Name: Dalena, Justin Miguel S.	Date Performed: Nov. 15, 2024
Course/Section: CPE212 - CPE31S21	Date Submitted: Nov. 15, 2024
Instructor: Engr. Robin Valenzuela	Semester and SY: 2024-2025

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

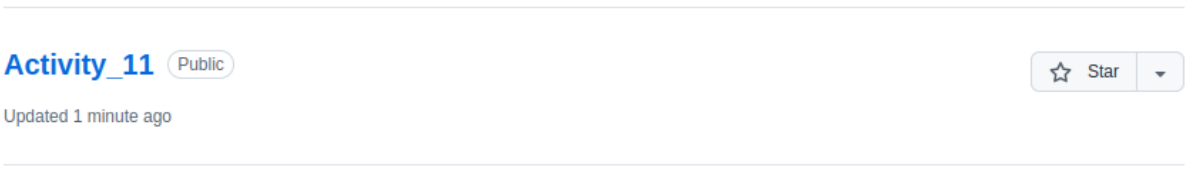
3. Tasks

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)

Create a new repository for this activity.

The image below shows the new repository and its name in my github account.



Install Docker and enable the docker socket.

using the command “sudo apt install docker.io” I have successfully installed docker to my system. Do take note that the commands used might change and be different from the one I used.

```
justin@workstation:~$ sudo apt install docker.io
[sudo] password for justin:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap 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 0 not upgraded.
Need to get 65.7 MB of archives.
After this operation, 292 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]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 bridge-utils amd64 1.5-15ubuntu1 [30.1 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 runc amd64 1.1.4-0ubuntu1~18.04.2 [3,822 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 containerd amd64 1.6.12-0ubuntu1~18.04.1 [31.5 MB]
Get:5 http://ph.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 docker.io amd64 20.10.21-0ubuntu1~18.04.3 [30.3 MB]
Get:6 http://ph.archive.ubuntu.com/ubuntu bionic/main amd64 ubuntu-fan all 0.12.10 [34.7 kB]
Fetched 65.7 MB in 45s (1,450 kB/s)
```

This part shows that I have enabled the docker in my system by using the commands “sudo systemctl start docker” and “sudo systemctl enable docker”. To properly check that the docker is running I used “sudo systemctl status docker” which will show the status of my docker in my workstation.

```
justin@workstation:~$ sudo systemctl start docker
justin@workstation:~$ sudo systemctl enable docker
justin@workstation:~$ 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 2024-11-15 07:52:33 PST; 1min 36s ago
     Docs: https://docs.docker.com
 Main PID: 3088 (dockerd)
    Tasks: 9
   CGroup: /system.slice/docker.service
           └─3088 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 15 07:52:30 workstation dockerd[3088]: time="2024-11-15T07:52:30.081530869+08:00" level=warning msg="Your kernel does not
Nov 15 07:52:30 workstation dockerd[3088]: time="2024-11-15T07:52:30.081536026+08:00" level=warning msg="Your kernel does not
Nov 15 07:52:30 workstation dockerd[3088]: time="2024-11-15T07:52:30.081539370+08:00" level=warning msg="Your kernel does not
Nov 15 07:52:30 workstation dockerd[3088]: time="2024-11-15T07:52:30.081654072+08:00" level=info msg="Loading containers: star
Nov 15 07:52:31 workstation dockerd[3088]: time="2024-11-15T07:52:31.851365344+08:00" level=info msg="Default bridge (docker0)
Nov 15 07:52:32 workstation dockerd[3088]: time="2024-11-15T07:52:32.537923763+08:00" level=info msg="Loading containers: done
Nov 15 07:52:33 workstation dockerd[3088]: time="2024-11-15T07:52:33.706353011+08:00" level=info msg="Docker daemon" commit="2
Nov 15 07:52:33 workstation dockerd[3088]: time="2024-11-15T07:52:33.706426061+08:00" level=info msg="Daemon has completed ini
Nov 15 07:52:33 workstation systemd[1]: Started Docker Application Container Engine.
Nov 15 07:52:33 workstation dockerd[3088]: time="2024-11-15T07:52:33.824414045+08:00" level=info msg="API listen on /var/run/d
```

Add to Docker group to your current user.

```
justin@workstation:~/Activity_11$ getent group docker
docker:x:127:
justin@workstation:~/Activity_11$ sudo usermod -aG docker $USER
[sudo] password for justin:
justin@workstation:~/Activity_11$ groups $USER
justin : justin adm cdrom sudo dip plugdev lpadmin sambashare docker
```

The images below shows the whole code and results of my ansible playbook. I have successfully installed docker.io to my Server and was able to add it to a docker group.

```
  GNU nano 2.9.3                                          Docker.yml

---
- name: Install Docker for remote servers
  hosts: Docker
  become: true
  tasks:
    - name: Update apt package
      apt:
        update_cache: true

    - name: Install docker
      apt:
        name: docker.io
        state: present

    - name: Run and enable docker
      service:
        name: docker
        state: started
        enabled: true

    - name: Adding current user to Docker Group
      user:
        name: "{{ ansible_user}}"
        groups: docker
        append: true
```

```
justin@workstation:~/Activity_11$ sudo nano Docker.yml
justin@workstation:~/Activity_11$ ansible-playbook --ask-become-pass Docker.yml -i Inventory
SUDO password:

PLAY [Install Docker for remote servers] ***************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [Server1]

TASK [Update apt package] ******************************************************************************
changed: [Server1]

TASK [Install docker] **********************************************************************************
changed: [Server1]

TASK [Run and enable docker] ***************************************************************************
ok: [Server1]

PLAY RECAP *********************************************************************************************
Server1                    : ok=4    changed=2    unreachable=0    failed=0
```

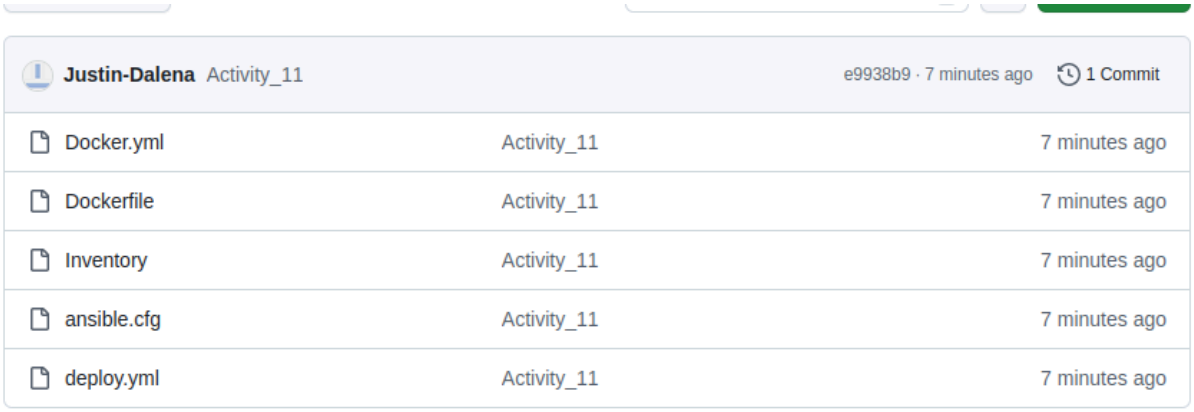
```
justin@workstation:~/Activity_11$ sudo nano Docker.yml
justin@workstation:~/Activity_11$ ansible-playbook --ask-become-pass Docker.yml -i Inventory
SUDO password:

PLAY [Install Docker for remote servers] ***************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [Server1]

TASK [Update apt package] ******************************************************************************
changed: [Server1]

TASK [Install docker] **********************************************************************************
ok: [Server1]

TASK [Run and enable docker] ***************************************************************************
ok: [Server1]

TASK [Adding current user to Docker Group] *************************************************************
changed: [Server1]

PLAY RECAP *********************************************************************************************
Server1                    : ok=5    changed=2    unreachable=0    failed=0
```

Create a Dockerfile to install web and DB server.

```
---
- hosts: Docker
  become: yes
  vars:
    ansible_python_interpreter: /usr/bin/python3  # Adjust this path based on your system
  tasks:
    - name: Install required packages for Debian
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
        state: present
      when: ansible_os_family == "Debian"

    - name: Install Docker from default repositories (Debian)
      apt:
        name: docker.io
        state: present
      when: ansible_os_family == "Debian"

    - name: Start and enable Docker
      service:
        name: docker
        state: started
        enabled: yes

    - name: Install Python pip for Debian
      apt:
        name: python3-pip
        state: present
      when: ansible_os_family == "Debian"

    - name: Install requests Python module
      pip:
        name: requests
        executable: pip3
        state: present

    - name: Install Docker Python module
      pip:
        name: docker
```

```
justin@workstation:~/Activity_11$ ansible-playbook --ask-become-pass deploy.yml -i Inventory
SUDO password:

PLAY [Docker] ******************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [Server1]

TASK [Install required packages for Debian] ************************************************************************
ok: [Server1]

TASK [Install Docker from default repositories (Debian)] ***********************************************************
ok: [Server1]

TASK [Start and enable Docker] *************************************************************************************
ok: [Server1]

TASK [Install Python pip for Debian] *******************************************************************************
ok: [Server1]

TASK [Install requests Python module] ******************************************************************************
ok: [Server1]

TASK [Install Docker Python module] ********************************************************************************
ok: [Server1]

TASK [Create Dockerfile directory] *********************************************************************************
ok: [Server1]

TASK [Copy Dockerfile to the control node] *************************************************************************
ok: [Server1]

TASK [Copy HTML files to the control node] **************************************************************************
ok: [Server1]

TASK [Build Docker image] ******************************************************************************************
fatal: [Server1]: FAILED! => {"changed": false, "msg": "Error building my-nginx-image - code: None, message: COPY fail
ed: file not found in build context or excluded by .dockerignore: stat nginx.conf: file does not exist, logs: ['Step 1
/5 : FROM nginx:alpine', '\\n', ' ---> a5967740120f\\n', 'Step 2/5 : RUN apk add --no-cache mysql mysql-client', '\\n'
, ' ---> Using cache\\n', ' ---> d4c7b5e56bd4\\n', 'Step 3/5 : COPY nginx.conf /etc/nginx/sites-available/default', '\
\n']"}
        to retry, use: --limit @/home/justin/Activity_11/deploy.retry

PLAY RECAP *********************************************************************************************************
Server1                    : ok=10   changed=0    unreachable=0    failed=1
```

Add, commit and push it to your repository.



Reflections:

Answer the following:

1. What are the benefits of implementing containerizations?
   - Implementing containerization offers numerous benefits, particularly in the context of software development, deployment, and operations. Containers package an application and its dependencies into a single, portable unit that can run consistently across different environments.

Conclusions:

Containerization offers portability, allowing applications to run consistently across different environments with all dependencies bundled together. It improves efficiency by being lightweight, enabling faster startup times and lower resource consumption compared to virtual machines. It enhances scalability and agility, supporting microservices architectures and seamless integration with CI/CD pipelines. Additionally, containerization provides better security, isolation, and simplified management, making it ideal for modern development and operations practices.