Name: Khelvin P. Nicolas	**Date Performed: 13/11/2024**
Course/Section: CPE 212 - CPE31S2	**Date Submitted: 13/11/2024**
Instructor: Engr. Robin Valenzuela	**Semester and SY:3rd Yr.**

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 a Docker group to your current user.
4. Create a Dockerfile to install web and DB servers.
5. Install and build the Dockerfile using Ansible.
6. Add, commit and push it to your repository

4. Output (screenshots and explanations)

GITHUB LINK: https://github.com/KHLVN/CPE212_Activity11

1. **Create a new repository for this activity.**

- I started by setting up a repository for this activity on GitHub to track all files and changes for this project.

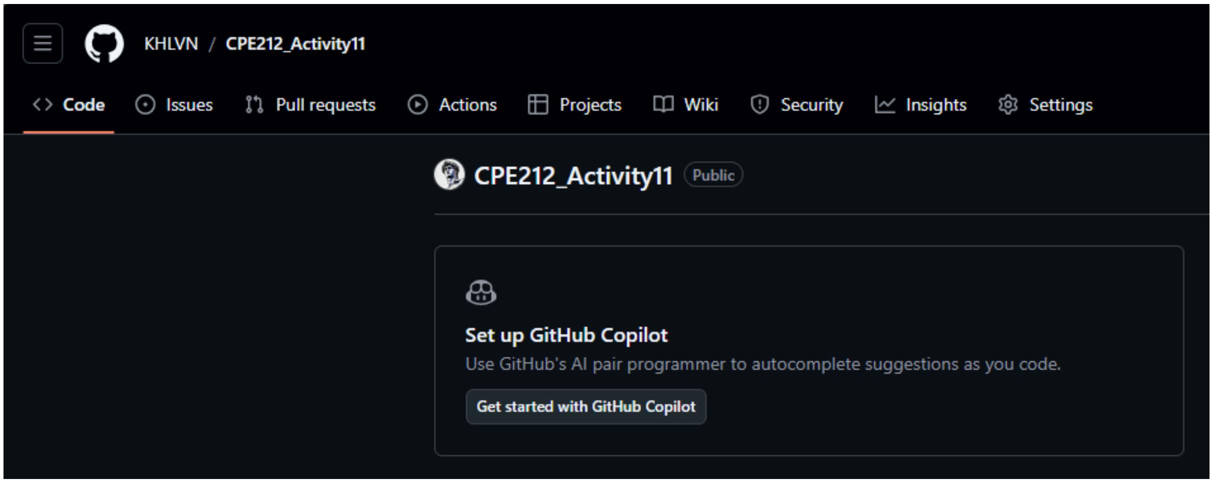
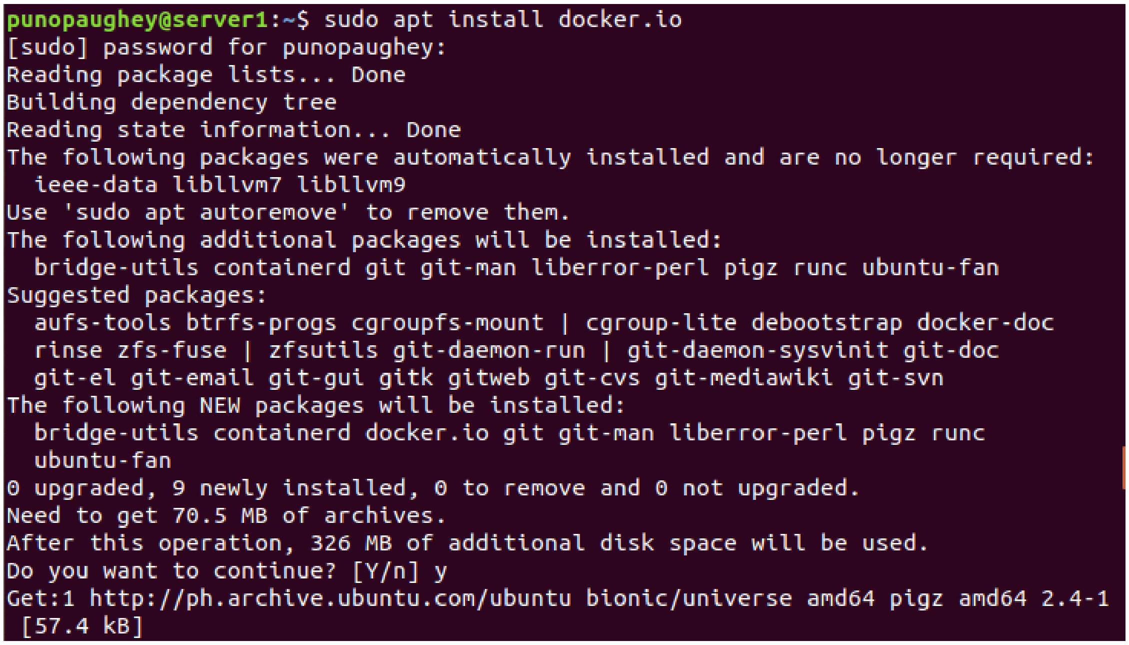

KHLVN / CPE212_Activity11
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
CPE212_Activity11
Public
Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot

Figure 1: Creating repository for Activity 11

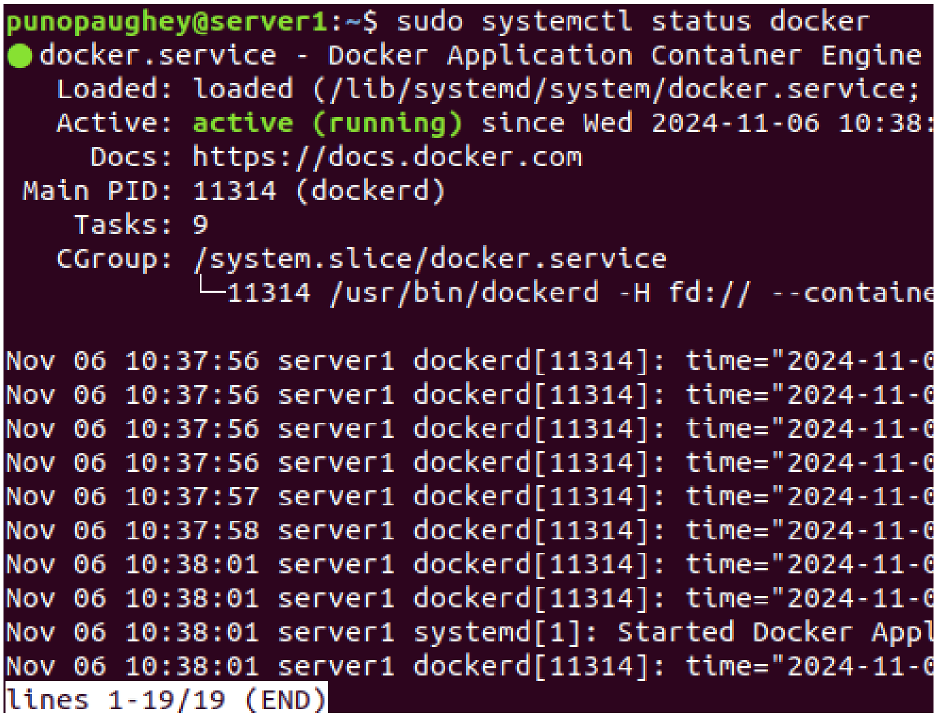
2. **Install Docker and enable the docker socket.**

- I installed Docker on my control node, then enabled the Docker socket using the ***"systemctl start docker"*** command so local applications could communicate with Docker's services.



```
punopaughey@server1:~$ sudo apt install docker.io
[sudo] password for punopaughey:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  ieee-data libllvm7 libllvm9
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc
  rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit git-doc
  git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 70.5 MB of archives.
After this operation, 326 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1
 [57.4 kB]
```

Figure 2.1: Installing docker through Ubuntu CLI

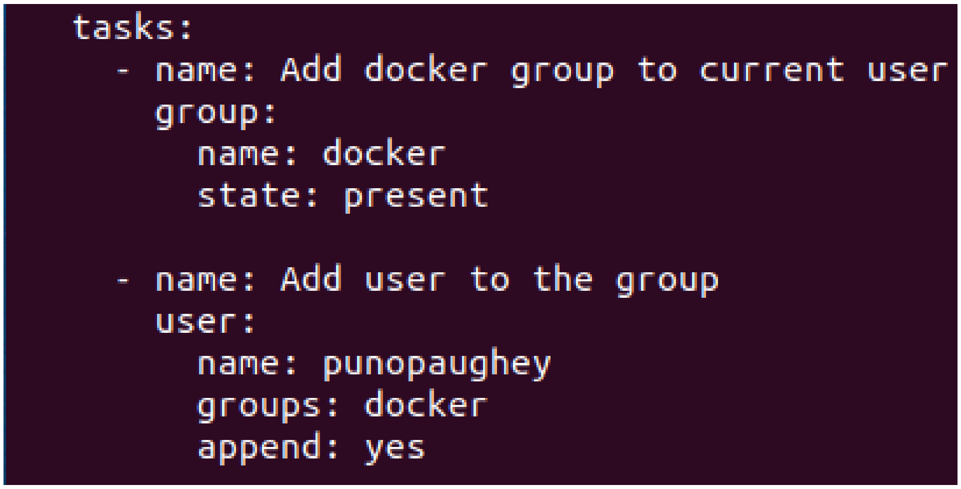


```
punopaughey@server1:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service;
   Active: active (running) since Wed 2024-11-06 10:38:
     Docs: https://docs.docker.com
 Main PID: 11314 (dockerd)
    Tasks: 9
   CGroup: /system.slice/docker.service
           └─11314 /usr/bin/dockerd -H fd:// --containe

Nov 06 10:37:56 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:37:56 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:37:56 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:37:56 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:37:57 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:37:58 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:38:01 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:38:01 server1 dockerd[11314]: time="2024-11-0
Nov 06 10:38:01 server1 systemd[1]: Started Docker Appl
Nov 06 10:38:01 server1 dockerd[11314]: time="2024-11-0
lines 1-19/19 (END)
```

Figure 2.2: Enabling the docker service

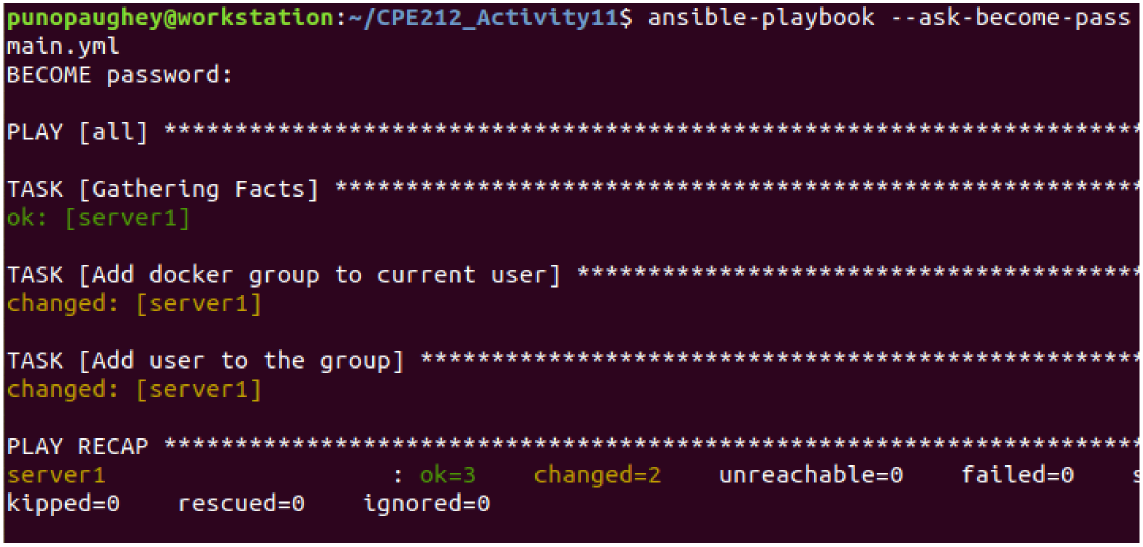
3. **Add a Docker group to your current user.**

- To streamline Docker commands, I added my user to the docker group. Before this, I have created a playbook that adds a group named docker and added my current user in it. This allowed me to manage Docker containers without needing to use sudo every time, which simplified the workflow of this activity.



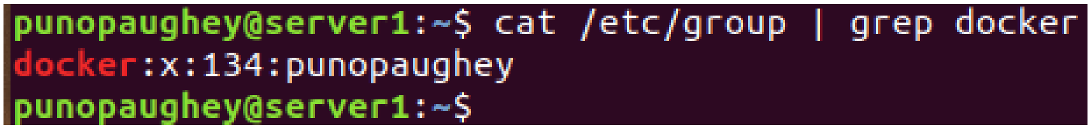
```
  tasks:
    - name: Add docker group to current user
      group:
        name: docker
        state: present

    - name: Add user to the group
      user:
        name: punopaughey
        groups: docker
        append: yes
```

Figure 3.1: Creating and adding docker group



```
punopaughey@workstation:~/CPE212_Activity11$ ansible-playbook --ask-become-pass
main.yml
BECOME password:

PLAY [all] ***********************************************************************

TASK [Gathering Facts] ***********************************************************
ok: [server1]

TASK [Add docker group to current user] *****************************************
changed: [server1]

TASK [Add user to the group] *****************************************************
changed: [server1]

PLAY RECAP ***********************************************************************
server1                    : ok=3    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figure 3.1: running the playbook

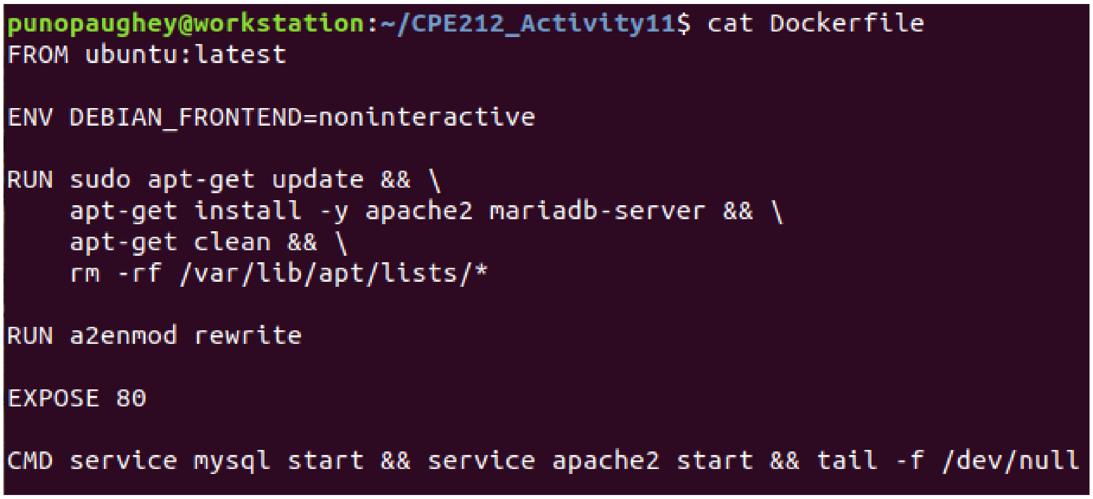


```
punopaughey@server1:~$ cat /etc/group | grep docker
docker:x:134:punopaughey
punopaughey@server1:~$
```

Figure 3.2: Verifying creation of docker group using cat and grep command

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

- I then wrote a Dockerfile to define the installation steps for a web server (Apache2) and a database server (MariaDB). This file served as a blueprint to be used later for building the image, ensuring that the containerized environment would have everything it needed.

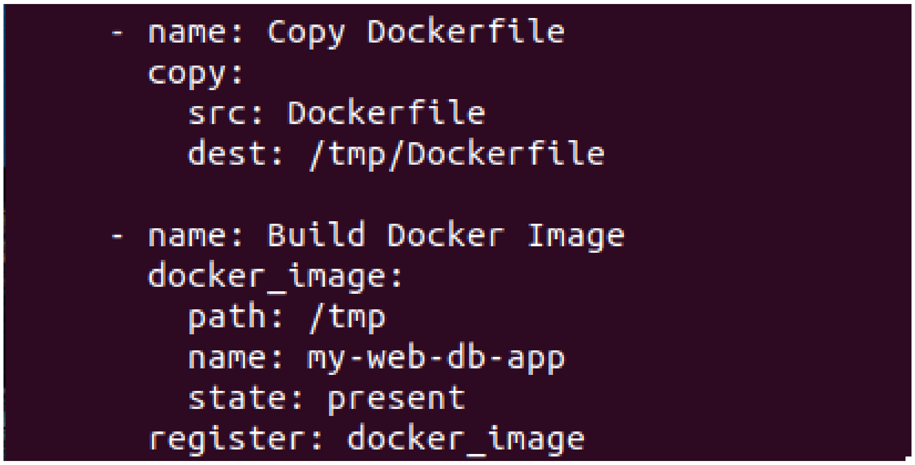


```
punopaughey@workstation:~/CPE212_Activity11$ cat Dockerfile
FROM ubuntu:latest

ENV DEBIAN_FRONTEND=noninteractive

RUN sudo apt-get update && \
    apt-get install -y apache2 mariadb-server && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

RUN a2enmod rewrite

EXPOSE 80

CMD service mysql start && service apache2 start && tail -f /dev/null
```

Figure 4.1: Dockerfile

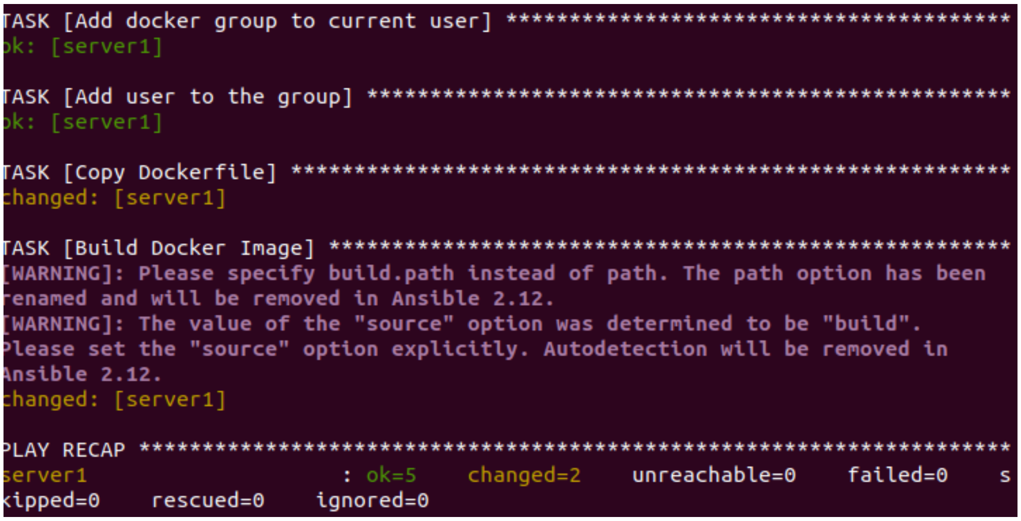
5. Install and build the Dockerfile using Ansible.

- Using Ansible, I automated the process of building the Docker image from the Dockerfile that I've created earlier with the help of the docker_image module from ansible. This approach ensured that the environment was set up consistently each time, avoiding manual errors.



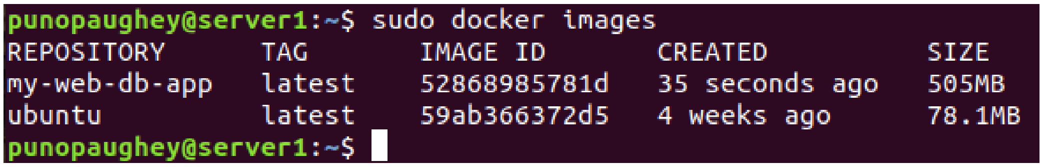
```
    - name: Copy Dockerfile
      copy:
        src: Dockerfile
        dest: /tmp/Dockerfile

    - name: Build Docker Image
      docker_image:
        path: /tmp
        name: my-web-db-app
        state: present
      register: docker_image
```

Figure 5.1: Ansible task for building docker image.



```
TASK [Add docker group to current user] ****************************************
ok: [server1]

TASK [Add user to the group] ***************************************************
ok: [server1]

TASK [Copy Dockerfile] *********************************************************
changed: [server1]

TASK [Build Docker Image] ******************************************************
[WARNING]: Please specify build.path instead of path. The path option has been
renamed and will be removed in Ansible 2.12.
[WARNING]: The value of the "source" option was determined to be "build".
Please set the "source" option explicitly. Autodetection will be removed in
Ansible 2.12.
changed: [server1]

PLAY RECAP *********************************************************************
server1                    : ok=5    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

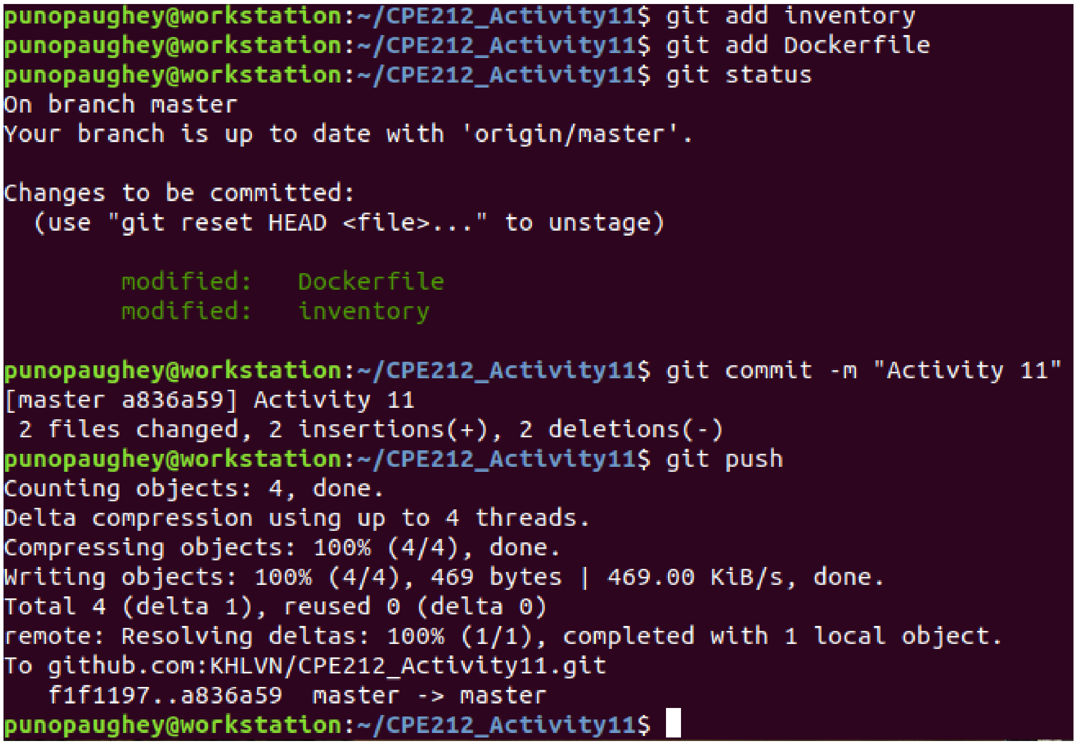
Figure 5.2: running the main.yml playbook



```
punopaughey@server1:~$ sudo docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
my-web-db-app    latest    52868985781d   35 seconds ago   505MB
ubuntu           latest    59ab366372d5   4 weeks ago      78.1MB
punopaughey@server1:~$ 
```

Figure 5.3: Checking the created docker image

6. **Add, commit and push it to your repository.**
   - Finally, I staged all changes, committed them, and pushed everything to the remote repository. This kept my work updated in the GitHub link provided above.



```
punopaughey@workstation:~/CPE212_Activity11$ git add inventory
punopaughey@workstation:~/CPE212_Activity11$ git add Dockerfile
punopaughey@workstation:~/CPE212_Activity11$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Dockerfile
        modified:   inventory

punopaughey@workstation:~/CPE212_Activity11$ git commit -m "Activity 11"
[master a836a59] Activity 11
 2 files changed, 2 insertions(+), 2 deletions(-)
punopaughey@workstation:~/CPE212_Activity11$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 469 bytes | 469.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:KHLVN/CPE212_Activity11.git
   f1f1197..a836a59  master -> master
punopaughey@workstation:~/CPE212_Activity11$
```

Figure 6.1: Committing changes to GitHub

Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?
   - Containers make applications portable, so they work consistently across different environments like development and production. They package everything with their dependencies, avoiding conflicts and providing a reliable, isolated environment. The setup is also lightweight, making it easy to scale up as needed without consuming too many resources, which helps reduce infrastructure costs.

Conclusions:

- By implementing containerization in this activity, I've gained several advantages regarding Docker containerization. The tasks done on this activity serves as our basis for the final project which is the Docker environment. We have to familiarize ourselves in utilizing Docker in order to lessen the struggles of troubleshooting other devices after the created applications don't work on other devices.