🧱 **What Is docker-compose?**

docker-compose lets you define and **run multiple containers** using a **single YAML file** (docker-compose.yml). It simplifies the orchestration of services.

✅ Perfect for:

• Running a Python app + Jupyter

• MLflow with PostgreSQL

• GenAI API + frontend

• Team sharing with consistent environments

✅ Step-by-Step Guide to Using **docker-compose**

🔹 **Step 1: Check if You Have docker-compose**

Most recent Docker Desktop installs it by default.

Run in terminal:

bash

CopyEdit

docker-compose --version

You should see something like:

mathematica

CopyEdit

Docker Compose version v2.20.0

If not, update Docker Desktop or install manually.

🔹 **Step 2: Create Project Folder**

bash

CopyEdit

mkdir genai-docker-compose

cd genai-docker-compose

Create your app file, e.g., app.py:

python

CopyEdit

# app.py

print("Hello from Docker Compose ML app!")

And a requirements.txt file:

txt

CopyEdit

# requirements.txt

numpy

🔹 **Step 3: Create Dockerfile**

Dockerfile

CopyEdit

# Dockerfile

FROM python:3.9

WORKDIR /app

COPY . .

RUN pip install --upgrade pip

RUN pip install -r requirements.txt

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

🔹 **Step 4: Create docker-compose.yml**

yaml

CopyEdit

# docker-compose.yml

version: '3.9'

services:

genai-app:

build: .

container\_name: genai\_ml\_app

volumes:

- .:/app

command: python app.py

🔹 **Step 5: Build and Run with Compose**

From the project directory:

bash

CopyEdit

docker-compose up --build

You should see:

python-repl

CopyEdit

Building genai-app

...

Hello from Docker Compose ML app!

🧹 **To Stop and Clean Up**

• To stop:  
 Ctrl+C in terminal  
 Then:  
 docker-compose down

💡 **Bonus: Extend to Jupyter or Multiple Services**

Want to run Jupyter Lab too? Here's an example service added:

yaml

CopyEdit

jupyter:

image: jupyter/scipy-notebook

ports:

- "8888:8888"

volumes:

- .:/home/jovyan/work

Now access Jupyter at http://localhost:8888!

✅ Summary

**Task**

**Command**

Run app

docker-compose up

Build and run

docker-compose up --build

Stop and clean

docker-compose down

🧱 **What Does docker build Do?**

The docker build command reads a file called Dockerfile and creates a **Docker image** from it.

Think of it like baking a cake:

• 🍰 **Dockerfile** = recipe

• 🧁 **Docker image** = baked cake (reusable)

• 👨‍🍳 docker build = the baking process

🛠 **Step-by-Step: Using docker build**

Let’s go step by step using a simple example.

✅ **Step 1: Create a Project Folder**

bash

CopyEdit

mkdir my-docker-app

cd my-docker-app

✅ **Step 2: Add Your App Code**

Create a simple Python script:

**app.py**

python

CopyEdit

print("Hello from Docker!")

✅ **Step 3: Create a Dockerfile**

Create a file named **Dockerfile** (no extension):

Dockerfile

CopyEdit

# Use an official Python base image

FROM python:3.11

# Copy the app.py file into the image

COPY app.py .

# Define the command to run the app

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

This Dockerfile says:

• Use Python 3.11

• Copy your Python script into the container

• Run the script when the container starts

✅ **Step 4: Run docker build**

Now you're ready to build the image:

bash

CopyEdit

docker build -t my-docker-app .

Let’s break it down:

• docker build: the command

• -t my-docker-app: gives your image a name ("tag")

• .: the location of the Dockerfile (here, current directory)

✅ **Step 5: See Your Image**

Run:

bash

CopyEdit

docker images

You should see something like:

perl

CopyEdit

REPOSITORY TAG IMAGE ID CREATED SIZE

my-docker-app latest 123abc456def X seconds ago 17MB

✅ **Step 6: Run Your Container**

bash

CopyEdit

docker run my-docker-app

You’ll see:

csharp

CopyEdit

Hello from Docker!

🎉 Your container ran successfully!

🔁 **Recap of docker build -t name .**

**Part**

**Meaning**

docker

The Docker command-line tool

build

Build a Docker image

-t name

Assign a name/tag to your image

.

The context — location of Dockerfile/code

—

🔖 **Significance of -t in docker build**

👇 **Syntax:**

bash

CopyEdit

docker build -t <name>:<tag> <path>

• -t stands for **tag**

• <name> is usually in the format:

◦ Just a name: myapp

◦ Or with your Docker Hub ID: harishanker/myapp

• <tag> is optional and usually used for versioning (defaults to latest)

🧠 **Why Use -t?**

✅ **1. Naming the image**

Without a name, the image will be hard to reference or reuse.

bash

CopyEdit

docker build -t my-first-image .

Now you can run it like this:

bash

CopyEdit

docker run my-first-image

✅ **2. Tagging versions**

You can version your image easily:

bash

CopyEdit

docker build -t harishanker/myapp:v1.0 .

You can later build another version:

bash

CopyEdit

docker build -t harishanker/myapp:v2.0 .

This is very useful for:

• Tracking changes

• CI/CD

• Pushing to Docker Hub

✅ **3. Pushing to Docker Hub**

When pushing:

bash

CopyEdit

docker push harishanker/myapp:v1.0

Docker Hub uses the name and tag to store and pull images.

✅ **4. Multiple Tags for Same Image**

You can tag the same image with multiple names:

bash

CopyEdit

docker tag myapp harishanker/myapp:latest

Then push both:

bash

CopyEdit

docker push harishanker/myapp:latest

📦 **Summary**

**Part**

**Meaning**

-t

Tag your image (name + optional version)

:

Separates the image name from the version/tag

**Example:**

bash

CopyEdit

docker build -t harishanker/myapp:v1.0 .

Means:  
🛠 Build an image called myapp  
👤 Belongs to user harishanker  
🏷 Version is v1.0

Push image to docker hub repo

🐳 **What You Need**

✅ Docker installed  
✅ Docker Hub account (Create one)  
✅ Logged into Docker from terminal: docker login  
✅ A built Docker image

🔁 **Overview of Steps**

**1 Login to Docker Hub**

**2 Tag your image (if not already)**

**3 Push your image**

**4 Verify on Docker Hub**

🧱 **Step-by-Step Guide**

✅ **Step 1: Login to Docker Hub**

Open your terminal and type:

bash

CopyEdit

docker login

• Enter your Docker ID and password

• If you use **2FA**, use a Personal Access Token

✅ **Step 2: Tag Your Image**

Docker images must be tagged with your Docker Hub username.

👇 **Format:**

bash

CopyEdit

docker tag <local-image-name> <dockerhub-username>/<repo-name>:<tag>

🔁 **Example:**

You built a local image:

bash

CopyEdit

docker build -t myapp .

Now tag it:

bash

CopyEdit

docker tag myapp harishanker/myapp:v1

You can also use :latest instead of a version:

bash

CopyEdit

docker tag myapp harishanker/myapp:latest

✅ **Step 3: Push the Image**

Use the docker push command:

bash

CopyEdit

docker push harishanker/myapp:v1

This uploads the image to your Docker Hub repo.

✅ Output will show layers being uploaded like this:

css

CopyEdit

The push refers to repository [docker.io/harishanker/myapp]

...

v1: digest: sha256:xxxx size: xxx

✅ **Step 4: Verify on Docker Hub**

Go to: https://hub.docker.com/repositories

You’ll see:

• Your repository (e.g., harishanker/myapp)

• Tags like v1 or latest

🧽 **Optional: Clean Up**

If you want to remove local images:

bash

CopyEdit

docker rmi myapp

docker rmi harishanker/myapp:v1

✅ **Summary**

**Step**

**Command**

Login

docker login

Tag

docker tag myapp harishanker/myapp:v1

Push

docker push harishanker/myapp:v1

Verify

Visit Docker Hub website

Run dock image

🧱 **Step-by-Step: How to Run a Docker Image**

✅ **Step 1: Check Available Images**

List images on your system:

bash

CopyEdit

docker images

Example output:

nginx

CopyEdit

REPOSITORY TAG IMAGE ID CREATED SIZE

myapp latest abc12345defg 5 minutes ago 130MB

Let’s say your image is called myapp.

✅ **Step 2: Run the Image**

Use the docker run command:

bash

CopyEdit

docker run myapp

This creates and runs a **container** from the image.

🛠 **Common Options**

**Option**

**Purpose**

-it

Interactive mode (for terminal-based apps)

--rm

Automatically remove container when it exits

-d

Run in background (detached mode)

--name mycontainer

Assign a name to your container

-p 5000:5000

Map container port to host port (for web apps)

-v host:container

Mount a volume (for saving data)

✅ **Example 1: Run and See Output**

For a basic image that just prints something:

bash

CopyEdit

docker run myapp

✅ **Example 2: Run a Web App**

If your app runs on port 5000:

bash

CopyEdit

docker run -p 5000:5000 myapp

Then open your browser and go to:  
http://localhost:5000

✅ **Example 3: Run in Background (Detached Mode)**

bash

CopyEdit

docker run -d -p 5000:5000 myapp

To check running containers:

bash

CopyEdit

docker ps

To stop it:

bash

CopyEdit

docker stop <container\_id>

🧼 **Optional: Clean Up**

• List all containers:  
bash  
CopyEdit  
  
  
docker ps -a

•

• Stop a container:  
bash  
CopyEdit  
  
  
docker stop <container\_id>

•

• Remove a container:  
bash  
CopyEdit  
  
  
docker rm <container\_id>

•

🧪 **Want to Try a Quick Demo?**

Here’s a one-liner to run an official Python container and enter it:

bash

CopyEdit

docker run -it python:3.11

You’ll enter a Python shell inside the container 🐍