Lab 8 – Loops

Introduction

There may be instances where you have a list of items you need to loop through and configure or create a lot of user id’s, ntp entries, etc. Loops allow us to achieve these types of results.

Please refer to the **Ansible-Pod-Info.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. Standard Loops

1.1 We can use standard loops to save typing and allow for repeated tasks to be written in short hand. We will use the short loop to loop through a few VLAN’s we want to create on a switch.

Ensure you are in the proper folder for this lab:

```
cd ~/ansible_labs/lab8-loops
```

1.2 Using what you already know create a **group_vars** structure, **inventory** file, **ansible.cfg** file and **base playbook** to run against your Nexus switch. We have provided the below task for you to use in the playbook to test out loop functionality.

Name the playbook **vlan_loop.yaml**

HINT: You could copy the group_vars and such from the previous lab to save time.

Loop Vlan Task

```
- name: Add Multiple Vlans

      nxos_vlan:
        vlan_id: "{{ item }}"

        state: present

      loop:

        - 50

        - 60

        - 70
```

So now we should have a complete playbook we can launch and create vlan 50, 60 and 70 respectively.

```
ansible-playbook vlan_loop.yaml --ask-vault
```

We should have output like the below:

```


Vault password:



PLAY [Loop Example] ****************************************
************************************************************
**



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

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

```
TASK [Add Multiple Vlans] ***********************************
*************************************************************
**

changed: [n9k-standalone-XX.localdomain] => (item=50)

changed: [n9k-standalone-XX.localdomain] => (item=60)

changed: [n9k-standalone-XX.localdomain] => (item=70)



PLAY RECAP **************************************************
*************************************************************
**

n9k-standalone-XX.localdomain : ok=2    changed=1    unreach
able=0    failed=0
```

Notice how it looped through each item and ran the task using that item value?

1.3 Now we also have the ability to store our items in a file and loop through them that way as well. Using the same playbook above we will refactor it to work this way.

First let’s modify our **group_vars/nxos/nxos.yaml** file and add a **vlan_list** variable which consists of a list of values.

Your file should look something like this one below.

```
ansible_connection: network_cli

ansible_network_os: nxos

ansible_user: "{{ n9k_user }}"
```

```
ansible_ssh_pass: "{{ n9k_pw }}"

transport: nxapi

vlan_list: [80,90,100]
```

Now we need to modify our playbook task to pull from this variable file instead of our list we supplied in the task.

```
- name: Add Multiple Vlans

      nxos_vlan:

        vlan_id: "{{ item }}"

        state: present

      loop: "{{ vlan_list }}"
```

Now if we run our playbook again we will see it will create 80, 90 and skips 100 since it already exists and it is pulling all these values from a variable file.

```
ansible-playbook vlan_loop.yaml --ask-vault
```

We should have output similar to below where we can see it looping through and setting up the vlan 80 and 90 and skipping 100 since it already exists.

```
Vault password:



PLAY [Loop Example] ****************************************
*************************************************************
**
```

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

ok: [n9k-standalone-XX.localdomain]



TASK [Add Multiple Vlans] **********************************
************************************************************
**

changed: [n9k-standalone-XX.localdomain] => (item=80)

changed: [n9k-standalone-XX.localdomain] => (item=90)

ok: [n9k-standalone-XX.localdomain] => (item=100)



PLAY RECAP *************************************************
************************************************************
**

n9k-standalone-X.localdomain : ok=2    changed=1    unreacha
ble=0    failed=0
```

1.4 Certain modules/plugins will allow for us to specify a list of items directly which is more optimal than actually looping over the task.

Using the documentation from the nxos_vlan module - https://docs.ansible.com/ansible/2.5/modules/nxos_vlan_module.html modify the vlan_list variable in the group_vars to work with the aggregate option and modify the playbook to use this.

Here is the base task to help you along, you will still need to modify the group_vars/nxos/nxos.yaml file to work with this. Remember this is a list so we have to find an option in the nxos_vlan module that can take in a list or list of dicts.

```
- name: Add Multiple Vlans

      nxos_vlan:

        aggregate: "{{ vlan_list }}"

        state: present
```

Was this easier or harder than the loop option?

Which option do you think would take longer if you were adding say 500 vlans?

2. Nested Loops

1.1 There are also concepts of nested loops in Ansible which allow us to run multiple operations on the same resource.

In the following example we will be setting up 2 lists, one list of all the interfaces we want to update, and the second list will include all the variables we are going to update like the description, MTU, etc.

The use case for something like this is we can have different lists for different configurations we need. Say we had a list of all our interfaces that are uplinks and we configure another list for the values those interfaces should be configured with, etc.

First let's create a new playbook called nested_example.yaml and use the following task with it, we will be targeting our nxos group still in the playbook.

```
- name: Add Multiple Vlans

      nxos_vlan:

        vlan_id: "{{ item[0] }}"

        name: "{{ item[1] }}"
```

```
          state: present

        loop:

          - ['101', 'prod']

          - ['102', 'dev']
```

So in this task we are basically creating a nested loop and telling the nxos_vlan module to use item[0] which would be 101 and 102 as the vlan ID, while using item[1] which is prod and dev for the names.

Now if we were to run our playbook we should see the vlan's get created with their respective names.

```
ansible-playbook nested_example.yaml --ask-vault
```

This is what our output should look like

```
Vault password:



PLAY [Loop Example] ***************************************
*************************************************************
**



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

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

```
TASK [Add Multiple Vlans] ******************************
*****************************************************************
**

changed: [n9k-standalone-XX.localdomain] => (item=[u'101', u
'prod'])

changed: [n9k-standalone-XX.localdomain] => (item=[u'102', u
'dev'])



PLAY RECAP *************************************************
*****************************************************************
**

n9k-standalone-XX.localdomain : ok=2    changed=1    unreach
able=0    failed=0
```

Notice how it sets up both vlan's and creates their names respectively? Login to your nexus switch and run **show vlan** to see the new vlan's that were created.

3. Knowledge Check - Change Multiple Interfaces

Create a playbook to update the interface description and mtu on your eos device. We need the following information followed for the lab:

- You can choose what the description and MTU values are set to.
- make a new folder called **knowledge_check_3** and put your files in there.
- Configure interfaces Ethernet 1-3 using the loop method you learned.
- The documentation for the eos_interface module can be found at - https://docs.ansible.com/ansible/2.5/modules/eos_interface_module.html?highlight=eos_interface

4. Knowledge Check - Time Differences Aggregate vs Loop

So recently in Ansible more modules have a concept of something called **aggregate** which allows for basically providing a dict of the interface in our example with the settings we want changed.

In this lab we will be using our nxos device to test changing 10 interface descriptions using a loop versus the aggregate method and see which one takes longer.

- The interface description can be set to whatever you like
- Make a new folder called **knowledge_check_4** and put your files in there.
- You will configure the description for interfaces Ethernet1/1-10
- You can use time before your ansible-playbook command to time each playbook to see which takes longer.
- Create 2 playbooks, one using the loop way looping through all interfaces changing the description, the second playbook will use the aggregate option without a loop specifying the interfaces and description in the format shown in the documentation listed here, https://docs.ansible.com/ansible/latest/modules/nxos_interface_module.html?highlight=nxos_interface
- Remember on the 2nd play to set the interface descriptions to a different value so it actually runs the complete play for the full time calculation

Why do you think one is faster than the other?

Can you see what the difference is in a loop versus aggregate?

How might this be valuable if you were doing say 100+ interfaces?

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.