

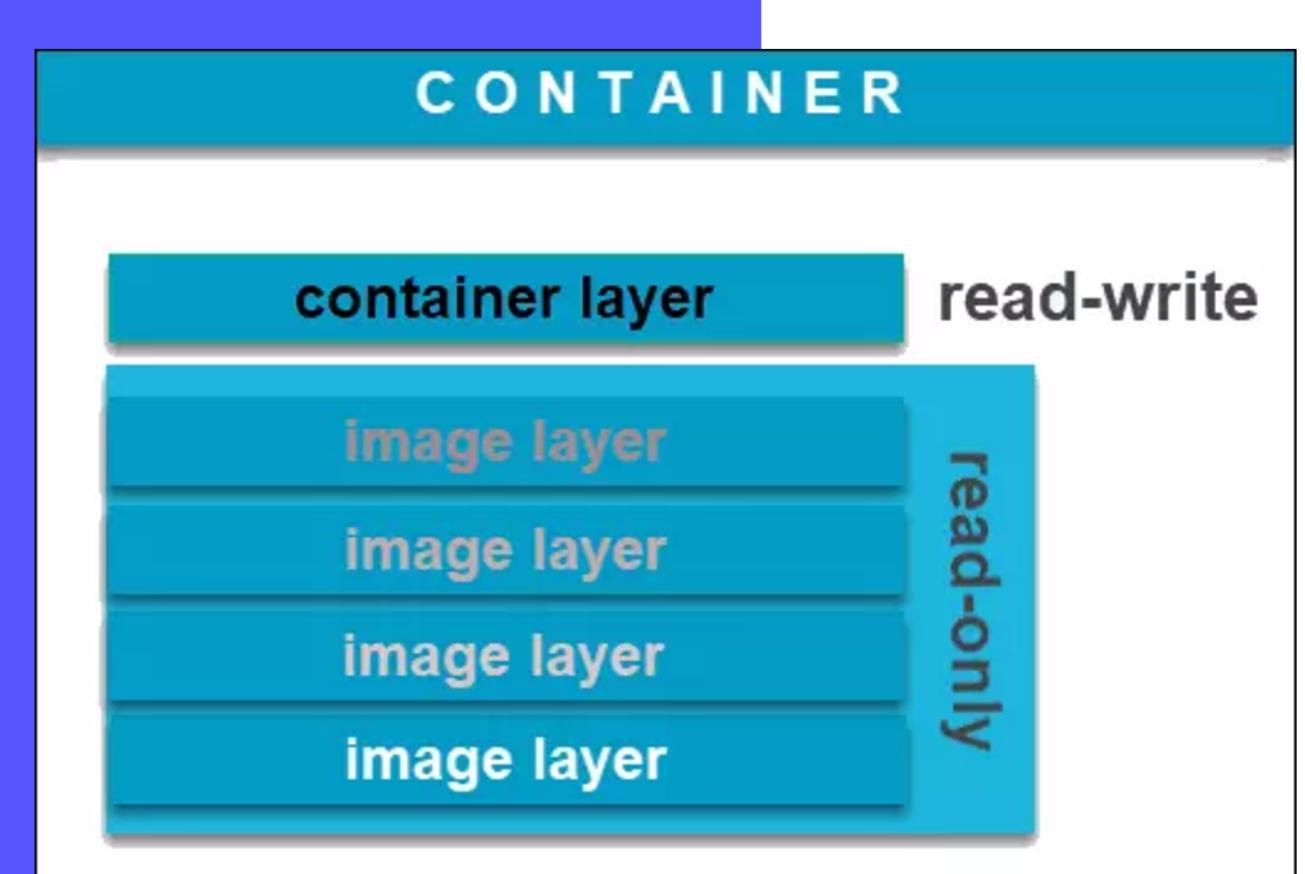
BÀI 5: BUILD DOCKER IMAGE (P1)



Docker Image là gì?

1

- Là một template (khuôn mẫu) chỉ đọc
- Chứa các thành phần cần thiết để chạy một ứng dụng trong Docker:
 - Mã nguồn ứng dụng (source code).
 - Các thư viện và dependencies mà ứng dụng yêu cầu.
 - Các tệp cấu hình cần thiết cho ứng dụng.
 - Thông tin cấu hình hệ thống để chạy ứng dụng (cổng, environment variables, v.v.).
- Nên tảng tạo ra các container.



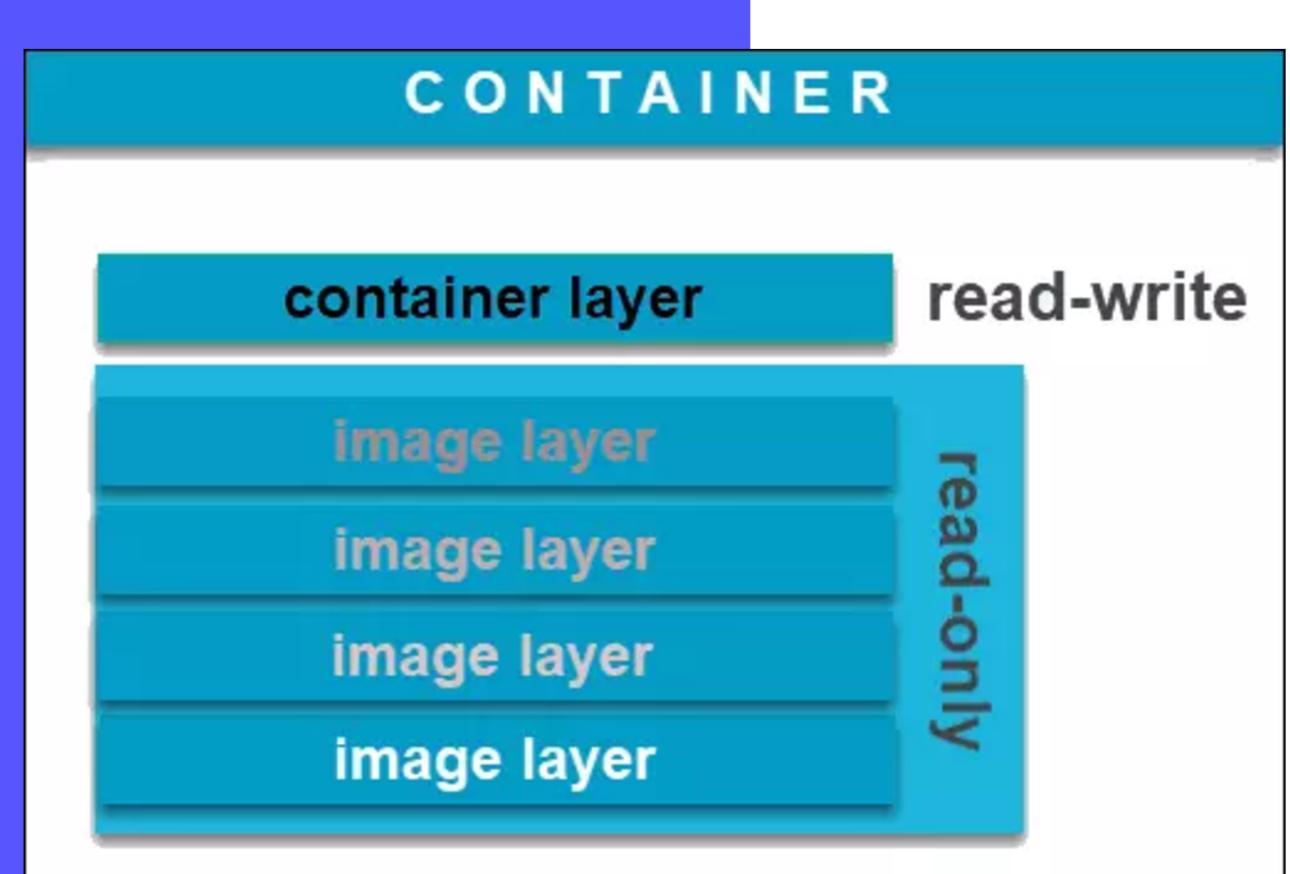
Principle:

1. Immutable Infrastructure (Hạ tầng bất biến)

Docker Image là một đối tượng chỉ đọc (read-only), không thay đổi sau khi được tạo. Điều này đảm bảo rằng mỗi lần bạn khởi tạo container từ cùng một image, bạn luôn nhận được kết quả giống hệt nhau.

2. Layered File System (Hệ thống tệp nhiều lớp)

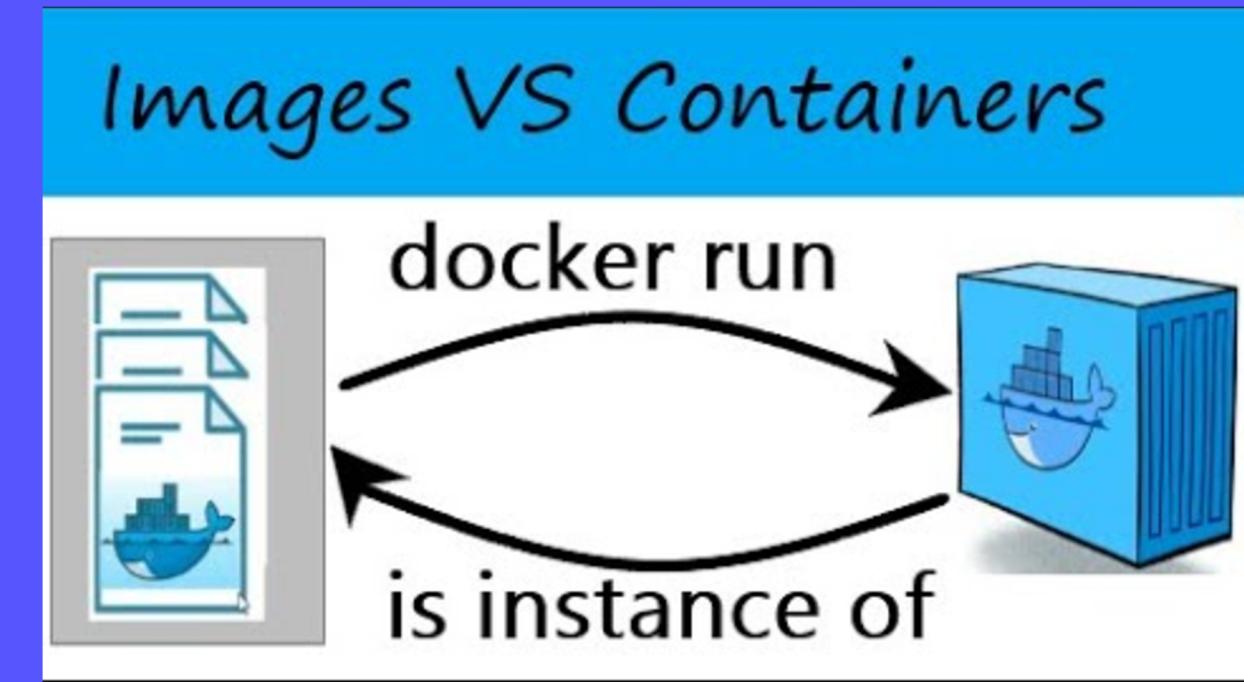
Docker Image được xây dựng dựa trên mô hình nhiều lớp (layered architecture). Mỗi chỉ thị trong Dockerfile như FROM, RUN, COPY sẽ tạo ra một lớp mới.



Docker image vs Docker Container

1

Docker Image	Docker Container
Bản thiết kế (blueprint).	Thực thi thực tế (runtime instance).
Chỉ đọc, không thay đổi được.	Có thể thay đổi trong quá trình chạy.
Không tiêu tốn tài nguyên hệ thống.	Tiêu tốn tài nguyên CPU, RAM khi chạy.



Gồm nhiều lớp (layers) chồng lên nhau

1. Base Layer:

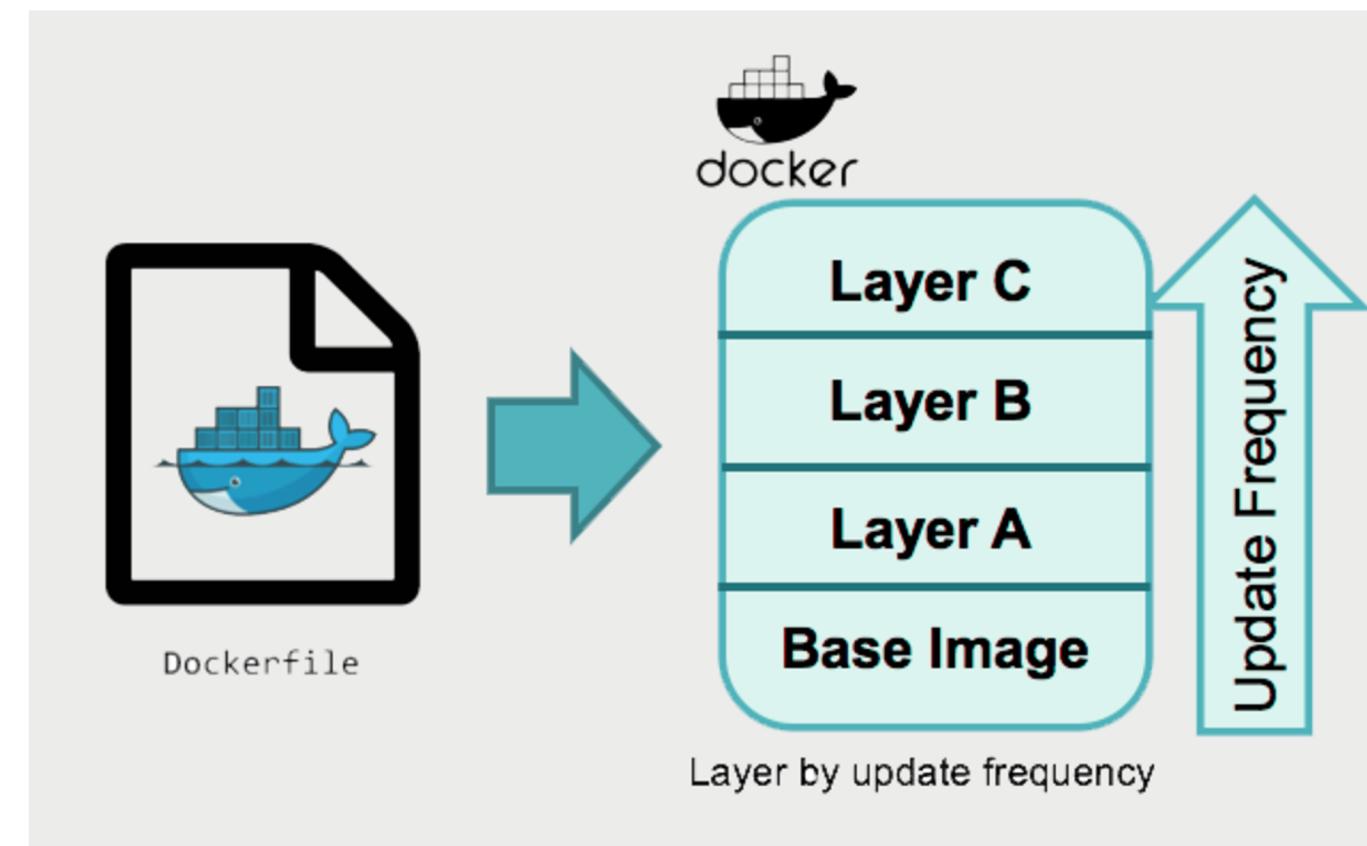
- Thường là một hệ điều hành tối giản như Ubuntu, Alpine.

2. Intermediate Layers (Lớp trung gian):

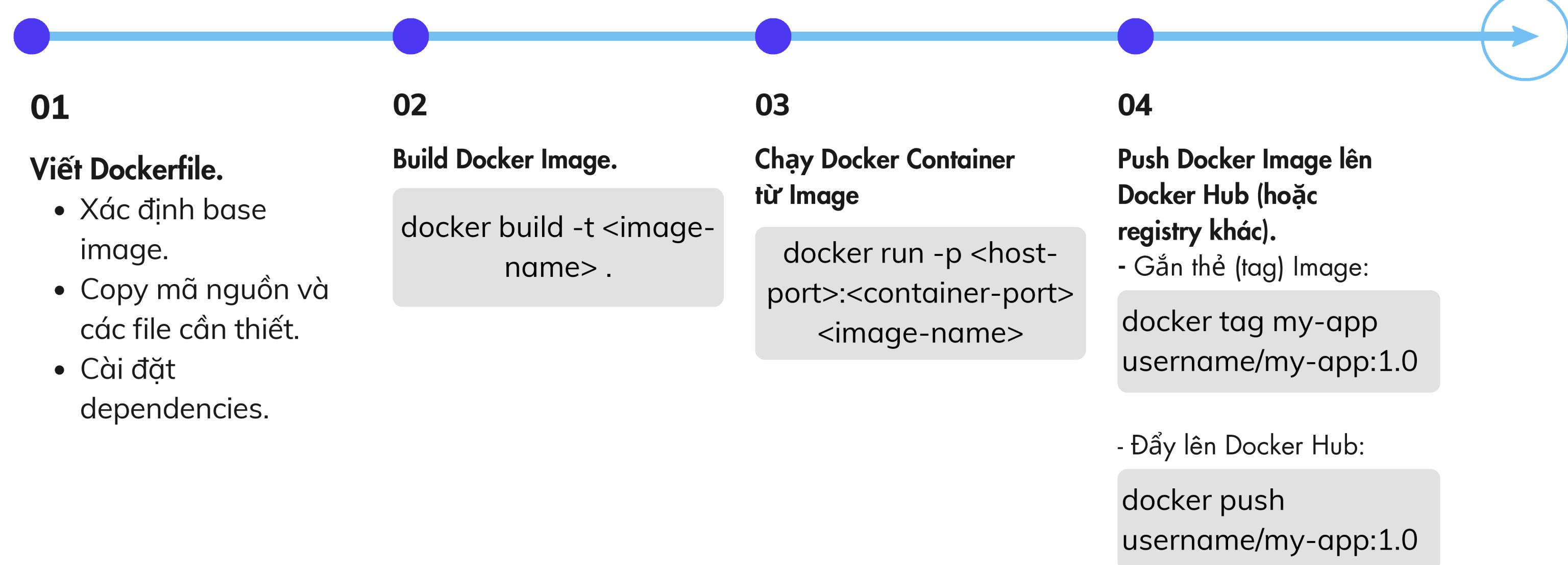
- Được tạo ra từ các lệnh trong Dockerfile như RUN, COPY, ADD.

3. Top Layer (Lớp Ứng dụng):

- Chứa code và cấu hình của ứng dụng.



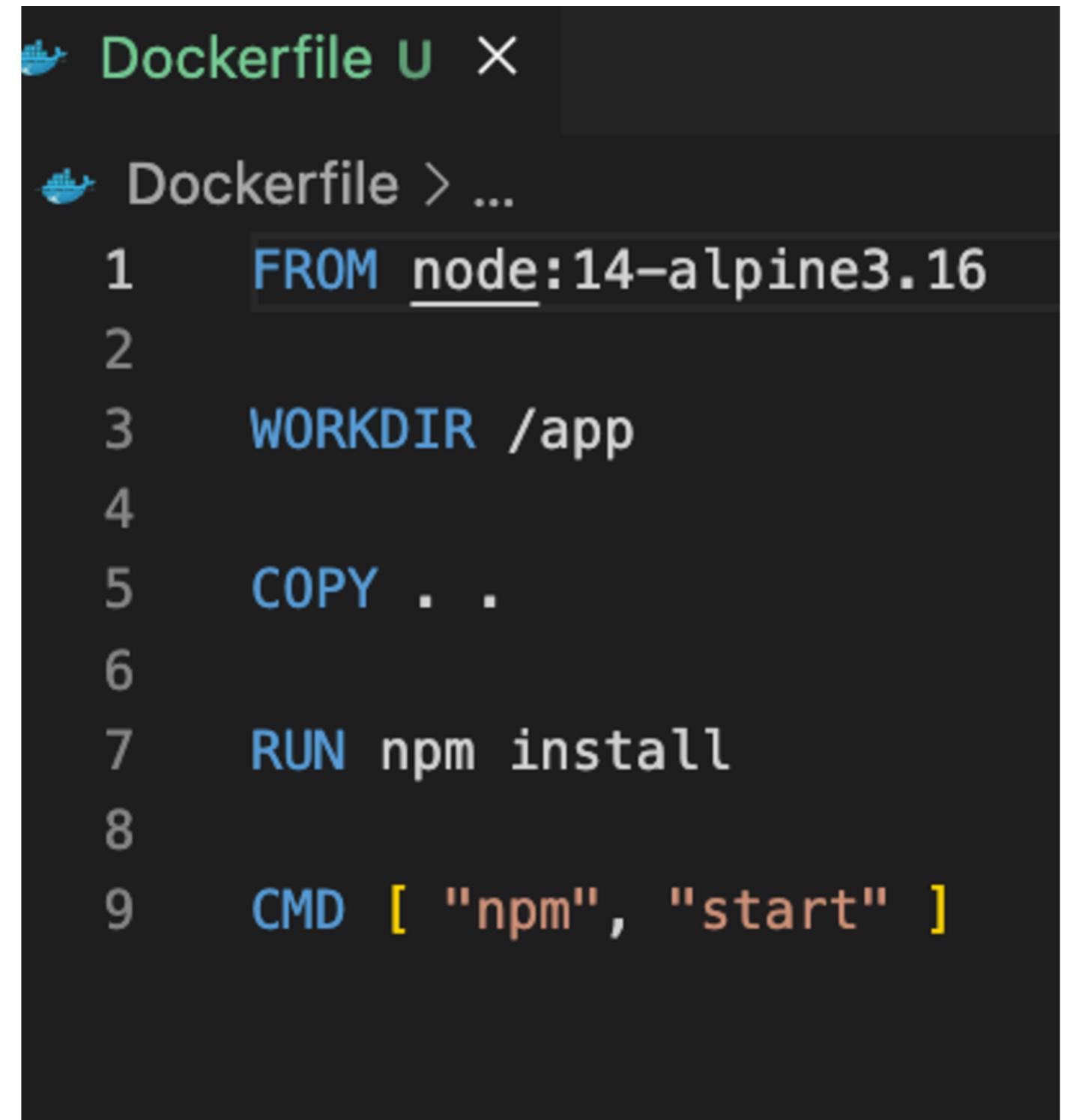
Quy trình làm việc với Docker Image



- Dockerfile là tập tin định nghĩa các bước cần thiết để xây dựng Docker Image.

Các lệnh cơ bản trong Dockerfile

- **FROM:**
 - Xác định base image.
 - Ví dụ: FROM node:16-alpine.
- **WORKDIR:**
 - Định nghĩa thư mục làm việc bên trong container.
 - Ví dụ: WORKDIR /app.
- **COPY:**
 - Copy file từ máy host vào container.
 - Ví dụ: COPY ..

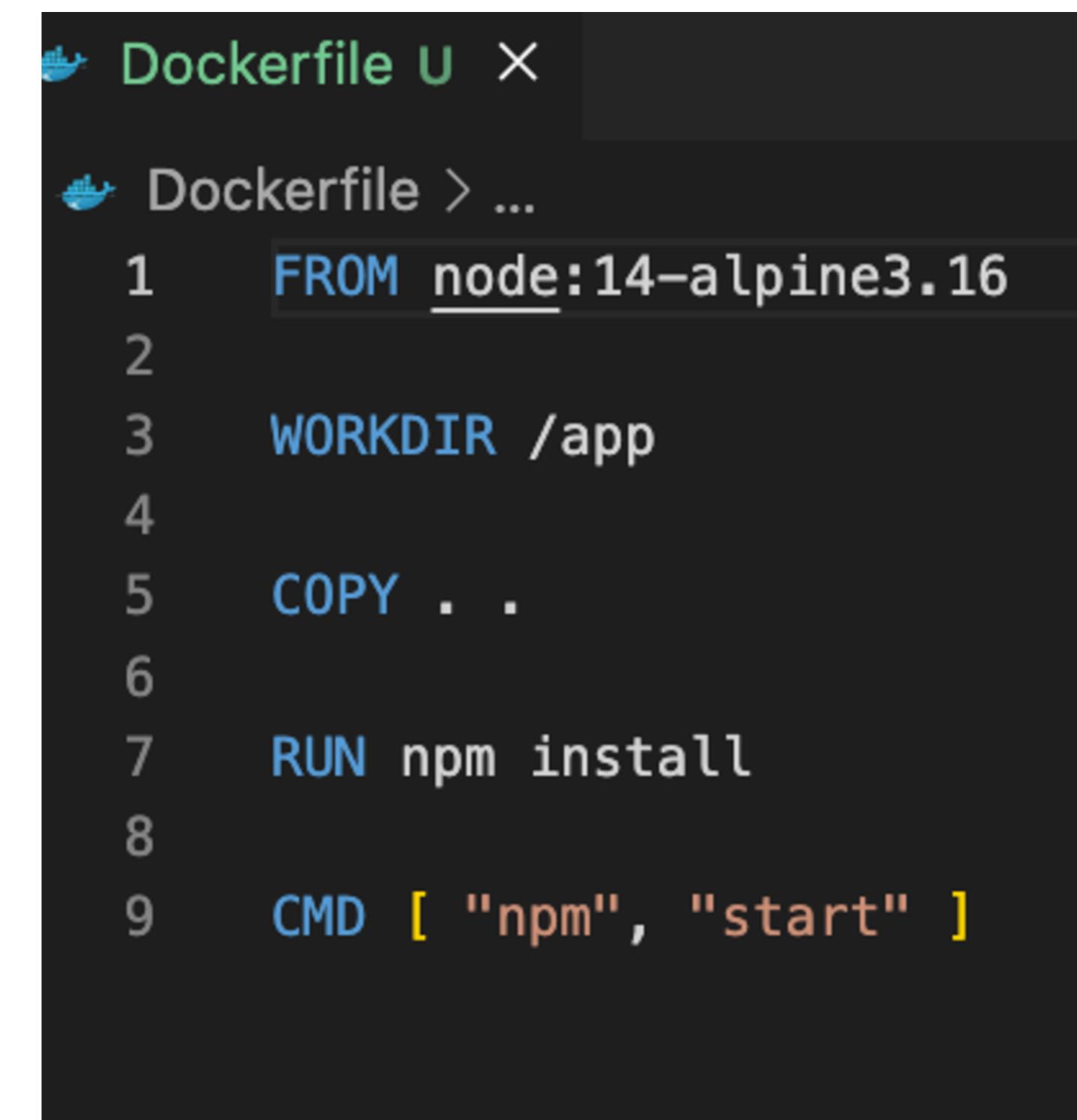


```
1  FROM node:14-alpine3.16
2
3  WORKDIR /app
4
5  COPY . .
6
7  RUN npm install
8
9  CMD [ "npm", "start" ]
```

- Dockerfile là tập tin định nghĩa các bước cần thiết để xây dựng Docker Image.

Các lệnh cơ bản trong Dockerfile

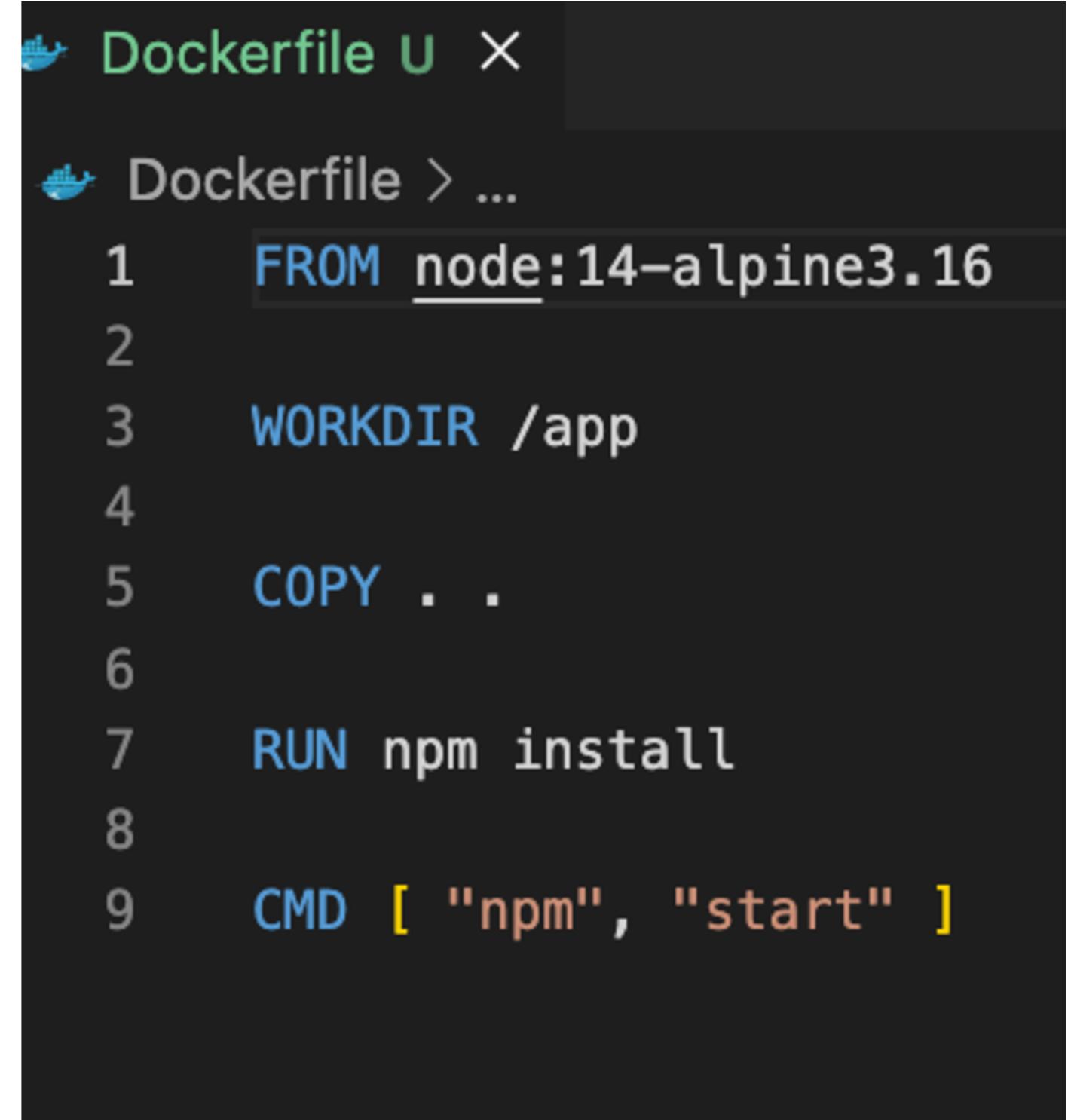
- **RUN:**
 - Chạy lệnh trong quá trình build.
 - Ví dụ: RUN npm install.
- **CMD hoặc ENTRYPOINT:**
 - Định nghĩa lệnh chạy khi container khởi chạy.
 - Format: CMD ["command", "argument1", "argument2"]
 - Ví dụ: CMD ["npm", "start"] (tương đương khi ta chạy: npm start)
- **EXPOSE:**
 - Khai báo cổng container sẽ sử dụng.
 - Ví dụ: EXPOSE 3000



```
1 FROM node:14-alpine3.16
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN npm install
8
9 CMD [ "npm", "start" ]
```

Lưu ý khi xây dựng Dockerfile

- **Chọn Base Image phù hợp:**
 - Base image nhẹ hơn giúp giảm kích thước image (ví dụ: alpine).
- **Hạn chế số lượng layers:**
 - Kết hợp các lệnh RUN với && để giảm số lớp.
- **Không lưu dữ liệu nhạy cảm:**
 - Không hardcode thông tin như mật khẩu trong Dockerfile.
- **Kiểm tra kích thước Image:**
 - Dùng lệnh: docker images
- **Sử dụng Multi-Stage Build để tối ưu hóa:**
 - Chỉ copy các file cần thiết sang final image.



```
1 FROM node:14-alpine3.16
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN npm install
8
9 CMD [ "npm", "start" ]
```

Ví dụ Dockerize Python app

```
FROM ubuntu:22.04

# install app dependencies
RUN apt-get update && apt-get install -y python3 python3-pip
RUN pip install flask==3.0.*

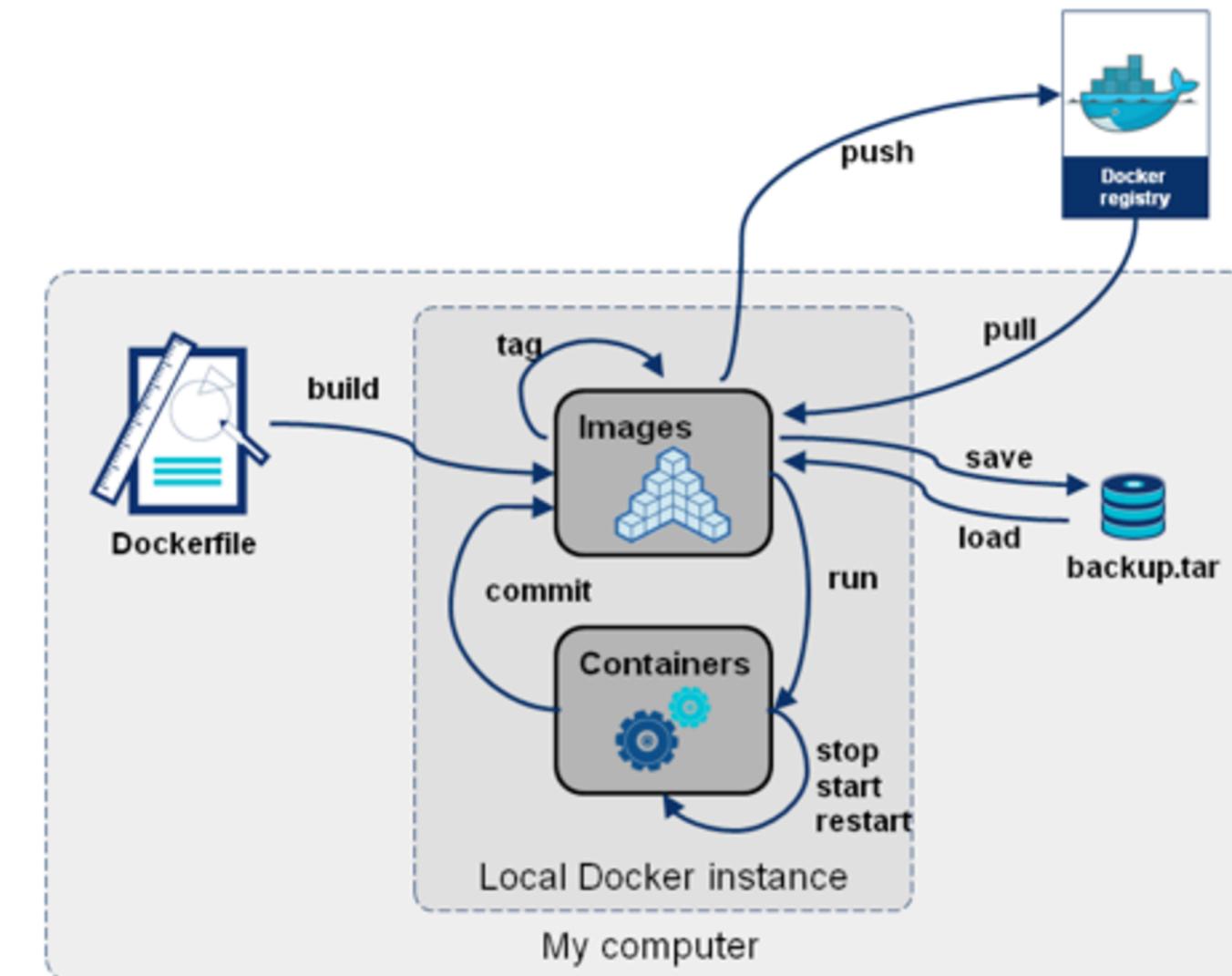
# install app
COPY hello.py /

# final configuration
ENV FLASK_APP=hello
EXPOSE 8000
CMD ["flask", "run", "--host", "0.0.0.0", "--port", "8000"]
```

Mục đích:

Tạo image từ Dockerfile

- Run container
- Đóng gói và chia sẻ (build and ship)



Cú pháp

```
docker buildx build [OPTIONS] PATH|URL| -
```

Alias: `docker build` | `docker builder build` | `docker image build` | `docker buildx b`

Options:

`-f, --file` : Name of the Dockerfile (default: PATH/Dockerfile)

`--build-arg`: Set build-time variables

`--build-context`: Additional build contexts (e.g., name=path)

`-t, --tag`: Name and optionally a tag (format: name:tag)

`--push`: Shorthand for `--output=type=registry`. Will automatically push the build result to registry.

Lưu ý:

Khi mới tiếp cận, nên tách từng bước, tránh gộp các bước dẫn đến khó debug. Ví dụ ta có thể tách ra từng bước:

- build
- gắn tag cho image
- push image lên registry

Ví dụ

```
docker buildx build [OPTIONS] PATH|URL|-
```

Alias: [docker build](#) | [docker builder build](#) | [docker image build](#) | [docker buildx b](#)

Mục đích:

Lưu trữ image lên Registry để thuận tiện trong việc chia sẻ, phân phối

Steps:

#1 (login to docker registry nếu cần)

docker login

#2 gắn tag cho image (nếu cần)

docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

VD:

docker tag httpd fedora/httpd:version1.0

docker tag httpd:test fedora/httpd:version1.0.test

docker tag 0e5574283393 fedora/httpd:version1.0

Lưu ý: Nếu muốn PUSH image lên registry nội bộ, TARGET_IMAGE phải bao gồm Registry host & port

docker tag 0e5574283393 myregistryhost:5000/fedora/httpd:version1.0

#3. Push lên registry

docker image push [OPTIONS] NAME[:TAG]

VD:

docker image push fedora/httpd:version1.0

docker image push myregistryhost:5000/fedora/httpd:version1.0

Bài tập

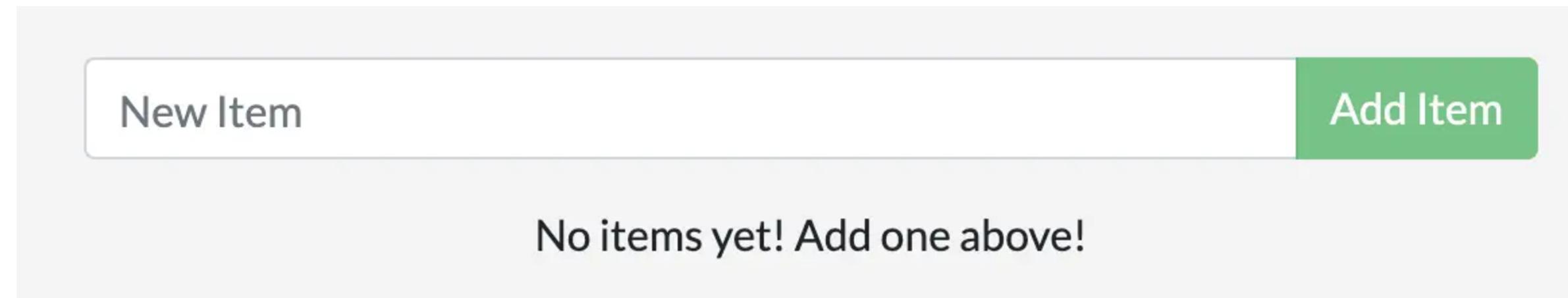
Checkout repo <https://github.com/docker/getting-started-app> và dockerize thành image.

Lưu ý: Expose port của Ứng dụng tại port 3000

Image sau khi build được gắn tag v0.0.1 và lưu trữ tại Docker Hub với tên “your_username/ha-course”

Tiêu chí hoàn thành:

- Khi run image “`docker run -d -p 3000:3000 your_username/ha-course:v0.0.1`” tại máy bất kì, truy cập `http://localhost:3000` sẽ hiển thị giao diện



Backend Team

THANK
YOU

