ANSIBLE - INTRODUCTION

Inventory, Config Files, Playbooks, Rolls, and Collections

ANSIBLE INTRODUCTION & OVERVIEW

- **Ansible** – An automation language leveraging modules to be used in one or more tasks on managed systems. Most Ansible automation leverages and Ansible playbook which is a YAML formatted file containing Ansible directives.

- **Ansible modules** – Components used by Ansible tasks and playbooks which are generally implemented and developed in Python. Ansible modules work with certain system utilities and are optimized to be leveraged as a declarative automation language and provide idempotency.

- **Ansible ad-hoc commands** – A way of executing a single Ansible task quickly that relies on a single Ansible module to perform the tests/changes of the task.

ANSIBLE COMPONENTS & COMMANDS

- **ansible.cfg** – Configuration file for running the **ansible** and **ansible-playbook** commands.
- **Inventory** – Inventory file used by the **ansible** and **ansible-playbook** commands identifying managed hosts/nodes.
- **ansible** – Command used to perform/execute Ansible ad-hoc commands against a managed node.
- **ansible-playbook** – Command used to execute and run Ansible playbooks
- **ansible-galaxy** – Command to create or utilize Ansible roles. Many of these roles are published on http://galaxy.ansible.com
- **Playbook** – Collection of Ansible tasks organized into one or more Ansible plays.
- **Task** – Collection/list of Ansible modules arranged into instructions. Each task utilizes an Ansible module to perform a given action.
- **Ansible Module** – Specific module (small program generally implemented in Python) which perform the commands and executes the program to get the desired state of a given task.

ANSIBLE (AAP2) COMPONENTS & COMMANDS

- **Ansible-navigator.yml** – Configuration file for running the **ansible-navigator** command. The biggest things specified here are the EEI (Execution Environment Image), the Pull Policy, and the mode.
  Ansible Module – Specific module (small program generally implemented in Python) which perform the commands and executes the program to get the desired state of a given task.
- **Ansible Collection** – Collections of Ansible content. Generally Ansible collections include Ansible modules, Ansible roles, and Ansible plugins. However, it is possible to include Ansible playbooks as part of collections.
- **ansible-navigator** – Command line utility that will execute Ansible playbooks in an execution environment using the defined EEI. The **ansible-navigator** command replaces all other ansible commands.
- **Execution Environment Image (EEI) and Execution Environment** – The EEI is a container image is a simple container image that contains OS and system packages, Python packages, and most importantly the Ansible runtime libraries and components along with a set of Ansible collections and modules. The Execution Environment (EE) is the container launched by the Ansible navigator command that runs the **ansible playbook** and essentially functions as the **control node.** It is important to remember with EEs, that **localhost** refers to the container running the playbook and not the system running the **ansible-navigator** command.

INVENTORY FILE

- The **inventory** file can contain both Ansible managed hosts/nodes as well as inventory variables to be used for the managed nodes.
  - Inventory location is generally specified by the **ansible.cfg** file
    - **./inventory** – Common practice to leverage inventory files with playbooks and the **ansible.cfg** file in the current working directory
    - **/etc/ansible/hosts** – Default inventory file deployed with the Ansible package

Important	Inventories must contain exact names and each unique hostname/IP/FQDN included in inventory appears as a separate managed node in Ansible.

ANSIBLE.CFG FILE

- This file defines the configuration directives which apply directly to how the **ansible and ansible-playbook** command interact with the Ansible application and which configuration items are applied to a given Ansible session.
  - **./ansible.cfg** – When located in the current working directory (CWD) this file is the highest precedence.
  - **~/.ansible.cfg** – When located in the user's home directory, this file will have precedence if an *ansible.cfg* doesn't exist in the CWD.
  - **/etc/ansible/ansible.cfg** – This is the default configuration file and has the lowest precedence. This file is used when no other *ansible.cfg* file exists.

Important	It is also possible to define the **ansible.cfg** file with the environment variable **ANSIBLE_CONFIG**. If this variable is used, it will override all other configuration files.

ANSIBLE-NAVIGATOR.YML FILE

- This file defines the configuration directives which apply directly to how the **ansible and ansible-playbook** command interact with the Ansible application and which configuration items are applied to a given Ansible session.
  - **./ansible-navigator.yml** – When located in the current working directory (CWD) this file is the highest precedence.
  - **~/.ansible-navigator.yml** – When located in the user's home directory, this file will have precedence if an *ansible.cfg* doesn't exist in the CWD.

Important	It is also possible to define the **ansible-navigator.yml** file with the environment variable **ANSIBLE_NAVIGATOR_CONFIG**. If this variable is used, it will override all other configuration files.

PLAYBOOK STRUCTURE (SIMPLE)

```
---
- name: Ansiblize Managed Hosts
  hosts: localhost

  tasks:
    - name: Create Ansible User
      debug:
          msg: This will use the USER  module
```

PLAYBOOK STRUCTURE (COMPLEX)

```
---
- name: Ansiblize Managed Hosts
  hosts: localhost
  vars_files:
    - variables.yml
  roles:
    - namespace.role_name

  pre-tasks:
    - name:  Pre-Task to login to the API for Ansible Tasks
      debug:
        msg: This will use a module to login to the API

  tasks:
    - name: Create Ansible User
      debug:
        msg: This will use the USER  module

  post-tasks:
    - name:  Pre-Task to login to the API for Ansible Tasks
      debug:
        msg: This will use a module to login to the API

  handlers:
    - name: Restart SSHD
      debug:
        msg: This sill use the SERVICE or SYSTEMD
```

CREATING AN ANSIBLE ROLE

- Use the **ansible-galaxy init <RoleName>** command to create a Role
- Empty directories or unused directories can be deleted to clean up the Role
- Populate the various Role structures
  - Must have the following components (at minimum):
    - README.md
    - meta/main.yaml
    - tasks/main.yaml

```
[student@workstation ~]$ ansible-galaxy init My_Role
- My_Role was created successfully
```

ANSIBLE ROLE STRUCTURE

```
[student@workstation ROLES]$ tree My_Role/
My_Role/
├── 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
```

ANSIBLE ROLE STRUCTURE

Subdirectory	Function of Directory
defaults	The **main.yml** file contains default variable values that are used by the role. These have the lowest precedence and priority.
files	Contains files that are used by the tasks in the role.
handlers	The **main.yml** file contains handlers that are executed by the role.
meta	The **main.yml** file contains information about the role. At a minimum you should modify the ***author***, ***license***, ***platforms***, and ***dependencies***.
tasks	The **main.yml** file contains the tasks being used by the Role.
templates	Contains Jinja2 templates used by tasks in the role.
tests	Contains an inventory and **test.yml** playbook that can be used to test the role.
vars	The **main.yml** file contains role variables and values. These are high precedence and not intended to be changed when used in a playbook.

ANSIBLE CONTENT COLLECTIONS

- Use the **ansible-galaxy collection install <collection_name>** command to install a collection.
  - You can install multiple collections and use the **–r requirements.yml** file to provide a list of content collections. You can also use the **–p <directory>** to specify where to install content collections. Typically, this would be in a folder in the current working directory at the project level called **collections.**
- Sdf

```
[student@workstation ~]$ ansible-galaxy collection install
```

PLAYBOOK TROUBLESHOOTING

```yaml
---
- name: Install a samba server
  hosts: samba_servers
  user: devops
  become: true
  vars:
    install_state: installed
    random_var: "This is colon: test"

  tasks:
  - name: install samba
    yum:
      name: samba
      state: "{{ install_state }}"

  - name: install firewalld
    yum:
```

.vimrc	autocmd FileType yaml setlocal ai ts=2 sw=2 et sts=2 colorcolumn=3,5,7,9,11