

# Network Professional Training: Hands-on Workbook

## Scope

This workbook covers configurations in the NVIDIA Academy GTC Workshop.

## Audience

This workbook is intended for Technical Training students registered to the “Configure High Mobility AI-Infrastructure in 5 min” GTC training session.

## Objectives

By the end of this workbook, students will be able to:

- Configure switches and servers using Ansible automation tool.
- Configure layer 2 and layer 3 protocols on NVIDIA Cumulus Linux switches.
- Verify configuration and connectivity

## Overview

Each student will be using the Cumulus Air © platform, exercises in this workbook on a group of devices (servers and switches).

## Notice

Please follow the instructions below carefully to successfully complete the practice.  
If you encounter technical issues, please contact the NVIDIA Academy instructors.

## Release Date

Revision 1.0 – November 2021

Good Luck,

NVIDIA Academy Team

## Table of Contents

Prerequisites and Guidelines .....	3
Academy Lab Topology .....	4
NVIDIA Academy Virtual Lab Access .....	5
 <b>Practice Units</b>	
Step-01: Getting started with Ansible .....	6
Step-02: FRR & NVUE .....	12
Step-03: VLANs and VRFs .....	16
Step-04: VRR .....	21
Step-05: VXLAN, EVPN, MH .....	28
Step-06: VXLAN Routing .....	31
Step-07: ??? .....	40

## Hands-on: Prerequisites and Guidelines

1. Enter the Cumulus Air web page : <https://air.nvidia.com/Login>

Click “GET STARTED” button.

WELCOME TO NVIDIA CUMULUS AIR

To get started, please sign in below.

GET STARTED

- If you have already created an account, use your credentials to [Login](#).
- To sign up for the first time, click “Register” and fill in your details.  
Once completed, a confirmation email will be sent, open it to activate your new account.

EMAIL ADDRESS

PASSWORD

[Forgot password?](#)

LOGIN

Don't have an account? [Register](#).

### Please note

Once you are registered to NVIDIA AIR, please provide your email address to the GTC instructor, the instructor will create the virtual lab which you can access via NVIDIA AIR.


2. Once you are logged in, you will reach the “My Simulations” dashboard.  
Wait for the lab to be [Loaded](#).
3. Click on the “[GTCAcademyWorkshop](#)” label.

GTCAcademyWorkshop

✓ Loaded

Never

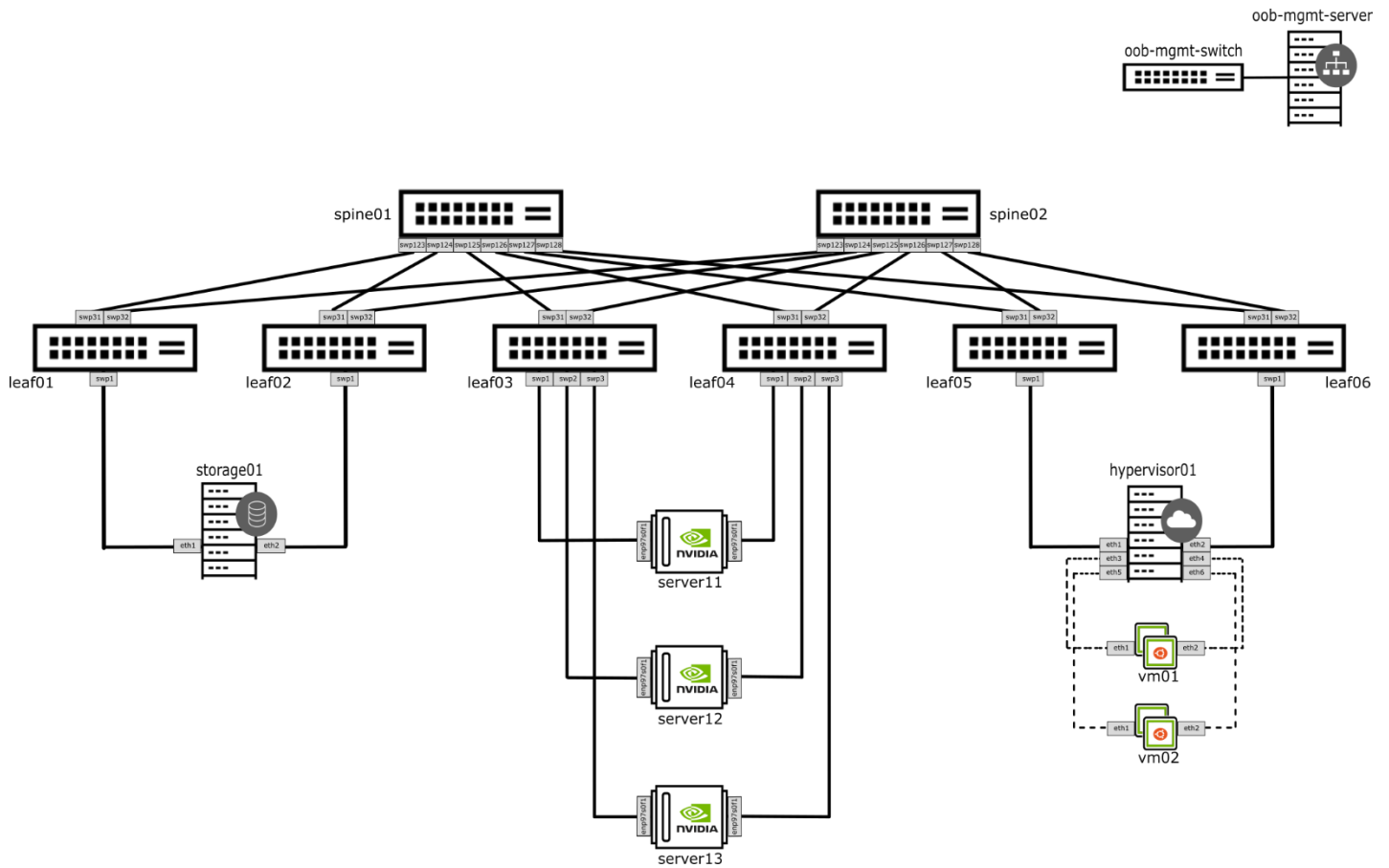
gtcStudent@gmail.com



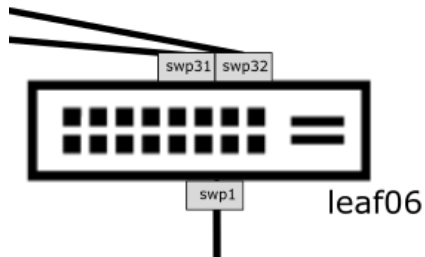
## ACADEMY LAB TOPOLOGY

The workshop lab is organized in the following topology:

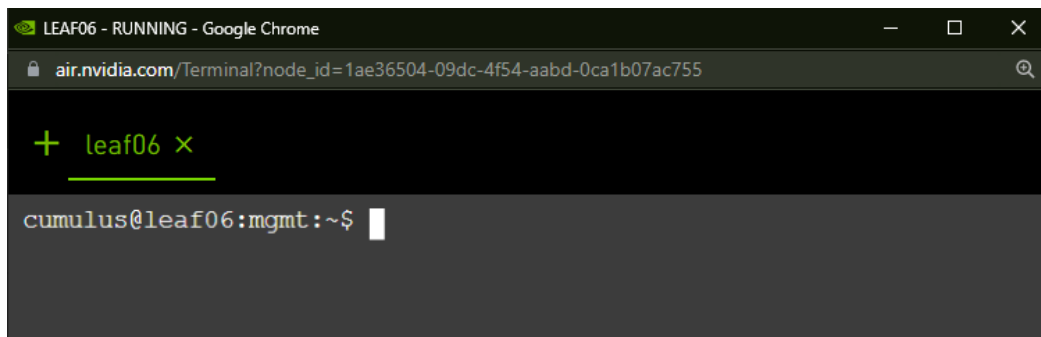


## NVIDIA Academy Virtual Lab Access

Click on a NODE to open its console



1. When the login prompt appears, enter the username – “**cumulus**”
2. When the password prompt appears, enter the password – “**Academy123**” and press Enter.
3. You should now be prompted with the node’s name. This indicates that you have successfully accessed the **node**.



### Please note

The lab can be accessed using SSH rather than the GUI console,  
please ask the GTC instructor for more information regarding SSH keys and connecting using  
your favorite SSH-client.

## Practice Units

## Practice 1.1: Getting Started with Ansible - Inventory

### Practice objectives:

In this practice session you perform the initial configurations required for Ansible to start working with the group servers and switches.

- You will configure **hosts** and **groups** in an Ansible hosts file.
- You will use **Ansible ping module** to validate the configuration.
- Last, you will use **Ansible Variables** to refine the hosts configuration.

## Task 1: Creating an Ansible Inventory (hosts) file

- a. Connect to the 'oob-mgmt-server', create a new directory and Use VIM, or another text editor to create a new file named **hosts** :

```
# mkdir practice1
# sudo vi practice1/hosts
```

[illegible]

⚠ to exit VIM:

1. Press ESC
2. Type “:”
3. Type “q!” to exit **without saving** or “wq” to **save and exit**

~  
~  
:wq

To edit the file using VIM go to insert mode by typing ‘a’  
(make sure the word “—INSERT --” appears at the end of the page).

```
~
~
~
~
~
-- INSERT --
```

## Task 2: Adding servers to the Inventory (hosts) file

- While in “INSERT” mode, add the servers host name to the hosts file.

```
# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server11 ansible_user=cumulus ansible_ssh_pass=Academy123
server12 ansible_user=cumulus ansible_ssh_pass=Academy123
server13 ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123

~
~
~
```

Please note:

- Every line that starts with ‘#’ is considered a comment and can be deleted.
- Instead of configuring each server in a different line, you can use a REGEX expression to capture all compute servers in one line **# server[11:13]**
- The ssh and user password are required for each server separately, but in the next tasks we will see how all hosts can share them using variables.

### Task 3: Testing Ansible connectivity using the “ping” module

- Save and Exit the hosts file (type ESC, ‘:’, ‘wq’ and <enter>)

```
# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server11 ansible_user=cumulus ansible_ssh_pass=Academy123
server12 ansible_user=cumulus ansible_ssh_pass=Academy123
server13 ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123
~
~
~
:wq
```

- Validate the configuration by using the ping module, make sure to use the inventory file you created (use the -i symbol, followed by the hosts file path)

**# ansible -i practice1/hosts server11 -m ping**

```
cumulus@oob-mgmt-server:~$ ansible -i practice1/hosts server11 -m ping
server11 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
cumulus@oob-mgmt-server:~$
```

Please note:

- You might receive a [DEPRECATION WARNING] telling you that there is a later Python version available, please ignore it.



## Task 4: Add servers to a “hosts” group

- a. Use VIM to edit the hosts file, and enter INSERT mode by typing ‘a’  
**# *sudo vi practice1/hosts***
- b. Add all servers to a group called “hosts” (use square brackets)

```
[hosts]

# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server11 ansible_user=cumulus ansible_ssh_pass=Academy123
server12 ansible_user=cumulus ansible_ssh_pass=Academy123
server13 ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123
```

- c. Exit VIM and use the Ansible “ping” module to test the new group configuration, make sure to use the inventory file you created (use the -i symbol, followed by the hosts file path)

**# *ansible -i practice1/hosts hosts -m ping***

```
cumulus@oob-mgmt-server:~$ ansible -i practice1/hosts hosts -m ping

storage01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
server12 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
server13 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
hypervisor01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
server11 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
vm02 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
vm01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

## Task 5: Add Cumulus Linux switches to the inventory

- Use VIM to edit the hosts file, and enter INSERT mode by typing 'a'  
**# *sudo vi practice1/hosts***
- Add the leaf switches ('leaf01' - 'leaf06') to the inventory file, also add the necessary credentials (user and password)  
**# *leaf01 ansible\_user=cumulus ansible\_ssh\_pass=Academy123***

```
[hosts]
# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server[11:13] ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123

leaf01 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf02 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf03 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf04 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf05 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf06 ansible_user=cumulus ansible_ssh_pass=Academy123

~
~
~
```

Please note:

- Instead of configuring each server in a different line, you can use a REGEX expression to capture all compute servers in one line **# *server[11:13]***.

- c. Exit VIM and use the Ansible “ping” module to test Ansible connectivity to the switches.

***ansible -i practice1/hosts leaf01 -m ping***

Please note:

- If the ansible user or password are incorrect, you will get the following error:

```
Leaf02 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: Permission denied
(publickey,password).",
  "unreachable": true}
```

```
cumulus@oob-mgmt-server:~$ ansible leaf1 -m ping
leaf01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

## Task 6: Add Cumulus Linux switches to a “switch” group

- a. Use VIM to edit the hosts file, and enter INSERT mode by typing ‘a’  
# ***sudo vi practice1/hosts***
- d. Add the **leaf** switches to a group called “leaves”

```
.
.
.

[leaves]
leaf01 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf02 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf03 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf04 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf05 ansible_user=cumulus ansible_ssh_pass=Academy123
leaf06 ansible_user=cumulus ansible_ssh_pass=Academy123
~
~
~
```

Please note:

- Instead of configuring each leaf in a different line, you can use a REGEX expression to capture all compute servers in one line # ***leaf[01:06]***.

- b. Add the **spine** switches, same way the leaves were added.

```
.
.
.
[leaves]
leaf[01:06] ansible_user=cumulus ansible_ssh_pass=Academy123

[spines]
Spine01 ansible_user=cumulus ansible_ssh_pass=Academy123
Spine02 ansible_user=cumulus ansible_ssh_pass=Academy123
~
~
~
```

- c. Add the **leaves** and **spines** to a group called “switches”.

```

:
:
:

[hosts]

# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server[11:13] ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123

[leaves]
leaf[01:06] ansible_user=cumulus ansible_ssh_pass=Academy123

[spines]
spine[01:02] ansible_user=cumulus ansible_ssh_pass=Academy123

[switches:children]
spines
leaves

~
~

```

- d. Exit VIM and use the Ansible “ping” module to test Ansible connectivity to the switches  
***ansible -i practice1/hosts switches -m ping***

```
cumulus@oob-mgmt-server:~$ ansible switches -m ping
leaf01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
leaf02 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
leaf03 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
leaf04 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
leaf05 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
leaf06 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
spine02 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
spine01 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

## Task 7: Add variables to be shared by the groups

- Add the username and password as variables, to be shared among all devices, then delete the definitions on each device.

```

:
:
:

[hosts]

# storage
storage01 ansible_user=cumulus ansible_ssh_pass=Academy123

# compute
server[11:13] ansible_user=cumulus ansible_ssh_pass=Academy123

# virtualization
hypervisor01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm01 ansible_user=cumulus ansible_ssh_pass=Academy123
vm02 ansible_user=cumulus ansible_ssh_pass=Academy123

[leaves]
leaf[01:06] ansible_user=cumulus ansible_ssh_pass=Academy123

[spines]
spine[01:02] ansible_user=cumulus ansible_ssh_pass=Academy123

[switches:children]
spines
leaves

[all:vars]
ansible_user=cumulus
ansible_ssh_pass=Academy123

```

Please note:

- Different variables can be shared with different groups.  
For example, a different user can be used for the “hosts” group.



## Practice 1.2: Getting Started with Ansible - Playbooks

### Practice objectives:

In this practice session you will create and execute a basic Ansible playbook.

- You will use the Ansible 'copy' module to set login messages to switches and hosts.
- You will execute the playbook you wrote.

## Task 1: Create a new Ansible playbook

- a. Access the Ansible controller server that was assigned to you, and create a new yaml file under the `/etc/ansible/` directory

```
# touch /etc/ansible/LabPlaybook.yaml
```

- b. Use VIM or another text editor to edit the `/etc/ansible/labPlaybook.yaml` file:

```
# vi /etc/ansible/LabPlaybook.yaml
```

(the file should be empty)

- You should apply similar commands to devices in the group that was assigned to you.
- Every line that starts with '#' is considered a comment and can be deleted.

Please note:

The tasks and commands that you will configure in the playbook are based on **Practice3** in this lab guide.

## Task 2: Resetting old configurations

- a. Add a new task to the playbook, the task purpose is to clear the old configurations before we apply our own. This task should be applied on the servers group.
- b. Add two additional sub-tasks under the task you configured:
  1. Clear the current eth2 interfaces IP addresses in each server
  2. Clear static routes in each server

```
# /etc/ansible/labPlaybook.yaml

- name: Clear server's old configurations
  hosts: servers
  tasks:
    - name: Clear eth2 interface existing IP
      command: ifconfig eth2 0.0.0.0

    - name: Clear static routes for network 172.30.0.0/16
      command: ip route del 172.30.0.0/16
      ignore_errors: yes
```

### Please note:

Clearing an empty routing table might cause errors, and those errors are not relevant to the playbook execution, therefore should be ignored.

Please make sure that the “**ignore\_errors**” value is set to “**yes**”.

- c. Add a new task to the playbook, the task purpose is to clear the old switch configurations before we apply our own. This task should be applied on the ‘**switches**’ group.
- d. Add one sub-task under the task you configured:
  1. Use the **NCLU** module to clear all switch configurations with the **net del all** command.

```
# /etc/ansible/labPlaybook.yaml
.
.
.
- hosts: switches
  tasks:
    - name: Clear old switch configurations
      nclu:
        commands:
          - del all
        commit: true
```

### Task 3: Configure switches according to the practice exercise

- a. Add a new task to the playbook, the task purpose is to apply configuration that all the switches share. This task should be applied on the **switch group**.
- b. add two sub-task under the task you configured, use **NCLU** module to apply:
  1. configure **trunks** between the switches
  2. create the **VLANs** 2,3.

```
# /etc/ansible/labPlaybook.yaml
.
.
.
- hosts: switches
  tasks:
    - name: Create a bridge and set inter switch links as trunk ports
      nclu:
        commands:
          - add bridge bridge ports swp1-2
        commit: false

    - name: Create VLANs
      nclu:
        commands:
          - add bridge bridge vids 2,3
        commit: true
```

- c. Add a new task to the playbook, the task purpose is to apply configuration that all **leaf** switches share. This task should be applied on the **leaves group**.
- d. add one sub-task under the task you configured, use **NCLU** module to apply:

1. set the host-facing ports as access ports, and associate each interface to it's designated VLAN.

```
# /etc/ansible/labPlaybook.yaml
.
.
.

- hosts: leaves
  tasks:

    - name: Set the host-facing ports as access ports in VLAN 1 and COMMIT
      changes
      nclu:
        commands:
          - add bridge bridge ports swp8-9
          - add interface swp8 bridge access 2
          - add interface swp9 bridge access 3
        commit: true
```

#### Task 4: Configure servers according to the practice exercise

- a. Add 4 new tasks to the playbook, the tasks purpose is to configure the IP address on each server. Each task should be applied on **each server**.

```
# /etc/ansible/labPlaybook.yaml
.
.
.

- name: Configure host1
  hosts: host1
  tasks:
    - name: Configure IP address
      command: ifconfig eth2 172.30.12.18/24

- name: Configure host2
  hosts: host2
  tasks:
    - name: Configure IP address
      command: ifconfig eth2 172.30.13.19/24

- name: Configure host3
  hosts: host3
  tasks:
    - name: Configure IP address
      command: ifconfig eth2 172.30.12.28/24

- name: Configure host4
  hosts: host4
  tasks:
    - name: Configure IP address
      command: ifconfig eth2 172.30.13.29/24
```

- b. Add 2 new tasks to the playbook, the tasks purpose is to configure static routes for each VLAN subnet. Each task should be applied on the servers associated with the VLAN.

#### Task 4: Executing the playbook

- a. Access the Ansible controller server that was assigned to you, and make sure that the host file is configured as follows

**# cat /etc/ansible/hosts**

```
.
.
.

[servers]
host[1:4]

[leaves]
leaf1
leaf2

[spines]
spine3
spine4

[switches:children]
spines
leaves

[switches:vars]
```

```
# /etc/ansible/labPlaybook.yaml
.
.
.

- name: Add static route entry for VLAN 2
  hosts: host1, host3
  tasks:
    - name: Add default gateway for 172.30.0.0/16
      command: ip route add 172.30.0.0/16 dev eth2 via 172.30.12.254

- name: Add static route entry for VLAN 3
  hosts: host2, host4
  tasks:
    - name: Add default gateway for 172.30.0.0/16
      command: ip route add 172.30.0.0/16 dev eth2 via 172.30.13.254
```

Please note:

- Configurations are demonstrated on group B servers and switches.  
You should apply similar commands to devices in the group that was assigned to you.
  - Every line that starts with '#' is considered a comment and can be deleted.
- b. Execute the playbook you wrote  
***# sudo ansible-playbook -b /etc/ansible/LabPlaybook.yaml***
- c. Verify configuration on the servers and switches

