ANSIBLE ROLES

Ansible Roles

An Ansible role is a reusable, self-contained unit of automation that is used to organize and manage tasks, variables, files, templates, and handlers in a structured way.

Roles help to encapsulate and modularize the logic and configuration needed to manage a particular system or application component.

This modular approach promotes reusability, maintainability, and consistency across different playbooks and environments.

Key Components of an Ansible Role

Tasks

The main list of actions that the role performs.

Handlers

Tasks that are triggered by changes in other tasks, typically used for actions like restarting services.

Files

Static files that need to be transferred to managed hosts.

Templates

Jinja2 templates that can be rendered and transferred to managed hosts.

Vars

Variables that are used within the role.

Defaults

Default variables for the role, which can be overridden.

Meta

Metadata about the role, including dependencies on other roles.

Library

Custom modules or plugins used within the role.

Module_defaults

Default module parameters for the role.

Lookup_plugins

Custom lookup plugins for the role.

Directory Structure of an Ansible Role

An Ansible role follows a specific directory structure:

```
<role_name>/
  ├── defaults/
  │   └── main.yml
  ├── files/
  ├── handlers/
  │   └── main.yml
  ├── meta/
  │   └── main.yml
  ├── tasks/
  │   └── main.yml
  ├── templates/
  ├── vars/
      └── main.yml
```

Why Use Ansible Roles?

Modularity

Roles allow you to break down complex playbooks into smaller, reusable components. Each role handles a specific part of the configuration or setup.

Reusability

Once created, roles can be reused across different playbooks and projects. This saves time and effort in writing redundant code.

Maintainability

By organizing related tasks into roles, it becomes easier to manage and maintain the code. Changes can be made in one place and applied consistently wherever the role is used.

Readability

Roles make playbooks cleaner and easier to read by abstracting away the details into logically named roles.

Collaboration

Roles facilitate collaboration among team members by allowing them to work on different parts of the infrastructure independently.

Consistency

Using roles ensures that the same setup and configuration procedures are applied uniformly across multiple environments, reducing the risk of configuration drift.

Creating a Basic Role

Let's create a simple role named apache2 that installs and starts apache2 and deploy simple app.

Create the Role Structure: You can manually create the directories and files listed above, or use the ansible-galaxy command to initialize the role structure:

ansible-galaxy role init role-name

- I Created a role named as webserver .The role created below structure.

```
role_name/
    ├── defaults/        # Default variables for the role
    │   └── main.yml
    ├── vars/            # Other variables for the role
    │   └── main.yml
    ├── tasks/           # Main list of tasks to be executed by the role
    │   └── main.yml
    ├── files/           # Contains files used by the role
    ├── templates/       # Contains templates used by the role
    ├── handlers/        # Handlers, which may be used within or outside this ro
    │   └── main.yml
    └── meta/            # Defines some meta data for this role.
        └── main.yml
```

- Previously I written a playbook To install apache2 webserver and Deploy a simple app on aws.
- Now I convert that playbook into role .

CREATE ROLE.

- Execute below command to create role.

ansible-galaxy role init role-name

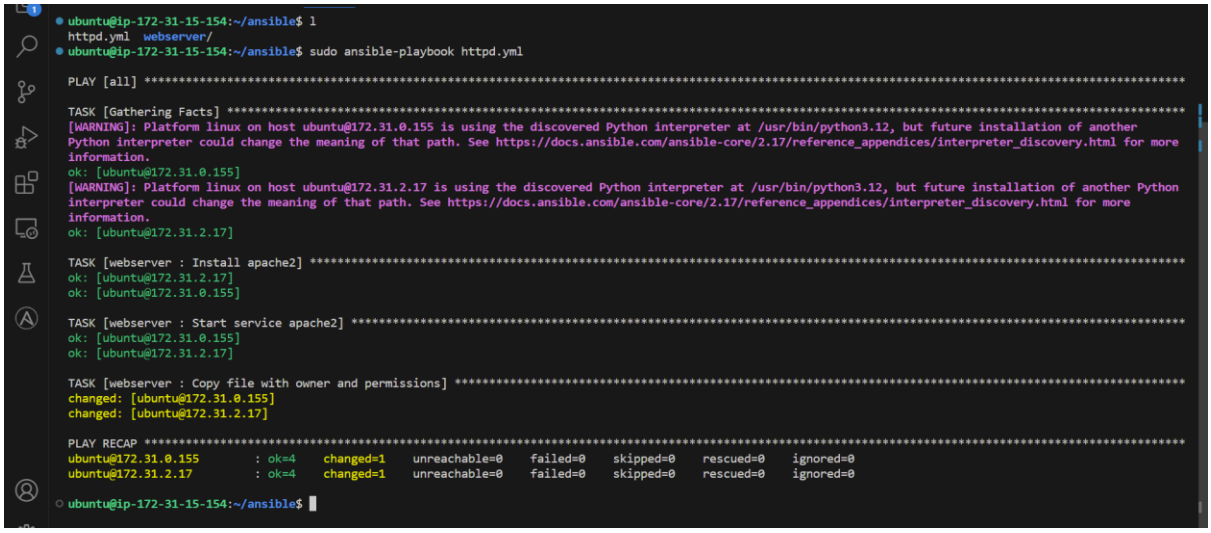
- Place Tasks in tasks/main.yml file.

- Place a File (index.html) in files directory.

- Need to Mention location of index.html in playbook/

- Mention the role in httpd.yml file.

- Execute the playbook



- Access the App using public_ip of managed nodes with port 80.