Introduction to Datacenter Automation using Ansible

LTRDCN-2920

Anis Edavalath
Ambili Sasidharan

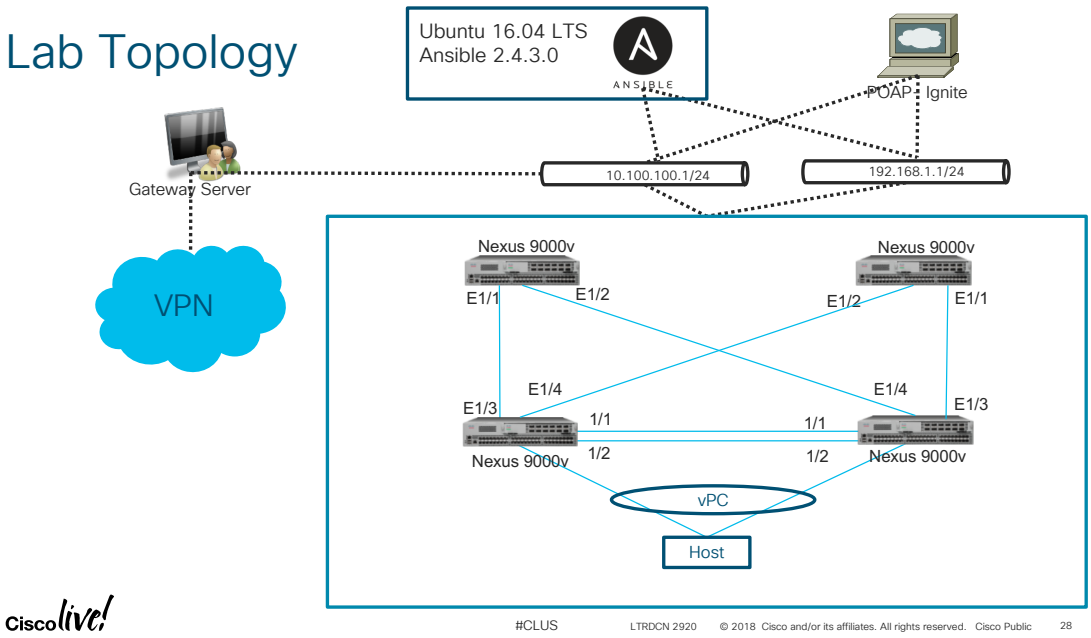
Contents

Learning Objectives

Upon completion of this lab, you will:

- Have a strong foundational knowledge on Ansible
- Install and replicate Ansible environment in your DC Production and QA environment
- Be able to create basic playbooks for DC automation tasks
- Understand the advanced features provided by Ansible
- Understand the Ansible community resources
- Be able to configure a task orchestration using Ansible

Network Diagram



Lab Access:

Use the handout for lab access details.
Login to the ansible control node using the credentials provided.

Module 1: Setting up and verifying the environment

Ansible has been pre-installed on your controller. You can verify the ansible versions and other ansible command line options in this section. In this module, you will create ansible configuration file and inventory file to customize your set up.

Objective: The objective of this module is to verify the pre-installed ansible environment. The required configuration and associated files will be created here. It is recomended to follow the steps exactly as mentioned in this lab guide.

Ansible Summary:
Ansible by default manages machines over the SSH protocol.
Ansible only need to install it on one machine (which could easily be a laptop) and it can manage an entire fleet of remote machines from that central point.

Task 1.1: Create the working directory

Create a new directory called 'playbooks' in under /home/cl-userX . Change directory to playbooks.

```
mkdir playbooks
cd playbooks
```

Note: Everytime we reference working directory, it will be /home/cl-userX/playbooks

Task 1.2: Create ansible.cfg file

Ansible config file is the file where you adjust the default settings based on your requirements. This file can be modified any time to customize the environment. You can use any editor of your choice (ex: vi, vim, nano etc..) to create the 'config' file.

Step 1: In the working directory directory open the file ansible.cfg using the editor of your choice.

```
/home/cl-userxx/playbooks: nano ansible.cfg
```

Step 2: Modify the following parameters in the ansible.cfg file. If its already there, just uncomment it by removing '#' or add the lines below.

```
[defaults]
hostfile =./hosts
host_key_checking = False
timeout = 30
deprecation_warnings= False
retry_files_enabled= False
DEFAULT_LOCAL_TMP = ./.ansible-cl-userX/tmp
```

Note: please user your corresponding user # where you see userX.

Step 3: Save and close the ansible.cfg file

Task 1.3: Configure the Ansible Inventory File

Ansible keeps track of all managed devices through a "hosts" file. We need to set up this file first before we can begin to communicate with our Nexus switches.

Ansible can work against multiple systems in your environment . It does this by selecting the portions of the systems listed in your inventory file, which defaults to /etc/ansible/hosts.

Note: For this lab, we will be configuring the hosts file in the working directory. /home/cl-userxx/

Step 1: Change to working directory and open the file hosts.

```
nano /home/cl-userxx/hosts
```

Step 2: Add the host switches that needs to be managed. Please add the following lines to the hosts file.

```
 [NXOS]

192.168.YYY.1 ansible_user=admin ansible_ssh_pass=nbv12345
192.168.YYY.2 ansible_user=admin ansible_ssh_pass=nbv12345

[ALL:children]

NXOS
```

Step 3: Save and close the host file

Task 1.4: Inventory verification

You can read the contents of inventory file and verify accuracy using the following commands. Give it a try.

```
$ ansible --list-hosts NXOS
$ ansible --list-hosts ALL
```

Task 1.5: Verify installed Ansible version

You can verify the version of ansible installed using the command:

```
ansible --version
```

Task 1.6: Familiarize Ansible Modules

Modules (also referred to as “task plugins” or “library plugins”) are discrete units of code that can be used from the command line or in a playbook task.

To list the installed ansible modules, run the following commands with different options.

```
$ ansible-doc -l
$ ansible-doc -l | wc -l
$ ansible-doc -l | grep ios
$ ansible-doc -l | grep NXOS
```

Task 1.7: Run adhoc commands

An ad-hoc command is something that you might type in to do something really quick, but don't want to save for later. Ad-hoc commands can also be used to do quick things that you might not necessarily want to write a full playbook for. Try the following options.

```
ansible 192.168.YYY.1 -m raw -a "sho interface brief"

ansible NXOS -m raw -a "show clock"

ansible ALL -m raw -a "sho clock"
```

Module 2: Simple Playbooks

Objective: The objective of this lab is to create the simple paybooks . The playbook will contain tasks to :

Use Raw module to execute commands on the switches
Save the results of the commands in to Variables
Print informational output to the console

Playbook Description: The playbook is the basis of ansible automation system. Playbooks can be used to declare configuration variables and orchestrate through the steps of any manual configurations or validations.

- Playbooks are expressed in YAML format.
- Each playbook consists of a list of one or more plays
- Each play consists of one or more tasks
- Each "tasks" section contains a list of modules.
- Each "module" consists a list of actions (~commands).

Task 2.1: Write a playbook to execute show commands

Step 1: Create a directory “module2” inside the working directory and copy the ‘ansible.cfg’ file and ‘hosts’ file to this directory. Change directory to ‘module2’ and ensure ‘ansible.cfg’ file and ‘hosts’ files are in this directory.

```
cd /home/cl-userX/playbooks
mkdir module2
copy ansible.cfg ./module2/
copy hosts ./module2/
cd module2
ls
```

Step 2: use any editor of your choice to create the file simpleplaybook1.yml

```
nano simpleplaybook1.yml
```

Step 3: Add the following playbook parameters to the simpleplaybook.yml file and save it.

```
---
- name: Run show commands
  hosts: NXOS
  gather_facts: false
```

```
connection: local
```

Create a task to find NXOS version

Step 4: Add the following task to the playbook simpleplaybook1.yml

Note that we are using 'nxos_command/ module and 'debug' module in the task. This module prints statements during execution and can be useful for debugging variables or expressions without necessarily halting the playbook.

```
tasks:
  - name: Run show version on nxos switches
    nxos_command:
      commands: show version
    register: version
  - name: Print output
    debug: var=version.stdout_lines
```

Execute the playbook

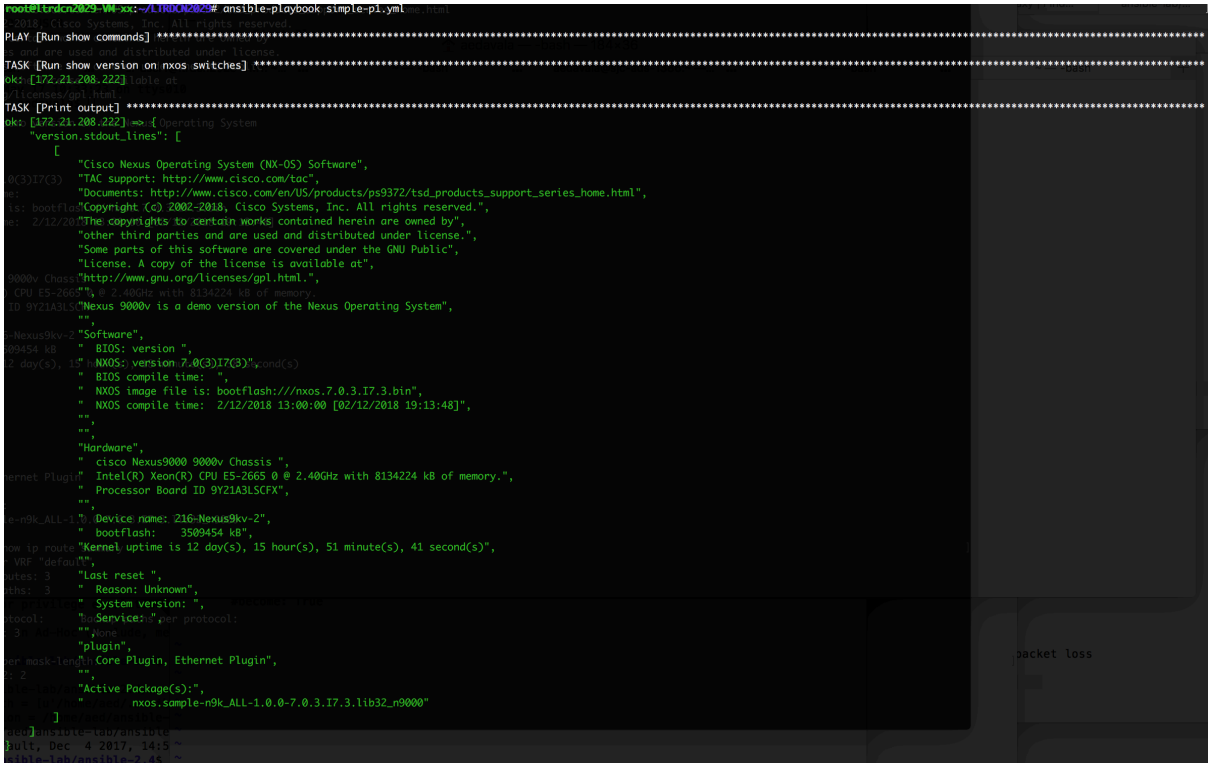
The playbook syntax can be verified by using the switch --syntax check

Step 5: Perform syntax check of the playbook using the command below

```
ansible-playbook simpleplaybook1.yml --syntax-check
```

Step 6: Execute the playbook with the following command

```
ansible-playbook simpleplaybook1.yml
```



Playbook tasks can be executed step by step by using **–step** switch . We can confirm each task one at a time by using this option

Task 2.2 : Using Conditionals

Objective: The objective of this task is to use the 'Conditional' constructs of ansible and find out if the output contains certain string. Here we are searching for the string "NXOS" in the output of show version to determine the type fo the device.

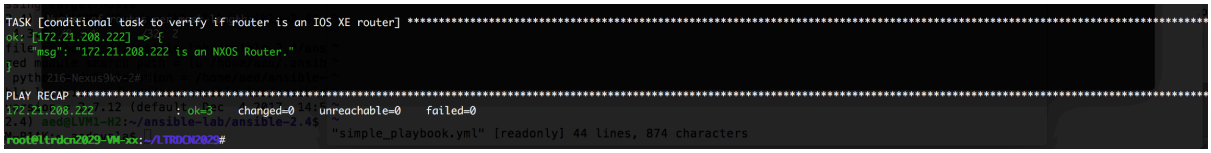
Step 1: **Create task to find the device in a NXOS device**

For this tak we will be using the playbook you created in the previous task. Add the following lines to the playbook simpleplaybook1.yml

```
- name: conditional task to verify if switch is an NXOS switch
  when: version.stdout | join('') is search('NXOS')
  debug: msg="{{inventory_hostname}} is an NXOS Router."
```

Step 2: Execute the playbook again

```
ansible-playbook simple-playbook1.yml
```



Task 2.3: Using Variables

Create playbook to execute show commands with variables

Step 1: Create a playbook with the name simpleplaybook2.yml and add the following lines in it. Variables can be defined at different locations in Ansible. In this task,we are going to get the output of show interface ethernet1/1 using variables. Note that the variable (interface name) is declared in the playbook and referenced inside the task.

```
---
- name: variable demo using NXOS routers show commands
 hosts: NXOS
 gather_facts: false
 connection: local
 vars:
  INTF: Eth1/1

 tasks:
  - name: read config
    nxos_command:
      commands:
        show run int {{INTF}}

    register: INTERFACEoutput

  - name: print output
    debug: var=INTERFACEoutput.stdout_lines
```

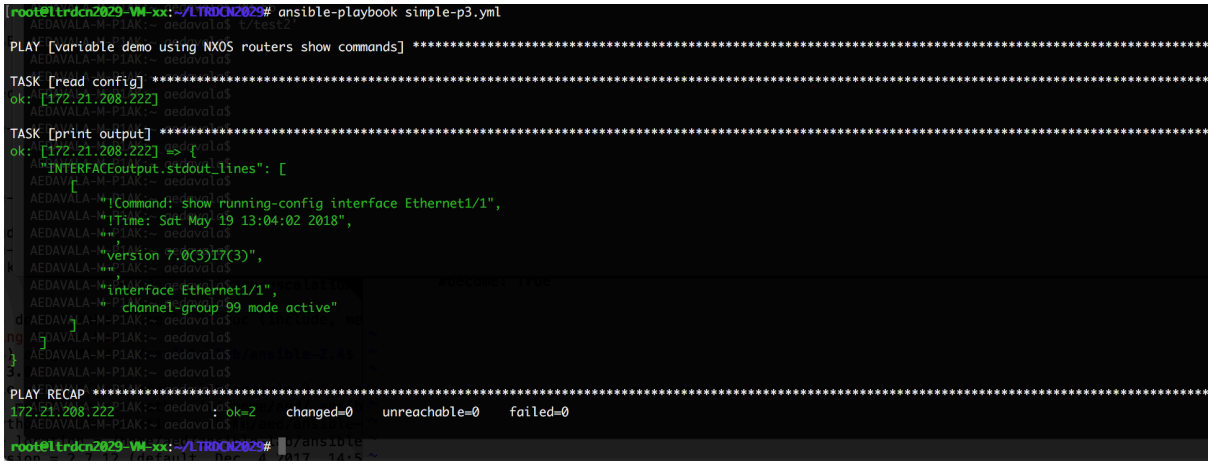
Playbook Execution:

Step 2: Perform syntax check of the play book.

```
ansible-playbook simpleplaybook2.yml --syntax-check
```

Step 3: Run the playbook using

```
ansible-playbook simpleplaybook2.yml
```



Task 2.4 : Using Loops

There are situations when you want to do many things in one task, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached. Loops allow us to repeat the same action against different states.

Create playbook to execute show commands with multiple Items

Step 1: Create a playbook with the name simpleplaybook3.yml and add the following lines in it . Variables can be defined at different locations in Ansible. In this tak, the variables are declared in the playbook and referenced inside the task.

```
---
- name: demo loop constructs in Ansible
  hosts: NXOS
  gather_facts: false
  connection: local

  tasks:
    - name: Collect Router Version and interface brief
      nxos_command:
        commands: "{{ item }}"

      register: LOOPEX

      with_items:
          - show system resources
          - show boot
          - show ip int brief
```

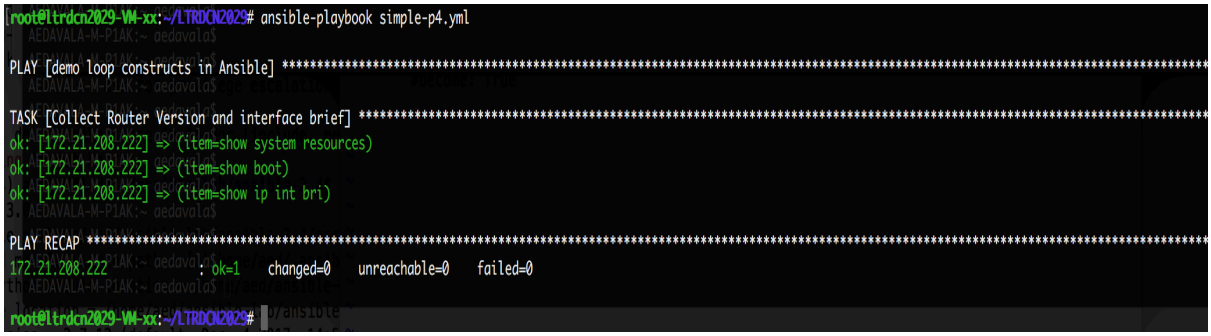
Execute the playbook:

Step 2: Verify the playbook syntax

```
ansible-playbook simpleplaybook3.yml --syntax-check
```

Step 3: Execute the playbook

```
ansible-playbook simpleplaybook3.yml
```


root@ltrdcn2029-VM-xx:~/LTRDCN2029# ansible-playbook simple-p4.yml
PLAY [demo loop constructs in Ansible] ****
TASK [Collect Router Version and interface brief] ****
ok: [172.21.208.222] => (item=show system resources)
ok: [172.21.208.222] => (item=show boot)
ok: [172.21.208.222] => (item=show ip int bri)
PLAY RECAP ****
172.21.208.222 : ok=1 changed=0 unreachable=0 failed=0
root@ltrdcn2029-VM-xx:~/LTRDCN2029#

Module 3: Network Operations tasks automation

Objective: Objective of this module is to automate few use case scenarios commonly observed in network operations.

Before you begin to create the playbook, Create a directory “module3” inside the working ‘/home/cl-userx ‘ directory and copy the ‘ansible.cfg’ file and ‘hosts’ file you created in module 1 to this directory. Change directory to ‘module3’ and ensure ‘ansible.cfg’ file and ‘hosts’ files are in this directory.

```
cd /home/cl-userX/playbooks
mkdir module3
copy ansible.cfg ./module3/
copy hosts ./module3/
cd module3
ls
```

Task 3.1: Playbook to back up switches and save at the server

In this task a playbook is developed to take back up from all the nodes in the inventory and save the backup in a location at the server . Later a cron job can be scheduled to execute this backup script at a specifc time.

Step 1: Create a playbook backup.yml to execute all the commands that is required in the back up configuration. Use loops to iterate through all the required commands

```
---
- name: Get Router Config and state from All Routers
  hosts: NXOS
  gather_facts: no
  connection: local
  tasks:
    - name: Collect Show run from all routers
      nxos_command:
        commands: show run

      register: bkp

    - debug: var=bkp
```

Step 2: Edit the previous playbook backup.yml and add the task to generate the variable with the current time value in it. This will be used to name the backup file.

```
# this line is to set the time to current value
- set_fact: time="{{lookup('pipe','date \"+%Y-%m-%d-%H-%M\"')}}"
```

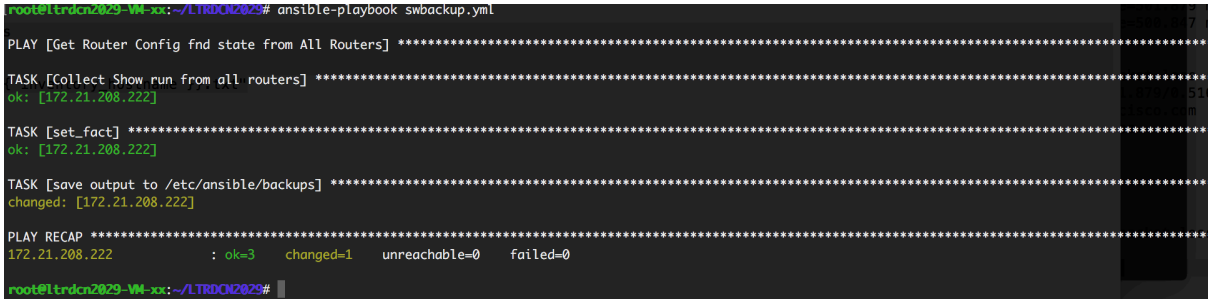
Step 3: Edit the playbook and add the task to copy the running config to a destination on the server. Ansible copy module is used here.

NOTE: Please edit the destination location based on your user#.

```
# this task is using the copy module to copy the file from the nxos device to a server location
    - name: save output to /etc/ansible/backups
      copy:
       content: "{{ bkp.stdout[0] }}"
       dest: "/home/cl-userX/module3/show_run_{{ inventory_hostname }}_run_cfg_{{ time }}.txt"
```

Step 4: Execute the playbook

```
ansible-playbook backup.yml
```



Task 3.2 : Enable features (OSPF, vPC, lacp) on all routers

Modules used: nxos_feature or nxos_command

3.2.1: Using nxos_feature module

Any feature can be enabled on the nxos switches using nxos module. Create a new playbook ‘enable_feature.yml ’to enable the features LACP, VPC and disable OSPF.

```
---
- name: Get Router Config and state from All Routers
  hosts: NXOS
  gather_facts: no
  connection: local
```

```
tasks:
  - name: Ensure lacp is enabled
    nxos_feature:
      feature: lacp
      state: enabled
  - name: Ensure ospf is disabled
    nxos_feature:
       feature: ospf
       state: disabled
  - name: Ensure vpc is enabled
    nxos_feature:
       feature: vpc
       state: enabled
```

3.2.2: Using nxos_command module (Not suggested)

The Nxos automation can be performed by using the raw nxos_command module . Though this is not a suggested approach, we can still configure the switch using this method. Special attention need to be paid to the The configuration hierarchy . Create the following task to your playbook module3.yml

```
---
- name: Get Router Config and state from All Routers
 hosts: NXOS
 gather_facts: no
 connection: local
 tasks:

   - name: fature using command
     nxos_command:
       commands:
       - conf t
       - feature lacp
       - feature vpc
       - feature ospf
```

Task 3.2.3: Verify the switch to see the result

Show run | I feature

Task 3.3: Configure Multiline configurations – Global configs

Objective: utilize nxos_config module to configure multiline configuration lines on nxos devices. We will configure few global parameters on the switch using this module

Task 3.1.1: Create a new playbook spanning-tree-global.yml and paste the following section to it

```
---
- name: Configure spanning tree global parameters

  hosts: NXOS
  gather_facts: false
  connection: local

  tasks:
  # tasks file for spanning_tree

  - name: Configure global STP Parameters
    nxos_config:
    lines:
      - spanning-tree port type edge default
      - spanning-tree port type edge bpduguard default
      - spanning-tree port type edge bpdufilter default
```

Task 3.3.2 : Execute the playbook:

```
ansible-playbook spanning-tree-global.yml
```

Task 3.3.3: verify the configuration on the switch

Task 3.4: Configure Multiline configurations – Access Lists

Objective: utilize nxos_config module to configure multiline configuration lines on nxos devices. We will configure access control configurations on to the switch using this module

Task 3.4.1: Create a new playbook acl.yml and paste the following section

```
---
- name: Configure access control lines
  hosts: NXOS
  gather_facts: false
  connection: local

  tasks:
  # tasks file for access list entries
  - name: Configure global STP Parameters
    nxos_config:
      lines:
        - 10 permit ip 1.1.1.1/32 any log
        - 20 permit ip 2.2.2.2/32 any log
        - 30 permit ip 3.3.3.3/32 any log
        - 40 permit ip 4.4.4.4/32 any log
        - 50 permit ip 5.5.5.5/32 any log
      parents: ip access-list test
      before: no ip access-list test
      match: exact
```

Task 3.4.2 : Execute the playbook

```
ansible-playbook acl.yml
```

Task 3.4.3 : Verify the config on the switch

Module 4: Heirarchical Playbooks for Nxos configuration using Roles

Ansible Roles allows you to break up configuration into more modular steps. Roles are a further level of abstraction that will help organize large playbooks into reusable file structures/files. Roles allow you to create very minimal playbooks that then look to a directory structure to determine the actual configuration steps they need to perform.

In order for ansible to correctly handle roles, we need to build a directory structure that it can find and understand. File structures can be created manually or automatically using via ansible CLI- "ansible-galaxy".

Configuration commands can be automated using the modules which are already available with ansible. Please refer to the ansible documentation for a list of all available modules for different vendors. In this lab, we will be using different nxos modules to configure different configuration sets using roles.

Objective: Learn how various components of ansible roles work together with a simple playbook that uses ansible roles and templates.

In this lab we will create a hirearchial playbook to configure VPC domain using the roles feature in ansible.The following sub tasks need to be completed to configure a vpc domain.

1. Configure vlans
2. Configure uplink ports on interfaces
3. Configure global spaning-tree parameters
4. Configure vpc

Before you begin to create the playbook, Create a directory "module4" inside the working directory '/home/cl-userx ' and copy the 'ansible.cfg' file and 'hosts' file you created in module 1 to this directory. Change directory to 'module4' and ensure 'ansible.cfg' file and 'hosts' files are in this directory.

```
cd /home/cl-userX/playbooks
mkdir module4
copy ansible.cfg ./module4/
copy hosts ./module4/
cd module4
ls
```

Task 4.1: Structuring the playbook

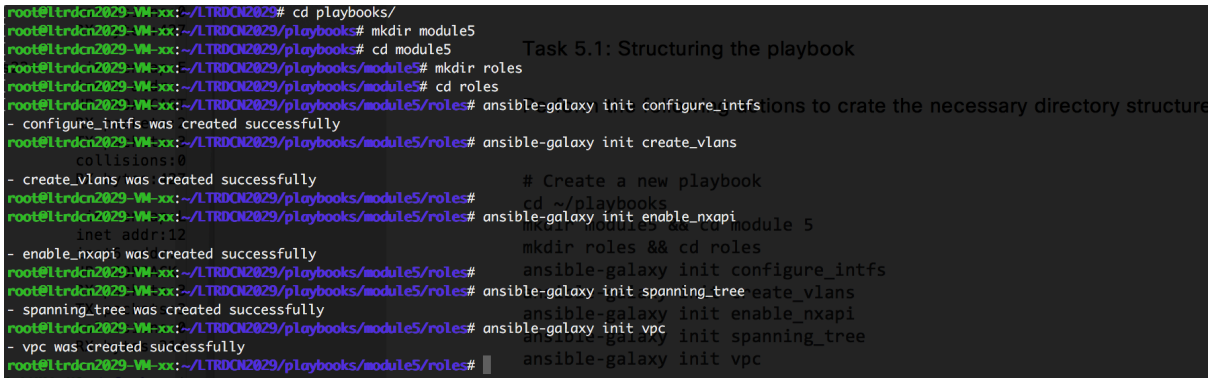
Create a new directory called module4.

Create another directory called "roles" in the directory module4- /home/cl-userX/playbooks/module4.
In order for Ansible to correctly handle roles, we need to build a directory structure that ansible can find and understand. For each roles that we are going to define, we will use ansible-galaxy command to create the required directory structure.

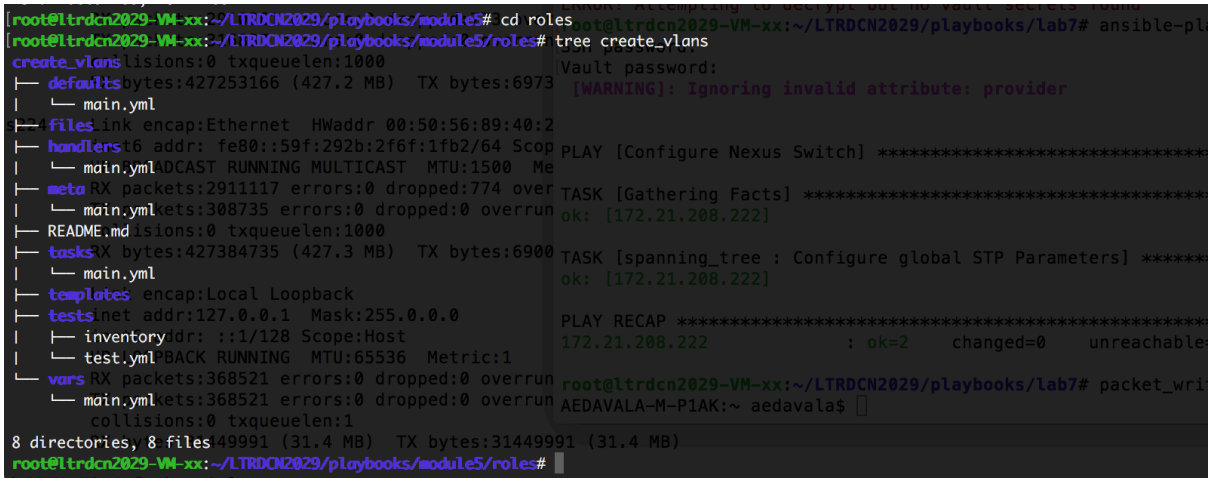
Perform the following actions to crate the necessary directory structure

```
# Create a new playbook
cd ~/playbooks
mkdir module5
cd module 5
mkdir roles
cd roles
ansible-galaxy init configure_intfs
ansible-galaxy init create_vlans
ansible-galaxy init enable_nxapi

ansible-galaxy init spanning_tree
ansible-galaxy init vpc
ansible-galaxy init portchannels
```



this will create the necessary folder structure required for organizing the playbook. Use the tree command to view the folder structure. If tree is not installed in your machine, install it using **sudo apt-get install tree**



In this lab the connection details are bundled using the ***provider*** parameter, which accepts a dictionary with the connection details.

Task 4.2: Organizing the variables

Create a file called global_vars.yml in the directory playbooks/module4 and add the following lines to it . Please pay attention to the formatting.

```
ansible_connection: local
nxos_username: admin
nxos_password: nbv12345
con_details:
        username: "{{ nxos_username }}"
        password: "{{ nxos_password }}"
        host: "{{ inventory_hostname }}"
        transport: cli
```

Create a variable file called host_vars.yml and add the following lines to it

```
---
vlans:
- id: 10
  name: Admin_Vlan
- id: 15
  name: Engineering_Vlan
- id: 20
  name: Finance_Vlan
- id: 25
  name: Marketing_Vlan
- id: 30
```

```
  name: Sales_Vlan
- id: 99
  name: native
- id: 100
  name: peer-keepalive


intf_details:
- id: 1/30
  description: Thirty
  vlan: 10
  mode: access

- id: 1/31
  description: Thirty_One
  vlan: 15
  mode: access

- id: 1/32
  description: Thirty_Two
  vlan: 20
  mode: access

- id: 1/33
  description: Thirty_Three
  vlan: 25
  mode: access

- id: 1/34
  description: Thirty_Four
  vlan: 30
  mode: access

- id: 1/35
  description: Uplink-1
  allowed_vlans: 10,15,20,25,30
  mode: trunk

- id: 1/40
  description: keepalive
  allowed_vlans: 100
  native_vlan: 99
  mode: trunk

vpc:
  - domain: 10
    systempri: 100
    rolepri: 10
    pklsrc: 100.100.100.1
    pkldst: 100.100.100.2
    pklvrf: keepalive
```

Task 4.3: Create roles for vlans

cd to the folder ~/playbooks/module4/roles/create_vlans/tasks and open the file main.yml. Add the following lines to that file

```
# tasks file for configure_vlan
- name: Create VLANs
  nxos_vlan:
      vlan_id: "{{ item.id }}"
      name: "{{ item.name }}"
      provider: "{{ con_details }}"
      state: present
  with_items:
    - "{{ vlans }}"
```

Task 4.4: Create roles for interfaces

cd to the folder ~/playbooks/module5/roles/configure_intfs/tasks and open the file main.yml. Add the following lines to that file

```
# tasks file for configure_vlan
- name: Interface - Put Interface into Default
  nxos_interface:
    interface: Ethernet{{ item.id }}
    provider: "{{ con_details }}"
    state: default
  with_items: "{{ intf_details }}"

- name: Interface - Configure Description and L2 mode for Interface
  nxos_interface:
      interface: Ethernet{{ item.id }}
      description: "{{ item.description }}"
      mode: layer2
      provider: "{{ con_details }}"
      state: present
  with_items: "{{ intf_details }}"

- name: Interface - Configure as access and assign VLAN
  nxos_switchport:
     interface: Ethernet{{ item.id }}
     mode: access
     access_vlan: "{{ item.vlan }}"
     provider: "{{ con_details }}"
  when: item.mode == "access"
  with_items: "{{ intf_details }}"
```

```
- name: Interface - Configure as uplink trunk and assign allowed VLANs
  nxos_switchport:
    interface: Ethernet{{ item.id }}
    mode: trunk
    trunk_vlans: "{{ item.allowed_vlans }}"
    provider: "{{ con_details }}"
  when: item.mode == "trunk"
  with_items: "{{ intf_details }}"

- name: Interface - Configure L2 for peer keepalive
  nxos_switchport:
     interface: Ethernet{{ item.id }}
     mode: trunk
     native_vlans: "{{ item.native_vlan }}"
     provider: "{{ con_details }}"
  when: item.mode == "trunk"
  with_items: "{{ intf_details }}"
```

Task 4.5: Create roles for spanning-tree

cd to the folder ~/playbooks/module4/roles/spanning_tree /tasks and open the file main.yml. Add the following lines to that file

```
# tasks file for spanning_tree

- name: Configure global STP Parameters
  nxos_config:
    provider: "{{ con_details }}"
    lines:
      - spanning-tree port type edge default
      - spanning-tree port type edge bpduguard default
      - spanning-tree port type edge bpdufilter default
- name: Configure network Uplink ports
  nxos_config:
     provider: "{{ con_details }}"
     parents: interface Ethernet{{ item.id }}
     lines:
       - spanning-tree port type network
  with_items: "{{ intf_details }}"
  when: item.mode == "trunk"
```

Task 4.6: Create roles for port channels

cd to the folder ~/playbooks/module5/roles/portchannels/tasks/ and open the file main.yml. Add the following lines to that file

```
---
# tasks file for portchannels
- name: create portchannels
  nxos_portchannel:
    members: "{{ item.int }}"
    group: "{{ item.group }}"
    mode: "{{ item.mode }}"
    state: present
    host: "{{ inventory_hostname }}"
    provider: "{{ con_details }}"
  with_items:
     - {int: ['Eth1/1','Eth1/2'], group: 100, mode: 'active'}
```

Task 4.7: Create roles for vpc

cd to the folder ~/playbooks/module5/roles/vpc/tasks and open the file main.yml. Add the following lines to that file

```
# task file for configuring vpc
- name: global vpc configuration params
  nxos_vpc:
     domain={{ item.domain }}
     system_priority={{ item.systempri }}
     role_priority={{ item.rolepri }}
     pkl_src={{ item.pklsrc }}
     pkl_dest={{ item.pkldest }}
     pkl_vrf={{ item.pkvrf }}
     host={{ inventory_hostname }}
     provider: "{{ con_details }}"

    with_items: vpc

- name: ensure VRF for peer keepalive is created
    nxos_vrf:
      vrf= {{ item.pkvrf }}
      host={{ inventory_hostname }}
      provider: "{{ con_details }}"
```

```
    with_items: vpc
```

Task 4.8: Create roles for vpc peer link confguration channels

cd to the folder ~/playbooks/module5/roles/vpc/tasks and open the file main.yml. Add the following lines to that file

```
# task file for configuring port channels

- name: portchannel vpc peer link configuration

  nxos_vpc_interface:
    portchannel=100
    peer_link=true
    host={{ inventory_hostname }}
    provider: "{{ con_details }}"

- name: portchannel vpc  configuration

    portchannel=11
    vpc=11
    host={{ inventory_hostname }}
    provider: "{{ con_details }}"
```

Task 4.8: Create the main playbook to combine roles

cd to the folder ~/playbooks/module4/ and create the file create_vpc.yml. Add the following lines to that file

```
---
- name: Configure vpc on the datacenter switch
  hosts: NXOS
  vars_files:
    - global_vars.yml
    - host_vars.yml
  roles:
    - { role: create_vlans, tags: [ "vlans" ] }
    - { role: configure_intfs, tags: [ "intfs" ] }
    - { role: spanning_tree, tags: [ "stp" ] }
    - { role: portchannels, tags: [ "portchannels" ] }
    - { role: vpc, tags: [ "vpc" ] }
```

Task 4.9: Execute the playbook

```
Ansible-playbook create_vpc.yml
```

Module 5: Advanced Ansible features

Objective : Familiarize with the additional advanced features available in Ansible including security features and the best practices in using them while automating with Ansible. Ansible Vault , logging best practices, Ansible, Galaxy and error handling will be covered in this module.

Ansiblle Vault: Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. The encrypted vault files can then deistributed in a source control system.

Ansible Vault can encrypt any structured data file . The include Group Vars, Host Vars, Inventory, GlobalVars, variables loaded by “include vars” or vars_files, Role variables and Role Defaults. Similarly, Ansible tasks and Handlers can also be encrypted.

Task 5.1: Encrypt and Decrypt files using ansible vault

Create a directory “module5” inside the working directory and copy the ‘ansible.cfg’ file and ‘hosts’ file to this directory. Change directory to ‘module5’ and ensure ‘ansible.cfg’ file and ‘hosts’ files are in this directory.

```
cd /home/cl-userX/playbooks
mkdir module5
copy ansible.cfg ./module5/
copy hosts ./module5/
cd module5
ls
```

We will demonstrate the use of ansible vault by encryprting the file global_vars. Create a new global_vars file or copy the global_vars file you created in module 4 to the new directory module5

```
Cp /home/cl-userX/playbooks/module4/global_vars.yml  ./
```

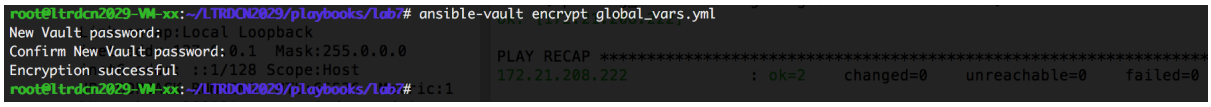
Encrypt the global vars file using Vault:

Execute the following command

```
ansible-vault encrypt global_vars.yml
```

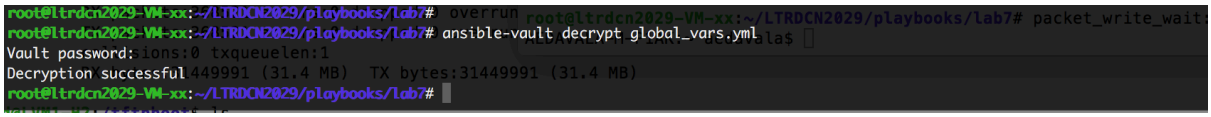
New Vault password:
Confirm New Vault password:

Encryption successful



Similarly to decrypt the file, use the switch – decrypt with above command

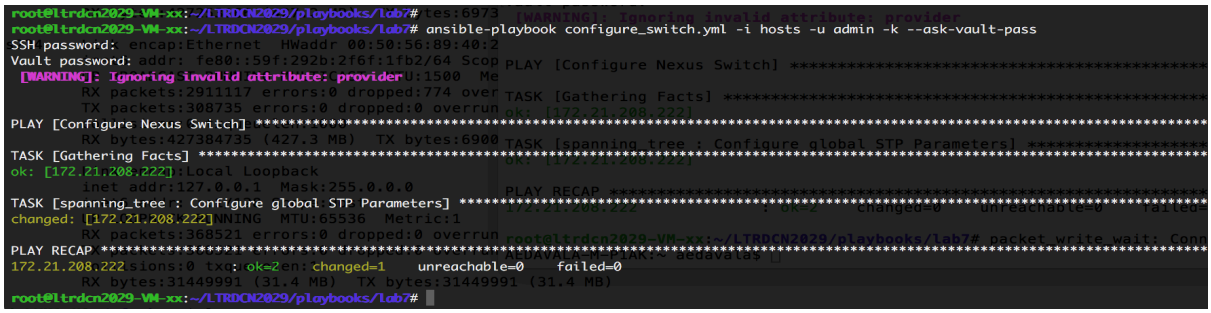
ansible-vault decrypt global_vars.yml



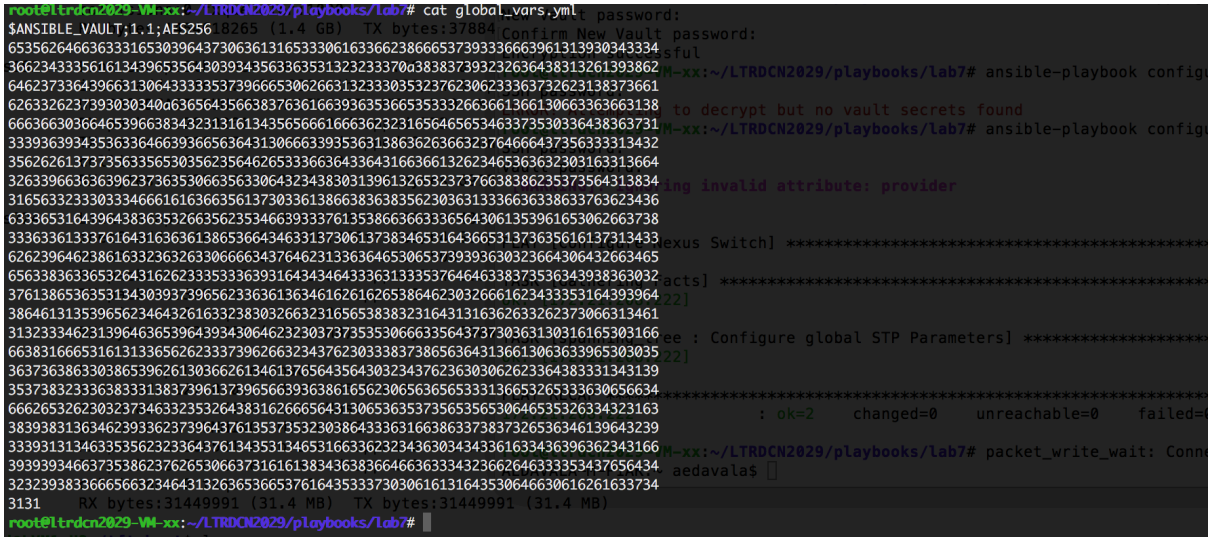
Executing the playbook with encrypted global vars file

Execute the playbook with the switch *--ask-vault-pass* command

Ansible-playbook playbook.yml –i hosts –ask-vault-pass



Verify the encrypted file



Task 5.2: Logging Best Practices

Ansible improved logging to help diagnose and troubleshoot issues regarding Ansible Networking modules. Logging is disabled by default and it can be enabled via ANSIBLE_LOG_PATH and ANSIBLE_DEBUG options.

```
# Specify the location for the log file
export ANSIBLE_LOG_PATH=~/ansible.log

# Enable Debug
export ANSIBLE_DEBUG=True

# Run with 4*v for connection level verbosity
ansible-playbook -vvvv ...
```

After Ansible finished running , you can inspect the logs to check for any troubleshooting

Log Snippet:

```
2018-05-20 11:27:59,758 p=21711 u=root |  PLAY [Configure Nexus Switch]
**********************************************************************************************************
2018-05-20 11:27:59,771 p=21711 u=root |  TASK [Gathering Facts]
**********************************************************************************************************
2018-05-20 11:28:00,615 p=21711 u=root |  ok: [172.21.208.222]
2018-05-20 11:28:00,626 p=21711 u=root |  TASK [spanning_tree : Configure global STP Parameters]
**************************************************************************
2018-05-20 11:28:01,515 p=21765 u=root |  creating new control socket for host 172.21.208.222:22 as user None
2018-05-20 11:28:01,516 p=21765 u=root |  control socket path is /home/cl-user-xx/.ansible/pc/5d868b76d7
```

```
2018-05-20 11:28:01,516 p=21765 u=root |  current working directory is /home/cl-user-xx/LTRDCN2029/playbooks/lab7
2018-05-20 11:28:01,516 p=21765 u=root |  using connection plugin network_cli
2018-05-20 11:28:01,565 paramiko.transport starting thread (client mode): 0xb768ffd0L
2018-05-20 11:28:01,566 paramiko.transport Local version/idstring: SSH-2.0-paramiko_2.4.1
```

p=21765 is the Process ID (PID) for ansible connection process

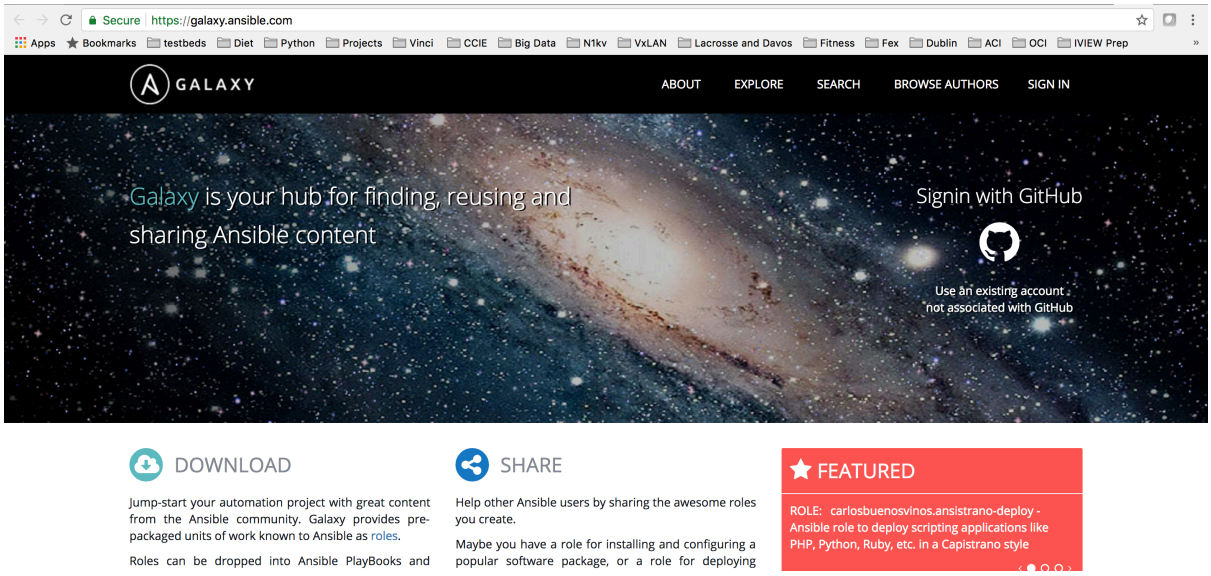
u=root is the user running ansible. Not the remote-user attempting to connect to device

Task 5.3 Ansible Galaxy

Galaxy, is a free site for finding, downloading, and sharing community developed roles. Downloading roles from Galaxy is a great way to jumpstart your automation projects.

You can also use the site to share roles that you create. By authenticating with the site using your GitHub account, you're able to *import* roles, making them available to the Ansible community. Imported roles become available in the Galaxy search index and visible on the site, allowing users to discover and download them.

Ansible's official community for sharing the roles developed by users.



Task 5.3.1: Install a role from ansible-galaxy

```
ansible-galaxy install robertwatson3.nxos_prepare
```

```
root@ltrdcn2029-VM-xx:~# ansible-galaxy install robertwatson3.nxos_prepare
- downloading role 'nxos_prepare', owned by robertwatson3
- downloading role from https://github.com/bobbywatson3/nxos_prepare/archive/master.tar.gz
- extracting robertwatson3.nxos_prepare to /home/cl-user-xx/.ansible/roles/robertwatson3.nxos_prepare
- robertwatson3.nxos_prepare (master) was installed successfully
root@ltrdcn2029-VM-xx:~#
```

The files will be downloaded in to the hidden folder in your home directory

```
root@ltrdcn2029-VM-xx:~# ll ~/.ansible/roles/
total 12
drwxr-xr-x 3 root root 4096 May 20 23:33 ./
drwx------ 6 root root 4096 May 20 23:33 ../
drwxr-xr-x 8 root root 4096 May 20 23:33 robertwatson3.nxos_prepare/
root@ltrdcn2029-VM-xx:~#
```

Task 5.3.2: Reuse the role in our playbook

Create a playbook galaxy-pb.yml and add the following lines to it

```
---
- hosts: nxos
  connection: local
  gather_facts: yes
  force_handlers: True

  vars:
    config_backup_path: "bootflash:/configs/config_backup.cfg"
    backup: True
    disable_nxapi: False

  vars_prompt:
    - name: switch_username
      prompt: "What is the switch username?"
      private: False
    - name: switch_password
      prompt: "What is the switch password?"

  roles:
    - nxos_prepare
```

Appendix 1: Optional Case Study - EVPN Based Vxlan DC Fabric Buildout

Objective : Utilize the ansible learning to familiarize with the steps required to build an evpn Vxlan datacenter Fabric from scratch. The Underlay will be configured in one playbook and the overlay is provisioned using the second one. Roles are used to organize the different tasks involved here. Refer to the Topology diagram provided for each user for reference.

The customized files for this exercise is available at the location specified in the lab handout . The users can download the files and modify the parameters specific to their testbed

Underlay Configurations:

1. Project Files: Copy the Project files to the "Playbooks" directory in your home directory, as specified in the Lab Access Handout. Untar the files in and follow this guide to customize

2. Config File: Open the "ansible.cfg" in your project directory and verify the following lines are present.

```
[defaults]
hostfile = ./hosts
host_key_checking = False
timeout = 30
deprecation_warnings= False
retry_files_enabled= False
```

3. Hosts file: Verify the file "hosts" in this directory and add the following lines.

```
[all:vars]
ansible_connection = local
un=admin
pwd=nbv12345

[spine]
218-spine1 ------------------→ replace with the name of your spine1
218-spine2 ------------------→ replace with the name of your spine2

[leaf]
user1-leaf1 ------------------→ replace with the name of your leaf1
user1-leaf2 ------------------→ replace with the name of your leaf2
```

4. host_vars Directory: Create the directory "host_vars" in your home directory and copy 4 files – one file for each f the switches in the topology (spine1, spine2, leaf1 and leaf2). All the files that is needed for this exercise is available in the tar format. Follow the download instruction provided in the Lb Access

handout to get the right files for your lab setup. Please pay attention to the naming convention of the files in this directory.

```
root@cl-cntrlr:~/ playbooks/vxlan-evpn/host_vars# ls -l
total 16
-rw-r--r-- 1 root root 225 Jun  4 18:03 218-spine1.yml
-rw-r--r-- 1 root root 223 Jun  4 18:03 218-spine2.yml
-rw-r--r-- 1 root root 232 Jun  5 13:48 user1-leaf1.yml
-rw-r--r-- 1 root root 233 Jun  5 14:16 user1-leaf2.yml
root@cl-cntrlr:~/ playbooks/vxlan-evpn/host_vars#
```

5. **Roles Directory:**

The underlay configuration uses two roles.

1. Spine
2. Leaf

Each role contains two folders under them with a file – "main.yml"

1. Tasks
2. Vars

6. **Role – Spine**

Open the file main.yml in the tasks directory under role spine.

All the underlay configuration required on spine switches are available in this file.

`~/playbooks/vxlan-evpn/roles/spine/tasks/main.yml`

All the role specific variables required to configure spine switches are available in this file.

`~/playbooks/vxlan-evpn/roles/spine/vars/main.yml`

7. Role - Leaf

Open the file main.yml in the tasks directory under role leaf.

All the underlay configuration required on leaf switches are available in this file.

`~/playbooks/vxlan-evpn/roles/leaf/tasks/main.yml`

All the role specific variables required to configure leaf switches are available in this file.

`~/playbooks/vxlan-evpn/roles/leaf/vars/main.yml`

8. Site.yml

site.yml available in the directory vxlan-evpn is the main playbook, which calls the two different roles required for the underlay configuration

```
~/playbooks/vxlan-evpn/site.yml
```

Overlay Configurations:

Overlay configuration is required only on the leaf switches. Untar the file “vxlan-evpn-overlay.tar” in your playbooks directory

1. Modify the hosts file to suite your environment

2. All the configurations required for overlay is provided in the role directory .

```
~/playbooks/vxlan-evpn-overlay/roles/leaf/tasks/main.yml
```

The leaf specific variables are available in the vars directory

```
~/playbooks/vxlan-evpn-overlay/roles/leaf/vars/main.yml
```

3. The roles are executed using the main file “site.yml” in the directory vxlan-evpn-overlay

```
~/playbooks/vxlan-evpn-overlay/site.yml
```

Appendix 2: Setting up the env and install Ansible

Objective: The objective of this module is to install Ansible controller in Ubuntu linux VM. The required configuration and associated files will be created here. It is recomended to follow the steps exactly as mentioned in this lab guide.

Ansible Summary:
Ansible by default manages machines over the SSH protocol.
Ansible only need to install it on one machine (which could easily be a laptop) and it can manage an entire fleet of remote machines from that central point.

Control Machine Requirements:
Currently Ansible can be run from any machine with Python 2.6 or 2.7 installed (Windows isn't supported for the control machine).

Installing Ansible on Ubuntu

Step 1: Open the SSH client and login to your assigned control VM with the provided credentials. Elevate to sudo privilege.

```
ssh cl-userx@172.21.208.ABC

cl-userx@172.21.208.ABC's password:

cl-userx @ltrdcn2029-VM:~ sudo -s

[sudo] password for cl-userx:

root@ltrdcn2029-VM:~
```

```
Last login: Sat May 19 00:31:32 on ttys019
AEDAVALA-M-P1AK:~ aedavala$ ssh cl-user1@172.21.208.233
cl-user1@172.21.208.233's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-31-generic x86_64)

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

622 packages can be updated.
336 updates are security updates.

Last login: Fri May 18 14:00:59 2018 from 10.20.192.212
cl-user1@ltrdcn2029-VM1:~$ sudo -s
[sudo] password for cl-user1:
root@ltrdcn2029-VM1:~#
```

Step 2: Install the packages

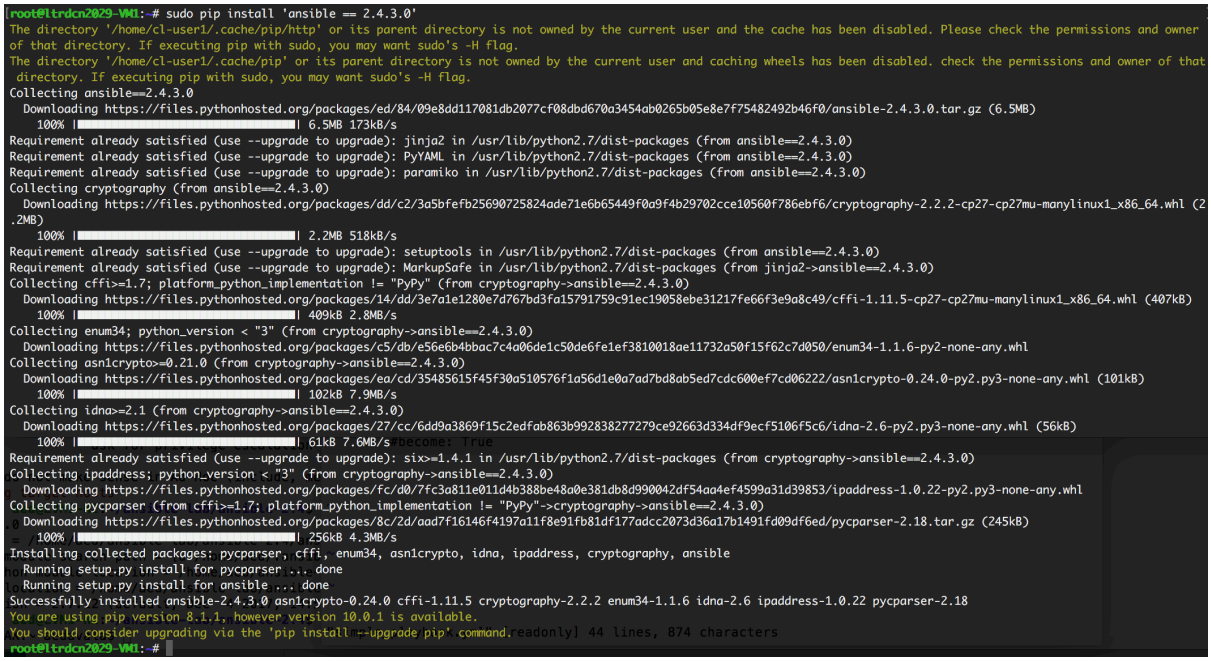
root@ltrdcn2029-VM:~sudo apt-get update

root@ltrdcn2029-VM:~sudo apt-get install python-pip

root@ltrdcn2029-VM:~sudo pip install ‘ansible == 2.4.3.0’

```
root@ltrdcn2029-VM1:~# sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
root@ltrdcn2029-VM1:~#
```

```
root@ltrdcn2029-VM1:~# sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  ieee-data python-crypto python-ecdsa python-httplib2 python-jinja2 python-markupsafe python-netaddr python-paramiko python-selinux python-six python-yaml
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libexpat1 libexpat1-dev libpython-all-dev libpython-dev libpython-stdlib libpython2.7 libpython2.7-dev libpython2.7-minimal libpython2.7-stdlib python python-all
  python-all-dev python-dev python-minimal python-pip-whl python-setuptools python-wheel python2.7 python2.7-dev python2.7-minimal
Suggested packages:
  python-doc python-tk python-setuptools-doc python2.7-doc binfmt-support
The following NEW packages will be installed:
  libexpat1-dev libpython-all-dev libpython-dev libpython2.7-dev python-all python-all-dev python-dev python-pip python-pip-whl python-setuptools python-wheel python2.7-dev
The following packages will be upgraded:
  libexpat1 libpython-stdlib libpython2.7 libpython2.7-minimal libpython2.7-stdlib python python-minimal python2.7 python2.7-minimal
9 upgraded, 12 newly installed, 0 to remove and 596 not upgraded.
Need to get 34.7 MB of archives.
After this operation, 44.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython2.7 amd64 2.7.12-1ubuntu0~16.04.3 [1,070 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python2.7 amd64 2.7.12-1ubuntu0~16.04.3 [224 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libexpat1 amd64 2.1.0-7ubuntu0.16.04.3 [71.2 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython2.7-stdlib amd64 2.7.12-1ubuntu0~16.04.3 [1,880 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python2.7-minimal amd64 2.7.12-1ubuntu0~16.04.3 [1,261 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython2.7-minimal amd64 2.7.12-1ubuntu0~16.04.3 [340 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python-minimal amd64 2.7.12-1~16.04 [28.1 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python amd64 2.7.12-1~16.04 [137 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython-stdlib amd64 2.7.12-1~16.04 [7,768 B]
Get:10 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libexpat1-dev amd64 2.1.0-7ubuntu0.16.04.3 [115 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython2.7-dev amd64 2.7.12-1ubuntu0~16.04.3 [27.8 MB]
Get:12 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython-dev amd64 2.7.12-1~16.04 [7,840 B]
Get:13 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpython-all-dev amd64 2.7.12-1~16.04 [1,006 B]
Get:14 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python-all amd64 2.7.12-1~16.04 [996 B]
Get:15 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 python2.7-dev amd64 2.7.12-1ubuntu0~16.04.3 [276 kB]
```


root@ltrdcn2029-VM1:~# sudo pip install 'ansible == 2.4.3.0'
The directory '/home/cl-user1/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/cl-user1/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting ansible==2.4.3.0
Downloading https://files.pythonhosted.org/packages/ed/84/09e8dd117081db2077cf08dbd670a3454ab0265b05e8e7f75482492b46f0/ansible-2.4.3.0.tar.gz (6.5MB)
100% | 6.5MB 173kB/s
Requirement already satisfied (use --upgrade to upgrade): jinja2 in /usr/lib/python2.7/dist-packages (from ansible==2.4.3.0)
Requirement already satisfied (use --upgrade to upgrade): PyYAML in /usr/lib/python2.7/dist-packages (from ansible==2.4.3.0)
Requirement already satisfied (use --upgrade to upgrade): paramiko in /usr/lib/python2.7/dist-packages (from ansible==2.4.3.0)
Collecting cryptography (from ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/dd/c2/3a5bfefb25690725824ade71e6b65449f0a9f4b29702cce10560f786ebf6/cryptography-2.2.2-cp27-cp27mu-manylinux1_x86_64.whl (2.2MB)
100% | 2.2MB 518kB/s
Requirement already satisfied (use --upgrade to upgrade): setuptools in /usr/lib/python2.7/dist-packages (from ansible==2.4.3.0)
Requirement already satisfied (use --upgrade to upgrade): MarkupSafe in /usr/lib/python2.7/dist-packages (from jinja2->ansible==2.4.3.0)
Collecting cffi>=1.7; platform_python_implementation != "PyPy" (from cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/14/dd/3e7a1e1280e7d767bd3fa15791759c91ec19058ebe31217fe66f3e9a8c49/cffi-1.11.5-cp27-cp27mu-manylinux1_x86_64.whl (407kB)
100% | 409kB 2.8MB/s
Collecting enum34; python_version < "3" (from cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/c5/db/e56e6b4bbac7c4a06de1c50de6fe1ef3810018ae11732a50f15f62c7d050/enum34-1.1.6-py2-none-any.whl
Collecting asn1crypto>=0.21.0 (from cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/ea/cd/35485615f45f30a510576f1a56d1e0a7ad7bd8ab5ed7cdc600ef7cd06222/asn1crypto-0.24.0-py2.py3-none-any.whl (101kB)
100% | 102kB 7.9MB/s
Collecting idna>=2.1 (from cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/27/cc/6dd9a3869f15c2edfab863b992838277279ce92663d334df9ecf5106f5c6/idna-2.6-py2.py3-none-any.whl (56kB)
100% | 61kB 7.6MB/s
Requirement already satisfied (use --upgrade to upgrade): six>=1.4.1 in /usr/lib/python2.7/dist-packages (from cryptography->ansible==2.4.3.0)
Collecting ipaddress; python_version < "3" (from cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/fc/d0/7fc3a811e011d4b388be48a0e381db8d990042df54aa4ef4599a31d39853/ipaddress-1.0.22-py2.py3-none-any.whl
Collecting pycparser (from cffi>=1.7; platform_python_implementation != "PyPy"->cryptography->ansible==2.4.3.0)
Downloading https://files.pythonhosted.org/packages/8c/2d/aad7f16146f4197a11f8e91fb81df177adcc2073d36a17b1491fd09df6ed/pycparser-2.18.tar.gz (245kB)
100% | 256kB 4.3MB/s
Installing collected packages: pycparser, cffi, enum34, asn1crypto, idna, ipaddress, cryptography, ansible
Running setup.py install for pycparser ... done
Running setup.py install for ansible ... done
Successfully installed ansible-2.4.3.0 asn1crypto-0.24.0 cffi-1.11.5 cryptography-2.2.2 enum34-1.1.6 idna-2.6 ipaddress-1.0.22 pycparser-2.18
You are using pip version 8.1.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
root@ltrdcn2029-VM1:~#

Appendix 3: Sample Play-books

Write a playbook to create vlan 99 using variables defined in playbook
Step1: Open a playbook vlan.yml using your favorite editor and copy the playbook given below.

```
---
- name: Configure vlan 99
  hosts: NXOS
  connection: local
  vars:
    vlan_id: 99
    vlan_name: Ansible_VLAN

  tasks:
    - name: Configure VLAN ID
      nxos_config:
        lines:
          - vlan {{ vlan_id }}

    - name: Configure VLAN Name
      nxos_config:
        lines:
          - name {{ vlan_name }}
        parents: vlan {{ vlan_id }}

    - name: Show VLAN
      nxos_command:
        commands: show vlan brief
      register: show_vlan

    - debug: var=show_vlan.stdout_lines
```

Appendix 4: More resources

Ansible Doumentation
Ansible Galaxy official community