Name: patrick cruz	Date Performed: oct 24 2025
Course/Section:cpe212 - cpe31s4	Date Submitted: oct 24 2025
Instructor: Engr. Robin valenzuela	Semester and SY: 1st sem

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/

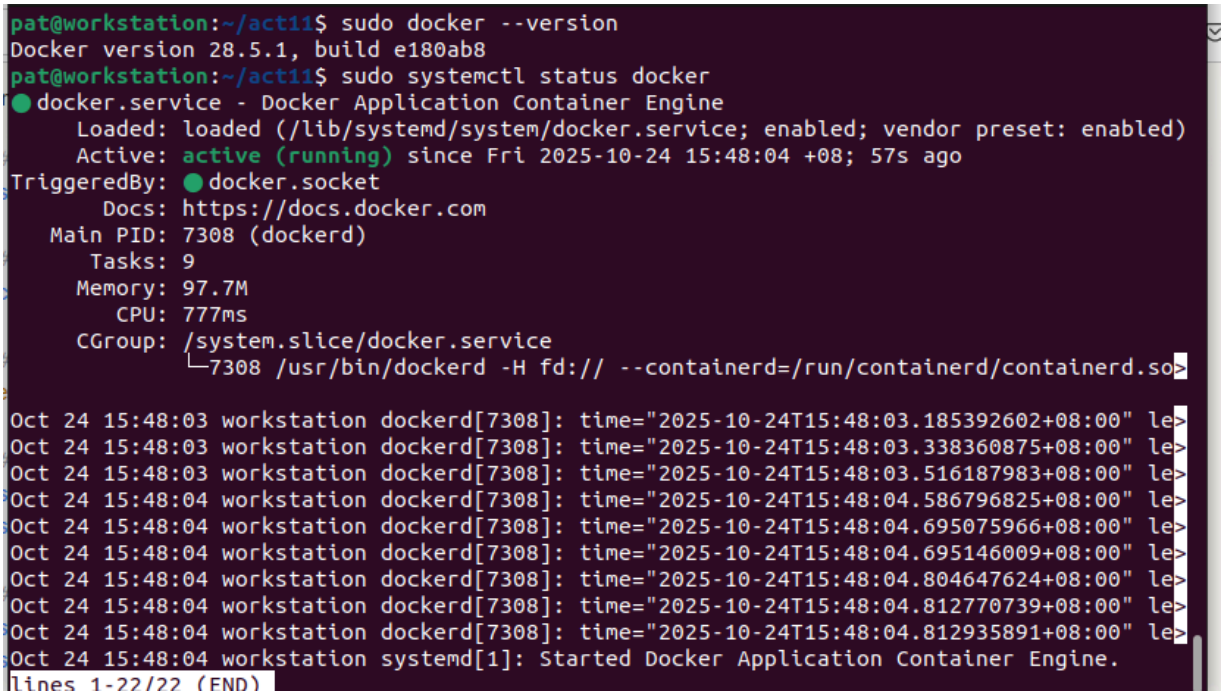
You may also check the difference between containers and virtual machines. Click the link given below.

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

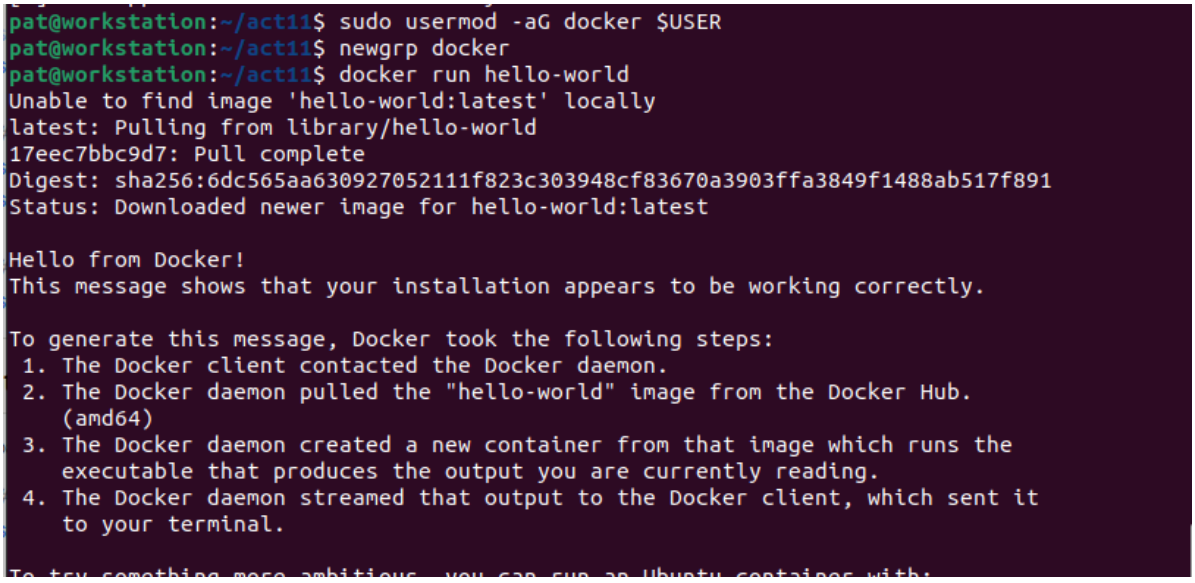
3. Tasks

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)



```
pat@workstation:~/act11$ sudo docker --version
Docker version 28.5.1, build e180ab8
pat@workstation:~/act11$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2025-10-24 15:48:04 +08; 57s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 7308 (dockerd)
      Tasks: 9
     Memory: 97.7M
        CPU: 777ms
     CGroup: /system.slice/docker.service
             └─7308 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.so>

Oct 24 15:48:03 workstation dockerd[7308]: time="2025-10-24T15:48:03.185392602+08:00" le>
Oct 24 15:48:03 workstation dockerd[7308]: time="2025-10-24T15:48:03.338360875+08:00" le>
Oct 24 15:48:03 workstation dockerd[7308]: time="2025-10-24T15:48:03.516187983+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.586796825+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.695075966+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.695146009+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.804647624+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.812770739+08:00" le>
Oct 24 15:48:04 workstation dockerd[7308]: time="2025-10-24T15:48:04.812935891+08:00" le>
Oct 24 15:48:04 workstation systemd[1]: Started Docker Application Container Engine.
lines 1-22/22 (END)
```

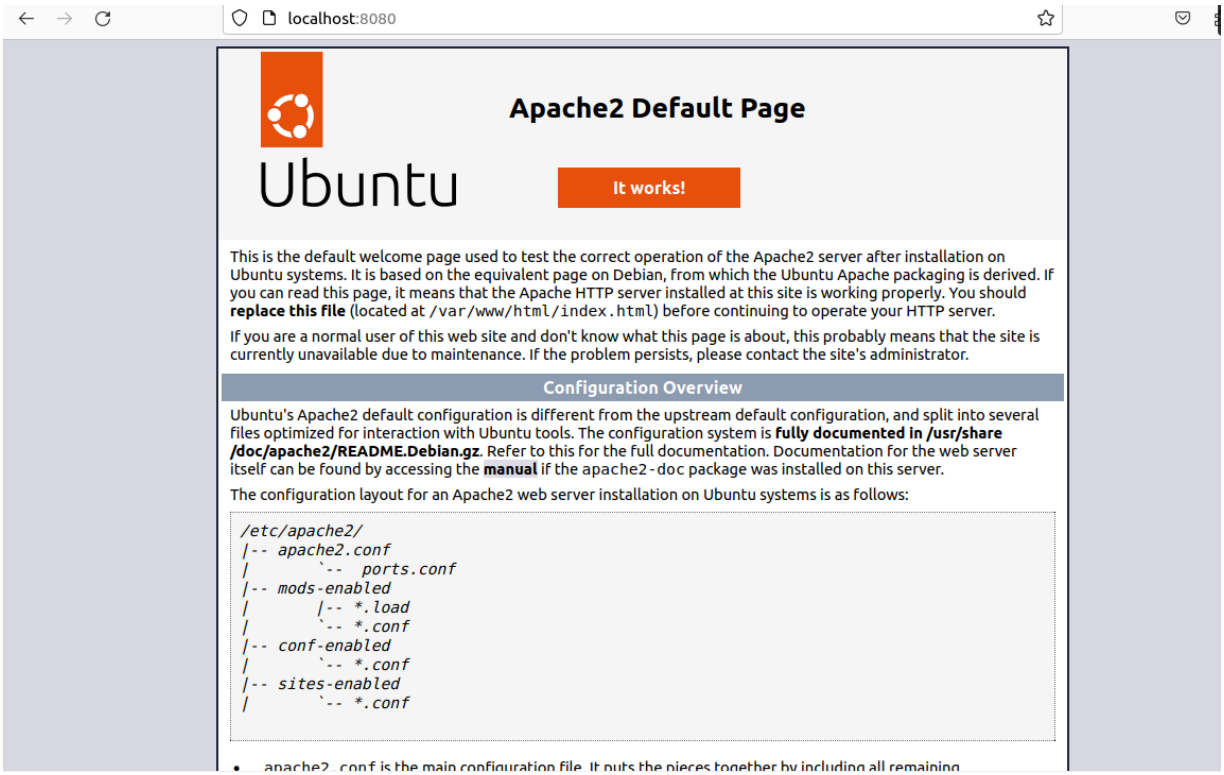


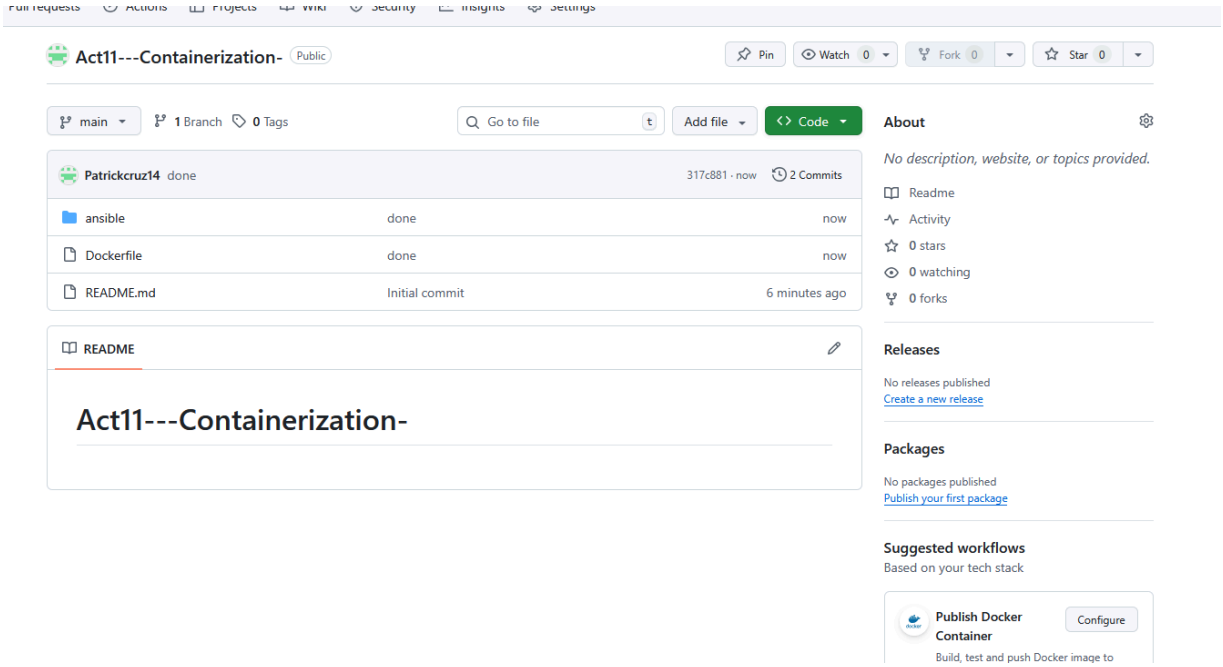
```
pat@workstation:~/act11$ sudo usermod -aG docker $USER
pat@workstation:~/act11$ newgrp docker
pat@workstation:~/act11$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:6dc565aa630927052111f823c303948cf83670a3903ffa3849f1488ab517f891
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:
```

```
pat@workstation:~/act11$ cat Dockerfile
FROM ubuntu:22.04

# Avoid prompts during package installation
ENV DEBIAN_FRONTEND=noninteractive

# Update and install packages
RUN apt-get update && apt-get install -y \
    apache2 \
    mysql-server \
    php \
    php-mysql \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Configure Apache
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

# Expose ports
EXPOSE 80 3306

# Start services
CMD service mysql start && apachectl -D FOREGROUND
```

```
pat@workstation:~/act11$ ansible-playbook -i ansible/inventory ansible/docker-playbook.yml

PLAY [Build Docker Image for Web and DB Server] ************************************

TASK [Gathering Facts] *************************************************************
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter
at /usr/bin/python3.10, but future installation of another Python interpreter could
change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [Ensure Docker service is running] ********************************************
ok: [localhost]

TASK [Install Python Docker module] ************************************************
ok: [localhost]

TASK [Copy Dockerfile to build location] *******************************************
changed: [localhost]

TASK [Build Docker image] **********************************************************
changed: [localhost]

TASK [Verify Docker image was created] *********************************************
```

```
TASK [Ensure Docker service is running] ****************************************************
ok: [localhost]

TASK [Install Python Docker module] ********************************************************
ok: [localhost]

TASK [Copy Dockerfile to build location] ***************************************************
changed: [localhost]

TASK [Build Docker image] ******************************************************************
changed: [localhost]

TASK [Verify Docker image was created] *****************************************************
ok: [localhost]

TASK [Display image information] ***********************************************************
ok: [localhost] => {
    "msg": "Image ['webapp-db:latest'] created successfully"
}

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





Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?

- **Containerization ensures applications run consistently across different environments by packaging all dependencies together. It's lightweight, fast, and efficient, making it easier to scale, update, and deploy software reliably while improving teamwork and reducing system conflicts.**

Conclusions:

In conclusion, this activity successfully demonstrated the power and practicality of containerization using Docker, automated through Ansible. We created a portable, self-contained environment that reliably runs both a web server and a database, effectively eliminating environment-specific inconsistencies. By integrating Ansible, we established a reproducible Infrastructure as Code (IaC) workflow, showcasing a foundational step towards a robust Continuous Delivery pipeline. This hands-on experience confirms that containerization is not just a theoretical concept but a vital tool for modern

software development, enabling faster deployments, improved scalability, and greater operational efficiency.