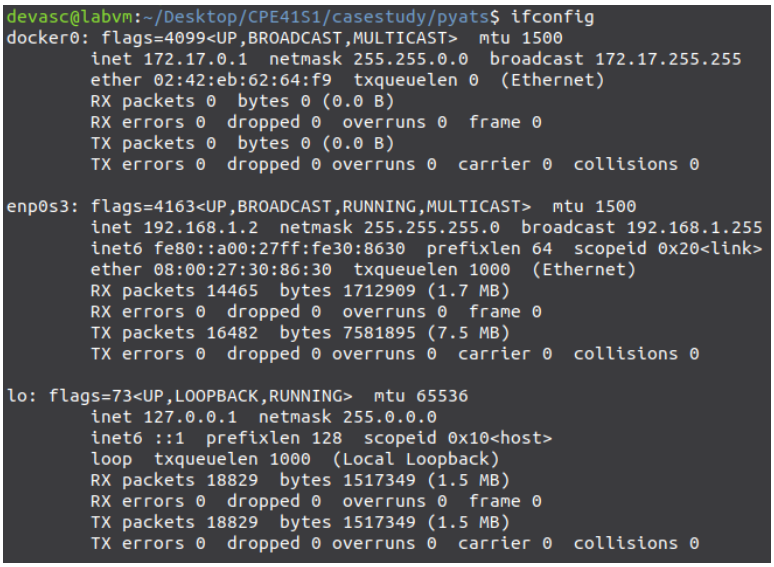
On R1:



On R2:



On DEVASC-LABVM:



On Ubuntu VM:

```
jelwxyz@jelwxyz:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.2  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::a00:27ff:fe14:7ef8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:14:7e:f8  txqueuelen 1000  (Ethernet)
        RX packets 66097  bytes 53129070 (53.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45452  bytes 5665168 (5.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 9631  bytes 798340 (798.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9631  bytes 798340 (798.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Part 2: Verify the SSH connections from DEVASC to R1, R2, and Ubuntu VM

In this part, we will verify if SSH connection can be established from DEVASC to other devices by opening the terminal on the DEVASC and executing SSH connections.

Step 1: Verify SSH connection from DEVASC to R1.

Open the terminal on DEVASC and run ***ssh cisco@10.0.0.1*** to establish an SSH connection from DEVASC to R1. If asked for a password, type ***cisco123*** and enter. After successfully entering on the router, type ***exit*** and enter.

```
devasc@labvm:~$ ssh cisco@10.0.0.1
Warning: Permanently added '10.0.0.1' (RSA) to the list of known hosts.
Password:

R1>exit
Connection to 10.0.0.1 closed.
```

Step 2: Verify SSH connection from DEVASC to R2.

Open the terminal on DEVASC and run ***ssh cisco@10.0.0.2*** to establish an SSH connection from DEVASC to R1. If asked for a password, type ***cisco123*** and enter. After successfully entering on the router, type ***exit*** and enter.

```
devasc@labvm:~$ ssh cisco@10.0.0.2
Warning: Permanently added '10.0.0.2' (RSA) to the list of known hosts.
Password:

R2>exit
Connection to 10.0.0.2 closed.
devasc@labvm:~$
```

Step 3: Verify SSH connection from DEVASC to Ubuntu.

Open the terminal on DEVASC and run ***ssh jelwxyz@192.168.2.2*** to establish an SSH connection from DEVASC to R1. If asked for a password, type ***cisco123*** and enter. After successfully entering on the router, type ***exit*** and enter.

```
devasc@labvm:~$ ssh jelwxyz@192.168.2.2
Warning: Permanently added '192.168.2.2' (ECDSA) to the list of known hosts.
jelwxyz@192.168.2.2's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

187 updates can be applied immediately.
112 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Jan 17 16:09:16 2022 from 192.168.1.2
jelwxyz@jelwxyz:~$ exit
logout
Connection to 192.168.2.2 closed.
```

Part 3: Create the ansible configurations files for OSPF, ACL, and Apache

In this part, we will create and run the ansible configuration files for configuring the ACL and OSPF of routers and the Apache using DEVASC.

Step 1: Create the ansible configuration file.

Open the VS Code and create a file with a name ***ansible.cfg***. After that, fill up the content with this code.

ansible.cfg

```
1  [defaults]
2  inventory= ./hosts
3  host_key_checking = False
4  retry_files_enabled = False
5  deprecation_warnings = False
```

Step 2: Create host file.

Open the VS Code and create a file with a name ***hosts***. After that, fill up the content with this code. This code contains information about the devices on the network (R1, R2, and Ubuntu) about their name, user, and passwords to access them.

hosts

```
[routers]
R1 ansible_host=10.0.0.1
R2 ansible_host=10.0.0.2

[routers:vars]
ansible_user=cisco
ansible_password=cisco123
ansible_connection=network_cli
ansible_network_os=ios
ansible_port=22
ansible_become=yes
ansible_become_method=enable
ansible_become_pass=cisco123

[ubuntu]
192.168.2.2
ansible_ssh_pass=12qwaszx
ansible_ssh_user=jelwxyz
ansible_password=12qwaszx
ansible_port=22
ansible_become=yes
ansible_sudo_pass=12qwaszx
```

Step 3: Create ACL YAML file.

Open the VS Code and create a file with a name ***acl.yaml***. After that, fill up the content with this code. This will be used to configure ACL on R2.

acl.yaml

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

  tasks:
    - name: Configure ACL inbound
      ios_config:
        lines:
          - ip access-group 110 in
        parents: interface FastEthernet0/0

    - name: Create inbound ACL rules for R2
      ios_config:
        lines:
          - access-list 110 deny icmp any any echo-reply
          - access-list 110 permit ip any any
        before: no access-list 110
        match: exact
```

Step 4: Create OSPF YAML file.

Open the VS Code and create a file with a name ***ospf.yaml***. After that, fill up the content with this code. This will be used to configure the OSPF on routers R1 and R2.

ospf.yaml

```yaml
---
- name: Configure OSPF on R1
  hosts: R1
  gather_facts: false
  connection: local

  vars:
    cli:
      username: cisco
      password: cisco123

  tasks:
    - name: Enable OSPF on R1
      ios_config:
        provider: "{{ cli }}"
        authorize: yes
        parents: router ospf 1
        lines:
          - network 0.0.0.0 255.255.255.255 area 0

      register: print_output
    - debug: var=print_output

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

  vars:
    cli:
      username: cisco
      password: cisco123

  tasks:
    - name: Enable OSPF on R2
      ios_config:
        provider: "{{ cli }}"
        authorize: yes
        parents: router ospf 1
        lines:
          - network 0.0.0.0 255.255.255.255 area 0

      register: print_output
    - debug: var=print_output
```

Step 5: Create APACHE YAML file.

Open the VS Code and create a file with a name ***apache.yaml***. After that, fill up the content with this code. This will be used to enable Apache server to run and used the Ubuntu as the webserver. The html file included in this file will be used to display when accessing the webserver.

apache.yaml

```yaml
---
- name: Installing Apache2 Server and Updating the Firewall
  hosts: ubuntu

  tasks:
  - name: Installing Apache
    apt:
      name: apache2
      state: present

  - name: Starting apache2 service
    service:
      name: apache2
      state: started
      enabled: yes

  - name: Allowing access to TCP Port 80
    ufw:
      rule: allow
      port: '80'
      proto: tcp

  - name: Enabling firewall
    ufw:
      state: enabled
```

```
- name: Creating the website
  hosts: ubuntu

  tasks:
  - name: Creating HTML test file
    copy:
      src: ./body.html
      dest: /var/www/html/index.html
      owner: root
      group: root
      mode: '0644'
    register: index

  - name: Starting the web service
    service:
      name: apache2
      state: restarted
      enabled: yes
    when: index.changed

  - name: Allowing access to TCP Port 80
    ufw:
      rule: allow
      port: '80'
      proto: tcp

  - name: Reloading ufw
    ufw:
      state: reloaded
```

Step 6: Create HTML test file.

Open the VS Code and create a file with a name ***body.html***. After that, fill up the content with this code. This will be used as the index html file when the user is accessing the web server though Apache.

```
body.html > body
<title>Jonnel Esponilla</title>
<body>
    <h1>Sample Website</h1>
    <p>This is a sample website. This is a sample paragraph.</p>
```

Part 4: Run the ansible configurations files for OSPF, ACL, and Apache

In this part, we will run the created configuration YAML files using ansible on the DEVASC.

Step 1: Run ACL YAML file configuration.

On the VS Code terminal or any terminal, type ***ansible-playbook -v acl.yaml*** and enter to run to configuration file.

```
devasc@labvm:~/Desktop/CPE41S1/casestudy$ ansible-playbook -v acl.yaml
Using /home/devasc/Desktop/CPE41S1/casestudy/ansible.cfg as config file
```

[some outputs are omitted to save space, below is the latest line of output]

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

Step 2: Run OSPF YAML file configuration.

On the VS Code terminal or any terminal, type ***ansible-playbook -v ospf.yaml*** and enter to run to configuration file.

```
devasc@labvm:~/Desktop/CPE41S1/casestudy$ ansible-playbook -v ospf.yaml
Using /home/devasc/Desktop/CPE41S1/casestudy/ansible.cfg as config file
```

[some outputs are omitted to save space, below is the latest line of output]

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

Step 3: Run APACHE YAML file configuration.

On the VS Code terminal or any terminal, type ***ansible-playbook -v apache.yaml*** and enter to run to configuration file.

```
devasc@labvm:~/Desktop/CPE41S1/casestudy$ ansible-playbook -v apache.yaml
Using /home/devasc/Desktop/CPE41S1/casestudy/ansible.cfg as config file
```

[some outputs are omitted to save space, below is the latest line of output]

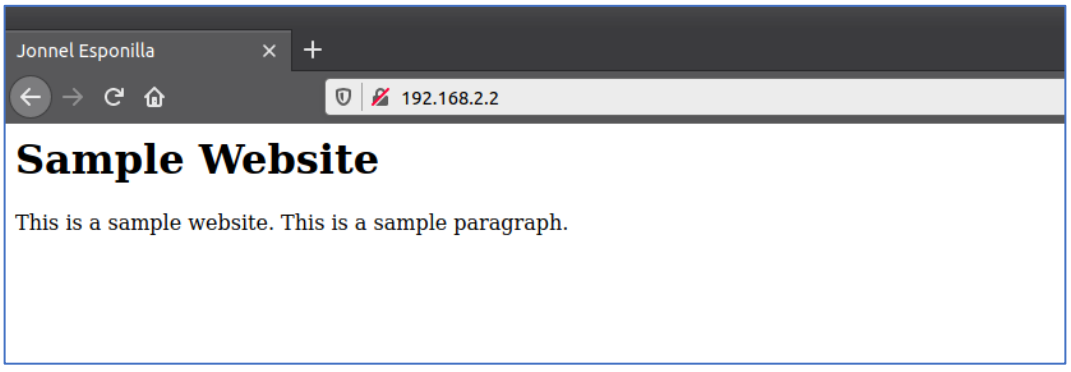
```
PLAY RECAP *********************************************************************************************************
192.168.2.2                : ok=9    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Part 5: Verify the connection from the webserver (Ubuntu VM)

In this part, we will verify if the webserver (Ubuntu VM) is accessible on the web browser of DEVASC-LABVM through Apache and if the Apache configuration is successfully displaying the HTML test file that we have created.

Step 1: Open a web browser on DEVASC and connect to the Web server.

Open any browser on the DEVASC-LABVM and type on the address the IP address of the Web server, which is ***192.168.2.2***. The browser should display the created HTML test file.



Part 6: Test the network using pyATS

In this part, we will test the network through running pyATS test file.

Step 1: Create a testbed file.

a) Go to the directory where the configuration files are store and create a python virtual environment with a name ***pyats*** using the command ***python3 -m venv pyats*** on a terminal.
b) Activate the created virtual environment by typing ***source bin/activate*** on the terminal and enter.
c) Once activated and inside, install the pyATS by typing ***pip install pyats.contrib*** on the terminal and enter. The installation of pyATS shall proceed.
d) Create a ***tesbed*** file using the command ***genie create testbed interactive –ouput nameoftestbed.yaml***. Change the ***nameofttestbed*** to a desired name of the file. Fill up the needed information about the devices and the testbed fille shall be created after. Below is my testbed file that I have created with a name ***esponilla.yaml.***

pyats > ! esponilla.yaml

```yaml
devices:
  R1:
    connections:
      cli:
        ip: 10.0.0.1
        protocol: ssh
    credentials:
      default:
        password: cisco123
        username: cisco
      enable:
        password: cisco123
    os: ios
    type: ios
  R2:
    connections:
      cli:
        ip: 10.0.0.2
        protocol: ssh
    credentials:
      default:
        password: cisco123
        username: cisco
      enable:
        password: cisco123
    os: ios
    type: ios
```

Step 2: Use the created testbed file to run several tests on the network.

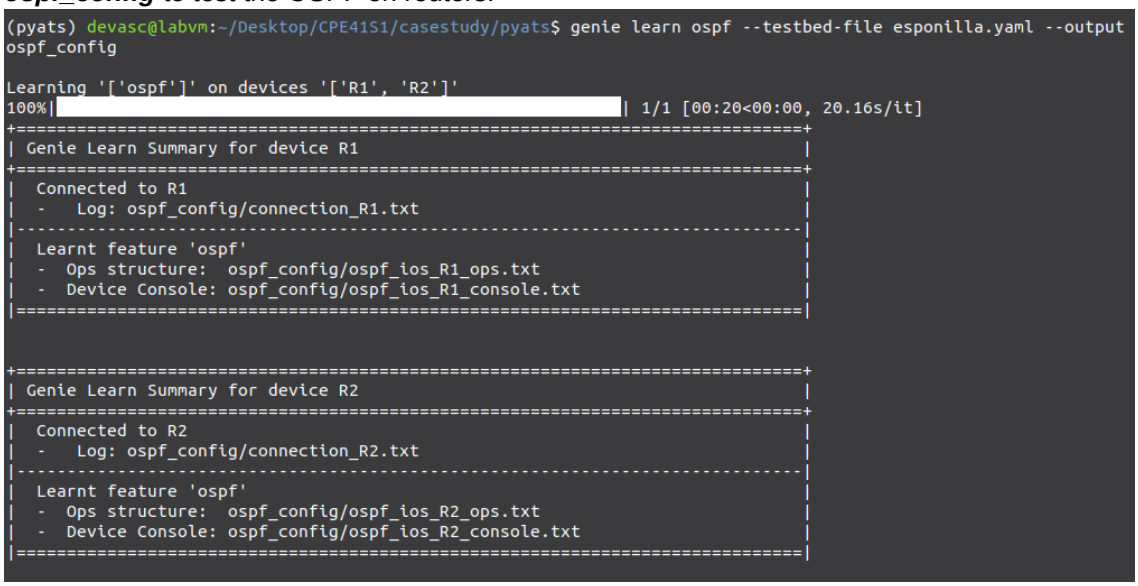
a) On the terminal, run the command ***genie parse "show version" –testbed-file esponilla.yaml –output show_version*** to show the version of the routers.

```
(pyats) devasc@labvm:~/Desktop/CPE41S1/casestudy/pyats$ genie parse "show version" --testbed-file esponilla.yaml
 --output show_version
100%|                                                                                | 1/1 [00:00<00:00,  1.21it/s]
+==============================================================================+
| Genie Parse Summary for R1                                                   |
+==============================================================================+
|  Connected to R1                                                             |
|   -  Log: show_version/connection_R1.txt                                     |
|------------------------------------------------------------------------------|
|  Parsed command 'show version'                                               |
|   -  Parsed structure: show_version/R1_show-version_parsed.txt               |
|   -  Device Console:   show_version/R1_show-version_console.txt              |
|------------------------------------------------------------------------------|
100%|                                                                                | 1/1 [00:00<00:00,  3.66it/s]
+==============================================================================+
| Genie Parse Summary for R2                                                   |
+==============================================================================+
|  Connected to R2                                                             |
|   -  Log: show_version/connection_R2.txt                                     |
|------------------------------------------------------------------------------|
|  Parsed command 'show version'                                               |
|   -  Parsed structure: show_version/R2_show-version_parsed.txt               |
|   -  Device Console:   show_version/R2_show-version_console.txt              |
|------------------------------------------------------------------------------|
```

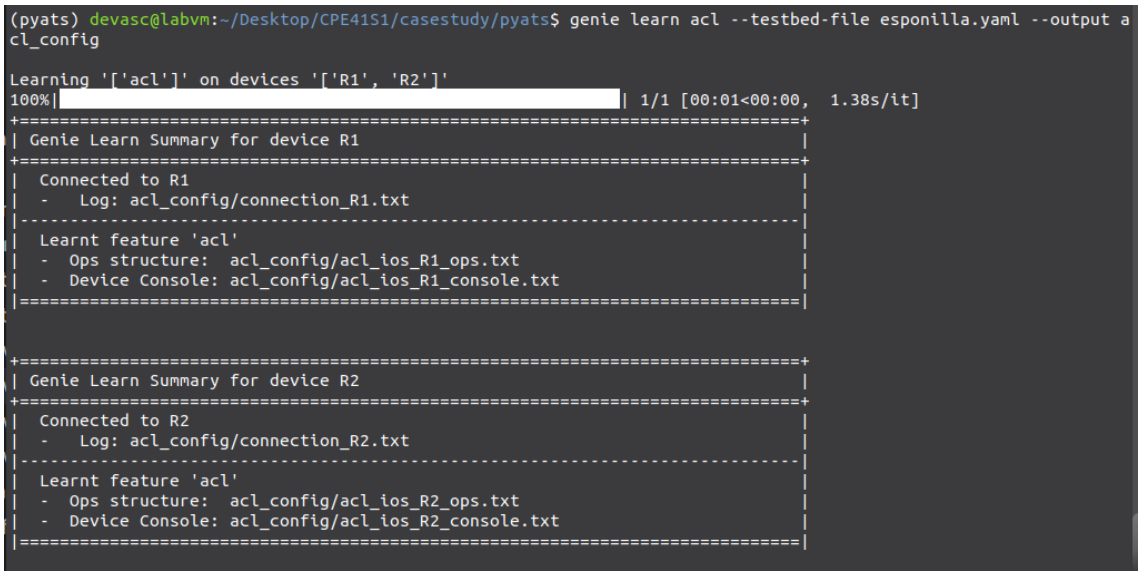
b) On the terminal, run the command ***genie parse "show ip interface brief" –testbed-file esponilla.yaml –output show_running_configuration*** to show the configurations of the routers.

```
(pyats) devasc@labvm:~/Desktop/CPE41S1/casestudy/pyats$ genie parse "show ip interface brief" --testbed-file esp
onilla.yaml --output show_running-configuration
100%|                                                                                | 1/1 [00:01<00:00,  1.16s/it]
+==============================================================================+
| Genie Parse Summary for R1                                                   |
+==============================================================================+
|  Connected to R1                                                             |
|   -  Log: show_running-configuration/connection_R1.txt                       |
|------------------------------------------------------------------------------|
|  Parsed command 'show ip interface brief'                                    |
|   -  Parsed structure: show_running-configuration/R1_show-ip-interface-      |
| brief_parsed.txt                                                             |
|   -  Device Console:   show_running-configuration/R1_show-ip-interface-      |
| brief_console.txt                                                            |
|------------------------------------------------------------------------------|
100%|                                                                                | 1/1 [00:00<00:00,  2.68it/s]
+==============================================================================+
| Genie Parse Summary for R2                                                   |
+==============================================================================+
|  Connected to R2                                                             |
|   -  Log: show_running-configuration/connection_R2.txt                       |
|------------------------------------------------------------------------------|
|  Parsed command 'show ip interface brief'                                    |
|   -  Parsed structure: show_running-configuration/R2_show-ip-interface-      |
| brief_parsed.txt                                                             |
|   -  Device Console:   show_running-configuration/R2_show-ip-interface-      |
| brief_console.txt                                                            |
|------------------------------------------------------------------------------|
```

c) On the terminal, run the command ***genie learn ospf –testbed-file Esponilla --output ospf_config*** to test the OSPF on routers.



d) On the terminal, run the command ***genie learn acl –testbed-file Esponilla --output acl_config*** to test the ACL on routers.



e) To exit the virtual environment, type on the terminal ***deactivate*** and enter.