Docker Practical 8

1. Prerequisites (Docker Desktop)

Install these on your host machine:

```
pip install ansible
pip install docker docker-compose
```

Verify:

```
ansible --version
docker --version
docker-compose --version
```

---

2. Create a project folder

```
mkdir ansible-deploy-demo && cd ansible-deploy-demo
```

---

3. Create SSH key for Ansible

```
ssh-keygen -t rsa -b 4096 -f ./ansible_demo_key -N ""
```

This gives you:

- ansible_demo_key (private key)
- ansible_demo_key.pub (public key)

---

4. Define Dockerfile for nodes

Make a Dockerfile.node:

```
FROM ubuntu:22.04


RUN apt-get update && \
    apt-get install -y openssh-server python3 python3-pip python3-venv && \
    mkdir /var/run/sshd
```

```
# root password
RUN echo 'root:root' | chpasswd

# enable root login
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
RUN sed -i 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' /etc/pam.d/sshd

EXPOSE 22
CMD ["/usr/sbin/sshd", "-D"]
```

👉 Using **Ubuntu 22.04** ensures Python 3.10+, fixing the earlier __future__.annotations error.

---

5. Docker Compose to spin up 2 servers

Create docker-compose.yml:

```
version: "3"
services:
  node1:
    build:
      context: .
      dockerfile: Dockerfile.node
    container_name: node1
    ports:
      - "2222:22"

  node2:
    build:
      context: .
      dockerfile: Dockerfile.node
    container_name: node2
    ports:
      - "2223:22"
```

Start them:

docker-compose up -d --build

---

6. Add your SSH key into the containers

```
docker exec -i node1 bash -lc 'mkdir -p /root/.ssh && cat >> /root/.ssh/authorized_keys' < ./ansible_demo_key.pub

docker exec -i node2 bash -lc 'mkdir -p /root/.ssh && cat >> /root/.ssh/authorized_keys' < ./ansible_demo_key.pub


docker exec node1 chmod 700 /root/.ssh && docker exec node1 chmod 600 /root/.ssh/authorized_keys

docker exec node2 chmod 700 /root/.ssh && docker exec node2 chmod 600 /root/.ssh/authorized_keys
```

Now you can test SSH:

```
ssh -i ./ansible_demo_key -p 2222 root@127.0.0.1 "python3 --version"

ssh -i ./ansible_demo_key -p 2223 root@127.0.0.1 "python3 --version"
```

---

7. Inventory file for Ansible

Create inventory.ini:

```
[node_servers]

node1 ansible_host=127.0.0.1 ansible_port=2222 ansible_user=root ansible_ssh_private_key_file=./ansible_demo_key ansible_python_interpreter=/usr/bin/python3

node2 ansible_host=127.0.0.1 ansible_port=2223 ansible_user=root ansible_ssh_private_key_file=./ansible_demo_key ansible_python_interpreter=/usr/bin/python3
```

---

8. Write a sample Python Flask app

app.py:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello from Ansible-deployed Flask app!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

requirements.txt:

```
flask
```

---

9. Ansible Playbook

Create deploy-python-app.yml:

```
- name: Deploy Python Flask app
  hosts: node_servers
  become: yes

  tasks:
    - name: Ensure /opt/myapp exists
      file:
        path: /opt/myapp
        state: directory

    - name: Copy application files
      copy:
```

```
    src: ./app.py
    dest: /opt/myapp/app.py

- name: Copy requirements.txt
  copy:
    src: ./requirements.txt
    dest: /opt/myapp/requirements.txt

- name: Create virtualenv
  command: python3 -m venv /opt/myapp/venv
  args:
    creates: /opt/myapp/venv

- name: Install dependencies
  command: /opt/myapp/venv/bin/pip install -r /opt/myapp/requirements.txt

- name: Run Flask app (systemd service)
  copy:
    dest: /etc/systemd/system/myapp.service
    content: |
      [Unit]
      Description=Flask App
      After=network.target

      [Service]
      ExecStart=/opt/myapp/venv/bin/python /opt/myapp/app.py
      WorkingDirectory=/opt/myapp
      Restart=always
```

```
          User=root


          [Install]
          WantedBy=multi-user.target

    - name: Reload systemd
      command: systemctl daemon-reload

    - name: Enable and start service
      systemd:
        name: myapp
        enabled: yes
        state: started
```

---

10. Run the playbook

```
ansible-playbook -i inventory.ini deploy-python-app.yml
```

---

11. Test the app

Now test both nodes:

```
curl http://localhost:5000
```

👉 You should see:

```
Hello from Ansible-deployed Flask app!
```

TERMINAL:

```
ssh-keygen -t rsa -b 4096 -f ./ansible_demo_key -N ""

docker-compose up -d –build

docker exec -i node1 bash -lc 'mkdir -p /root/.ssh && cat >>
/root/.ssh/authorized_keys' < ./ansible_demo_key.pub
```

```
docker exec -i node2 bash -lc 'mkdir -p /root/.ssh && cat >> /root/.ssh/authorized_keys' < ./ansible_demo_key.pub

docker exec node1 chmod 700 /root/.ssh && docker exec node1 chmod 600 /root/.ssh/authorized_keys

docker exec node2 chmod 700 /root/.ssh && docker exec node2 chmod 600 /root/.ssh/authorized_keys

ssh-keygen -R "[127.0.0.1]:2222"

ssh -i ./ansible_demo_key -p 2222 root@127.0.0.1 "python3 --version"

ssh-keygen -R "[127.0.0.1]:2223"

ssh -i ./ansible_demo_key -p 2223 root@127.0.0.1 "python3 --version"

ansible-playbook -i inventory.ini deploy-python-app.yml
```

✅ Key Fixes from Earlier Errors

1. **Python version mismatch** → Fixed by using ubuntu:22.04 with Python ≥3.10.
2. **dpkg lock issues** → Gone, since we install Python at build-time in Dockerfile.
3. **wait_for deserialization bug** → Removed; we directly rely on Python being preinstalled.
4. **SSH setup** → Keys copied into containers before running Ansible.