

TÍNH TOÁN HIỆU SUẤT CAO

Nhóm 1



**Triển khai hệ thống phân loại ảnh trên
Docker Official Image**

CẤU TRÚC BÀI BÁO CÁO

01 GIỚI THIỆU VỀ HPC VÀ ĐỀ TÀI

02 CƠ SỞ LÝ THUYẾT

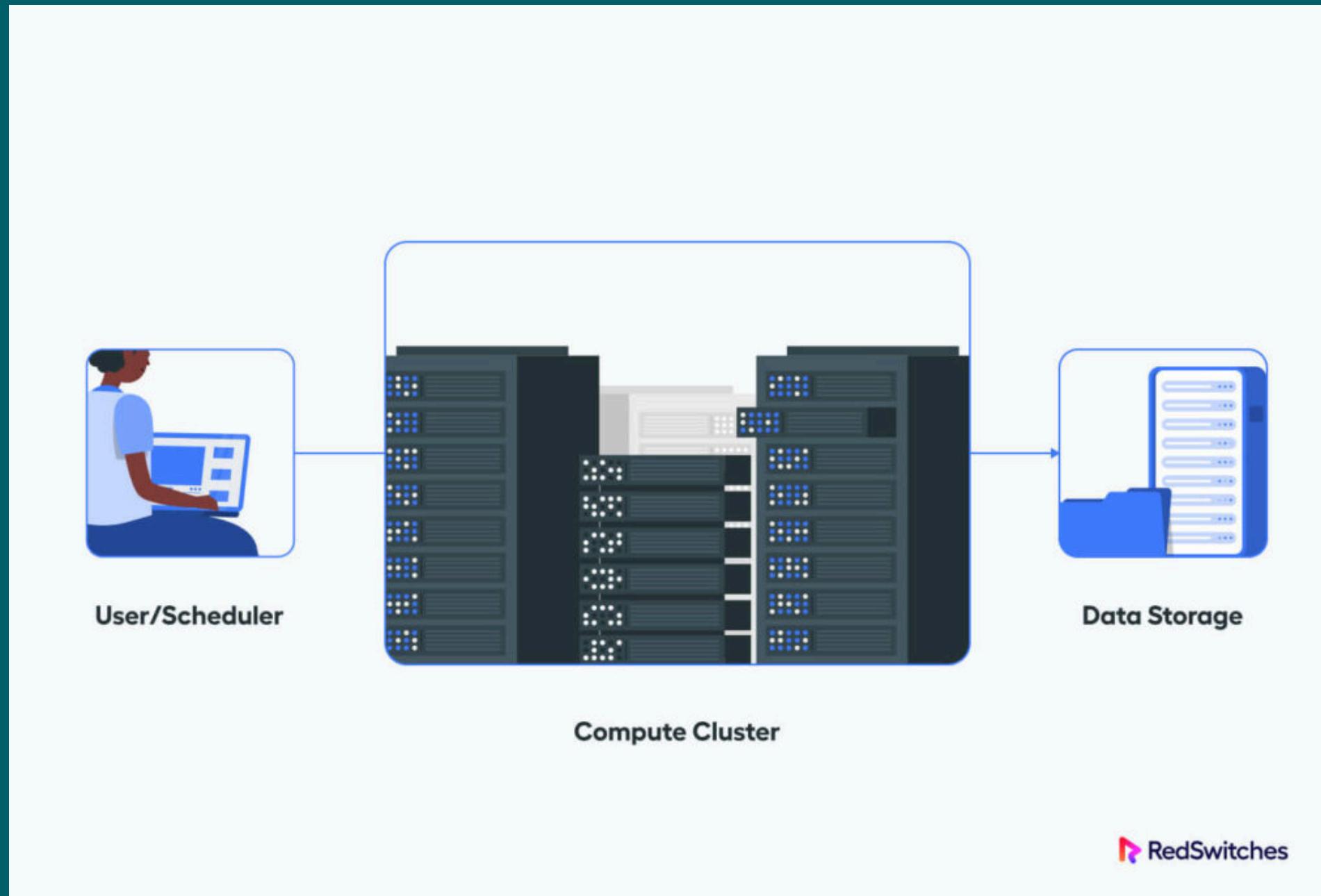
03 TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

04 BÀN LUẬN VÀ ĐÁNH GIÁ

05 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

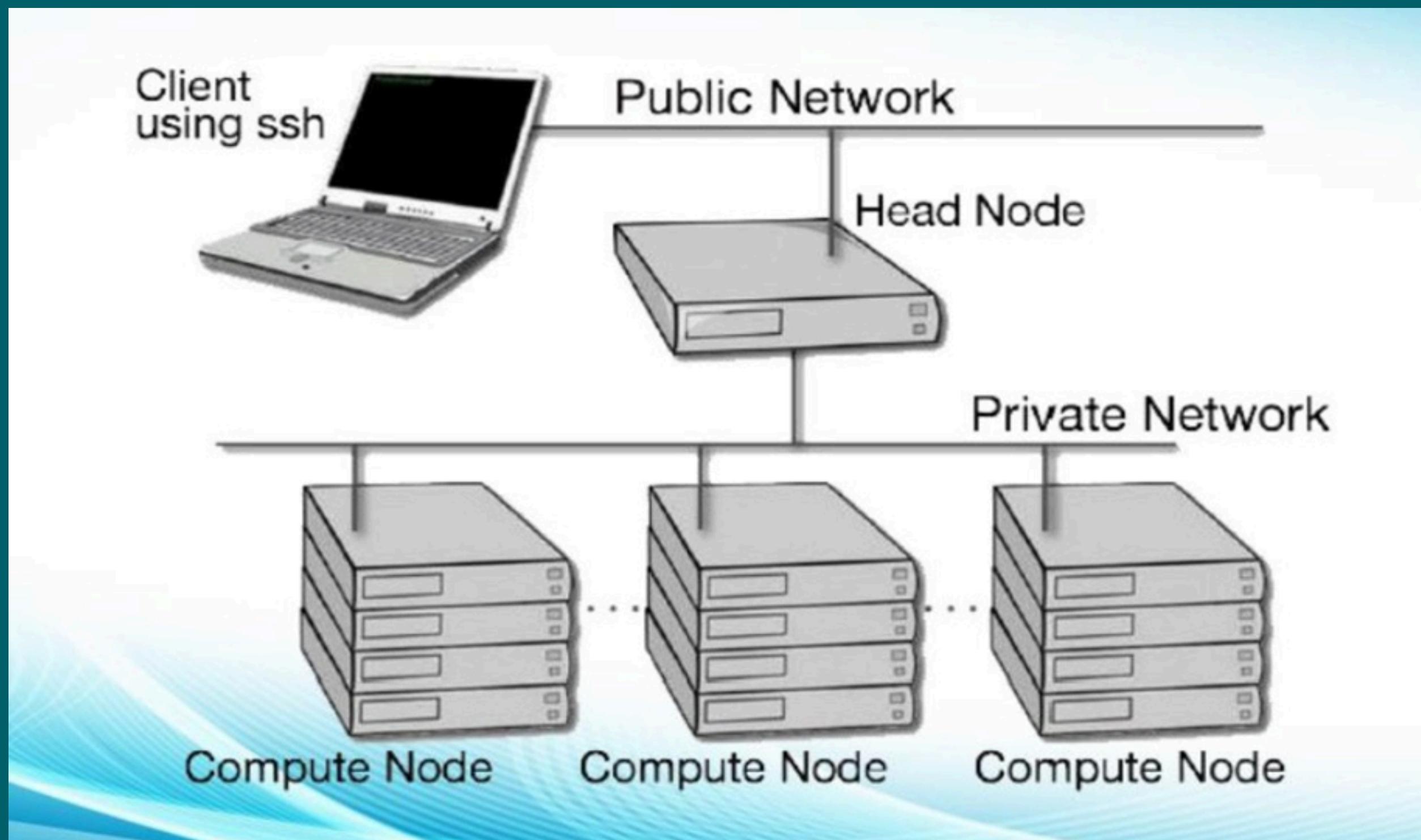
1. GIỚI THIỆU TÍNH TOÁN HIỆU SUẤT CAO (HPC) VÀ DỰ ÁN THỰC HIỆN

HỆ THỐNG HPC LÀ GÌ



- Hệ thống HPC: gồm nhiều máy tính với các CPU, GPU, RAM,...
- Hệ thống lưu trữ: Dùng để lưu trữ dữ liệu lâu dài của hệ thống, bao gồm ổ đĩa cứng (HDD) và ổ đĩa đám mây (SSD).
- Mạng kết nối: Cầu nối cho việc truyền tải dữ liệu giữa các nút
- Phần mềm và thư viện: Bao gồm các phần mềm và thư viện phần mềm như hệ điều hành, công cụ phân tích dữ liệu lớn

MÔ HÌNH TRIỀN KHAI HPC

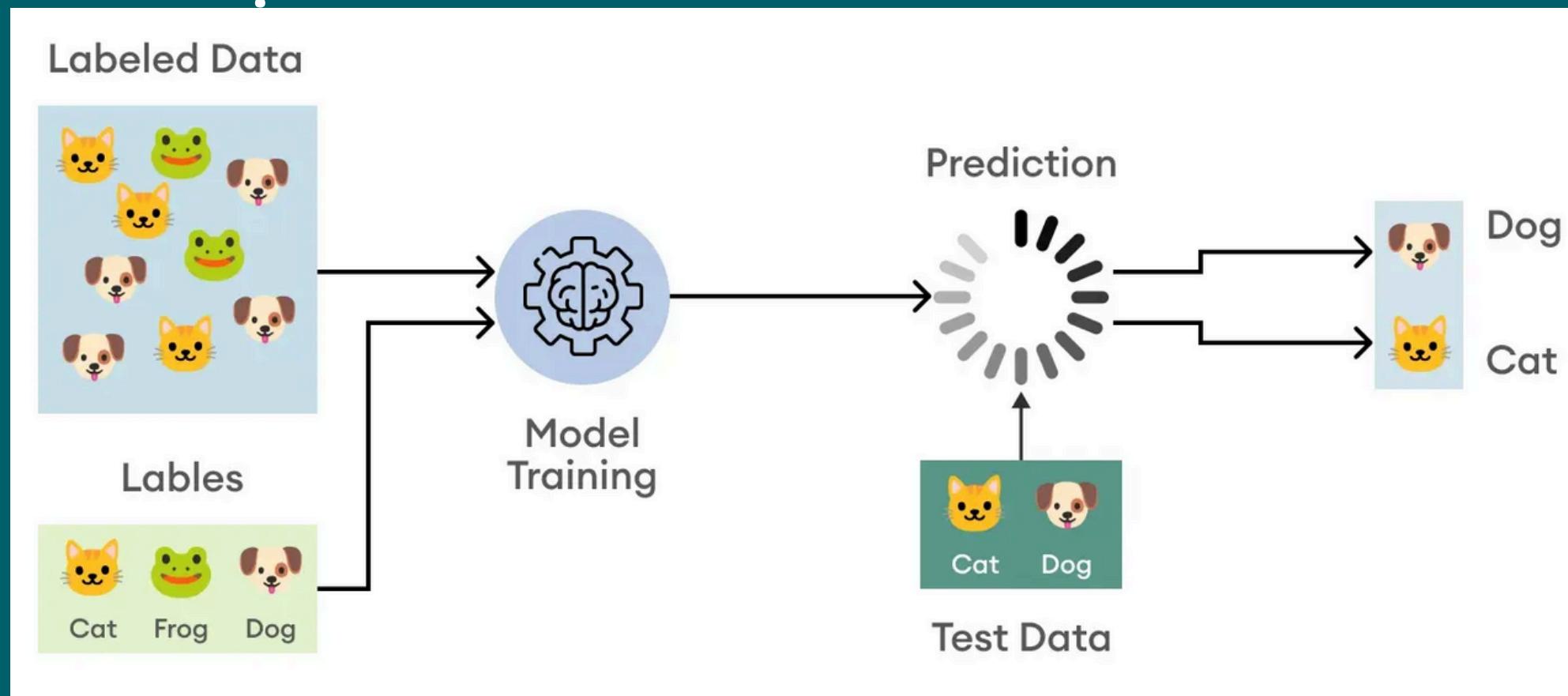


ỨNG DỤNG CỦA HPC



GIỚI THIỆU ĐỀ TÀI

Phân loại ảnh

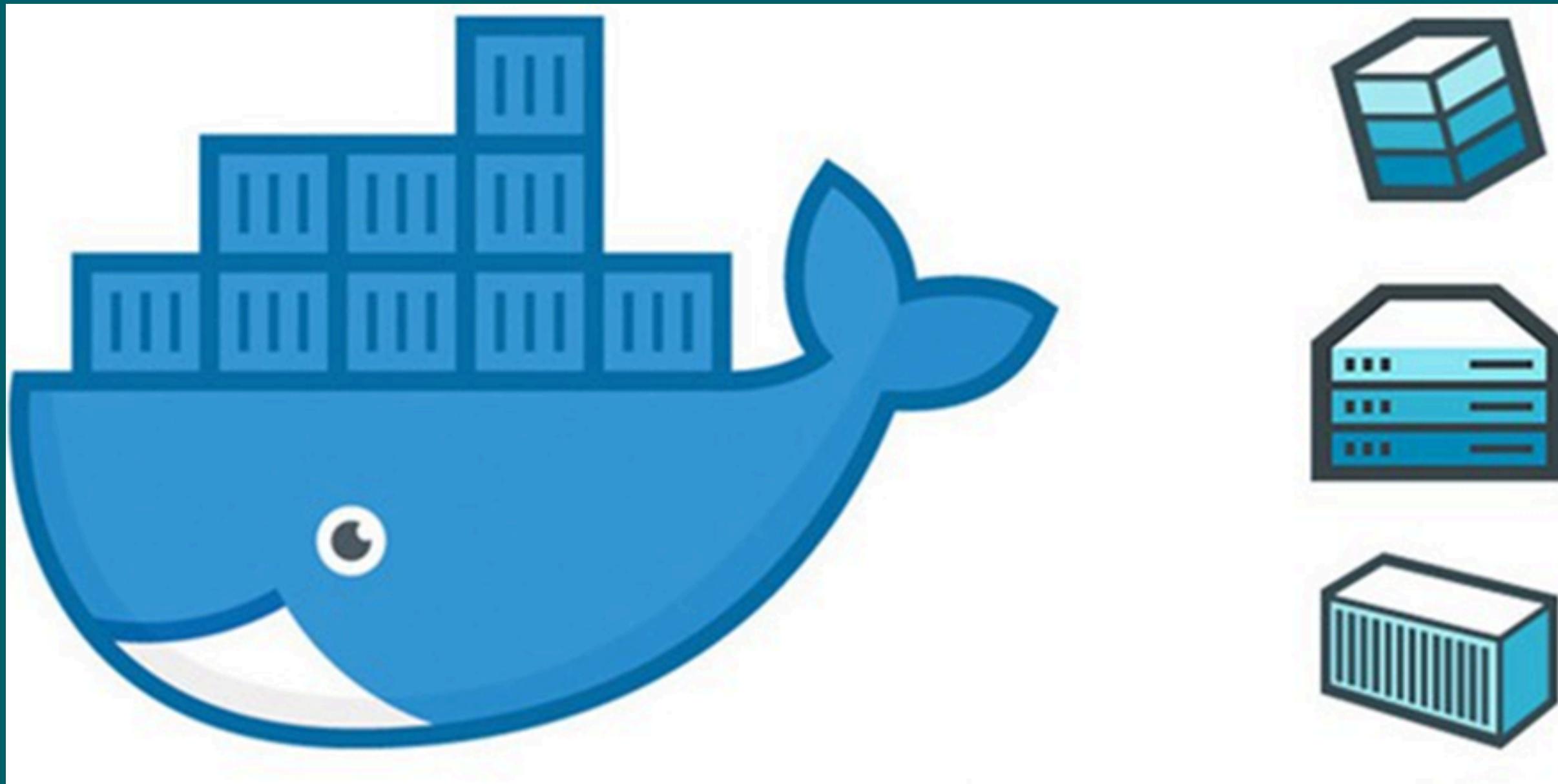


Mục đích

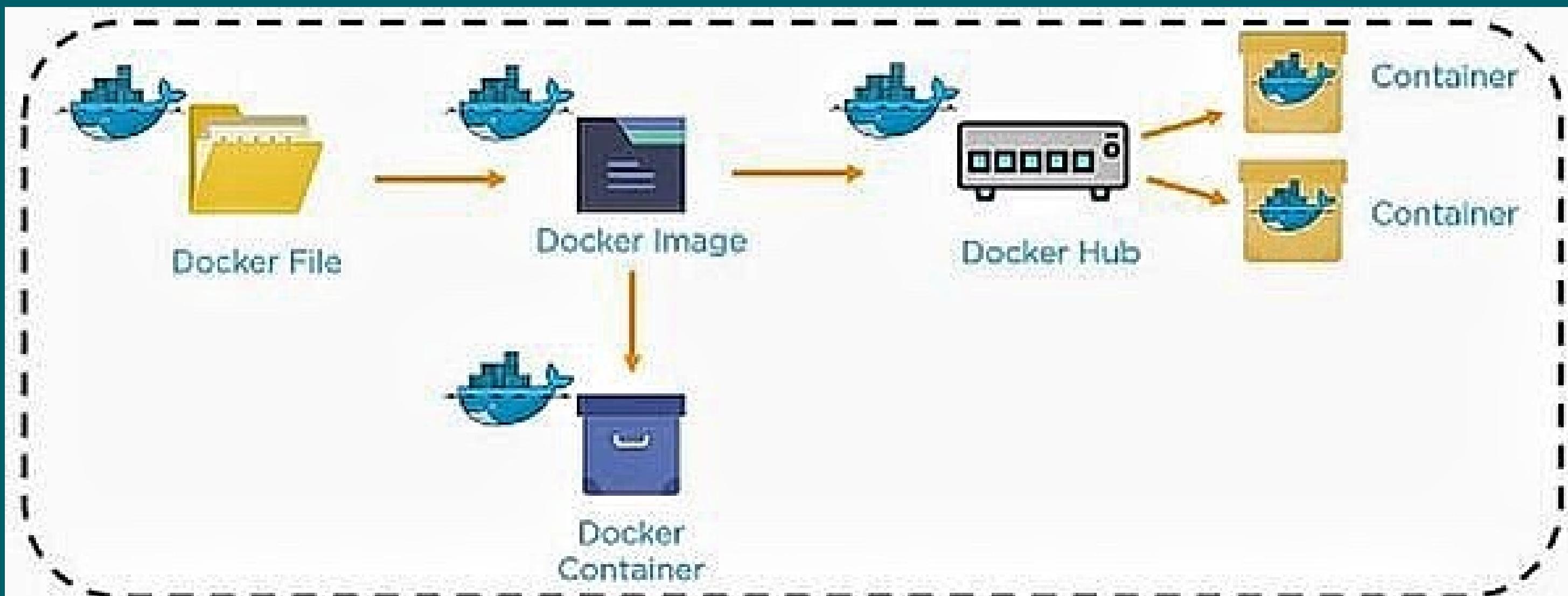
- Triển khai, xây dựng hệ thống phân loại ảnh sử dụng Docker và các Framework của Python như PyTorch, FastAPI.
- Đánh giá hiệu quả của hệ thống này trên các tập dữ liệu ảnh tiêu chuẩn
- Phân tích các yếu tố ảnh hưởng đến hiệu quả của hệ thống, từ đó đề xuất các giải pháp nhằm cải thiện hiệu quả

2. CƠ SỞ LÝ THUYẾT

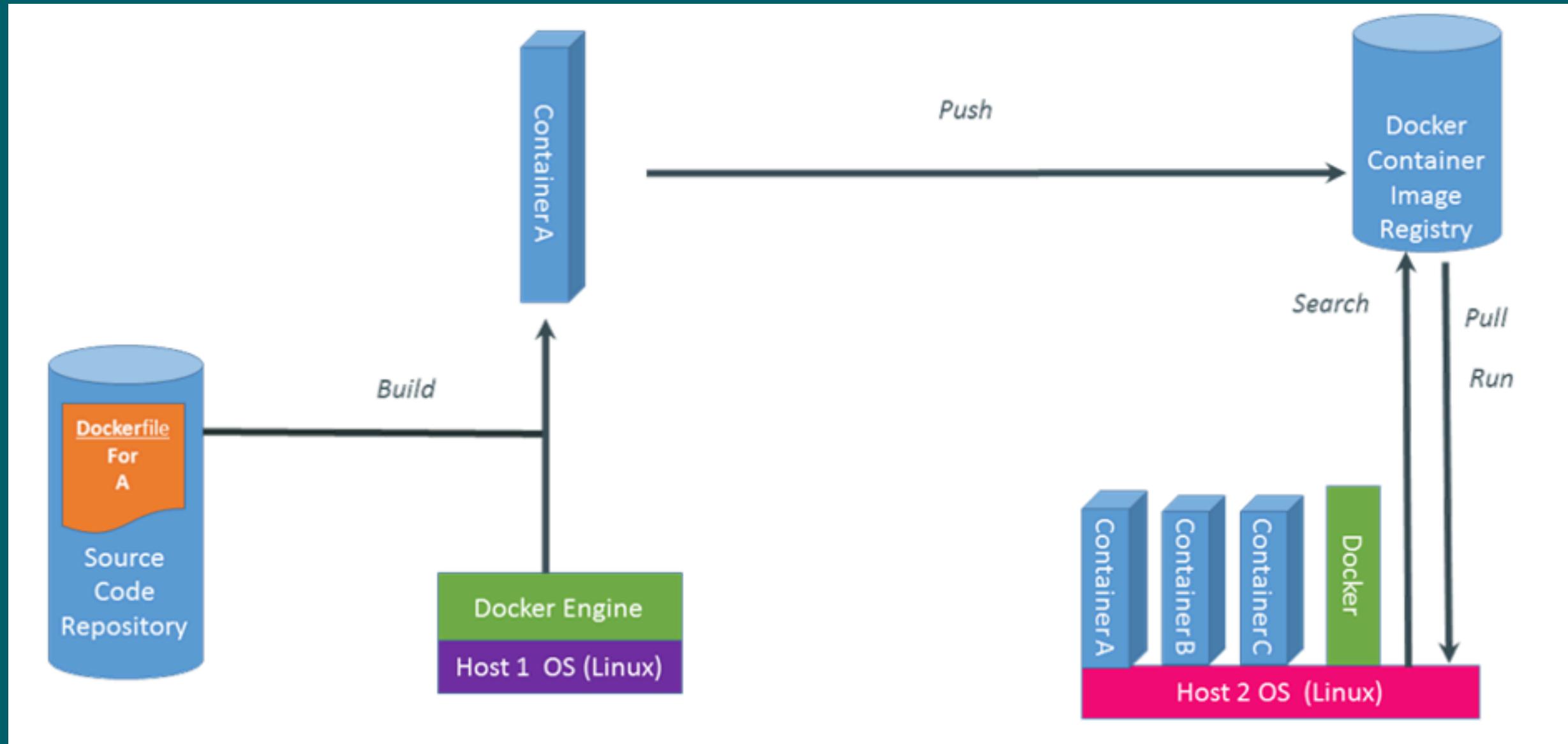
KHÁI NIỆM VỀ DOCKER



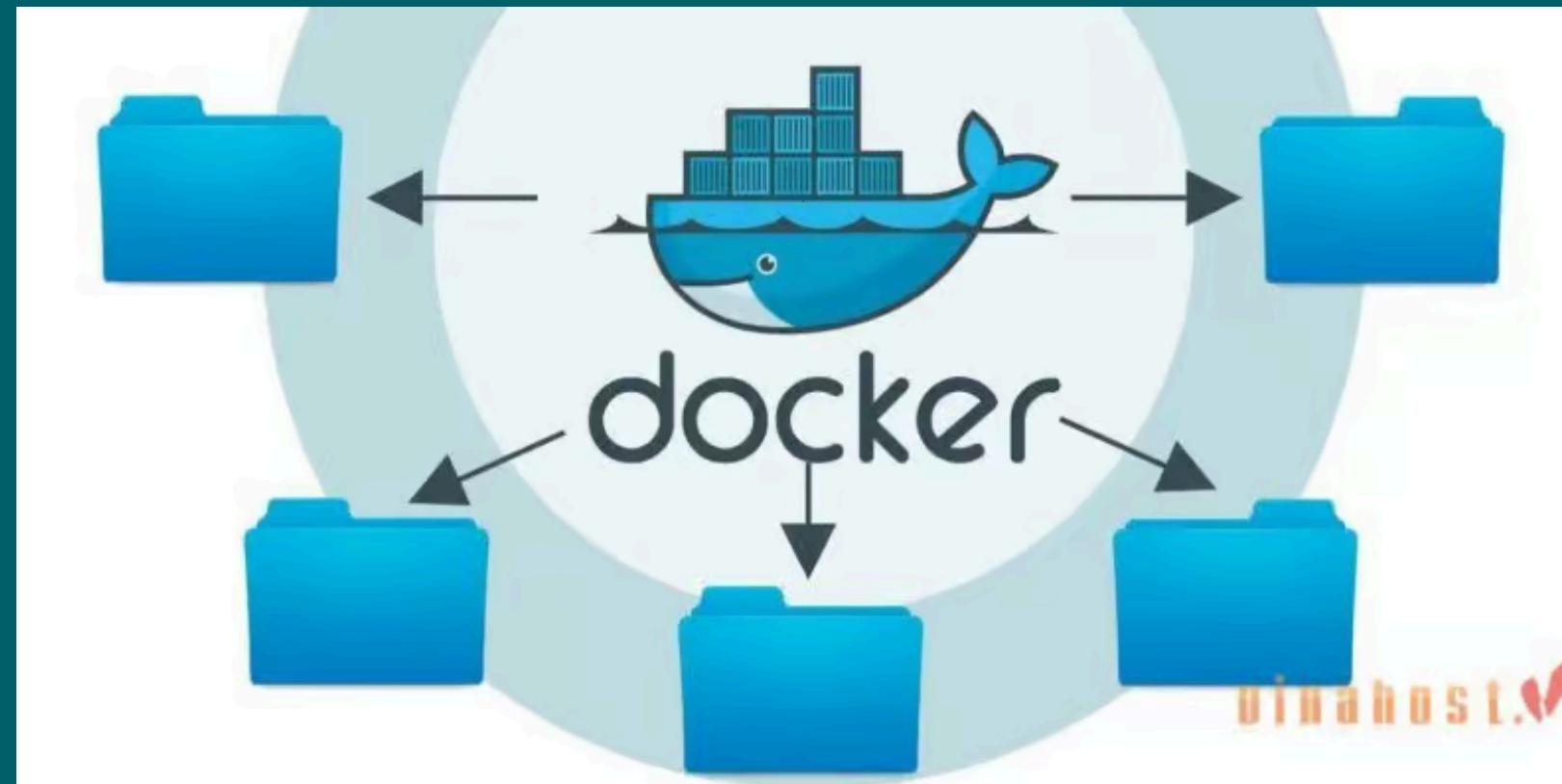
CẤU TRÚC CƠ BẢN CỦA DOCKER



CÁCH THỨC VÀ QUY TRÌNH HOẠT ĐỘNG CỦA DOCKER



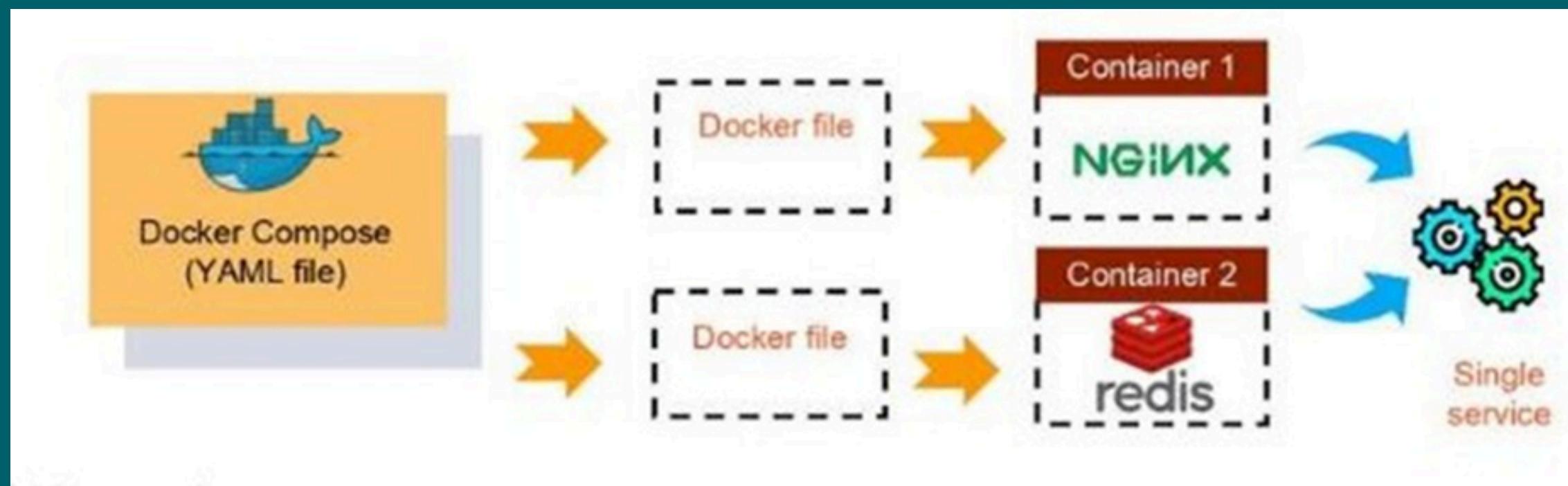
LỢI ÍCH KHI SỬ DỤNG DOCKER



- Nhanh và Nhẹ, Dễ Vận Chuyển:
- Dễ Dàng Mở Rộng
- Giảm Chi Phí Cơ Sở Hạ Tầng:
- Tiện Lợi và Tiết Kiệm Thời Gian
- Quản Lý Phiên Bản và Tái Sử Dụng

KHÁI NIỆM VỀ DOCKER COMPOSE

Docker Compose là một công cụ giúp tổ chức và kết hợp nhiều images có sẵn trên Docker Hub. Nó cho phép liên kết các Container riêng lẻ với nhau, tạo điều kiện để xử lý đồng thời nhiều nhiệm vụ.



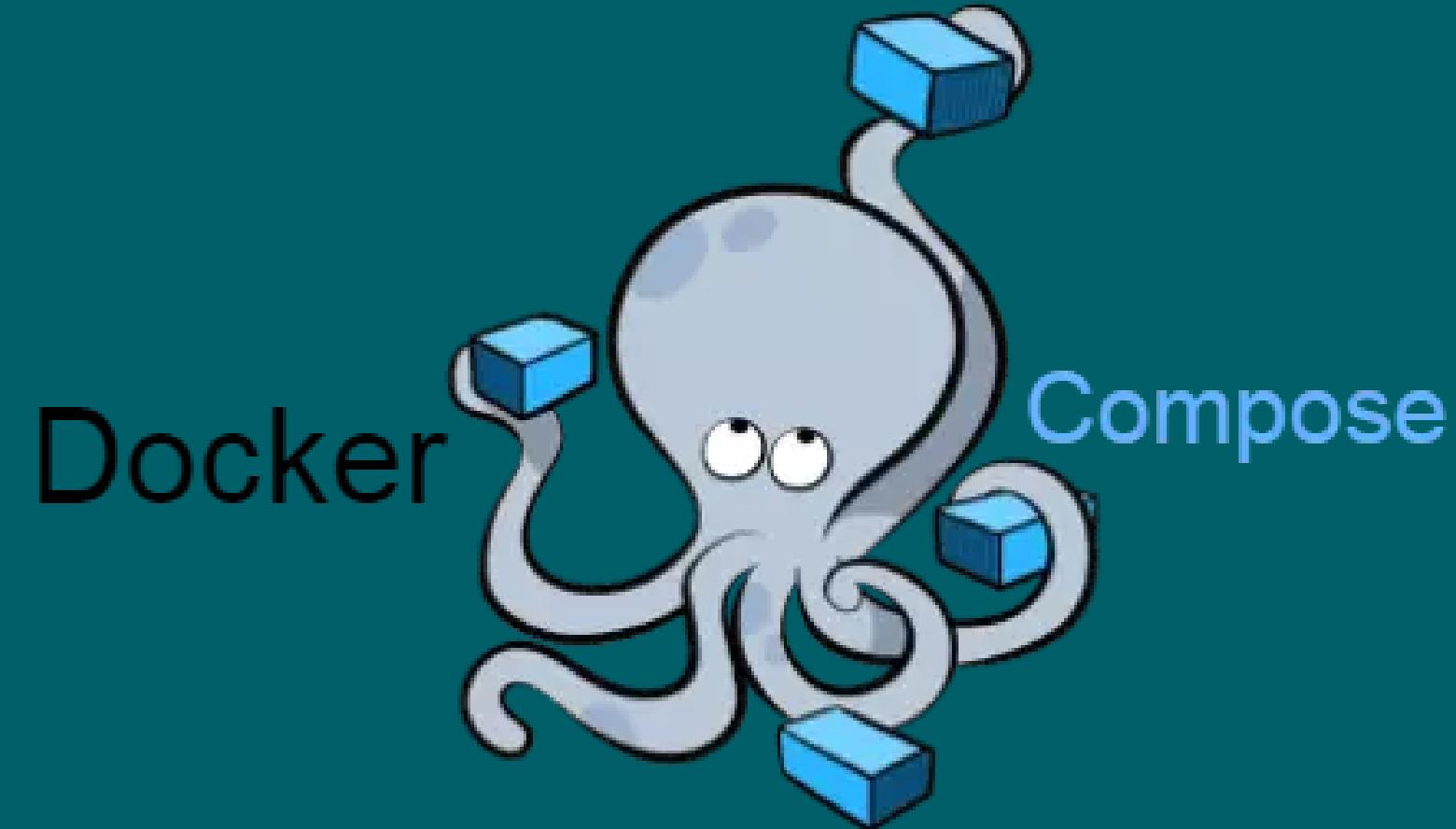
CÁCH THỨC SỬ DỤNG DOCKER COMPOSE

- Tạo một file docker-compose.yml để định nghĩa các version, service cần thiết để nó có thể cùng hoạt động mà chạy application
- Sử dụng lệnh docker-compose up -d để khởi tạo và chạy chương trình.
- Sử dụng lệnh docker-compose down để dừng và xóa các dịch vụ khi không cần thiết nữa.

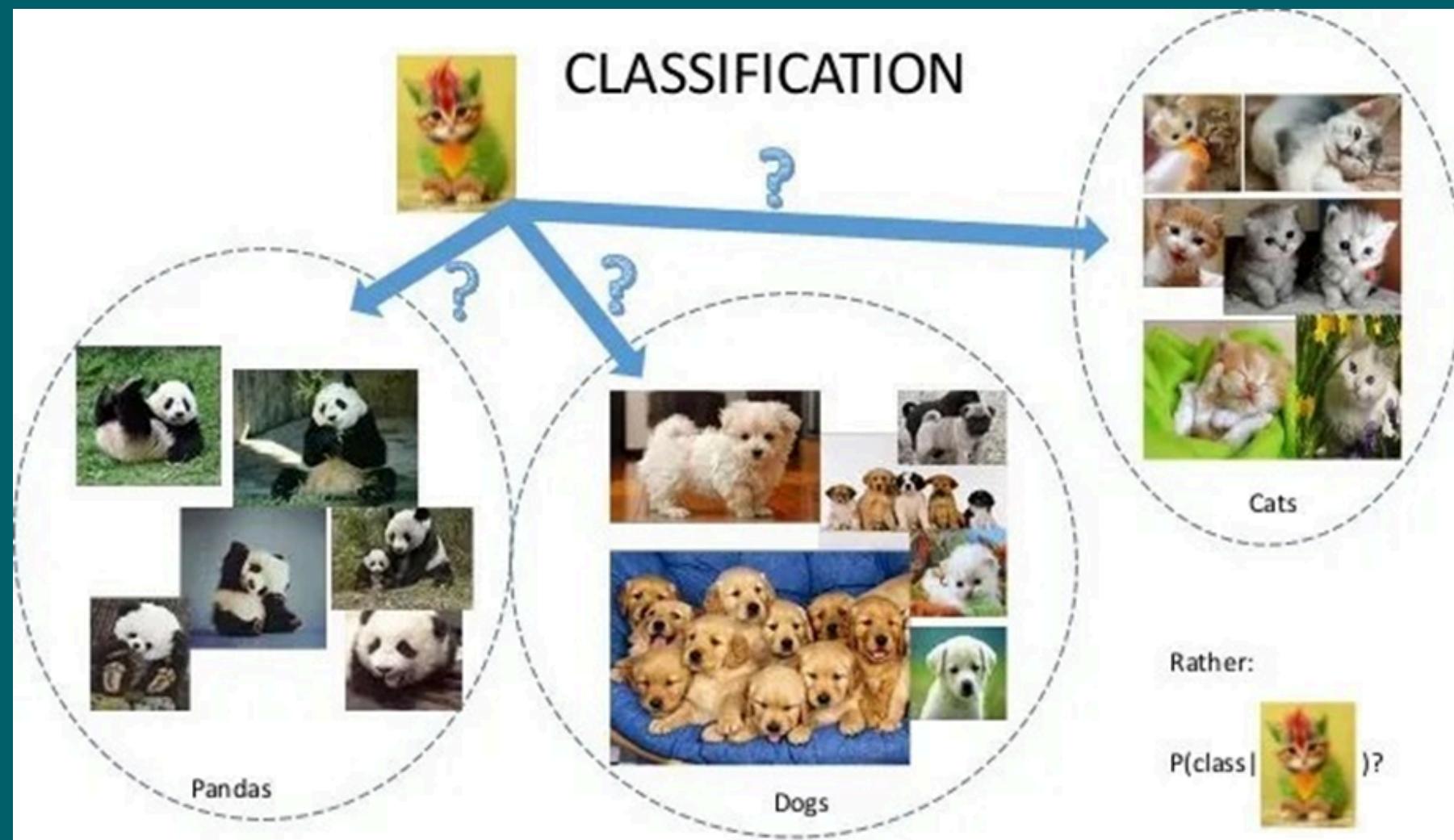
```
version: '3.8'
services:
  webapp:
    build: ./dir
    ports:
      - "5000:80"
    volumes:
      - webapp-volume:/var/lib/webapp
    depends_on:
      - db
  db:
    image: postgres:9.4
    volumes:
      - db-volume:/var/lib/postgresql/data
volumes:
  webapp-volume:
    external: true
  db-volume:
    driver: local
networks:
  app-network:
    driver: bridge
```

LỢI ÍCH KHI SỬ DỤNG DOCKER COMPOSE

- Quản lý đa container dễ dàng
- Tích hợp dễ dàng
- Dễ dàng xây dựng và phát triển ứng dụng
- Tích hợp với CI/CD
- Giảm rủi ro

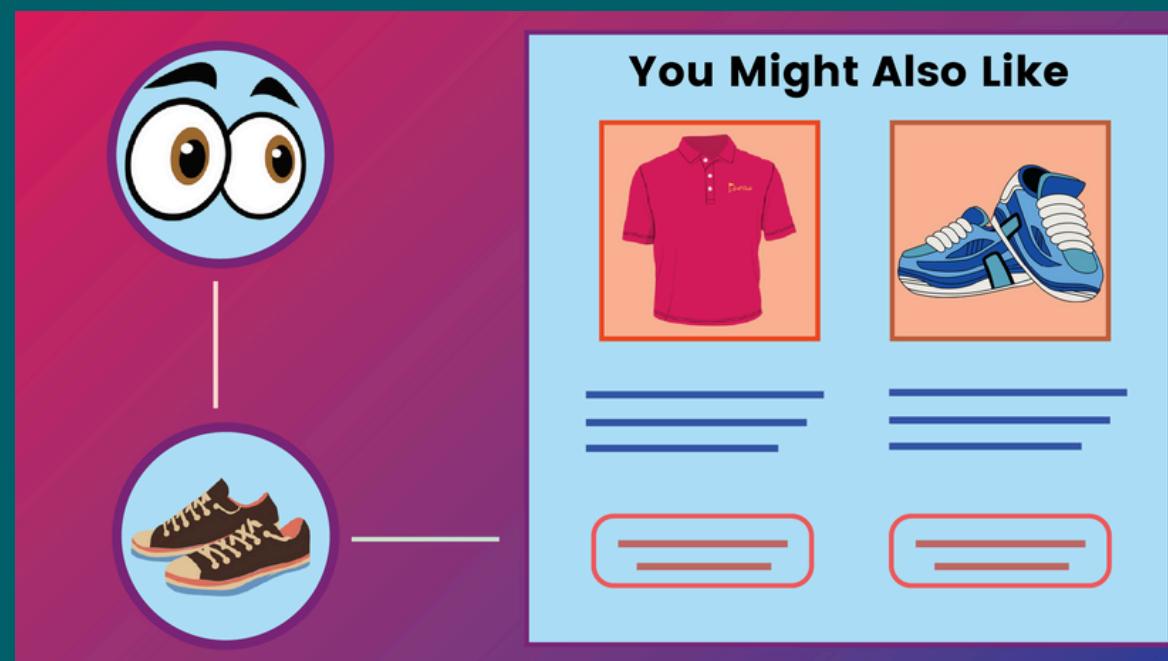
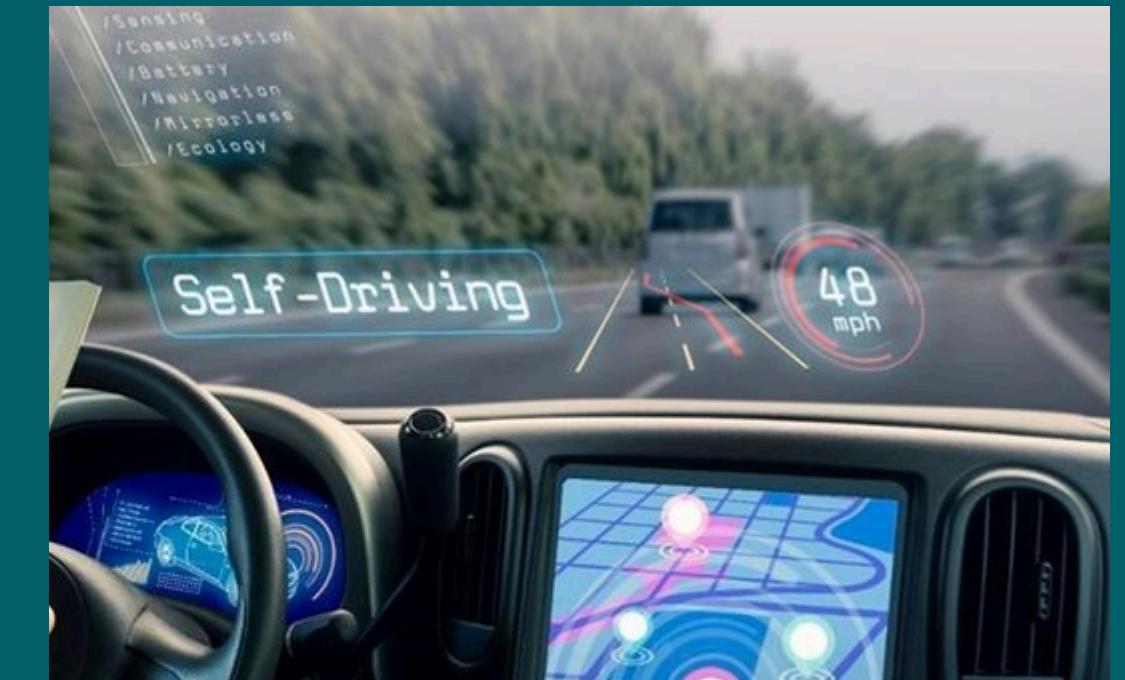
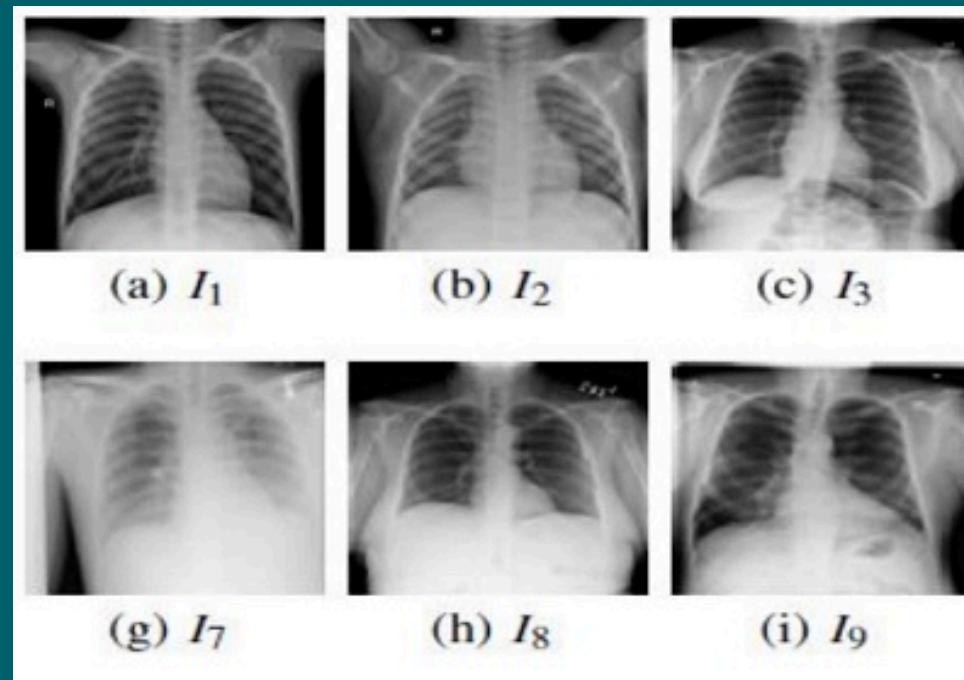


PYTHON VÀ BÀI TOÁN PHÂN LOẠI ẢNH



- Python là ngôn ngữ lập trình bậc cao với cú pháp đơn giản, dễ đọc, dễ học
- Phân loại hình ảnh là một bài toán với mục tiêu từ hình ảnh đầu vào, sử dụng các kỹ thuật học máy để nhận diện và cho ra được kết quả đầu ra là một bức ảnh với label cụ thể

ỨNG DỤNG THỰC TẾ CỦA BÀI TOÁN PHÂN LOẠI ẢNH



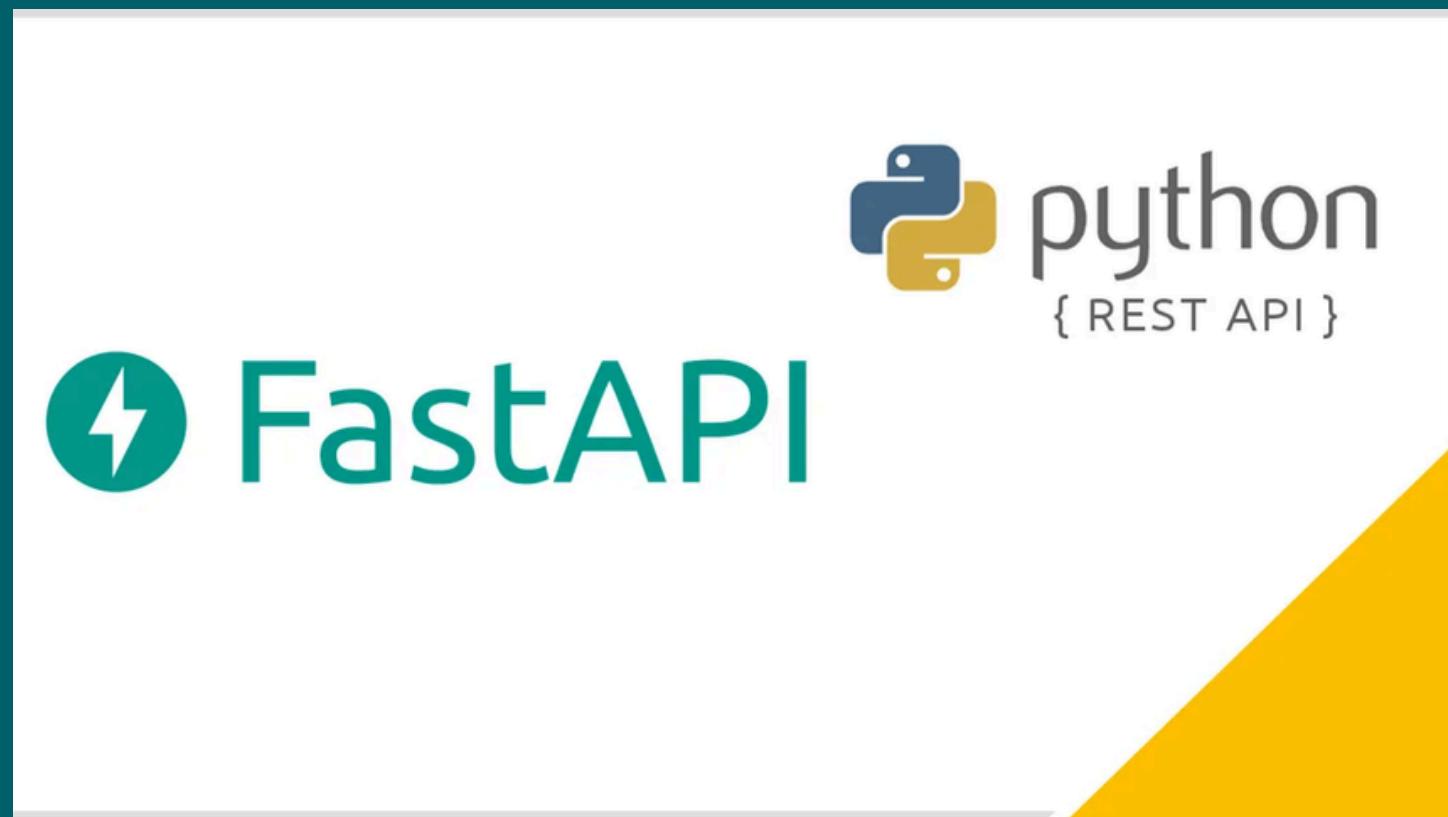
FRAMEWORK PYTORCH



Một số đặc điểm nổi bật của Pytorch

- **Tensor:** Là đơn vị cơ bản trong PyTorch, có thể chạy trên GPU để tăng tốc độ tính toán
- **Dynamic Computation Graphs:** PyTorch sử dụng cơ chế đồ thị tính toán động
- **Autograd:** Cung cấp Hệ thống tự động tính đạo hàm
- **Modularity:** PyTorch cung cấp một API đơn giản và mô-đun
- **Integration:** Dễ dàng tích hợp với các thư viện và công cụ khác trong hệ sinh thái Python

FRAMEWORK FASTAPI



Một số ưu điểm nổi bật của FastAPI:

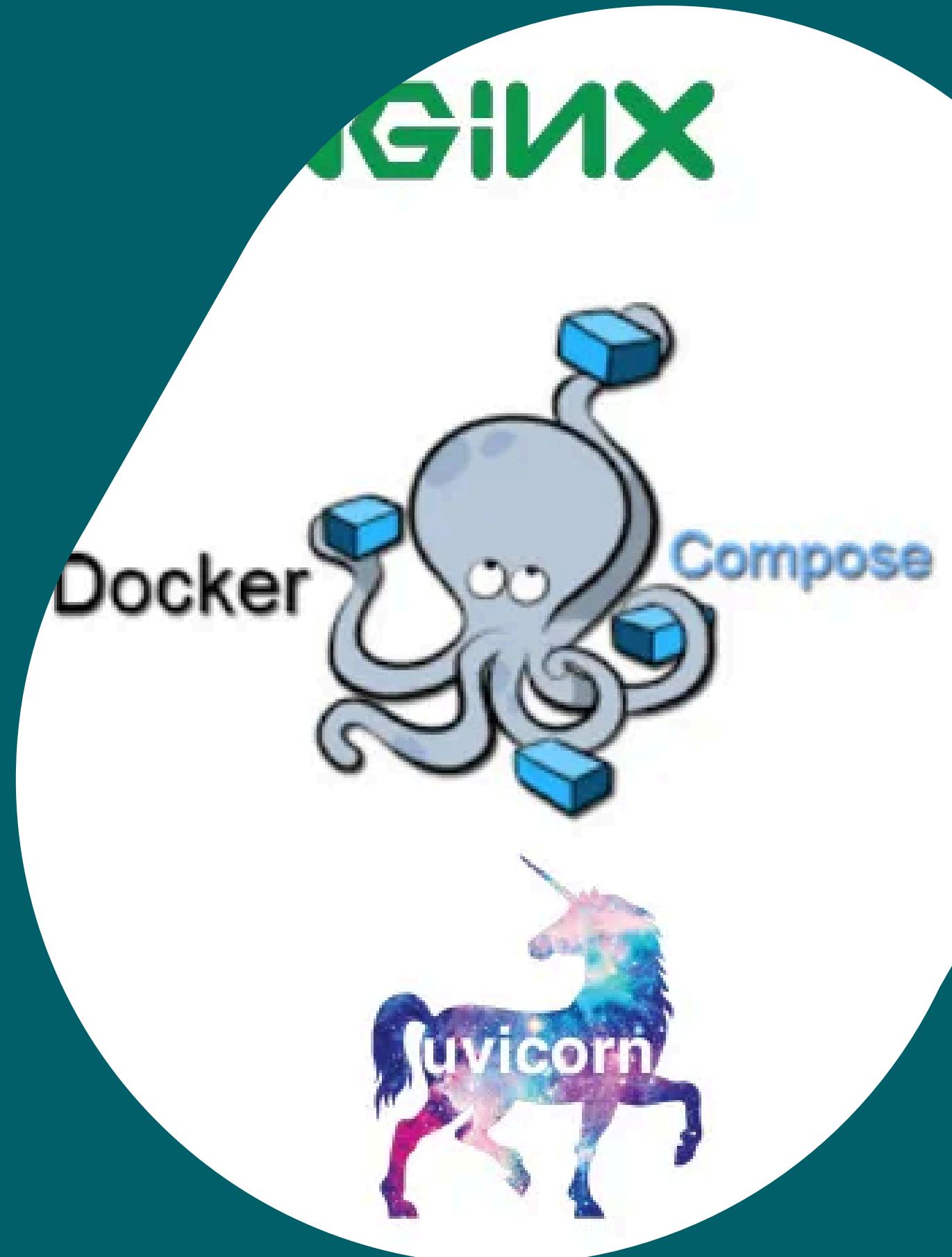
- **Hiệu suất cao**
- **Xử lý bất đồng bộ**
- **Tự động tạo tài liệu API**
- **Dễ dàng sử dụng và viết mã**
- **Hỗ trợ kiểm tra kiểu dữ liệu**
- **Tích hợp tốt với các công cụ khác**
- **Bảo mật**
- **Tính mở rộng và mô-đun**
- **Hỗ trợ quốc tế hóa**

LỢI ÍCH KHI TRIỂN KHAI HỆ THỐNG PHÂN LOẠI HÌNH ẢNH SỬ DỤNG FASTAPI TRÊN DOCKER COMPOSE

Sử dụng FastAPI, ta có thể tạo ra các endpoint API để nhận đầu vào hình ảnh từ người dùng và trả về kết quả phân loại

Docker cho phép đóng gói toàn bộ ứng dụng, bao gồm cả FastAPI và các dependencies, vào một container duy nhất

Bằng cách định nghĩa và quản lý các container trong một tệp YAML duy nhất, Docker Compose giúp đơn giản hóa quá trình triển khai và mở rộng



3. TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

Cấu trúc
source code

```
PROBLEMS DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
● PS C:\Users\84898\Desktop\Image Classification> dir

Directory: C:\Users\84898\Desktop\Image Classification

Mode                LastWriteTime         Length Name
----                -----          -----
d-----              5/26/2024 11:46 AM            config
d-----              5/26/2024 12:30 PM            data
d-----              5/26/2024 11:46 AM            logs
d-----              5/26/2024 11:46 AM            middleware
d-----              5/26/2024 11:46 AM            models
d-----              5/26/2024 11:46 AM            routes
d-----              5/26/2024 11:46 AM            schemas
d-----              5/26/2024 11:46 AM            utils
d-----              5/26/2024 2:10 PM             __pycache__
○ -a---- 5/26/2024 12:36 PM           625 .dockerignore
----- 3/26/2024 3:16 PM            286 app.py
-a---- 5/26/2024 5:35 PM            667 compose.yaml
-a---- 5/26/2024 5:35 PM           1553 Dockerfile
-a---- 5/26/2024 12:36 PM           826 README.Docker.md
----- 3/30/2024 6:38 PM            90 requirements.txt
-a---- 5/26/2024 12:28 PM           111 server.py
```

TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

Sử dụng Docker init

```
PROBLEMS DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
● PS C:\Users\84898\Desktop\Image Classification> docker init
Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
- .dockerignore
- Dockerfile
- compose.yaml
- README.Docker.md

Let's get started!
? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.9) 3.10.0
? What version of Python do you want to use? 3.10.0
? What port do you want your app to listen on? (8000)
? What port do you want your app to listen on? 8000
? What is the command to run your app? (uvicorn 'app:app' --host=0.0.0.0 --port=8000) uvicorn? What is the command to run your app? uvicorn app:app --reload=True --host=127.0.0.1 --port=8000

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✓ Your Docker files are ready!

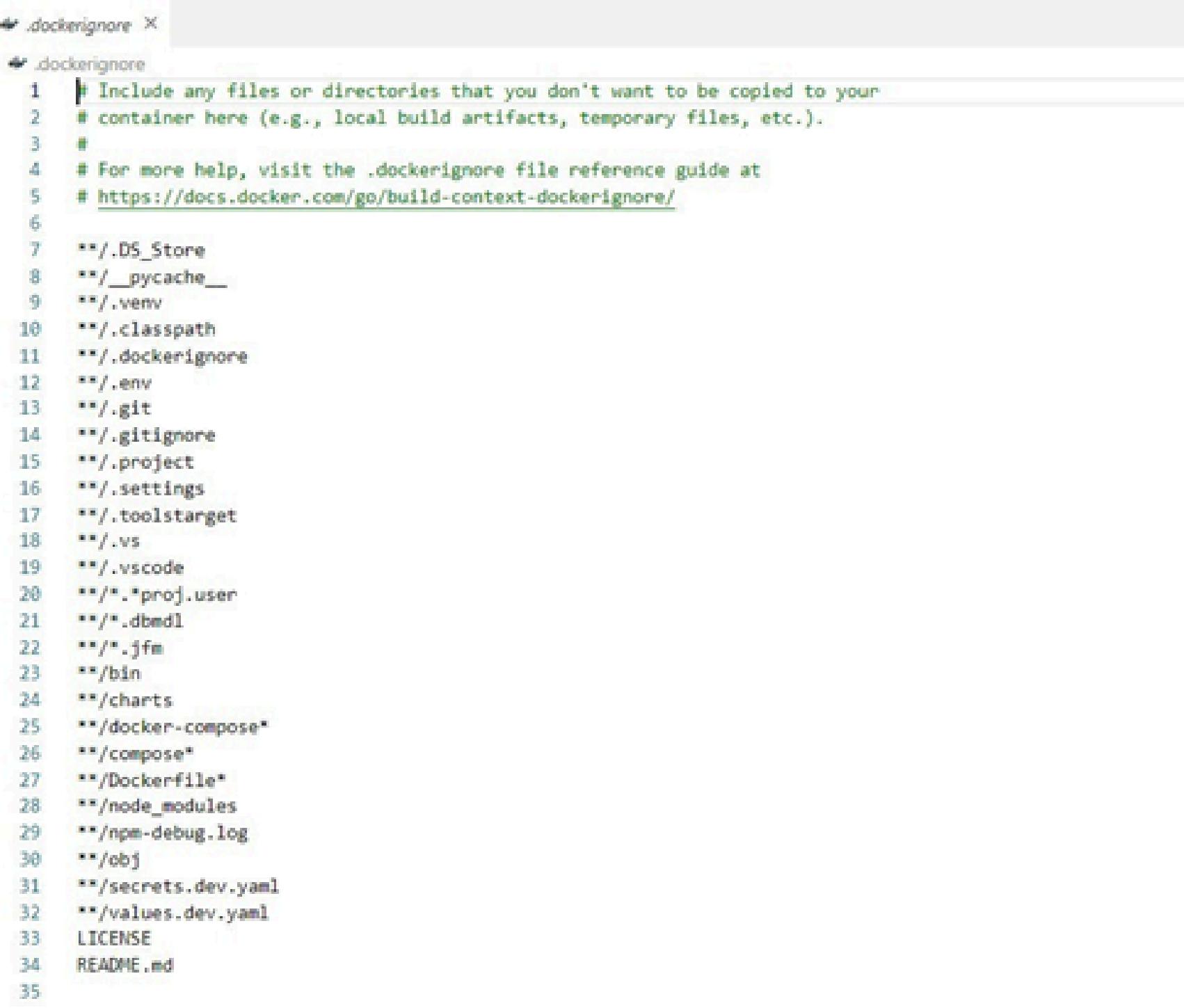
Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:8000
```

TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

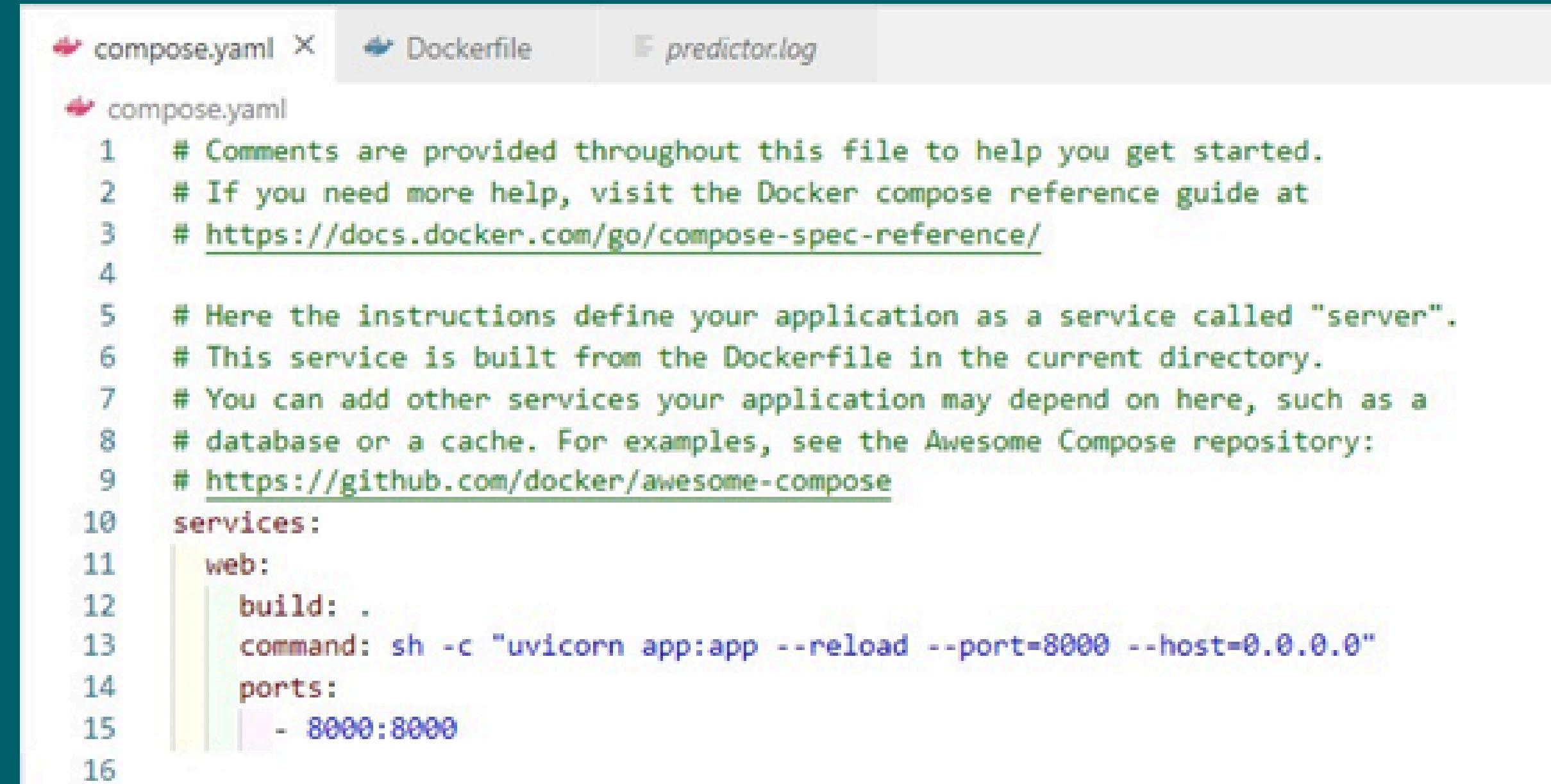
File .dockerignore



```
# .dockerignore
# Include any files or directories that you don't want to be copied to your
# container here (e.g., local build artifacts, temporary files, etc.).
#
# For more help, visit the .dockerignore file reference guide at
# https://docs.docker.com/go/build-context-dockerignore/
#
**/.DS_Store
**/__pycache__
**/.venv
**/.classpath
**/.dockerignore
**/.env
**/.git
**/.gitignore
**/.project
**/.settings
**/.toolstarget
**/.vs
**/.vscode
**/*.*proj.user
**/*.dbmdl
**/*.jfm
**/bin
**/charts
**/docker-compose*
**/compose*
**/Dockerfile*
**/node_modules
**/npm-debug.log
**/obj
**/secrets.dev.yaml
**/values.dev.yaml
LICENSE
README.md
```

TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

File compose.yaml



The screenshot shows a code editor window with three tabs: 'compose.yaml' (selected), 'Dockerfile', and 'predictor.log'. The 'compose.yaml' tab displays the following YAML configuration:

```
compose.yaml
1 # Comments are provided throughout this file to help you get started.
2 # If you need more help, visit the Docker compose reference guide at
3 # https://docs.docker.com/go/compose-spec-reference/
4
5 # Here the instructions define your application as a service called "server".
6 # This service is built from the Dockerfile in the current directory.
7 # You can add other services your application may depend on here, such as a
8 # database or a cache. For examples, see the Awesome Compose repository:
9 # https://github.com/docker/awesome-compose
10 services:
11   web:
12     build: .
13     command: sh -c "uvicorn app:app --reload --port=8000 --host=0.0.0.0"
14     ports:
15       - 8000:8000
16
```

TRIỂN KHAI THỬ NGHIỆM HỆ THỐNG

Dockerfile

```
compose.yaml  Dockerfile X
Dockerfile > ...
0
7 ARG PYTHON_VERSION=3.10.0
8 FROM python:${PYTHON_VERSION}-slim as base
9
10 # Prevents Python from writing pyc files.
11 ENV PYTHONDONTWRITEBYTECODE=1
12
13 # Keeps Python from buffering stdout and stderr to avoid situations where
14 # the application crashes without emitting any logs due to buffering.
15 ENV PYTHONUNBUFFERED=1
16
17 WORKDIR /app
18
19 # Create a non-privileged user that the app will run under.
20 # See https://docs.docker.com/go/dockerfile-user-best-practices/
21 ARG UID=10001
22 RUN adduser \
23     --disabled-password \
24     --gecos "" \
25     --home "/nonexistent" \
26     --shell "/sbin/nologin" \
27     --no-create-home \
28     --uid "${UID}" \
29     appuser
```

```
30
31 # Download dependencies as a separate step to take advantage of Docker's caching.
32 # Leverage a cache mount to /root/.cache/pip to speed up subsequent builds.
33 # Leverage a bind mount to requirements.txt to avoid having to copy them into
34 # into this layer.
35 RUN --mount=type=cache,target=/root/.cache/pip \
36     --mount=type=bind,source=requirements.txt,target=requirements.txt \
37     python -m pip install -r requirements.txt
38
39 # Switch to the non-privileged user to run the application.
40 USER appuser
41
42 # Copy the source code into the container.
43 COPY .
44
45 # Expose the port that the application listens on.
46 EXPOSE 8000
47
48 # Run the application.
49 CMD unicorn app:app --reload=True --host=127.0.0.1 --port=8000
50 Ctrl+L to chat, Ctrl+K to generate
```

HOÀN THÀNH CÀI ĐẶT PYTHON VÀ TIẾN HÀNH TRUY CẬP ỨNG DỤNG

sử dụng lệnh
docker compose
up -build

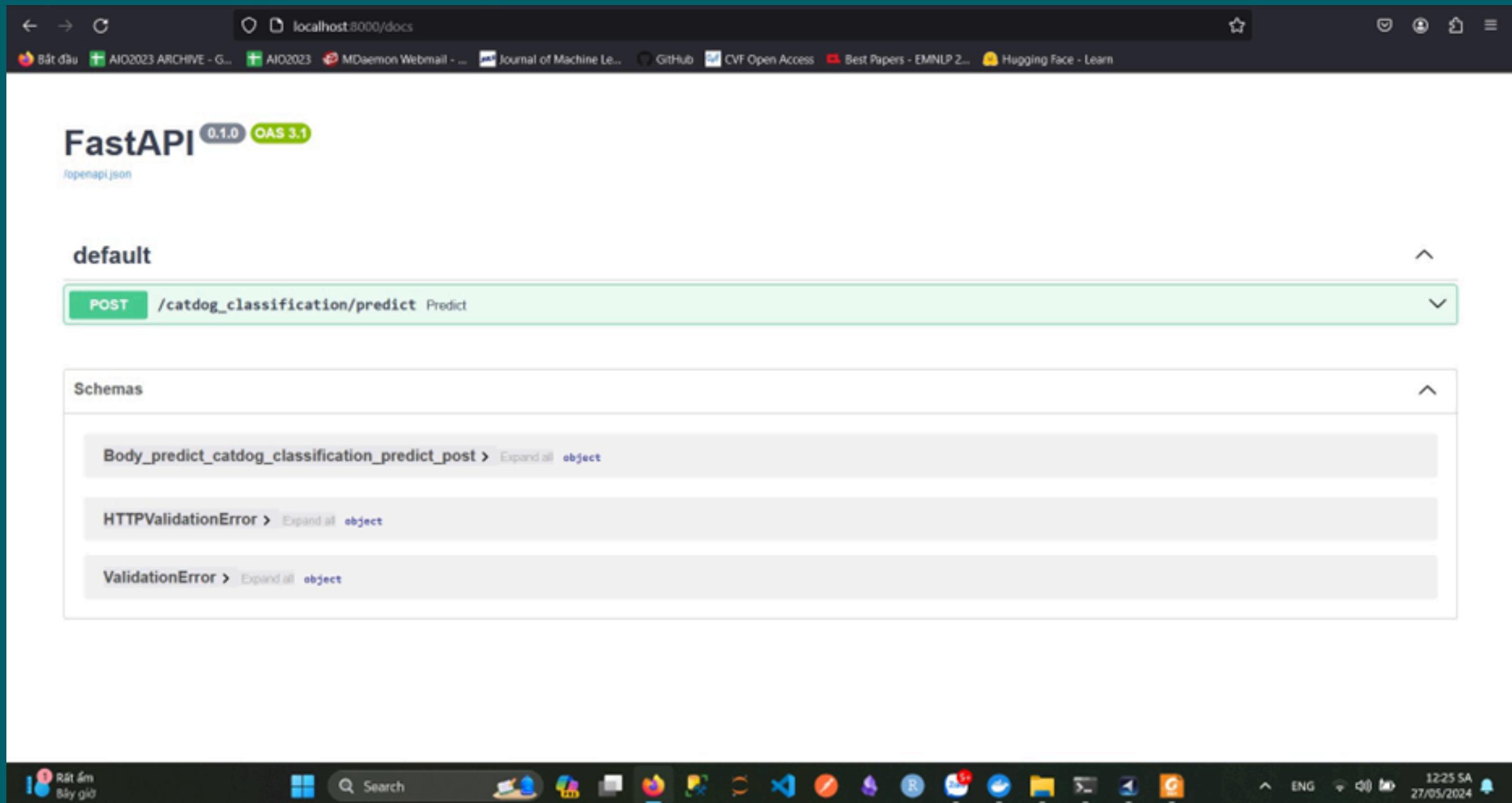


```
PROBLEMS DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
----- 3/30/2024 6:38 PM 90 requirements.txt
----- 5/26/2024 12:28 PM 111 server.py

PS C:\Users\84898\Desktop\Image Classification> docker compose up --build
[+] Building 8.2s (14/14) FINISHED
=> [web internal] load .dockerrcignore
=> => transferring context: 667B
=> [web internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.59kB
=> [web] resolve image config for docker.io/docker/dockerfile:1
=> [web auth] docker/dockerfile:pull token for registry-1.docker.io
=> CACHED [web] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e
=> [web internal] load metadata for docker.io/library/python:3.10.0-slim
=> [web auth] library/python:pull token for registry-1.docker.io
=> [web base 1/5] FROM docker.io/library/python:3.10.0-slim@sha256:ad548a471260fee5e5e1a99ee2acf142efe8c279a7a54315160d8033ba88f0d8
=> [web internal] load build context
=> => transferring context: 968B
=> CACHED [web base 2/5] WORKDIR /app
=> CACHED [web base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --no-create-home --uid "10001" appuser
=> CACHED [web base 4/5] RUN --mount=type=cache,target=/root/.cache/pip --mount-type=bind,source=requirements.txt,target=requirements.txt python -m pip install -r requirements.txt
=> CACHED [web base 5/5] COPY . /app/
=> [web] exporting to image
=> => exporting layers
=> => writing image sha256:bfad24cf5135163915a6b96843de85ddaf364f3d5a97912c89ca2fde8e3c04d2
=> => naming to docker.io/library/imageclassification-web
[+] Running 2/2
✓ Network imageclassification_default Created
✓ Container imageclassification-web-1 Created
Attaching to web-1
web-1 | INFO: Will watch for changes in these directories: ['/app']
web-1 | INFO: Uvicorn running on http://0.0.0:8000 (Press CTRL+C to quit)
web-1 | INFO: Started reloader process [7] using StatReload
web-1 | Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100.0%
web-1 | INFO: Started server process [9]
web-1 | INFO: Waiting for application startup.
web-1 | INFO: Application startup complete.
```

HOÀN THÀNH CÀI ĐẶT PYTHON VÀ TIẾN HÀNH TRUY CẬP ỨNG DỤNG

Truy cập vào
localhost



HOÀN THÀNH CÀI ĐẶT PYTHON VÀ TIẾN HÀNH TRUY CẬP ỨNG DỤNG

Mô hình chạy
thuật toán và đưa
ra kết quả

The screenshot shows a web browser window with the URL `localhost:8000/docs#/default/predict_catdog_classification_predict_post`. The page displays the results of a POST request to the endpoint `/predict_catdog_classification/predict`. The response code is 200, and the response body is a JSON object:

```
{  
    "probs": [  
        0.031292482876377106,  
        0.9687006343765259  
    ],  
    "best_prob": 0.9687006343765259,  
    "predicted_id": 1,  
    "predicted_class": "Dog",  
    "predictor_name": "resnet18"  
}
```

The response headers are:

```
access-control-allow-credentials: true  
access-control-allow-origin: *  
content-length: 141  
content-type: application/json  
date: Sun, 26 May 2024 17:26:23 GMT  
server: uvicorn
```

The browser's taskbar at the bottom shows various open tabs and system icons.

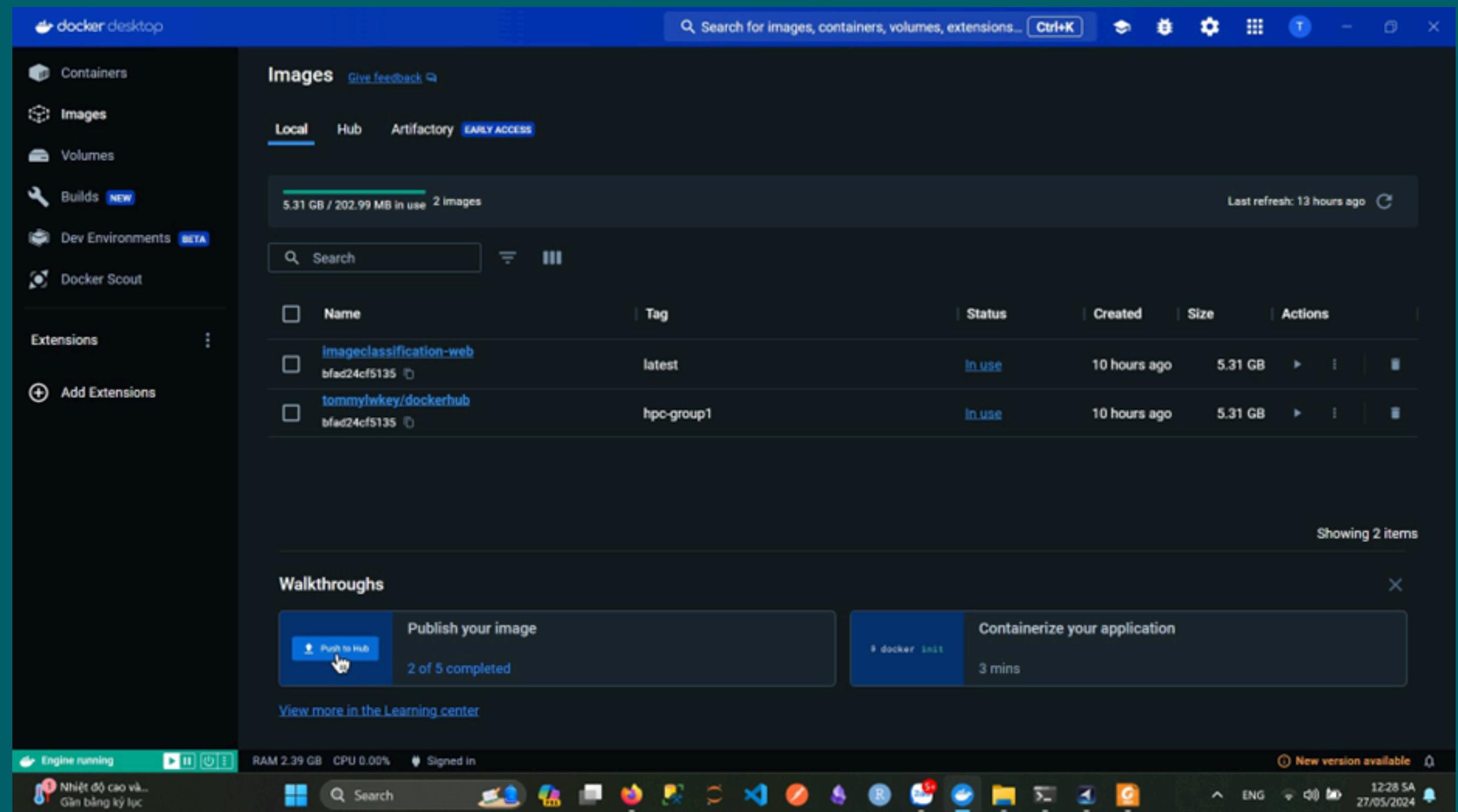
HOÀN THÀNH CÀI ĐẶT PYTHON VÀ TIẾN HÀNH TRUY CẬP ỨNG DỤNG

Các logging khi sử dụng mô hình

```
-> [web] resolve image config for docker.io/docker/dockerfile:1
-> [web auth] docker/dockerfile:pull token for registry-1.docker.io
-> CACHED [web] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e
-> [web internal] load metadata for docker.io/library/python:3.10.0-slim
-> [web auth] library/python:pull token for registry-1.docker.io
-> [web base 1/5] FROM docker.io/library/python:3.10.0-slim@sha256:ad540a471260fee5e5e1a99ee2acf142e7e8c279a7a54315160d8033ba88f0d8
-> [web internal] load build context
-> -> transferring context: 9688
-> CACHED [web base 2/5] WORKDIR /app
-> CACHED [web base 3/5] RUN adduser --disabled-password --gecos "" --home "/nonexistent" --shell "/sbin/nologin" --no-create-home --uid "10001" appuser
-> CACHED [web base 4/5] RUN --mount-type=cache,target=/root/.cache/pip --mount-type=bind,source=requirements.txt,target=requirements.txt python -m pip install -r requirements.txt
-> CACHED [web base 5/5] COPY . /app
-> [web] exporting to image
-> -> exporting layers
-> -> writing image sha256:bfad24cf5135163915a6b96843de85ddaf364f3d5a97912c89ca2fde8e3c04d2
-> -> naming to docker.io/library/imageclassification-web
[+] Running 2/2
✓ Network imageclassification_default Created
✓ Container imageclassification-web-1 Created
Attaching to web-1
web-1 | INFO: Will watch for changes in these directories: ['/app']
web-1 | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
web-1 | INFO: Started reloader process [7] using StatReload
web-1 | Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100.0%
web-1 | INFO: Started server process [9]
web-1 | INFO: Waiting for application startup.
web-1 | INFO: Application startup complete.
web-1 | INFO: 172.19.0.1:39048 - "GET / HTTP/1.1" 404 Not Found
web-1 | INFO: 172.19.0.1:39048 - "GET /favicon.ico HTTP/1.1" 404 Not Found
web-1 | INFO: 172.19.0.1:39048 - "GET /docs HTTP/1.1" 200 OK
web-1 | INFO: 172.19.0.1:39048 - "GET /openapi.json HTTP/1.1" 200 OK
web-1 | INFO: 172.19.0.1:39052 - "POST /catdog_classification/predict HTTP/1.1" 200 OK
web-1 | INFO: 172.19.0.1:39054 - "POST /catdog_classification/predict HTTP/1.1" 200 OK
```

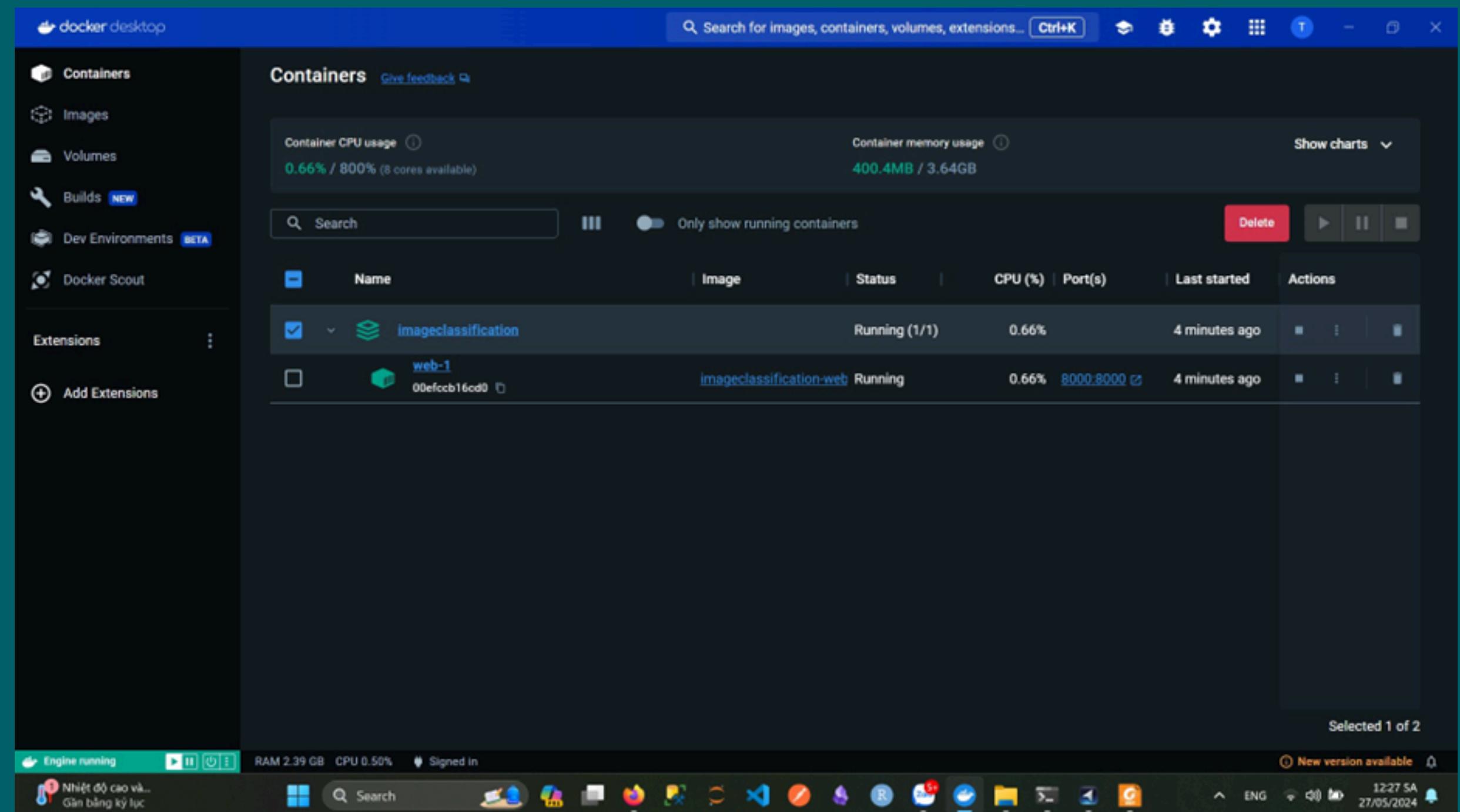
THỬ NGHIỆM TẮT CONTAINER

2 image đang có trên máy



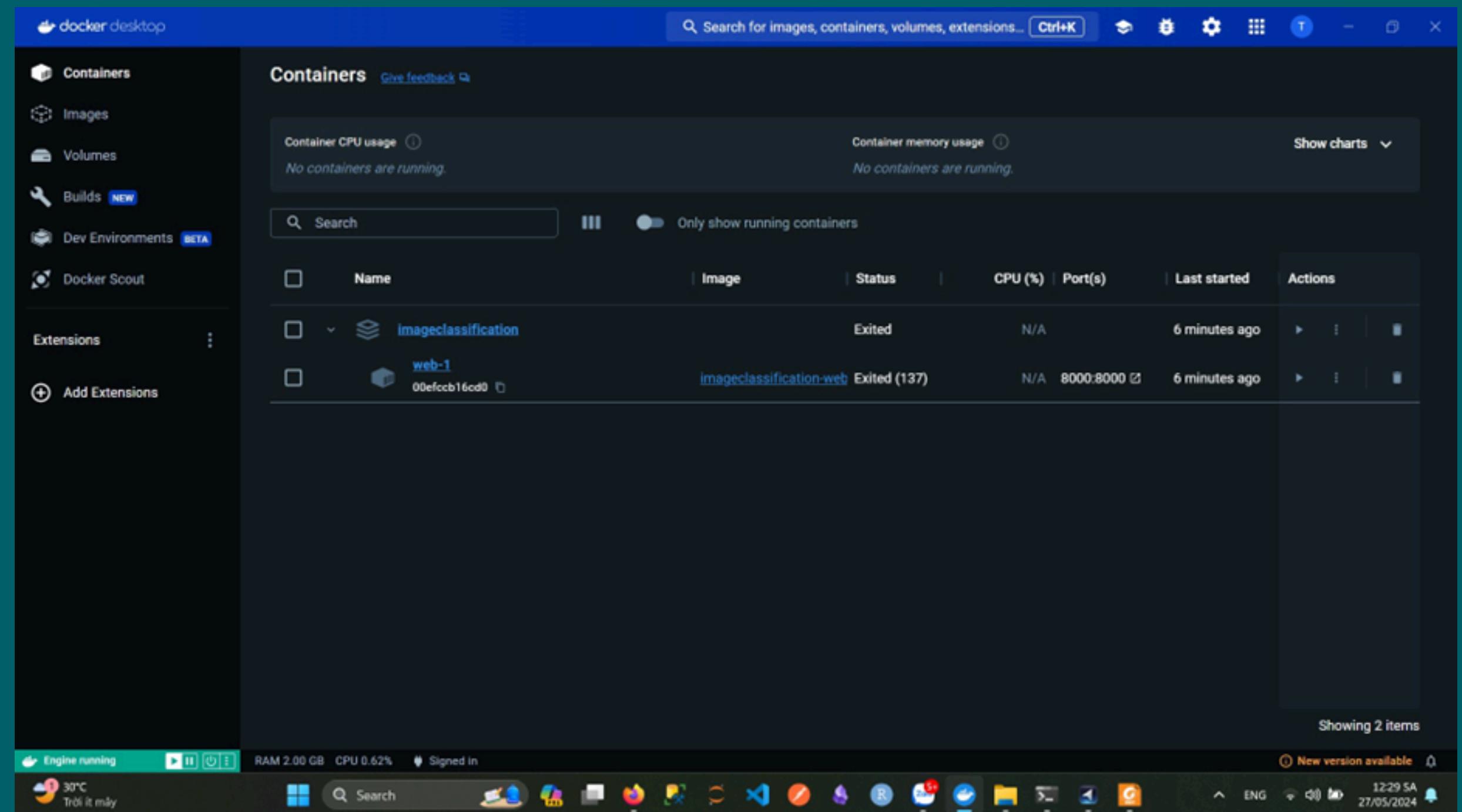
THỬ NGHIỆM TẮT CONTAINER

Container web-1
đang chạy



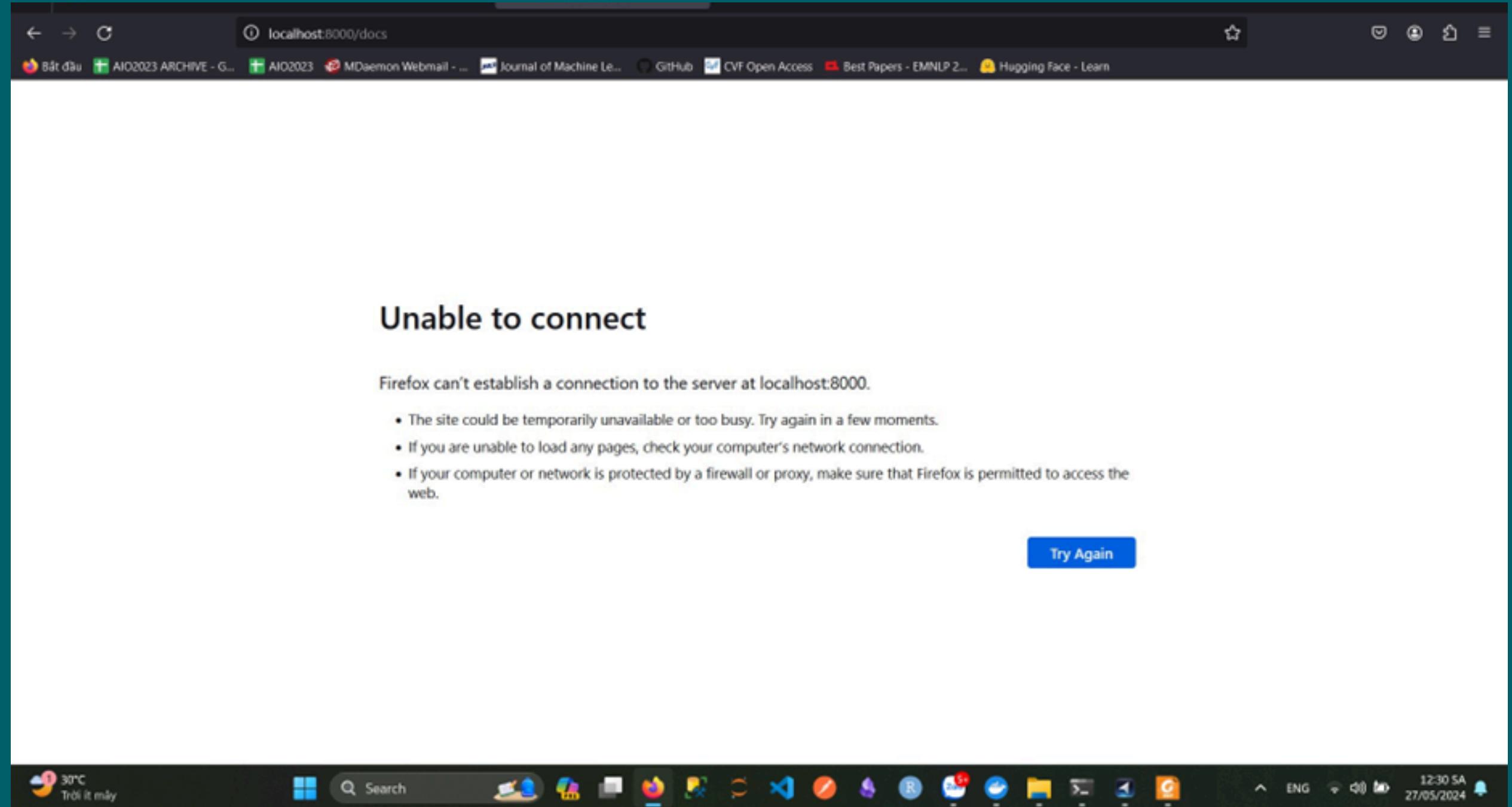
THỬ NGHIỆM TẮT CONTAINER

Tắt container đó đi



THỬ NGHIỆM TẮT CONTAINER

**Không thể kết nối
tới đường dẫn của
ứng dụng được nữa**



ĐĂNG TẢI LÊN DOCKER HUB

Tải image của ứng
dụng này lên
Docker hub

```
C:\Users\84898\Desktop\Image Classification>docker tag imageclassification-web:latest tommylwkey/dockerhub:hpc-group1
C:\Users\84898\Desktop\Image Classification>docker push tommylwkey/dockerhub:hpc-group1
The push refers to repository [docker.io/tommylwkey/dockerhub]
6d15691c42d1: Pushed
ba79c2023f3f: Pushed
ab6f8cdba580: Pushed
eab2ed4702fc: Pushed
3922ae00a1e0: Pushed
ae04e3a13052: Pushed
19f7565baf43: Pushed
456910ac81d3: Pushed
9321ff862abb: Pushed
hpc-group1: digest: sha256:4c497f32c85409a3dd95ce4e48077785936dcfad5c4d016cc2286f7859767f4c size: 2211
```

ĐĂNG TẢI LÊN DOCKER HUB

Thông tin của image
đã tải lên

The screenshot shows a Docker Hub repository page for the user 'tommylwkey'. The repository name is 'tommylwkey/dockerhub' under the 'General' tab. The page displays information about the repository, including its description ('ImageClassify: Pre-built env for image recognition tasks.'), Docker commands for pushing a new tag, and a list of tags. One tag, 'hpc-group1', is listed with details: OS (Ubuntu), Type (Image), Pulled 2 days ago, and Pushed 3 days ago. The page also features sections for Automated Builds and a navigation bar at the bottom.

tommylwkey / [Repositories](#) / [dockerhub](#) / [General](#)

General Tags Builds Collaborators Webhooks Settings

tommylwkey/dockerhub [Edit](#)

Updated 3 days ago

ImageClassify: Pre-built env for image recognition tasks. [Edit](#)

[DATA SCIENCE](#) [MACHINE LEARNING & AI](#) [Edit](#)

Docker commands

To push a new tag to this repository:

```
docker push tommylwkey/dockerhub:tagname
```

[Public View](#)

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
hpc-group1	Ubuntu	Image	2 days ago	3 days ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

[Upgrade](#)

https://hub.docker.com/billing/core/purchase

Nhiều máy 31°C

Search

29/05/2024 9:34 SA

4. BÀN LUẬN VÀ ĐÁNH GIÁ

BÀN LUẬN VÀ ĐÁNH GIÁ

Bàn luận về tổng quan hệ
thống

Việc triển khai hệ thống phân loại văn bản mà
còn đảm bảo tính linh hoạt, hiệu quả và khả năng
mở rộng của hệ thống.

PyTorch, với tính năng mạnh mẽ trong việc xây
dựng và huấn luyện các mô hình học sâu, cung cấp
nền tảng vững chắc cho việc phân loại văn bản

FastAPI, với khả năng tạo API RESTful nhanh
chóng và hiệu năng cao. Khả năng xử lý không
đồng bộ của FastAPI cũng tối ưu hóa hiệu suất

Kết hợp với Docker, hệ thống có thể được container
hóa, đảm bảo tính nhất quán của môi trường triển
khai và dễ dàng di chuyển giữa các máy chủ khác
nhau

BÀN LUẬN VÀ ĐÁNH GIÁ

Bàn luận về mô hình học sâu

Nhóm lựa chọn sử dụng mô hình học sâu là resnet18, với bộ trọng số có được từ việc fine-tune lại mô hình cho ảnh chó mèo. Có thể thấy mô hình hoạt động khá tốt, với thời gian dự đoán khá nhanh và kết quả cũng chính xác.

Tuy nhiên cũng có thể còn nhiều mô hình khác cũng rất phù hợp cho bài toán này, ta chỉ cần phải đánh đổi giữa tốc độ chạy của thuật toán với độ chính xác của thuật toán

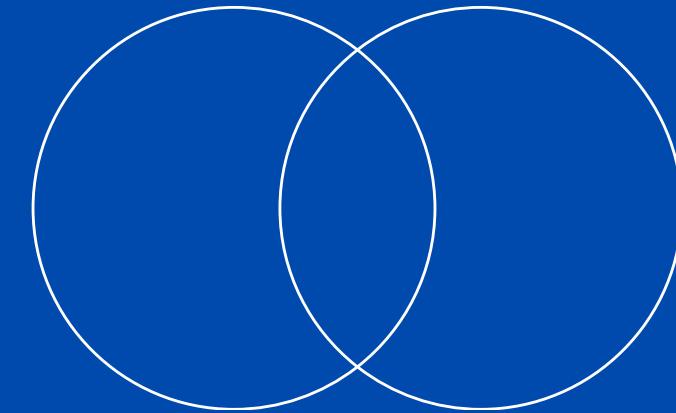
BÀN LUẬN VÀ ĐÁNH GIÁ

Đánh Giá

- Xét chung về tổng thể, hệ thống hoạt động rất tốt, tuy nhiên một vấn đề đáng lưu tâm đó là kích thước của image để có thể build lên hệ thống hoàn chỉnh khá nặng (5.14GB). Có thể thấy lý do cho việc này là do nhóm vẫn chưa thật sự tối ưu được những gì cần thiết để xây dựng một image. Do đó tối ưu tài nguyên cũng là một trong những hướng đi cần chú ý của ứng dụng này
- Bên cạnh đó, về lâu về dài, khi mà chúng ta có nhiều ứng dụng hơn, chúng ta thường sẽ cần nhiều Docker Container hơn, do đó mà chúng ta có thể sử dụng thêm các chương trình bên ngoài khác để quản lý, phát triển và duy trì các container một cách hiệu quả hơn, ví dụ như Kubernetes hay là Docker Swarm

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN



Kết luận:

Ứng dụng phân loại ảnh với một endpoint duy nhất , có thiết kế đơn giản nhưng mạnh mẽ, ứng dụng này có thể xử lý và phân loại các hình ảnh một cách nhanh chóng và chính xác. Khả năng tích hợp dễ dàng với các hệ thống khác và sự tiện lợi trong việc sử dụng API giúp nó trở thành một công cụ hữu ích cho nhiều lĩnh vực

Một số giải pháp cải tiến:

- Cải thiện độ chính xác và hiệu suất
- Tăng cường bảo mật và quyền riêng tư
- Phân loại đa nhiệm
- Tích hợp với các dịch vụ khác
- Hỗ trợ đa ngôn ngữ và đa văn hóa
- Giao diện người dùng và trải nghiệm người dùng (UI/UX)
- Cung cấp tính năng phân tích và báo cáo



College of
Technology and Design

SCHOOL OF BUSINESS INFORMATION TECHNOLOGY

CHEERS