Name: De Omampo, Julius Mark A.	**Date Performed:** 11/13/24
Course/Section: CPE212 – CPE31S2	**Date Submitted:** 11/13/24
Instructor: Engr. Robin Valenzuela	**Semester and SY:**

Activity 11: Containerization

1. Objectives

Create a Dockerfile and form a workflow using Ansible as Infrastructure as Code (IaC) to enable Continuous Delivery process

2. Discussion

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Source: https://docs.docker.com/get-started/overview/

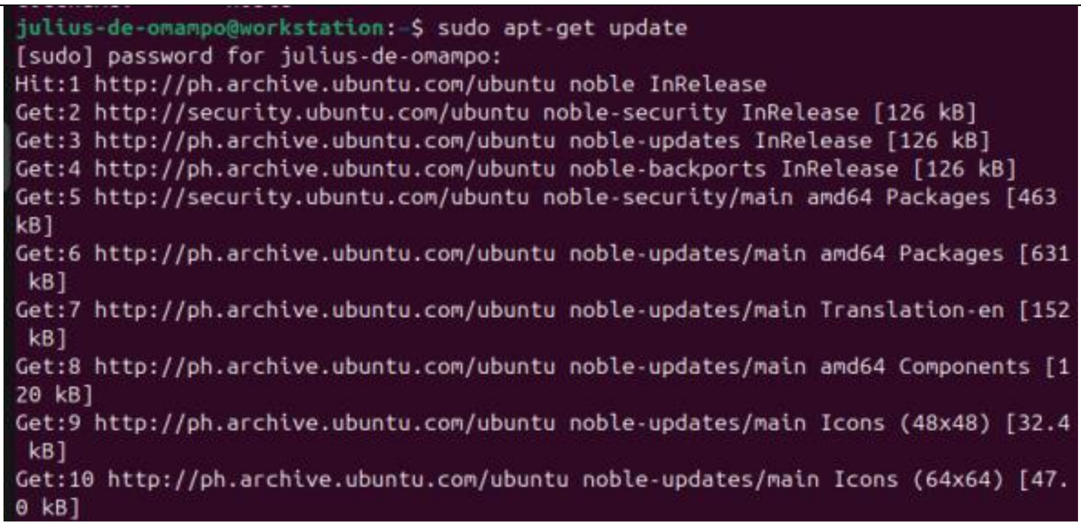
You may also check the difference between containers and virtual machines. Click the link given below.

Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

3. Tasks

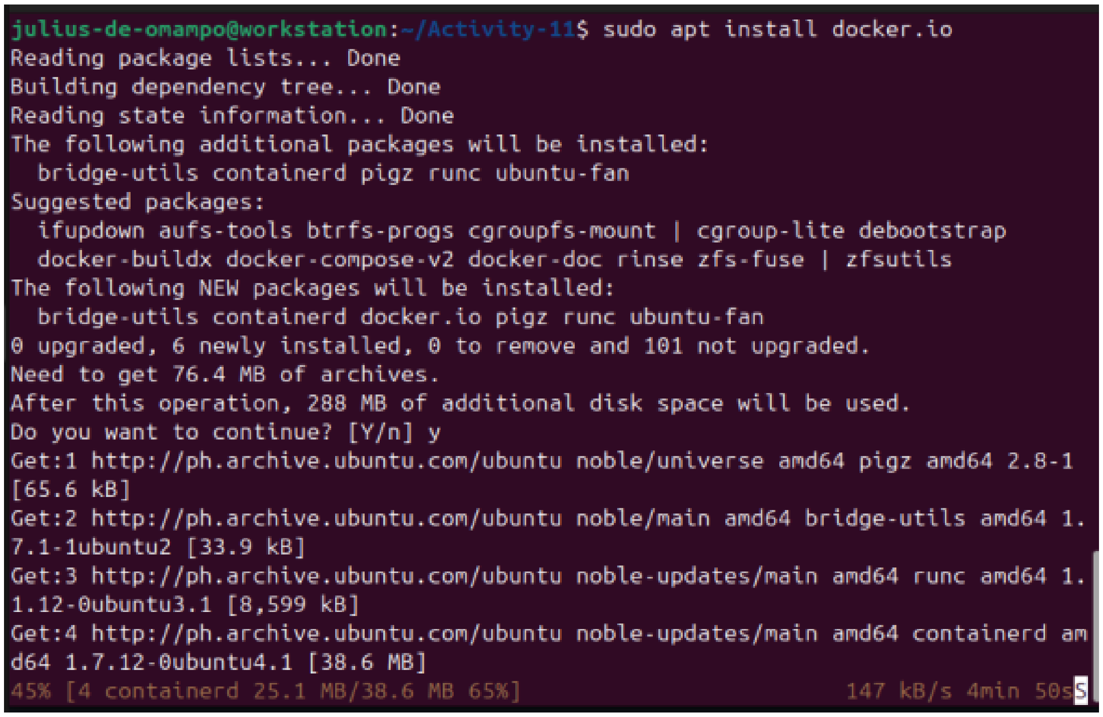
1. Create a new repository for this activity.
2. Install Docker and enable the docker socket.
3. Add to Docker group to your current user.
4. Create a Dockerfile to install web and DB server.
5. Install and build the Dockerfile using Ansible.
6. Add, commit and push it to your repository.

4. **Output** (screenshots and explanations)



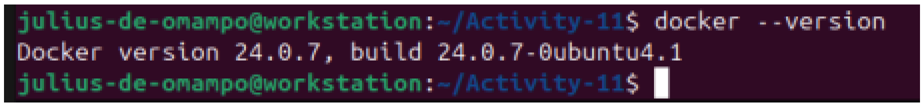
```
julius-de-omampo@workstation:~$ sudo apt-get update
[sudo] password for julius-de-omampo:
Hit:1 http://ph.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [463
kB]
Get:6 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [631
 kB]
Get:7 http://ph.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [152
 kB]
Get:8 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [1
20 kB]
Get:9 http://ph.archive.ubuntu.com/ubuntu noble-updates/main Icons (48x48) [32.4
 kB]
Get:10 http://ph.archive.ubuntu.com/ubuntu noble-updates/main Icons (64x64) [47.
0 kB]
```

check for latest updates



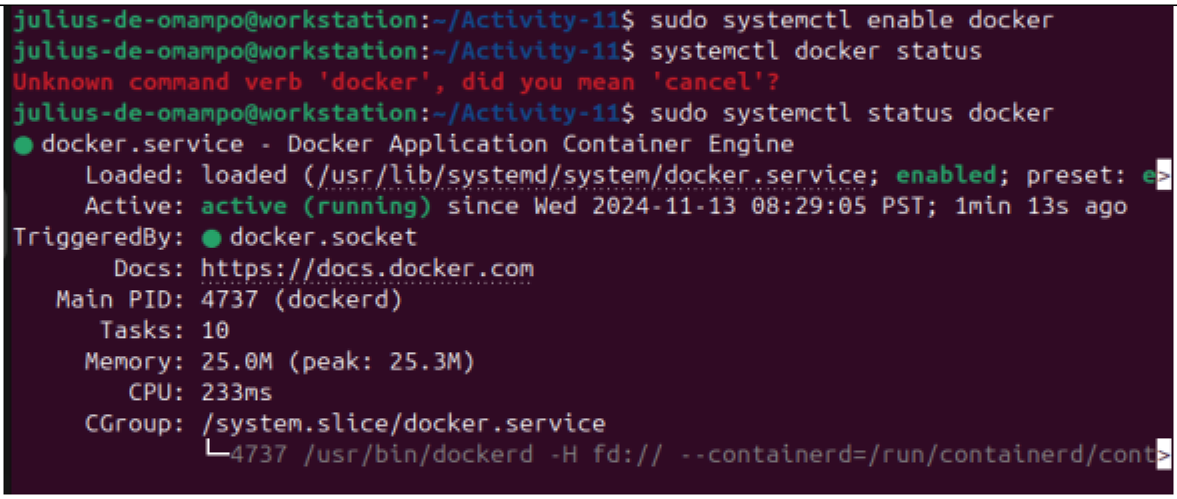
```
julius-de-omampo@workstation:~/Activity-11$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
  docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 101 not upgraded.
Need to get 76.4 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1
[65.6 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.
7.1-1ubuntu2 [33.9 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.
1.12-0ubuntu3.1 [8,599 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd am
d64 1.7.12-0ubuntu4.1 [38.6 MB]
45% [4 containerd 25.1 MB/38.6 MB 65%]                    147 kB/s 4min 50s
```

install docker.io



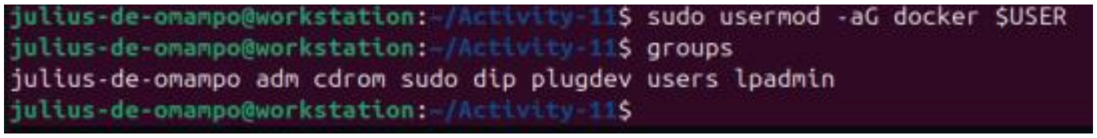
```
julius-de-omampo@workstation:~/Activity-11$ docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
julius-de-omampo@workstation:~/Activity-11$
```

Docker complete installation



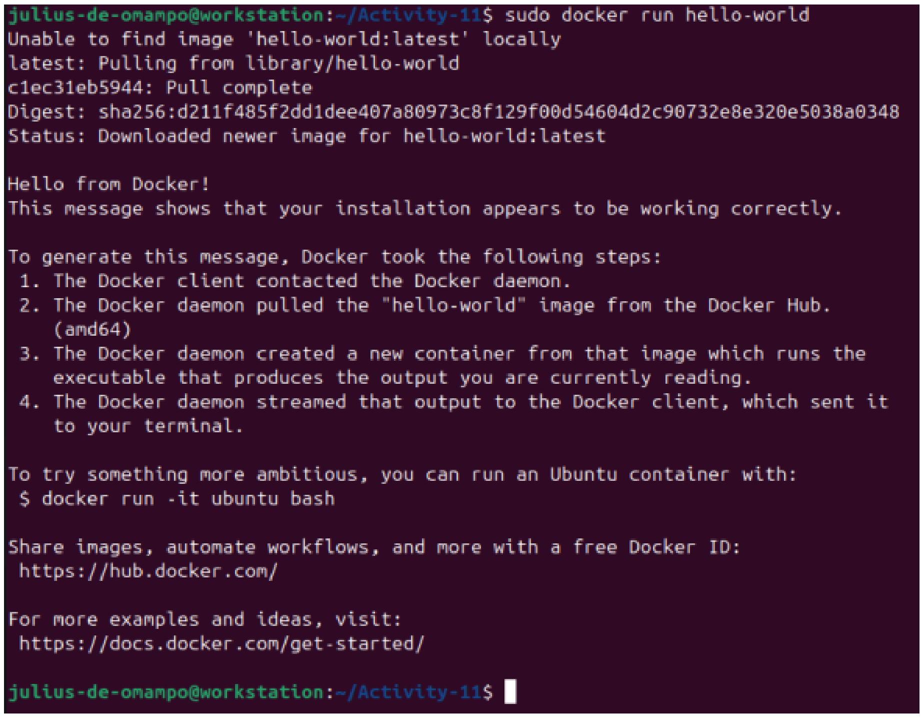
```
julius-de-omampo@workstation:~/Activity-11$ sudo systemctl enable docker
julius-de-omampo@workstation:~/Activity-11$ systemctl docker status
Unknown command verb 'docker', did you mean 'cancel'?
julius-de-omampo@workstation:~/Activity-11$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: e>
     Active: active (running) since Wed 2024-11-13 08:29:05 PST; 1min 13s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 4737 (dockerd)
      Tasks: 10
     Memory: 25.0M (peak: 25.3M)
        CPU: 233ms
     CGroup: /system.slice/docker.service
             └─4737 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont>
```

Docker systemctl enabling and status



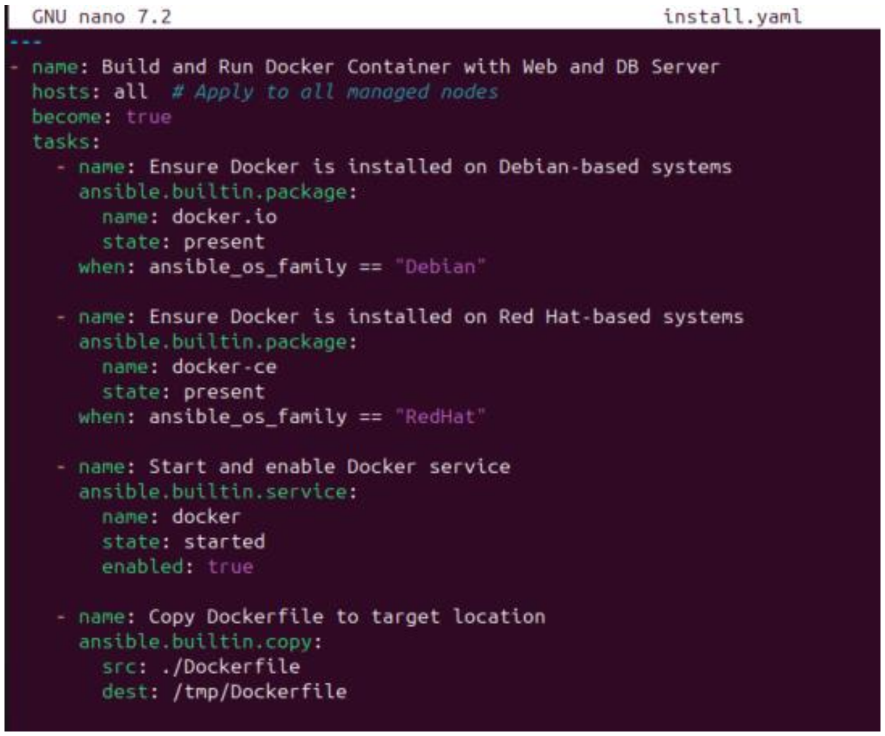
```
julius-de-omampo@workstation:~/Activity-11$ sudo usermod -aG docker $USER
julius-de-omampo@workstation:~/Activity-11$ groups
julius-de-omampo adm cdrom sudo dip plugdev users lpadmin
julius-de-omampo@workstation:~/Activity-11$
```

Adding current user to Docker Group



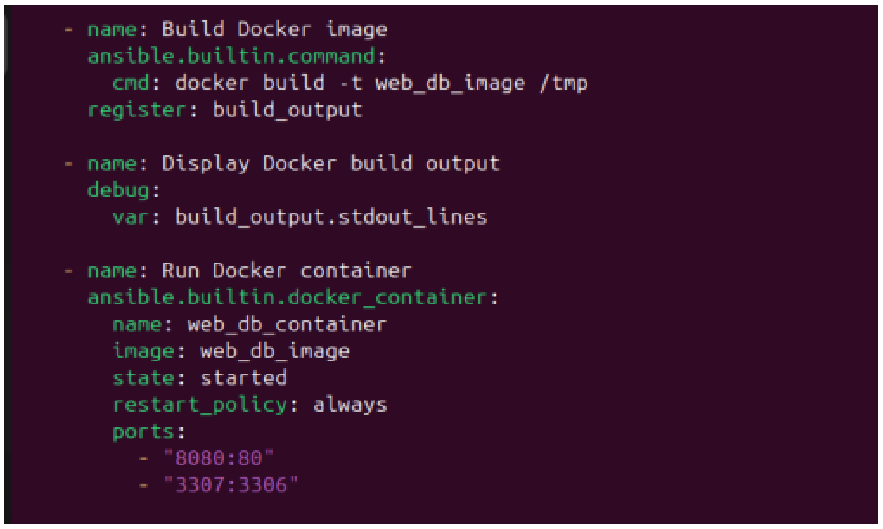
```
julius-de-omampo@workstation:~/Activity-11$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

julius-de-omampo@workstation:~/Activity-11$
```

Testing Docker is functioning



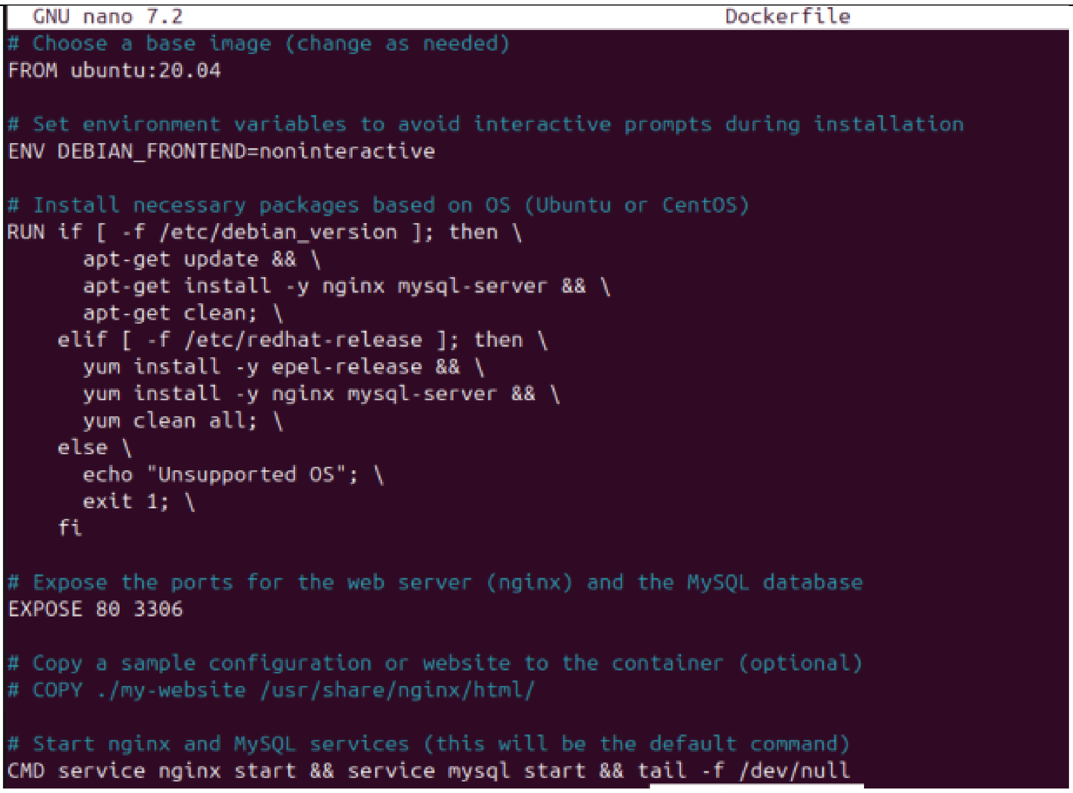
```
GNU nano 7.2                                              install.yaml
---
- name: Build and Run Docker Container with Web and DB Server
  hosts: all  # Apply to all managed nodes
  become: true
  tasks:
    - name: Ensure Docker is installed on Debian-based systems
      ansible.builtin.package:
        name: docker.io
        state: present
      when: ansible_os_family == "Debian"

    - name: Ensure Docker is installed on Red Hat-based systems
      ansible.builtin.package:
        name: docker-ce
        state: present
      when: ansible_os_family == "RedHat"

    - name: Start and enable Docker service
      ansible.builtin.service:
        name: docker
        state: started
        enabled: true

    - name: Copy Dockerfile to target location
      ansible.builtin.copy:
        src: ./Dockerfile
        dest: /tmp/Dockerfile
```

install.yaml (1)



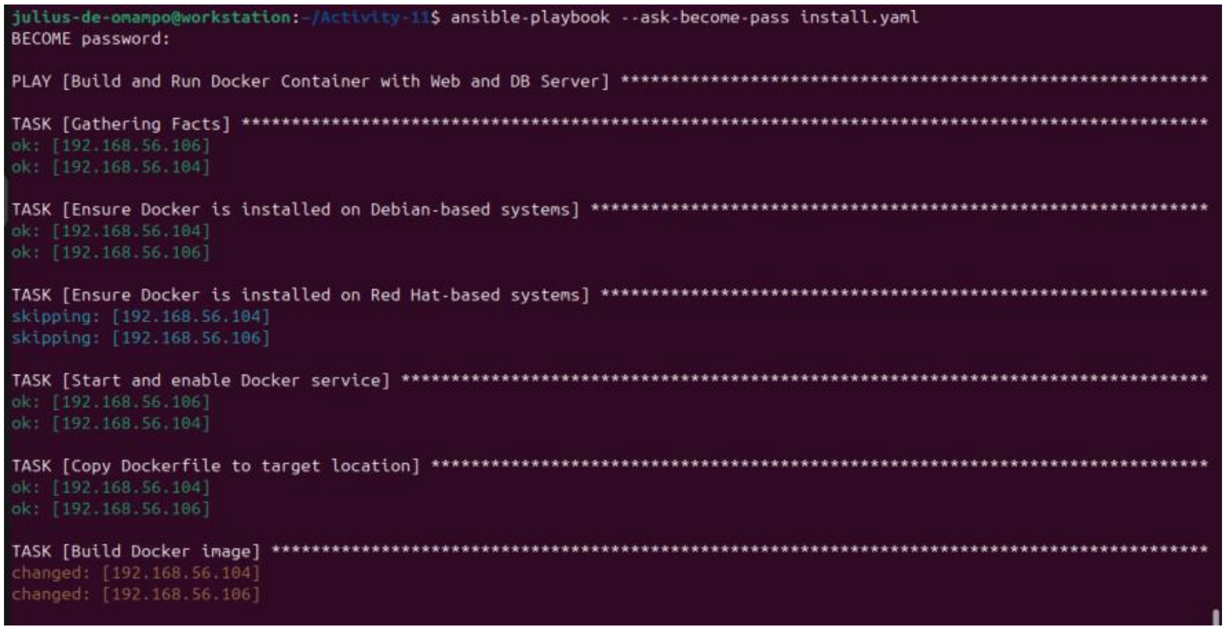
```
    - name: Build Docker image
      ansible.builtin.command:
        cmd: docker build -t web_db_image /tmp
      register: build_output

    - name: Display Docker build output
      debug:
        var: build_output.stdout_lines

    - name: Run Docker container
      ansible.builtin.docker_container:
        name: web_db_container
        image: web_db_image
        state: started
        restart_policy: always
        ports:
          - "8080:80"
          - "3307:3306"
```

install.yaml (2)



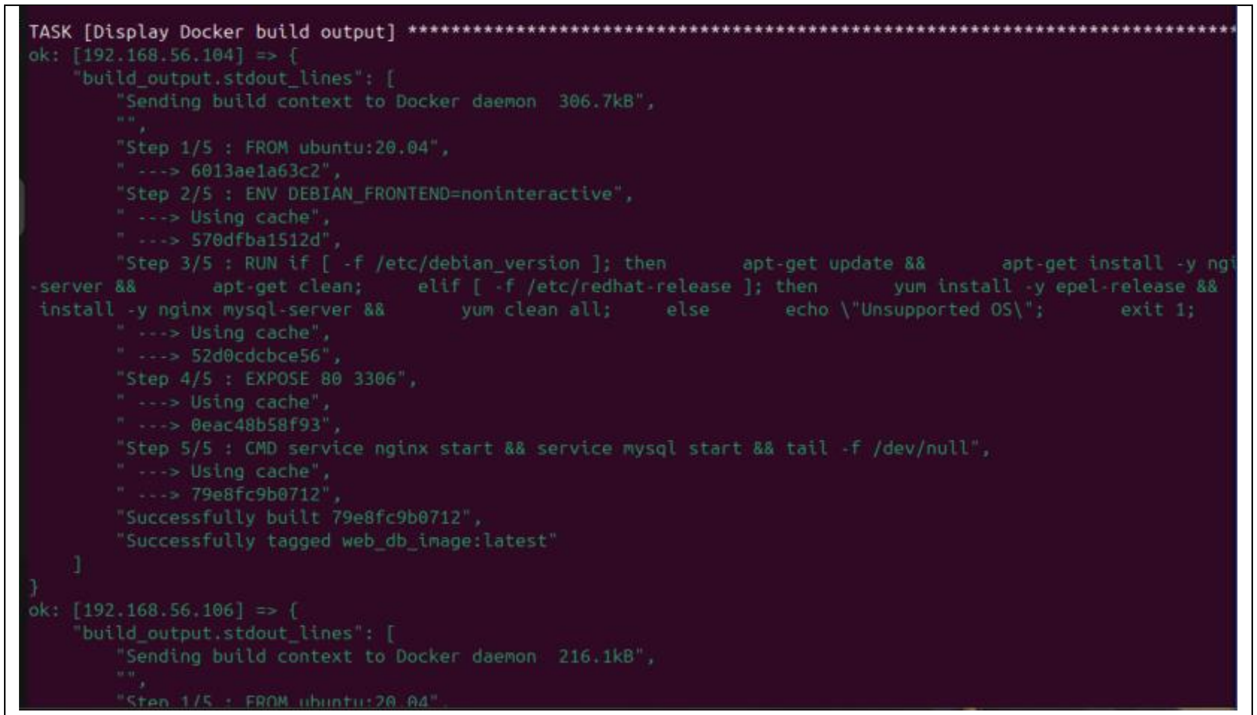
```
  GNU nano 7.2                                            Dockerfile
# Choose a base image (change as needed)
FROM ubuntu:20.04

# Set environment variables to avoid interactive prompts during installation
ENV DEBIAN_FRONTEND=noninteractive

# Install necessary packages based on OS (Ubuntu or CentOS)
RUN if [ -f /etc/debian_version ]; then \
      apt-get update && \
      apt-get install -y nginx mysql-server && \
      apt-get clean; \
    elif [ -f /etc/redhat-release ]; then \
      yum install -y epel-release && \
      yum install -y nginx mysql-server && \
      yum clean all; \
    else \
      echo "Unsupported OS"; \
      exit 1; \
    fi

# Expose the ports for the web server (nginx) and the MySQL database
EXPOSE 80 3306

# Copy a sample configuration or website to the container (optional)
# COPY ./my-website /usr/share/nginx/html/

# Start nginx and MySQL services (this will be the default command)
CMD service nginx start && service mysql start && tail -f /dev/null
```

Dockerfile



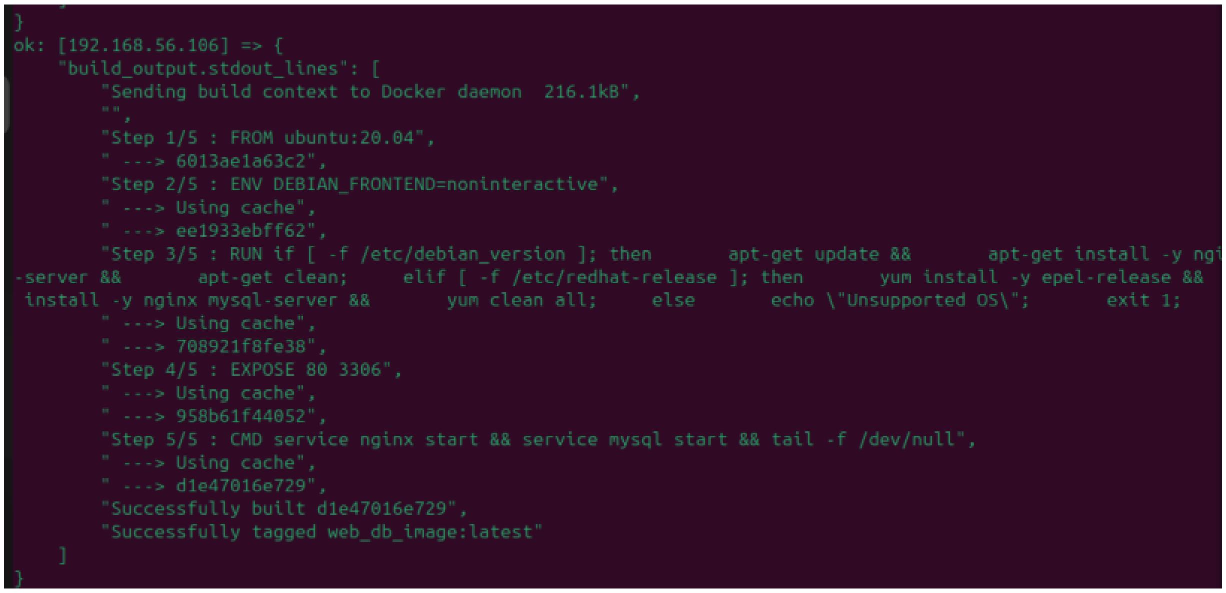
```
julius-de-omampo@workstation:~/Activity-11$ ansible-playbook --ask-become-pass install.yaml
BECOME password:

PLAY [Build and Run Docker Container with Web and DB Server] ***************************************************************

TASK [Gathering Facts] *****************************************************************************************************
ok: [192.168.56.106]
ok: [192.168.56.104]

TASK [Ensure Docker is installed on Debian-based systems] ******************************************************************
ok: [192.168.56.104]
ok: [192.168.56.106]

TASK [Ensure Docker is installed on Red Hat-based systems] *****************************************************************
skipping: [192.168.56.104]
skipping: [192.168.56.106]

TASK [Start and enable Docker service] *************************************************************************************
ok: [192.168.56.106]
ok: [192.168.56.104]

TASK [Copy Dockerfile to target location] **********************************************************************************
ok: [192.168.56.104]
ok: [192.168.56.106]

TASK [Build Docker image] **************************************************************************************************
changed: [192.168.56.104]
changed: [192.168.56.106]
```

Running the playbook (1)



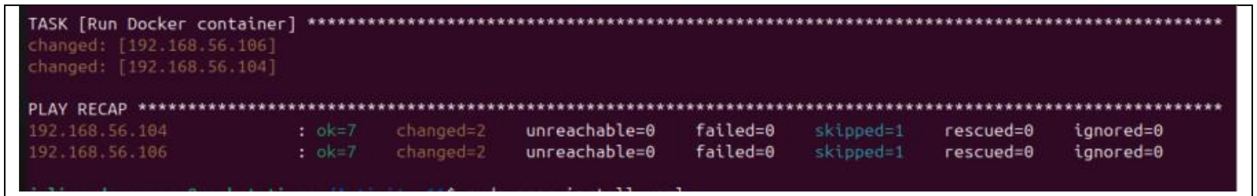
```
TASK [Display Docker build output] ********************************************************************
ok: [192.168.56.104] => {
    "build_output.stdout_lines": [
        "Sending build context to Docker daemon  306.7kB",
        "",
        "Step 1/5 : FROM ubuntu:20.04",
        " ---> 6013ae1a63c2",
        "Step 2/5 : ENV DEBIAN_FRONTEND=noninteractive",
        " ---> Using cache",
        " ---> 570dfba1512d",
        "Step 3/5 : RUN if [ -f /etc/debian_version ]; then       apt-get update &&       apt-get install -y ngi
-server &&       apt-get clean;     elif [ -f /etc/redhat-release ]; then       yum install -y epel-release &&
 install -y nginx mysql-server &&       yum clean all;     else       echo \"Unsupported OS\";       exit 1;
        " ---> Using cache",
        " ---> 52d0cdcbce56",
        "Step 4/5 : EXPOSE 80 3306",
        " ---> Using cache",
        " ---> 0eac48b58f93",
        "Step 5/5 : CMD service nginx start && service mysql start && tail -f /dev/null",
        " ---> Using cache",
        " ---> 79e8fc9b0712",
        "Successfully built 79e8fc9b0712",
        "Successfully tagged web_db_image:latest"
    ]
}
ok: [192.168.56.106] => {
    "build_output.stdout_lines": [
        "Sending build context to Docker daemon  216.1kB",
        "",
        "Step 1/5 : FROM ubuntu:20.04",
```

Running the playbook (2)



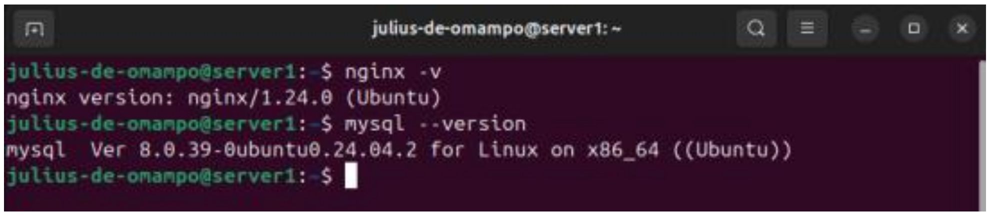
```
}
ok: [192.168.56.106] => {
    "build_output.stdout_lines": [
        "Sending build context to Docker daemon  216.1kB",
        "",
        "Step 1/5 : FROM ubuntu:20.04",
        " ---> 6013ae1a63c2",
        "Step 2/5 : ENV DEBIAN_FRONTEND=noninteractive",
        " ---> Using cache",
        " ---> ee1933ebff62",
        "Step 3/5 : RUN if [ -f /etc/debian_version ]; then       apt-get update &&       apt-get install -y ngi
-server &&       apt-get clean;     elif [ -f /etc/redhat-release ]; then       yum install -y epel-release &&
 install -y nginx mysql-server &&       yum clean all;     else       echo \"Unsupported OS\";       exit 1;
        " ---> Using cache",
        " ---> 708921f8fe38",
        "Step 4/5 : EXPOSE 80 3306",
        " ---> Using cache",
        " ---> 958b61f44052",
        "Step 5/5 : CMD service nginx start && service mysql start && tail -f /dev/null",
        " ---> Using cache",
        " ---> d1e47016e729",
        "Successfully built d1e47016e729",
        "Successfully tagged web_db_image:latest"
    ]
}
```

Runing the playbook (3)



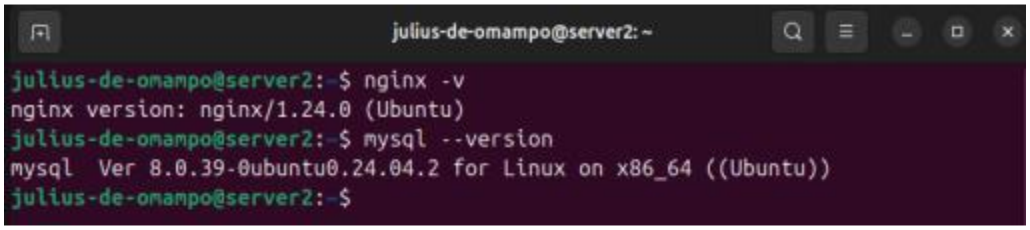
```
TASK [Run Docker container] ****************************************************************************
changed: [192.168.56.106]
changed: [192.168.56.104]

PLAY RECAP *********************************************************************************************
192.168.56.104             : ok=7    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
192.168.56.106             : ok=7    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

Running the playbook (4)



```
julius-de-omampo@server1:~$ nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
julius-de-omampo@server1:~$ mysql --version
mysql  Ver 8.0.39-0ubuntu0.24.04.2 for Linux on x86_64 ((Ubuntu))
julius-de-omampo@server1:~$
```

server1



```
julius-de-omampo@server2:~$ nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
julius-de-omampo@server2:~$ mysql --version
mysql  Ver 8.0.39-0ubuntu0.24.04.2 for Linux on x86_64 ((Ubuntu))
julius-de-omampo@server2:~$
```

server2

GitHub Link:

https://github.com/jmado-biscoff/Activity-11.git

Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?

Conclusions:

Using Docker with Ansible offers a powerful combination for automating the deployment and management of containerized applications. Docker provides a consistent, isolated environment for applications, while Ansible streamlines configuration management and orchestration. With Ansible, you can automate the setup of Docker containers, manage network configurations, and ensure reproducible environments. This integration simplifies complex workflows, enabling consistent deployments across different systems, and improving scalability and efficiency in managing containerized infrastructures.