DOCKER CONTAINERS

Docker Containers

- *Docker is a platform for developing, shipping, and running applications inside the containers.*
- *Docker is widely used for its portability and efficiency in managing dependencies.*
- *Containers provide a consistent and isolated environment for applications.*

Advantages of Docker

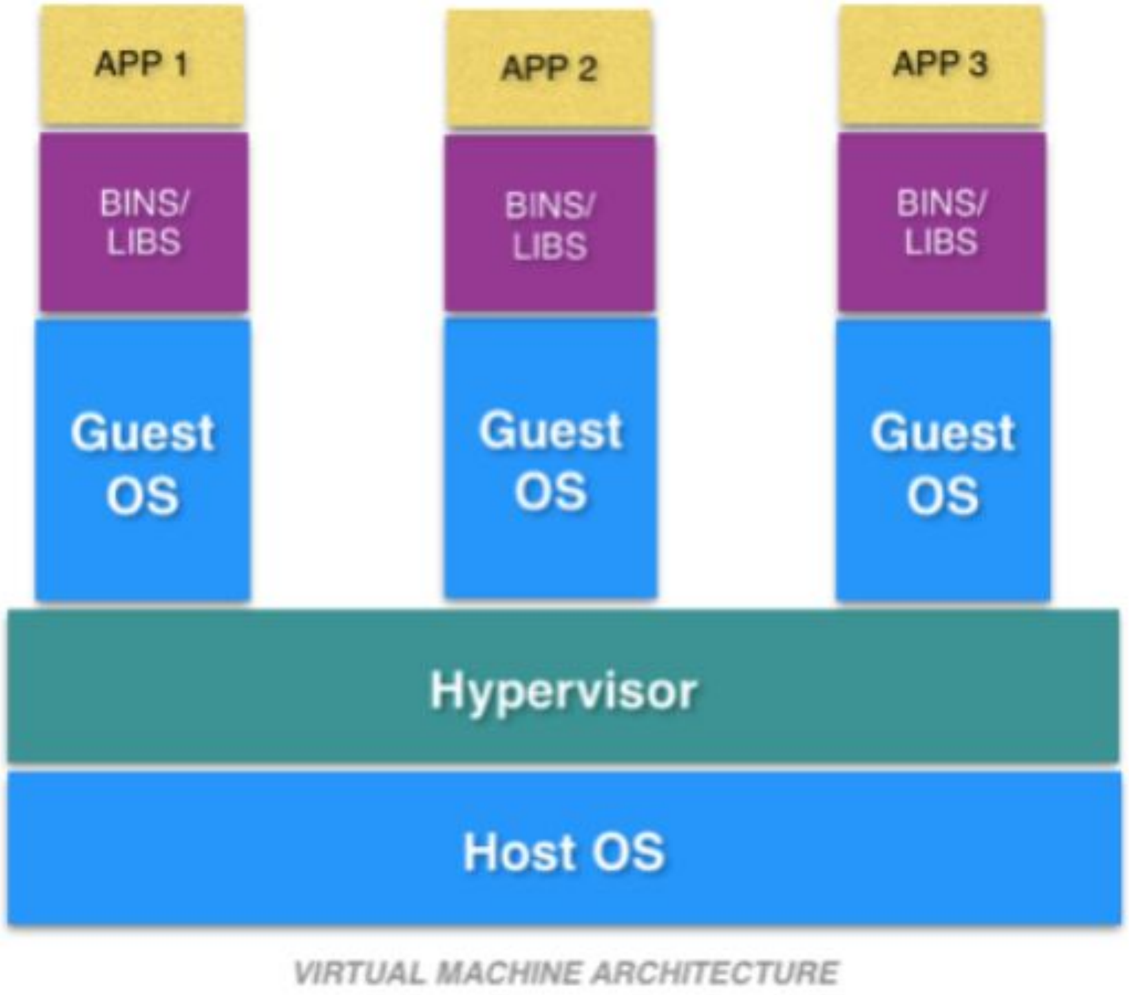
1. **Consistency:** Docker ensures consistency between development, testing, and production environments by packaging applications and their dependencies into containers.
2. **Portability:** Containers can run on any platform that supports Docker, enabling easy migration of applications across different environments.
3. **Efficiency:** Docker's lightweight containers consume fewer resources compared to virtual machines, leading to faster deployment and scalability.
4. **Isolation:** Containers provide isolation for applications, preventing conflicts between dependencies and ensuring security.
5. **DevOps Integration:** Docker facilitates DevOps practices by streamlining the development, testing, and deployment processes through containerization.

Difference between Docker and VMs

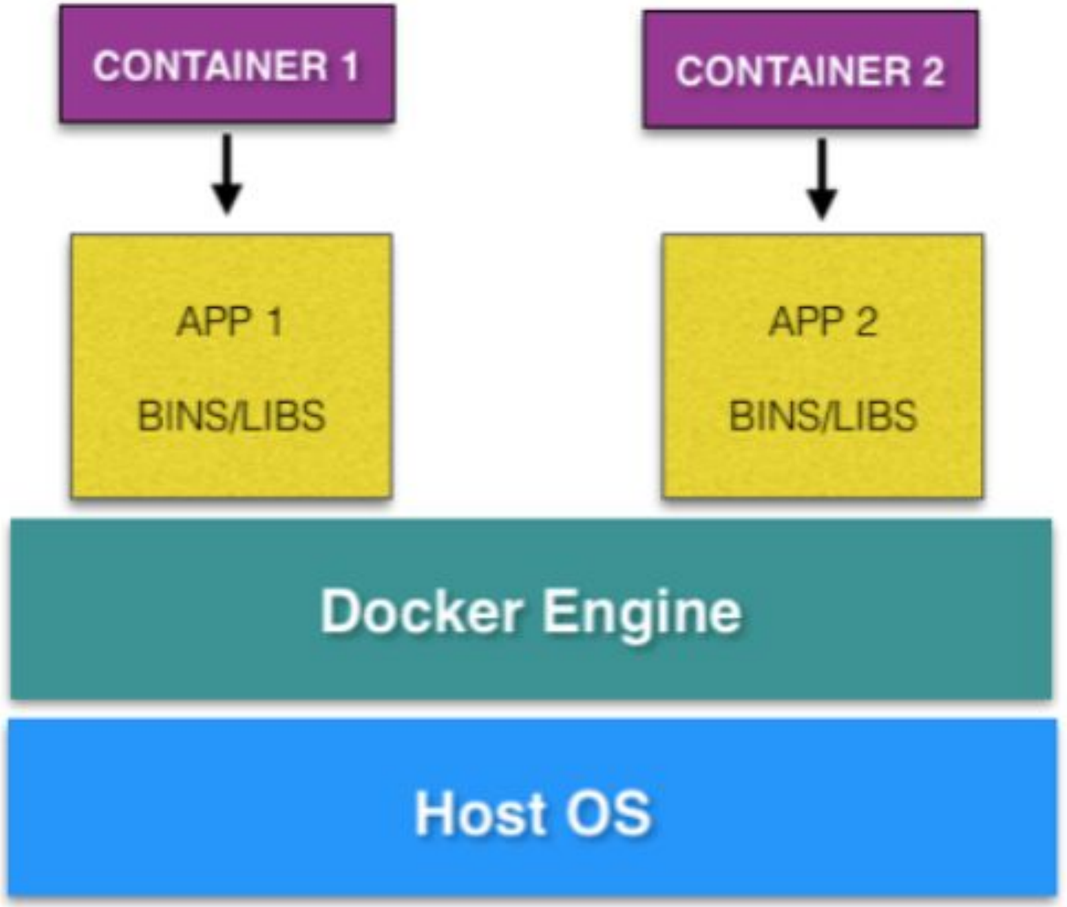
Docker and Virtual Machines serve similar purposes but operate at different levels of abstraction and have distinct use cases

	Docker	Virtual Machines (VMs)
Boot-Time	Boots in a few seconds.	It takes a few minutes for VMs to boot.
Runs on	Dockers make use of the execution engine.	VMs make use of the hypervisor.
Memory Efficiency	No space is needed to virtualize, hence less memory.	Requires entire OS to be loaded before starting the surface, so less efficient.
Isolation	Prone to adversities as no provisions for isolation systems.	Interference possibility is minimum because of the efficient isolation mechanism.
Deployment	Deploying is easy as only a single image, containerized can be used across all platforms.	Deployment is comparatively lengthy as separate instances are responsible for execution.
Usage	Docker has a complex usage mechanism consisting of both third party and docker managed tools.	Tools are easy to use and simpler to work with.

VM vs Docker Container



VIRTUAL MACHINE ARCHITECTURE



DOCKER ARCHITECTURE

Which one to use?

- Use Docker if: You need lightweight, portable, and efficient isolation for microservices architectures, cloud-native applications, and modern development practices.
- Use VMs if: You require stronger isolation, support for legacy applications, or heterogeneous environments with diverse infrastructure

Key concepts of docker

1. Containerization: Containers encapsulate applications and their dependencies, ensuring consistency and isolation. Unlike virtual machines, containers share the host operating system's kernel, making them lightweight and fast to start.

2. Images: Docker images are read-only templates used to create containers. Images contain the application code, runtime, libraries, and dependencies needed to run the application.

3. Containers: Containers are instances of Docker images that run applications in isolated environments. Containers can be started, stopped, moved, and deleted, providing a scalable and flexible deployment model.

Docker Setup on Windows

System Requirements:

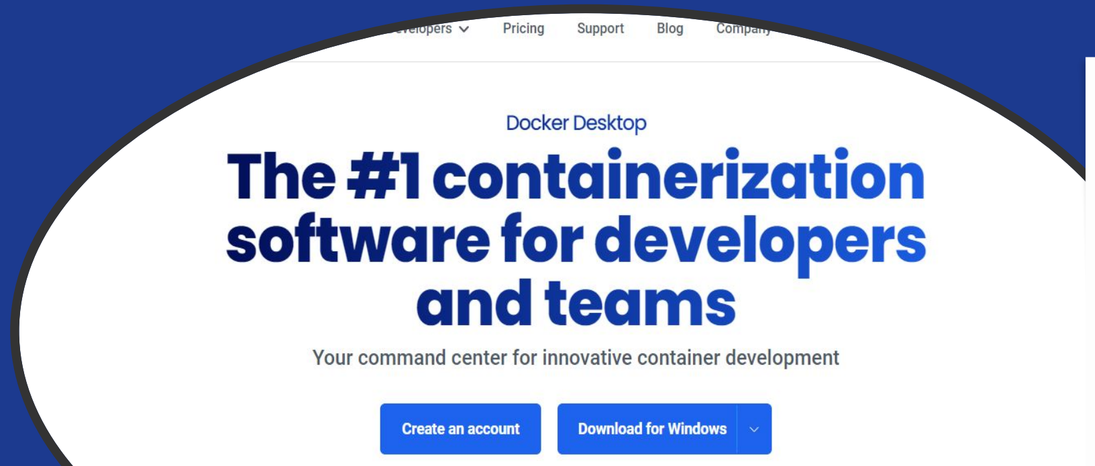
- Windows 10 64-bit: Home, Pro, or Enterprise edition (Build 19018 or higher)
- Hyper-V enabled
- Hardware-assisted virtualization and data execution prevention must be enabled in BIOS settings

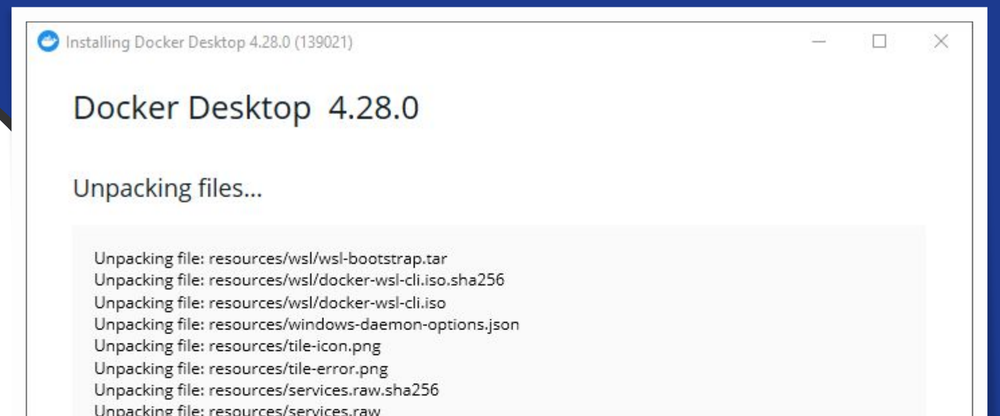
Configuration:

- Docker Desktop provides a system tray icon to access settings and manage Docker resources
- You can configure Docker settings such as resource allocation, network settings, and shared drives through the Docker Desktop interface

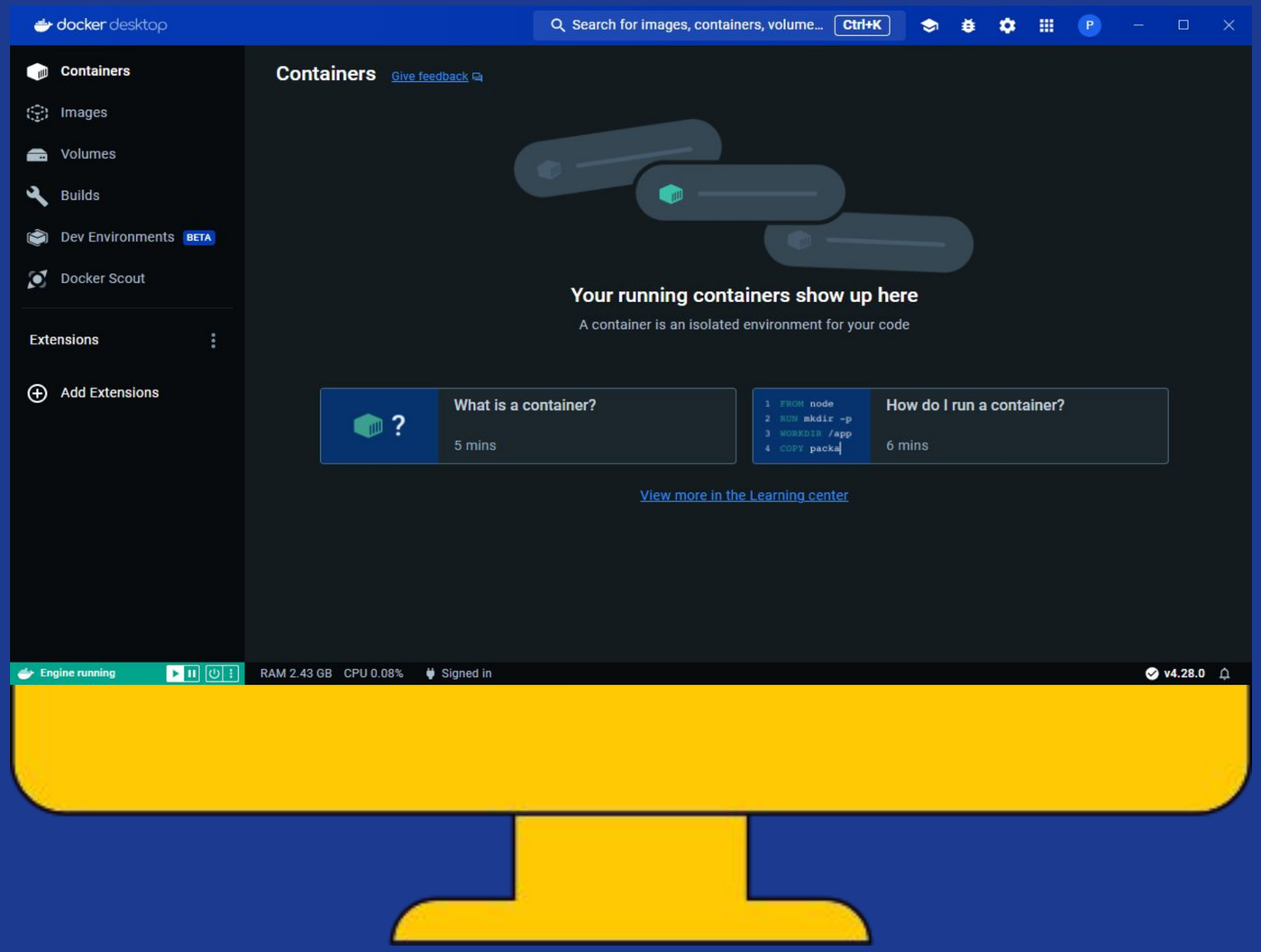
Installation Steps:

- Download Docker Desktop for Windows from the official Docker website
- Double-click the installer and follow the on-screen instructions to install Docker Desktop
- Once installed, Docker Desktop will start automatically and run in the background





Docker Desktop Interface



DEFINITION

Docker Desktop provides an intuitive user interface for managing Docker resources on Windows..

PURPOSE

- Dashboard: View container status, resource usage, and logs.
- Settings: Configure Docker engine, network, and shared drives.
- Docker CLI Integration: Access Docker commands through the integrated terminal.

BASIC STRUCTURE

What is a Dockerfile?

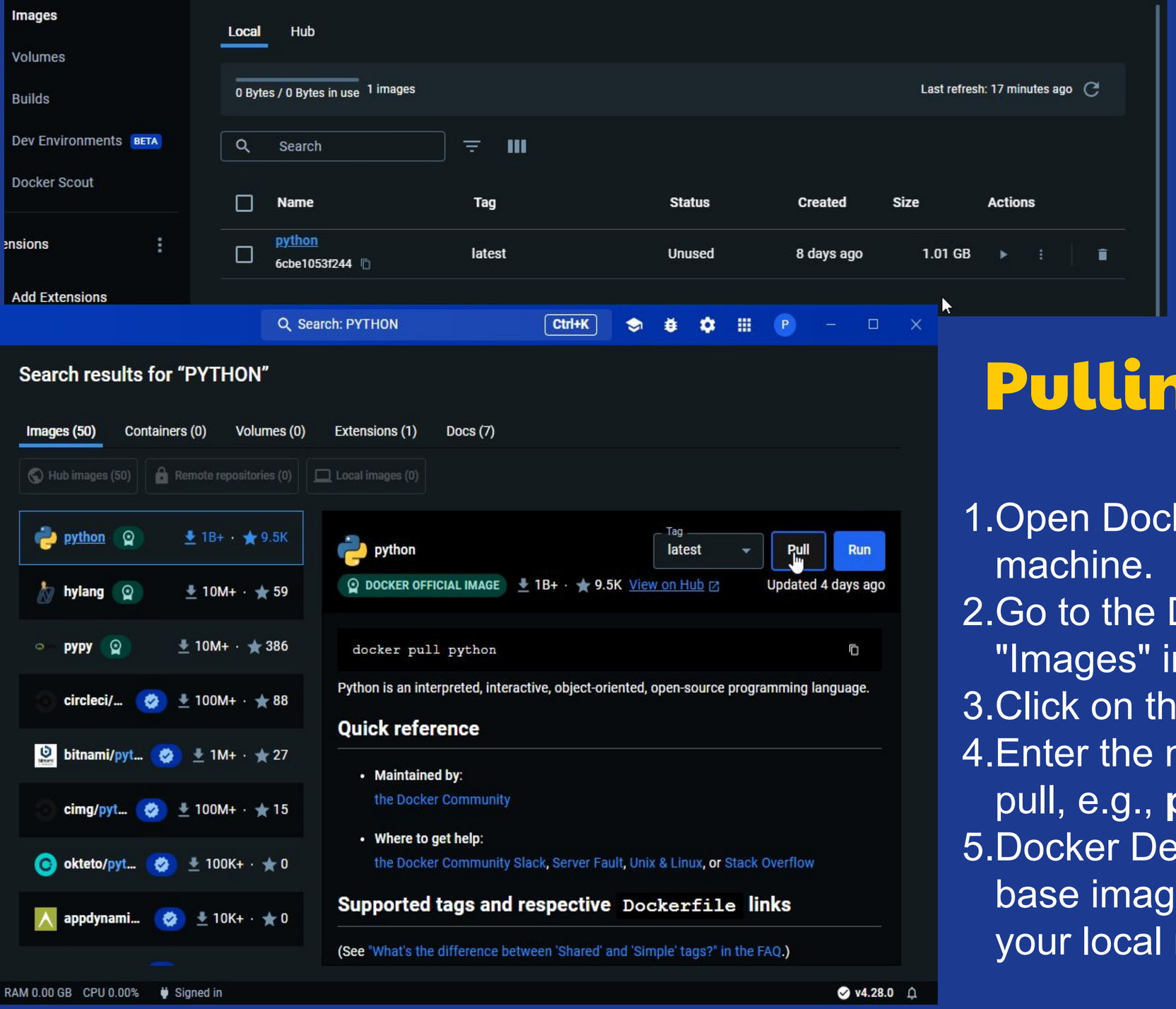
•A Dockerfile is a text document that contains instructions for building a Docker image.
•It specifies the environment and configuration needed to create a containerized application.

Base Image:

•The FROM instruction in a Dockerfile specifies the base image to build upon.
•A base image is the starting point for creating a new Docker image and provides the runtime environment for the application..

Writing Dockerfile Instructions:

•COPY or ADD: Copies files from the host into the image.
•RUN: Executes commands inside the container during the build process.
•CMD or ENTRYPOINT: Specifies the default command to run when the container starts.



Pulling a Base Image:

1. Open Docker Desktop on your Windows machine.
2. Go to the Docker Dashboard and click on "Images" in the sidebar menu.
3. Click on the "Pull" button.
4. Enter the name of the base image you want to pull, e.g., **python:3.9-slim**.
5. Docker Desktop will download the specified base image from the Docker Hub repository to your local machine

Writing a Dockerfile

Dockfile

Dockerfile > ...

```
1   FROM python:3.8
2
3   WORKDIR /fastapi-app
4
5   COPY requirements.txt .
6
7   RUN pip install -r requirements.txt
8
9   COPY ./app ./app
10
11  CMD ["python", "./app/main.py"]
```

requirements.txt

```
albumentations
ipython
object-detection-metrics
opencv-python
pandas
Pillow
pycocotools
tensorboard==2.14.0
tensorboardX==2.6.2.2
torch==2.2
torchaudio==2.2
torchinfo==1.8
torchvision==0.17
tqdm
```

- Within the project directory, create a new file named "Dockerfile".

Define Base Image:

- Use the FROM instruction to specify the base image.

"FROM python:3.8"

Copy Application Files: Use the COPY instruction to copy application files from the host to the container.

"COPY app /app"

COPY <source> <destination>

Set Working Directory:

- Use the **WORKDIR** instruction to set the working directory inside the container.

"WORKDIR /app"

Define Default Command:

Use the **CMD** instruction to specify the default command to run when the container starts.

"**CMD** ["python", "main.py"]"

Install Dependencies (Optional):

Use the **RUN** instruction to execute commands inside the container.

'RUN pip install -r requirements.txt"

Building a Docker Image

1.Open Terminal:
Open a terminal or command prompt on your Windows machine.

2.Navigate to Project Directory:
Use the cd command to navigate to the directory containing your Dockerfile and application files.

3.Build Docker Image:
Run the following command to build the Docker image:
“docker build -t python-app “. -Replace “python-app” with your desired image name.

4.Wait for Build to Complete:
- Docker will execute the instructions in the Dockerfile and build the image.
- Once the build process is complete, you'll see a message indicating the successful creation of the image.

5.Verify Image:
- Use the following command to list all Docker images on your system and verify that your image is listed:

"docker images"

6.Image Optimization (Optional):
- You can optimize your Dockerfile to minimize image size and improve build speed.

Image Tagging (Optional):
- Tag your image with a version number or label for easy identification and version control.

“docker tag my-python-app my-python-app:v1.0”

Build and Run Example

Building a Docker Image
Title: *Building Images from Dockerfile*
Content:-*Docker images are built using the docker build command*.

docker build -t python-neelu .

Running a Docker Container

- Title: *Starting Containers from Images*
- Content:
  - *Docker containers are instances of Docker images.*

docker run python-neelu

Running a Docker Container

01 02 03 04

Run Docker Container

•After building the Docker image, you can run a container using the docker run command:

“docker run my-python-app”

Container Execution:

Docker will create a new container from the specified image and execute the default command defined in the Dockerfile.

Container Management:

•Use docker ps to list running containers and monitor their status.
•Use docker stop <container_id> to stop a running container when done.

Interactive Mode (Optional):

For interactive sessions, use the -i flag:

“docker run –t -i python-app”

Uploading Docker Image to Docker Hub

Create Docker Hub Account:	Visit the Docker Hub website and sign up for an account if you haven't already.
Create Docker Hub Repository:	Log in to Docker Hub and create a new repository with a unique name. You can make a public or private repository
Tag Image:	Tag your local Docker image with your Docker Hub username and repository name: “docker tag my-python-app username/my-python-app”
Login to Docker Hub:	Use the “ docker login” command to authenticate with Docker Hub
Push Image to Docker Hub:	Push the tagged image to Docker Hub: “docker push username/my-python-app”
Verify Upload:	Visit your Docker Hub repository on the web to verify that the image has been uploaded successfully.

Accessing Uploaded Image:

Others can pull your image from Docker Hub using the “docker pull” command:

“docker pull

```
NV C :\1python\jupyter notebook\internship\docker1> docker tag python-demo neelu1483/python-demo:v1.0
NV C :\1python\jupyter notebook\internship\docker1> docker push neelu1483/python-demo:v1.0
The push refers to repository [docker.io/neelu1483/python-demo:v1.0]
4f19f59a069e: Pushed
b5c2673d8f60: Pushed
41a9f98caaa5: Pushed
17b02461857a: Pushed
5e7745c5bee2: Pushed
3aff9f9c9f44: Pushed
e077e19b6682: Pushed
21e1c4948146: Pushed
68866beb2ed2: Pushed
e6e2ab10dba6: Pushed
0238a1790324: Pushed
v1.0: digest: sha256:f431caf42d81cd311f984bb9a921296581cc197f0fe135c28f639f2b6c524a65 size: 2632
```

```
NV C: \1python\jupyter notebook\internship\docker1> docker pull neelu1483/python-demo:v1.0
v1.0: Pulling from neelu1483/python-demo:v1.0
71215d55680c: Already exists
3cb8f9c23302: Already exists
5f899db30843: Already exists
567db630df8d: Already exists
d68cd2123173: Already exists
```

Some Docker Image Commands

```
Usage:  docker image COMMAND
Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  pull        Download an image from a registry
  push        Upload an image to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)

Run 'docker image COMMAND --help' for more information on a command.
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
python-demo   latest    18a99559afe4   2 minutes ago    1.17GB
python        latest    6cbe1053f244   2 weeks ago      1.02GB
```

THANK YOU

Fill in the missing part to log a table to WandB

Multiple choice

```
data = wandb.Table(columns=["input", "output"])
for batch in dataset:
    input_img, output_img = batch

    #Add input_img and output_img to the Table
    _____
wandb.log({"my_table": data})
```

Complete the code snippet to log a Pandas DataFrame column as a WandB scatter plot:

```
import pandas as pd
import wandb

df = pd.DataFrame({"x": [1, 2, 3], "y": [4, 5, 6]})
wandb.init()

_____
```

- wandb.log({"scatter": wandb.plot.scatter(df["x"], df["y"])})
- wandb.log_scatter("scatter", df["x"], df["y"])
- wandb.log({"scatter": wandb.Scatter(x=df["x"], y=df["y"])})

Assignment on transfer learning

In this assignment, you have to take a pretrained convnet and apply it in the tasks given below. Odd roll numbers must take inceptionnet-v1 and even roll numbers must take inceptionnet-v3.

1. Remove the last linear layer and replace it with appropriate linear layer(s) to train on the dataset given in this link. Your model must only train the last linear layer, thereby using the pretrained model. Perform the finetuning and testing by dividing the dataset into train-test.[1+2 marks]

2. Create a function to output the **saliency maps** corresponding to any 1 image from each class in the following two cases:
   a. Finetune only the last layer and test it.[2 marks]
   b. Re-train the entire network on the new dataset and test it.[2 marks]

3. Evaluate the performance of the finetuned and original network based on the recall and accuracy metrics.
4. Plot the training loss curve. Finally, write plausible explanation for the difference in metric values you obtained.[3 marks]