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

**Final Case Study | Network Automation and Programmability**

**Diagram

Description automatically generated**

**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.

Graphical user interface

Description automatically generated

**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

Graphical user interface, application

Description automatically generated

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]

Graphical user interface

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

In the config file type this:

Host \*

Port 22

User cisco

StrictHostKeyChecking=no

UserKnownHostsFile=/dev/null

KexAlgorithms +diffie-hellman-group1-sha1

Ciphers 3des-cbc

Graphical user interface, text

Description automatically generated

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](mailto:cisco@192.168.10.62)

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, chat or text message

Description automatically generated

**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

Graphical user interface, text

Description automatically generated

**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

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text, application, Teams

Description automatically generated

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

Graphical user interface

Description automatically generated

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

Graphical user interface, text

Description automatically generated

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"

Graphical user interface, text

Description automatically generated

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

Text

Description automatically generated

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)

**A screenshot of a computer

Description automatically generated**

**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)

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated

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."**