Terna Engineering College

## Computer Engineering Department

Program: Sem VI

## Course: Cloud Computing Lab (CSL605) PART A

(PART A: TO BE REFFERED BY STUDENTS)

Experiment No.9

**Title**: To study and Implement Containerization using Docker

* 1. **Objective**: To know the basic differences between Virtual machine and Container. It involves demonstration of creating, finding, and building, installing, and running Linux/Windows application containers inside local machine or cloud platform.

## Prerequisite:

Knowledge of Networking, Distributed Computing and knowledge of Software architectures.

## Objective:

Objectives this experiment is to provide students an overview AWS, its Features and Services.

## Outcome: (LO3)

After successful completion of this experiment student will be able to run Linux/ windows application containers inside local machine.

## Theory :

**Docker** is the containerization platform that is used to package your application and all its dependencies together in the form of containers to make sure that your application works seamlessly in any environment which can be developed or tested or in production.



## Containerization:

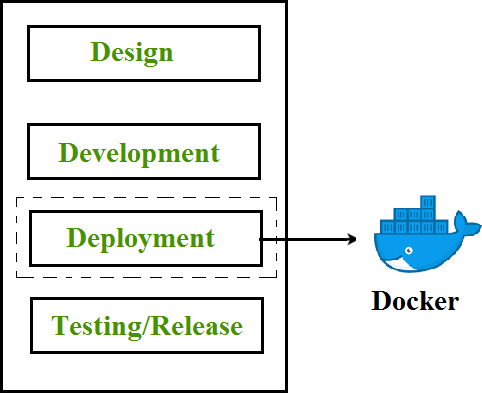
Containerization is OS-based virtualization that creates multiple virtual units in the userspace, known as Containers. Containers share the same host kernel but are isolated from each other through private namespaces and resource control mechanisms at the OS level. Container-based Virtualization provides a different level of abstraction in terms of virtualization and isolation when compared with hypervisors. Hypervisors use a lot of hardware which results in overhead in terms of virtualizing hardware and virtual device drivers. A full operating system (e.g -Linux, Windows) runs on top of this virtualized hardware in each virtual machine instance. But in contrast, containers implement isolation of processes at the operating system level, thus avoiding such overhead. These containers run on top of the same shared operating system kernel of the underlying host machine and one or more processes can be run within each container. In containers you don’t have to pre-allocate any RAM, it is allocated dynamically during the creation of containers while in VMs you need to first pre-allocate the memory and then create the virtual machine. Containerization has better resource utilization compared to VMs and a short boot-up process. It is the next evolution in virtualization.

Containers can run virtually anywhere, greatly easy development and deployment: on Linux, Windows, and Mac operating systems; on virtual machines or bare metal, on a developer’s machine or in data centers on-premises; and of course, in the public cloud. Containers virtualize CPU, memory, storage, and network resources at the OS level, providing developers with a sandboxed view of the OS logically isolated from other applications. Docker is the most popular open-source container format available and is supported on Google Cloud Platform and by Google Kubernetes Engine.

After you understand the concept of Docker, now let’s get into the implementation. There

are several steps that we will do:

* + 1. Install Docker
    2. Create a file called Dockerfile
    3. Build the image
    4. Run the image



Useful link:

<https://docs.docker.com/>

## PART B

**(PART B: TO BE COMPLETED BY STUDENTS)**

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the ERP or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no ERP access available)***

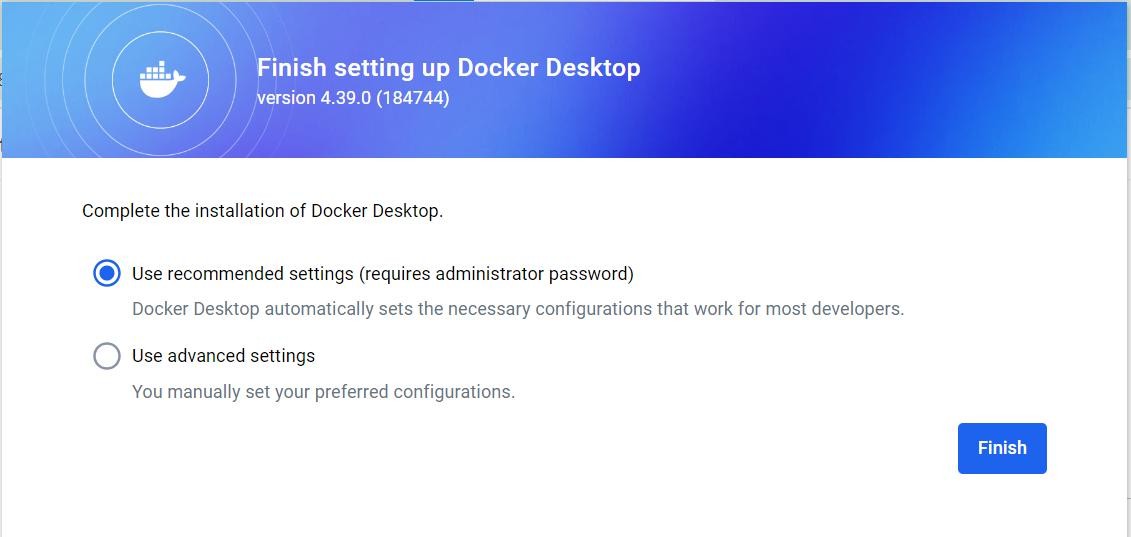
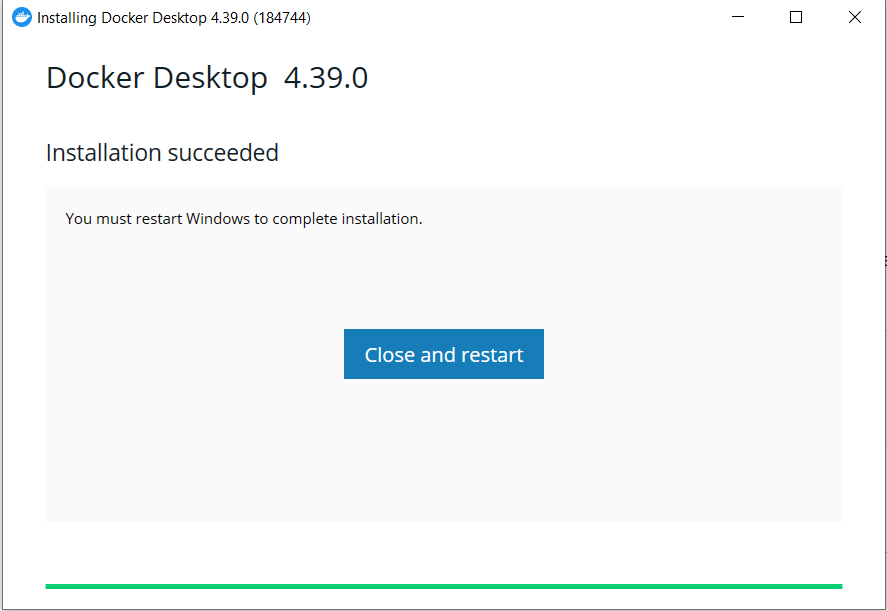
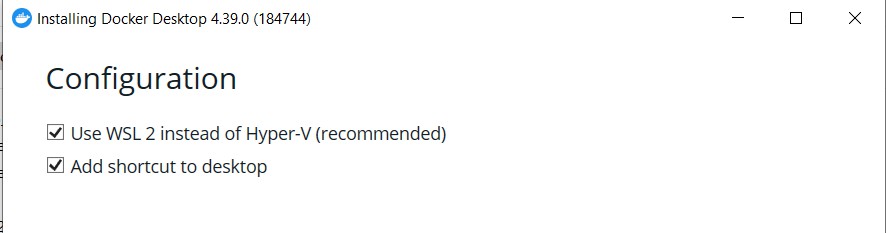
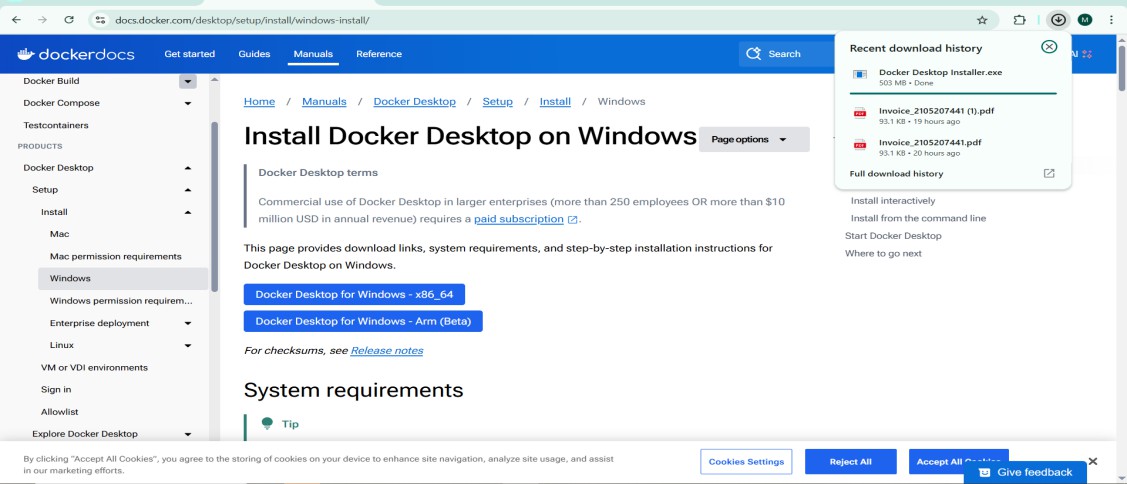
|  |  |
| --- | --- |
| Roll No. B30 | Name: Pranjal Bhatt |
| Class : TE B Comps | Batch : B2 |
| Date of Experiment: | Date of Submission: |
| Grade : |  |

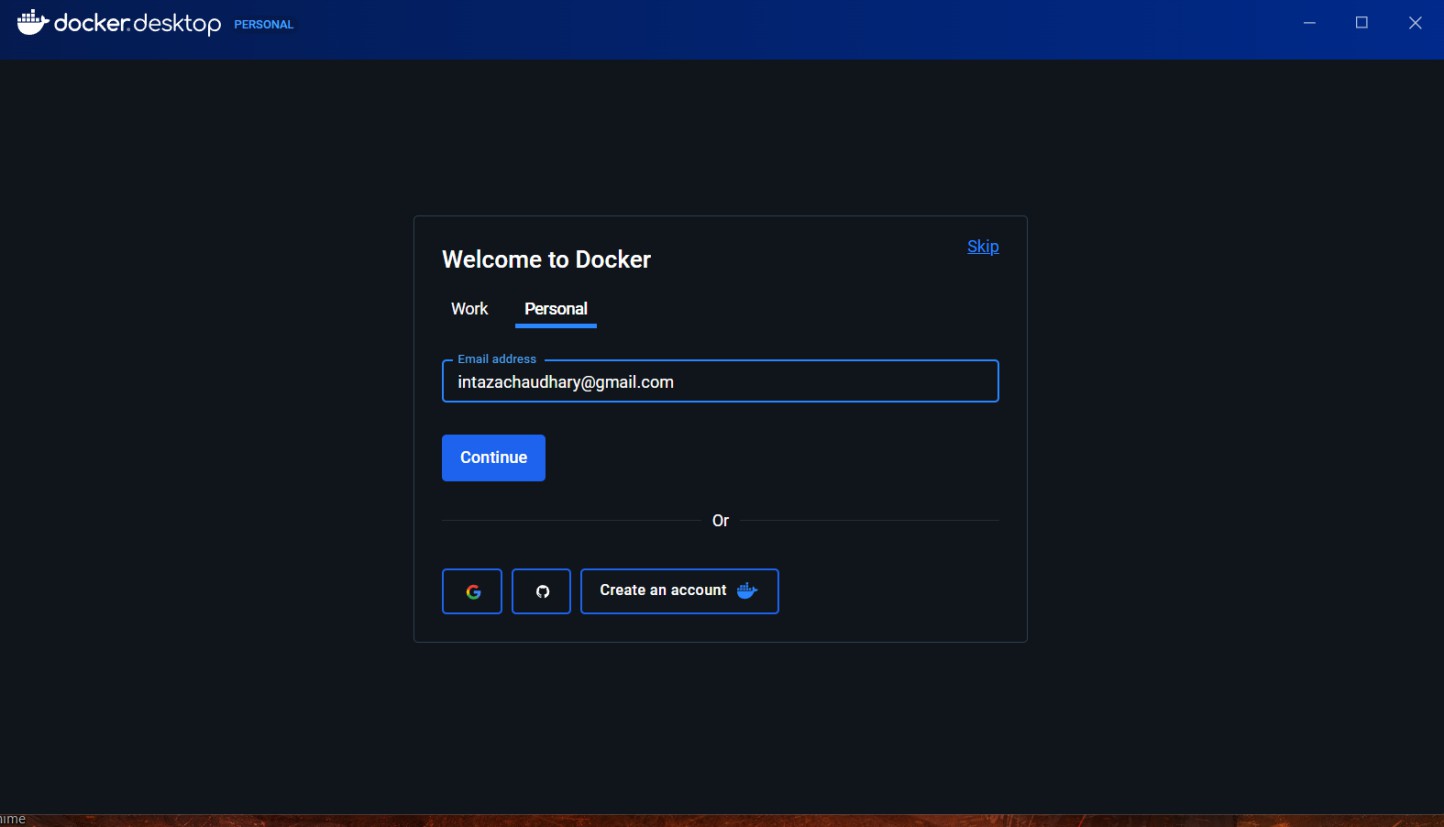
**Step 1: Install Docker on Windows**

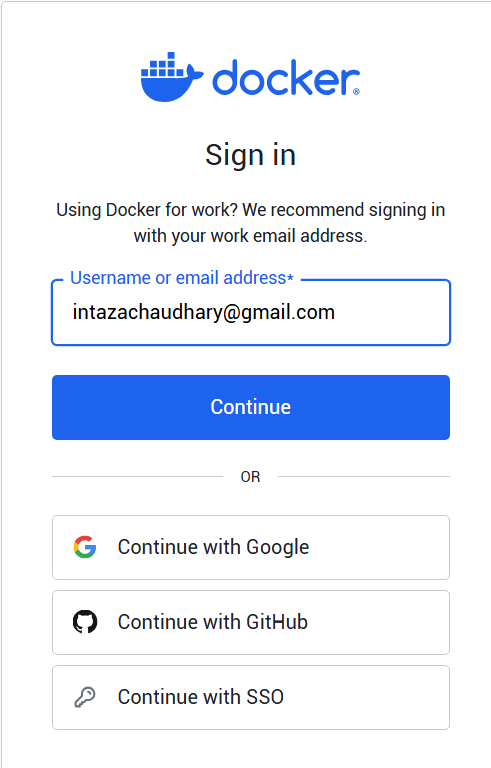
* + - 1. Go to the official Docker download page:

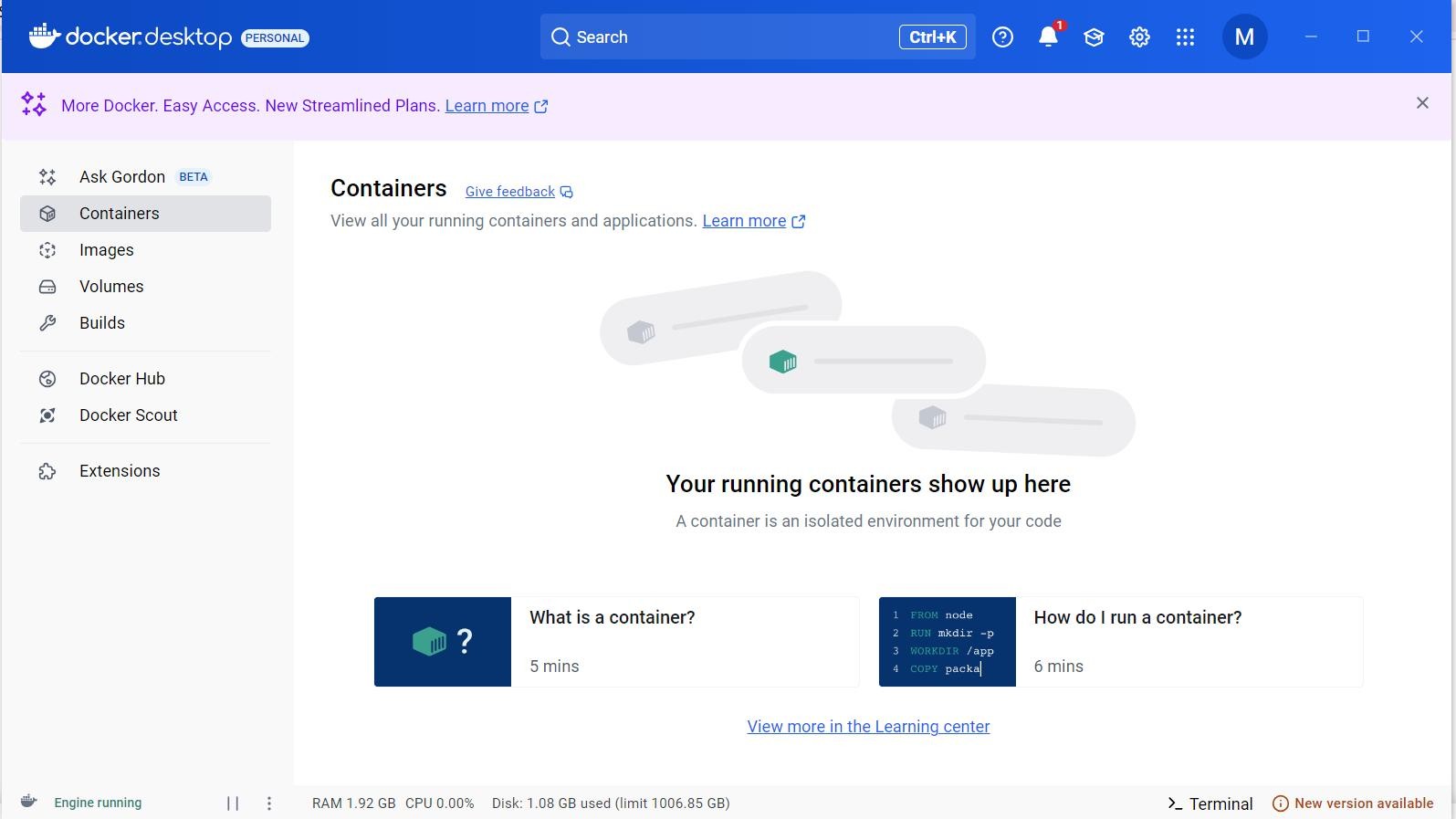
³ https://docs.docker.com/desktop/install/windows-install

* + - 1. **Download Docker Desktop for Windows**.
      2. **Run the installer** and follow the on-screen instructions:
         * Ensure **WSL 2** is enabled (Docker will prompt you to install it if needed).
         * You may need to restart your system.
      3. After installation, **launch Docker Desktop**. You should see the whale icon in the system tray.







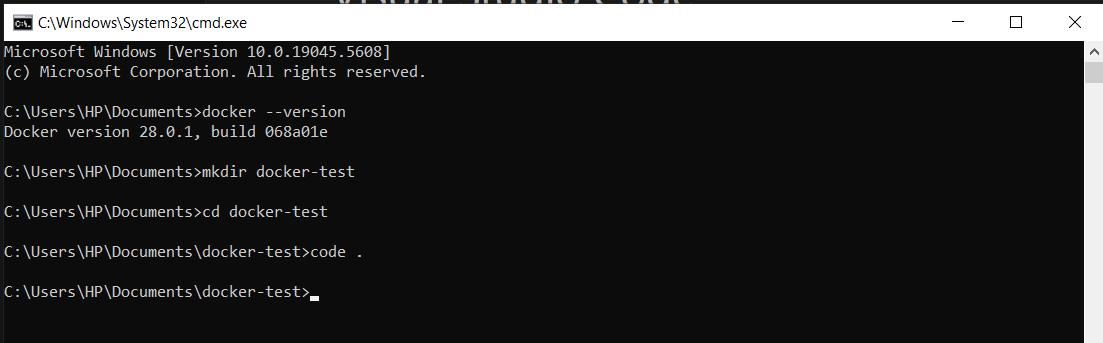


Step 2:To verify installation, open **Command Prompt (CMD)** or **PowerShell** and type:

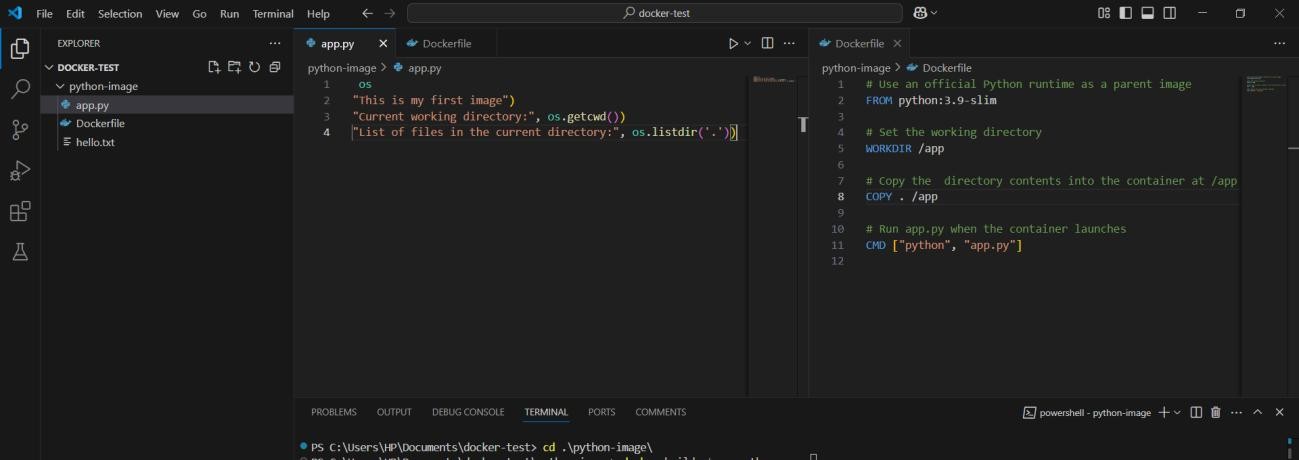
docker –version

**Create a new folder** for your project:

mkdir docker-test cd docker-test



**Step 3:Open any code editor** (e.g., VS Code) and create a file named Dockerfile (no extension).



Step 4: Add the following sample content

## App.py

import os

print("This is my first image")

print("Current working directory:", os.getcwd())

print("List of files in the current directory:", os.listdir('.'))

Dockerfile

# Use an official Python runtime as a parent image FROM python:3.9-slim

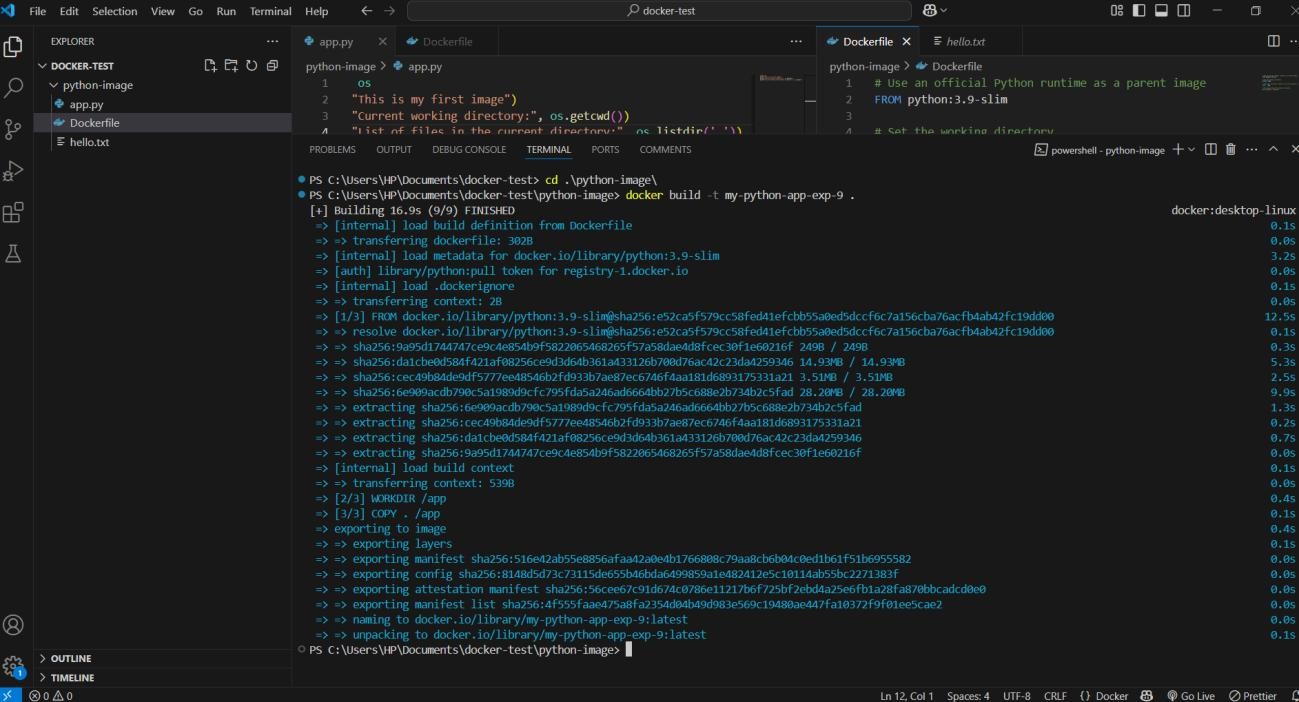
# Set the working directory WORKDIR /app

# Copy the directory contents into the container at /app COPY . /app

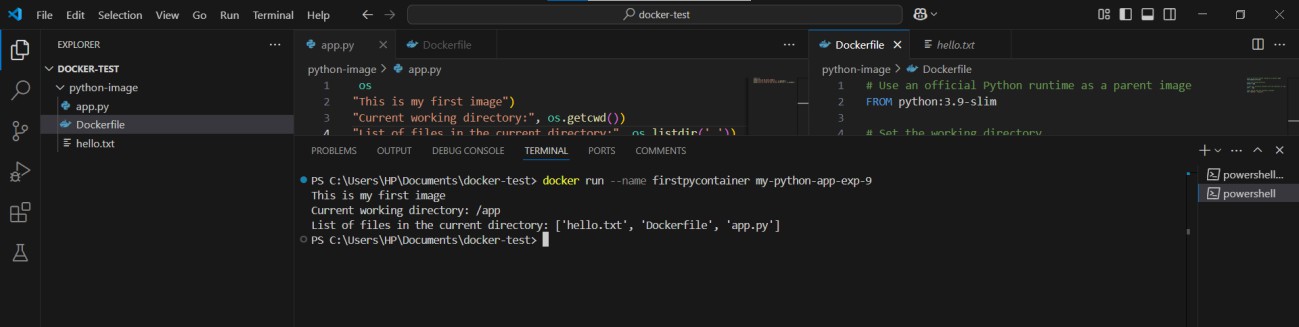
# Run app.py when the container launches CMD ["python", "app.py"]

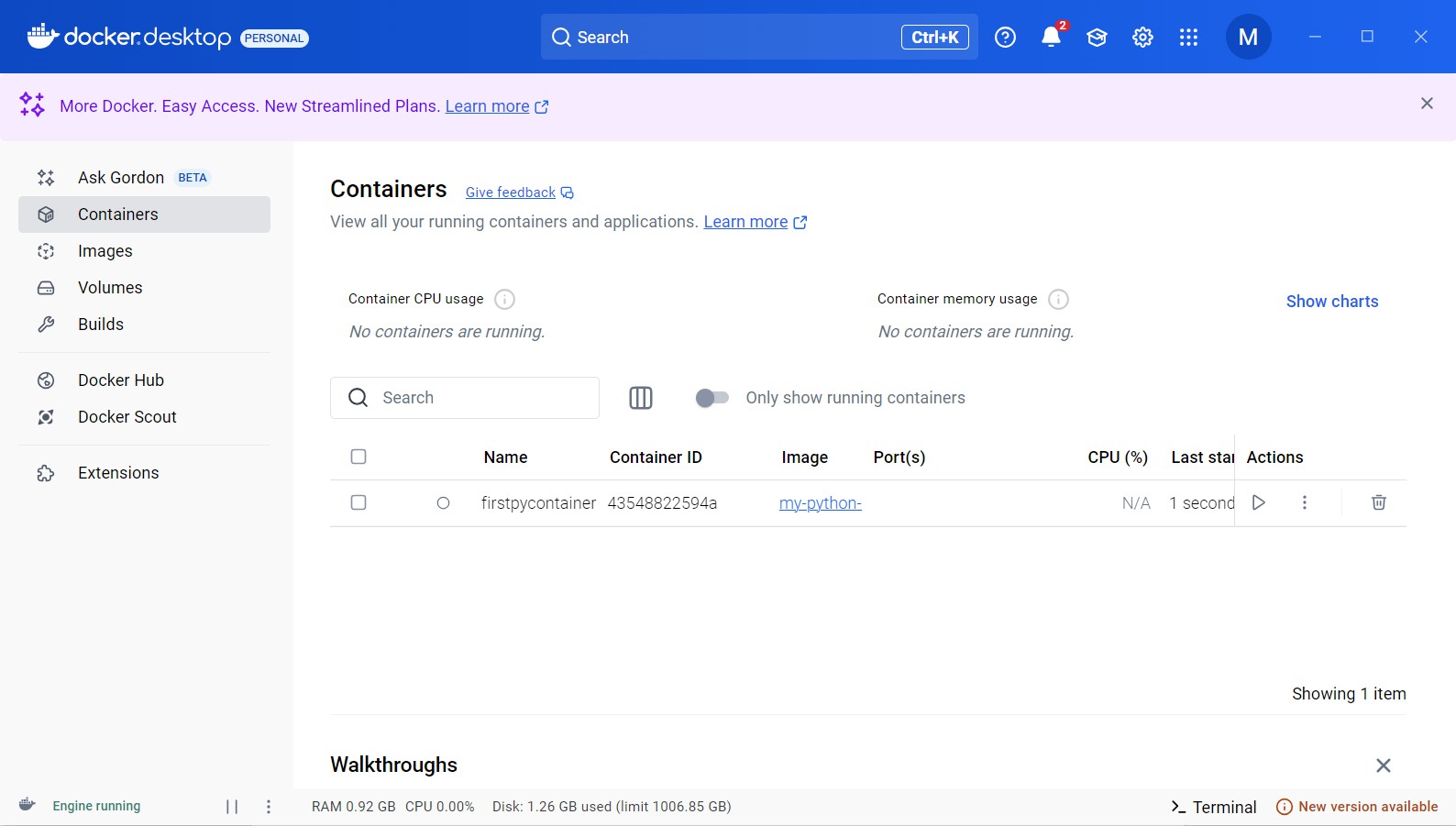
Hello.txt hii

Step 5;Build the Docker Image



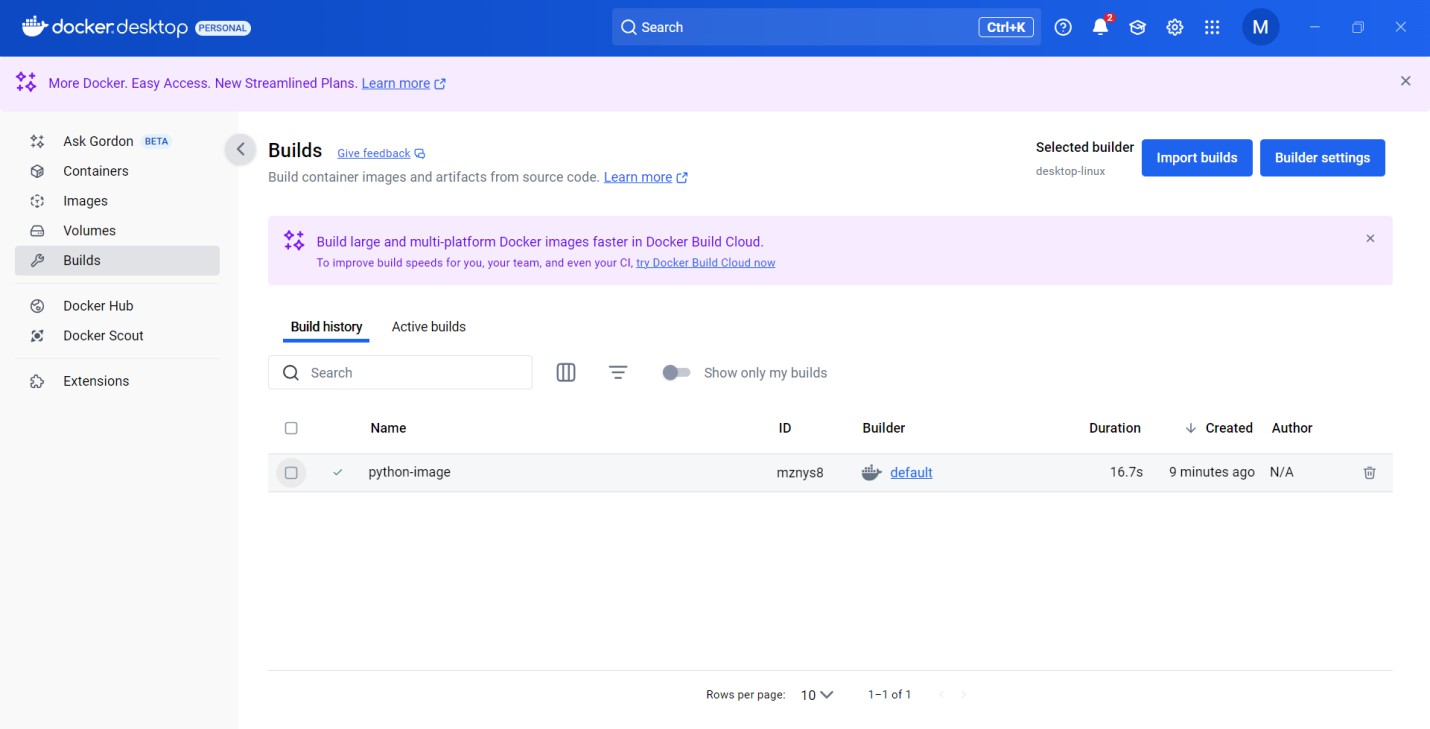
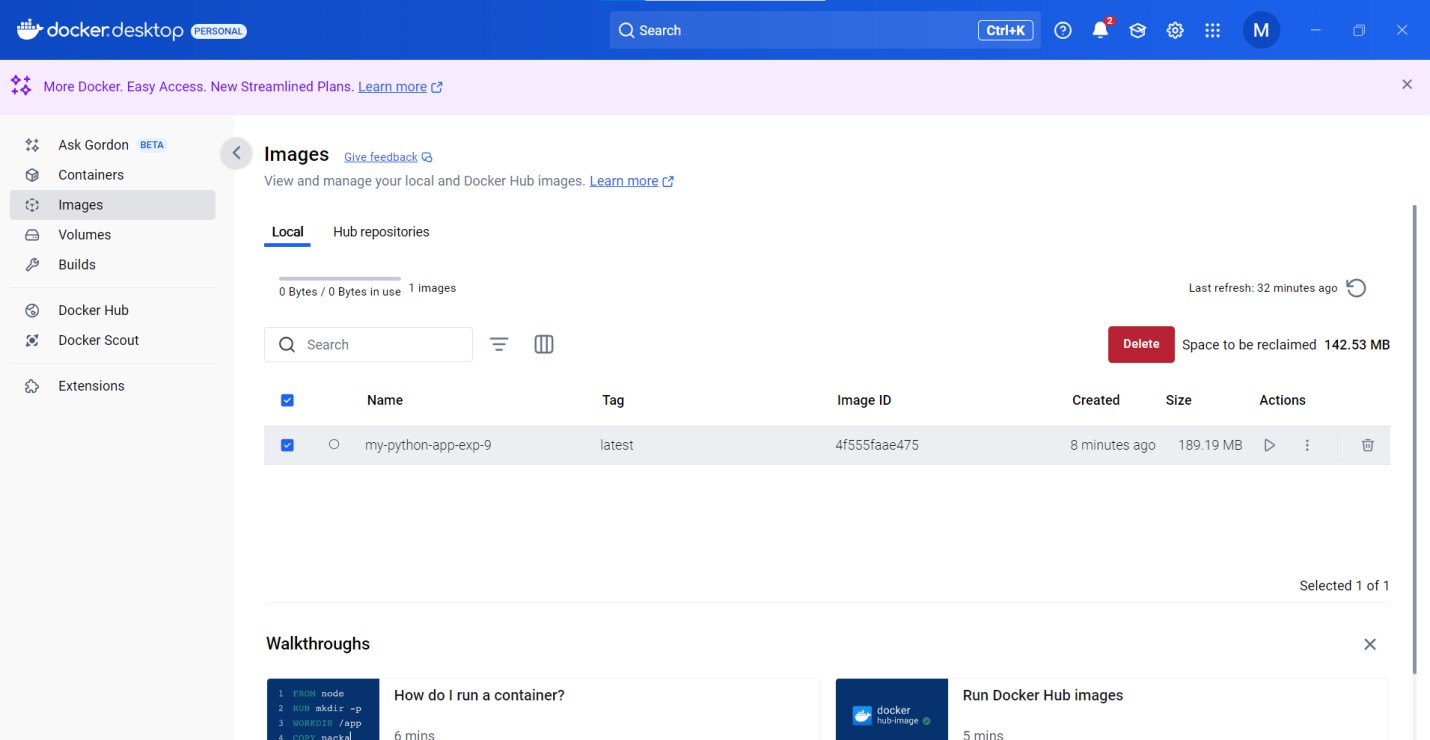
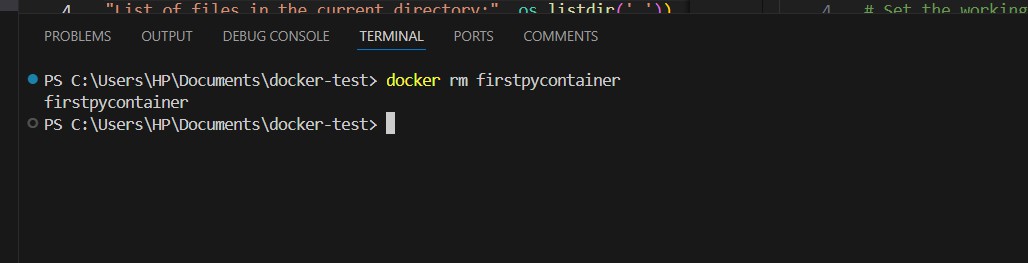
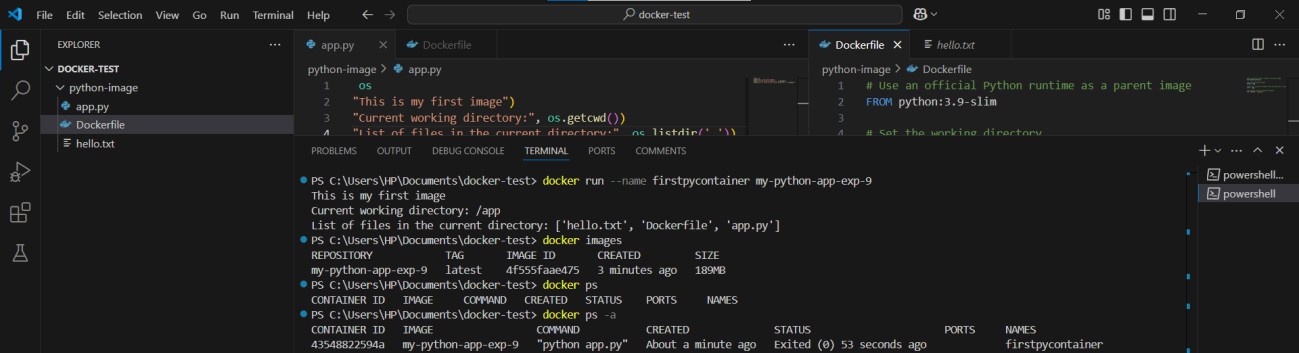
Step 6: Run the Docker Image

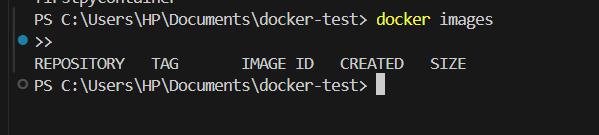




**Step 7:Bonus Commands**

* **List all images**: docker images
* **List running containers**: docker ps
* **List all containers (including stopped)**: docker ps -a
* **Stop a running container** (use container ID): docker stop <container-id>





## B.1Question of Curiosity:

* 1. *What is containerization*?

# **Ans:** Containerization is a method of packaging, distributing, and running applications within isolated environments called containers. These containers encapsulate everythingneeded to run an application, including code, runtime, libraries, and dependencies. Unliketraditional virtual machines, containers share the host operating system's kernel, makingthem lightweight and more efficient. Containerization facilitates portability, scalability, and consistency across different computing environments, enabling developers to build, deploy, and manage applications more efficiently across various platforms and infrastructure.

* 1. *Difference between virtualization and containerization***.**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Virtualization (VM)** | **Containerization (Docker)** |
| **Isolation** | Full OS-level isolation with its own kernel | Process-level isolation using the host OS kernel |
| **Resource Usage** | Heavy; each VM runs its own full OS | Lightweight; containers share the host OS |
| **Startup Time** | Slow (in minutes) | Fast (in seconds) |
| **Performance** | Slower due to full OS overhead | Faster due to minimal abstraction |
| **Size** | Large (GBs, due to OS image) | Small (MBs, since only app and dependencies are included) |
| **Portability** | Less portable; depends on hypervisor and OS | Highly portable; runs the same way on any OS with Docker installed |
| **Boot Process** | Full OS boot required | Just starts the app process |
| **Isolation Scope** | Hardware-level using hypervisor | OS-level using namespaces and cgroups |
| **Use Cases** | Best for running multiple different OSes | Best for microservices, lightweight app deployment |

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Virtualization (VM)** | **Containerization (Docker)** |
| **Example Tools** | VMware, VirtualBox, Hyper-V | Docker, Podman, containerd |

* 1. *What is docker image?*

# **Ans:** A Docker image is a lightweight, standalone, and executable package that contains all the necessary components to run a software application, including code, libraries, dependencies, and runtime environment. It serves as a blueprint for creating Docker containers. Docker images are built from a Docker file, which specifies the configuration and instructions for assembling the image layer by layer. These images can be easily shared, distributed, and deployed across different environments, ensuring consistency and reproducibility in application deployment processes.

* 1. *What is docker container?*

# **Ans:** A Docker container is a runtime instance of a Docker image. It is a lightweight, standalone, and executable environment that encapsulates an application along with its dependencies. Containers run isolated from each other and from the underlying host system, leveraging kernel-level features like namespaces and control groups for resourceisolation and management. Docker containers provide consistency in application deployment across different environments, enabling developers to package, deploy, and manage applications efficiently and reliably, irrespective of the underlying infrastructure.

**B.3 Conclusion:**

After performing this experiment, I learned the core concepts of containerization using Docker. I understood the difference between virtualization and containerization and successfully created and ran a Docker container on my local machine. This will help me deploy apps in a more efficient and portable way.