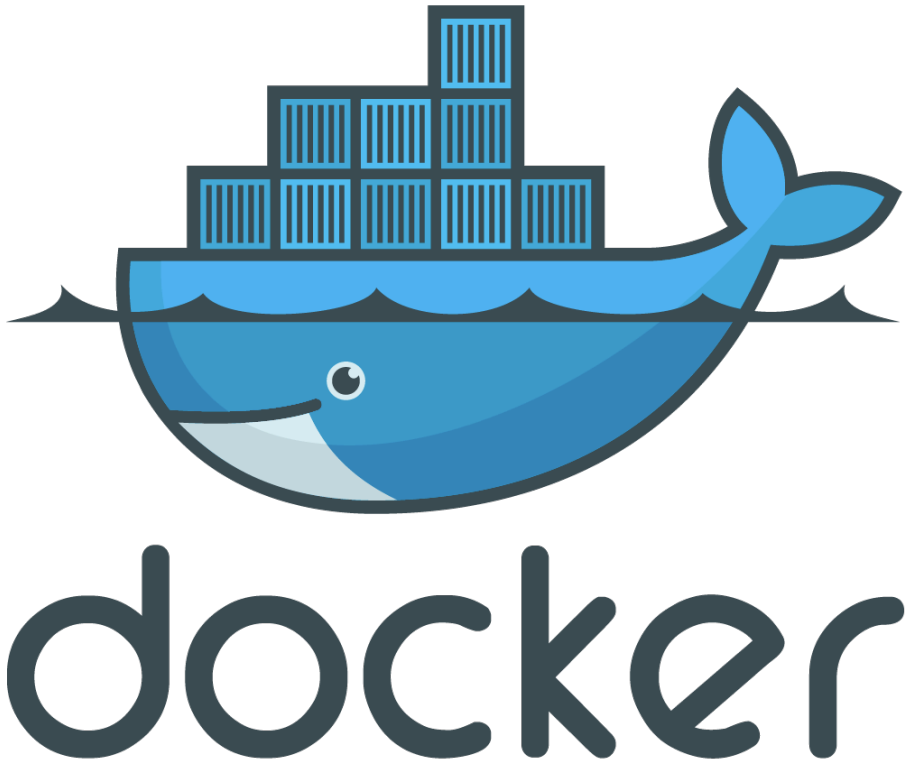
TechMaster
Học là có việc

Docker

Nguyễn Hàn Duy
duy@techmaster.vn

Nội dung

- Dockerfile
- Docker Compose



Dockerfile

Dockerfile

- 1 file text lưu các bước để tạo một Docker Image
- Mỗi bước có cấu trúc dạng: INSTRUCTION arguments

Dockerfile example

```
FROM node:alpine

RUN mkdir -p /usr/src/app

WORKDIR /usr/src/app

COPY package.json /usr/src/app/

RUN npm install

COPY . /usr/src/app

RUN npm run build

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

Dockerfile docs

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

Task #1: 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

Yêu cầu:

- 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)

Tips

Chọn base image 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**

Tận dụng layer caching

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

```
FROM node:13-alpine

WORKDIR /app

COPY package.json package-lock.json ./

RUN npm install

COPY . .

CMD ["npm", "start"]
```

Sử dụng file .dockerignore

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

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

Task #2: Viết Dockerfile đóng gói ứng dụng Java

Các bước tiến hành

1. Tạo 1 thư mục có tên docker-java để chứa source code
2. Bên trong thư mục docker-java: Tạo file **Hello.java** (nội dung chi tiết xem ở slide dưới)
3. Bên trong thư mục docker-java: Tạo file **Dockerfile** (yêu cầu chi tiết xem ở slide dưới)
4. Dùng lệnh **docker build** để tạo Docker Image
5. Tạo container từ Image vừa tạo bằng lệnh: **docker run** tên-image-vừa-tạo. Màn hình terminal cần hiển thị dòng chữ: **Chào các bạn**

Source code ứng dụng Java

File Hello.java có nội dung như sau:

```
class Hello{
    public static void main(String[] args){
        System.out.println("Chào các bạn");
    }
}
```

Yêu cầu với file Dockerfile

- Sử dụng base image **openjdk:8-alpine**
- Sau khi đã copy source code vào image, chạy lệnh:

  javac Hello.java
- Câu lệnh để chạy ứng dụng: **java Hello**

Multi-stage build

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

```
#Dockerfile
FROM golang:alpine as builder

COPY . /go/src/api
WORKDIR /go/src/api

RUN apk update \
&& apk add git \
&& go get ./vendor/database \
&& go get ./ \
&& go build \
&& rm -rf /var/cache/apk/*


FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /app
COPY --from=builder /go/src/api/ /app
CMD ["./api"]
EXPOSE 8001
```

Task #3: Viết Dockerfile cho ứng dụng Angular

Yêu cầu

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**
- 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**

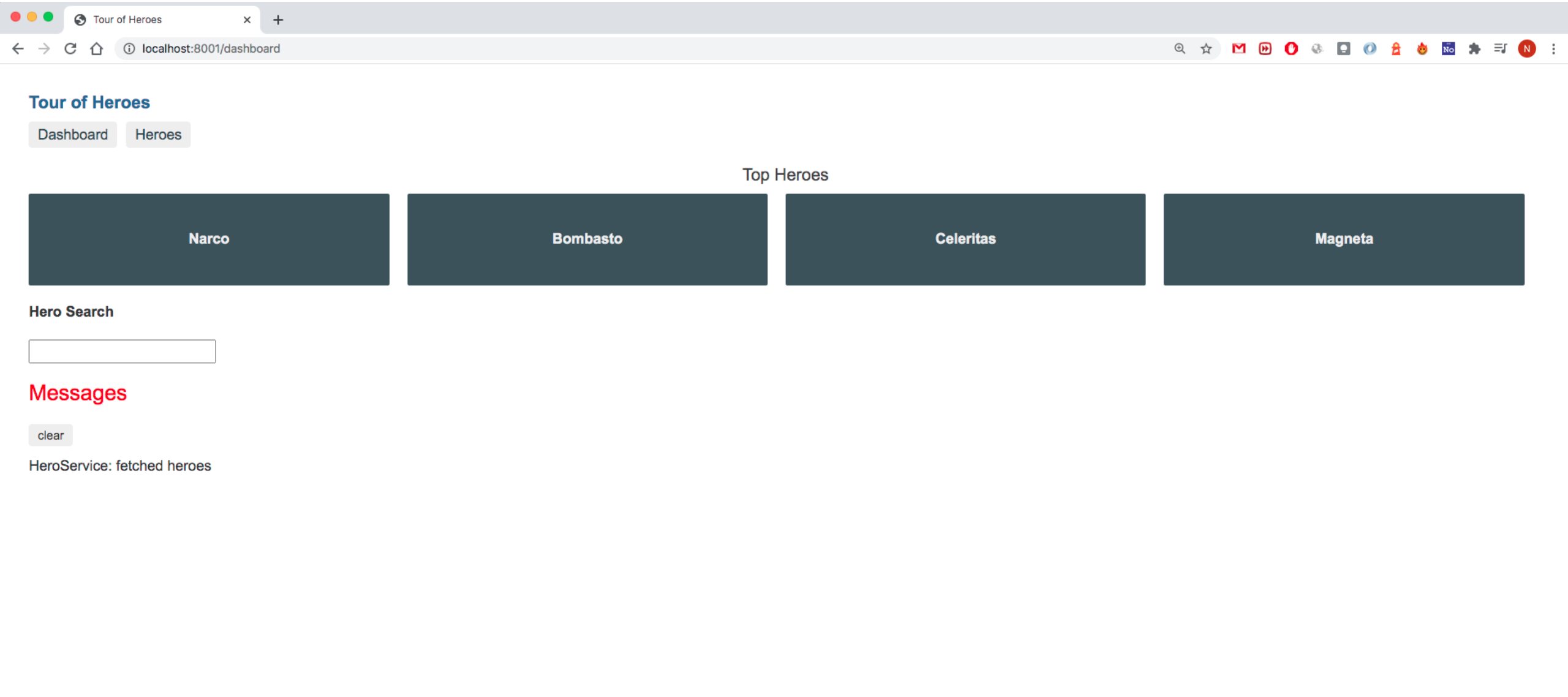
Stage 2:

- 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**
- Dùng lệnh sau để start container: **nginx -g daemon off;**

Các bước tiến hành (tiếp theo)

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

Giao diện ứng dụng Angular



Task #4: Viết Dockerfile đóng gói ứng dụng SpringBoot

Yêu cầu

Source code của ứng dụng:

https://github.com/handuy/spring-app-demo

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 **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

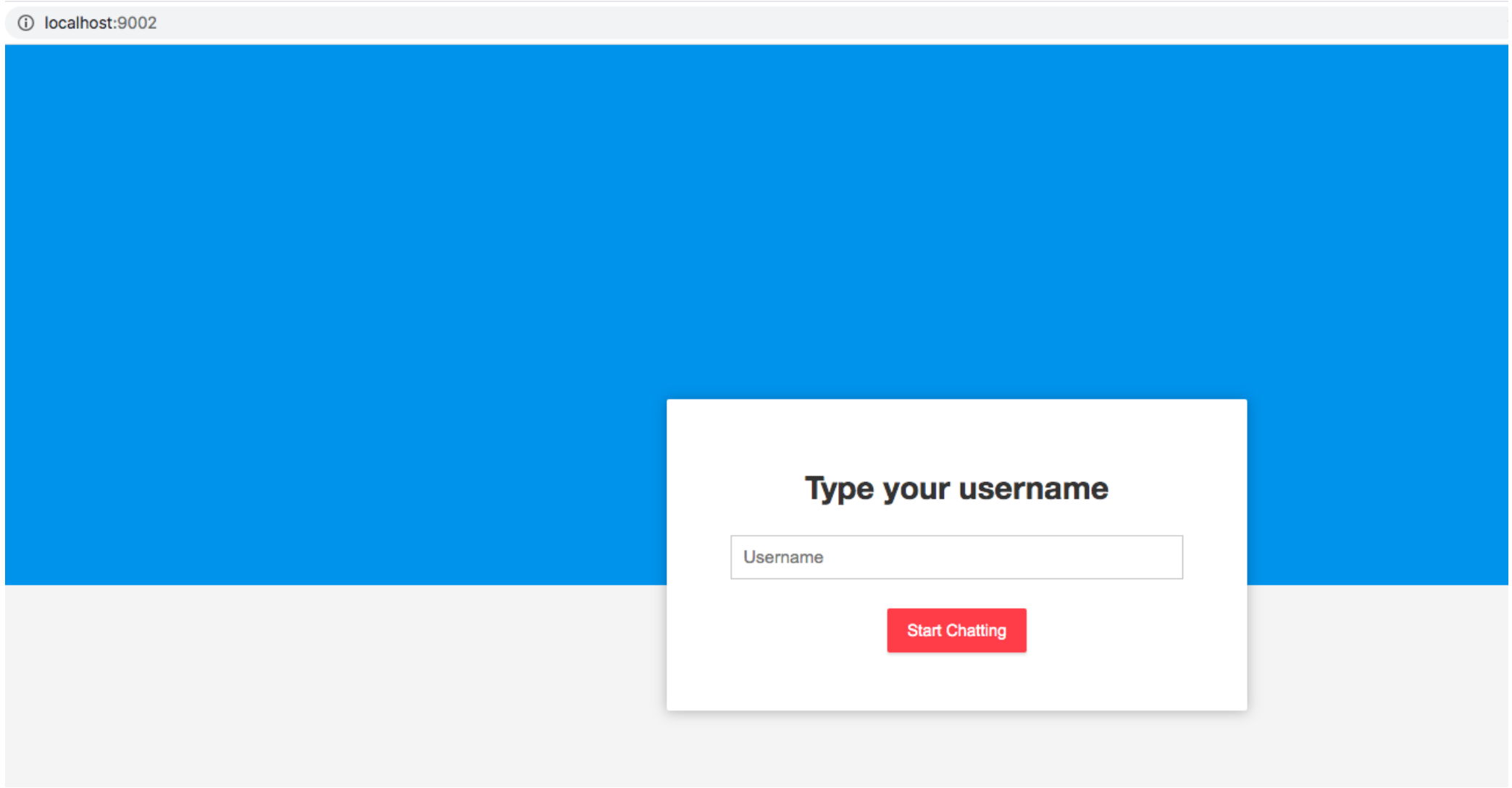
Stage 2:

- 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 để start container:
  java -Djava.security.egd=file:/dev/./urandom -jar websocket-demo-0.0.1-SNAPSHOT.jar

Các bước tiến hành (tiếp theo)

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

Giao diện ứng dụng Spring Boot



Docker Compose

Docker Compose

- Định nghĩa cấu hình các container vào file YAML
- Tự động đặt các container vào cùng 1 network
- Chạy tất cả các container cần thiết cho ứng dụng:

docker-compose up -d

docker run

docker run … image-1

docker run … image-2

docker run … image-3

…

docker-compose.yml

```yaml
version: '3.3'

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

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

docker-compose.yml

docker-compose.yml

```yaml
version: '3.3'
services:
    db:
        image: mysql:5.7
        volumes:
            - /var/lib/mysql
        restart: always
        environment:
            MYSQL_ROOT_PASSWORD: somewordpress
            MYSQL_DATABASE: wordpress
            MYSQL_USER: wordpress
            MYSQL_PASSWORD: wordpress

    wordpress:
        depends_on:
            - db
        image: wordpress:latest
        ports:
            - "8000:80"
        restart: always
        environment:
            WORDPRESS_DB_HOST: db:3306
            WORDPRESS_DB_USER: wordpress
            WORDPRESS_DB_PASSWORD: wordpress
            WORDPRESS_DB_NAME: wordpress
```

DNS name trong bridge network giữa các container

Tên image

Map volume ra host

Tự động chạy lại khi app crash hoặc chạy lại docker daemon

Cấu hình các biến môi trường

Kết nối tới service db

Cú pháp docker-compose.yml

https://docs.docker.com/compose/compose-file/

Task #5: Viết docker-compose.yml 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 dung

Yêu cầu chi tiết xem ở 2 slide sau

Dockerfile cho ứng dụng NodeJS

- Sử dụng base image là **node:13-alpine**
- Câu lệnh để start ứng dụng: **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.yml cho ứng dụng NodeJS + MongoDB

Yêu cầu

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

Task #7: Viết docker-compose.yml 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

- Sử dụng base image là **maven**
- Câu lệnh để start ứng dụng: **mvn spring-boot:run**
- Build thành Docker Image bằng lệnh docker build

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 download file obo.sql: https://techmaster.vn/media/download/source-code/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 https://hub.docker.com/_/mysql)

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