EXPERIMENT NO.: 6

🔍 Aim: Explore Docker commands for content management.

📝 What is Docker?

Docker is a tool designed to make it easier to **create**, **deploy**, and **run applications** using **containers**. A **container** is like a lightweight virtual machine—it has its own filesystem, processes, and network, but shares the host system's OS.

⚙️ Before You Begin

1. ✅ **Install Docker**
   Make sure Docker is installed on your system.
   - On Windows/Mac: Install Docker Desktop
   - On Linux: Use your package manager (e.g., sudo apt install docker.io)
2. ✅ **Check if Docker is working**
   Open terminal or command prompt:
3. docker --version

Example output:

Docker version 24.0.5, build abc123

🔧 Step-by-Step Docker Commands with Examples

---

1. 🚀 docker run – Run a new container

docker run --name mycontainer -it ubuntu:16.04 /bin/bash

📌 **What it does:**

- --name mycontainer: Names the container "mycontainer"
- -it: Interactive mode + Terminal (useful for user input)
- ubuntu:16.04: Uses Ubuntu 16.04 image
- /bin/bash: Starts a bash shell inside the container

📩 **Result:** You are now **inside the container's shell**. Try:

```
ls
```

To exit the container shell, type:

```
exit
```

---

2. 🟢 docker start – Start a stopped container

```
docker start mycontainer
```

📌 **What it does:**

- Starts a container named mycontainer that was previously stopped or exited.

✅ To **attach** and go inside again:

```
docker attach mycontainer
```

---

3. 🔴 docker stop – Stop a running container

```
docker stop mycontainer
```

📌 **What it does:**

- Gracefully **stops** the container named mycontainer.

✅ Use this before removing or restarting a container.

---

4. 🗑️ docker rm – Remove a container

```
docker rm mycontainer
```

📌 **What it does:**

- **Deletes** the container named mycontainer.

🚨 The container must be stopped before removing.
If it's running, stop it first:

```
docker stop mycontainer
```

```
docker rm mycontainer
```

---

5. 📋 docker ps – List running containers

docker ps

📌 **What it does:**

- Shows all **currently running containers**.
- Columns include:
  - Container ID
  - Image name
  - Status
  - Ports

📌 To see **all containers** (including stopped):

docker ps -a

---

6. 🖼️ docker images – List available images

docker images

📌 **What it does:**

- Lists **all images** available locally on your machine.
- Columns include:
  - REPOSITORY
  - TAG
  - IMAGE ID
  - SIZE

---

7. ⬇️ docker pull – Download an image from Docker Hub

docker pull ubuntu:16.04

📌 **What it does:**

- Downloads the **Ubuntu 16.04 image** from Docker Hub to your system.

💡 Useful if the image is not already on your system.

---

8. ⬆️ docker push – Upload an image to Docker Hub

docker push myimage

📌 **What it does:**

- Pushes a **local image** named myimage to your Docker Hub account.

⚠️ **Before pushing:**

1. You must be **logged in** to Docker:
2. docker login
3. Your image must be **tagged correctly**, e.g.:
4. docker tag myimage your_dockerhub_username/myimage
5. docker push your_dockerhub_username/myimage

✅ Quick Recap Table

Command	Use
docker run	Run a new container
docker start	Start an existing container
docker stop	Stop a running container
docker rm	Remove a container
docker ps	List running containers
docker ps -a	List all containers (running + stopped)
docker images	List downloaded images
docker pull	Download image from Docker Hub
docker push	Upload image to Docker Hub

🧪 Try This Simple Practice Flow

docker pull ubuntu:16.04

```
docker run --name testubuntu -it ubuntu:16.04 /bin/bash
# Inside the container:
apt update
exit

docker ps -a
docker start testubuntu
docker attach testubuntu
# Inside again:
echo "Hello from inside container"
exit

docker stop testubuntu
docker rm testubuntu
docker images
```

📌 **Tips for Beginners**

- Containers are **temporary environments**—you can recreate them easily.
- Images are **templates**; containers are the **running instances**.
- Always exit from a container before using stop or rm.

👉 **Create a Docker image** and **run it inside a Docker container**.

🔥 **Goal:**

Containerize the application (HTML or Flask) using Docker on Windows.

We'll cover **both methods** (HTML only ✅ and Flask ✅) so you can choose either.

🔹 **Option 1: Dockerize Pure HTML Web App**

📁 **Folder structure:**

```
hello-html-docker/
    ├── index.html
    └── Dockerfile
```

📝 **Step 1: Create Files**

1. Make a folder:
   hello-html-docker
2. Inside it, create:
   - index.html (your Hello World HTML page)
   - Dockerfile

✍️ **index.html**

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello from Docker</title>
</head>
<body>
  <h1>Hello World from HTML in Docker!</h1>
</body>
```

</html>

✍️ **Dockerfile**

```
# Use a lightweight Nginx image
FROM nginx:alpine

# Copy HTML file to the default nginx HTML directory
COPY index.html /usr/share/nginx/html/index.html
```

🖥️ **Step 2: Build and Run Docker Image**

Open **Command Prompt** in the hello-html-docker folder:

```
cd path\to\hello-html-docker

# Build the Docker image
docker build -t html-hello .

# Run the Docker container
docker run -d -p 8080:80 html-hello
```

🌐 **Open in Browser:**

http://localhost:8080

🔹 **Option 2: Dockerize Flask Web App**

📁 **Folder structure:**

```
hello-flask-docker/
        ├── app.py
        ├── requirements.txt
        ├── Dockerfile
```

```
└── templates/
    └── index.html
```

✍️ **app.py**

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template("index.html")

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

✍️ **templates/index.html**

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask Docker</title>
</head>
<body>
    <h1>Hello World from Flask in Docker!</h1>
</body>
</html>
```

✍️ **requirements.txt**

```
flask
```

✍️ **Dockerfile**

```
# Use base Python image
FROM python:3.9-slim

# Set work directory
WORKDIR /app

# Copy files
COPY . .

# Install dependencies
RUN pip install -r requirements.txt

# Expose port
EXPOSE 5000

# Run the app
CMD ["python", "app.py"]
```

💻 **Step 2: Build and Run Docker Image**

Open Command Prompt in hello-flask-docker folder:

```
cd path\to\hello-flask-docker

# Build the Docker image
docker build -t flask-hello .

# Run the Docker container
```

docker run -d -p 5000:5000 flask-hello

🌐 **Open in Browser:**

http://localhost:5000

✅ **Summary of Docker Commands**

Task	Command
Build Docker image	docker build -t imagename .
Run container from image	docker run -d -p host_port:container_port imagename
List running containers	docker ps
Stop a container	docker stop container_id
Remove a container	docker rm container_id
Remove an image	docker rmi imagename

📌 **Example Image & Container Commands (Windows)**

HTML

cd Desktop\hello-html-docker

docker build -t html-hello .

docker run -d -p 8080:80 html-hello

Flask

cd Desktop\hello-flask-docker

docker build -t flask-hello .

docker run -d -p 5000:5000 flask-hello

🔧 How to Create a Dockerfile in VS Code (Windows)

🧰 Step 1: Set Up Folder

Create a new folder on your Desktop:

hello-html-docker

Inside this folder, you will place:

- index.html (your HTML file)
- Dockerfile (your Docker build script)

📥 Step 2: Open the Folder in VS Code

1. Open **Visual Studio Code**
2. Click on File → Open Folder… → select your hello-html-docker folder

---

📝 Step 3: Create Files in VS Code

1. Create a new file:

File Name: index.html
Right-click on the folder → New File → name it index.html

🔽 Paste this code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello from Docker</title>
</head>
<body>
  <h1>Hello World from HTML in Docker!</h1>
</body>
</html>
```

---

2. Create the Dockerfile:

File Name: Dockerfile
Right-click → New File → name it exactly Dockerfile (no extension)

🔽 Paste this code:

```
# Use official Nginx web server image

FROM nginx:alpine


# Copy our HTML file into the default nginx folder

COPY index.html /usr/share/nginx/html/index.html
```

✅ Step 4: Save and Close VS Code

Use **Ctrl + S** to save all files. Now you're ready to build the Docker image!

🚀 Step 5: Run Docker Commands from Command Prompt (Windows)

1. Open Command Prompt (Windows + R → cmd)
2. Go to your project folder:

```
cd Desktop\hello-html-docker
```

3. Build the Docker image:

```
docker build -t html-hello .
```

✅ This creates an image named html-hello

4. Run the container:

```
docker run -d -p 8080:80 html-hello
```

✅ Now open your browser and go to:

http://localhost:8080

You'll see your **Hello World from HTML in Docker!** message 🎉

💡 Tips

- Always **name your file exactly**: Dockerfile (no .txt or .docker)
- You can view the running container:
- docker ps
- You can stop the container using:
- docker stop <container_id>

✅ **Summary**

Task	How to Do It
Create Dockerfile	Use VS Code in your project folder
Write Dockerfile content	Paste code based on app (HTML/Flask)
Build image	docker build -t image-name .
Run container	docker run -d -p 8080:80 image-name
View in browser	http://localhost:8080 or :5000 (Flask)
