WEEK 7 DAY 4

DEVELOPING AND DEPLOYING A USER PROFILE APP WITH DOCKER

This demo application is a basic user profile management app built with multiple technologies, including:

- **Frontend**: A simple index.html page styled with pure JavaScript and CSS.
- **Backend**: A Node.js server running with the Express framework.
- **Database**: MongoDB for storing user data.

The entire application is designed to run seamlessly in a Dockerized environment, enabling easy deployment and management of each component as a containerized service. The app's DockerHub repository can be found at: DockerHub Repo - Nana Node App.

Running the Application Using Docker

Step-by-Step Guide to Start the Application

Step 1: **Create a Docker Network**

To allow communication between the MongoDB and Mongo Express containers, create a Docker network:

```
docker network create mongo-network
```

Step 2: **Start MongoDB Container**

Run the MongoDB container with specific environment variables for the root username and password. This configuration sets up MongoDB on port 27017:

```
docker run -d -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin -e
MONGO_INITDB_ROOT_PASSWORD=password --name mongodb --network mongo-
network mongo:4.4.24
```

Step 3: **Start Mongo Express**

Mongo Express provides a web-based interface to interact with MongoDB. Start it on port 8081 and connect it to the previously created mongo-network:

```
docker run -d -p 8081:8081 -e ME_CONFIG_MONGODB_ADMINUSERNAME=admin -e
ME_CONFIG_MONGODB_ADMINPASSWORD=password --network mongo-network --
name mongo-express -e ME_CONFIG_MONGODB_SERVER=mongodb mongo-express
```

Note: Creating a custom Docker network (mongo-network) is optional. If you prefer to use Docker's default network, simply omit the --network flag in the docker run commands for MongoDB and Mongo Express.

Step 4: **Access Mongo Express in the Browser**

Once Mongo Express is running, open it in a web browser to manage MongoDB visually:

http://localhost:8081

Step 5: **Create Database and Collection**

Within the Mongo Express UI, create a new database named user-account and a collection named users to store user profile data.

Step 6: **Start the Node.js Application Locally**

Navigate to the application directory (app folder) and install the required dependencies. Start the Node.js server using the following commands:

```
cd app

npm install

node server.js
```

Step 7: **Access the Node.js Application**

Open a web browser and access the application's UI on http://localhost:3000.

Running the Application Using Docker Compose

Docker Compose allows us to define and run multiple containers with a single command. The following steps outline how to use Docker Compose to start the application:

Step-by-Step Guide Using Docker Compose

Step 1: **Start MongoDB and Mongo Express Using Docker Compose**

Run Docker Compose with the docker-compose.yaml file to start both the MongoDB and Mongo Express containers:

```
docker-compose -f docker-compose.yaml up
```

You can access Mongo Express from your browser at http://localhost:8080.

Step 2: **Set Up Database and Collection in Mongo Express**

Once Mongo Express is accessible, create a new database named my-db and a collection named users within my-db.

Step 3: **Start the Node.js Server**

Navigate to the app directory, install the necessary Node.js dependencies, and start the server:

```
cd app

npm install

node server.js
```

Step 4: **Access the Application**

Open a web browser and access the application UI at http://localhost:3000.

Building a Docker Image for the Application

To package the application as a Docker image, use the following command:

```
docker build -t node-app:1.0 .
```

The dot (.) at the end of the command specifies the location of the Dockerfile, which is expected to be in the current directory. This command builds the Docker image with the tag node-app:1.0, ready to be deployed as a containerized service.