Name: Planta, Calvin Earl L.	Date Performed: 10/24/25
Course/Section: CPE 212 - CPE31S2	Date Submitted: 10/24/25
Instructor: Engr. Robin Valenzuela	Semester and SY: 1st Sem S.Y. 25-26

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.

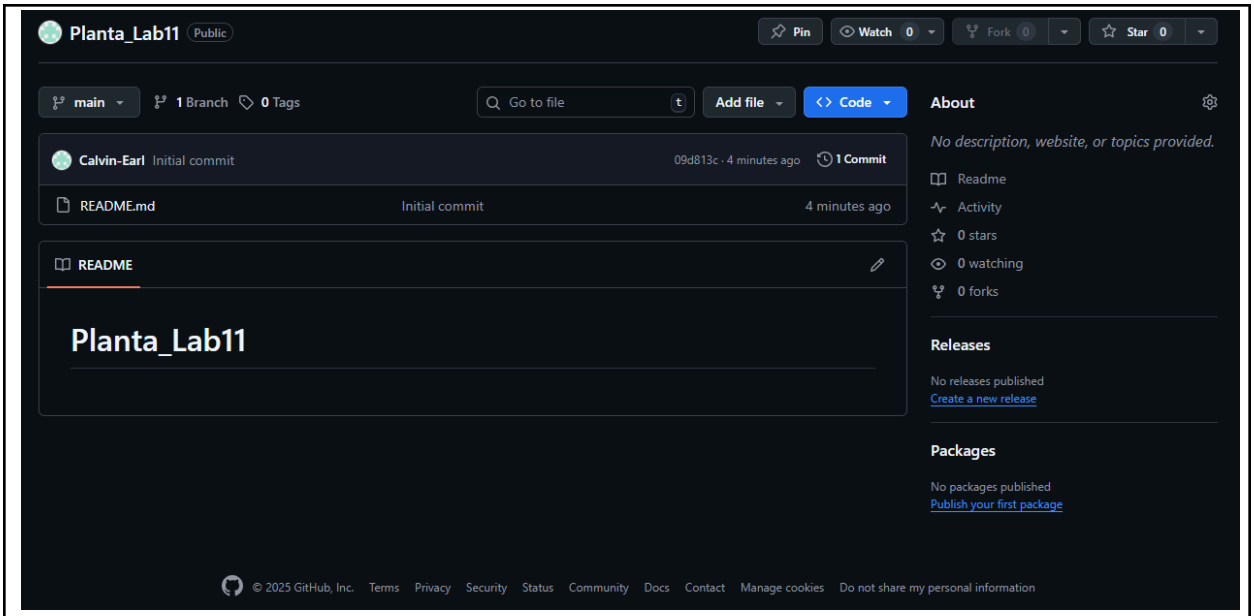
Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm

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)

1. Create a new repository for this activity.



2. Install Docker and enable the docker socket.

Ubuntu Workstation and Server

```
  GNU nano 7.2                    installdocker.yml
---
- name: Install Docker on all nodes
  hosts: ubuntu
  become: yes
  tasks:
    - name: Update APT package cache
      apt:
        update_cache: yes

    - name: Install prerequisite packages
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
        state: present
```

```
GNU nano 7.2                    installdocker.yml
```

```
- name: Add Docker's official GPG key
  ansible.builtin.apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
    state: present

- name: Add Docker repository
  ansible.builtin.apt_repository:
    repo: "deb [arch=amd64] https://download.docker.com/linux/ubuntu {{ ans>
    state: present

- name: Install Docker Engine
  apt:
    name: docker-ce
    state: present
    update_cache: yes
```

```
- name: Install Docker SDK for Python
  apt:
    name: python3-docker
    state: present

- name: Ensure Docker service is started and enabled on boot
  ansible.builtin.service:
    name: docker
    state: started
    enabled: yes
```

Output

```
vbearl@workstation:~/Planta_Lab11$ ansible-playbook installdocker.yml -K
BECOME password:

PLAY [Install Docker on all nodes] *********************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.56.108]
fatal: [192.168.56.106]: UNREACHABLE! => {"changed": false, "msg": "Failed to co
nnect to the host via ssh: ssh: connect to host 192.168.56.106 port 22: No route
 to host", "unreachable": true}
ok: [192.168.56.107]

TASK [Update APT package cache] ************************************************
changed: [192.168.56.108]
changed: [192.168.56.107]

TASK [Install prerequisite packages] *******************************************
ok: [192.168.56.108]
changed: [192.168.56.107]

TASK [Add Docker's official GPG key] *******************************************
ok: [192.168.56.108]
changed: [192.168.56.107]

TASK [Add Docker repository] ***************************************************
ok: [192.168.56.108]
changed: [192.168.56.107]

TASK [Install Docker Engine] ***************************************************
ok: [192.168.56.108]
changed: [192.168.56.107]

TASK [Install Docker SDK for Python] *******************************************
ok: [192.168.56.108]
changed: [192.168.56.107]

TASK [Ensure Docker service is started and enabled on boot] ********************
ok: [192.168.56.108]
ok: [192.168.56.107]

PLAY RECAP *********************************************************************
192.168.56.106             : ok=0    changed=0    unreachable=1    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.107             : ok=8    changed=6    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
192.168.56.108             : ok=8    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

CentOS Server

```
 GNU nano 7.2                    centosdocker.yml
---
- name: Install Docker on all CentOS nodes
  hosts: centos
  become: yes
  tasks:
    - name: Ensure required packages are installed
      ansible.builtin.yum:
        name:
          - yum-utils
          - device-mapper-persistent-data
          - lvm2
        state: present

    - name: Add Docker CE repository
      ansible.builtin.command:
        cmd: yum-config-manager --add-repo https://download.docker.com/linux/ce>
        creates: /etc/yum.repos.d/docker-ce.repo
```

```
 GNU nano 7.2                    centosdocker.yml

    - name: Install Docker Engine
      ansible.builtin.yum:
        name:
          - docker-ce
          - docker-ce-cli
          - containerd.io
        state: present

    - name: Install Docker SDK for Python
      yum:
        name: python3-docker
        state: present

    - name: Ensure Docker service is started and enabled
      ansible.builtin.service:
        name: docker
        state: started
        enabled: yes
```

This playbook installs the docker tool and starts its service. First, I installed prerequisite packages for the Docker installation, then I imported its repository and official GPG key. Next, I installed Docker, then started its service.

Output

```
vbearl@workstation:~/Planta_Lab11$ ansible-playbook centosdocker.yml -K
BECOME password:

PLAY [Install Docker on all CentOS nodes] ***************************************

TASK [Gathering Facts] **********************************************************
ok: [192.168.56.117]

TASK [Ensure required packages are installed] ***********************************
ok: [192.168.56.117]

TASK [Add Docker CE repository] *************************************************
ok: [192.168.56.117]

TASK [Install Docker Engine] ****************************************************
ok: [192.168.56.117]

TASK [Install Docker SDK for Python] ********************************************
changed: [192.168.56.117]

TASK [Ensure Docker service is started and enabled] *****************************
changed: [192.168.56.117]

PLAY RECAP **********************************************************************
192.168.56.117             : ok=6    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

3. Add a Docker group to your current user.

```
vbearl@workstation:~/Planta_Lab11$ sudo usermod -aG docker $USER
```

```
vbearl@server1:~$ sudo usermod -aG docker $USER
[sudo] password for vbearl:
```

```
[vbearl@centOS ~]$ sudo usermod -aG docker $USER
[sudo] password for vbearl:
```

To add the users to the Docker group, I ran the usermod command with options -aG to append, instead of replace the users' group. To apply the same to the other servers, I connected to them via ssh and ran the same command.

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

```
  GNU nano 7.2                                  dockerfile
FROM mariadb

WORKDIR /app

EXPOSE 80 3306

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

This is the dockerfile that I created, which creates an image of mariadb to be built and containerized.

5. Install and build the Dockerfile using Ansible.

```
  GNU nano 7.2                              build_dockerfile.yml
---
- name: Deploy web+DB container
  hosts: all
  become: yes
  vars:
    docker_ports:
      - "8080:80"
      - "3307:3306"
  tasks:

    - name: Copy Dockerfile to remote host
      copy:
        src: "~/Planta_Lab11/dockerfile"
        dest: "/tmp/Dockerfile"

    - name: Build Docker image
      community.docker.docker_image:
        name: web-db-server
        tag: latest
        build:
                                        [ Read 32 lines ]
```

```
        build:
          src: ~/Planta/Lab11/dockerfile
          path: /tmp
          dockerfile: dockerfile
          state: present

    - name: Run Docker container
      community.docker.docker_container:
        name: web-db
        image: "web_db:latest"
        state: started
        restart_policy: always
        published_ports: "{{ docker_ports }}"
```

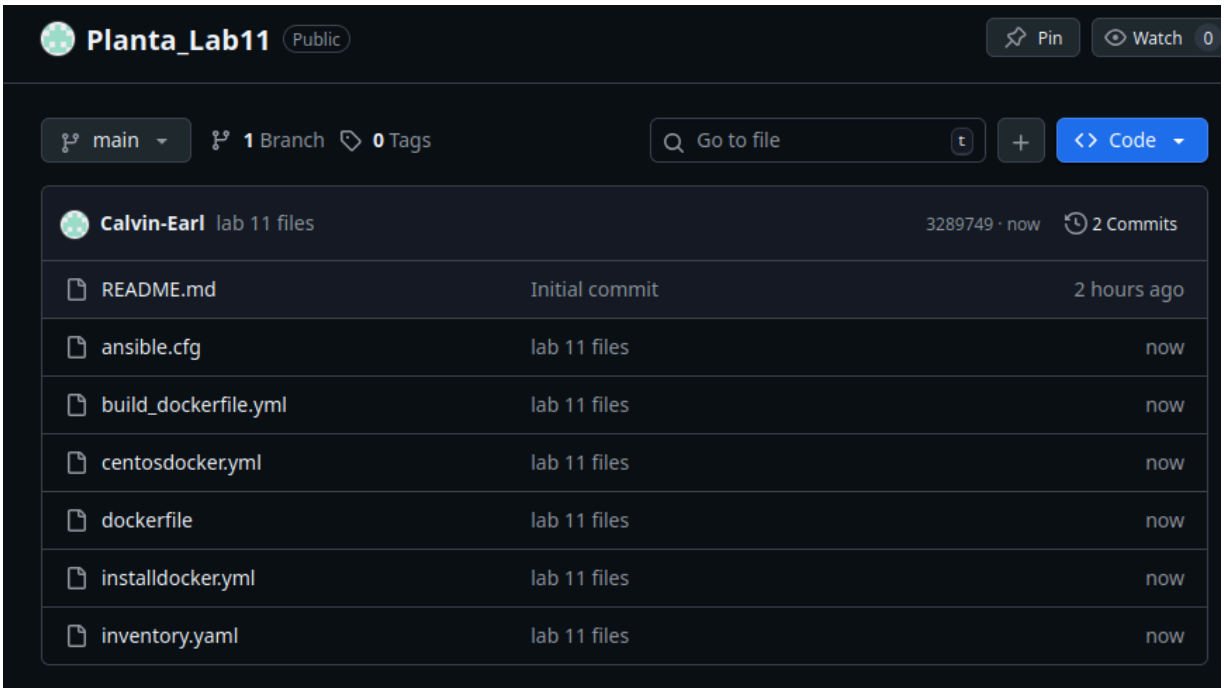
6. Add, commit and push it to your repository.

```
vbearl@workstation:~/Planta_Lab11$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ansible.cfg
        build_dockerfile.yml
        centosdocker.yml
        dockerfile
        installdocker.yml
        inventory.yaml
```

```
vbearl@workstation:~/Planta_Lab11$ git add .
vbearl@workstation:~/Planta_Lab11$ git commit -m "lab 11 files"
[main 3289749] lab 11 files
 6 files changed, 144 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 build_dockerfile.yml
 create mode 100644 centosdocker.yml
 create mode 100644 dockerfile
 create mode 100644 installdocker.yml
 create mode 100644 inventory.yaml
```

```
vbearl@workstation:~/Planta_Lab11$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 5 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.82 KiB | 1.82 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Calvin-Earl/Planta_Lab11.git
   09d813c..3289749  main -> main
```



https://github.com/Calvin-Earl/Planta_Lab11

Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?
    - Implementing containerization allows us to run apps on varying environments, as it bundles an application along with its dependencies in a container, allowing the container to run independently.

Conclusions:

In conclusion, I combined both Docker and Ansible to automate creating an image out of an existing application, building the image, and applying containerization to the application. Using the docker tool, I tried creating a mariadb image to be built and deployed, however, there are a lot of tasks to do before being able to deploy the docker container, such as installing the docker tool, adding users to the Docker group,

and building the image. To be able to do this, I used ansible to automate the said tasks.