

<b>Name:</b> Bacong, El Cid A.	<b>Date Performed:</b> 24 October 2025
<b>Course/Section:</b> CPE212 - CPE31S4	<b>Date Submitted:</b> 24 October 2025
<b>Instructor:</b> Engr. Robin Valenzuela	<b>Semester and SY:</b> 1st Sem 2025-2026

### **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)**

The screenshot shows a GitHub repository page for 'Act11'. At the top, there's a navigation bar with icons for search, issues, pull requests, actions, projects, wiki, security, insights, and more. Below the navigation bar, the repository name 'Act11' is displayed along with its status as 'Public'. There are buttons for 'Pin', 'Watch' (0), 'Fork' (0), and 'Star' (0).  
  
The main content area has two sections: 'Start coding with Codespaces' and 'Add collaborators to this repository'.  
  
A large blue box titled 'Quick setup — if you've done this kind of thing before' contains instructions for setting up the repository. It includes options for 'Set up in Desktop' (selected), 'HTTPS', 'SSH', and a URL field with 'https://github.com/moussecake22/Act11.git'. It also provides links for creating a new file or uploading an existing file, and recommends including a README, LICENSE, and .gitignore.  
  
Below this, there are two code snippets for command-line setup:  
  
1. For creating a new repository:

```
echo "# Act11" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/moussecake22/Act11.git  
git push -u origin main
```

  
2. For pushing an existing repository:

```
git remote add origin https://github.com/moussecake22/Act11.git  
git branch -M main
```

*Figure 1: Creating a new repository*

```
changed: [192.168.56.104]

TASK [Add Docker's official GPG key] *****
changed: [192.168.56.104]

TASK [Add Docker repository] *****
changed: [192.168.56.104]

TASK [Install Docker Engine] *****
changed: [192.168.56.104]

TASK [Install python3-docker via apt] *****
changed: [192.168.56.104]

TASK [Ensure Docker service is started and enabled on boot] *****
ok: [192.168.56.104]

TASK [Add user to docker group] *****
ok: [192.168.56.104]

PLAY RECAP *****
192.168.56.104 : ok=9    changed=6    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

```
cidee@workstation:~/Act11$ docker --version
Docker version 24.0.2, build cb74dfc
```

Figure 2: Installing Docker and Adding user to docker group

```
cidee@workstation:~$ sudo systemctl start docker
cidee@workstation:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
cidee@workstation:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
    Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: e
    Active: active (running) since Fri 2025-10-24 14:56:50 +08; 1min 59s ago
      Docs: https://docs.docker.com
   Main PID: 13771 (dockerd)
     Tasks: 7
    CGroup: /system.slice/docker.service
            └─13771 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/contai

Oct 24 14:56:49 workstation dockerd[13771]: time="2025-10-24T14:56:49.954402426+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.193101619+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.199409204+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.568486403+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.646499624+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.735604944+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.735845187+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.740126709+
Oct 24 14:56:50 workstation dockerd[13771]: time="2025-10-24T14:56:50.799082909+
```

Figure 3: Enabling Docker Socket

```

👉 dockerfile
1 # Use Ubuntu as base image
2 FROM ubuntu:20.04
3
4 # Set environment variables to avoid interactive prompts
5 ENV DEBIAN_FRONTEND=noninteractive
6
7 # Update package repository and install web server (Apache) and DB server (MySQL)
8 RUN apt-get update && \
9     apt-get install -y \
10    apache2 \
11    mysql-server \
12    php \
13    libapache2-mod-php \
14    php-mysql \
15    && apt-get clean
16
17 # Configure Apache
18 RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
19
20 # Create a simple PHP test page
21 RUN echo "<?php phpinfo(); ?>" > /var/www/html/info.php
22 RUN echo "<h1>Web and DB Server Container</h1><p>Apache + MySQL running in Docker</p>" >
23
24 # Configure MySQL
25 RUN mkdir -p /var/run/mysqld && \
26     chown mysql:mysql /var/run/mysqld
27
28 # Initialize MySQL (minimal setup for demo)
29 RUN mysqld_safe & \
30     sleep 5 && \
31     mysql -e "CREATE DATABASE webappdb;" && \
32     mysql -e "ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';"
33     mysql -e "FLUSH PRIVILEGES;"
34
35 # Expose ports for web (80) and database (3306)
36 EXPOSE 80 3306
37
38 # Create startup script
39 RUN echo '#!/bin/bash\n\n' \
40     service mysql start\n\n' \
41     service apache2 start\n\n' \
42     tail -f /var/log/apache2/access.log' > /start.sh
43

```

Do you want to save the file?

Failed to save 'Dockerfile'.  
Please make sure you have write access to the directory.

Figure 4: Creating Dockerfile for Web and DB Server

```
build-docker-image.yml
1  ---
2  - name: Build custom Docker image with Web and DB server
3    hosts: all
4    become: yes
5
6  tasks:
7    - name: Copy Dockerfile to remote host
8      copy:
9        src: Dockerfile
10       dest: /tmp/Dockerfile
11
12    - name: Build Docker image from Dockerfile
13      command: docker build -t web-db-server /tmp/
14      args:
15        chdir: /tmp
16
17    - name: Run the web-db container
18      command: |
19        docker run -d \
20          --name web-db-container \
21          -p 8080:80 \
22          -p 3306:3306 \
23          web-db-server
24
25    - name: Verify container is running
26      command: docker ps
27      register: container_list
28
29    - name: Display running containers
30      debug:
31        var: container_list.stdout
32
```

*Figure 5: Install and Build Dockerfile using Ansible*

```
TASK [Update APT package cache] ****
changed: [192.168.56.104]

TASK [Install prerequisite packages] ****
ok: [192.168.56.104]

TASK [Add Docker's official GPG key] ****
ok: [192.168.56.104]

TASK [Add Docker repository] ****
ok: [192.168.56.104]

TASK [Install Docker Engine] ****
ok: [192.168.56.104]

TASK [Ensure Docker service is started and enabled] ****
ok: [192.168.56.104]

TASK [Add user to docker group] ****
ok: [192.168.56.104]
```

Figure 6: Execute Workflow

```
TASK [Check Docker containers] ****
changed: [192.168.56.104]

TASK [Display container status] ****
ok: [192.168.56.104] => {
    "containers.stdout": "CONTAINER ID     IMAGE      COMMAND      CREATED      STATUS
    PORTS      NAMES"
}

TASK [Check Docker images] ****
changed: [192.168.56.104]

TASK [Display available images] ****
ok: [192.168.56.104] => {
    "images.stdout": "REPOSITORY      TAG      IMAGE ID      CREATED      SIZE"
```

Figure 7: Execute Deployment

```
cidee@workstation:~/Docker_Act11$ git add *
cidee@workstation:~/Docker_Act11$ git commit -m Docker
[main fc266f9] Docker
 9 files changed, 302 insertions(+)
  create mode 100644 ansible.cfg
  create mode 100644 build-docker-image.yml
  create mode 100644 complete-workflow.yml
  create mode 100644 docker-install.yml
  create mode 100644 dockerfile
  create mode 100644 index.html
  create mode 100644 inventory.ini
  create mode 100644 nginx-deploy.yml
  create mode 100644 verify-deployment.yml
cidee@workstation:~/Docker_Act11$ git push origin main
Counting objects: 11, done.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (11/11), 3.40 KiB | 1.70 MiB/s, done.
Total 11 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To github.com:moussecake22/Docker_Act11.git
  4fe30ff..fc266f9  main -> main
```

*Figure 8: Saving my work on the New Repository I created*

*GitHub link: [https://github.com/moussecake22/Docker\\_Act11](https://github.com/moussecake22/Docker_Act11)*

#### **Reflections:**

Answer the following:

1. What are the benefits of implementing containerizations?

Containerization makes it easier for developers to run apps anywhere without problems. It keeps everything the app needs in one package, so it works the same on any computer or server. Containers are lightweight and fast, which makes testing and deployment quicker. They also keep apps separated for better security and fewer errors. Overall, containerization helps developers save time, avoid setup issues, and makes the whole process of building and updating apps smoother.

#### **Conclusions:**

In this activity, I learned how Docker makes app development faster and easier by packaging everything an app needs to run anywhere. Using Docker and Ansible helped me understand how automation and containerization support continuous delivery and make deploying updates more efficient.