Node-Mongo-Nginx-Docker-Project

Objectives:

The aim of this project was to create a simple NodeJS application using express and pug, which would prompt the user to enter their first name, surname and location into a mongo dB database. For this project, we were to create three docker containers running on the same network, one for the app, one for the database and the last one to run a Nginx reverse-proxy. Finally, the last objective was to allow the webpage for the application to be running on https for security.

Solution:

To achieve all the objectives of this project, I broke it down into stages. The first step I took was developing the node application itself, and then connecting the application to a mongo database. The application had the ability for the user to input data and save it to the database, and another function where the user was able to view the registrations that had been entered so far. Once that was fully functional, I added the Nginx reverse-proxy to direct requests to the app server. After these were running locally, it was time to create them inside docker containers. To do this, I used Dockerfiles to build docker images of the app, Nginx and mongo, which would be used to run the individual containers. In order to create and maintain the three different containers, I used a docker-compose.yml file. Inside the docker-compose file, I used volumes to provide links to the local machine for each of the three containers. For the app container, volumes were used to mount the app source code into the container so that the application could run properly. For the mongo container, volumes were used to link a folder on my local machine to the data saved in the database. This was so that when the container was shut down and restarted, the registrations in the database would not be lost. And finally, I used volumes for the Nginx container in order to transfer my self-signed certificates, for the use of https, from my local machine into the container.

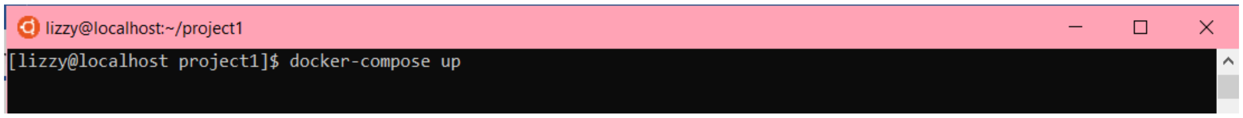
Challenges:

Throughout this project I faced several challenges. Firstly, I had never coded my own NodeJS application before, or used express and pug, so the whole process and syntax was new to me. However, I found that with some research I was soon able to slowly pick things up and understand how the process worked. By the end of the project I feel more confident in knowing a bit more understanding about NodeJS applications which should help me with future projects. The second challenge I faced in this project was using docker volumes to mount the code of my application into the docker container. Having researched docker volumes, the process itself seemed rather simple, however when I was testing my container it was failing at the start due to it not being able to find the code. After a fair while, I realised my error was that I was not using the complete path to my local directory where the app code was stored, so that when the container was created, it was looking in the incorrect folder for the application. Once rectified, the docker volume mounted correctly and the application ran smoothly. One final problem I faced was implementing self-signed certificates to allow for a https secure connection on my localhost. Having never generated SSL certificates before, or implemented their use, this was another new challenge I spent a fair amount of time investigating. After some research, I was able to generate, on my local machine, my own self-signed certificate and key pairing to be used for my localhost. Once created locally, I was able to mount these certificates into my Nginx docker container using a docker volume. Finally, I adjusted the reverse-proxy configuration file that my Nginx container was using so that it was listening on port 443 and had links to my certificate and key. With that implemented, I was then able to run the entire application on https://localhost.

Results:

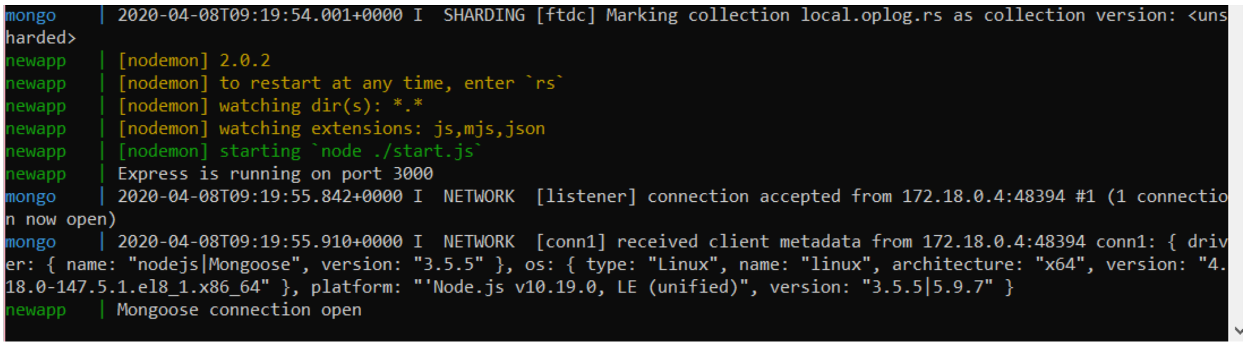
The outcome of this project can be illustrated below:

- To get the project running, I simply have to run the 'docker-compose up' command in my project folder.



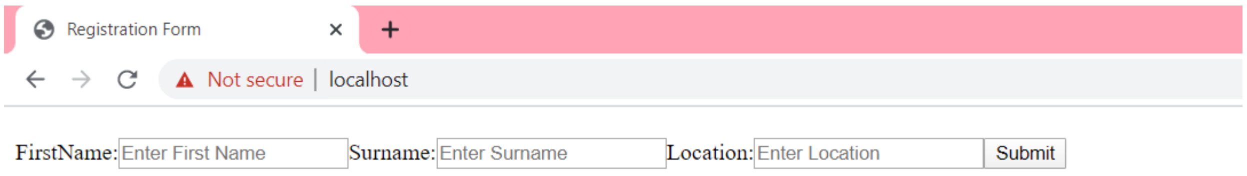
```
lizzy@localhost:~/project1
[lizzy@localhost project1]$ docker-compose up
```

- This will then inform me that my application is running and connected to the mongo database via mongoose.



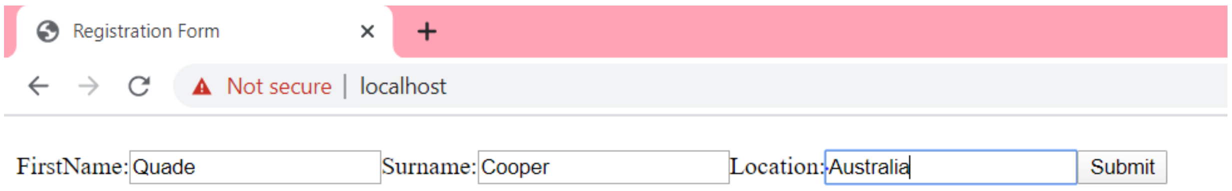
```
mongo     | 2020-04-08T09:19:54.001+0000 I  SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <uns
harded>
newapp    | [nodemon] 2.0.2
newapp    | [nodemon] to restart at any time, enter `rs`
newapp    | [nodemon] watching dir(s): *.*
newapp    | [nodemon] watching extensions: js,mjs,json
newapp    | [nodemon] starting `node ./start.js`
newapp    | Express is running on port 3000
mongo     | 2020-04-08T09:19:55.842+0000 I  NETWORK  [listener] connection accepted from 172.18.0.4:48394 #1 (1 connectio
n now open)
mongo     | 2020-04-08T09:19:55.910+0000 I  NETWORK  [conn1] received client metadata from 172.18.0.4:48394 conn1: { driv
er: { name: "nodejs|Mongoose", version: "3.5.5" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "4.
18.0-147.5.1.el8_1.x86_64" }, platform: "'Node.js v10.19.0, LE (unified)", version: "3.5.5|5.9.7" }
newapp    | Mongoose connection open
```

- To access the application, I then head to my web browser and type https://localhost

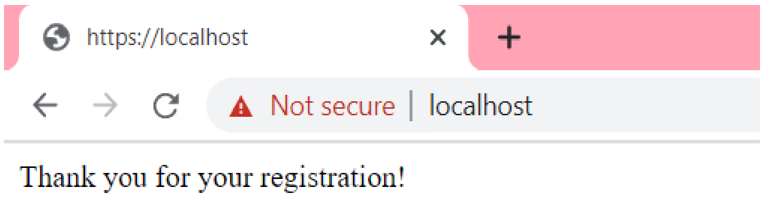


FirstName: Enter First Name Surname: Enter Surname Location: Enter Location Submit

- I am then able to enter my details, press submit, and it shall save them into the database.

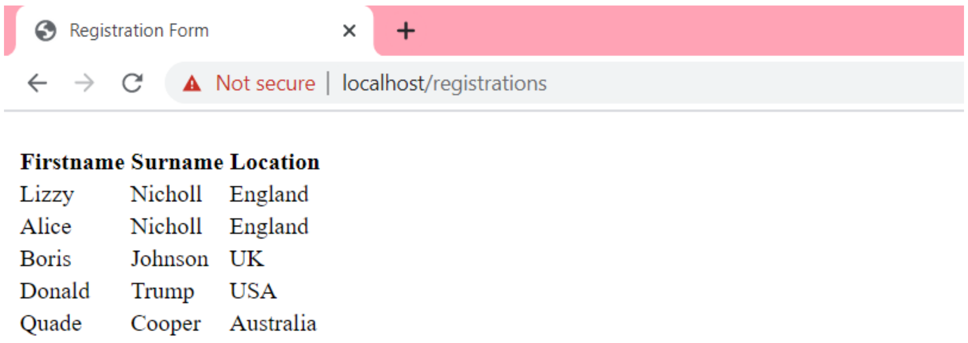


FirstName: Quade Surname: Cooper Location: Australia Submit



Thank you for your registration!

- And finally, to check my details have been entered, and to view the other registered members of the database, I navigate to https://localhost/registrations.



Firstname	Surname	Location
Lizzy	Nicholl	England
Alice	Nicholl	England
Boris	Johnson	UK
Donald	Trump	USA
Quade	Cooper	Australia