

Technological Institute of the Philippines
CPE 028 - Developing Applications and Automation
Final Case Study

Network Automation

Lab – Use Ansible for Network Automation

Objectives

- Part 1: Launch and Check the Configurations of GNS3 and Virtual Machines**
- Part 2: Build the given topology**
- Part 3: Implementing basic configurations to the devices**
- Part 4: Set-up the Ansible**
- Part 5: Configure and implement Open Shortest Path First (OSPF) routing protocol**
- Part 6: Configure and implement AAA security to each router**
- Part 7: Configure and implement Access Control List (ACL) to PC_Manager and PC_Supervisor**
- Part 8: Complete Source Code of YAML**
- Part 9: Check and test the network configurations**
- Part 10: Uploading the file to GitHub**

Background / Scenario

In this Case Study, you will build the given topology and apply basic configurations to the devices. After configuring all the basic configurations, you will need to implement three different topics such as Open Shortest Path First (OSPF), AAA login authentication, and Access Control List (ACL). Those three network configurations will be implemented using ansible to represent the process of network automation.

Required Resources:

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

Addressing Table:

Device	Interface	IP Address	Subnet Mask
R1	F0/1	172.168.1.1	255.255.255.192
	F0/0	192.168.1.1	255.255.255.0
	S0/2	10.3.1.1	255.255.255.252
	S0/0	10.1.1.1	255.255.255.252

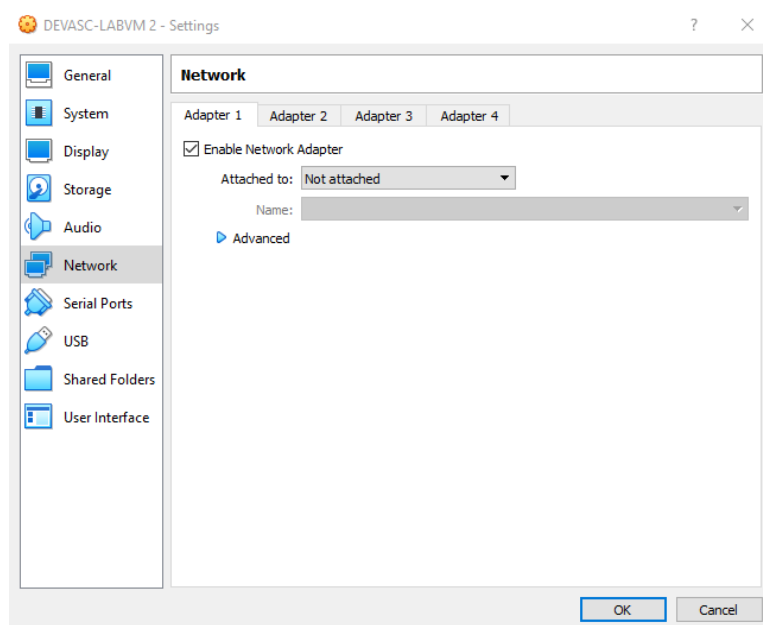
R2	S0/0	10.1.1.2	255.255.255.252
	S0/1	10.2.1.1	255.255.255.252
R3	S0/2	10.3.1.2	255.255.255.252
	S0/1	10.2.1.2	255.255.255.252
	F1/0	172.169.3.1	255.255.255.192
	F0/1	172.168.2.1	255.255.255.192
PC_MANAGER	NIC	172.168.1.2	255.255.255.192
PC_ADMIN	NIC	192.168.1.2	255.255.255.0
PC_EMPLOYEE	NIC	172.169.3.2	255.255.255.192
PC_SUPERVISOR	NIC	172.168.2.2	255.255.255.192

Procedures:

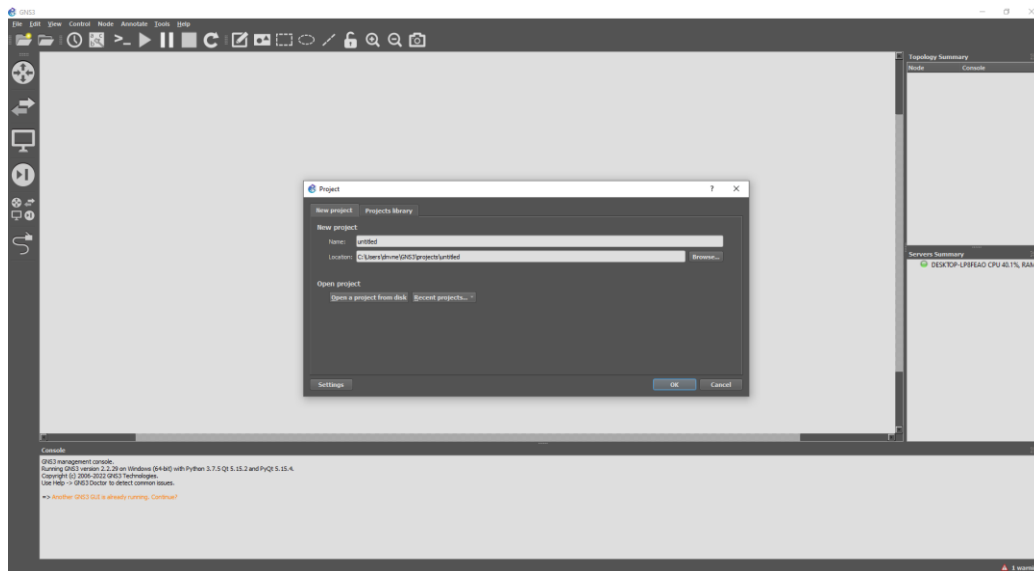
Part 1: Launch and Setup the Configurations of GNS3 and Virtual Machines

Step 1: If you have not already completed the Lab - Install the Virtual Machine Lab Environment, do so now. If you have already completed that lab, launch the DEVASC VM now.

Note: Make sure that the network adapter is set to "Not Attached". Because you will not be able to connect it to the routers in GNS3 if it is attached to a specific adapter.

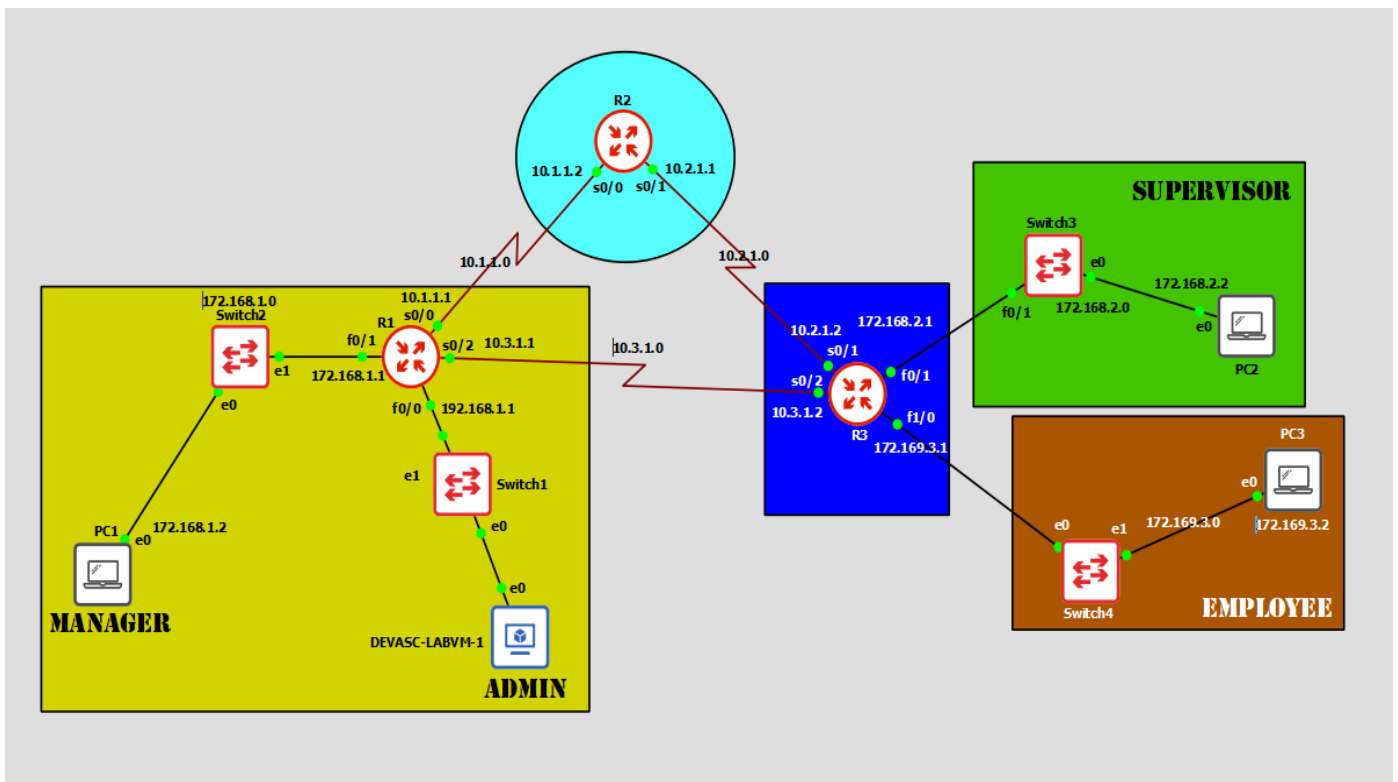


Step 2: Launch GNS3 to build a topology



Part 2: Build the given topology

1. Build the given topology using GNS3 and c3725 routers.



Part 3: Implementing basic configurations to the devices

Applying basic configurations to the routers must be well configured. Possible errors may occur if the configurations are not managed properly. **Refer to the addressing table above for the addresses.**

a. R1 configurations

```

R1#
R1#
R1#en
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#hostname R1
R1(config)#no ip domain-lookup
R1(config)#enable secret class
R1(config)#service password-encryption
R1(config)#banner motd "Unauthorized Access is Prohibited"
R1(config)#line con 0
R1(config-line)#password cisco
R1(config-line)#login
R1(config-line)#line vty 0 15
R1(config-line)#password cisco
R1(config-line)#login
R1(config-line)#int f0/0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#
*Mar 1 00:07:35.663: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar 1 00:07:36.663: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#int s0/0
R1(config-if)#ip address 10.1.1.1 255.255.255.252
R1(config-if)#no shut
R1(config-if)#
*Mar 1 00:08:08.995: %LINK-3-UPDOWN: Interface Serial0/0, changed state to up
R1(config-if)#
*Mar 1 00:08:09.999: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
R1(config-if)#do copy run st
Destination filename [startup-config]?
Building configuration...
[OK]
R1(config-if)#
*Mar 1 00:08:32.627: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
R1(config-if)#
*Mar 1 00:10:32.619: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
R1(config-if)#

```

b. R2 configurations

```

R2#
R2#en
R2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#hostname R2
R2(config)#no ip domain-lookup
R2(config)#enable secret class
R2(config)#service password-encryption
R2(config)#banner motd "Unauthorized Access is Prohibited"
R2(config)#line con 0
R2(config-line)#password cisco
R2(config-line)#login
R2(config-line)#line vty 0 15
R2(config-line)#password cisco
R2(config-line)#login
R2(config-line)#int s0/0
R2(config-if)#ip address 10.1.1.2 255.255.255.252
R2(config-if)#no shut
R2(config-if)#
*Mar 1 00:06:33.663: %LINK-3-UPDOWN: Interface Serial0/0, changed state to up
R2(config-if)#
*Mar 1 00:06:34.667: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
R2(config-if)#int s0/1
R2(config-if)#ip address 10.2.2.1 255.255.255.252
R2(config-if)#no shut
R2(config-if)#
*Mar 1 00:06:55.175: %LINK-3-UPDOWN: Interface Serial0/1, changed state to up
R2(config-if)#
*Mar 1 00:06:56.179: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to up
R2(config-if)#do copy run st
Destination filename [startup-config]?
Building configuration...
[OK]
R2(config-if)#
*Mar 1 00:07:22.811: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to down
R2(config-if)#
*Mar 1 00:08:52.811: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to up
R2(config-if)#

```

c. R3 configurations

```

R3#
R3#
R3#en
R3#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R3(config)#hostname R3
R3(config)#no ip domain-lookup
R3(config)#enable secret class
R3(config)#service password-encryption
R3(config)#banner motd "Unauthorized Access is Prohibited"
R3(config)#line con 0
R3(config-line)#password cisco
R3(config-line)#login local
R3(config-line)#line vty 0 15
R3(config-line)#password cisco
R3(config-line)#login local
R3(config-line)#int s0/1
R3(config-if)#ip address 10.2.2.1 255.255.255.252
R3(config-if)#no shut
R3(config-if)#
*Mar 1 00:05:34.699: %LINK-3-UPDOWN: Interface Serial0/1, changed state to up
R3(config-if)#
*Mar 1 00:05:35.703: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1, changed state to up
R3(config-if)#exit
R3(config)#do copy r s
Destination filename [startup-config]?
Building configuration...
[OK]
R3(config)#ip route 0.0.0.0 0.0.0.0 10.2.1.1
R3(config)#do sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        o - ODR, P - periodic downloaded static route

```

d. PC-1 Configuration

```

PC1> ping 10.3.2.1
host (10.3.2.1) not reachable

PC1> ip 172.168.1.2 255.255.255.192 172.168.1.1
Checking for duplicate address...
PC1 : 172.168.1.2 255.255.255.192 gateway 172.168.1.1

PC1> ping 172.169.3.2
172.169.3.2 icmp_seq=1 timeout
172.169.3.2 icmp_seq=2 timeout
172.169.3.2 icmp_seq=3 timeout
172.169.3.2 icmp_seq=4 timeout
172.169.3.2 icmp_seq=5 timeout

PC1> save
Saving startup configuration to startup.vpc
. done

PC1> ping 172.169.3.2
172.169.3.2 icmp_seq=1 timeout
172.169.3.2 icmp_seq=2 timeout
84 bytes from 172.169.3.2 icmp_seq=3 ttl=62 time=14.902 ms
84 bytes from 172.169.3.2 icmp_seq=4 ttl=62 time=8.977 ms
84 bytes from 172.169.3.2 icmp_seq=5 ttl=62 time=11.029 ms

PC1> ping 172.169.3.2
84 bytes from 172.169.3.2 icmp_seq=1 ttl=62 time=19.818 ms
84 bytes from 172.169.3.2 icmp_seq=2 ttl=62 time=9.174 ms
84 bytes from 172.169.3.2 icmp_seq=3 ttl=62 time=10.010 ms
84 bytes from 172.169.3.2 icmp_seq=4 ttl=62 time=17.752 ms
84 bytes from 172.169.3.2 icmp_seq=5 ttl=62 time=16.727 ms

PC1> save
Saving startup configuration to startup.vpc
. done

PC1>

```

e. PC-2 Configuration

```

R3 R2 R1 PC x PC3 PC1
Copyright (c) 2007-2014, Paul Meng (mirnshi@gmail.com)
All rights reserved.

VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file

PC2> ip 172.168.2.2 255.255.255.192 172.168.2.1
Checking for duplicate address...
PC1 : 172.168.2.2 255.255.255.192 gateway 172.168.2.1

PC2> save
Saving startup configuration to startup.vpc
. done

PC2> show ip

NAME       : PC2[1]
IP/MASK    : 172.168.2.2/26
GATEWAY    : 172.168.2.1
DNS        :
MAC        : 00:50:79:66:68:01
LPORT      : 10032
RHOST:PORT : 127.0.0.1:10033
MTU        : 1500

PC2> ping 172.168.1.2
84 bytes from 172.168.1.2 icmp_seq=1 ttl=62 time=20.747 ms
84 bytes from 172.168.1.2 icmp_seq=2 ttl=62 time=19.830 ms
84 bytes from 172.168.1.2 icmp_seq=3 ttl=62 time=21.228 ms
84 bytes from 172.168.1.2 icmp_seq=4 ttl=62 time=11.923 ms
84 bytes from 172.168.1.2 icmp_seq=5 ttl=62 time=12.658 ms

PC2>
solarwinds | Solar-PuTTY free tool © 2019 SolarWinds Worldwide, LLC. All rights reserved.

```

f. PC-3 Configuration

```

R3 R2 R1 PC2 PC x PC1
Welcome to Virtual PC Simulator, version 0.6.2
Dedicated to Daling.
Build time: Apr 10 2019 02:42:20
Copyright (c) 2007-2014, Paul Meng (mirnshi@gmail.com)
All rights reserved.

VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file

PC3> ping 172.168.1.2
host (172.168.1.2) not reachable

PC3> ip 172.169.3.2 255.255.255.192 172.169.3.1
Checking for duplicate address...
PC1 : 172.169.3.2 255.255.255.192 gateway 172.169.3.1

PC3> save
Saving startup configuration to startup.vpc
. done

PC3> ping 172.169.3.2
172.169.3.2 icmp_seq=1 ttl=64 time=0.001 ms
172.169.3.2 icmp_seq=2 ttl=64 time=0.001 ms
172.169.3.2 icmp_seq=3 ttl=64 time=0.001 ms
172.169.3.2 icmp_seq=4 ttl=64 time=0.001 ms
172.169.3.2 icmp_seq=5 ttl=64 time=0.001 ms

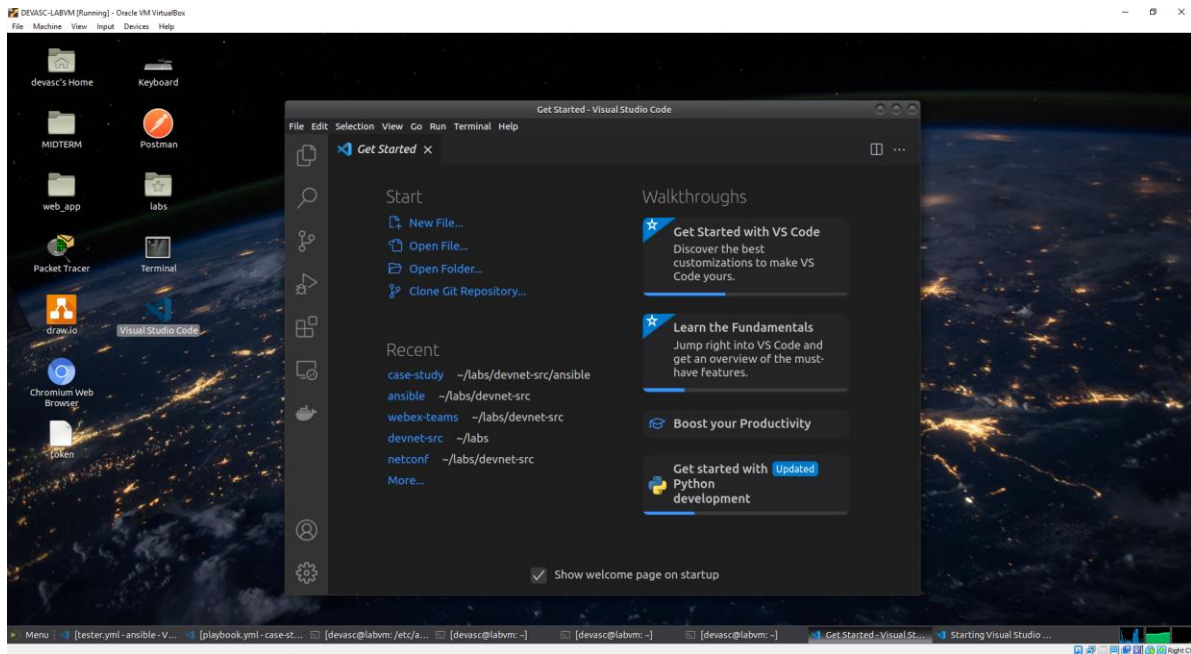
PC3> save
Saving startup configuration to startup.vpc
. done

PC3>
solarwinds | Solar-PuTTY free tool © 2019 SolarWinds Worldwide, LLC. All rights reserved.

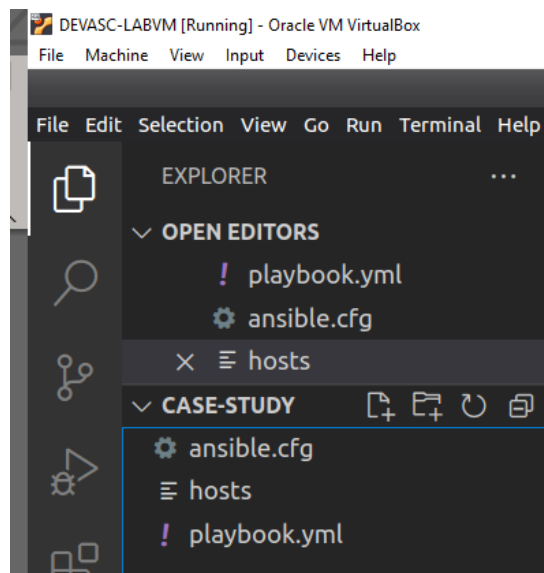
```

Part 4: Set-up the Ansible

Step 1: Go to DEVASC Virtual Machine and open VS Code.

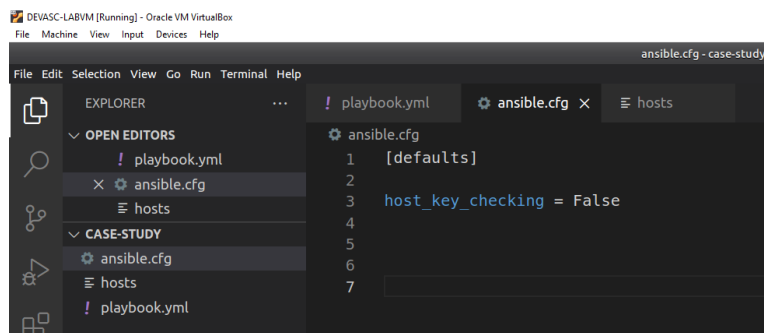


Step 2: Create a folder name it "CASE-STUDY".

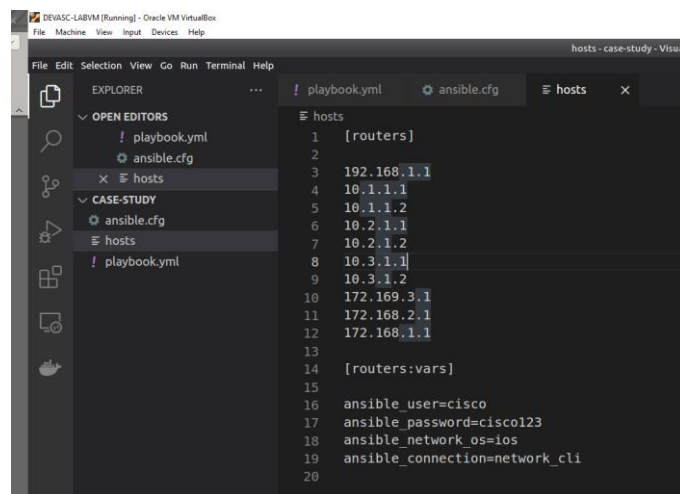


Step 3: Create files below inside of the CASE-STUDY folder.

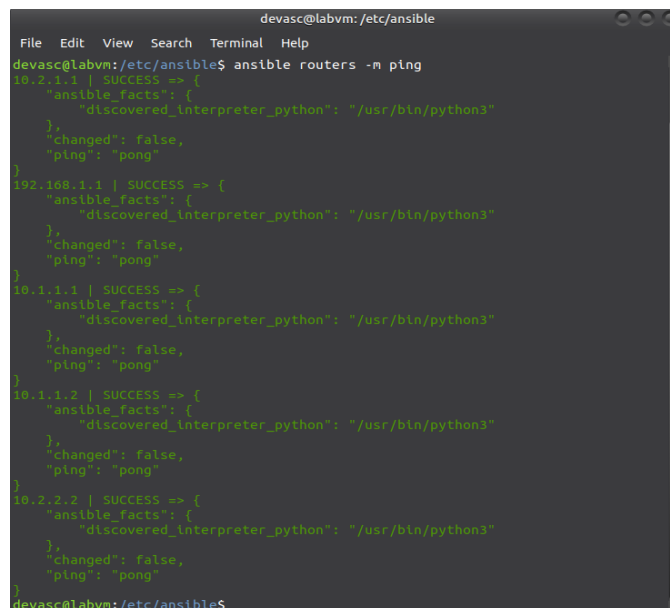
a. ansible.cfg – host_key_checking = False



b. hosts – IP addresses of the routers



Step 4: Ping the routers from the hosts to check if the configurations are correct.



Note: If the output displayed “ping”: “pong”, it means that ping is successful, and configurations are correct. If not, you need to check your configurations and set-up.

Part 5: Configure and implement Open Shortest Path First (OSPF) routing protocol

You will need to complete the steps above before proceeding to this section. Failure to implement basic configuration on routers and set-up for the ansible will not be able to do this part. **Implementing OSPF is the first topic for this network automation** that will be deployed to the routers using Ansible. OSPF is used to create a routing protocol as well as determine the open shortest path first of the router to avoid network traffic.

Step 1: In the folder that we created on the VS Code, create a YAML file and name the file as “playbook.yml”. The playbook.yml will be the container of three topics source codes.

a. Copy the source code below to playbook.yml file to implement the OSPF configuration.

```
---
#OSPF
- hosts: routers
gather_facts: no
connection: network_cli
become_method: enable

tasks:
- name: Configuring OSPF for R1....
  when: ansible_host == "10.1.1.1"
  ios_config:
    parents: router ospf 100
    lines:
      - router-id 1.1.1.1
      - network 10.1.1.0 0.0.0.3 area 0
      - network 192.168.1.0 0.0.0.255 area 0
      - network 172.168.1.0 0.0.0.63 area 0
      - network 10.3.1.0 0.0.0.3 area 0
      - passive-interface FastEthernet0/0
      - passive-interface FastEthernet0/1

- name: Configuring OSPF for R2....
  when: ansible_host == "10.2.1.1"
  ios_config:
    parents: router ospf 100
    lines:
      - router-id 2.2.2.2
```

- network 10.2.1.0 0.0.0.3 area 0
- network 10.1.1.0 0.0.0.3 area 0
- default-information originate

- name: Configuring OSPF for R3....

when: ansible_host == "10.2.1.2"

ios_config:

parents: router ospf 100

lines:

- router-id 3.3.3.3
- passive-interface f0/1
- passive-interface f1/0
- network 10.2.1.0 0.0.0.3 area 0
- network 172.168.2.0 0.0.0.63 area 0
- network 172.169.3.0 0.0.0.63 area 0
- network 10.3.1.0 0.0.0.3 area 0

b. Save the file and run it using the command below. Take note that the BECOME password is **class**. This is the password of enable that we set in the basic configuration from the previous steps.

devasc@labvm:~/labs/devnet-src/ansible/case-study\$ **ansible-playbook -i hosts playbook.yml -K -b**

BECOME password:

PLAY [routers] *****

TASK [Configuring OSPF for R1....] *****

skipping: [10.2.1.1]

skipping: [192.168.1.1]

skipping: [10.1.1.2]

skipping: [10.2.1.2]

skipping: [10.3.1.1]

skipping: [10.3.1.2]

skipping: [172.169.3.1]

skipping: [172.168.2.1]

skipping: [172.168.1.1]

ok: [10.1.1.1]

TASK [Configuring OSPF for R2....] *****

skipping: [192.168.1.1]

skipping: [10.1.1.2]
skipping: [10.1.1.1]
skipping: [10.2.1.2]
skipping: [10.3.1.1]
skipping: [172.168.2.1]
skipping: [10.3.1.2]
skipping: [172.169.3.1]
skipping: [172.168.1.1]
ok: [10.2.1.1]

TASK [Configuring OSPF for R3....] *****

skipping: [192.168.1.1]
skipping: [10.1.1.1]
skipping: [10.2.1.1]
skipping: [10.1.1.2]
skipping: [10.3.1.1]
skipping: [172.169.3.1]
skipping: [10.3.1.2]
skipping: [172.168.2.1]
skipping: [172.168.1.1]
changed: [10.2.1.2]

c. Verify that the OSPF configurations are successfully implemented to each router. To easily check the configuration, you can use the command **show run** and find the output below.

1. R1

```
!
router ospf 100
router-id 1.1.1.1
log-adjacency-changes
passive-interface FastEthernet0/0
passive-interface FastEthernet0/1
network 10.1.1.0 0.0.0.3 area 0
network 10.3.1.0 0.0.0.3 area 0
network 172.168.1.0 0.0.0.3 area 0
network 172.168.1.0 0.0.0.63 area 0
network 192.168.1.0 0.0.0.255 area 0
!
```

2. R2

```
router ospf 100
router-id 2.2.2.2
log-adjacency-changes
network 10.1.1.0 0.0.0.3 area 0
network 10.2.1.0 0.0.0.3 area 0
default-information originate
!
```

3. R3

```
router ospf 100
router-id 3.3.3.3
log-adjacency-changes
passive-interface FastEthernet0/1
passive-interface FastEthernet1/0
network 10.2.1.0 0.0.0.3 area 0
network 10.2.2.0 0.0.0.3 area 0
network 10.3.1.0 0.0.0.3 area 0
network 172.168.2.0 0.0.0.63 area 0
network 172.169.3.0 0.0.0.63 area 0
```

Part 6: Configure and implement AAA security to each router

You will need to complete the steps above before proceeding to this section. Failure to implement basic configuration on routers and set-up for the ansible will not be able to do this part. Implementing AAA security improves security framework of the routers. **AAA security is the second topic for this network automation using ansible.**

Step 1: Use the playbook.yml file that we created from the previous steps.

- a. Copy the source code below to configure and implement AAA security to the routers.

#AAA

- name: Configuring AAA login Authentication for console access...

ios_config:

commands:

- aaa new-model
- aaa authentication login default local

- name: Verifying the line console to implement the defined AAA authentication...

ios_config:

commands:

- login authentication default

parents: line console 0

- name: Allowing AAA login Authentication via SSH-LOGIN Local...

ios_config:

commands:

- aaa authentication login SSH-LOGIN local

- name: Configuring vty lines to the authentication phase...

ios_config:

commands:

- login authentication SSH-LOGIN

parents: line vty 0 4

b. Save the file and run it using the command below. Take note that the BECOME password is **class**. This is the password of enable that we set in the basic configuration from the previous steps.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ ansible-playbook -i hosts playbook.yml -K -b
```

BECOME password:

```
PLAY [routers] *****
```

```
TASK [Configuring OSPF for R1....] *****
```

```
skipping: [10.2.1.1]
```

```
skipping: [192.168.1.1]
```

```
skipping: [10.1.1.2]
```

```
skipping: [10.2.1.2]
```

```
skipping: [10.3.1.1]
```

```
skipping: [10.3.1.2]
```

```
skipping: [172.169.3.1]
```

```
skipping: [172.168.2.1]
```

```
skipping: [172.168.1.1]
```

```
ok: [10.1.1.1]
```

```
TASK [Configuring OSPF for R2....] *****
```

```
skipping: [192.168.1.1]
```

```
skipping: [10.1.1.2]
```

```
skipping: [10.1.1.1]
```

```
skipping: [10.2.1.2]
```

```
skipping: [10.3.1.1]
```

```
skipping: [172.168.2.1]
```

```
skipping: [10.3.1.2]
```

```
skipping: [172.169.3.1]
```

```
skipping: [172.168.1.1]
```

```
ok: [10.2.1.1]
```

```
TASK [Configuring OSPF for R3....] *****
```

```
skipping: [192.168.1.1]
```

```
skipping: [10.1.1.1]
```

```
skipping: [10.2.1.1]
```

```
skipping: [10.1.1.2]
```

```
skipping: [10.3.1.1]
```

skipping: [172.169.3.1]

skipping: [10.3.1.2]

skipping: [172.168.2.1]

skipping: [172.168.1.1]

changed: [10.2.1.2]

TASK [Configuring AAA login Authentication for console access...] *****

ok: [10.2.1.1]

ok: [10.1.1.1]

ok: [10.2.1.2]

ok: [10.1.1.2]

ok: [192.168.1.1]

ok: [10.3.1.2]

ok: [10.3.1.1]

ok: [172.169.3.1]

ok: [172.168.1.1]

ok: [172.168.2.1]

TASK [Verifying the line console to implement the defined AAA authentication...] *****

changed: [10.2.1.2]

changed: [10.1.1.2]

changed: [10.1.1.1]

changed: [192.168.1.1]

changed: [10.2.1.1]

changed: [10.3.1.1]

changed: [172.168.1.1]

changed: [10.3.1.2]

changed: [172.169.3.1]

changed: [172.168.2.1]

TASK [Allowing AAA login Authentication via SSH-LOGIN Local...] *****

changed: [10.2.1.2]

changed: [192.168.1.1]

changed: [10.2.1.1]

changed: [10.1.1.1]

changed: [10.1.1.2]

ok: [10.3.1.1]

ok: [10.3.1.2]
ok: [172.168.1.1]
ok: [172.169.3.1]
ok: [172.168.2.1]

TASK [Configuring vty lines to the authentication phase...] *****

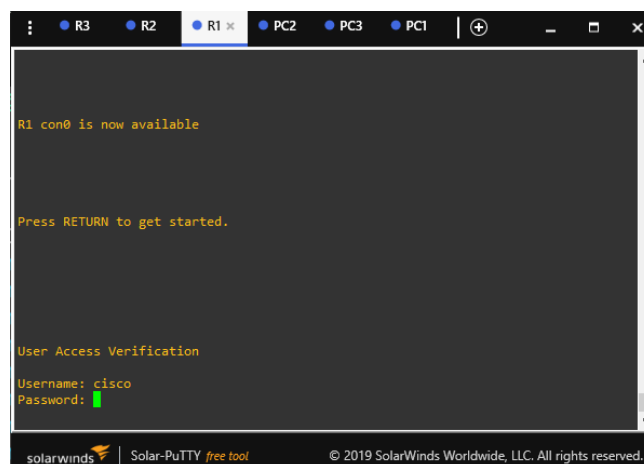
changed: [10.2.1.2]
changed: [10.1.1.2]
changed: [10.2.1.1]
ok: [10.3.1.1]
changed: [192.168.1.1]
changed: [10.1.1.1]
ok: [10.3.1.2]
ok: [172.168.1.1]
ok: [172.169.3.1]
ok: [172.168.2.1]

c. Verify that the OSPF configurations are successfully implemented to each router. To easily check the configuration, you can use the command **show run** and find the output below.

1. All routers must have the output below:

```
aaa new-model
!
!
aaa authentication login default local
aaa authentication login SSH-LOGIN local
!
```

2. In GNS3, exit the router then press enter. All the first screen of the routers must have User Access Verification since we just put AAA for the login console.



Part 7: Configure and implement Access Control List (ACL)

You will need to complete the steps above before proceeding to this section. Failure to implement basic configuration on routers and set-up for the ansible will not be able to do this part. The purpose of Access Control List (ACL) is to limit the access of an specific interface to other interfaces.

Step 1: Use the playbook.yml file that we created from the previous steps.

a. Copy the source code below to configure and implement Access control list to Manager's PC and Supervisor's PC.

#ACL

- name: Implementing Access List for Manager
when: ansible_host == "172.168.1.1"
ios_config:
 lines:
 - deny 172.169.3.0 0.0.0.63
 - permit any
 parents: ip access-list standard PC_MANAGER

- name: Implementing access group on Interface F0/1
when: ansible_host == "172.168.1.1"
ios_config:
 lines:
 - int f0/1
 - ip access-group PC_MANAGER out

- name: Implementing Access List for SUPERVISOR
when: ansible_host == "172.168.2.1"
ios_config:
 lines:
 - deny 172.169.3.0 0.0.0.63
 - permit any
 parents: ip access-list standard PC_SUPERVISOR

- name: Implementing access group on Interface F0/1
when: ansible_host == "172.168.2.1"
ios_config:
 lines:
 - int f0/1
 - ip access-group PC_SUPERVISOR out

b. Save the file and run it using the command below. Take note that the BECOME password is **class**. This is the password of enable that we set in the basic configuration from the previous steps.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ ansible-playbook -i hosts playbook.yml -K -b  
BECOME password:
```

```
PLAY [routers] *****
```

```
TASK [Configuring OSPF for R1....] *****
```

```
skipping: [10.2.1.1]  
skipping: [192.168.1.1]  
skipping: [10.1.1.2]  
skipping: [10.2.1.2]  
skipping: [10.3.1.1]  
skipping: [10.3.1.2]  
skipping: [172.169.3.1]  
skipping: [172.168.2.1]  
skipping: [172.168.1.1]  
ok: [10.1.1.1]
```

```
TASK [Configuring OSPF for R2....] *****
```

```
skipping: [192.168.1.1]  
skipping: [10.1.1.2]  
skipping: [10.1.1.1]  
skipping: [10.2.1.2]  
skipping: [10.3.1.1]  
skipping: [172.168.2.1]  
skipping: [10.3.1.2]  
skipping: [172.169.3.1]  
skipping: [172.168.1.1]  
ok: [10.2.1.1]
```

```
TASK [Configuring OSPF for R3....] *****
```

```
skipping: [192.168.1.1]  
skipping: [10.1.1.1]  
skipping: [10.2.1.1]  
skipping: [10.1.1.2]  
skipping: [10.3.1.1]
```

skipping: [172.169.3.1]
skipping: [10.3.1.2]
skipping: [172.168.2.1]
skipping: [172.168.1.1]
changed: [10.2.1.2]

TASK [Configuring AAA login Authentication for console access...] *****

ok: [10.2.1.1]
ok: [10.1.1.1]
ok: [10.2.1.2]
ok: [10.1.1.2]
ok: [192.168.1.1]
ok: [10.3.1.2]
ok: [10.3.1.1]
ok: [172.169.3.1]
ok: [172.168.1.1]
ok: [172.168.2.1]

TASK [Verifying the line console to implement the defined AAA authentication...] *****

changed: [10.2.1.2]
changed: [10.1.1.2]
changed: [10.1.1.1]
changed: [192.168.1.1]
changed: [10.2.1.1]
changed: [10.3.1.1]
changed: [172.168.1.1]
changed: [10.3.1.2]
changed: [172.169.3.1]
changed: [172.168.2.1]

TASK [Allowing AAA login Authentication via SSH-LOGIN Local...] *****

changed: [10.2.1.2]
changed: [192.168.1.1]
changed: [10.2.1.1]
changed: [10.1.1.1]
changed: [10.1.1.2]
ok: [10.3.1.1]

ok: [10.3.1.2]
ok: [172.168.1.1]
ok: [172.169.3.1]
ok: [172.168.2.1]

TASK [Configuring vty lines to the authentication phase...] *****

changed: [10.2.1.2]
changed: [10.1.1.2]
changed: [10.2.1.1]
ok: [10.3.1.1]
changed: [192.168.1.1]
changed: [10.1.1.1]
ok: [10.3.1.2]
ok: [172.168.1.1]
ok: [172.169.3.1]
ok: [172.168.2.1]

TASK [Implementing Access List for Manager] *****

skipping: [10.1.1.1]
skipping: [192.168.1.1]
skipping: [10.2.1.1]
skipping: [10.1.1.2]
skipping: [10.2.1.2]
skipping: [10.3.1.1]
skipping: [10.3.1.2]
skipping: [172.169.3.1]
skipping: [172.168.2.1]
changed: [172.168.1.1]

TASK [Implementing access group on Interface F0/1] *****

skipping: [10.1.1.1]
skipping: [192.168.1.1]
skipping: [10.1.1.2]
skipping: [10.2.1.2]
skipping: [10.2.1.1]
skipping: [10.3.1.1]
skipping: [172.169.3.1]

skipping: [172.168.2.1]
skipping: [10.3.1.2]
changed: [172.168.1.1]

TASK [Implementing Access List for SUPERVISOR] *****

skipping: [192.168.1.1]
skipping: [10.1.1.2]
skipping: [10.1.1.1]
skipping: [10.2.1.1]
skipping: [10.2.1.2]
skipping: [10.3.1.1]
skipping: [172.169.3.1]
skipping: [10.3.1.2]
skipping: [172.168.1.1]
changed: [172.168.2.1]

TASK [Implementing access group on Interface F0/1] *****

skipping: [10.1.1.1]
skipping: [192.168.1.1]
skipping: [10.1.1.2]
skipping: [10.2.1.1]
skipping: [10.2.1.2]
skipping: [10.3.1.1]
skipping: [172.169.3.1]
skipping: [10.3.1.2]
skipping: [172.168.1.1]
changed: [172.168.2.1]

PLAY RECAP *****

10.1.1.1	: ok=5	changed=3	unreachable=0	failed=0	skipped=6	rescued=0	ignored=0
10.1.1.2	: ok=4	changed=3	unreachable=0	failed=0	skipped=7	rescued=0	ignored=0
10.2.1.1	: ok=5	changed=3	unreachable=0	failed=0	skipped=6	rescued=0	ignored=0
10.2.1.2	: ok=5	changed=4	unreachable=0	failed=0	skipped=6	rescued=0	ignored=0
10.3.1.1	: ok=4	changed=1	unreachable=0	failed=0	skipped=7	rescued=0	ignored=0
10.3.1.2	: ok=4	changed=1	unreachable=0	failed=0	skipped=7	rescued=0	ignored=0
172.168.1.1	: ok=6	changed=3	unreachable=0	failed=0	skipped=5	rescued=0	ignored=0
172.168.2.1	: ok=6	changed=3	unreachable=0	failed=0	skipped=5	rescued=0	ignored=0

```
172.169.3.1      : ok=4  changed=1  unreachable=0  failed=0  skipped=7  rescued=0  ignored=0
192.168.1.1      : ok=4  changed=3  unreachable=0  failed=0  skipped=7  rescued=0  ignored=0
```

c. Verify that the ACL configurations are successfully implemented Manager's PC on R1 and Supervisor's PC on R3. To easily check the configuration use the command below:

1. R1

```
R1(config)#do sh access-list
```

```
Standard IP access list PC_MANAGER
```

```
10 deny 172.169.3.0, wildcard bits 0.0.0.63 (10 matches)
```

```
20 permit any
```

```
R1(config)#
```

2. R2

```
R3(config)#do sh access-list
```

```
Standard IP access list PC_SUPERVISOR
```

```
10 deny 172.169.3.0, wildcard bits 0.0.0.63 (10 matches)
```

```
20 permit any
```

```
R3(config)#
```

Part 8: Complete Source Code of YAML

a. Complete source code of three different topics in playbook.yml file.

```
---
```

```
#OSPF
```

```
- hosts: routers
```

```
gather_facts: no
```

```
connection: network_cli
```

```
become_method: enable
```

```
tasks:
```

```
- name: Configuring OSPF for R1....
```

```
  when: ansible_host == "10.1.1.1"
```

```
  ios_config:
```

```
    parents: router ospf 100
```

```
    lines:
```

```
      - router-id 1.1.1.1
```

```
      - network 10.1.1.0 0.0.0.3 area 0
```

```
      - network 192.168.1.0 0.0.0.255 area 0
```

```
      - network 172.168.1.0 0.0.0.63 area 0
```

- network 10.3.1.0 0.0.0.3 area 0
- passive-interface FastEthernet0/0
- passive-interface FastEthernet0/1

- name: Configuring OSPF for R2....

when: ansible_host == "10.2.1.1"

ios_config:

parents: router ospf 100

lines:

- router-id 2.2.2.2
- network 10.2.1.0 0.0.0.3 area 0
- network 10.1.1.0 0.0.0.3 area 0
- default-information originate

- name: Configuring OSPF for R3....

when: ansible_host == "10.2.1.2"

ios_config:

parents: router ospf 100

lines:

- router-id 3.3.3.3
- passive-interface f0/1
- passive-interface f1/0
- network 10.2.1.0 0.0.0.3 area 0
- network 172.168.2.0 0.0.0.63 area 0
- network 172.169.3.0 0.0.0.63 area 0
- network 10.3.1.0 0.0.0.3 area 0

#AAA

- name: Configuring AAA login Authentication for console access...

ios_config:

commands:

- aaa new-model
- aaa authentication login default local

- name: Verifying the line console to implement the defined AAA authentication...

ios_config:

commands:

- login authentication default

parents: line console 0

- name: Allowing AAA login Authentication via SSH-LOGIN Local...

ios_config:

commands:

- aaa authentication login SSH-LOGIN local

- name: Configuring vty lines to the authentication phase...

ios_config:

commands:

- login authentication SSH-LOGIN

parents: line vty 0 4

#ACL

- name: Implementing Access List for Manager

when: ansible_host == "172.168.1.1"

ios_config:

lines:

- deny 172.169.3.0 0.0.0.63

- permit any

parents: ip access-list standard PC_MANAGER

- name: Implementing access group on Interface F0/1

when: ansible_host == "172.168.1.1"

ios_config:

lines:

- int f0/1

- ip access-group PC_MANAGER out

- name: Implementing Access List for SUPERVISOR

when: ansible_host == "172.168.2.1"

ios_config:

lines:

- deny 172.169.3.0 0.0.0.63

- permit any

parents: ip access-list standard PC_SUPERVISOR

```
- name: Implementing access group on Interface F0/1
  when: ansible_host == "172.168.2.1"
  ios_config:
    lines:
      - int f0/1
      - ip access-group PC_SUPERVISOR out
```

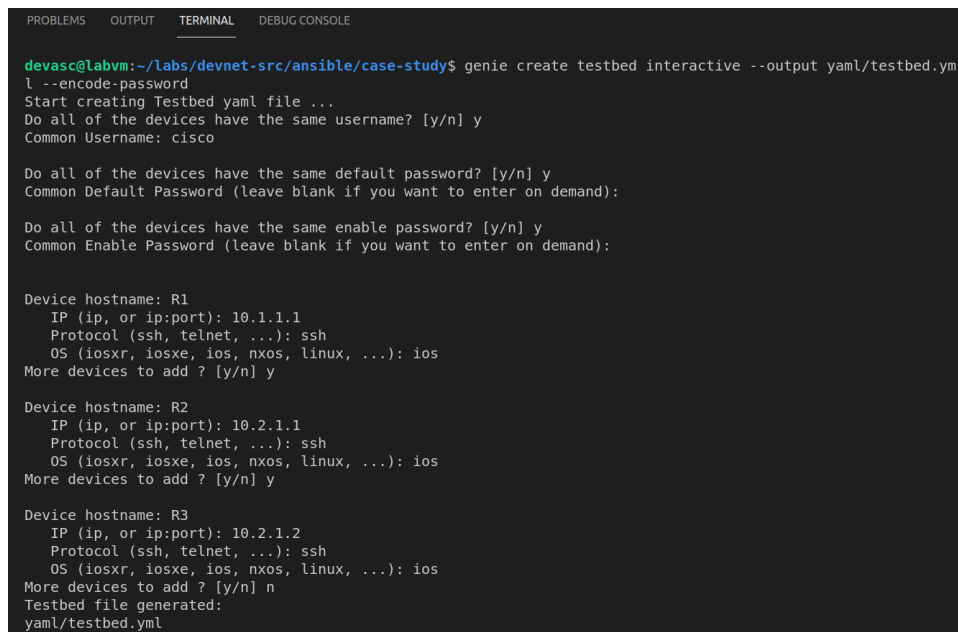
Part 9: Check and test the network configurations

Checking the configurations is important to verify the connections to each router. This also helps the user to verify that all implementations are well implemented to each router. Genie is being used to test the network configurations since this is a library solution for pyATS.

Step 1: Create a Testbed YAML

a. The code below is used to create pyATS testbed YAML. Pyats testbed YAML is used to be able come up a format to describe the physical devices on how each device can link to each other from the topology. Use **genie create testbed interactive --output yaml/testbed.yml --encode-password** to create a testbed YAML.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ genie create testbed interactive --output
yaml/testbed.yml --encode-password
```



```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ genie create testbed interactive --output yaml/testbed.yml
l --encode-password
Start creating Testbed yaml file ...
Do all of the devices have the same username? [y/n] y
Common Username: cisco

Do all of the devices have the same default password? [y/n] y
Common Default Password (leave blank if you want to enter on demand):

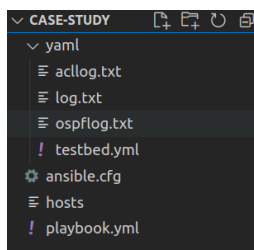
Do all of the devices have the same enable password? [y/n] y
Common Enable Password (leave blank if you want to enter on demand):

Device hostname: R1
IP (ip, or ip:port): 10.1.1.1
Protocol (ssh, telnet, ...): ssh
OS (iosxr, iosxe, ios, nxos, linux, ...): ios
More devices to add ? [y/n] y

Device hostname: R2
IP (ip, or ip:port): 10.2.1.1
Protocol (ssh, telnet, ...): ssh
OS (iosxr, iosxe, ios, nxos, linux, ...): ios
More devices to add ? [y/n] y

Device hostname: R3
IP (ip, or ip:port): 10.2.1.2
Protocol (ssh, telnet, ...): ssh
OS (iosxr, iosxe, ios, nxos, linux, ...): ios
More devices to add ? [y/n] n
Testbed file generated:
yaml/testbed.yml
```

b. Verify if the testbed yaml file is already created in the case-study folder.



Step 2: Parse unstructured output using testbed YAML

a. Using the testbed that you create from the previous step, parse unstructured output from the command **show ip interface brief**. The output of this command will display to the text file log.txt. The testbed YAML will serve as the formatting for the routers that we set from the previous step that will be used to output the interfaces of the routers to the log.txt file.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ genie parse "show ip interface brief" --testbed-
file yaml/testbed.yml --devices > yaml/log.txt
```

```
100%|
| 1/1 [00:01<00:00, 1.64s/it]
```

```
100%|
| 1/1 [00:00<00:00, 2.68it/s]
```

```
100%|
| 1/1 [00:00<00:00, 2.54it/s]
```

b. The output of the log.txt file should look like the output below, but this is only the first 26 lines of the output. The output below shows that the configurations per routers are successful.

```
! playbook.yml  log.txt  ansible.cfg  hosts
yaml > log.txt
1  {
2    "interface": {
3      "FastEthernet0/0": {
4        "interface_is_ok": "YES",
5        "ip_address": "192.168.1.1",
6        "method": "NVRAM",
7        "protocol": "up",
8        "status": "up"
9      },
10     "FastEthernet0/1": {
11       "interface_is_ok": "YES",
12       "ip_address": "172.168.1.1",
13       "method": "NVRAM",
14       "protocol": "up",
15       "status": "up"
16     },
17     "FastEthernet1/0": {
18       "interface_is_ok": "YES",
19       "ip_address": "unassigned",
20       "method": "NVRAM",
21       "protocol": "down",
22       "status": "administratively down"
23     },
24     "Serial0/0": {
25       "interface_is_ok": "YES",
26       "ip_address": "10.1.1.1",
```

c. Using the testbed that you create from the previous step, parse unstructured output from the command **show ip ospf**. The output of this command will display to the text file ospflog.txt. The testbed YAML will serve as the formatting for the routers that we set from the previous step that will be used to output the interfaces of the routers to the ospflog.txt.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ genie parse "show ip ospf" --testbed-file
yaml/testbed.yml --devices > yaml/ospflog.txt
```

100% | 1/1 [00:00<00:00, 1.30it/s]

```
100%|██████████| 1/1 [00:00<00:00, 3.69it/s]
```

100% | 1/1 [00:00<00:00, 3.82it/s

d. The output of the ospflog.txt file should look like the output below, but this is only a portion of the output. But the output shows that the OSPF is configured successfully in the router since it has a router-id of 1.1.1.1 on R1, 2.2.2 on R2, and 3.3.3.3 on R3.

```
! playbook.yml  log.txt  ospflog.txt x  ansible.cfg  hosts
```

```
yaml > ospflog.txt
56      "external_lsa": 1,
57      "external_lsa_checksum": "0x00BB6B",
58      "opaque_as_lsa": 0,
59      "opaque_as_lsa_checksum": "0x000000"
60    },
61    "opaque_lsa": true,
62    "retransmission_pacing_timer": 66,
63    "router_id": "1.1.1.1",
64    "spf_control": {
65      "incremental_spf": false,
66      "throttle": {
67        "lsa": {
68          "arrival": 1000
69        },
70        "spf": {
71          "hold": 10000,
72          "maximum": 10000
```

```

! playbook.yml  log.txt  ospflog.txt  ansible.cfg  hosts
yaml > ospflog.txt
156     "external_lsa": 1,
157     "external_lsa_checksum": "0x00BB6B",
158     "opaque_as_lsa": 0,
159     "opaque_as_lsa_checksum": "0x000000"
160 },
161 "opaque_lsa": true,
162 "redistribution": {},
163 "retransmission_pacing_timer": 66,
164 "router_id": "2.2.2.2",
165 "spf_control": {
166     "incremental_spf": false,
167     "throttle": {
168         "lsa": {
169             "arrival": 1000
170         },
171         "spf": {
172             "hold": 10000

```

```
! playbook.yml | log.txt | ospflog.txt x | ansible.cfg | hosts
```

```
yaml > ospflog.txt
253         "external_lsa": 1,
254         "external_lsa_checksum": "0x00BB6B",
255         "opaque_as_lsa": 0,
256         "opaque_as_lsa_checksum": "0x000000"
257     },
258     "opaque_lsa": true,
259     "retransmission_pacing_timer": 66,
260     "router_id": "3.3.3.3",
261     "spf_control": {
262         "incremental_spf": false,
263         "throttle": {
264             "lsa": {
265                 "arrival": 1000
266             },
267             "spf": {
268                 "hold": 10000,
```

e. Using the testbed that you create from the previous step, parse unstructured output from the command **show access-list**. The output of this command will display to the text file ospflog.txt. The testbed YAML will serve as the formatting for the routers that we set from the previous step that will be used to output the interfaces of the routers to the ospflog.txt.

```
devasc@labvm:~/labs/devnet-src/ansible/case-study$ genie parse "show access-list" --testbed-file
yaml/testbed.yml --devices > yaml/aclog.txt
```

The image displays three horizontal progress bars, each representing a different stage or type of command execution. Each bar has a yellow segment indicating progress, followed by a black segment, and ends with a vertical bar. Below each bar is a text label providing details about the progress.

100% | 1/1 [00:00<00:00, 1.27it/s]

0% | 0/1 [00:00<?, ?it/s] Parsed command 'show access-list' but it returned empty

100% | 1/1 [00:00<00:00, 3.18it/s]

100% | 1/1 [00:00<00:00, 3.26it/s]

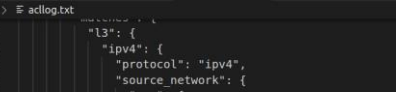
Note: You can see the output saying "show access-list" but it returned empty". It is ok since not all routers are used to implement ACL. Only R1 and R3 since those routers has the PC_MANAGER and PC_SUPERVISOR.

f. The output of the ospflog.txt file should look like the output below, but this is only a portion of the output.

```

1  {
2    "PC_MANAGER": {
3      "aces": {
4        "10": {
5          "actions": {
6            "forwarding": "deny"
7          },
8          "matches": {
9            "l3": {
10             "ipv4": {
11               "protocol": "ipv4",
12               "source_network": {
13                 "172.169.3.0 0.0.0.63": {
14                   "source_network": "172.169.3.0 0.0.0.63"
15                 }
16             }
17           }
18         }
19       }
20     }
21   }
22 }

```



```

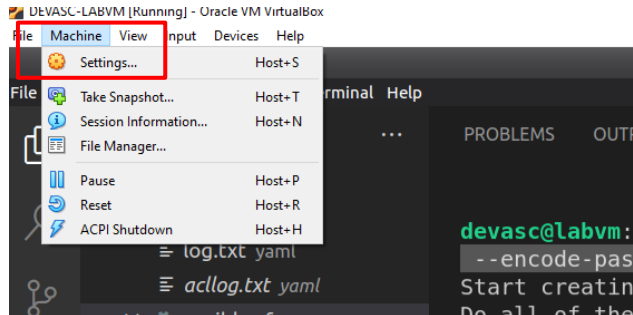
77     "l3": {
78         "ipv4": {
79             "protocol": "ipv4",
80             "source_network": {
81                 "any": {
82                     "source_network": "any"
83                 }
84             }
85         }
86     },
87     "name": "20"
88 },
89 },
90 },
91 "name": "PC_SUPERVISOR",
92 "type": "ipv4-acl-type"

```

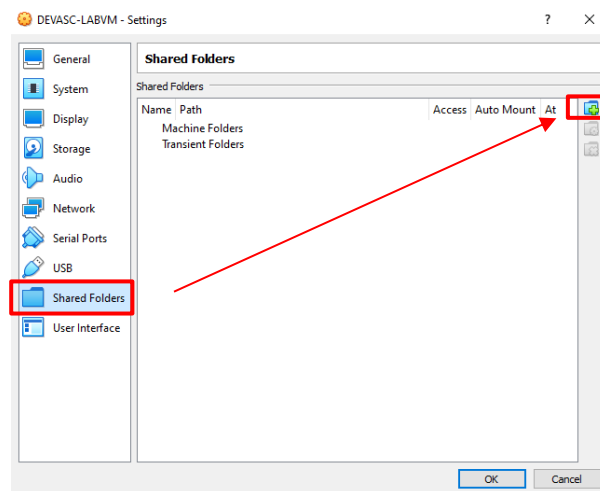
Part 10: Uploading the file to GitHub

Step 1: Enable the shared folders setting of Devasc VM in Oracle Virtual Box.

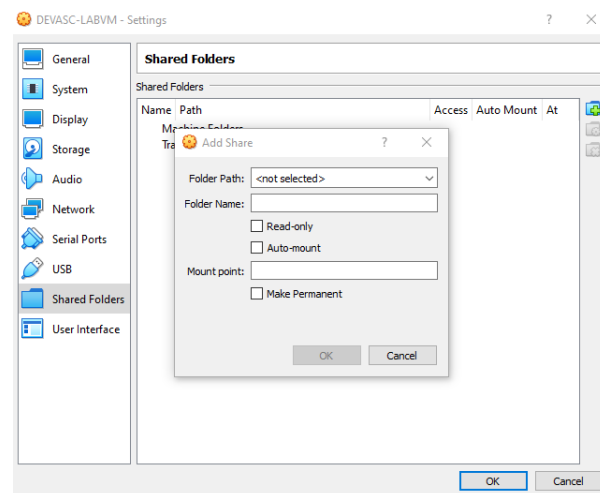
- a. Click the “Machine” in the top left corner of the Devasc VM window then click “Settings”.



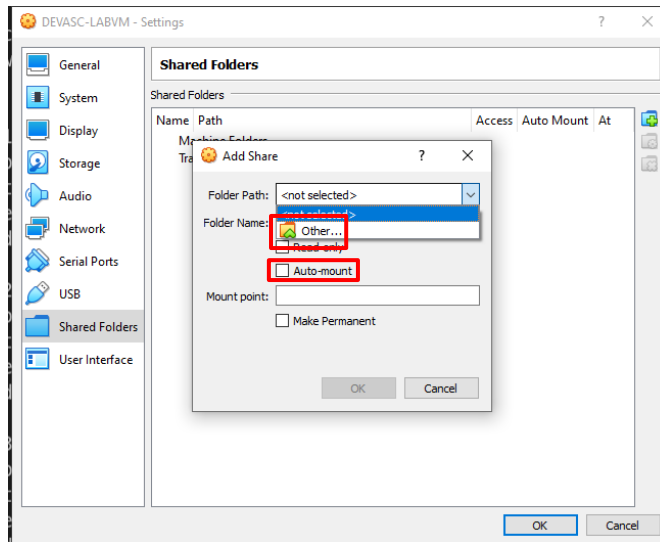
- b. The Devasc VM setting window will pop-out. Click “Shared Folders”. Then Click “Add Share”.



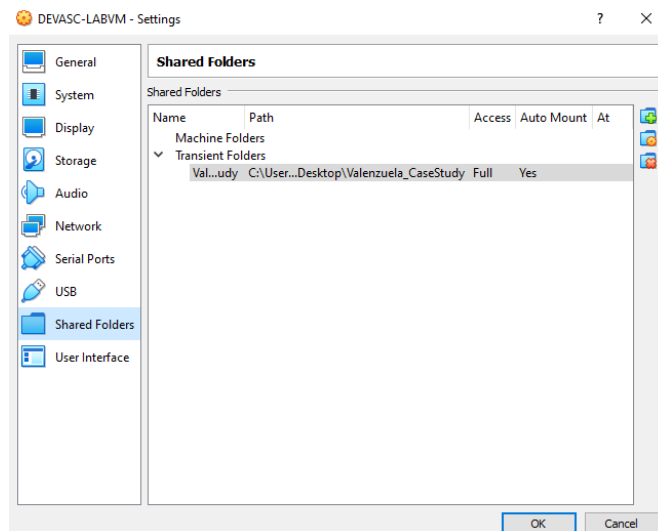
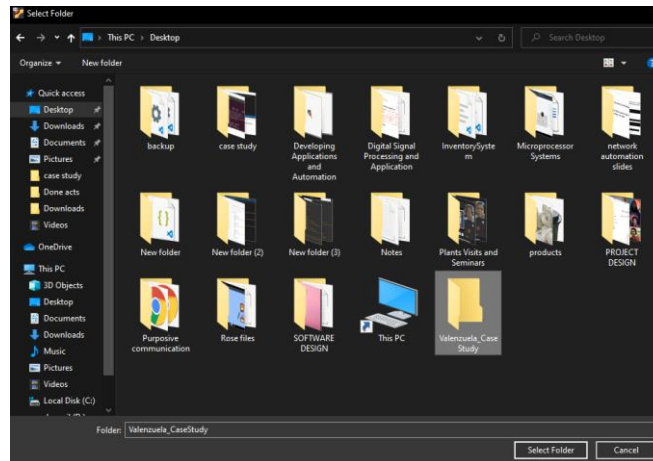
- c. Click “Add Share”.



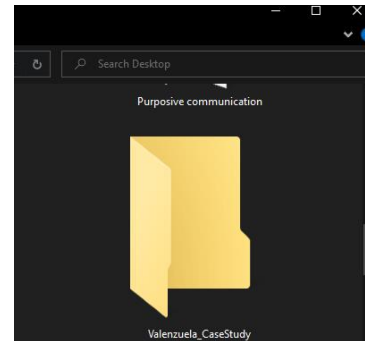
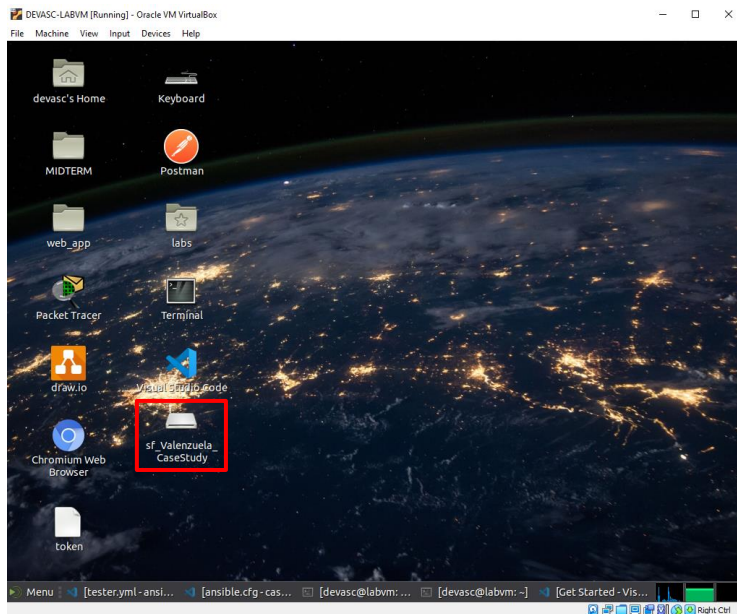
d. Drop down the folder path and select “Other”. Put a check also the “Auto-mount”.



e. Create a new folder and name it and select that folder. Then you will see it to the folders.



f. You will now see the folder in Devasc desktop and Windows desktop.



Step 2: Upload the files using Git bash in Windows.

a. Configure the user and follow the commands.

```
MINGW64/c/Users/dnvne/Desktop/Valenzuela_CaseStudy

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy
$ git config --global user.email qdn1valenzuela@tip.edu.ph

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy
$ git config --global user.name dvalenzuela-tip

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy
$ git init
Initialized empty Git repository in C:/Users/dnvne/Desktop/Valenzuela_CaseStudy/.git/

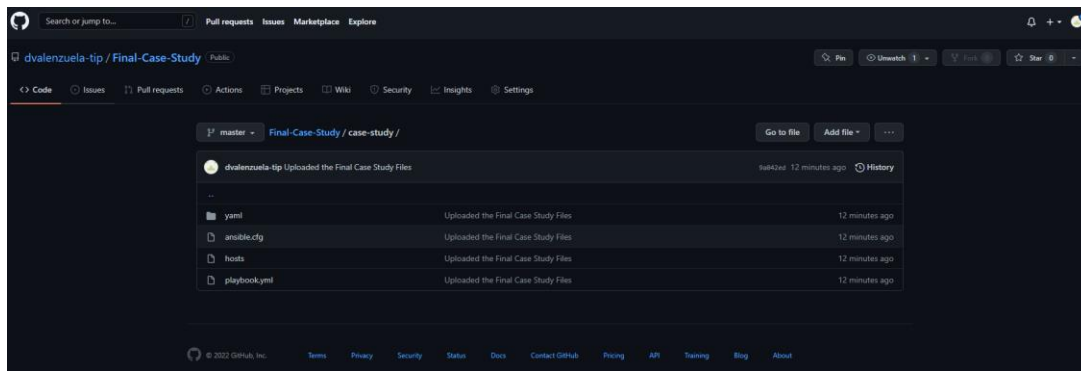
dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy (master)
$ git remote add origin https://github.com/dvalenzuela-tip/Final-Case-Study.git

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy (master)
$ git add .
warning: LF will be replaced by CRLF in case-study/ansible.cfg.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/hosts.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/playbook.yml.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/yaml/aclog.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/yaml/log.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/yaml/ospflog.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in case-study/yaml/testbed.yml.
The file will have its original line endings in your working directory

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy (master)
$ git commit -m "Uploaded the Final Case Study Files"
[master (root-commit) 9a042ed] Uploaded the Final Case Study Files
7 files changed, 779 insertions(+)
create mode 100644 case-study/ansible.cfg
create mode 100644 case-study/hosts
create mode 100644 case-study/playbook.yml
create mode 100644 case-study/yaml/aclog.txt
create mode 100644 case-study/yaml/log.txt
create mode 100644 case-study/yaml/ospflog.txt
create mode 100644 case-study/yaml/testbed.yml

dnvnei@DESKTOP-LP8FEAO MINGW64 ~/Desktop/Valenzuela_CaseStudy (master)
$ git push -u origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 3.01 KiB | 1.51 MiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dvalenzuela-tip/Final-Case-Study.git
 * [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

b. Verify the uploaded file in the Github.



Conclusion:

In the end of the laboratory activity, I was able to recall the basic configurations and IP addressing method for the routers. I was also able to build a topology that has three topics such as AAA for the security, ACL for the access control, and OSPF for the routing protocol. I also learned more about setting up an ansible for the network automation process of the three topics. I was also able to configure and implement the AAA security in the topology that I have created to improve the security standards of each router. I also implemented the Access Control List (ACL) to limit the access of the interface for a specific router. I also learned more about Open Shortest Path First (OSPF) since I was able to implement it to the topology and configure it using ansible. I was also able to test the configurations that I had implemented for each router using pyATS Genie.