BÀI TẬP CONTAINER & DOCKER

1. Docker Image & Container:

Bài Tập 1: Chạy Container Nginx từ Docker Hub

Mục tiêu: Chạy container Nginx từ image có sẵn trên Docker Hub và truy cập vào dịch vụ Nginx qua trình duyệt.

Yêu cầu:

- Kéo image Nginx từ Docker Hub về máy.
- Chạy container Nginx và ánh xạ cổng 80 của container với cổng 8080 trên máy host.
- Kiểm tra dịch vụ Nginx bằng cách truy cập http://localhost:8080.

Hướng dẫn:

- Kéo image Nginx từ Docker Hub: docker pull nginx
- Chạy container: docker run -d -p 8080:80 --name mynginx nginx
- Kiểm tra Nginx bằng lệnh curl

Bài Tập 2: Chạy Container Redis từ Docker Hub

Mục tiêu: Chạy container Redis từ Docker Hub và kết nối với Redis CLI để thực hiện các lệnh cơ bản.

Yêu cầu:

- Kéo image Redis từ Docker Hub.
- Chạy container Redis và ánh xạ cổng 6379 trên container với máy host.
- Kết nối vào Redis CLI bên trong container để thực hiện các lệnh cơ bản như PING, SET, GET.

Hướng dẫn:

- Kéo image Redis từ Docker Hub: docker pull redis
- Chạy container Redis: docker run -d -p 6379:6379 --name myredis redis
- Kết nối vào Redis CLI: docker exec -it myredis redis-cli
- Thực hiện lệnh trong Redis CLI:

```
PING
SET mykey "Docker"
GET mykey
```

Bài Tập 3: Chạy Container MySQL từ Docker Hub

Mục tiêu: chạy container MySQL từ Docker Hub, thiết lập mật khẩu root và kết nối vào MySQL từ dòng lệnh.

Yêu cầu:

- Kéo image MySQL từ Docker Hub.
- Chạy container MySQL và thiết lập mật khẩu root.
- Kết nối vào MySQL CLI bên trong container và tạo một cơ sở dữ liệu mới.

Hướng dẫn:

- Kéo image MySQL từ Docker Hub: docker pull mysql
- Chạy container MySQL với mật khẩu root: docker run -d -p 3306:3306 --name mymysql -e MYSQL_ROOT_PASSWORD=my-secret-pw mysql
- Kết nối vào MySQL CLI: docker exec -it mymysql mysql -u root -p
- Tạo cơ sở dữ liệu mới:

```
CREATE DATABASE testdb;
SHOW DATABASES;
```

Bài Tập 4: Chạy Container MongoDB từ Docker Hub

Mục tiêu: chạy container MongoDB từ Docker Hub và kết nối vào Mongo Shell để thực hiện các thao tác cơ bản.

Yêu cầu:

- Kéo image MongoDB từ Docker Hub.
- Chạy container MongoDB và ánh xạ cổng 27017 trên host.
- Kết nối vào Mongo Shell bên trong container và tạo một cơ sở dữ liệu mới.

Hướng dẫn:

- Kéo image MongoDB từ Docker Hub: docker pull mongo

- Chạy container MongoDB: docker run -d -p 27017:27017 --name mymongo mongo
- Kết nối vào Mongo Shell: docker exec -it mymongo mongo
- Thực hiện lệnh trong Mongo Shell:

```
use mydb
db.test.insert({name: "Docker"})
db.test.find()
```

Bài Tập 5: Xóa Docker Image Bằng Tên (Name) Hoặc ID Mục tiêu:

Xóa một Docker Image cụ thể bằng cách sử dụng tên image hoặc ID của image.

Yêu cầu:

- Liệt kê các Docker Images đã sử dụng ở bài tập 1 và 2.
- Xóa image đó bằng lệnh Docker.

Hướng dẫn:

- Liệt kê tất cả các Docker Images: docker images
- Xóa image bằng tên: docker rmi <image_name>
- Stop container đang sử dụng image đó nếu có thông báo lỗi: docker stop <container_name>

Bài Tập 6: Xóa Docker Image Không Sử Dụng (Dangling Images)

Mục tiêu: xóa các Docker Images "dangling" (image không có tag hoặc không được container nào sử dụng), giúp giải phóng dung lượng lưu trữ.

Yêu cầu:

- Stop các container ở bài tập 3 và 4
- Liệt kê tất cả các dangling images.
- Sử dụng lệnh Docker để xóa tất cả các dangling images.

Hướng dẫn:

- Liệt kê các dangling images: docker images -f dangling=true
- Xóa tất cả các dangling images: docker image prune

2. Dockerfile:

Bài Tập 1: Tạo Dockerfile Đơn Giản Cho Ứng Dụng Node.js

Mục tiêu: Tạo Dockerfile đơn giản cho một ứng dụng **Node.js** và chạy container từ Docker Image được tạo.

Yêu cầu:

- Tạo một Dockerfile để triển khai ứng dụng Node.js đơn giản.
- Sử dụng base image **node:14**.
- Cài đặt các phụ thuộc từ tệp **package.json**.
- Khai báo cổng và lệnh để chạy ứng dụng.
- Xây dựng Docker Image và chạy container từ image đó.

Hướng dẫn:

- Tạo tệp Dockerfile:

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

- Xây dựng image: docker build -t mynodeapp .
- Chạy container: docker run -d -p 3000:3000 mynodeapp

Bài Tập 2: Tạo Dockerfile Cho Ứng Dụng Python Flask

Mục tiêu: Tạo Dockerfile để triển khai một ứng dụng Python Flask đơn giản.

Yêu cầu:

- Tạo Dockerfile cho ứng dụng Flask sử dụng base image python:3.8.
- Cài đặt các thư viện cần thiết từ requirements.txt.
- Khai báo cổng và chạy ứng dụng bằng Flask.
- Xây dựng Docker Image và kiểm tra container.

Hướng dẫn:

- Tạo tệp Dockerfile:

```
FROM python:3.8
WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python", "app.py"]
```

- Xây dựng image: docker build -t myflaskapp .
- Chạy container: docker run -d -p 5000:5000 myflaskapp

Bài Tập 3: Tạo Dockerfile Với Nginx Làm Reverse Proxy

Mục tiêu: Các bạn sẽ học cách tạo Dockerfile để sử dụng Nginx làm reverse proxy cho một ứng dụng tĩnh.

Yêu cầu:

- Sử dụng base image nginx:alpine.
- Tạo Dockerfile để cấu hình Nginx phục vụ một ứng dụng tĩnh (ví dụ: trang HTML đơn giản).
- Tạo file index.html
- Xây dựng và kiểm tra Docker Image.

Hướng dẫn:

Tạo tệp Dockerfile:

```
FROM nginx:alpine
COPY ./index.html /usr/share/nginx/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Tạo file index.html với nội dung: “Chao mung den voi FIT@HCMUS”
- Xây dựng image: docker build -t mynginxapp .
- Chạy container: docker run -d -p 8080:80 mynginxapp

Bài Tập 4: Tạo Dockerfile Với .dockerignore Để Tối Ưu Hóa Kích Thước Image

Mục tiêu: Tạo Dockerfile và sử dụng tệp .dockerignore để loại bỏ các thư mục và tệp không cần thiết, tối ưu kích thước Docker Image.

Yêu cầu:

- Tạo một ứng dụng Node.js và Dockerfile để triển khai.
- Sử dụng .dockerignore để loại trừ các thư mục không cần thiết như
- node_modules và các tệp cấu hình cục bộ.
- Xây dựng Docker Image và kiểm tra kích thước của image.

Hướng dẫn:

- Tạo tệp Dockerfile như bài tập 1.
- Tạo tệp .dockerignore:

```
node_modules
.git
.env
```

- Xây dựng image: docker build -t myoptimizednodeapp .
- Kiểm tra kích thước image: docker images

Bài Tập 5: Tạo Dockerfile Với Multi-Stage Build

Mục tiêu: Sử dụng **multi-stage** build để tạo Docker Image tối ưu cho một ứng dụng React hoặc Vue.js.

Yêu cầu:

- Tạo Dockerfile với **multi-stage build** cho ứng dụng React hoặc Vue.js.
- Trong giai đoạn đầu tiên, xây dựng ứng dụng từ mã nguồn.
- Trong giai đoạn thứ hai, sử dụng Nginx để phục vụ ứng dụng tĩnh.
- Xây dựng và chạy Docker Image.

Hướng dẫn:

- Tạo tệp Dockerfile:

```
# Stage 1: Build the React app
FROM node:14 AS builder
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Stage 2: Serve the app with Nginx
FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Xây dựng image: docker build -t myreactapp .
- Chạy container: docker run -d -p 8080:80 myreactapp

3. Docker registry:

Bài Tập 1: Cài Đặt và Chạy Docker Private Registry

Mục tiêu: cài đặt và khởi chạy một **Docker Private Registry** trên máy chủ cục bộ.

Yêu cầu:

- Sử dụng Docker để chạy một Private Docker Registry trên máy cục bộ.
- Chạy Docker Registry trên cổng 5000.
- Kiểm tra xem Docker Registry có hoạt động bằng cách truy cập địa chỉ http://localhost:5000/v2/_catalog.

Hướng dẫn:

- Chạy Docker Registry: docker run -d -p 5000:5000 --name myregistry registry:2
- Kiểm tra hoạt động của Docker Registry:
- Mở trình duyệt và truy cập http://localhost:5000/v2/_catalog. Nếu Docker Registry đang chạy, bạn sẽ nhận được kết quả là một danh sách repository rỗng.

Bài Tập 2: Tạo và Đẩy Docker Image Lên Docker Private Registry

Mục tiêu: Xây dựng một Docker Image và đẩy nó lên Docker Private Registry.

Yêu cầu:

- Tạo Dockerfile cho một ứng dụng Node.js đơn giản.
- Xây dựng Docker Image từ Dockerfile.
- Gắn thẻ Docker Image để đẩy lên Private Docker Registry.
- Đẩy Docker Image lên registry.

Hướng dẫn:

- Tạo tệp Dockerfile:

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

- Xây dựng Docker Image: docker build -t mynodeapp .
- Gắn thẻ Image cho Private Registry: docker tag mynodeapp localhost:5000/mynodeapp
- Đẩy Image lên Docker Registry: docker push localhost:5000/mynodeapp

Bài Tập 3: Kéo Docker Image Từ Docker Private Registry

Mục tiêu: Các bạn sẽ học cách kéo (pull) Docker Image từ một Docker Private Registry đã được cài đặt.

Yêu cầu:

- Sử dụng Docker Private Registry đã cài đặt ở bài tập trước.
- Kéo Docker Image từ Private Registry về máy cục bộ.
- Chạy container từ image vừa kéo về.

Hướng dẫn:

- Kéo image từ Private Registry: docker pull localhost:5000/mynodeapp
- Chạy container từ image vừa kéo về: docker run -d -p 3000:3000 localhost:5000/mynodeapp

Bài Tập 4: Thiết Lập Docker Registry Với Xác Thực

Mục tiêu: Thiết lập Docker Private Registry với xác thực cơ bản để bảo vệ truy cập vào Docker Registry.

Yêu cầu:

- Cài đặt Docker Registry với xác thực bằng username và password.
- Tạo tệp htpasswd để lưu thông tin người dùng và mật khẩu.
- Cấu hình Docker Registry để sử dụng tệp htpasswd.
- Đăng nhập vào Docker Registry và đẩy image sau khi đã bật xác thực.

Hướng dẫn:

- Tạo tệp htpasswd:

```
sudo apt-get install apache2-utils
htpasswd -Bc /auth/htpasswd myuser
```

- Chạy Docker Registry với xác thực:

```
docker run -d \
  -p 5000:5000 \
  --name registry \
  -v /auth:/auth \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
  -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
  -e "REGISTRY_STORAGE_DELETE_ENABLED=true" \
  registry:2
```

- Đăng nhập vào Docker Registry: docker login localhost:5000
- Sau khi đăng nhập thành công, đẩy image lên Private Registry: docker push localhost:5000/mynodeapp

Bài Tập 5: Xóa Tag Hoặc Manifest Của Docker Image Trong Private Registry

Mục tiêu:Xóa một Docker Image khỏi **Docker Private Registry** bằng cách sử dụng API để xóa manifest hoặc tag.

Yêu cầu:

- Tạo và đẩy một Docker Image lên Docker Private Registry.
- Xóa tag hoặc manifest của Docker Image bằng cách sử dụng API của Docker Registry.

Hướng dẫn:

- Lấy danh sách tag của image: curl http://localhost:5000/v2/mynodeapp/tags/list
- **Lấy manifest digest:** Để xóa một image, trước tiên cần lấy **digest** (hàm băm) của manifest bằng cách sử dụng API: curl -I -H "Accept: application/vnd.docker.distribution.manifest.v2+json" http://localhost:5000/v2/mynodeapp/manifests/1.0

  Kết quả sẽ hiển thị digest trong header:

  ```
  Docker-Content-Digest: sha256:abcd1234...
  ```

- Xóa manifest hoặc tag của image:Sau khi có digest, có thể xóa image bằng lệnh sau: curl -u myuser:mypassword -X DELETE http://localhost:5000/v2/mynodeapp/manifests/sha256:abcd1234...
- Kiểm tra lại tag của image: curl -u myuser:mypassword http://localhost:5000/v2/mynodeapp/tags/list

4. Docker Hub:

Bài Tập 1: Đẩy Docker Image Lên Docker Hub

Mục tiêu: Đẩy Docker Image từ máy cục bộ lên Docker Hub, quản lý image trên Docker Hub.

Yêu cầu:

- Tạo tài khoản Docker Hub
- Đẩy Docker Image bài tập 3 phần dockerfile từ máy cục bộ lên Docker Hub.
- Kiểm tra image trên Docker Hub.

Hướng dẫn:

- Tạo tài khoản: hub.docker.com
- Login docker: docker login
- Gắn thẻ image để chuẩn bị đẩy lên Docker Hub: docker tag <image_name> <dockerhub_username>/<image_name>:1.0
- Đẩy Docker Image lên Docker Hub: docker push <dockerhub_username>/myapp:1.0
- Truy cập Docker Hub và tìm image vừa đẩy trong repository.

Bài Tập 2: Cập Nhật Docker Image và Đẩy Phiên Bản Mới Lên Docker Hub

Mục tiêu: cập nhật Docker Image và đẩy phiên bản mới lên Docker Hub với tag khác.

Yêu cầu:

- Sửa lại nội dung file index.html bài tập 2 phần dockerfile thành “Chao mung ban den voi FIT@HCMUS – khoa hoc DevOps co ban” sau đó thực hiện build lại docker image và gắn thẻ phiên bản mới.
- Đẩy phiên bản mới lên Docker Hub với tag khác.

Hướng dẫn:

- Cập nhật Dockerfile và nội dung file index.html
- Xây dựng lại Docker Image
- Gắn thẻ image cho phiên bản mới: docker tag <image_name> <dockerhub_username>/<image_name>:2.0
- Đẩy phiên bản mới lên Docker Hub: docker push <dockerhub_username>/myapp:2.0
- Kiểm tra phiên bản mới trên Docker Hub.

5. Docker Network:

Bài Tập 1: Sử Dụng Mạng Mặc Định (Bridge Network)

Mục tiêu: Sử dụng bridge network, mạng mặc định của Docker khi chạy container để container có thể giao tiếp với máy host.

Yêu cầu:

- Chạy một container Nginx và ánh xạ cổng từ container ra máy host.
- Kiểm tra xem có thể truy cập Nginx từ máy host bằng địa chỉ localhost.

Hướng dẫn:

- Chạy container Nginx và ánh xạ cổng: docker run -d --name nginxcontainer -p 8080:80 nginx
- Truy cập http://localhost:8080 để xem Nginx hoạt động.
- Kiểm tra địa chỉ IP của container: docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' nginxcontainer

Bài Tập 2: Tạo Mạng Bridge Tùy Chỉnh Và Kết Nối Nhiều Container

Mục tiêu: tạo một **bridge network** tùy chỉnh và kết nối nhiều container vào mạng này để chúng có thể giao tiếp với nhau.

Yêu cầu:

- Tạo một mạng bridge tùy chỉnh.
- Chạy hai container (ví dụ: Nginx và MySQL) và kết nối chúng vào mạng này.
- Kiểm tra khả năng giao tiếp giữa các container qua tên container.

Hướng dẫn:

- Tạo mạng bridge tùy chỉnh: docker network create mybridgenetwork Chạy container Nginx và MySQL trong mạng tùy chỉnh:

```
docker run -d --name mynginx --network mybridgenetwork nginx
docker run -d --name mymysql --network mybridgenetwork -e MYSQL_ROOT_PASSWORD=root mysql
```

- Từ Nginx, kiểm tra khả năng ping MySQL bằng tên container: ping mymysql

Bài Tập 3: Sử Dụng Mạng Host Network

Mục tiêu: Sử dụng **host network**, cho phép container chia sẻ mạng với máy host.

Yêu cầu:

- Chạy container Nginx sử dụng host network.
- Kiểm tra xem container có sử dụng địa chỉ IP của máy host hay không.
- Truy cập Nginx từ máy host mà không cần ánh xạ cổng.

Hướng dẫn:

- Chạy container Nginx với host network: docker run -d --name nginxhost -- network host nginx
- Kiểm tra địa chỉ IP của container:
- Truy cập http://localhost không cần ánh xạ cổng.

Bài Tập 4: Quản Lý Mạng Docker (Liệt Kê, Xóa)

Mục tiêu: Các bạn sẽ học cách quản lý các mạng Docker bằng cách liệt kê, kiểm tra thông tin chi tiết và xóa mạng không sử dụng.

Yêu cầu:

- Xóa container đã tạo ở bài tập 1 và 2
- Liệt kê tất cả các mạng Docker hiện có.

- Kiểm tra thông tin chi tiết của một mạng cụ thể.
- Xóa một mạng không còn được sử dụng.

Hướng dẫn:

- Liệt kê tất cả các mạng Docker: docker network ls
- Kiểm tra thông tin chi tiết của mạng: docker network inspect mybridgenetwork
- Xóa mạng không còn sử dụng: docker network rm mybridgenetwork

6. Docker Volume:

Bài Tập 1: Tạo và Sử Dụng Docker Volume

Mục tiêu: Tạo Docker Volume và gắn kết nó với container.

Yêu cầu:

- Tạo một Docker Volume.
- Chạy container với volume vừa tạo và gắn kết nó với một thư mục trong container.
- Kiểm tra tính bền vững của volume bằng cách lưu dữ liệu vào container.

Hướng dẫn:

- Tạo Docker Volume: docker volume create myvolume
- Gắn kết volume myvolume vào thư mục /data trong container: docker run -d -- name mycontainer -v myvolume:/data busybox
- Truy cập vào container và lưu dữ liệu vào thư mục /data:

```
echo "Hello from Docker Volume" > /data/test.txt
exit
```

- Kiểm tra dữ liệu: Xóa container và khởi động lại một container khác để kiểm tra xem dữ liệu vẫn còn.

```
docker rm -f mycontainer
docker run -it --name newcontainer -v myvolume:/data busybox sh
cat /data/test.txt
```

Bài Tập 3: Gắn Kết Thư Mục Của Máy Chủ (Bind Mount) Với Container

Mục tiêu: gắn kết một thư mục từ máy chủ vào container bằng cách sử dụng bind mount.

Yêu cầu:

- Gắn kết một thư mục từ hệ thống host (máy chủ) vào container.
- Thực hiện các thay đổi trong container và kiểm tra xem thư mục trên máy chủ có được cập nhật hay không.

Hướng dẫn:

- Tạo thư mục trên hệ thống host: mkdir /tmp/mybindmount
- Chạy container và gắn kết bind mount: docker run -d --name bindcontainer -v
- /tmp/mybindmount:/data busybox
- Tạo tệp trong container:

```
echo "This is a bind mount example" > /data/bindfile.txt
exit
```

- Truy cập vào /tmp/mybindmount trên hệ thống host và kiểm tra tệp bindfile.txt có tồn tại không.

Bài Tập 4: Chia Sẻ Volume Giữa Nhiều Container

Mục tiêu: Chia sẻ một Docker Volume giữa nhiều container để các container có thể truy cập chung một dữ liệu.

Yêu cầu:

- Tạo một Docker Volume và gắn kết nó với nhiều container khác nhau.
- Lưu dữ liệu từ một container và truy cập dữ liệu đó từ container khác.

Hướng dẫn:

- Tạo Docker Volume: docker volume create sharedvolume
- Chạy container đầu tiên và lưu dữ liệu vào volume:

```
docker run -d --name container1 -v sharedvolume:/data busybox
docker exec -it container1 sh
echo "Shared data" > /data/sharedfile.txt
exit
```

- Chạy container thứ hai và kiểm tra dữ liệu:

```
docker run -it --name container2 -v sharedvolume:/data busybox sh
cat /data/sharedfile.txt
```

Bài Tập 5: Quản Lý Docker Volumes – Liệt Kê và Xóa Volumes Không Sử Dụng

Mục tiêu: Quản lý Docker Volumes, bao gồm việc liệt kê và xóa các volume không còn sử dụng để tiết kiệm dung lượng lưu trữ.

Yêu cầu:

- Liệt kê tất cả các Docker Volumes hiện có trên hệ thống.
- Xóa các Docker Volumes không còn được container nào sử dụng.

Hướng dẫn:

- Liệt kê tất cả các Docker Volumes: docker volume ls
- Xóa các Docker Volumes không sử dụng: docker volume prune
- Xóa Docker Volume cụ thể (chỉ xóa được nếu volume không còn sử dụng bởi container nào): docker volume rm myvolume

7. Docker composer:

Bài Tập 1: Triển Khai Ứng Dụng Web Nginx Với Docker Compose

Mục tiêu: Các bạn sẽ học cách tạo tệp docker-compose.yml để triển khai dịch vụ web đơn giản sử dụng Nginx.

Yêu cầu:

- Tạo thư mục bt1-docker-compose sau đó đặt tất cả các file và thư mục liên quan đến bài tập này vào thư mục này.
- Tạo một dịch vụ Nginx với Docker Compose.
- Cấu hình Nginx để lắng nghe trên cổng 8087 của máy chủ cục bộ.
- Kiểm tra khả năng truy cập Nginx từ trình duyệt hoặc dòng lệnh hoặc trình duyệt.

Hướng dẫn:

- Truy cập vào thư mục bt1-docker-compose
- Tạo tệp docker-compose.yml với nội dung sau:

```yaml
services:
  web:
    image: nginx:latest
    ports:
      - "8087:80"
```

- Khởi động Docker Compose:
- Kiểm tra kết quả: Mở trình duyệt và truy cập vào http://localhost:8087.

Bài Tập 2: Triển Khai Ứng Dụng Node.js Và MongoDB

Mục tiêu: Các bạn sẽ biết cách cấu hình Docker Compose để triển khai một ứng dụng Node.js đơn giản kết nối với cơ sở dữ liệu MongoDB.

Yêu cầu:

- Tạo thư mục bt2-docker-compose sau đó đặt tất cả các file và thư mục liên quan đến bài tập này vào thư mục này(các tệp Dockerfile, package.json, app.js và docker-compose.yml)
- Tạo hai dịch vụ Node.js và MongoDB trong tệp docker-compose.yml.
- Node.js có thể kết nối với MongoDB thông qua network tên là node-app.
- Kiểm tra kết nối giữa Node.js và MongoDB bằng cách gửi yêu cầu đến ứng dụng.

Hướng dẫn:

- Truy cập vào thư mục bt2-docker-compose.
- Tạo Dockerfile cho Node.js:

```dockerfile
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

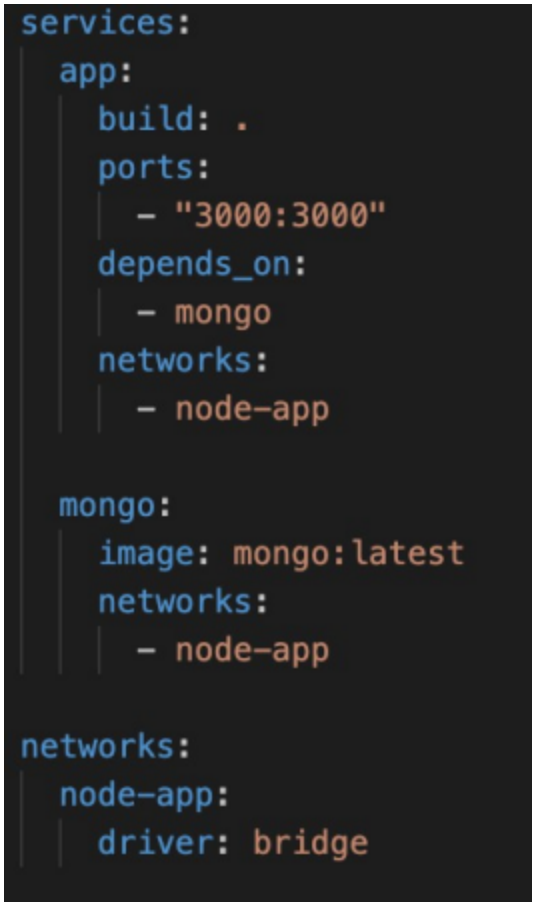
- Tạo tệp package.json:

```
{
  "name": "nodeapp",
  "version": "1.0.0",
  "main": "app.js",
  "dependencies": {
    "express": "^4.17.1",
    "mongodb": "^3.6.0"
  },
  "scripts": {
    "start": "node app.js"
  }
}
```

- Tạo tệp app.js:

```
const express = require('express');
const MongoClient = require('mongodb').MongoClient;
const app = express();

const url = 'mongodb://mongo:27017';
const dbName = 'testdb';

MongoClient.connect(url, (err, client) => {
  if (err) throw err;
  console.log("ke noi thanh cong den MongoDB");
  const db = client.db(dbName);

  app.get('/', (req, res) => {
    res.send('Hello tu Node.js va MongoDB!');
  });

  app.listen(3000, () => {
    console.log('Ung dung Node.js dang chay tren cong 3000');
  });
});
```

- Tạo tệp docker-compose.yml:



```yaml
services:
  app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      - mongo
    networks:
      - node-app

  mongo:
    image: mongo:latest
    networks:
      - node-app

networks:
  node-app:
    driver: bridge
```

- Khởi động Docker Compose: docker-compose up -d
- Kiểm tra kết quả: Truy cập http://localhost:3000 để kiểm tra kết nối giữa Node.js và MongoDB.

Bài Tập 3: Triển Khai Ứng Dụng WordPress và MySQL

Mục tiêu: Các bạn sẽ biết cách triển khai một ứng dụng **WordPress** kết hợp với cơ sở dữ liệu **MySQL** bằng Docker Compose.

Yêu cầu:

- Tạo thư mục bt3-docker-compose sau đó đặt tất cả các file và thư mục liên quan đến bài tập này vào thư mục này.
- Tạo tệp docker-compose.yml để thiết lập hai dịch vụ: WordPress và MySQL.
- Tạo mạng wordpress-network để WordPress có thể kết nối với MySQL với mạng wordpress-network
- Mở port 8090 trên wordpress để người dùng có thể truy cập.
- Kiểm tra trang WordPress cài đặt thành công trên trình duyệt.

Hướng dẫn:

- Truy cập vào thư mục bt3-docker-compose
- Tạo tệp docker-compose.yml với cấu hình cho WordPress và MySQL:

```yaml
services:
  wordpress:
    image: wordpress:latest
    ports:
      - "8090:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: rootpassword
      WORDPRESS_DB_NAME: wordpress
    depends_on:
      - db
    networks:
      - worpdress-network

  db:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: wordpress
    networks:
      - worpdress-network

networks:
  mynetwork:
    driver: bridge
```

- Khởi động Docker Compose: docker-compose up -d
- Kiểm tra kết quả:
    - Mở trình duyệt và truy cập http://localhost:8090
    - Thực hiện các bước cài đặt WordPress trên giao diện trình duyệt.