Deploying Apache Server on Docker Containers Using Ansible

1. Prerequisites

Before running the playbook, the following software and tools need to be installed on your system:

- **Python 3.x**: Ansible requires Python to be installed on the system. Ansible typically uses Python 3 for execution.
- **Ansible**: The tool used to automate the configuration, deployment, and orchestration of systems.
- **Docker**: An essential component, Docker is required to deploy and run the Apache server in containers.
- **Virtualenv (Optional)**: Recommended to create an isolated Python environment for managing dependencies.

2. Directory Structure of the Ansible Playbook

The Ansible project is structured as follows:

```
ansible-apache-docker/
├── inventory.ini           # The inventory file that lists target
hosts
├── playbook.yml            # The Ansible playbook to deploy
Apache in Docker
└── roles/
    ├── docker_install/     # Role to install Docker
    └── apache_deploy/      # Role to deploy Apache server in
```

3. Detailed Explanation of the Code

3.1 Playbook: `playbook.yml`

```
---
- name: Deploy Apache Server on Docker Containers  # Name of the
playbook
  hosts: webservers  # The group of hosts to run this playbook on
(defined in inventory)
  become: yes  # Ensure that tasks requiring elevated privileges
are executed with 'sudo'
```

```
  gather_facts: yes  # Gather system facts for the target hosts
(e.g., OS, architecture, etc.)

  # Roles are reusable sets of tasks that are organized in a
directory structure.
  roles:
    - docker_install  # Role to install Docker on the target hosts
    - apache_deploy  # Role to deploy Apache server in a Docker
container
```

- **name**: Provides a description of the playbook.
- **hosts**: Specifies the target hosts or groups to which the playbook will apply. Here, `webservers` refers to the target servers.
- **become**: Ensures tasks that require root privileges (e.g., installing software) are executed with `sudo`.
- **gather_facts**: Collects facts about the system, such as the OS, architecture, Python version, etc., to use in conditional statements.
- **roles**: The playbook executes two roles—`docker_install` (for installing Docker) and `apache_deploy` (for deploying the Apache server).

3.2 Role: `docker_install`

```
---
- name: Install Docker on Linux
  become: yes
  apt:
    name: docker.io
    state: present
    update_cache: yes
  when: ansible_os_family == 'Debian'

- name: Skip Docker installation for macOS
  debug:
    msg: "Docker is already installed on macOS. Skipping
installation."
  when: ansible_os_family == 'Darwin'  # Skip on macOS (Darwin)
```

- **Install Docker on Linux**: This task installs Docker on Debian-based systems using the `apt` module.

- **Skip Docker installation for macOS**: Skips the Docker installation on macOS (Darwin) because Docker is assumed to already be installed.

3.3 Role: `apache_deploy`

```
---
- name: Pull Apache Docker image
  docker_image:
    name: httpd
    source: pull

- name: Run Apache in Docker container
  docker_container:
    name: apache
    image: httpd
    state: started
    exposed_ports:
      - "80"
    published_ports:
      - "8080:80"
```

- **Pull Apache Docker image**: This task pulls the official Apache HTTP server Docker image (`httpd`) from Docker Hub.
- **Run Apache in Docker container**: This task runs the Apache HTTP server inside a Docker container, exposes port 80, and maps it to port 8080 on the host.

4. Inventory File: `inventory.ini`

```
[webservers]
server1 ansible_host=192.168.x.1
server2 ansible_host=192.168.x.2
server3 ansible_host=192.168.x.3
```

- The `inventory.ini` file contains the target servers. Replace the IP addresses (`192.168.x.1`, `192.168.x.2`, `192.168.x.3`) with the actual IPs of the servers you want to target.

5. Testing the Playbook in a Local Environment

5.1 Steps to Test the Playbook

Install Required Packages: Make sure Python, Ansible, and Docker are installed on your local machine.

Create Virtual Environment (Optional but recommended): To avoid conflicts, you can create a virtual environment to isolate Python dependencies.

```
python3 -m venv ansible-env
source ansible-env/bin/activate
```

Install Required Python Libraries: You may need to install some additional Python libraries, like **`requests`**, to ensure that Ansible modules work correctly.
`pip install requests`

Use Inventory File: You can test locally by using the `inventory_local.ini` file that points to `localhost`.

```
[webservers]
localhost ansible_connection=local
```

Run the Playbook: Execute the playbook using the following command:

```
ansible-playbook -i inventory_local.ini playbook.yml
```

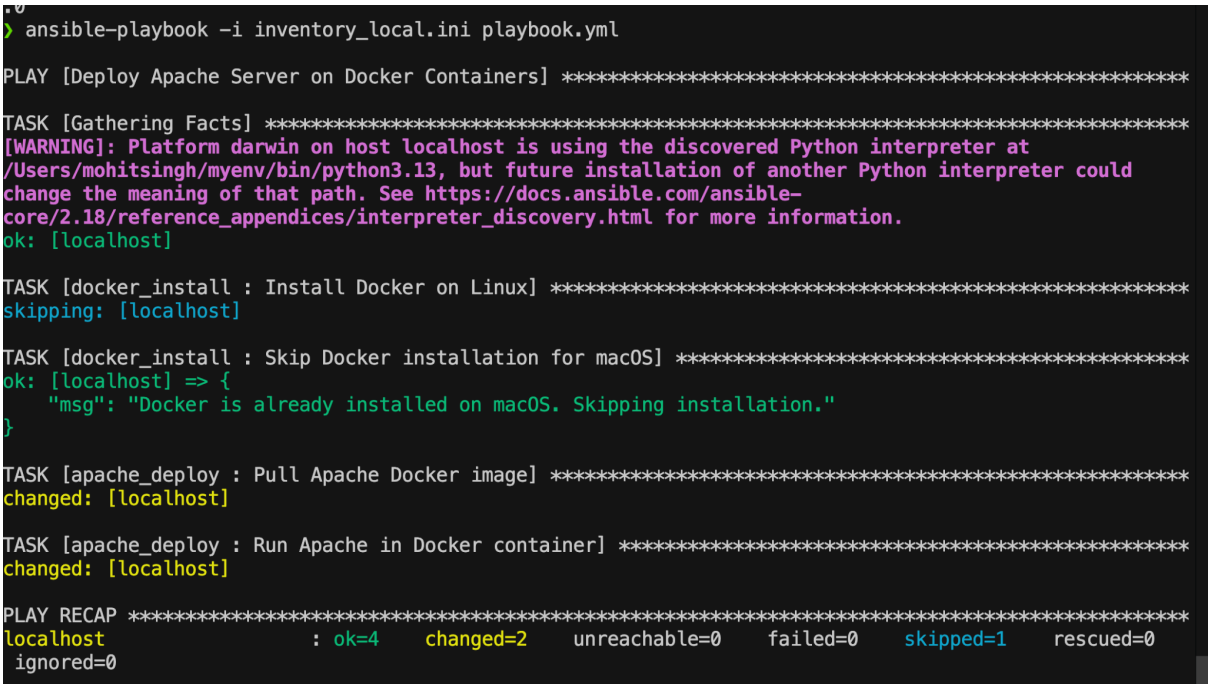
5.2 Expected Output

If the playbook runs successfully, you should see output like this:

```
PLAY [Deploy Apache Server on Docker Containers]
*********************************************

TASK [docker_install : Skip Docker installation for macOS]
*******************************
ok: [localhost] => {
    "msg": "Docker is already installed on macOS. Skipping
installation."
}
```

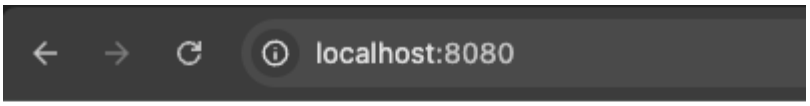
```
TASK [apache_deploy : Pull Apache Docker image]
*********************************************
changed: [localhost]

TASK [apache_deploy : Run Apache in Docker container]
***********************************
changed: [localhost]

PLAY RECAP
*****************************************************************
************
localhost                  : ok=4    changed=2    unreachable=0
failed=0    skipped=1    rescued=0    ignored=0
```

- **msg**: The task informs you that Docker is already installed (for macOS).
- **changed**: Indicates that changes were made (e.g., pulling the Docker image and starting the container).
- **failed**: None of the tasks should fail if everything is set up correctly.



5.3 Verify Apache Deployment

1. Open a browser and go to **`http://localhost:8080`**.
2. If the Apache server is running correctly, you should see the default Apache HTTP server page.



It works!