Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Semester: V

Subject: DevOps Laboratory (DJS23OLOE501)

Experiment 1a

(Virtualization and Containerization with Docker)

Name : Soham Kishor Walam **Roll No : D102**

Aim: To Install Docker and Run Basic Containers.

Theory:

Virtualization:
Virtualization allows multiple operating systems to run on a single physical machine using a hypervisor. It abstracts hardware resources and creates **virtual machines (VMs)**, each with its own OS, libraries, and applications.
Containerization:
Containerization is a lightweight alternative to virtualization. Containers package the application code along with its dependencies, libraries, and configurations, but **share the host OS kernel**.

Introduction to Docker

Before Docker, there was a big issue faced by most developers whenever they created any code that code was working on that developer computer, but when they try to run that particular code on the server, that code was not working. This happened because apps need the right environment to run (like the right OS, libraries, and settings). If something was different on your computer vs. the server, things would break.
To solve this issue, Docker container comes. Docker is a popular open-source containerization platform. It enables developers to package, ship, and run applications in isolated environments called containers. Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container. Containers are light in weight, which makes them portable, and they are isolated from the underlying infrastructure and from each other's container. You can run the docker image as a docker container in any machine where docker is installed without depending on the operating system. There are two big pieces to Docker: The Docker Engine, which is the Docker binary that's running on your local machine and servers and does the work to run your software. The Docker Hub is a website and cloud service that makes it easy for everyone to share their docker images.

Key Components of Docker

The following are the some of the key components of Docker:

- Docker Engine: Docker Engine is a core part of docker, that handles the creation and management of containers.
- Docker Image: Docker Image is a read-only template that is used for creating containers, containing the application code and dependencies.
- Docker Hub: It is a cloud-based repository that is used for finding and sharing the container images.
- Dockerfile: It is a file that describes the steps to create an image quickly.
- Docker Registry: It is a storage distribution system for docker images, where you can store the images in both public and private modes.

What is a Dockerfile?

The Dockerfile uses DSL (Domain Specific Language) and contains instructions for generating a Docker image. Dockerfile will define the processes to quickly produce an image. While creating your application, you should create a Dockerfile in order since the **Docker daemon** runs all of the instructions from top to bottom.

The Dockerfile is the source code of the image.

(The Docker daemon, often referred to simply as "Docker," is a background service that manages Docker containers on a system.)

- It is a text document that contains necessary commands which on execution help assemble a Docker Image.
- Docker image is created using a Dockerfile.

What is Docker Image?

It is a file, comprised of multiple layers, used to execute code in a Docker container. They are a set of instructions used to create docker containers. Docker Image is an executable package of software that includes everything needed to run an application.

This image informs how a container should instantiate, determining which software components will run and how. Docker Container is a virtual environment that bundles application code with all the dependencies required to run the application. The application runs quickly and reliably from one computing environment to another.

What is Docker Container?

Docker container is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies. Docker Containers are runtime

instances of Docker images. Containers contain the whole kit required for an application, so the application can be run in an isolated way.

Common Docker Commands

- docker run: Start a new container,Container Management
- docker ps: List running containers,Monitoring
- docker build: Build an image from a Dockerfile,Image Management
- docker images: List images,Image Management
- docker stop: Stop a running container,Container Management
- docker-compose up: Start all services defined in docker-compose.yml,Multi-Container Management
- docker exec: Run a command inside a container,Container Management
- docker ps: Show only running containers
- docker ps -a: Show all containers (including stopped ones)

1. To install Docker on Linux or Windows.

Step 1: Update the system
sudo apt update
sudo apt upgrade -y

Step 2: Install required packages
sudo apt install apt-transport-https ca-certificates curl software-properties-common –y

It is commonly used in **Ubuntu/Debian-based Linux systems** to **prepare the system for installing packages from external repositories**, especially those over HTTPS.

Step 3: Add Docker's GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

Step 4: Set up the Docker repository
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Step 5: Install Docker Engine
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io -y

Step 6: Start and enable Docker
sudo systemctl start docker
sudo systemctl enable docker

Step 7: Verify Docker Installation
Run these commands to confirm Docker is installed and functioning:
docker --version
docker info

2. To run a simple container.

- **Run the Hello-World Container**

Use the following command:
sudo docker run hello-world

When you run the docker run hello-world command, Docker first checks if the hello-world image is available locally; if not, it automatically pulls the image from Docker Hub. Once the image is available, Docker creates and runs a container from it, executing the default program inside. If everything is set up correctly, the container displays a confirmation message indicating that Docker is installed and working properly.
Expected Output
Hello from Docker!
This message shows that your installation appears to be working correctly.

- **Start an Nginx web server**

sudo docker run -d -p 8080:80 nginx
-d: run in detached mode
-p 8080:80: maps your VM's port 8080 to container's port 80
nginx: image name (Docker will pull it if not present)

Check that it's running:
sudo docker ps

Open Port in GCP Firewall
If you're using port 8080, you must open it:

1. Go to VPC network > Firewall rules in GCP.

2. Click “Create firewall rule”.
3. Set:
    - Name: allow-http-8080
    - Targets: All instances in the network
    - Source IP ranges: 0.0.0.0/0
    - Protocols/ports: TCP: 8080
4. Save.

Access the Running App
Find your VM’s external IP from the Google Cloud Console and open in a browser:
http://<your-vm-ip>:8080

Verify Running Containers
sudo docker ps

View All Containers (Including Exited Ones)
sudo docker ps -a

Remove the Hello-World Container
First, find the container ID from docker ps -a, then run:
sudo docker rm <container_id>

Lab experiment to be performed in this session:

1. Run an Ubuntu Container and Display Date.

```
walamsoham@exp1:~$ sudo docker run ubuntu date
Mon Aug 11 14:25:23 UTC 2025
```

2. Perform the following tasks using appropriate Docker commands:

    - List all currently running containers.

```
walamsoham@exp1:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS              PORTS     NAMES
56de386fe07b   ubuntu    "bash"    About a minute ago   Up About a minute             ubuntu-date
```

    - List all containers, including those that are stopped.

```
walamsoham@exp1:~$ sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS                          PORTS
    NAMES
34cee3821a43   ubuntu        "date"                   About a minute ago   Exited (0) About a minute ago
    priceless_newton
56de386fe07b   ubuntu        "bash"                   3 minutes ago        Exited (129) 50 seconds ago
    ubuntu-date
c6fb30a6aa35   nginx         "/docker-entrypoint.…"   10 hours ago         Exited (0) 10 hours ago
    sweet_moser
68a22ddefe6e   hello-world   "/hello"                 10 hours ago         Exited (0) 10 hours ago
    priceless_gates
```

- List the most recently created container, regardless of its state (running or exited).

```
walamsoham@exp1:~$ sudo docker ps -l
CONTAINER ID   IMAGE     COMMAND   CREATED              STATUS                          PORTS     NAMES
34cee3821a43   ubuntu    "date"    About a minute ago   Exited (0) About a minute ago             priceless_newt
on
```

- Pause a running container and then unpause it.

```
walamsoham@exp1:~$ sudo docker pause 34cee3821a43
Error response from daemon: container 34cee3821a431c38b1f4a1a290d591318b5f01068f99d1094a51487e2085110e is not ru
nning
walamsoham@exp1:~$ sudo docker unpause 34cee3821a43
Error response from daemon: Container 34cee3821a431c38b1f4a1a290d591318b5f01068f99d1094a51487e2085110e is not pa
used
```

- Remove all stopped containers in one go.

```
walamsoham@exp1:~$ sudo docker container prune -f
Deleted Containers:
34cee3821a431c38b1f4a1a290d591318b5f01068f99d1094a51487e2085110e
56de386fe07b453b4b779b9a2d59a566e10fc5407d3a6168670ca0f2ad3af2af
c6fb30a6aa35a21221aa36f75d00d46cf8f3aa7bbbfe979d86b6056351f92742
68a22ddefe6ea0d370bac28d74ec9ed8a2d6a11d22360bee14f07c167c8fea04

Total reclaimed space: 1.121kB
```

Document the output (terminal screenshots) for each command.