

DOCKER NÂNG CAO

PRESENTATION BY NGUYEN NHU THUONG

NỘI DUNG CHÍNH

VIẾT DOCKERFILE ĐÓNG GÓI DOCKER IMAGE

Tìm hiểu Dockerfile là gì? Dockerfile để làm gì? Cách viết Dockerfile tối ưu nhất.

THỰC HÀNH VIẾT DOCKERFILE

Thực hành viết Dockerfile qua các ví dụ, usecase thực tế khác nhau khi viết Dockerfile

DOCKER COMPOSE

Tìm hiểu docker-compose là gì? Tại sao phải sử dụng docker-compose và các cú pháp trong docker-compose.

THỰC HÀNH VỚI DOCKER COMPOSE

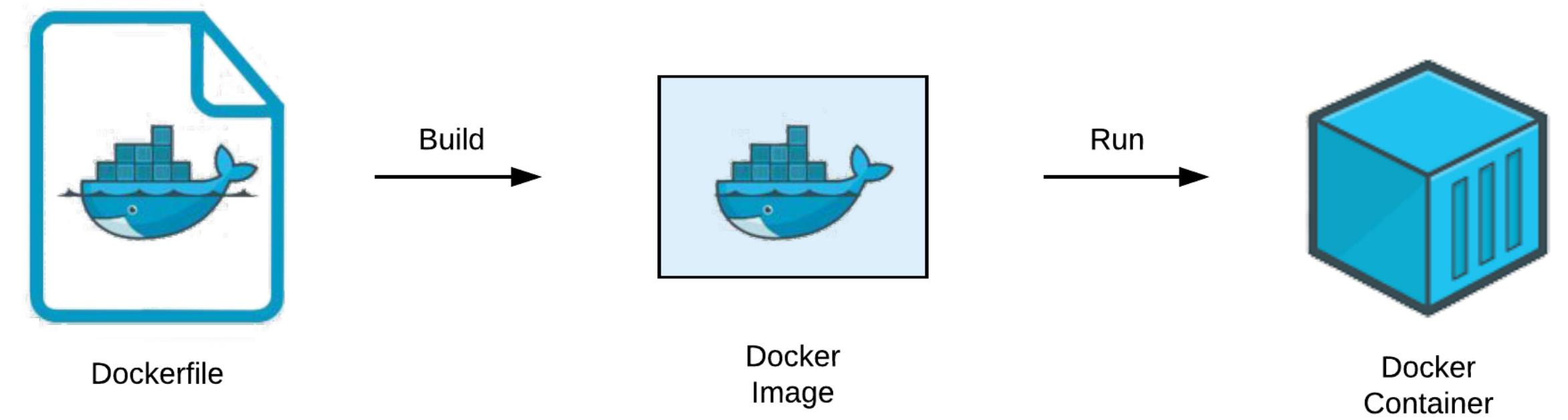
Thực hành viết docker-compose qua các ví dụ, hướng dẫn triển khai hệ thống với docker-compose.



VIẾT DOCKERFILE ĐÓNG GÓI IMAGE

Dockerfile là gì?

- **Dockerfile** là một file dạng text không có phần đuôi mở rộng, chứa các đặc tả về một trường thực thi phần mềm, cấu trúc cho **Docker Image**.
- Từ những câu lệnh đó, Docker sẽ build ra Docker image (thường có dung lượng nhỏ từ vài MB đến lớn vài GB).



TASK #1

VIẾT DOCKERFILE CHẠY ỨNG DỤNG JAVA SPRING BOOT

chuẩn bị java project

- Truy cập <https://start.spring.io> để khởi tạo java spring boot project
- Viết một dịch vụ đơn giản bằng java spring boot
- Chạy lệnh `./mvnw package` để build dự án java và kiểm tra file jar đã được build

```
package com.learning.hello;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class HelloApplication {

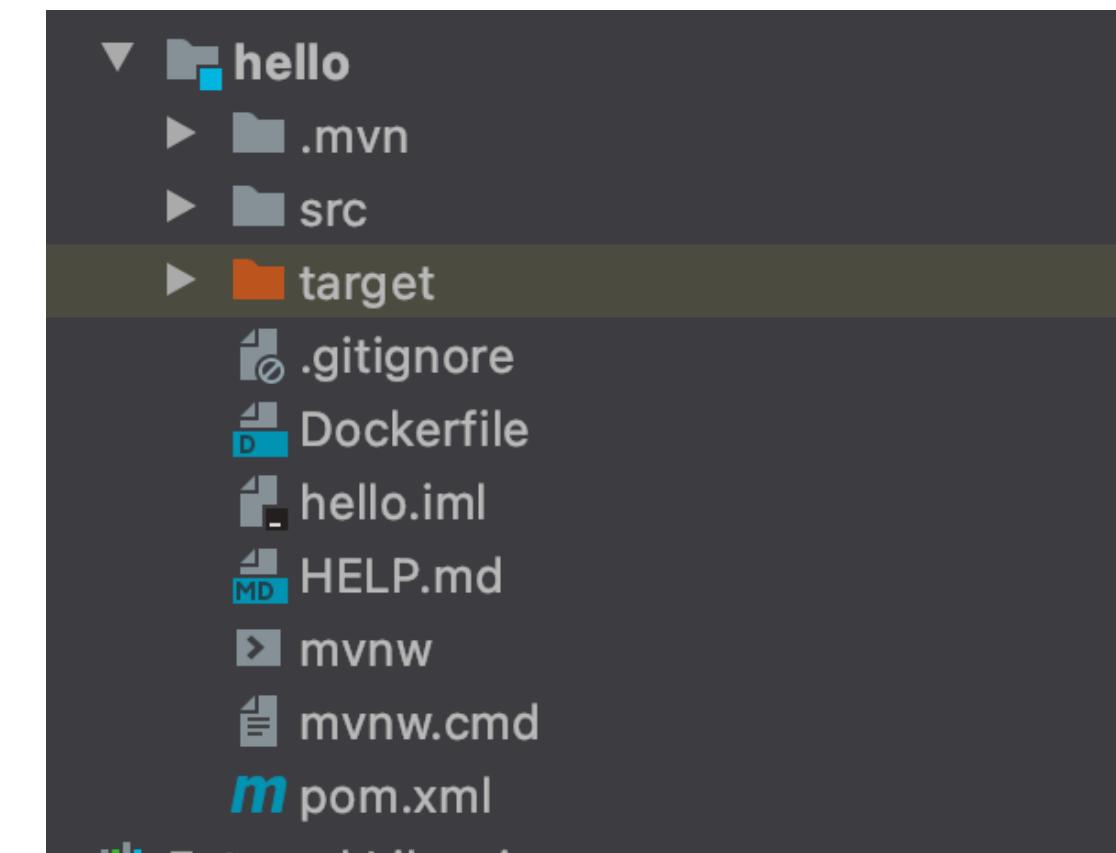
    @RequestMapping("/")
    public String home() {
        return "Hello Docker World";
    }

    public static void main(String[] args) {
        SpringApplication.run(HelloApplication.class, args);
    }
}
```

viết Dockerfile

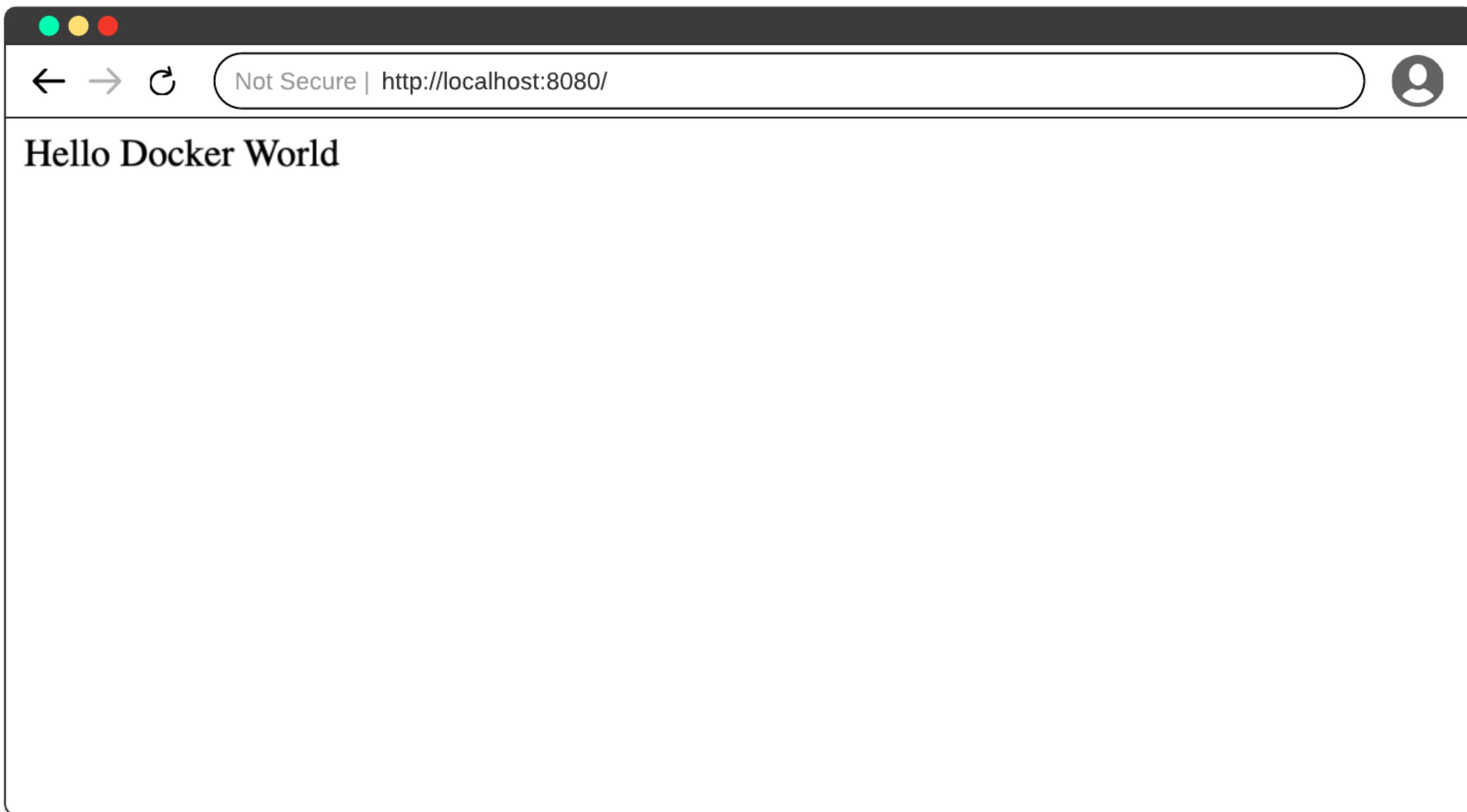
```
● ● ●

FROM openjdk:8-jdk-alpine
ARG JARFILE=target/*.jar
COPY ${JARFILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```



Câu lệnh	Ý nghĩa
FROM	Image base dùng để xây dựng java image
ARG	Xác định biến môi trường (có thể cấu hình từ bên ngoài)
COPY	Sao chép files từ thư mục hiện tại vào trong image
ENTRYPOINT	Câu lệnh dùng để chạy ứng dụng java

kiểm tra kết quả



Truy cập vào địa chỉ
<http://localhost:8080>
để kiểm tra

Cấu trúc Dockerfile

Cú pháp chung của một Dockerfile có dạng:

INSTRUCTION arguments

- **INSTRUCTION** là tên các chỉ thị có trong Dockerfile được Docker quy định. Khi khai báo các chỉ thị này phải được viết bằng chữ IN HOA.
- Một Dockerfile bắt buộc phải bắt đầu bằng chỉ thị **FROM** để khai báo base image sử dụng để xây dựng nên image.
- **arguments** là phần nội dung của các chỉ thị, quyết định chỉ thị sẽ làm gì.

```

1 FROM node:alpine
2
3 RUN mkdir -p /usr/src/app
4 WORKDIR /usr/src/app
5
6 COPY package.json /usr/src/app/
7 RUN npm install
8
9 COPY ./ /usr/src/app
10 RUN npm run build
11
12 EXPOSE 3000
13
14 CMD [ "npm", "run", "start" ]

```

Ý nghĩa từng INSTRUCTION

INSTRUCTION	Ý nghĩa
FROM	Dùng để chỉ ra image được build từ image gốc nào. Tùy vào mỗi ứng dụng cần đóng gói mà chúng ta sẽ sử dụng image gốc khác nhau
RUN	Dùng để chạy một lệnh nào đó khi build image.
WORKDIR	Dùng để thiết lập thư mục làm việc. Mọi chỉ thị RUN, CMD, ENTRYPOINT, COPY và ADD sau đó đều sẽ diễn ra bên trong thư mục WORKDIR này
COPY	COPY thư mục nguồn từ máy host vào filesystem của image
CMD	Dùng để cung cấp câu lệnh mặc định sẽ được chạy khi Docker Container khởi động từ Image đã build, chỉ có thể có duy nhất 1 chỉ thị CMD

- <https://docs.docker.com/engine/reference/builder>
- https://kapeli.com/cheat_sheets/Dockerfile.docset/Contents/Resources/Documents/index

TASK #2

VIẾT DOCKERFILE ĐÓNG GÓI ỨNG DỤNG NODEJS

yêu cầu

Cho source code của 1 ứng dụng React tại địa chỉ

<https://github.com/ahfarmer/calculator>

Thực hiện các bước sau

- Clone source code về máy
- Bên trong thư mục chứa source code, viết file Dockerfile
- Build ra Docker image
- Chạy thử Docker image, expose cổng 3000 ra cổng 8080 trên máy host

Dockerfile cho ứng dụng NodeJS

- Sử dụng base image `node:12-alpine`
- Copy các file `package.json` và `package-lock.json` từ host vào base image
- Chạy lệnh `npm install` để download các thư viện dependencies
- Copy toàn bộ file và thư mục từ host vào image
- Viết lệnh CMD (tham khảo source code trên Github lệnh để chạy ứng dụng)

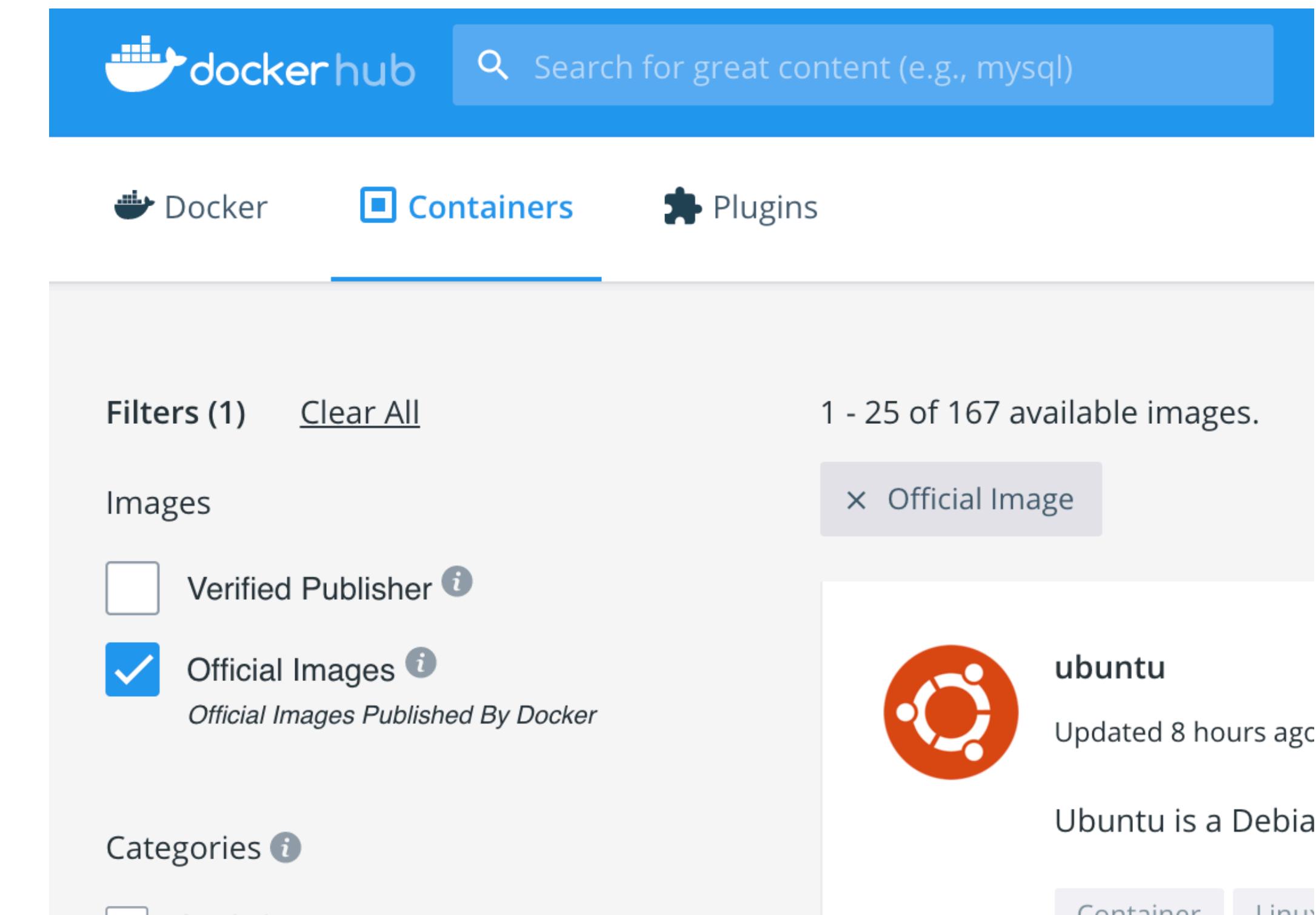


BEST PRACTICES với DOCKERFILE

Chọn base image offical và có dung lượng nhẹ

Nếu có thể, hãy sử dụng base image hệ alpine

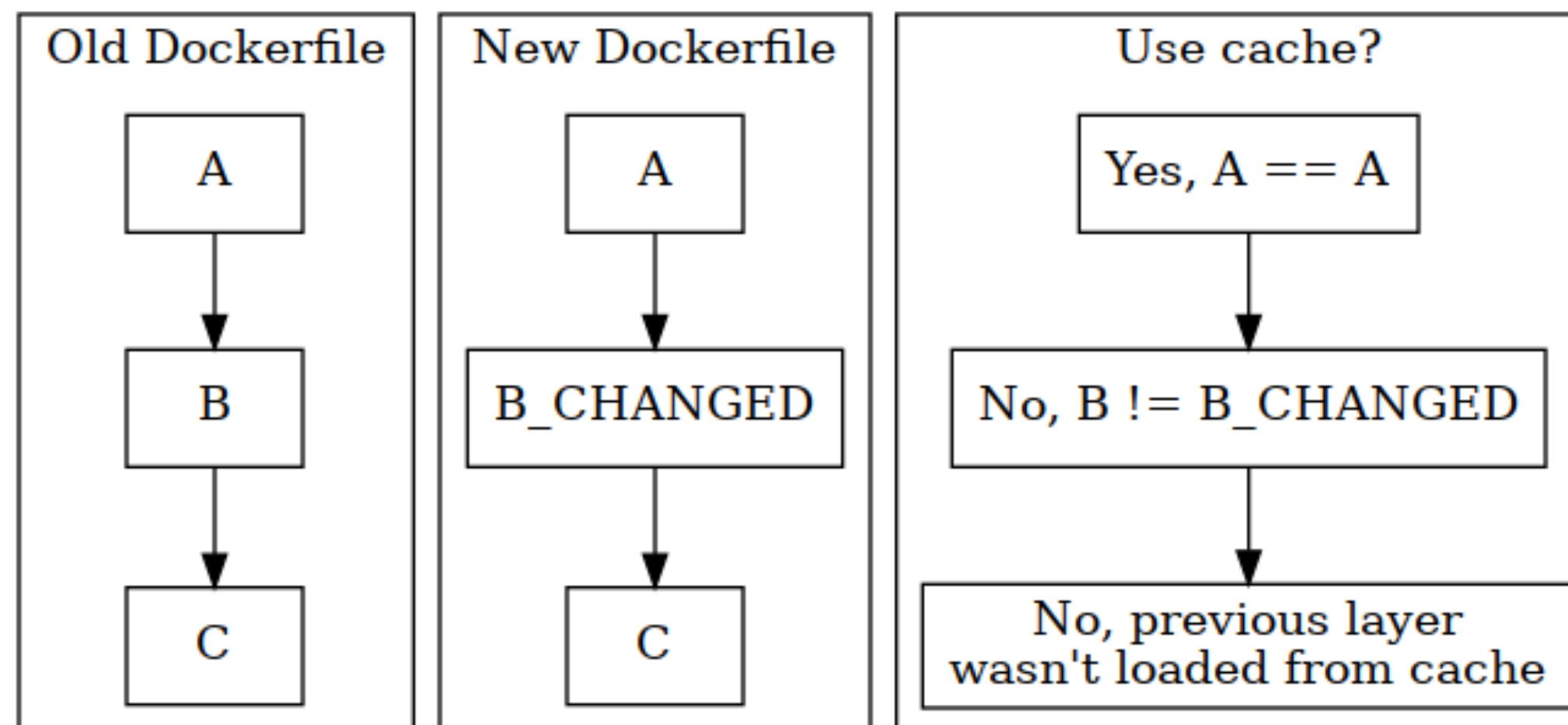
- node:<version>-alpine
- maven:<version>-alpine
- openjdk:<version>-alpine
- golang:<version>-alpine
- python:<version>-alpine
- mcr.microsoft.com/dotnet/sdk:5.0-alpine
- mcr.microsoft.com/dotnet/aspnet:5.0-alpine



<https://hub.docker.com>

Tận dụng layer caching

Phần lệnh ít thay đổi sẽ ở trên, phần thay đổi thường xuyên sẽ ở dưới



Order matters for caching

```
FROM debian
COPY ./app
RUN apt-get update
RUN apt-get -y install openjdk-8-jdk ssh vim
COPY . /app
CMD ["java", "-jar", "/app/target/app.jar"]
```

Order from least to most frequently changing content.

Sử dụng file .dockerignore

Ignore những file/folder không cần thiết cho quá trình build code

.env	.npm	logs
.node_repl_history	*.log	dist
.node_repl_history	*.sublime-project	docs
*.sublime-workspace	*.md	.git
coverage	Dockerfile	.nyc_output
docker-compose	.grunt	deploy
.lock-wscript	.gitlab-ci.yml	node_modules

<https://docs.docker.com/engine/reference/builder/#dockerignore-file>

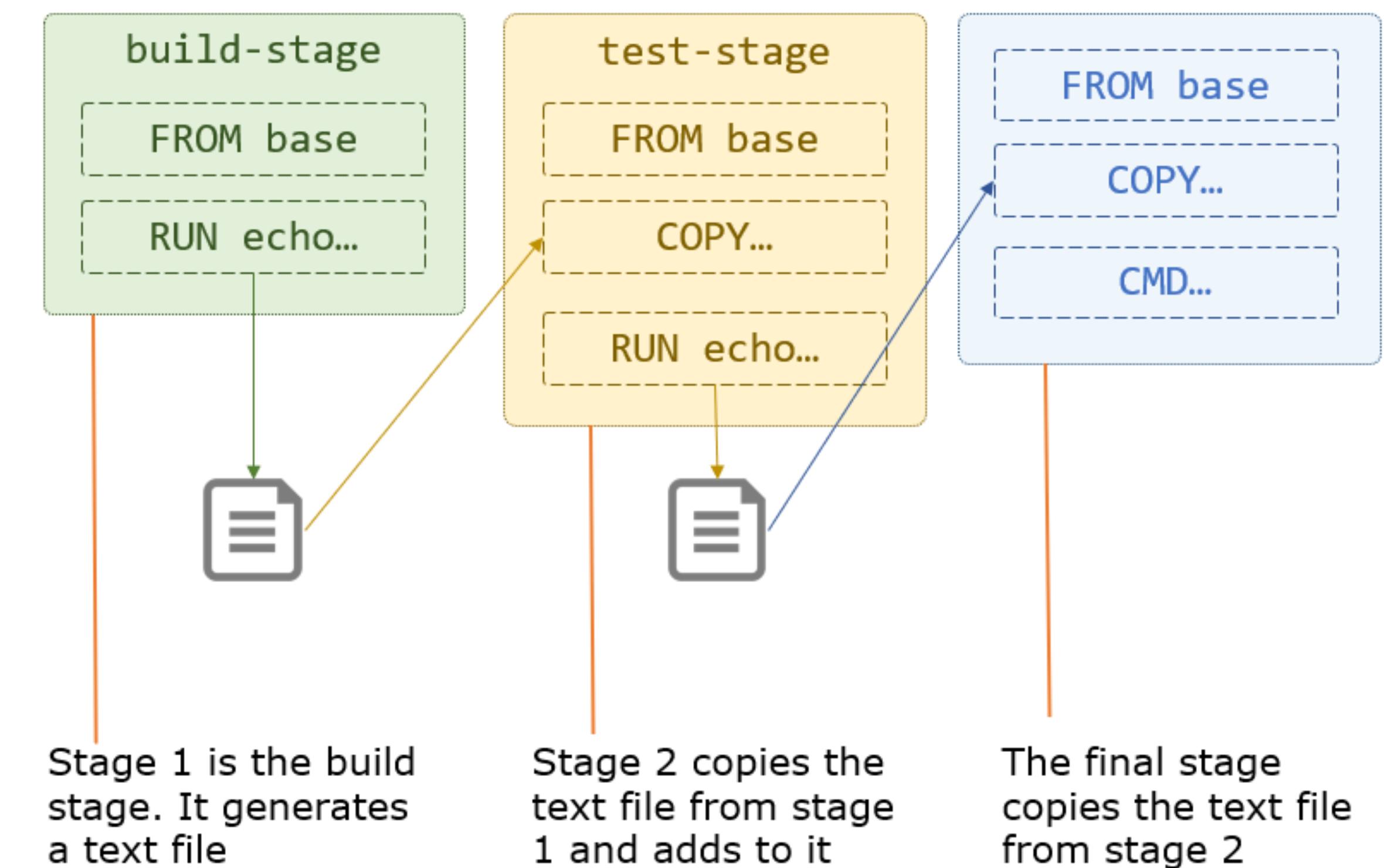


MULTI-STAGE BUILD

Multi-stage build là gì?

Nếu chương trình chỉ cần chạy 1 hoặc vài file thực thi, cấu hình, nhưng để có được các file ấy lại cần cài đặt môi trường, package, module rất phức tạp và tốn dung lượng khiến cho images của bạn nặng nề.

→ có thể thực hiện các công việc cài đặt đó ở các stage có đầy đủ môi trường, rồi copy file cần thiết sang stage có base image nhẹ hơn, nhưng đủ để execute/run ứng dụng của bạn.



Multi-stage build là gì?



```
FROM golang:1.16
WORKDIR /go/src/github.com/alexellis(href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .
```

```
FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=0 /go/src/github.com/alexellis(href-counter/app .
CMD [ "./app" ]
```

TASK #3

VIẾT DOCKERFILE CHO ỨNG DỤNG ANGULAR

yêu cầu

Cho source code của ứng dụng <https://github.com/handuy/angular-hero>

Viết Dockerfile dưới dạng multi-stage, sau đó build ứng dụng thành Docker Image và khởi tạo container.

các bước tiến hành

1. Clone source code về máy
2. Bên trong thư mục chứa source code, tạo file Dockerfile, chia làm 2 stage:

Stage 1:

- Sử dụng base image **node:13-alpine**
- **SET WORKDIR = /app**
- Copy file package.json vào image, sau đó chạy lệnh **npm install**
- Copy các file và thư mục còn lại vào image, sau đó chạy lệnh **npm run build**

Stage 2:

- Sử dụng base image **nginx:1.17-alpine**
- Copy thư mục **/app/dist** được tạo ra từ stage 1 vào thư mục **/usr/share/nginx/html**

chạy container

3. Từ Dockerfile build thành Docker Image có tên là angular-app.
4. Khởi động container từ image angular-app: container chạy ngầm, expose cổng 80 của container ra cổng 8001 của host.
5. Truy cập localhost:8001 để kiểm tra kết quả.

TASK # 4

VIẾT DOCKERFILE ĐÓNG GÓI ỨNG DỤNG SPRINGBOOT

yêu cầu

Source code của ứng dụng: <https://github.com/handuy/spring-app-demo>

Cách 1: Viết Dockerfile dưới dạng multi-stage

Cách 2: Thực hiện build source code ra thư mục target chứa các file .jar, sau đó copy thư mục target vào image

các bước tiến hành (cách 1)

1. Clone source code về máy
2. Bên trong thư mục chứa source code, tạo file Dockerfile, chia làm 2 stage:

Stage 1:

- Sử dụng base image **maven:ibmjava-alpine**
- Copy source code vào image
- Từ source code build ra file **websocket-demo-0.0.1-SNAPSHOT.jar** bằng lệnh: **mvn clean package**.
- File **websocket-demo-0.0.1-SNAPSHOT.jar** sẽ nằm trong thư mục **target** của source code

Stage 2:

- Sử dụng base image **openjdk:8-alpine**
- Copy file **websocket-demo-0.0.1-SNAPSHOT.jar** từ stage 1 sang stage 2
- Dùng lệnh sau làm CMD: **java -Djava.security.egd=file:/dev/.urandom -jar websocket-demo-0.0.1-SNAPSHOT.jar**

các bước tiến hành (cách 2)

1. Clone source code về máy
2. Bên trong thư mục chứa source code, sử dụng maven để build source code ra thư mục target
3. Viết Dockerfile

Yêu cầu:

- Sử dụng base image **openjdk:8-alpine**
- Copy file **websocket-demo-0.0.1-SNAPSHOT.jar** vào docker image
- Dùng lệnh sau làm CMD: **java -Djava.security.egd=file:/dev/.urandom -jar websocket-demo-0.0.1-SNAPSHOT.jar**

chạy container

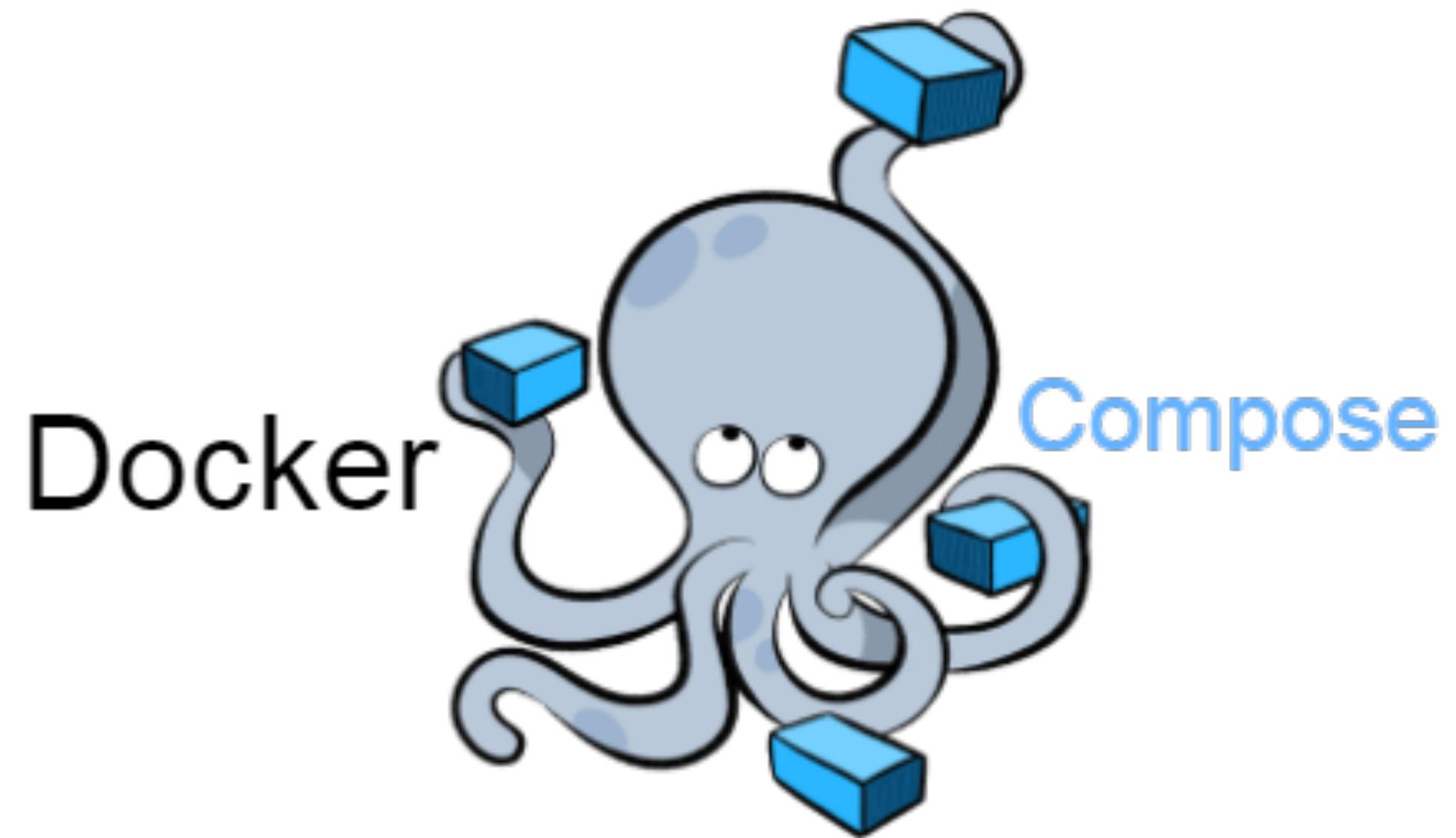
1. Từ Dockerfile build thành Docker Image có tên là spring-app.
2. Khởi động container từ image spring-app: container chạy ngầm, expose cổng 8080 của container ra cổng 9002 của host.
3. Truy cập localhost:9002 để kiểm tra kết quả.



DOCKER COMPOSE LÀ GÌ?

Docker compose là gì?

- Định nghĩa cấu hình các container vào file YAML.
- Tự động đặt các container vào cùng 1 network, khai báo biến môi trường, volume cho ứng dụng.
- Chạy tất cả các container cần thiết cho ứng dụng: **docker-compose up -d**



Docker compose là gì?

```
$ docker run -d --rm --name db -e
  MYSQL_ROOT_PASSWORD=somewordpress -e
  MYSQL_DATABASE=exampledbs -e
  MYSQL_USER=exampleuser -e
  MYSQL_PASSWORD=examplepass -v
  db_data:/var/lib/mysql --net wordpress-network
mysql:5.7
```

```
$ docker run -d --rm --name wordpress -e
  WORDPRESS_DB_HOST=db -e
  WORDPRESS_DB_USER=exampleuser -e
  WORDPRESS_DB_PASSWORD=examplepass -e
  WORDPRESS_DB_NAME=exampledbs -v
  wordpress:/var/www/html --net wordpress-network -p
  8080:80 wordpress:latest
```

```
version: '3.3'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: exampledbs
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      db_data: {}
```

Docker compose là gì?

1 DNS name trong bridge network giữa các container

4 Tự động chạy lại app khi app crash

5 Cấu hình các biến môi trường

6 Kết nối tới db service

```

version: '3.3'

services:
  db:
    image: mysql:5.7   Tên image 2
    volumes:
      - db_data:/var/lib/mysql   Map volume ra host 3
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      db_data: {}

```

cú pháp docker compose

Chi tiết xem tại: <https://docs.docker.com/compose/compose-file>

TASK #5

VIẾT DOCKER COMPOSE CHO ỨNG DỤNG NODEJS & POSTGRESQL

yêu cầu

1. Viết Dockerfile để tạo Docker Image cho ứng dụng NodeJS:

<https://github.com/handuy/nodejs-todolist>

2. Viết docker-compose.yml để triển khai ứng dụng

Yêu cầu chi tiết xem ở 2 slide sau

Dockerfile cho ứng dụng NodeJS

- Sử dụng base image là **node:13-alpine**
- CMD: **node server.js**
- Đóng gói thành Docker Image bằng lệnh docker build

viết docker-compose.yml

Với container chạy NodeJS:

- Sử dụng Docker Image cho ứng dụng NodeJS vừa build ở bước 1
- Expose cổng 8080 của NodeJS app ra cổng 8181 của máy host
- PostgreSQL được khởi tạo trước, sau đó mới đến NodeJS app

Với container chạy PostgreSQL

- Bind mount file **init.sql** từ host vào **/docker-entrypoint-initdb.d/** của PostgreSQL container để tạo sẵn bảng
- Volume thư mục **/var/lib/postgresql/data** của container ra một thư mục bất kỳ trên máy host
- Tên của service chạy PostgreSQL phải là **db**
- Biến môi trường **POSTGRES_PASSWORD** có giá trị là **postgres**

nội dung file init.sql

```
CREATE TABLE task(  
    id SERIAL PRIMARY KEY NOT NULL, task text,  
    status INTEGER DEFAULT 0  
) ;
```

TASK # 6

VIẾT DOCKER COMPOSE CHO ỨNG DỤNG NODEJS & MONGODB

yêu cầu

Chi tiết xem tại: <https://github.com/handuy/nodejs-mongodb>

TASK #7

VIẾT DOCKER COMPOSE CHO ỨNG DỤNG SPRINGBOOT & MYSQL

yêu cầu

1. Viết Dockerfile để tạo Docker Image ứng dụng SpringBoot:

<https://github.com/handuy/obo>

2. Viết docker-compose.yml để triển khai ứng dụng

Yêu cầu chi tiết xem ở 2 slide sau

Dockerfile cho ứng dụng SpringBoot

Cách 1:

- Build source code ra thư mục target chứa file.jar
- Sử dụng base image là openjdk:<java-version>-alpine
- COPY file.jar vào image
- CMD để bật ứng dụng: java-jar<tên-file>.jar

Cách 2:

- Sử dụng base image là maven
- CMD để start ứng dụng: mvn spring-boot:run

viết docker-compose.yml

Với container chạy MySQL

- Mount volume file obo.sql từ host vào **/docker-entrypoint-initdb.d/init.sql** của MySQL container để mockup data.
- Link tải file obo.sql: <https://techmaster.vn/media/download/sourcecode/btq4ftc51co41h2qcrc0>
- Tên của service chạy MySQL phải là **mysql**
- Khi container chạy MySQL được khởi tạo, bên trong đã có sẵn 1 database tên là obo, 1 user admin với password là 123456 (tham khảo cách set biến môi trường cho MySQL [tại đây](#))

Với container chạy ứng dụng SpringBoot

- Sử dụng Docker Image vừa build ở bước trước
- Expose cổng 8080 của SpringBoot app ra cổng 8005 của máy host
- MySQL được khởi tạo trước, sau đó mới đến SpringBoot app

Thanks for watching

Email: thuongnn666@gmail.com
