4.8 ANSIBLE ROLES

Roles provide a framework for fully independent, or interdependent collections of variables, tasks, files, templates, and modules.

In Ansible, the role is the primary mechanism for breaking a playbook into multiple files. This simplifies writing complex playbooks, and it makes them easier to reuse. The breaking of playbook allows you to logically break the playbook into reusable components.

Each role is basically limited to a particular functionality or desired output, with all the necessary steps to provide that result either within that role itself or in other roles listed as dependencies.

Roles are not playbooks. Roles are small functionality which can be independently used but have to be used within playbooks. There is no way to directly execute a role. Roles have no explicit setting for which host the role will apply to.

Top-level playbooks are the bridge holding the hosts from your inventory file to roles that should be applied to those hosts.

Creating a New Role

The directory structure for roles is essential to create a new role.

Role Structure

Roles have a structured layout on the file system. The default structure can be changed but for now let us stick to defaults.

Each role is a directory tree in itself. The role name is the directory name within the /roles directory.

```
$ ansible-galaxy -h
```

Usage

```
ansible-galaxy [delete|import|info|init|install|list|login|remove|search|setup] [--help] [options] ...
```

Options

-h, --help − Show this help message and exit.

-v, --verbose − Verbose mode (-vvv for more, -vvvv to enable connection debugging)

--version − Show program's version number and exit.

Creating a Role Directory

The above command has created the role directories.

```
$ ansible-galaxy init vivekrole

ERROR! The API server (https://galaxy.ansible.com/api/) is not responding, please try again later.

$ ansible-galaxy init --force --offline vivekrole

- vivekrole was created successfully

$ tree vivekrole/

vivekrole/
 defaults
    main.yml
 files  handlers
    main.yml
 meta
    main.yml
 README.md  tasks
    main.yml
 templates  tests     inventory
    test.yml
 vars
     main.yml

8 directories, 8 files
```

Not all the directories will be used in the example and we will show the use of some of them in the example.

Utilizing Roles in Playbook

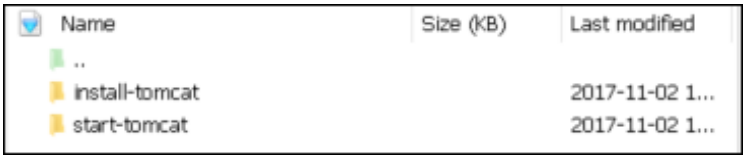
This is the code of the playbook we have written for demo purpose. This code is of the playbook vivek_orchestrate.yml. We have defined the hosts: tomcat-node and called the two roles install-tomcat and start-tomcat.

The problem statement is that we have a war which we need to deploy on a machine via Ansible.

```
---
- hosts: tomcat-node
roles:
  - {role: install-tomcat}
  - {role: start-tomcat}
```

Contents of our directory structure from where we are running the playbook.


```
$ ls
ansible.cfg  hosts  roles  vivek_orchestrate.retry vivek_orchestrate.yml
```



There is a tasks directory under each directory and it contains a main.yml. The main.yml contents of install-tomcat are −

```
---
```

```
#Install vivek artifacts
-
  block:
    - name: Install Tomcat artifacts
      action: >
        yum name = "demo-tomcat-1" state = present
      register: Output

  always:
    - debug:
      msg:
        - "Install Tomcat artifacts task ended with message: {{Output}}"
        - "Installed Tomcat artifacts - {{Output.changed}}"
```

The contents of main.yml of the start tomcat are −

```
#Start Tomcat
-
  block:
    - name: Start Tomcat
    command: <path of tomcat>/bin/startup.sh"
    register: output
    become: true

  always:
    - debug:
      msg:
        - "Start Tomcat task ended with message: {{output}}"
        - "Tomcat started - {{output.changed}}"
```

The advantage of breaking the playbook into roles is that anyone who wants to use the Install tomcat feature can call the Install Tomcat role.

Breaking a Playbook into a Role

If not for the roles, the content of the main.yml of the respective role can be copied in the playbook yml file. But to have modularity, roles were created.

Any logical entity which can be reused as a reusable function, that entity can be moved to role. The example for this is shown above

Ran the command to run the playbook.

-vvv option for verbose output verbose output

$ cd vivek-playbook/

This is the command to run the playbook

$ sudo ansible-playbook -i hosts vivek_orchestrate.yml vvv

4.9 ADHOC COMMANDS IN ANSIBLE

Ansible ad hoc commands are single-line commands executed directly from the command line on the Ansible control node to perform quick, one-time tasks on one or more managed nodes. Unlike playbooks, which are designed for complex, reusable automation, ad hoc commands are ideal for immediate, specific actions.

Ad hoc commands are commands which can be run individually to perform quick functions. These commands need not be performed later.

For example, you have to reboot all your company servers. For this, you will run the Adhoc commands from /usr/bin/ansible.

These ad-hoc commands are not used for configuration management and deployment, because these commands are of one-time usage.

- ansible-playbook is used for configuration management and deployment.

Structure of an Ad Hoc Command:

ansible [pattern] -m [module] -a "[module options]"

- ansible: The command-line tool used to execute ad hoc commands.
- [pattern]: Specifies the target hosts or groups from your inventory to run the command on (e.g., all, webservers, host1).
- -m [module]: Specifies the Ansible module to use (e.g., ping, shell, apt, service).
- -a "[module options]": Provides arguments or parameters to the chosen module.

Common Use Cases and Examples:

(i) Checking Connectivity (Ping).

ansible all -m ping

(ii) Executing Shell Commands.

ansible webservers -a "uptime"

ansible databases -m shell -a "df -h /var/lib/mysql"

(iii) Managing Services.

ansible appservers -m service -a "name=nginx state=started"

ansible all -m service -a "name=httpd state=restarted"

(iv) installing packages.

ansible webservers -m apt -a "name=apache2 state=present"

(v) copying files.

ansible backupservers -m copy -a "src=/tmp/config.txt dest=/etc/config.txt owner=root group=root mode=0644"

(vi) rebooting servers.

ansible all -m reboot

Key Characteristics:

- **Quick and Simple:**

Designed for rapid execution of individual tasks.

- **Non-reusable:**

Not intended for complex, repeatable automation as they lack the structure and features of playbooks.

- **Idempotent (where applicable):**

Many Ansible modules used in ad hoc commands are idempotent, meaning they check the current state before making changes, ensuring the desired state is achieved without unnecessary operations.

- **Demonstrates Ansible Power:**

Provides a straightforward way to understand Ansible's capabilities and module functionality before delving into playbooks.