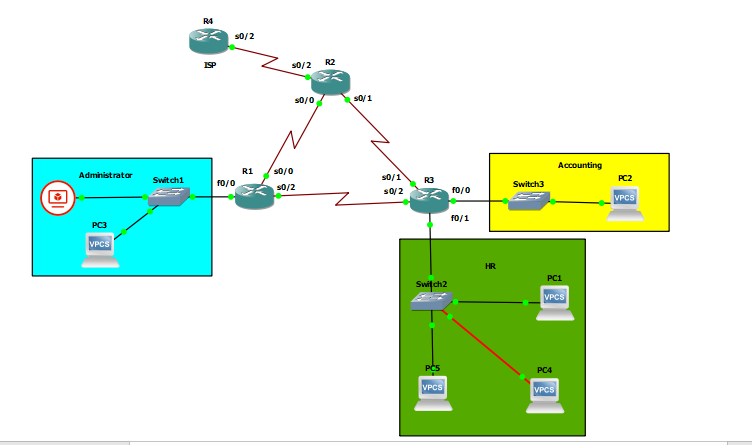
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device** | **Interface** | **IP Address** | **Subnet Mask** | **Default Gateway** |
| R1 | S0/0 | 10.0.0.1 | 255.255.255.252 | N / A |
|  | S0/2 | 10.0.0.5 | 255.255.255.252 | N / A |
|  | F0/0 | 192.168.1.1 | 255.255.255.0 | N / A |
| R2 | S0/0 | 10.0.0.2 | 255.255.255.252 | N / A |
|  | S0/1 | 10.0.0.10 | 255.255.255.252 | N / A |
|  | S0/2 | 200.200.200.1 | 255.255.255.252 | N / A |
| R3 | S0/1 | 10.0.0.9 | 255.255.255.252 | N / A |
|  | S0/2 | 10.0.0.6 | 255.255.255.252 | N / A |
|  | F0/0 | 192.168.3.1 | 255.255.255.0 | N / A |
|  | F0/1 | 192.168.2.1 | 255.255.255.0 | N / A |
| ISP | S0/2 | 200.200.200.2 | 255.255.255.252 | N / A |
| Accounting | NIC | 192.168.3.10 | 255.255.255.0 | 192.168.3.1 |
| HR | NIC | 192.168.2.10 | 255.255.255.0 | 192.168.2.1 |
|  | NIC | 192.168.2.20 | 255.255.255.0 | 192.168.2.1 |
|  | NIC | 192.168.2.30 | 255.255.255.0 | 192.168.2.1 |
| ADMIN | NIC | 192.168.1.10 | 255.255.255.0 | 192.168.1.1 |
|  | NIC | 192.168.1.20 | 255.255.255.0 | 192.168.1.1 |



**Objectives:**

|  |  |
| --- | --- |
| **Part 1:** | **Launch GNS3** |
| **Part 2:** | **Creating the Network Topology and Applying Basic Configuration** |
| **Part 3:** | **Launch the DEVASC VM** |
| **Part 4:** | **Setting Up Ansible in DEVASC** |
| **Part 5:** | **Applying Open Shortest Path First (OSPF)** |
| **Part 6:** | **Applying Authentication, Authorization and Accounting (AAA)** |
| **Part 7:** | **Applying Access Control Lists (ACL)** |
| **Part 8:** | **Testing Network using pyATS and Genie** |
| **Part 9:** | **Uploading to GitHub** |

**Background / Scenario:**

In this activity, you will need to design a laboratory activity that discusses three different network topics, specifically OSPF, AAA, and ACL. You will use Ansible as an application-deployment tool. To test the network, pyATS, a Python Automated Test Systems will be used.

The devices used in the topology are configured with:

* Console Password: **cisco123**
* Enable Password: **cisco**

**Required Resources:**

* Personal Computer with Operating System of your choice.
* GNS3
* Oracle Virtual Box or VMWare
* DEVASC Virtual Machine

# Instructions

# Part 1: Launch the GNS3

# Part 2: Creating the Network Topology and Applying Basic Configuration

Step 1: Construct the network according to the topology. Connect the essential devices and connections as indicated in the topology..

Step 2: Configure the basic configuration for router R1.

R1> **en**

R1# **conf t**

R1(config)# **hostname R1**

R1(config)# **username cisco password cisco123**

R1(config)# **enable secret class**

R1(config)# **service password-encryption**

R1(config)# **banner motd "Unauthorized Access is Prohibited"**

R1(config)# **ip domain-name www.abc.com**

R1(config)# **crypto key gen rsa**

The name for the keys will be: R1.www.example.com

Choose the size of the key modulus in the range of 360 to 2048 for your

General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.

How many bits in the modulus [512]: **1024**

% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#

\*Mar 1 00:25:19.663: %SSH-5-ENABLED: SSH 1.99 has been enabled

R1(config)# **ip ssh ver 2**

R1(config)# **line con 0**

R1(config-line)# **password cisco**

R1(config-line)# **login local**

R1(config-line)# **line vty 0 15**

R1(config-line)# **login local**

R1(config-line)# **transport input ssh**

R1(config-line)# **do copy r s** Destination filename [startup-config]?

Building configuration...

[OK]

R1(config-line)#

Step 3: Do the same procedure with R2 and R3..

Step 4: IP Addressing of the Router Interface should be done in accordance with the IP Addressing Table.

R1(config)# **int f0/0**

R1(config-if)# **ip address 192.168.1.1 255.255.255.0**

R1(config-if)# **no shut**

R1(config-if)# **int s0/0**

R1(config-if)# **ip address 10.0.0.1 255.255.255.252**

R1(config-if)# **no shut**

R1(config-if)# **int s0/2**

R1(config-if)# **ip address 10.0.0.5 255.255.255.252**

R1(config-if)# **no shut**

R1(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.2**

R1(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.6**

R2(config)# **int s0/0**

R2(config-if)# **ip address 10.0.0.2 255.255.255.252**

R2(config-if)# **no shut**

R2(config-if)# **int s0/1**

R2(config-if)# **ip address 10.0.0.10 255.255.255.252**

R2(config-if)# **no shut**

R2(config-if)# **int s0/2**

R2(config-if)# **ip address 200.200.200.1 255.255.255.252**

R2(config-if)# **no shut**

R2(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.1**

R2(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.9**

R2(config)# **ip route 0.0.0.0 0.0.0.0 200.200.200.2**

R3(config)# **int s0/1**

R3(config-if)# **ip address 10.0.0.9 255.255.255.252**

R3(config-if)# **no shut**

R3(config-if)# **int s0/2**

R3(config-if)# **ip address 10.0.0.6 255.255.255.252**

R3(config-if)# **no shut**

R3(config-if)# **int f0/0**

R3(config-if)# **ip address 192.168.3.1 255.255.255.0**

R3(config-if)# **no shut**

R3(config-if)# **int f0/1**

R3(config-if)# **ip address 192.168.2.1 255.255.255.0**

R3(config-if)# **no shut**

R3(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.5**

R3(config)# **ip route 0.0.0.0 0.0.0.0 10.0.0.10**

ISP(config)# **int s0/2**

ISP (config-if)# **ip address 200.200.200.2 255.255.255.252**

ISP (config-if)# **no shut**

ISP (config)# **ip route 0.0.0.0 0.0.0.0 200.200.200.1**

Step 5: Configure the Hostname and IP Address of Accounting Virtual PC.

PC1> **set pcname Accounting**

Accounting> **ip 192.168.3.10 255.255.255.0 192.168.3.1**

Checking for duplicate address...

PC1 : 192.168.3.10 255.255.255.0 gateway 192.168.3.1

PC2> **set pcname HR**

HR> **ip 192.168.2.10 255.255.255.0 192.168.2.1**

Checking for duplicate address...

PC2 : 192.168.3.10 255.255.255.0 gateway 192.168.3.1

# Part 3: Launch DEVASC VM

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.

Step 2: Once you’re done cloning, try the connectivity of DEVASC VM to router R1, which is the default gateway in GNS3.

devasc@labvm:~$ ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

Output Omitted ...

4 packets transmitted, 4 received, 0% packet loss, time 3036ms

... devasc@labvm:~$

devasc@labvm:~$ ping 10.0.0.1

PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.

Output Omitted ...

4 packets transmitted, 4 received, 0% packet loss, time 3023ms

... devasc@labvm:~$

devasc@labvm:~$ ping 10.0.0.2

PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

Output Omitted ...

4 packets transmitted, 4 received, 0% packet loss, time 3024ms

... devasc@labvm:~$

devasc@labvm:~$ ping 10.0.0.9

PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.

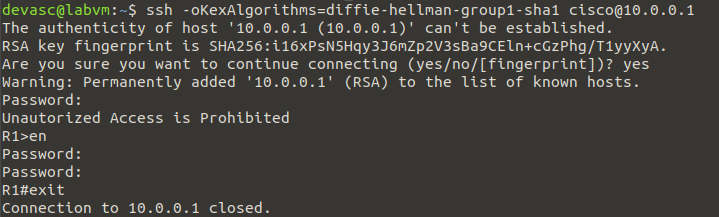
Output Omitted ...

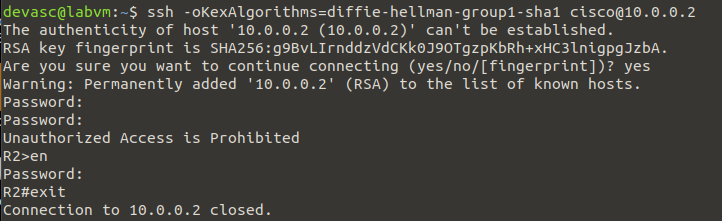
4 packets transmitted, 4 received, 0% packet loss, time 3024ms

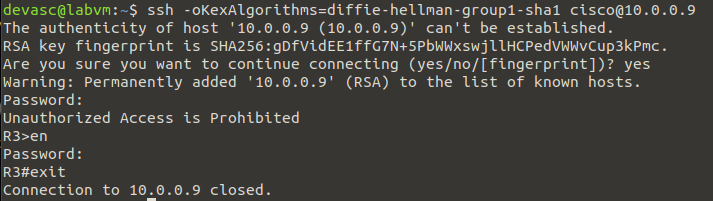
... devasc@labvm:~$

Step 3: Test the connectivity of the DEVASC VM to other routers.

Step 4: After verifying that DEVASC VM. You will be able to access them through SSH connection. This is an important set for applying the three network topics.



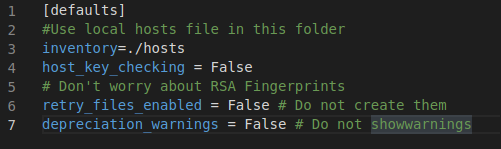




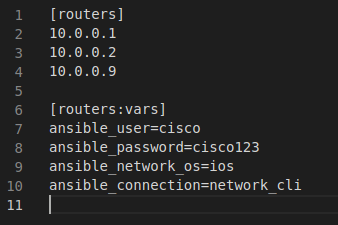
# Part 4: Setting Up Ansible in DEVASC

Step 1: In DEVASC, create a folder on a directory of your choice and name it ‘**casestudy\_palarion**’.

Step 2: Create the ansible configuration file. Copy the following code for the **ansible.cfg** file.



Step 3: After creating the ansible.cfg file, you can now create the hosts file. Here we will include one of the ip addresses of each router. 10.0.0.1 for ansible\_host of router R1, 10.0.0.2 for R2, and 10.0.0.9 for R3. Additionally, we need to modify the variables for each router such as the username, password, os and the connection.



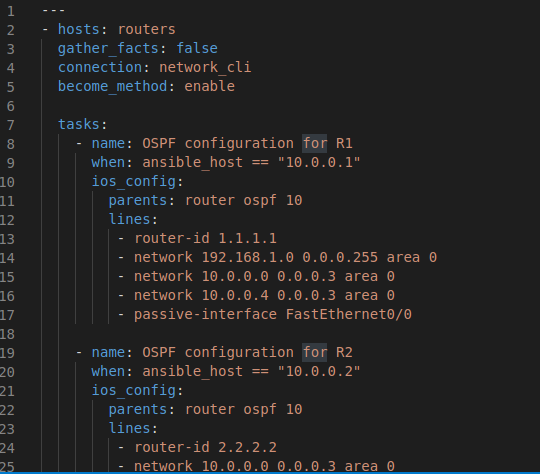
Step 4: Once the ansible.cfg and the hosts file was created, you can now test the connectivity using ansible by running the following code.

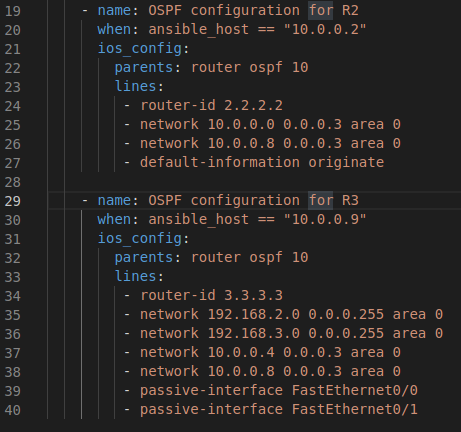


# Part 5: Applying Open Shortest Path First (OSPF)

Here are the following configuration expected for the OSPF of each routers:

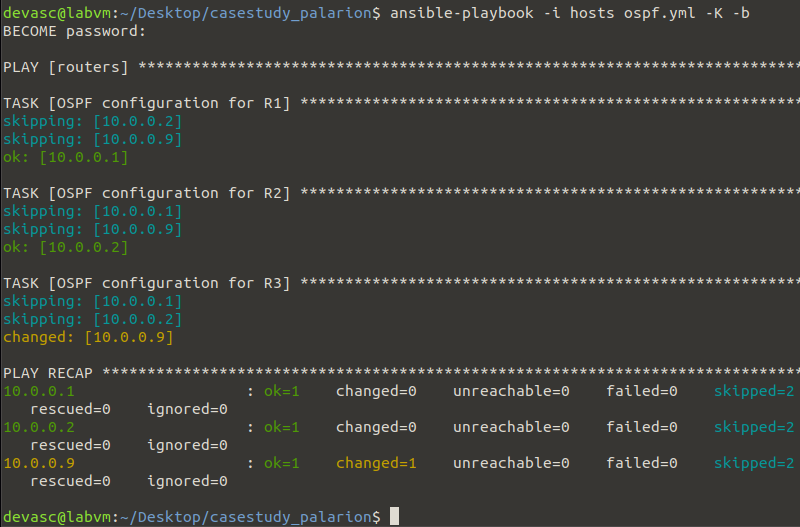
Step 1: In the casestudy\_palarion folder, create a YAML file named ospf.yml for OSPF application. Copy the following code below that provides the capability of applying OSPF to all the routers in the network. Included here is the hosts to be configured, the connection, the become\_method, and the tasks which is for OSPF configuration. In this lab, we’ve used router ospf 10 to all and set the router-id for each routers.





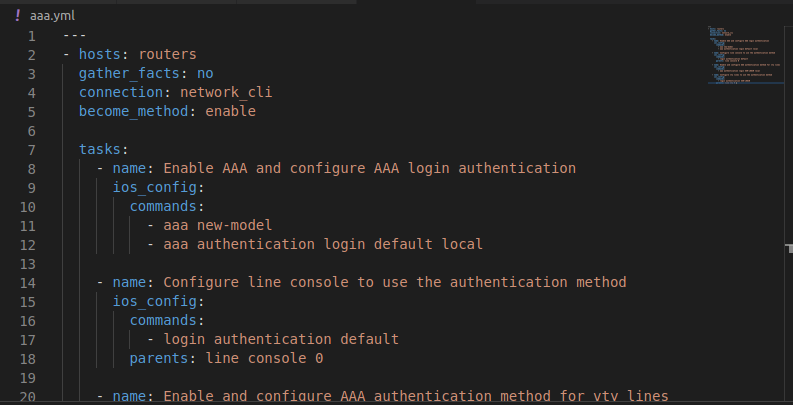
Step 2: After setting up the ospf.yml file, we need to run the following code to apply the configuration created. Here you will need the ansible-playbook to run the ospf.yml file, the -i hosts for the hosts file, -K - b for the prompt of enable password.

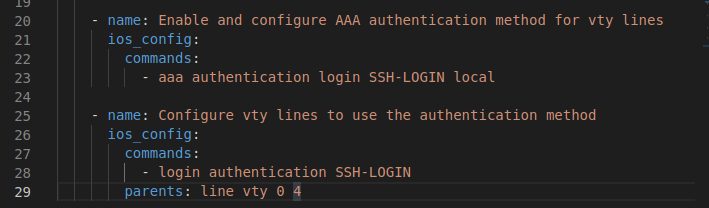




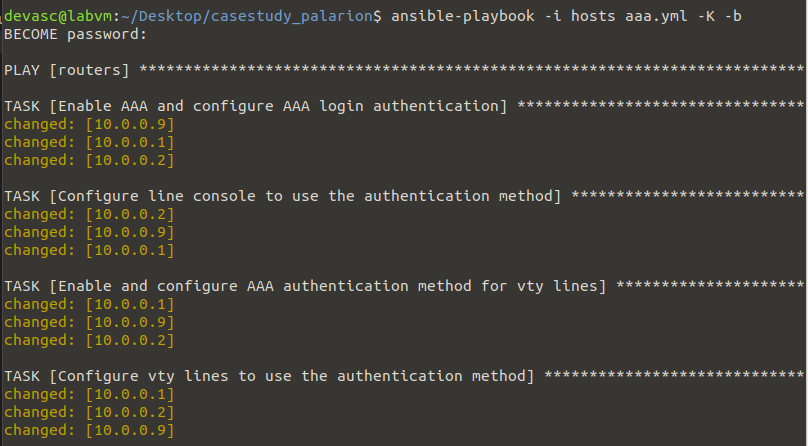
# Part 5: Applying Authentication, Authorization and Accounting (AAA)

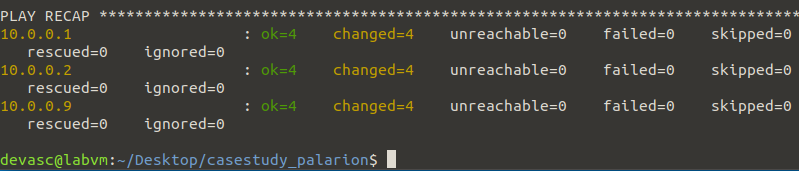
Step 1: In the Case\_Study-LASTNAME folder, create a YAML file named aaa.yml for AAA Security application. Copy the following code below that provides the capability of applying AAA Security to all the routers in the network. Included here are the hosts to be configured, the connection, the become\_method, and the tasks which is for AAA Security configuration.





Step 2: After setting up the aaa.yml file, we need to run the following code to apply the configuration created. Here you will need the ansible-playbook to run the aaa.yml file, the -i hosts for the hosts file, -K -b for the prompt of enable password.

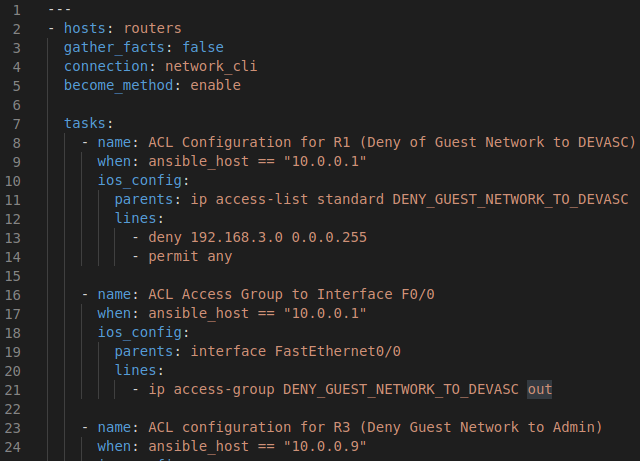


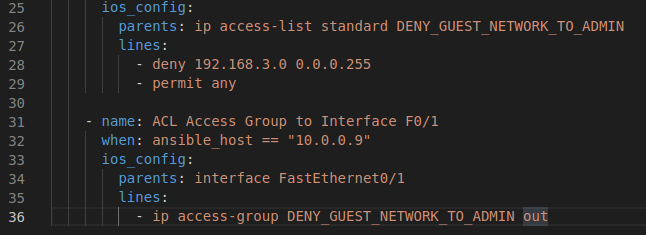


# Part 7: Applying Access Control Lists (ACL)

Before starting the application of ACL in each router, you must have already finished **Part 6: Applying Authentication, Authorization and Accounting (AAA).** If you have not already done that part, you are not able to run the codes properly.

Step 1: In the Case\_Study-LASTNAME folder, create a YAML file named acl.yml for Access Control List application. Copy the following code below that provides the capability of applying ACL to the routers that needs configuration in the network. Included here are the hosts to be configured, the connection, the become\_method, and the tasks which is for ACL configuration.

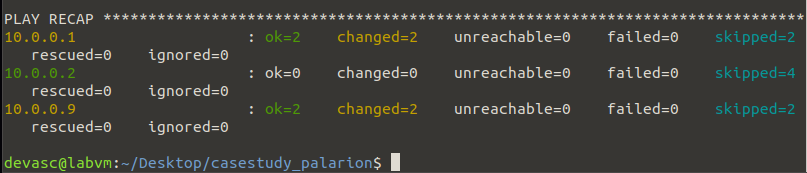




Step 2: After setting up the acl.yml file, we need to run the following code to apply the configuration created. Here you will need the ansible-playbook to run the acl.yml file, the -i hosts for the hosts file, -K -b for the prompt of enable password.

devasc@labvm:~/labs/devnet-src/ansible/Case\_Study-Quebral$ ansible-playbook -i hosts acl.yml -K -b





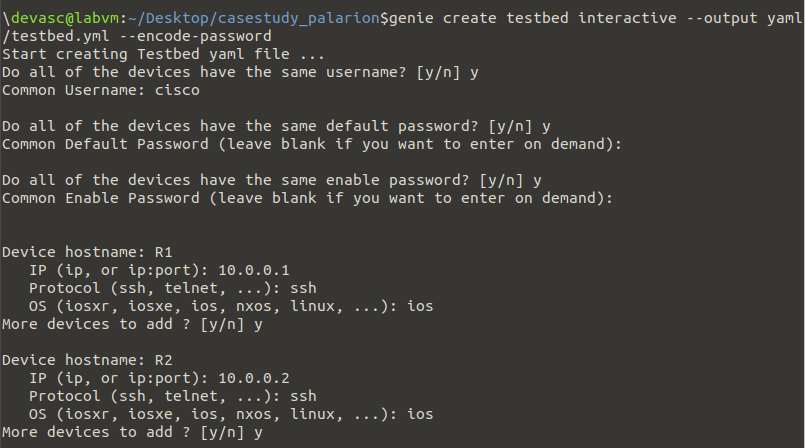
# Part 8: Testing Network using pyATS and Genie

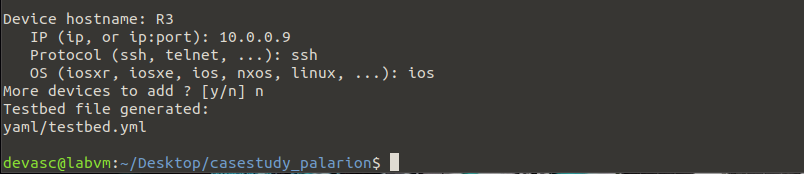
At this part, you have already applied OSPF and ACL in each router. Verifying the existence of each network application will be determined by using pyATS and Genie. PyATS or Python Automated Testing Systems is a Cisco related package for network verification and testing purposes. Genie on the other hand is for simplification of test automation.

But before you can proceed for testing, you need to create a testbed file which will serve as the connection to the device. You need to utilize Genie in order to create it.

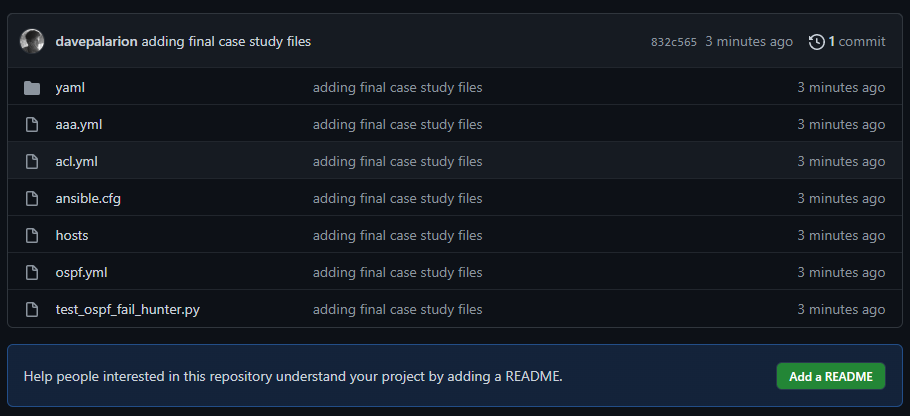
Step 1: Create a testbed file for test automation. Copy the following code to create the testbed.yml.







**Part 8: Upload all files in github repository**



Github link: <https://github.com/davepalarion/Final_Case_Study.git>

Video link: https://drive.google.com/file/d/1VTd6wb8gtlSuGN5oDrJs1iykFcw82Hdy/view?usp=sharing