**Phần 2:**

|  |  |
| --- | --- |
| Bài 1: Chạy một container đơn giản với Docker Compose |  |
|  | ***Yêu cầu:*** |
|  | Tạo một container chạy Nginx bằng Docker Compose. |
|  | Map cổng 8080 của máy host với cổng 80 của container. |
|  |  |

Trong thư mục Nginx, tạo một thư mục con tên html để chứa file HTML:

mkdir html

Tạo file HTML

Tạo một file index.html trong thư mục html với nội dung đơn giản

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>Hello World</h1>

</body>

</html>

### Tạo file Docker Compose

Trong thư mục Nginx, tạo file docker-compose.yml với nội dung sau:

version: '3'

services:

  web:

    image: nginx:latest

    container\_name: nginx-container

    ports:

      - "80:80"

    volumes:

      - ./html:/usr/share/nginx/html

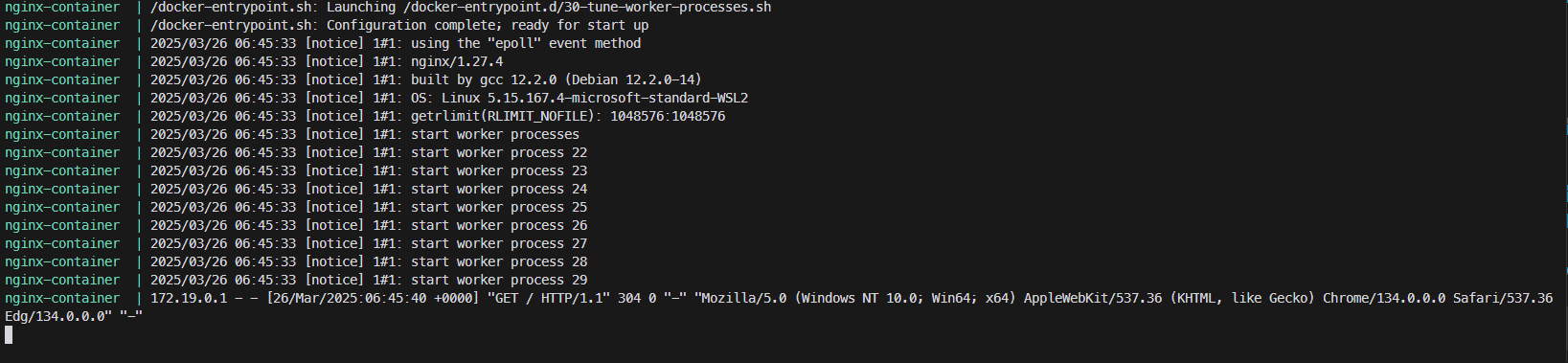
    restart: unless-stopped

#### **Giải thích chi tiết:**

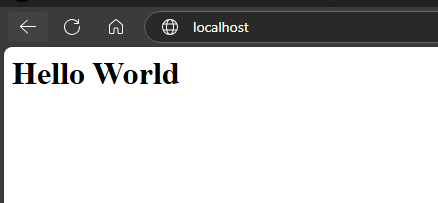
* version: '3': Định dạng file Docker Compose (phiên bản 3 là phổ biến và ổn định).
* services: Định nghĩa các container sẽ chạy.
* nginx: Tên service (có thể đặt tùy ý).
* image: nginx:latest: Dùng image Nginx mới nhất từ Docker Hub.
* container\_name: Đặt tên cụ thể cho container để dễ quản lý.
* ports: Ánh xạ port 8080 trên máy host tới port 80 trong container (bạn có thể đổi 8080 thành port khác nếu port 80 đã bị chiếm dụng).
* volumes: Liên kết thư mục ./html (trên host) với /usr/share/nginx/html (thư mục mặc định Nginx phục vụ nội dung web).
* restart: unless-stopped: Đảm bảo container tự khởi động lại nếu crash, trừ khi bạn dừng nó bằng tay.

**Chạy Docker Compose**

docker-compose up



**Kết quả:**



Để ánh xạ (map) cổng 8080 của máy host với cổng 80 của container trong Docker Compose ta chỉnh sửa file docker-compose.yml như sau:

version: '3'

services:

  nginx:

    image: nginx:latest

    container\_name: my-nginx

    ports:

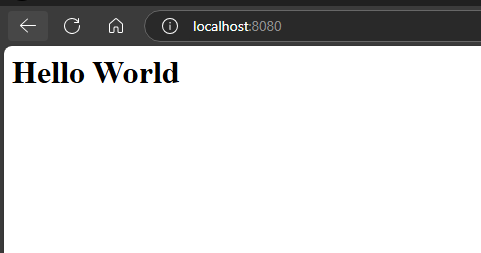
      - "8080:80"  *# Ánh xạ cổng 8080 (host) với cổng 80 (container)*

    volumes:

      - ./html:/usr/share/nginx/html

    restart: unless-stopped

Dùng lệnh docker-compose up và kết quả



|  |  |  |
| --- | --- | --- |
| Bài 2: Chạy MySQL với Docker Compose |  |  |
|  | ***Yêu cầu:*** |  |
|  | Tạo một container chạy MySQL phiên bản 8.0. |  |
|  | Đặt username là user, password là password và database là mydb |  |

Tạo nội dung của file docker-compose.yml:

version: '3'

services:

  mysql:

    image: mysql:8.0

    container\_name: my-mysql

    environment:

      MYSQL\_USER: user

      MYSQL\_PASSWORD: password

      MYSQL\_DATABASE: mydb

      MYSQL\_ROOT\_PASSWORD: rootpassword

    ports:

      - "3306:3306"

    volumes:

      - mysql-data:/var/lib/mysql

    restart: unless-stopped

volumes:

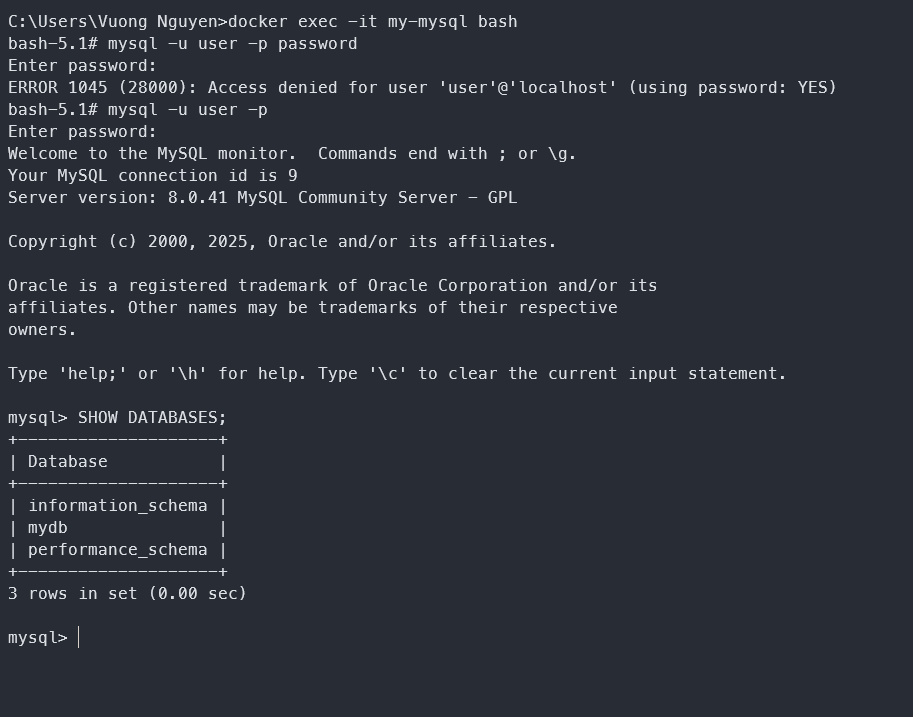
  mysql-data:

#### **Giải thích chi tiết:**

1. **version: '3'**: Phiên bản định dạng Docker Compose.
2. **services**: Định nghĩa các service (container).
3. **mysql**: Tên service (có thể đặt tùy ý).
4. **image: mysql:8.0**: Sử dụng image MySQL phiên bản 8.0 từ Docker Hub.
5. **container\_name: my-mysql**: Đặt tên container để dễ quản lý.
6. **environment**: Các biến môi trường để cấu hình MySQL:
   * MYSQL\_USER: user: Tạo user với tên user.
   * MYSQL\_PASSWORD: password: Mật khẩu cho user user.
   * MYSQL\_DATABASE: mydb: Tạo database tên mydb.
   * MYSQL\_ROOT\_PASSWORD: rootpassword: Mật khẩu cho tài khoản root (bắt buộc khi dùng MySQL).
7. **ports: "3306:3306"**: Ánh xạ cổng 3306 trên host với cổng 3306 trong container (cổng mặc định của MySQL).
8. **volumes: mysql-data:/var/lib/mysql**: Lưu trữ dữ liệu MySQL vào volume mysql-data để dữ liệu không bị mất khi container dừng.
9. **restart: unless-stopped**: Container tự khởi động lại trừ khi bị dừng thủ công.
10. **volumes: mysql-data**: Khai báo volume để lưu dữ liệu.

Sau đó tiến hành kiểm tra:

* docker exec -it my-mysql bash
* mysql -u user –p
* Nhập pwd: password
* SHOW DATABASES;



|  |  |
| --- | --- |
| Bài 3: Kết nối MySQL với PHPMyAdmin |  |
|  | ***Yêu cầu:*** |
|  | Chạy MySQL và PHPMyAdmin với Docker Compose. |
|  | PHPMyAdmin chạy trên cổng 8081. |

Tạo file docker-compose.yml với nội dung:

version: '3'

services:

mysql:

image: mysql:8.0

container\_name: my-mysql

environment:

MYSQL\_USER: user

MYSQL\_PASSWORD: password

MYSQL\_DATABASE: mydb

MYSQL\_ROOT\_PASSWORD: rootpassword

ports:

- "3306:3306"

volumes:

- mysql-data:/var/lib/mysql

restart: unless-stopped

phpmyadmin:

image: phpmyadmin:latest

container\_name: my-phpmyadmin

environment:

PMA\_HOST: mysql

PMA\_USER: user

PMA\_PASSWORD: password

ports:

- "8081:80"

depends\_on:

- mysql

restart: unless-stopped

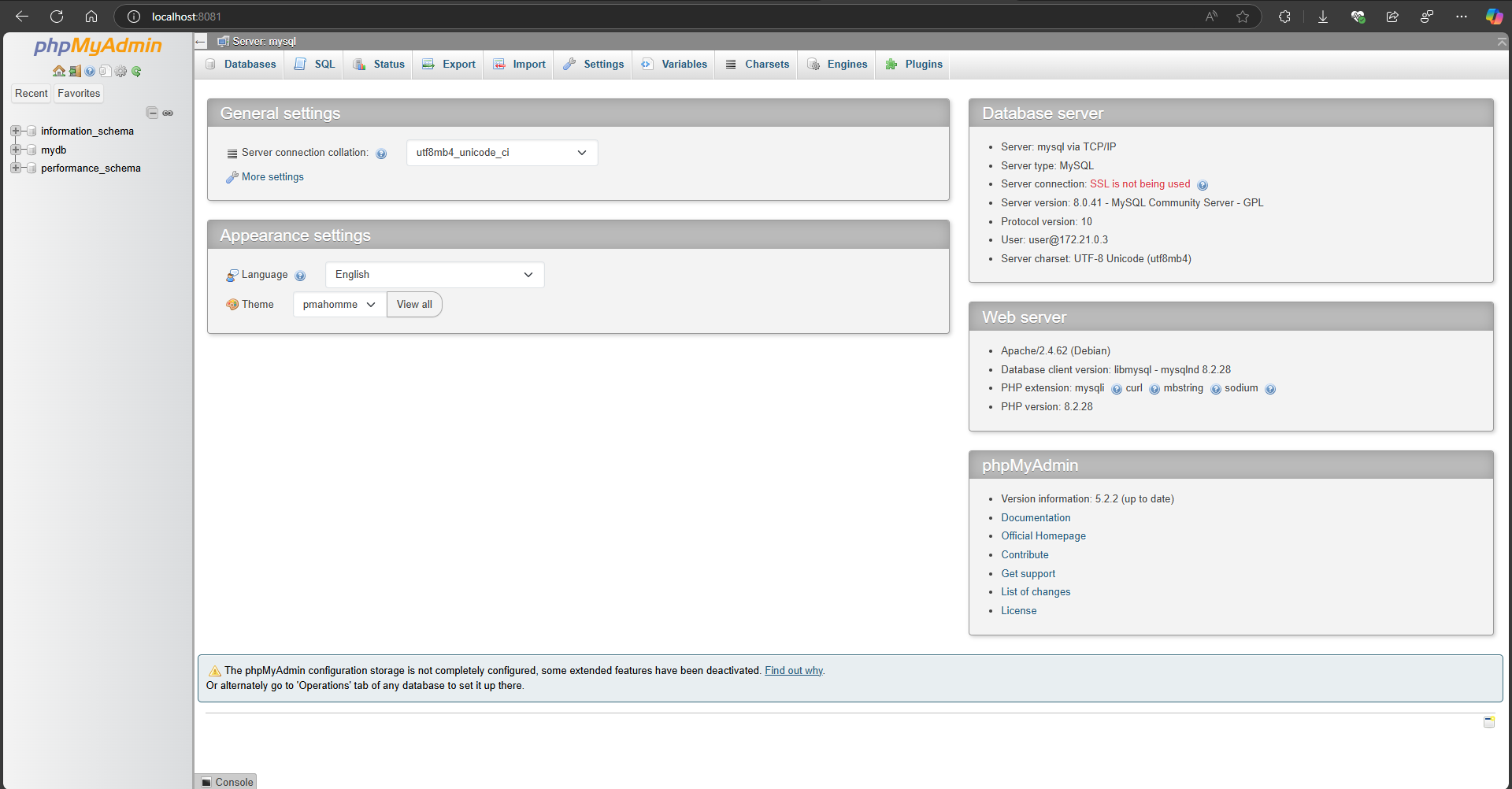
volumes:

mysql-data:

#### Giải thích chi tiết:

1. **Service mysql**:
   * image: mysql:8.0: Dùng MySQL phiên bản 8.0.
   * environment: Cấu hình user user, password password, database mydb, và root password rootpassword.
   * ports: "3306:3306": Ánh xạ cổng 3306 để truy cập MySQL từ host (nếu cần).
   * volumes: Lưu dữ liệu vào volume mysql-data.
2. **Service phpmyadmin**:
   * image: phpmyadmin:latest: Dùng image PHPMyAdmin mới nhất.
   * container\_name: my-phpmyadmin: Đặt tên container.
   * environment:
     + PMA\_HOST: mysql: Kết nối tới service mysql (tên service trong Docker Compose).
     + PMA\_USER: user: Username để đăng nhập.
     + PMA\_PASSWORD: password: Password tương ứng.
   * ports: "8081:80": Ánh xạ cổng 8081 trên host tới cổng 80 trong container (PHPMyAdmin mặc định chạy trên cổng 80).
   * depends\_on: mysql: Đảm bảo MySQL khởi động trước PHPMyAdmin.
3. **volumes: mysql-data**: Volume để lưu trữ dữ liệu MySQL.

Chạy Docker Compose và kiểm tra trên localhost:8081:



|  |  |
| --- | --- |
| Bài 4: Chạy ứng dụng Node.js với Docker Compose |  |
|  | ***Yêu cầu:*** |
|  | Chạy một ứng dụng Node.js đơn giản với Express. |

Cấu trúc thư mục:

node-app/

├── src/

│ ├── index.js

│ └── package.json

├── Dockerfile

└── docker-compose.yml

**Tạo file package.json**:

{

"name": "node-express-app",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"start": "node index.js"

},

"dependencies": {

"express": "^4.18.2"

}

}

Tạo file index.js

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello, World!');

});

app.listen(port, () => {

console.log(`Server running at http://localhost:${port}`);

});

Tạo Dockerfile:

FROM node:18

WORKDIR */usr/src/app*

COPY *src/package.json* *./*

RUN *npm* *install*

COPY *src/index.js* *./*

EXPOSE *3000*

CMD *[*"npm"*,* "start"*]*

#### Giải thích chi tiết:

 **FROM node:18**:

* **Ý nghĩa**: Chỉ định image cơ sở (base image) mà bạn muốn xây dựng từ đó. Ở đây, chúng ta dùng image chính thức của Node.js phiên bản 18 từ Docker Hub.
* **Vai trò**: Cung cấp môi trường Node.js để chạy ứng dụng.

 **WORKDIR /usr/src/app**:

* **Ý nghĩa**: Đặt thư mục làm việc mặc định trong container là /usr/src/app. Tất cả các lệnh tiếp theo (COPY, RUN, CMD) sẽ thực thi trong thư mục này.
* **Vai trò**: Đảm bảo file và lệnh được thực thi trong một thư mục cố định.

 **COPY src/package.json ./**:

* **Ý nghĩa**: Sao chép file package.json từ thư mục src trên host vào thư mục /usr/src/app trong container.
* **Vai trò**: Chuẩn bị file chứa thông tin dependencies (như Express) để cài đặt.

 **RUN npm install**:

* **Ý nghĩa**: Thực thi lệnh npm install trong container để cài đặt các dependencies được liệt kê trong package.json (ví dụ: Express).
* **Vai trò**: Tạo thư mục node\_modules chứa các thư viện cần thiết.

 **COPY src/index.js ./**:

* **Ý nghĩa**: Sao chép file index.js từ thư mục src trên host vào /usr/src/app trong container.
* **Vai trò**: Đưa code ứng dụng vào container.

 **EXPOSE 3000**:

* **Ý nghĩa**: Thông báo rằng container sẽ lắng nghe trên cổng 3000 (cổng mà ứng dụng Node.js chạy). Đây chỉ là tài liệu, không thực sự mở cổng.
* **Vai trò**: Gợi ý cho người dùng và Docker về cổng ứng dụng sử dụng.

 **CMD ["npm", "start"]**:

* **Ý nghĩa**: Chỉ định lệnh mặc định để chạy khi container khởi động. Ở đây, nó chạy npm start, tương ứng với node index.js trong package.json.
* **Vai trò**: Khởi động ứng dụng Node.js.

Tiếp theo là docker-compose.yml

version: '3'

services:

  app:

    build:

      context: .

      dockerfile: Dockerfile

    container\_name: node-app

    ports:

      - "3000:3000"

    restart: unless-stopped

#### Giải thích chi tiết:

* **version: '3'**:
* **Ý nghĩa**: Chỉ định phiên bản định dạng của Docker Compose. Phiên bản 3 là phổ biến và hỗ trợ nhiều tính năng hiện đại.
* **Vai trò**: Đảm bảo cú pháp file tương thích với Docker Compose.
* **services:**:
* **Ý nghĩa**: Khai báo các service (container) mà Docker Compose sẽ quản lý. Ở đây chỉ có một service tên là app.
* **Vai trò**: Tổ chức các container trong ứng dụng.
* **app:** (tên service):
* **Ý nghĩa**: Định nghĩa một service cụ thể, bạn có thể đặt tên tùy ý (ví dụ: app, web, db).
* **Vai trò**: Mỗi service tương ứng với một container.
* **build:**:
* **Ý nghĩa**: Chỉ định cách xây dựng image cho service này.
  + context: .: Thư mục chứa Dockerfile (ở đây là thư mục hiện tại).
  + dockerfile: Dockerfile: Tên file dùng để build image.
* **Vai trò**: Nói với Docker Compose rằng image sẽ được build từ Dockerfile thay vì kéo từ Docker Hub.
* **container\_name: node-app**:
* **Ý nghĩa**: Đặt tên cụ thể cho container là node-app thay vì tên ngẫu nhiên do Docker tạo.
* **Vai trò**: Dễ dàng quản lý và tham chiếu container.
* **ports: - "3000:3000"**:
* **Ý nghĩa**: Ánh xạ cổng 3000 trên host tới cổng 3000 trong container.
  + Số bên trái (3000) là cổng trên máy host.
  + Số bên phải (3000) là cổng trong container (được khai báo bởi EXPOSE trong Dockerfile).
* **Vai trò**: Cho phép truy cập ứng dụng từ bên ngoài container (ví dụ: http://localhost:3000).
* **restart: unless-stopped**:
* **Ý nghĩa**: Container sẽ tự động khởi động lại nếu crash, trừ khi bạn dừng nó thủ công bằng docker stop hoặc docker-compose down.
* **Vai trò**: Đảm bảo ứng dụng luôn chạy ổn định.

Run docker-compose và kết quả:

