**Introduction to Ansible Roles and Folder Organization**

Ansible is a powerful automation tool used to manage configurations, deployments, and orchestration. One of the best practices in Ansible is organizing your playbooks using **roles**. Roles help structure the Ansible project, making it modular, reusable, and easier to maintain.

---

**1. What are Ansible Roles?**

Ansible roles provide a way to organize playbooks by breaking them down into smaller, reusable components. Instead of writing a large and complex playbook, roles allow you to separate tasks, variables, handlers, and templates into different folders.

**Example**

Imagine you want to install and configure Apache. Instead of putting everything in one file, you can create a role called apache that organizes related tasks and files.

---

**2. Folder Structure for Ansible Roles**

When using roles, a typical folder structure looks like this:

project-directory/

|-- playbook.yml

|-- roles/

| ├— apache/

| | ├— defaults/

| | | ├— main.yml

| | ├— vars/

| | | ├— main.yml

| | ├— tasks/

| | | ├— main.yml

| | | ├— install.yml

| | ├— handlers/

| | | ├— main.yml

| | ├— templates/

| | | ├— apache.conf.j2

| | ├— files/

| | | ├— index.html

**Explanation of Each Folder**

| Folder Name | Description |
|---|---|
| defaults/ | Contains default variables (lowest priority). |
| vars/ | Stores variables with higher priority than defaults. |
| tasks/ | Defines the tasks to be executed (main logic). |
| handlers/ | Contains handlers, which are triggered by tasks. |
| templates/ | Stores Jinja2 templates for configuration files. |
| files/ | Holds static files that need to be copied. |

**3. Creating an Ansible Role**

To create a role, use the following command:

ansible-galaxy init apache

This will automatically generate the required folder structure for your role.

**Example: Installing Apache**

1. **Define the task in tasks/main.yml:**

```
- name: Install Apache
  apt:
    name: apache2
    state: present
  notify: Restart Apache
```

2. **Define the handler in handlers/main.yml:**

```
- name: Restart Apache
  service:
    name: apache2
    state: restarted
```

3. **Create a template file in templates/apache.conf.j2:**

```
<VirtualHost *:80>
    ServerName {{ server_name }}
    DocumentRoot /var/www/html
</VirtualHost>
```

4. **Define variables in vars/main.yml:**

server_name: example.com

5. **Include the role in a playbook (playbook.yml):**

- hosts: webservers

  roles:

    - apache

---

**4. Running the Playbook**

Once everything is set up, execute the playbook using:

ansible-playbook playbook.yml

This will install and configure Apache based on the role structure.

---

**5. Benefits of Using Roles**

- **Modularity:** Reuse roles across multiple projects.
- **Readability:** Easier to understand and manage.
- **Scalability:** Can be shared and maintained easily.
- **Best Practices:** Helps in organizing the code efficiently.

---

**Conclusion**

Using roles in Ansible is a best practice that improves the maintainability and reusability of automation scripts. By following a structured folder format, you can easily manage configurations, handle tasks efficiently, and keep your playbooks clean and organized.