Lab 10 – Roles & Directory Structure

Introduction

Up to this point so far we have primarily only dealt with single playbooks and such. However Ansible gives us the option to split up our files and make them more reusable, one way we can accomplish this is using Roles.

In this lab we will be discussing Roles and the directory structure requirements and best practices when using Roles.

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. Role Directory Structure

1.1 Roles can be used to automatically load certain vars, tasks or handlers based on a specific file structure. By grouping content with roles it allow allows for easier sharing of roles and tasks with other users.

Here is an example project structure from the Ansible best practice section of the documentation modified for more network related concepts.

```
site.yml

switches.yml

routers.yml

roles/

   common/

     tasks/
```

```
      handlers/

      files/

      templates/

      vars/

      defaults/

      meta/

    switches/

      tasks/

      defaults/

      meta/
```

Roles will expect files to be in certain directory names, the roles must include at least one of these directories you see above under roles. We can exclude any we don't want by not using them, any folder structure we want to use we need to ensure a main.yml file resides in that said folder.

So if we wanted to use the tasks folder under roles we would need a main.yml file in that folder with our tasks.

1.2 Here is some more information on each option below the role and what they are used for:

- tasks - contains the main list of tasks to be executed
- handlers - containers handlers, which can be used by this role or outside the role.
- defaults - default variables used by the role
- vars - other variables for the role
- files - any files that might be deployed with the role

- templates - any templates like jinja2 to be used.
- meta - defines any meta information for the role. Can be used to link role dependencies for example.

1.3 So you might be asking yourself now, ok great how do we use these said roles though? It is actually pretty easy and makes the playbook a lot less complicated as well.

Take this example for instance we have a playbook example here that could setup a few things on a new switch for example:

```
---

- hosts: switches

  roles:

     - ntp

     - vlan

     - edge
```

So in this example we have 3 roles being specified ntp, vlan, and edge. These are actually bigger tasks that could run multiple things for us to setup a new switch. Maybe on the edge task we setup a bunch of descriptions and other things required of edge devices for example.

Notice how much smaller our playbook is now without all the tasks and instead we are just specifying roles now?

1.4 You can see below some of the information on how Ansible performs order of execution and the steps on roles, this is from the Ansible documentation located at https://docs.ansible.com/ansible/2.6/user_guide/playbooks_reuse_roles.html

- If roles/x/tasks/main.yml exists, tasks listed therein will be added to the play.
- If roles/x/handlers/main.yml exists, handlers listed therein will be added to the play.
- If roles/x/vars/main.yml exists, variables listed therein will be added to the play.
- If roles/x/defaults/main.yml exists, variables listed therein will be added to the play.
- If roles/x/meta/main.yml exists, any role dependencies listed therein will be added to the list of roles (1.3 and later).
- Any copy, script, template or include tasks (in the role) can reference files in roles/x/{files,templates,tasks}/ (dir depends on task) without having to path them relatively or absolutely.

Order of Execution:

- Any pre_tasks defined in the play.
- Any handlers triggered so far will be run.
- Each role listed in roles will execute in turn. Any role dependencies defined in the roles meta/main.yml will be run first, subject to tag filtering and conditionals.
- Any tasks defined in the play.
- Any handlers triggered so far will be run.

- Any post_tasks defined in the play.
- Any handlers triggered so far will be run.

2. Creating Our First Role

2.1 We will now begin creating our first role layout, we will be following the best practices information from Ansible for our directory structure.

Let’s ensure we are in the right folder and create our directory layout, we have a few different commands we are running here to set everything up.

```
cd ~/ansible_labs/lab10-roles/

mkdir -p roles/nxos/tasks

mkdir -p group_vars/nxos



cat > ansible.cfg <<EOF

[defaults]

host_key_checking = False

inventory = ./inventory

EOF



cat > group_vars/nxos/vlans.yml <<EOF

vlans:
 - { vlanid: 5, name: voip }

 - { vlanid: 6, name: data }

 - { vlanid: 7, name: dmz }

EOF


cat > group_vars/nxos/nxos.yml <<EOF

---

ansible_become: yes

ansible_network_os: nxos

ansible_become_method: enable

ansible_user: "admin"

ansible_become_pass: "{{ nxos_pw }}"

EOF


cat > roles/nxos/tasks/main.yml <<EOF

---

- name: Configure Nexus Vlans

  nxos_vlan:

    vlan_id: "{{ item.vlanid }}"

    name: "{{ item.name }}"
  loop: "{{ vlans }}"

EOF
```

Now we have our base directory layout which should look like this which will allow us to have 1 group var’s setup for our nxos devices.

```
roles/

   nxos/

   tasks/

   eos/

   tasks/
```

2.2 Now after running everything above we should have a nice directory structure. You can do ls -lah to see our current structure.

```
ls -lah

-rw-r--r--.  1 root root   61 Sep XX 08:36 ansible.cfg

drwxr-xr-x.  3 root root   18 Sep XX 09:13 group_vars

-rw-r--r--.  1 root root   37 Sep XX 09:09 inventory

drwxr-xr-x.  3 root root   18 Sep XX 09:11 roles

-rw-r--r--.  1 root root   73 Sep XX 09:11 site.yaml
```

2.3 Now from what you already know build a vault file in the group_vars/nxos folder and reference the username/password for your Nexus device and update it accordingly. Remember we are using **n9k_user** and **n9k_pw** as our variables.

What other file have we forgotten to make so far? Try to run the below command see if you can figure out what we are missing.

2.4 Now we should be able to run our site.yaml file and see our changes happen. Notice how small our site.yaml file is?

```
---

- name: Setup Vlan's for Switches

  hosts: nxos

  roles:

     - nxos
```

So we can now run this playbook to see our output:

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

And we should have some output similar to this

```
Vault password:



PLAY [Setup Vlan's for Switches] ***************************
************************************************************
*****
```

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

ok: [n9k-standalone-XX.localdomain]



TASK [nxos : Configure Nexus Vlans] ************************
************************************************************
*****

changed: [n9k-standalone-XX.localdomain] => (item={u'name':
u'voip', u'vlanid': 5})

changed: [n9k-standalone-XX.localdomain] => (item={u'name':
u'data', u'vlanid': 6})

changed: [n9k-standalone-XX.localdomain] => (item={u'name':
u'dmz', u'vlanid': 7})



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

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

3. Knowledge Check - Setup Role For Vlan Creation Multiple Devices

Using what you just learned above modify this site.yaml file to include another role for EOS that configures the same vlan's above in the variable file on your EOS device.

Ensure you are in the working directory **knowledge_check3** and check your **Ansible-Pod-Info.docx** for the information for your EOS device to use.

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.