EXPERIMENT 8

Aim: To study and Implement Containerization using Docker

Objectives:

- Certainly, here are the objectives for studying and implementing containerization using Docker in point format:
- Understand the fundamental concepts of containerization and its significance in modern application development and deployment.
- Explore Docker, a leading containerization platform, including its architecture, components, and core features.
- Learn how to install Docker on various operating systems and set up Docker Engine for container deployment.
- Gain proficiency in Dockerfile syntax and Docker Compose for defining container configurations and multi-container applications.
- Practice building Docker images, creating containers from images, and managing container lifecycles using Docker commands.

Theory: Docker is an open source software platform used to create, deploy and manage virtualized application containers on a common operating system (OS), with an ecosystem of allied tools. Docker container technology debuted in 2013. At that time, Docker Inc. was formed to support a commercial edition of container management software and be the principal sponsor of an open source version. Mirantis acquired the Docker Enterprise business in November 2019. Docker gives software developers a faster and more efficient way to build and test containerized portions of an overall software application. This lets developers in a team concurrently build multiple pieces of software. Each container contains all elements needed to build a software component and ensure it's built, tested and deployed smoothly. Docker enables portability for when these packaged containers are moved to different servers or environments.

A container is an isolated environment for your code. This means that a container has no knowledge of your operating system, or your files. It runs on the environment provided to you by Docker Desktop. Containers have everything that your code needs in order to run, down to a base operating system. You can use Docker Desktop to manage and explore your containers.

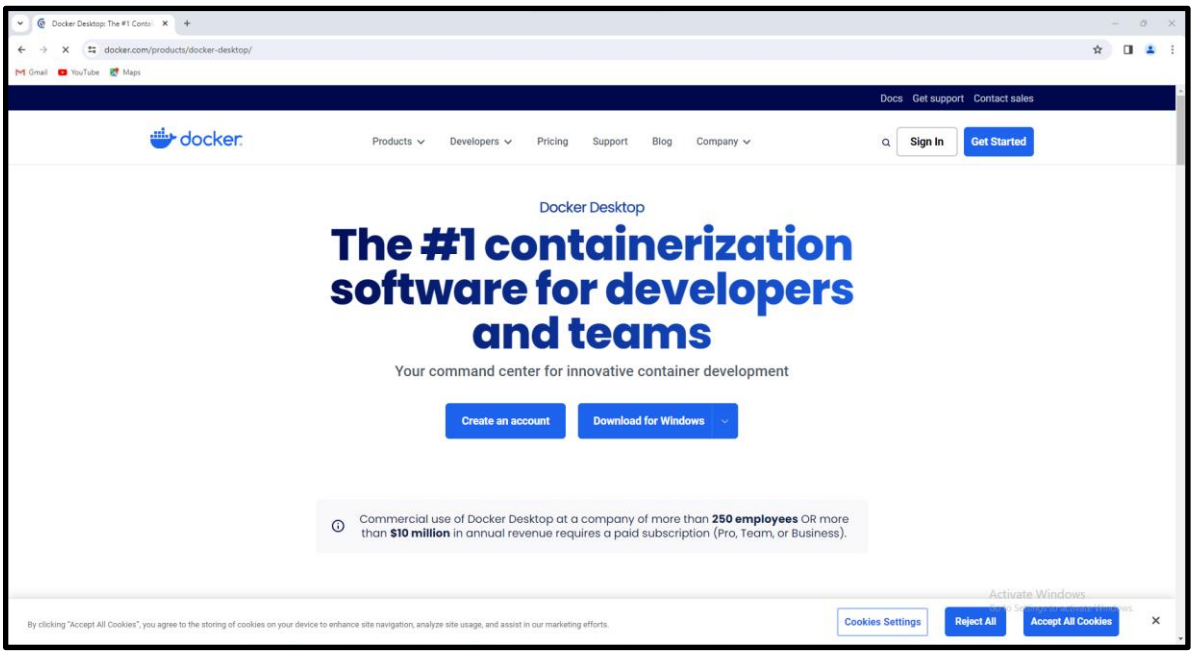
How Docker works

Docker packages, provisions and runs containers. Container technology is available through the operating system: A container packages the application service or function with all of the libraries,

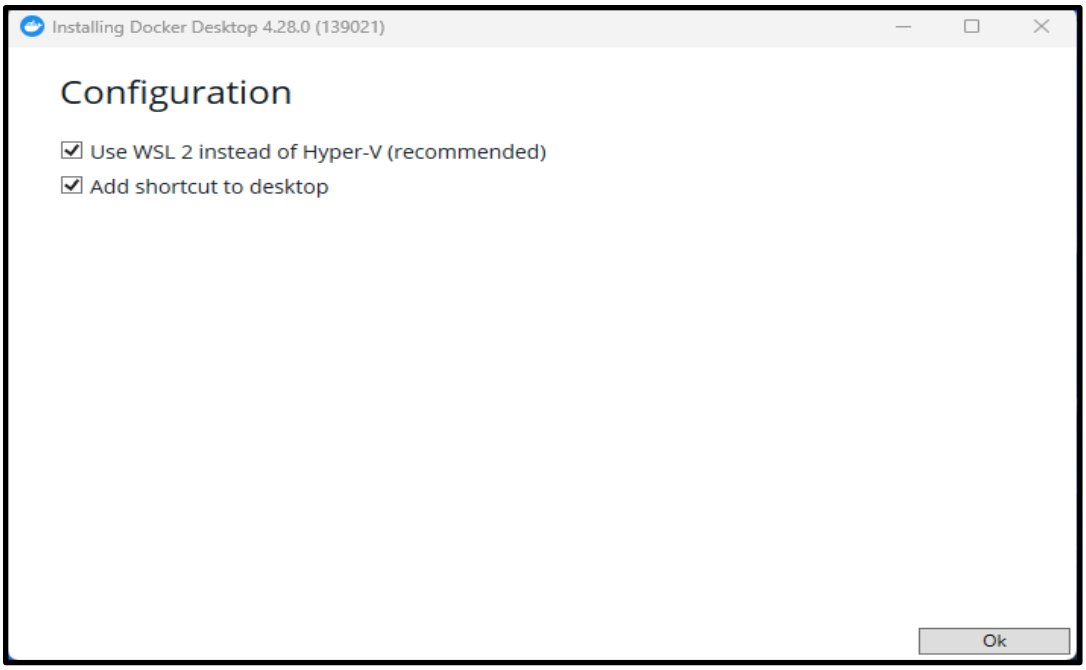
configuration files, dependencies and other necessary parts and parameters to operate. Each container shares the services of one underlying OS. Docker images contain all the dependencies needed to execute code inside a container, so containers that move between Docker environments with the same OS work with no changes. Docker uses resource isolation in the OS kernel to run multiple containers on the same OS. This is different from virtual machines (VMs), which encapsulate an entire OS with executable code on top of an abstracted layer of physical hardware resources.

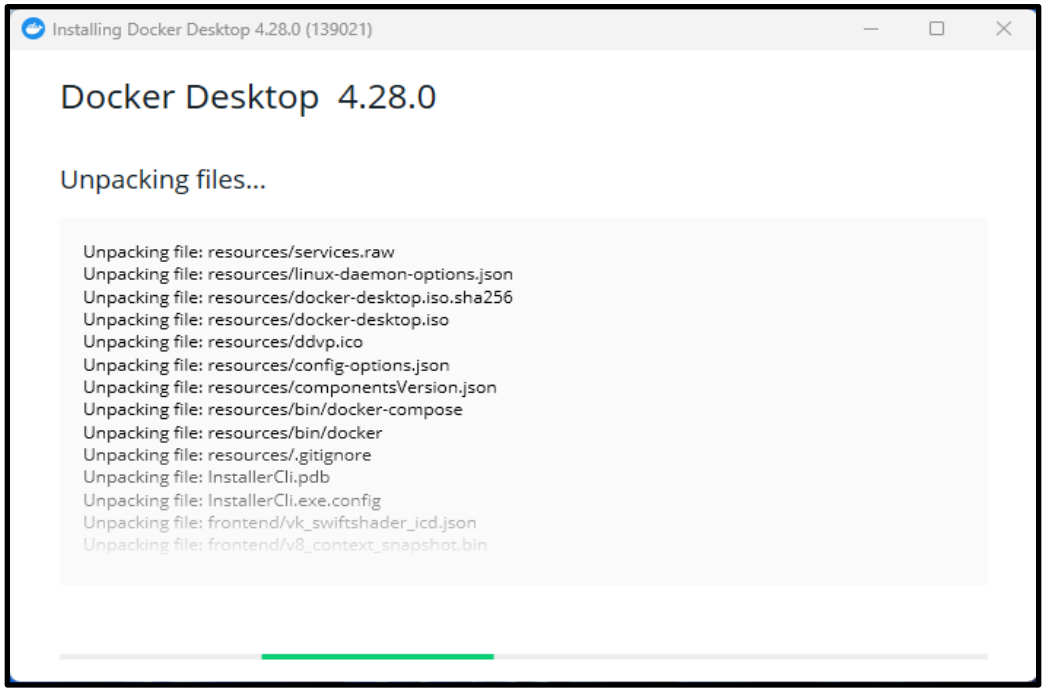
Implementation and Output:

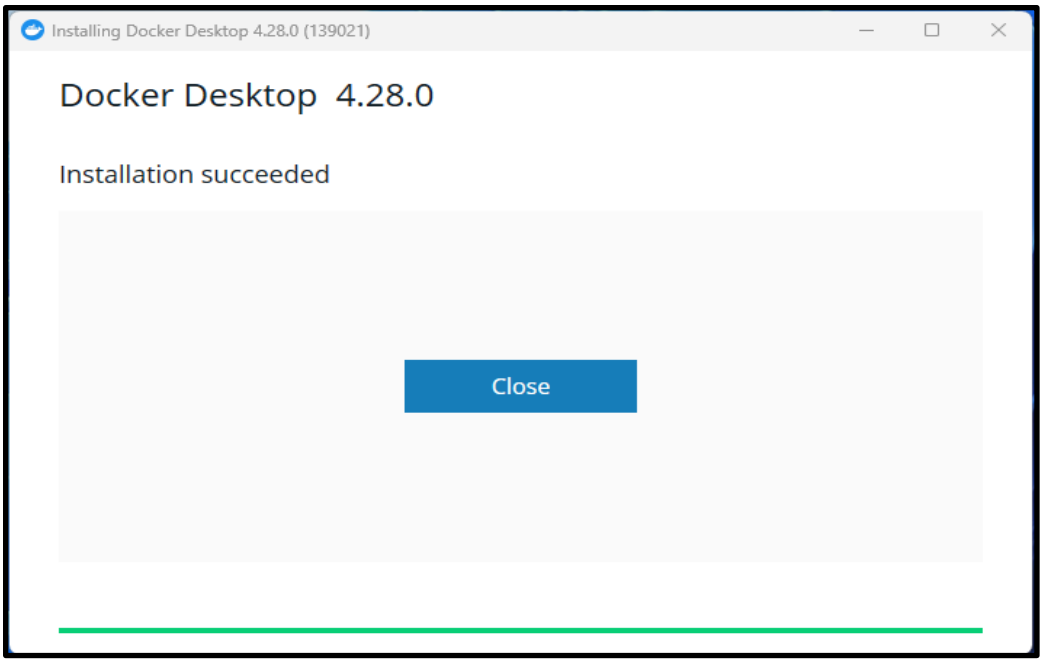
Step 01: Downloading Docker https://www.docker.com/products/docker-desktop/ using the link on windows.



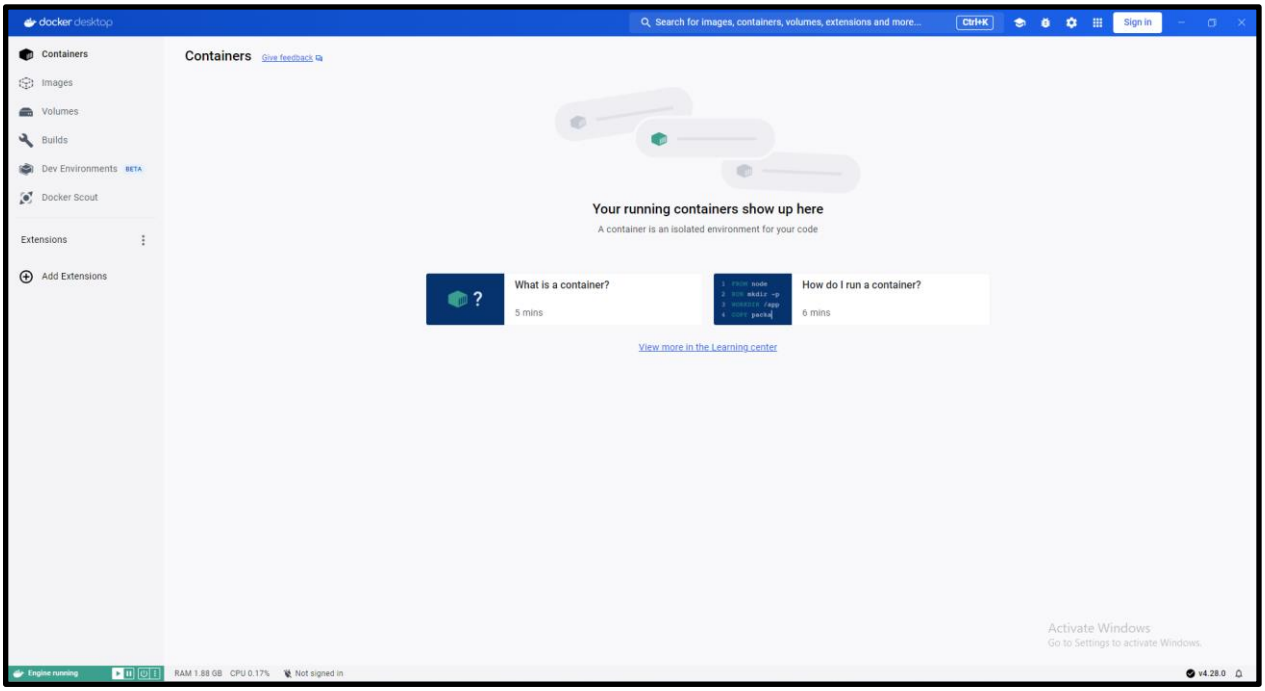
Step 02: Installing Docker-Desktop on windows.


Installing Docker Desktop 4.28.0 (139021)
Configuration
Use WSL 2 instead of Hyper-V (recommended)
Add shortcut to desktop
Ok

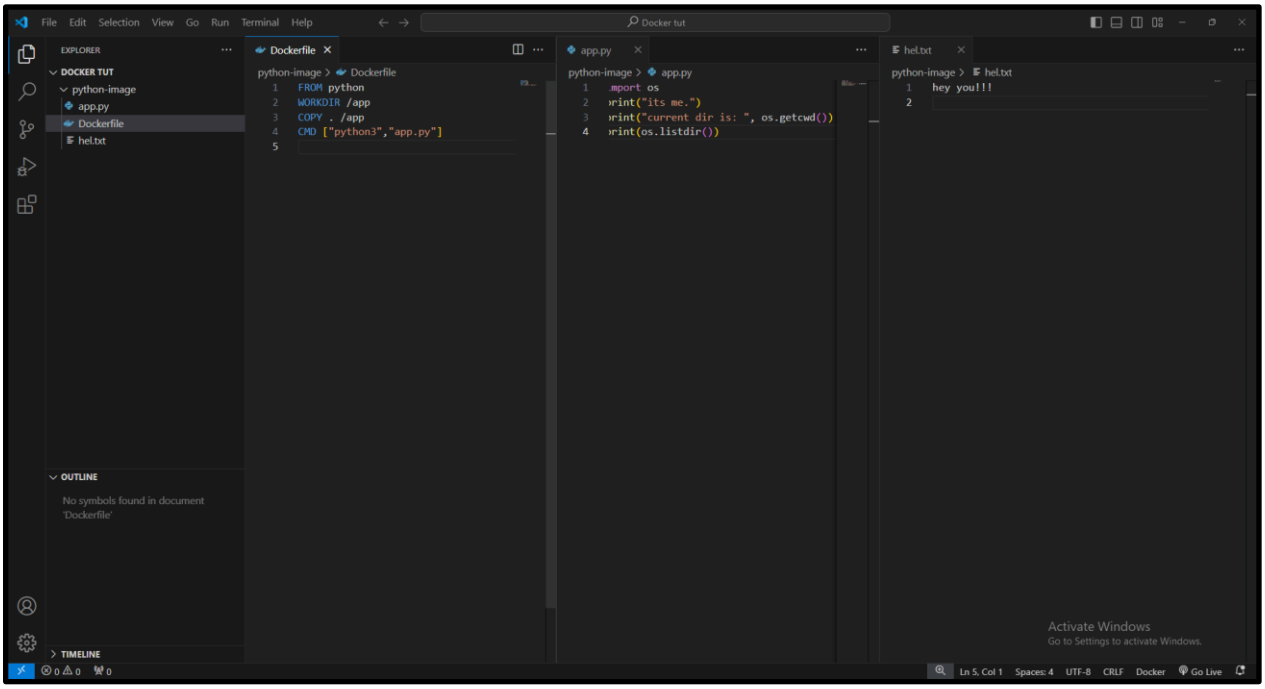

Installing Docker Desktop 4.28.0 (139021)
Docker Desktop 4.28.0
Unpacking files...
Unpacking file: resources/services.raw
Unpacking file: resources/linux-daemon-options.json
Unpacking file: resources/docker-desktop.iso.sha256
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin



Step 03: Now here we will have no containers in the installed docker.

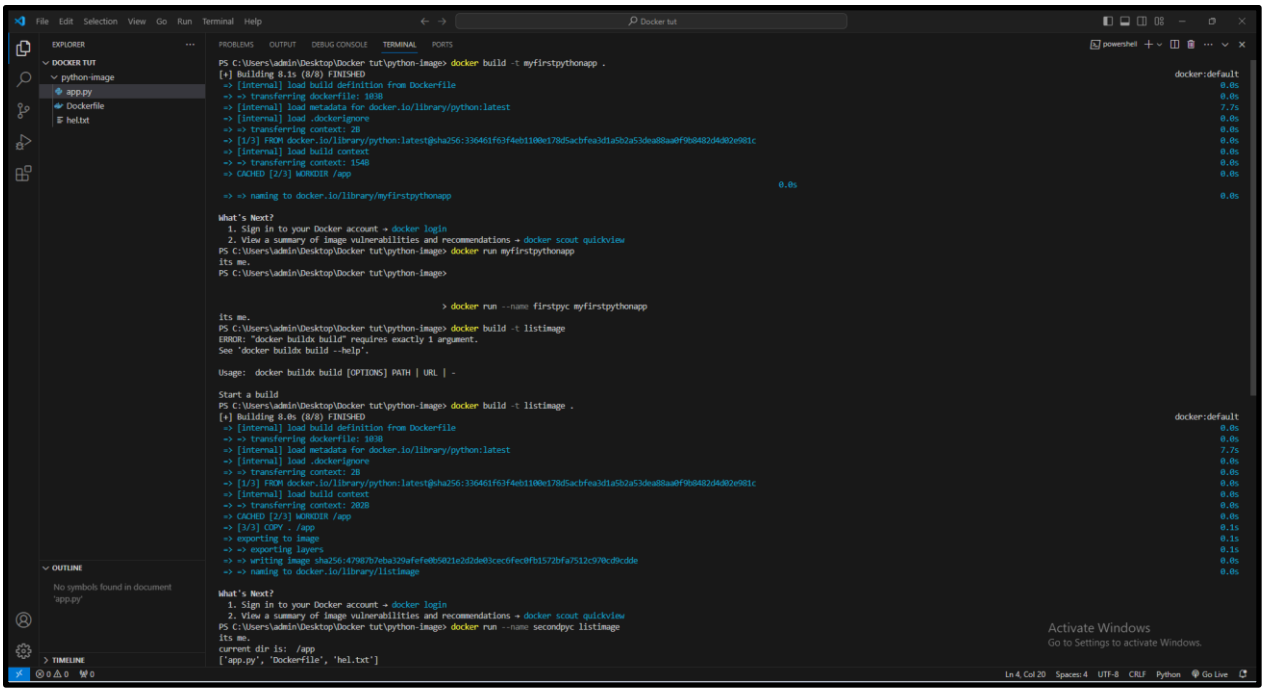


Step 04: Let's Create Containers for our code and files. First Open the VS Code (visual studio code) and create one folder. Create one more folder in the same folder and write code in the app.py and Dockerfile by creating new files. Also create one text file i.e. hel.txt.

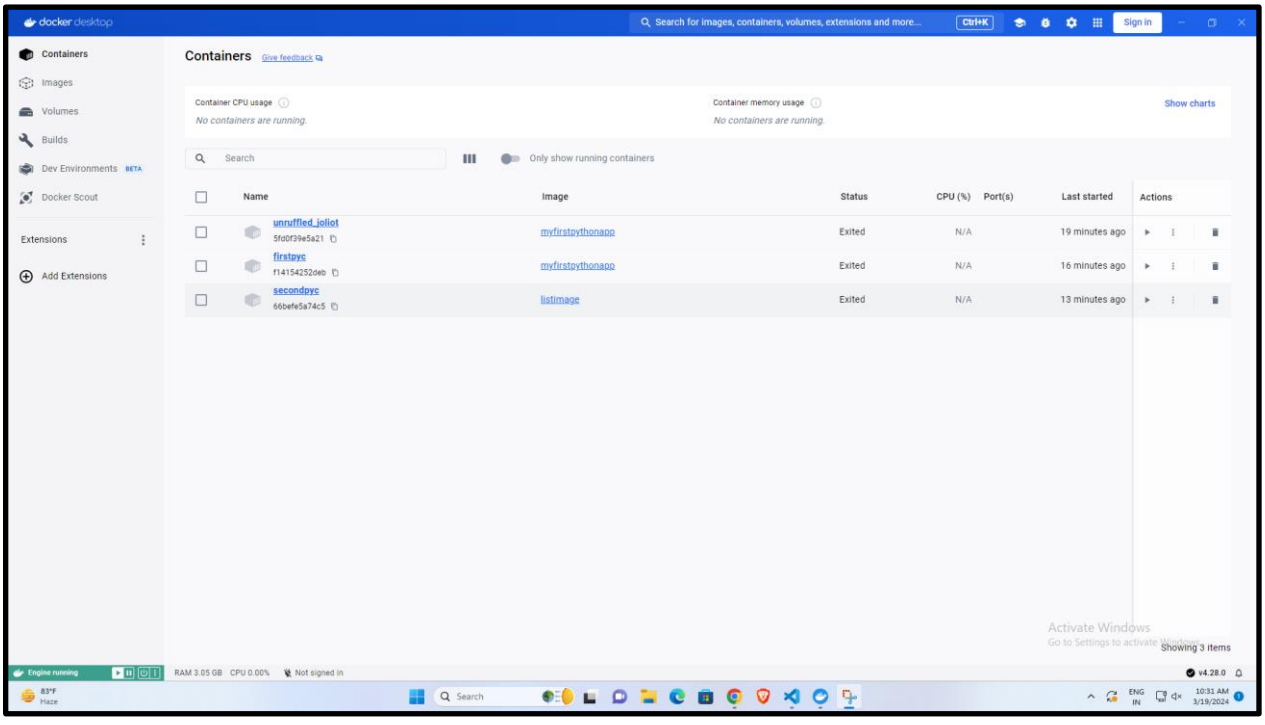


Step 05: Open the new terminal on VS Code and execute commands one by one as shown below:

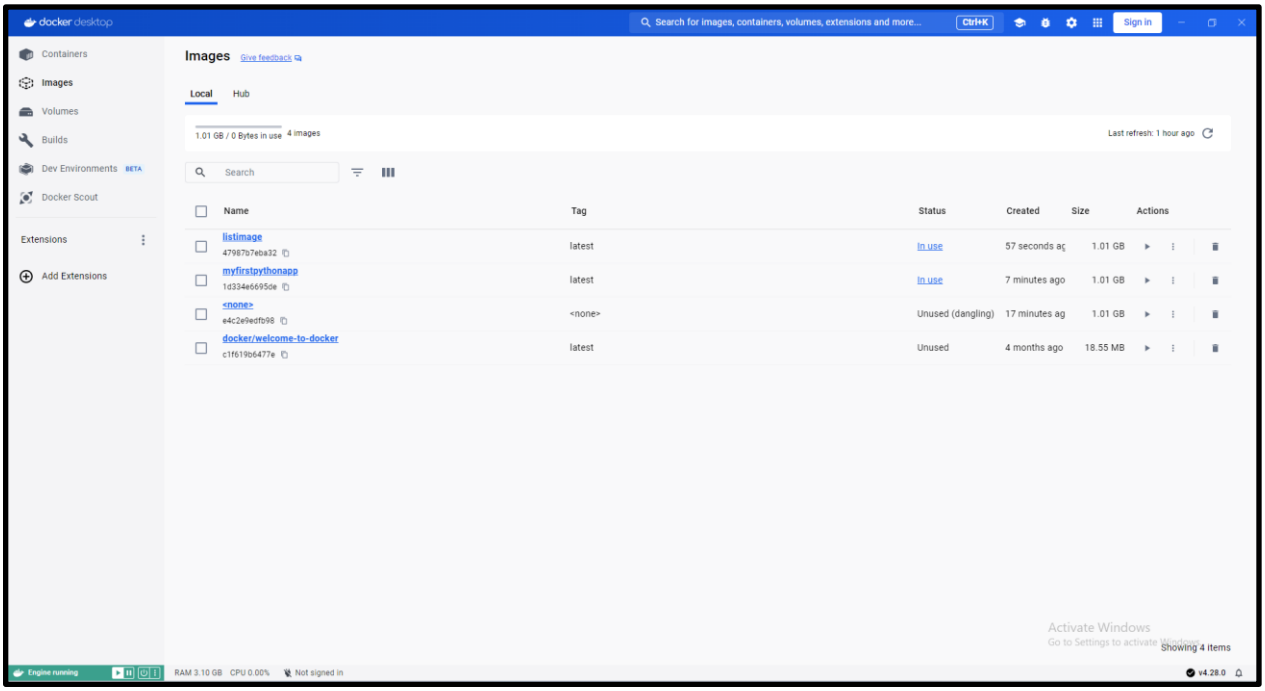
1. cd .\your_inner_folder_name
2. docker build -t myfirstpythonapp .
3. docker run myfirstpythonapp
4. docker run --name firstpyc myfirstpythonapp
5. docker build -t listimage .
6. docker run --name secondpyc listimage



Step 06: Open the Docker and click the Containers menu. Now you can see the created Containers.



Step 07: Open the Docker and click the Images menu. Now you can see the created Images on the Docker.



Conclusion: Docker uses fewer resources compared to traditional virtual machines. Because containers share the same underlying operating system kernel, they are much lighter and require less disk space and RAM. This allows more applications to run on a single machine, which saves hardware costs and eases resource management. Docker has revolutionized the way we develop, deliver and run applications. Its focus on portability, isolation, resource efficiency, scalability, and development speed make it an essential tool for any development and operations team.