Lab 5 – Playbooks

Introduction

Playbooks are a lot like instruction manuals for your orchestration or whatever task you want to accomplish. It is our way of telling Ansible what to do and the order to do them in.

They are based on the YAML format that we have already discussed and remember YAML is based a lot on spacing and correct formatting.

Please refer to the **Lab Connection Information.docx** file for information on connecting to your Ansible host.

Don't forget to change the XX in your inventory file based on your Pod information.

1. Ansible Plays and Tasks

1.1 The first topic we are going to go over are the actual "Plays" which are really just a group of hosts mapped to tasks or roles that tell it what to do. A task is a call to a ansible module like NXOS, JUNOS, IOS, etc to perform a specific action.

As we go along we will discuss how you can have multiple plays in a playbook to complete multiple tasks or a stage of tasks for a certain job you are trying to perform.

Let's start with this very basic playbook example provided below to you and let's run this playbook and see what our output is and then we will discuss what is happening here.

Let's make sure we are in the lab5 dir when working on this lab. When it asks for a pw specify the Nexus 9K pw on your sheet.

In our command we are specifying the -u to supply a username and the -k to ask for a password

The first command is a way we can see what hosts this playbook would run against, then we actually run our playbook.

```
cd ~/ansible_labs/lab5-playbooks
```

```
ansible-playbook -i inventory nxos_facts_play.yaml --list-hosts

ansible-playbook -i inventory nxos_facts_play.yaml -u admin
-k -e ansible_network_os=nxos
```

We should see some output similar to below for the two commands:

Notice how it shows us what host this playbook will run against.

```
playbook: nxos_facts_play.yaml

  play #1 (nxos): Gather Nexus Facts    TAGS: []

    pattern: [u'nxos']

    hosts (1):

      n9k-standalone-XX.localdomain
```

```
PLAY [Gather Nexus Facts] ***********************************
*************************************************************
***********************



TASK [Gathering Facts] **************************************
*************************************************************
***********************

ok: [n9k-standalone-XX.localdomain]
```

```
TASK [nxos_facts] ****************************************
************************************************************
***********************

ok: [n9k-standalone-XX.localdomain]



TASK [Get All Ip Addresses] ********************************
************************************************************
***********************

ok: [n9k-standalone-XX.localdomain] => {

    "ansible_net_all_ipv4_addresses": [

        "10.1.150.13",

        "192.169.1.1",

        "192.169.1.5",

        "192.168.1.1",

        "172.100.1.1",

        "172.101.1.1",

        "172.102.1.1"

    ]

}



PLAY RECAP ************************************************
************************************************************
***********************
```

```
n9k-standalone-XX.localdomain : ok=3    changed=0    unreach
able=0    failed=0
```

Now let’s talk about this playbook and what each piece is doing.

1.2 The top section of the playbook is where we will start first.

```
---



- name: Gather Nexus Facts

  hosts: nxos

  connection: httpapi

  gather_facts: yes
```

So we start by using the --- which in YAML specifies the beginning of the file, this is optional.

Next we specify our **hosts**. In ansible playbooks the hosts line can be one more more groups or host patterns, etc. We will play with a few different options throughout the lab to familiarize ourselves with it.

The **connection** line is a new option as of 2.5 that is used to specify the provider you are using. In this case we are using **httpapi** and specifying the nxapi as a transport option to use the NXAPI.

This used to be the **provider** line which is now deprecated. The NXOS module allows for 2 options here either we can use **network_cli** which runs everything as CLI connections or we can use local with a **transport** of **nxapi** to use the NXOS API built into the switch.

Since we want to gather facts about the device and will be using it in the playbook we have set this to yes, this could be set to no if you aren’t using any of the facts data.

1.3 Now let's talk about the actual tasks or plays in this playbook.

```
tasks:

    - nxos_facts:

    - name: Get All Ip Addresses

      debug: var=ansible_net_all_ipv4_addresses
```

So you can see we have a tasks line setup here with some objects underneath it. Since we only have 1 task this is considered a 1 Play playbook. We could have multiple tasks setup and trigger multiple plays in our playbook.

A use case for multiple plays might be 1 task to configure a router, and another task or play to configure a switch but it's a part of a bigger playbook to setup a new network/etc. We will discuss this further as the lab progresses.

Each play will consist of a list of tasks like we have above here, they are executed in order one at a time against the host pattern we specify.

Our first task here is to just run **nxos_facts** which just gathers information about the device via the facts module. We can then use the different variables associated with this module that are documented https://docs.ansible.com/ansible/latest/modules/nxos_facts_module.html#nxos-facts-module and debug them to print them out to view and use them in our playbook logic.

The **name** line is just for giving our task a name so that when its running in the CLI we can see the name come up and know what part of the playbook it is on.

Right below name we are specifying the module **debug** and giving it the var **ansible_net_all_ipv4_addresses** which is a variable that was documented on the module information above.

2. Create Multi Task Playbook

Use Case: Need a report of all switch versions and models

2.1 Using the above information as a reference now we want you to create a 2 task playbook that will achieve the following:

Your working folder will be **section2_multitask**

1. Gather the facts for the host group **switches**
2. Use the inventory file named **sectiontwo_inv** in your directory when running playbook.
3. Debug the output of the model of the device.
4. Debug the output of all system version on the device.
5. Please use the nxos_facts module to accomplish this.
6. You can use the existing inventory file and create your own playbook for this, call the playbook **two_task.yaml** for consistency.
7. Don't forget to use the **-e ansible_network_os=nxos** in your command.
8. When done and have it working add this file to your git repo, commit it, and push it up.

3. Host & Group Vars

3.1 While you can specify variables in the inventory file it is not best practice or recommended. That is where group and host variables come in to play.

We can create files with either no extension or .yml, .yaml, or .json relative to the location of the inventory file to specify values we want to use for certain groups or hosts, this way we don't have to specify it each time in a playbook.

In our example we have a folder called **group_vars** that we created in the same folder here as our inventory file. This means anything we specify in this folder will be used with our inventory file when we call playbooks referencing it.

Ansible will try automatically to use group vars that are relative to the inventory file and match the group specified, so we don't have to specify anything in our playbook but our host group or host that we have created vars for.

Ensure you are in the lab5 -playbooks/section5_snmp folder.

```
cd lab5-playbooks/section5_snmp
```

Let’s take a look at the directory structure of the **group_vars** folder.

```
ls -lah group_vars
```

Notice how we are getting back there is a file called nxos

```
total 4.0K

drwxr-xr-x. 2 root root  18 Aug 24 10:56 .

drwxr-xr-x. 4 root root 213 Aug 24 10:49 ..

-rw-r--r--. 1 root root  88 Aug 24 10:56 nxos
```

Let’s cat this nxos file and see what is inside it.

```
cat group_vars/nxos
```

Here we can see some of the ansible variables we are specifying so we don’t have to do it either in the playbook or on the CLI when calling the playbook

```
ansible_connection: httpapi

ansible_network_os: nxos

ansible_user: admin

transport: nxapi
```

Remember before we had to specify the connection type and such before in the playbook? Well now we are using the **ansible_connection** variable to set this as a group var so we don’t have to anymore.

We are also telling ansible this is a **network_os** of NXOS and are also giving it the **ansible_user** to connect with in this case admin. This way we no longer have to specify -u admin on the CLI when running our playbook.

The transport is there as a requirement for the NXOS module and so we specify that as well here to prevent from having issues with using the NXAPI.

What are some other things you can think of we can use as variables?

Why would we not want to store a secret or secure variable in this group var file?

How might you use this in your own playbooks?

4. Assert & Verifying Changes

4.1 When we make changes we can use a module known as assert in order to verify our changes were truly made. In the examples below we are going to modify the port description for a port then using the nxos_facts module and assert we will verify it was modified.

4.2 First thing we need to do is create a playbook in order to accomplish this, we can copy our existing nxos_facts_play.yaml as a starter to modify and work from.

```
cp nxos_facts_play.yaml assert_example.yaml
```

Now we have an assert_example.yaml we can modify.

4.3 We will now modify this file based on what we have learned and using the nxos_interface module to modify Ethernet 1/1 description to “uplink to core”

We will perform the following tasks in the playbook.

1. Change the port description on eth 1/1 to uplink to core.
2. Debug to show the output of the new value after we re-gather the facts again.
3. Using assert with a jinja template to verify that our interface Ethernet1/1 description was actually changed to “uplink to core”.

Note: The environment variable here that nxos supplies treats the output in quotes (to escape the quotes around uplink to core).

```
---

- name: Change Ethernet Description and Assert

  hosts: nxos

  connection: httpapi

  gather_facts: yes


  tasks:

    - name: Change Eth 1/1 Port Description

      nxos_interface:

        name: Ethernet1/1

        description: 'uplink to core'

    - nxos_facts:

    - name: Debuging Description

      debug: var=ansible_net_interfaces['Ethernet1/1']['desc
ription']

    - name: Verify change was made

      assert: { that: "ansible_net_interfaces['Ethernet1/1']
['description'] == '\"uplink to core\"'" }
```

4.4 Now if we run this it should change our description and then tell us at the end that all assertions passed so we know that our change was made.

```
ansible-playbook -i inventory assert_example.yaml -u admin -
k
```

We should see something like below:

```
PLAY [Change Ethernet Description and Assert] **************
************************************************************
***********************



TASK [Gathering Facts] *************************************
************************************************************
***********************

ok: [n9k-standalone-01.localdomain]



TASK [Change Eth 1/1 Port Description] *********************
************************************************************
***********************

changed: [n9k-standalone-XX.localdomain]
```

```
TASK [nxos_facts] ******************************************
************************************************************
**********************

ok: [n9k-standalone-XX.localdomain]



TASK [Debuging Description] ********************************
************************************************************
**********************

ok: [n9k-standalone-XX.localdomain] => {

    "ansible_net_interfaces['Ethernet1/1']['description']":
"uplink to core"

}



TASK [Verify change was made] ******************************
************************************************************
**********************

ok: [n9k-standalone-XX.localdomain] => {

    "changed": false,

    "msg": "All assertions passed"

}



PLAY RECAP *************************************************
************************************************************
**********************
```

```
n9k-standalone-XX.localdomain : ok=5    changed=1    unreach
able=0    failed=0
```

5. Knowledge Check - Configure SNMP Community String

Use Case: Need to setup new monitoring software on all switches that requires a new SNMP community string to be created across all switches. Assert to ensure the change was made.

With the supplied inventory file and **group_vars/nxos** group file configure a playbook that can create or delete a snmp community string by simply changing the **group_vars/nxos** group file value.

Your working directory for this section will be folder **section5_snmp**

1. Documentation for the module we will use can be found at https://docs.ansible.com/ansible/latest/modules/nxos_snmp_community_module.html#nxos-snmp-community-module
2. We have supplied the nxos group file with the keys, you can fill out the values with what is required for the module to work. Remember in your playbook for now you can use "{{ variable }}" to replace a value from your group vars file. We will discuss this more in the upcoming labs.
3. Using these values created in the nxos group file create the playbook that will use these values and create a snmp community string.
4. Use assert to verify that the change was made.

Version Control Commit

Now add all your new files to your repository and push it up, reference the GitHub lab if you get stuck or ask for help.