**Dockerize Neo4j and GraphQL**

# Download Docker

To begin with the process, we need to first download Docker Desktop into our environment. Docker Desktop provides an easy-to-use graphical interface and all the necessary components to build, run, and manage Docker containers on our system. It can be downloaded from the official website ([here](https://www.docker.com/products/docker-desktop/)).

Upon downloading, we can verify it by typing the command below in our terminal or accessing the Docker Desktop application.

A black background with white text

Description automatically generated

*Figure1: Verification of Docker version*

A screenshot of a computer

Description automatically generated

*Figure2: Docker Desktop User Interface*

# Create Docker File

## Pre-requisite files:

* node modules folder
* package\*.json
* GraphQL application (index.js)

## Repository Structure

A diagram of a computer program

Description automatically generated

*Figure3: Project File Structure*

## Docker File

The Docker File is created under the root directory and the main purpose of this file is to build a Docker image for the GraphQL application and then running it in a container. This file should include the necessary ependencies and command to start the application. Below is the explanation of the Docker File.

A screenshot of a computer program

Description automatically generated

*Figure4: Docker File Code*

Line1: We are using node base image from docker hub. In this example we are using version 14 of it.

Line3: Setting our working directory in the container.

Line5: Copying the package.json & package-lock.json file to container.

Line7: Installing all related dependencies.

Line9: Expose the port that our GraphQL application will be using.

Line11: Command to start our GraphQL server.

*\*We are currently not copying the application files over as we will bind mount our directory from host machine to the Docker container later. (In short, we can understand volume as a mechanism to persist data generated or used by Docker containers.)*

# Create docker-compose.yml file

A screenshot of a computer program

Description automatically generated

*Figure5: YAML file*

The dokcer-compose.yml file created under the root repository.

Understanding that the GraphQL application is dependent on the Neo4j Database, we need to create another container to hosts the database. This can be done by defining a service section in the YAML file to define the individual services (containers) that make up our application. Each service will be consisting of its own specific configurations too.

The breakdown of the YAML files is as below:

Line 4: Creating the base image for our service (which is Neo4j in our application) from Docker Hub.

Line 5: Defining the name for our Neo4j database container.

Line 7-8: Mapping the external ports to internal port (E.g., mapping external 7484 to internal 7474).

Line 10: Setting our environmental variables (username & password).

Line 12-13: Host mount local directory folders to container files.

Line 18: Generate GraphQL application based on the Docker File we created in the previous section.

Line 19: Defining the name for the GraphQL application container.

Line 21: Mapping the external ports to internal port.

Line 23: To specify the order in which services should be started when running multiple containers.

Line 25: Host mount root directory to application file.

# Building Docker Image and Container

Having the docker-compose.yml file, we can now create the containers to start our application.

Option 1:

Open the terminal or command prompt from the root directory and run the following to create docker images for each service.

A screenshot of a computer program

Description automatically generated

## Option 2:

Open Docker Desktop and start our containers.

A screenshot of a computer

Description automatically generated

# Running the Application

The application should be now accessible at the external port that we mapped to.

Neo4j Database: <http://localhost:7484>

A screenshot of a computer

Description automatically generated

GraphQL Application: <http://localhost:4010/>

A screenshot of a computer

Description automatically generated