Name:Tracey Dee Bringuela	**Date Performed:13/11/24**
Course/Section:CPE31S2	**Date Submitted:13/11/24**
Instructor: 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.

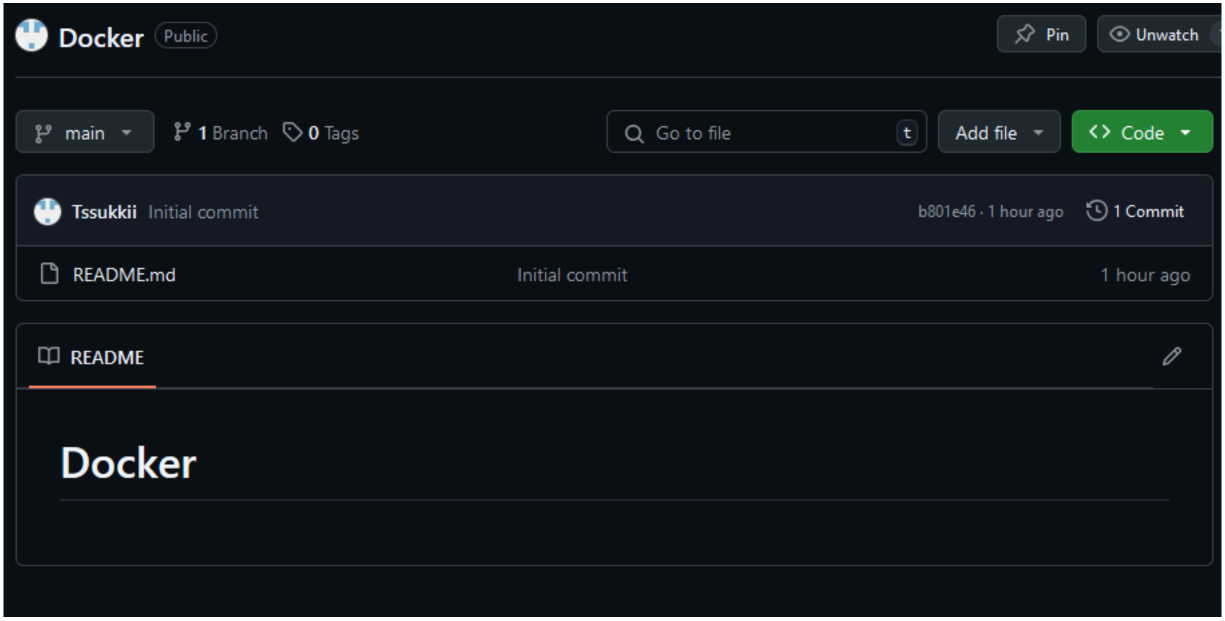
Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

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)

1. create a new repository for this activity

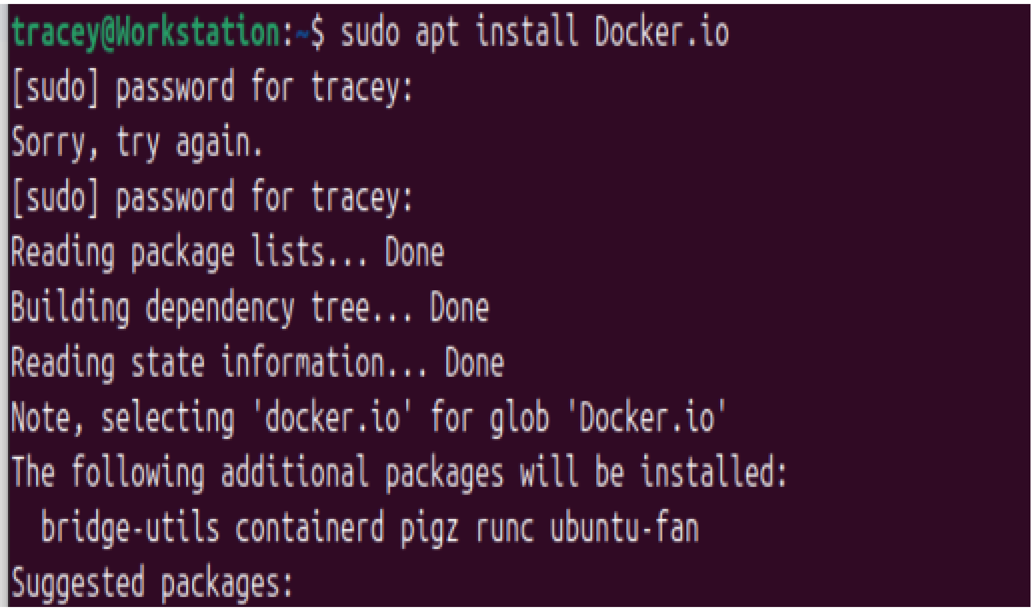


```
Tsukkii  Initial commit                         b801e46 · 1 hour ago   1 Commit
README.md            Initial commit                                  1 hour ago
```

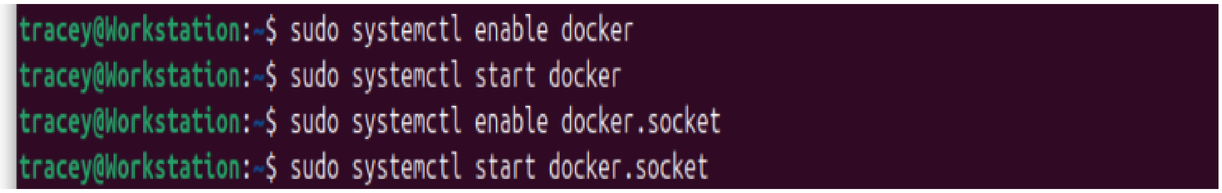
README

Docker

2. Install Docker and enable the docker socket.

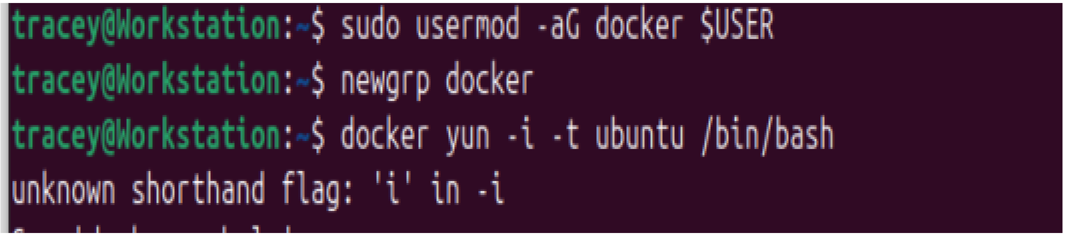


```
tracey@Workstation:~$ sudo apt install Docker.io
[sudo] password for tracey:
Sorry, try again.
[sudo] password for tracey:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'docker.io' for glob 'Docker.io'
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
```



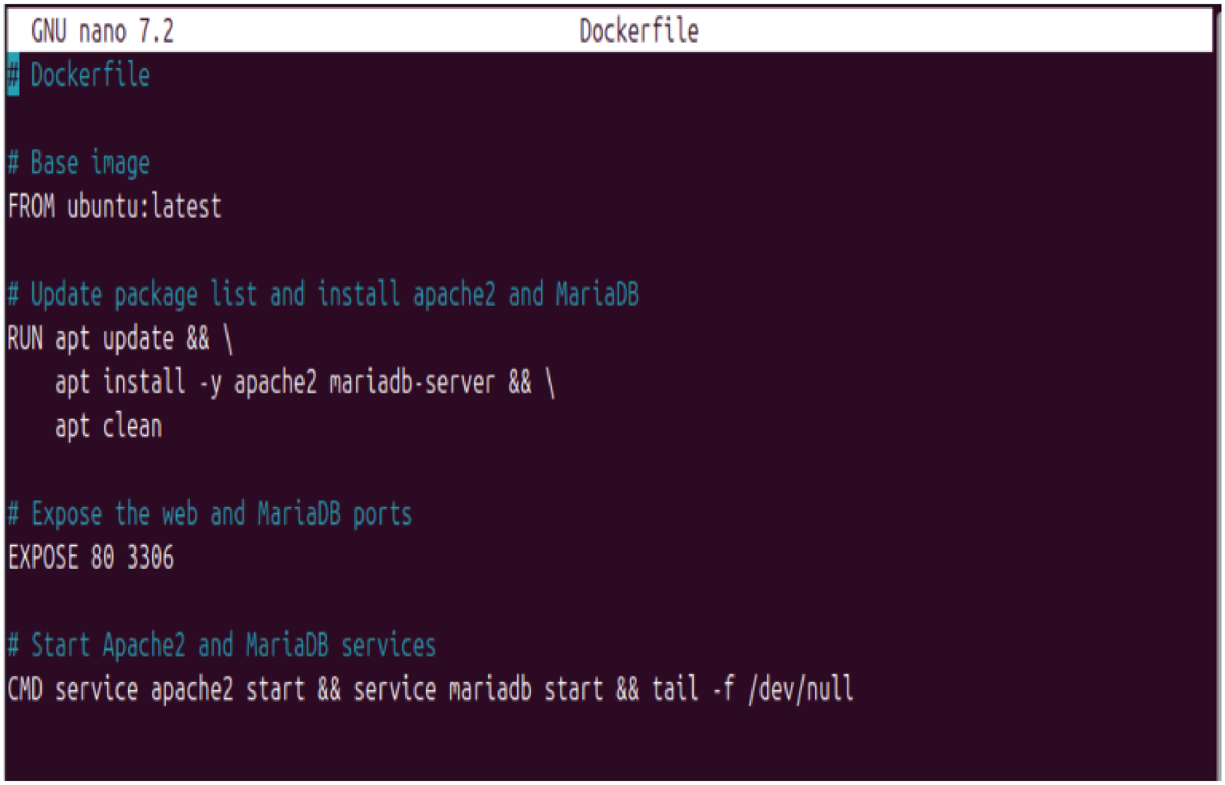
```
tracey@Workstation:~$ sudo systemctl enable docker
tracey@Workstation:~$ sudo systemctl start docker
tracey@Workstation:~$ sudo systemctl enable docker.socket
tracey@Workstation:~$ sudo systemctl start docker.socket
```

3. add the docker to the user group



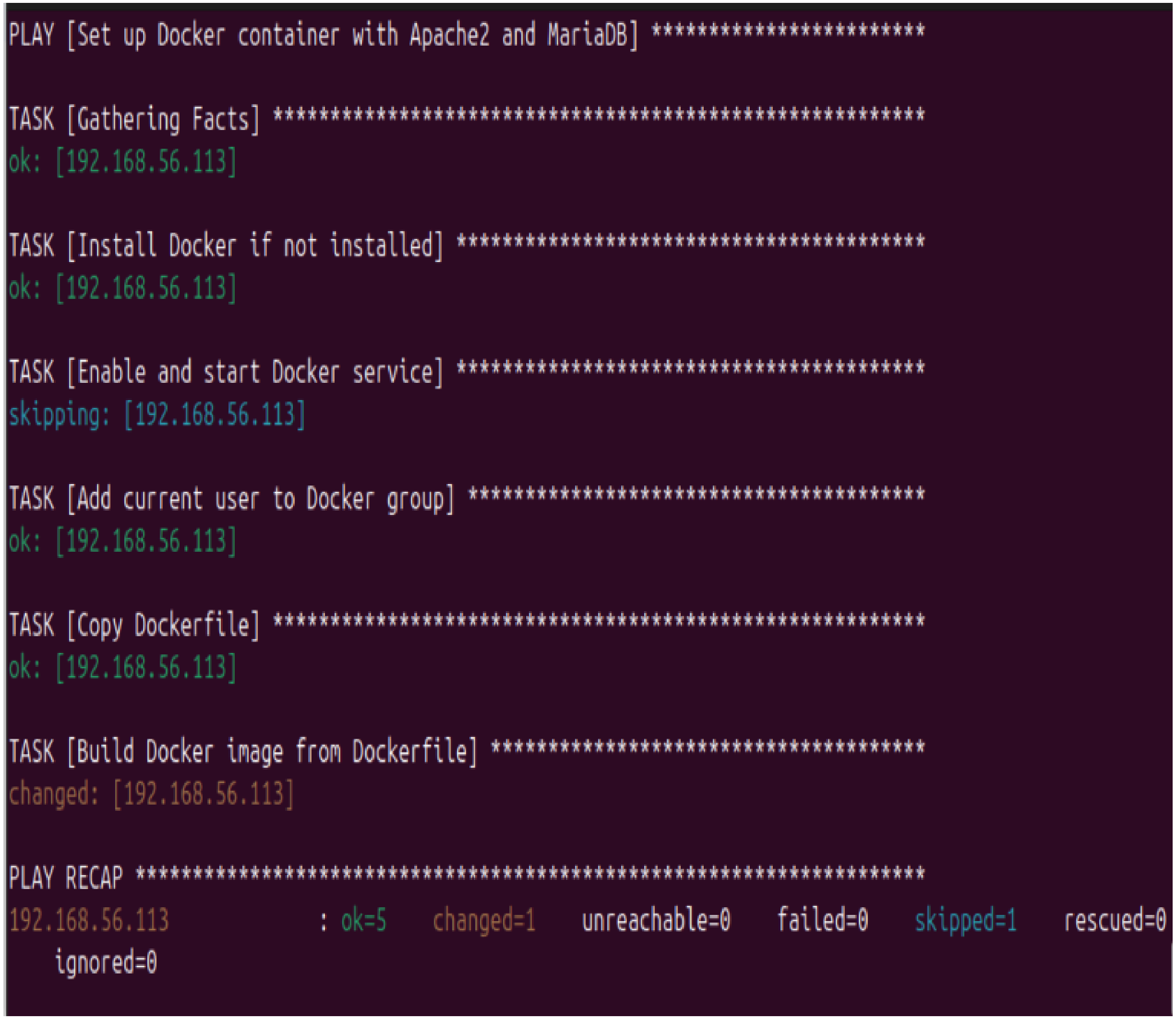
```
tracey@Workstation:~$ sudo usermod -aG docker $USER
tracey@Workstation:~$ newgrp docker
tracey@Workstation:~$ docker yun -i -t ubuntu /bin/bash
unknown shorthand flag: 'i' in -i
```

4.Create a Dockerfile to install web and DB server.

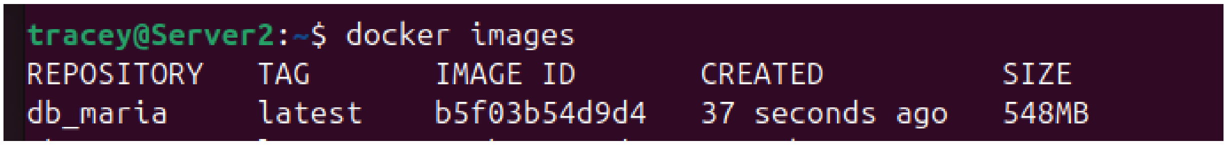


```
# Dockerfile

# Base image
FROM ubuntu:latest

# Update package list and install apache2 and MariaDB
RUN apt update && \
    apt install -y apache2 mariadb-server && \
    apt clean

# Expose the web and MariaDB ports
EXPOSE 80 3306

# Start Apache2 and MariaDB services
CMD service apache2 start && service mariadb start && tail -f /dev/null
```

5. Install and build the Dockerfile using Ansible.

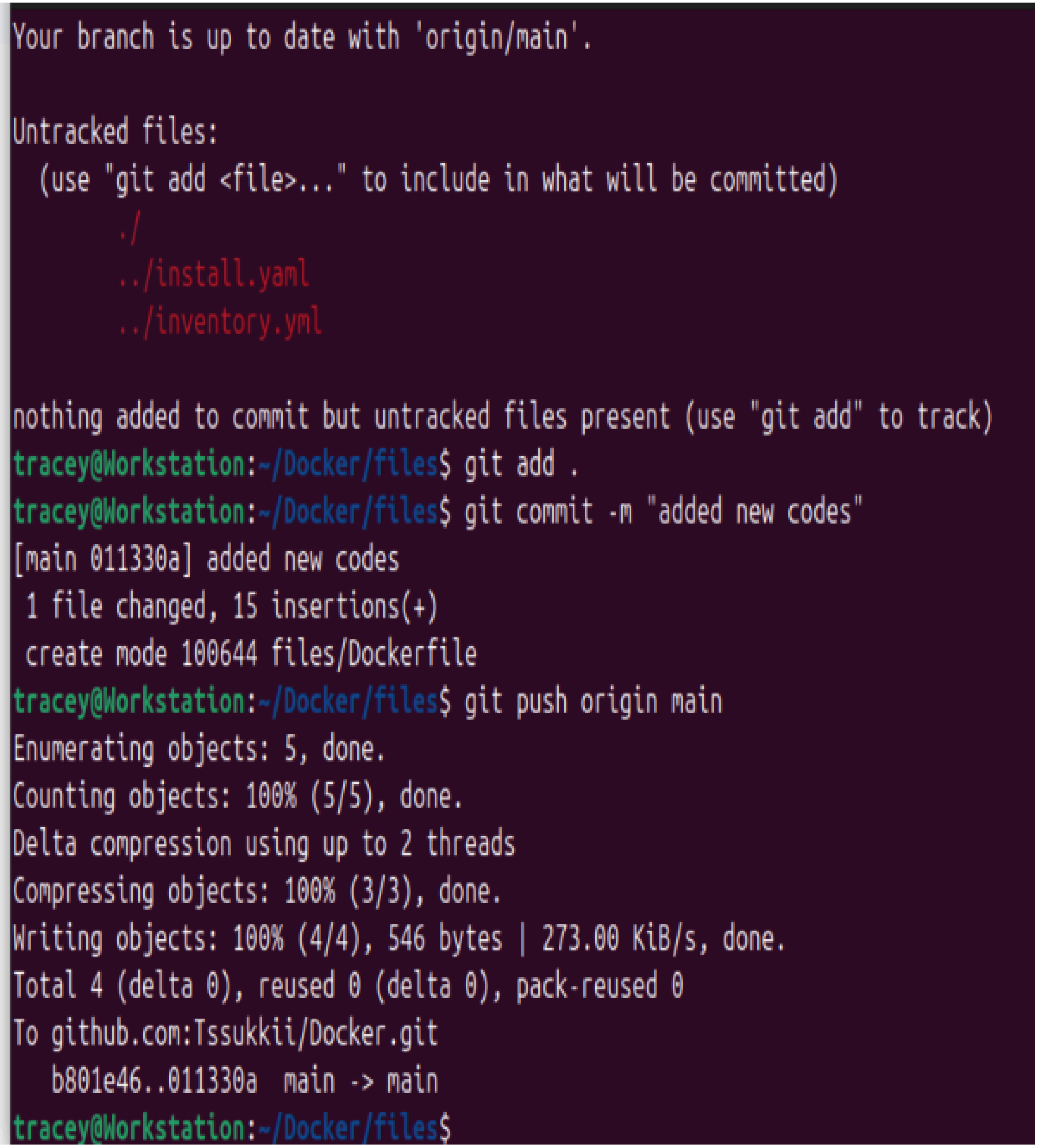


```
PLAY [Set up Docker container with Apache2 and MariaDB] ************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.113]

TASK [Install Docker if not installed] *****************************************
ok: [192.168.56.113]

TASK [Enable and start Docker service] *****************************************
skipping: [192.168.56.113]

TASK [Add current user to Docker group] ****************************************
ok: [192.168.56.113]

TASK [Copy Dockerfile] *********************************************************
ok: [192.168.56.113]

TASK [Build Docker image from Dockerfile] **************************************
changed: [192.168.56.113]

PLAY RECAP *********************************************************************
192.168.56.113             : ok=5    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
    ignored=0
```



```
tracey@Server2:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
db_maria      latest    b5f03b54d9d4   37 seconds ago   548MB
```

6. Add, commit and push it to your repository.



```
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
	./
	../install.yaml
	../inventory.yml

nothing added to commit but untracked files present (use "git add" to track)
tracey@Workstation:~/Docker/files$ git add .
tracey@Workstation:~/Docker/files$ git commit -m "added new codes"
[main 011330a] added new codes
 1 file changed, 15 insertions(+)
 create mode 100644 files/Dockerfile
tracey@Workstation:~/Docker/files$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 546 bytes | 273.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Tssukkii/Docker.git
   b801e46..011330a  main -> main
tracey@Workstation:~/Docker/files$
```

Reflections:
Answer the following:

1. What are the benefits of implementing containerizations?
   Containerization enhances application deployment by ensuring consistency across various environments, which reduces the "it works on my machine" problem. It improves resource utilization and scalability, allowing multiple containers to run on a single host without interference, thus optimizing infrastructure costs. Additionally, containerization facilitates faster development and deployment cycles through automation and orchestration, enabling teams to deliver updates and new features more efficiently.

Conclusions:

In conclusion, implementing a Dockerfile and leveraging Ansible as Infrastructure as Code (IaC) significantly streamlines the Continuous Delivery process by automating the deployment of applications and their dependencies. This approach not only enhances the efficiency of managing infrastructure but also promotes rapid application development and deployment, allowing teams to respond swiftly to changing requirements and deliver high-quality software with minimal downtime. By integrating Docker and Ansible, organizations can achieve a more consistent and scalable environment, ultimately driving productivity and innovation in their software development lifecycle.