**Module 5: Continuous deployments**

**1. Docker Introduction**

**What is Docker?**  
Docker is a platform designed to help developers build, ship, and run applications inside lightweight, portable containers.

**Why Use Docker?**

* **Containerization**: Docker packages your application along with all its dependencies, libraries, and configuration files into a single unit called a container.
* **Portability**: Docker containers can run on any system that supports Docker, making deployment easy and consistent across different environments.
* **Speed**: Containers start quickly and use fewer resources than traditional virtual machines.
* **Consistency**: Ensures the application runs the same in development, testing, and production.

**What is a Container?**

A container is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, system tools, and libraries. Containers share the host system's OS kernel but remain isolated from each other.

**Main Components of Docker**

* **Dockerfile** – A text file with a set of instructions to create a Docker image.
* **Image** – A snapshot of the application and its environment.
* **Container** – A running instance of a Docker image.
* **Docker Hub** – A cloud-based registry where Docker images can be stored and shared.

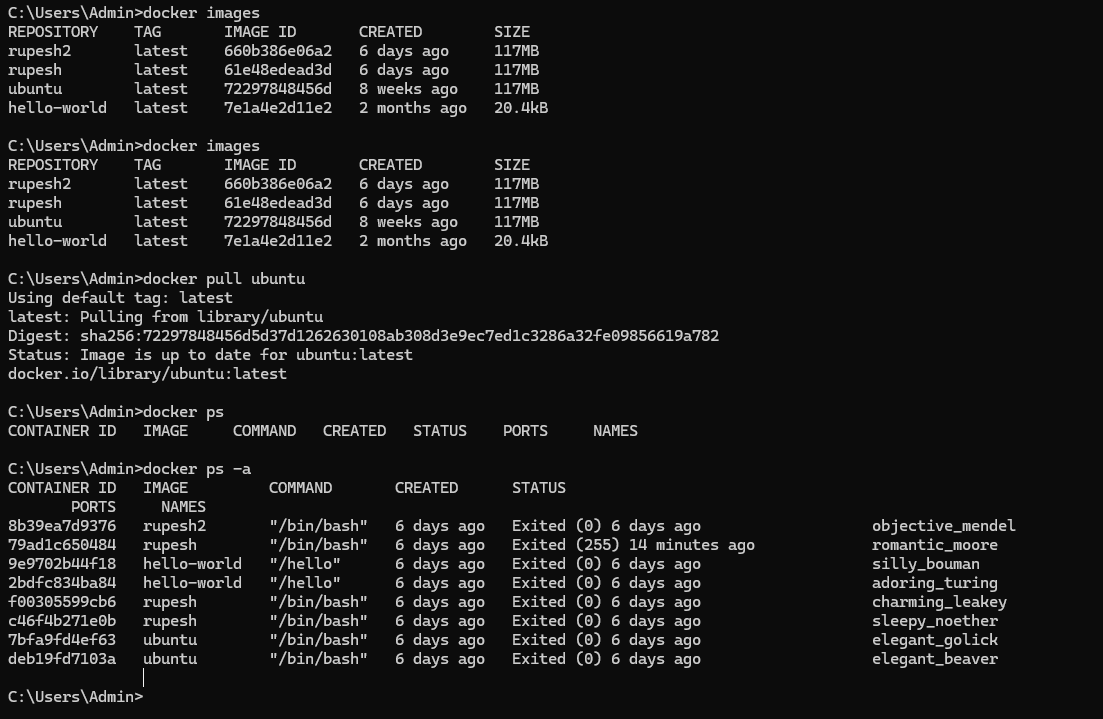
**Basic Docker Commands**

* docker build – Builds a Docker image from a Dockerfile.
* docker run – Runs a container based on an image.
* docker ps – Lists running containers.
* docker stop – Stops a running container.

**2. Explore docker commands for content management**

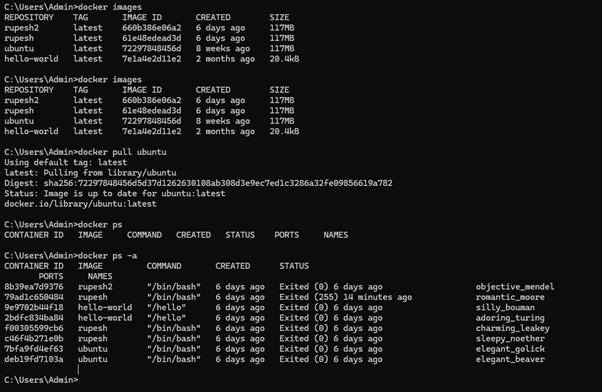
**docker images**

Shows a list of all the Docker images stored on your system.  
**Usage:** To see available images you can use to create containers.



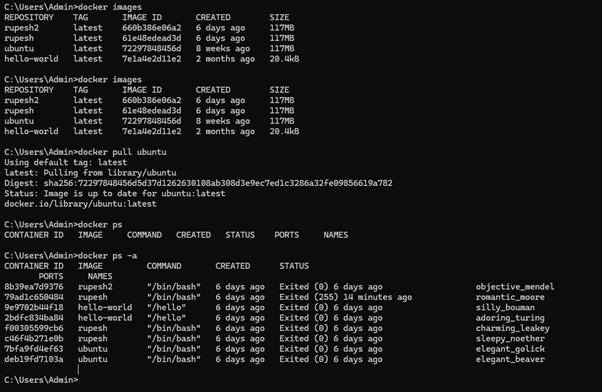
**docker pull ubuntu**

Downloads the **Ubuntu** image from Docker Hub to your local system.  
**Use case:** To get a base image (like Ubuntu) for creating containers.



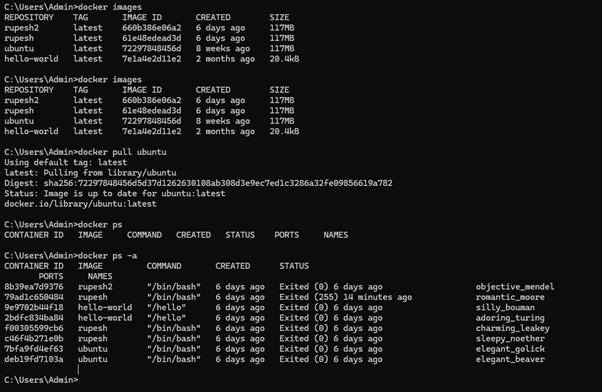
**docker ps**

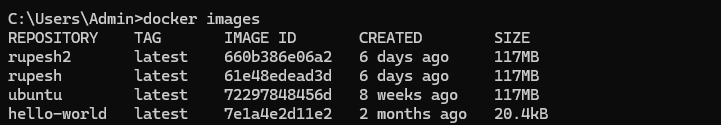
Lists only the containers that are currently running.  
**Use case:** To check which containers are active at the moment.



**docker ps -a**

Lists all containers, including those that are stopped.  
**Use case:** To see both running and exited containers.





**docker run -it -d <imageId>**

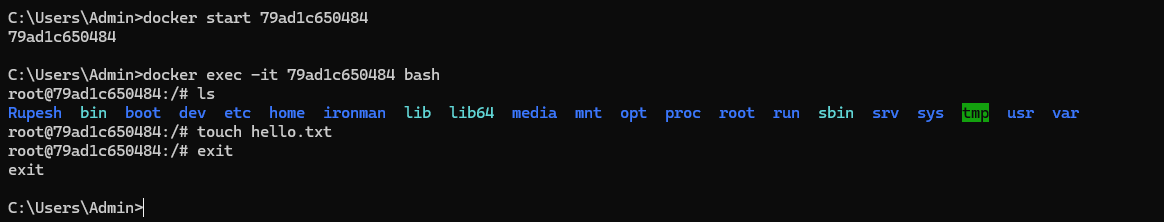
Runs a container from the specified image:

* -i: interactive mode
* -t: allocates a pseudo-TTY (terminal)
* -d: detached mode (runs in the background)  
  **Use case:** To start a container in the background with terminal access.



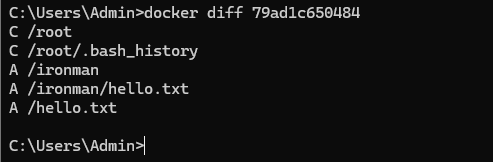
**docker start <container\_id>**

Starts a previously stopped container.  
**Use case:** To resume a container without creating a new one.



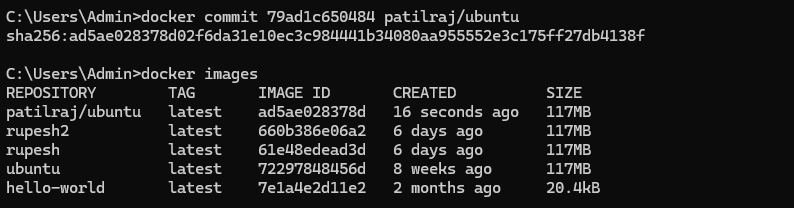
**docker diff <container\_id>**

Shows the changes made to the container’s file system (added, modified, or deleted files).  
**Use case:** To inspect what has changed inside a container.



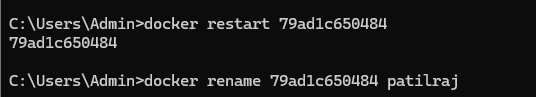
**docker commit <container\_id> <new\_image\_name>**

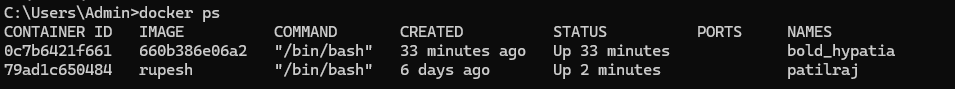
Creates a new image from the current state of a container.  
**Use case:** To save the modified container as a new image.



**docker restart <container\_id>**

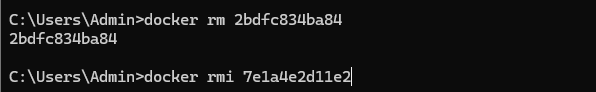
Stops and then starts the container again.  
**Use case:** To refresh the container or apply configuration changes.





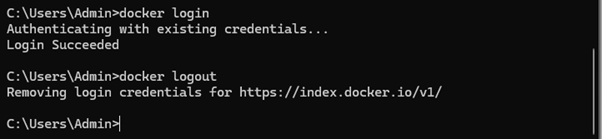
**docker rm <container\_id>**

Removes a stopped container.  
**Use case:** To clean up containers you no longer need.



**docker login**

Logs into your Docker Hub account via the terminal.  
**Use case:** Required before pushing images to your Docker Hub repository.



**docker stop <container\_id>**

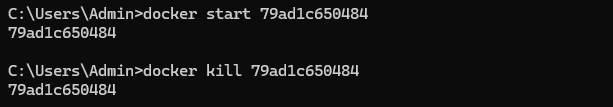
Stops a running container.  
**Use case:** To safely terminate a container.



**docker kill <container\_id>**

**Description:**  
This command **immediately stops** a running container by sending a **SIGKILL** signal to the main process inside the container.

**Use case:**  
Use docker kill when you need to forcefully stop a container **right away**, especially if it's not responding to a normal docker stop (which uses a gentler SIGTERM signal).



**docker push <image\_name>**

Uploads a Docker image to Docker Hub.  
**Use case:** To share your image with others or store it online for deployment.

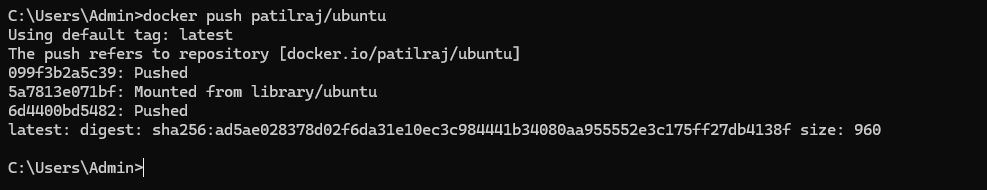
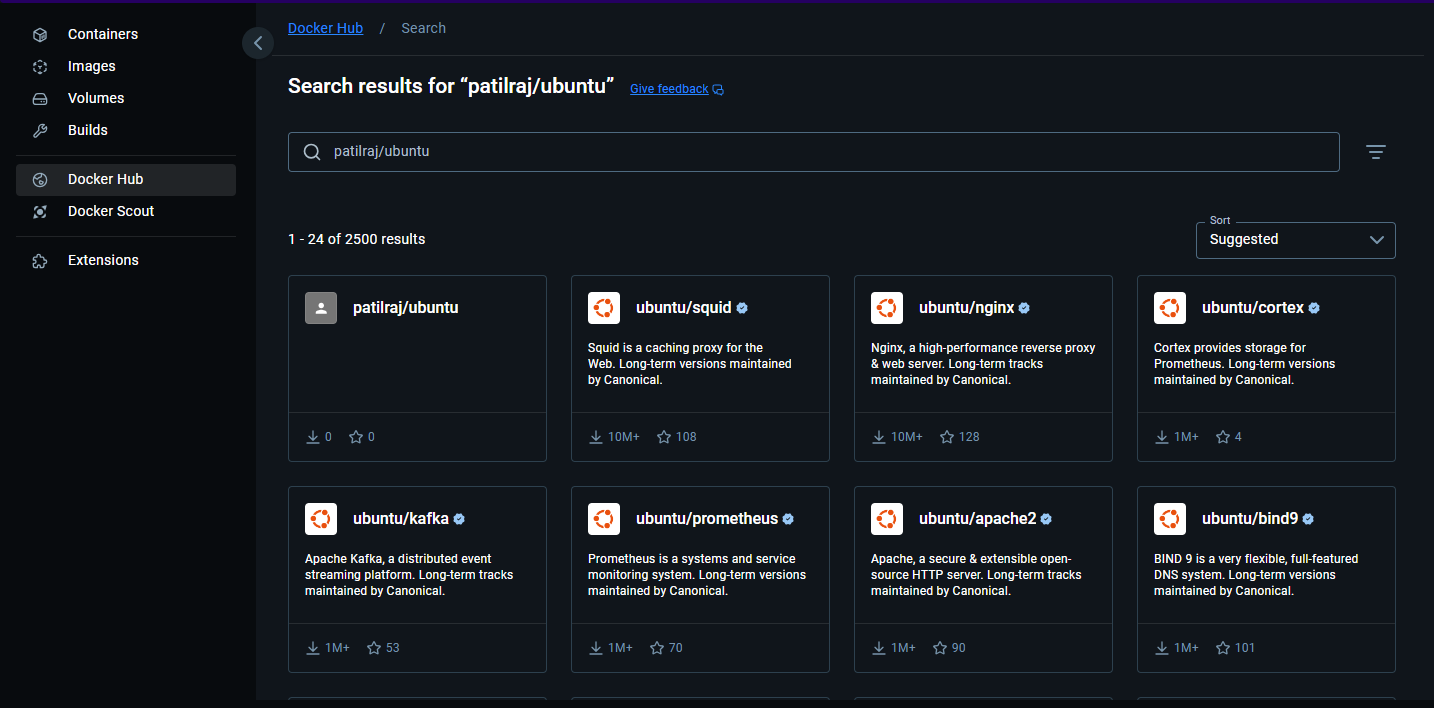
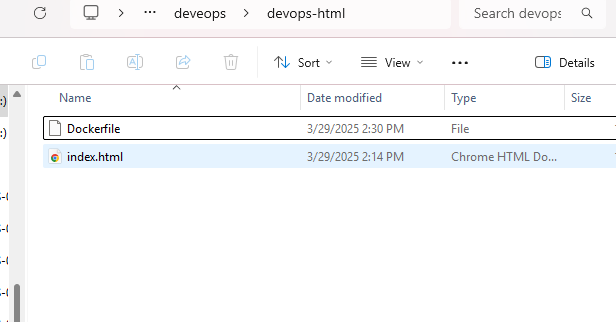


Image is uploaded on docker hub



**3. Develop a simple containerized application using Docker**

**Docker app**

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>This is a Docker Html app </h1>

<p>This is a paragraph.</p>

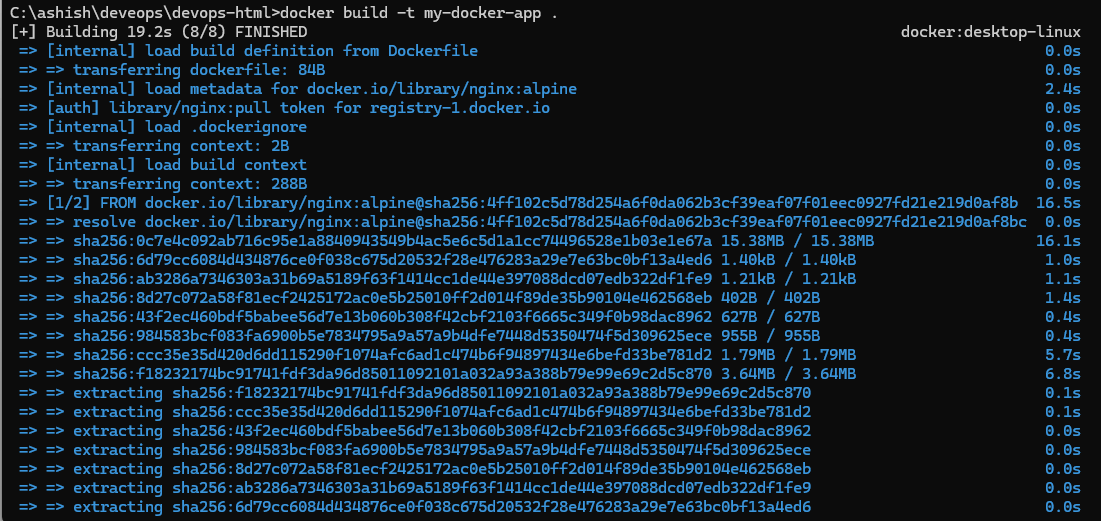
</body>

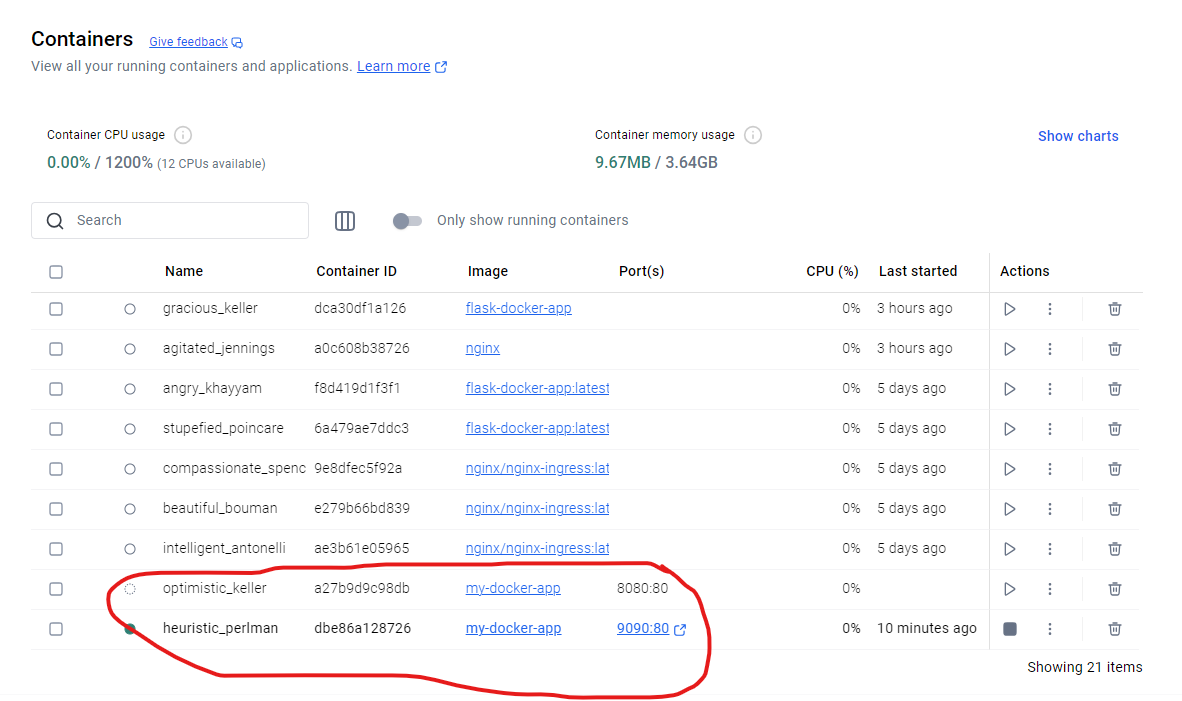
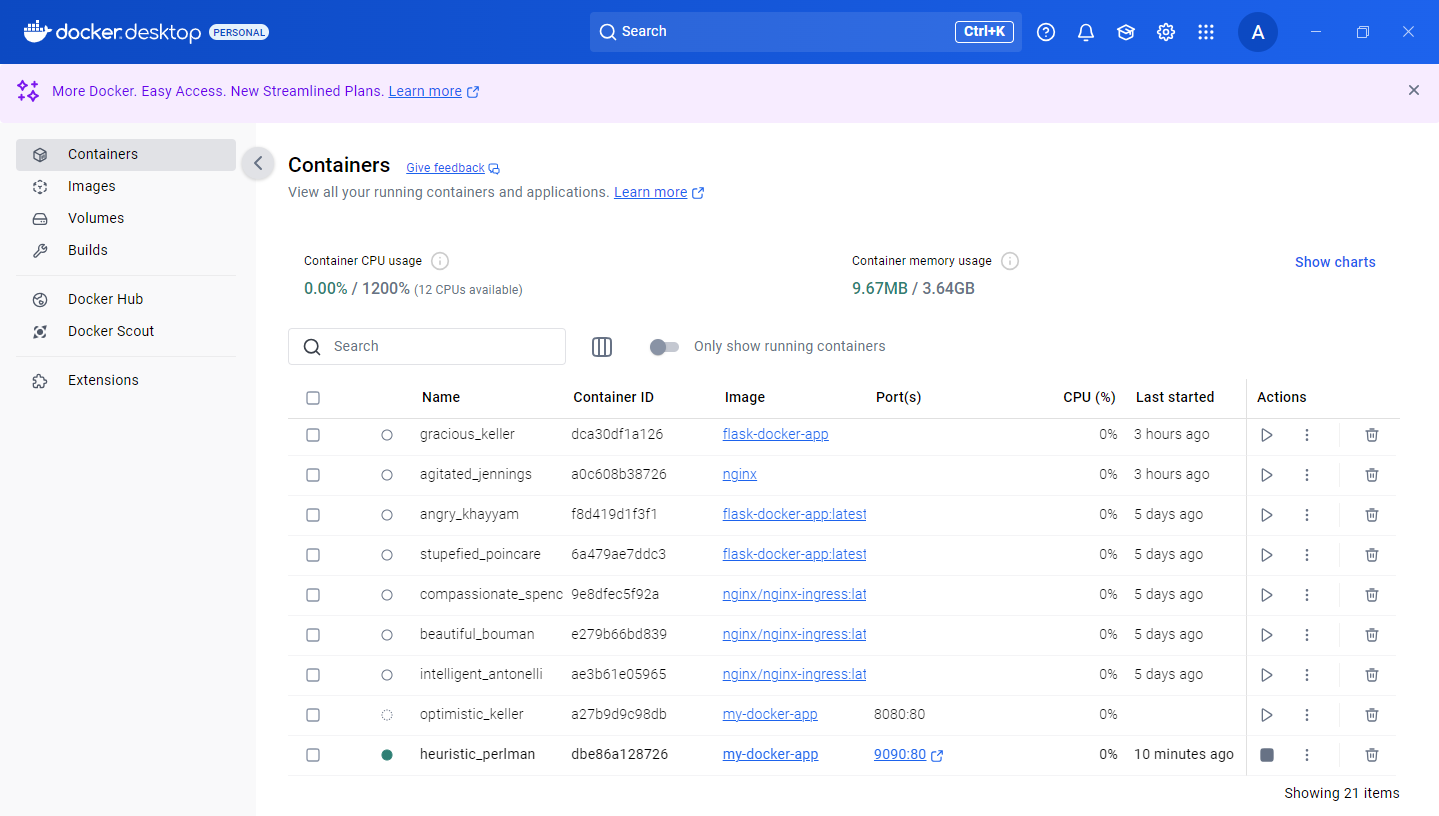
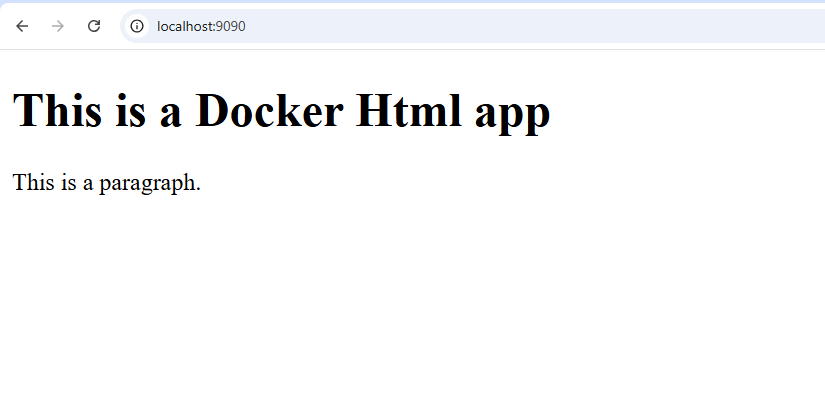
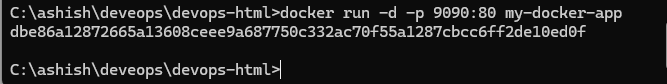
</html>

**Dokerfile**

FROM nginx:alpine

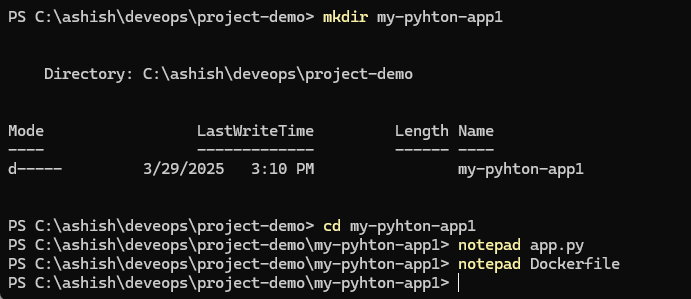
COPY . /usr/share/nginx/html





**4. Build, deploy and manage web/software application on Docker Engine.**

**python**



**app.py**

print("hello docker file")

**Dockerfile**

FROM python:3.10

# Set the working directory inside the container

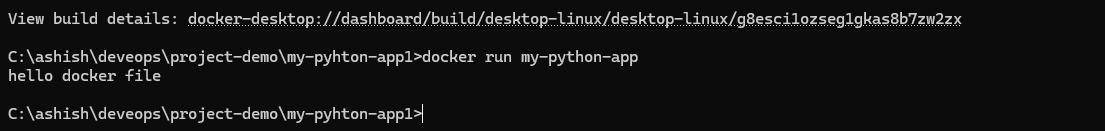
WORKDIR /app

# Copy the application code to the container's /app directory

COPY app.py /app/

# Run the Python application

CMD ["python", "app.py"]



**App.py**

import time

while True:

print("hello docker file")

time.sleep(2)

