Step-by-Step Guide

1. **Setup Instructions**

First, I will install Docker if I don't have it on my system. I can download it from the official Docker Website https://www.docker.com/. Docker Compose usually comes with Docker Desktop, so I will make sure it's installed too.

Next, I will clone the repository to my local machine by running:

```
Cloning into 'FullStack-ToDo-List-Docker'...
remote: Enumerating objects: 279, done.
remote: Counting objects: 100% (279/279), done.
remote: Compressing objects: 100% (137/137), done.
remote: Total 279 (delta 113), reused 279 (delta 113), pack-reused 0 (from 0)
Receiving objects: 100% (279/279), 2.65 MiB | 64.00 KiB/s, done.
Resolving deltas: 100% (113/113), done.
edem@edem-ThinkPad-T450:~$ ls
Desktop    FullStack-ToDo-List-Docker  myenv                     Public
Documents  get-docker.sh               number-classification-api  Templates
Downloads  Music                       Pictures                  Videos
```

2. **Creating the Dockerfiles**

Each component of the application (Frontend, Backend, and Database) has its own **Dockerfile**, which defines how the container is built and configured.

Frontend Dockerfile:

Frontend > dockerfile > ...

```
1   FROM node:lts-alpine
2   ENV NODE_ENV=production
3   WORKDIR /usr/src/app
4   COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
5   RUN npm install --include dev
6   COPY . .
7   EXPOSE 5173
8   RUN chown -R node /usr/src/app
9   USER node
10  CMD ["npm", "run", "dev"]
11
```

- Uses a lightweight **Node.js Alpine** image.
- Copies the project files and installs dependencies.
- Runs the frontend application on port **5173**.

Backend Dockerfile:

Backend > dockerfile > ...

```
FROM node:lts-alpine
ENV NODE_ENV=production
WORKDIR /usr/src/app
COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
RUN npm install --production --silent && mv node_modules ../
COPY . .
EXPOSE 3000
RUN chown -R node /usr/src/app
USER node
CMD ["npm", "start"]
```

- Uses **Node.js Alpine** for efficiency.
- Installs only production dependencies to optimize the build.
- Runs the backend on port **3000**.

Database Dockerfile:

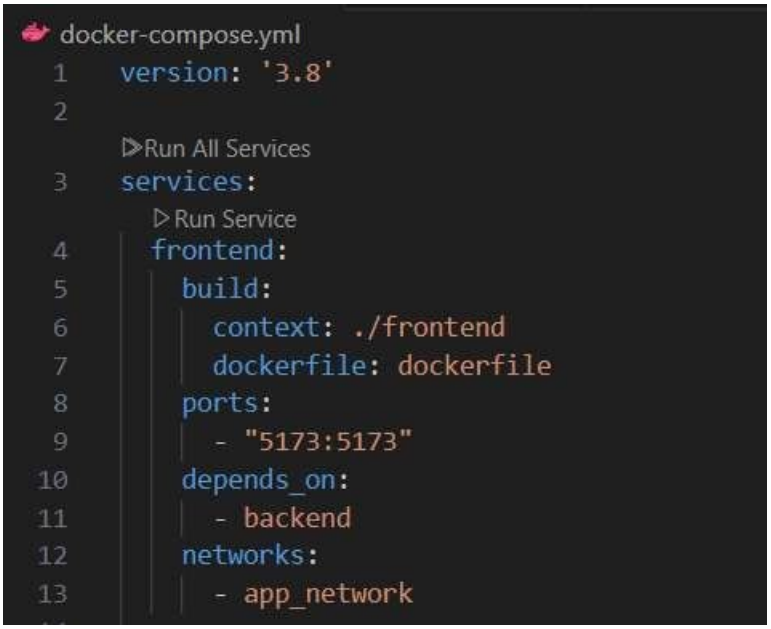
Backend > DB > dockerfile > ...

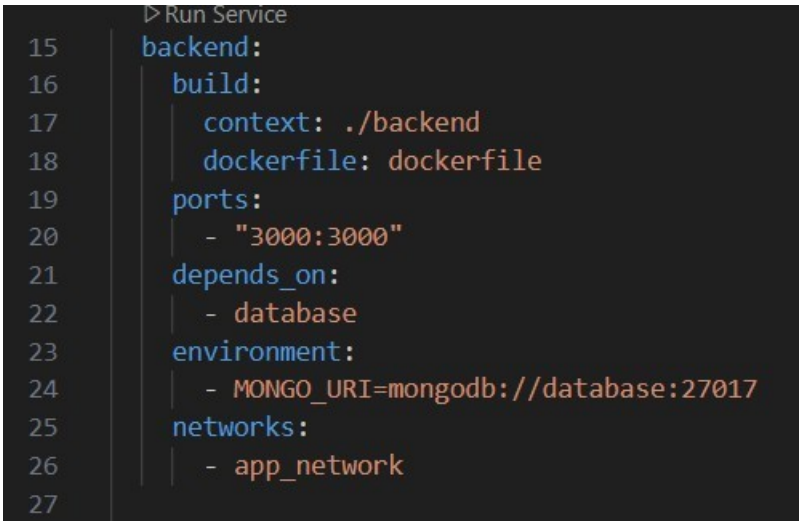
```
FROM mongo:latest

WORKDIR /data/db

EXPOSE 27017

CMD ["mongod"]
```

- Uses the **MongoDB** official image.
- Exposes **port 27017**, which is MongoDB's default port.
- Runs the **MongoDB daemon** (mongod).

3. Creating the Docker Compose file

The docker-compose.yml file defines how all three services (Frontend, Backend, and Database) work together.





Defines **three services**: frontend, backend, and database.

Networks:

- All services are connected using a custom **bridge network** (app_network).

Volumes:

- The database uses a persistent volume (mongo_data) to store data.

Dependencies:

- The **Frontend** depends on the **Backend**.
- The **Backend** depends on the **Database**.

4. **Containers**

Build the Containers:

Use Docker Compose to build the necessary containers for the frontend, backend, and

database:

```
edem@edem-ThinkPad-T450:~/FullStack-ToDo-List-Docker$ docker-compose up --build
Creating network "fullstack-todo-list-docker_app_network" with driver "bridge"
Creating volume "fullstack-todo-list-docker_mongo_data" with default driver
Building database
[+] Building 1156.5s (6/6) FINISHED                                docker:default
 => [internal] load build definition from dockerfile                         0.0s
 => => transferring dockerfile: 106B                                         0.0s
 => [internal] load metadata for docker.io/library/mongo:latest             13.0s
 => [internal] load .dockerignore                                            0.0s
 => => transferring context: 2B                                              0.0s
 => [1/2] FROM docker.io/library/mongo:latest@sha256:98028cf281bb5d49a  1143.3s
 => => resolve docker.io/library/mongo:latest@sha256:98028cf281bb5d49ace5  0.0s
 => => sha256:883cc53df4d14401922df6eb2b6a16b4ac461afd04a 2.48kB / 2.48kB  0.0s
 => => sha256:14bce8bf75c7ffb890108dea16aae00d9f1667b86f1 7.86kB / 7.86kB  0.0s
 => => sha256:d9d352c11bbd3880007953ed6eec1cbace76898 29.72MB / 29.72MB  153.6s
 => => sha256:0a4282d2a9c9631b43bac0401761e63c994b9d22944 1.22kB / 1.22kB  1.0s
 => => sha256:98028cf281bb5d49ace5e1ddbd4509e8f1382fe80ef 3.25kB / 3.25kB  0.0s
 => => sha256:e88cb4c0b31e04d06143fbb77ee0dd2fa5188ddabe 1.51MB / 1.51MB  11.1s
 => => sha256:06b43d55bbbc28524f0252a0034c930062ec1d941e 1.13MB / 1.13MB  12.0s
 => => sha256:697905244cafadcc2be9f968b8da0a6d31eb238486d621 116B / 116B  13.0s
 => => sha256:ebd0c6090698c0d71bfc98dcba0c8746d733b40f665d30 262B / 262B  14.7s
 => => sha256:3e961522d85cd6b1aa0293250773d96d51f9 259.77MB / 259.77MB  1136.3s
 => => sha256:35581a5e0588d5e5b5a449d9b312512dc68256864b 5.00kB / 5.00kB  15.5s
 => => extracting sha256:d9d352c11bbd3880007953ed6eec1cbace76898828f34349  1.1s
 => => extracting sha256:0a4282d2a9c9631b43bac0401761e63c994b9d229447a606  0.0s
 => => extracting sha256:e88cb4c0b31e04d06143fbb77ee0dd2fa5188ddabe7da190  0.1s
 => => extracting sha256:06b43d55bbbc28524f0252a0034c930062ec1d941e66106d  0.0s
 => => extracting sha256:697905244cafadcc2be9f968b8da0a6d31eb238486d6218c  0.0s
 => => extracting sha256:ebd0c6090698c0d71bfc98dcba0c8746d733b40f665d3021  0.0s
 => => extracting sha256:3e961522d85cd6b1aa0293250773d96d51f989a8723ea543  5.8s
 => => extracting sha256:35581a5e0588d5e5b5a449d9b312512dc68256864b0eda10  0.0s
 => [2/2] WORKDIR /data/db                                                   0.1s
 => exporting to image                                                       0.0s
 => => exporting layers                                                      0.0s
 => => writing image sha256:4d505134e47e867d539c67d6c8de0ae66fe7e25f29e70  0.0s
 => => naming to docker.io/library/fullstack-todo-list-docker_database      0.0s
Building backend
[+] Building 161.9s (11/11) FINISHED                               docker:default
 => [internal] load build definition from dockerfile                         0.0s
 => => transferring dockerfile: 323B                                         0.0s
 => [internal] load metadata for docker.io/library/node:lts-alpine           7.2s
```

```
 => => extracting sha256:65b9c193e6b7c9d299d1f52e4accba0a196032bdc252ed87  1.6s
 => => extracting sha256:826f8ad948ffe9f7dc092f12ede9e6f14c1db6dae6c0c20b  0.0s
 => => extracting sha256:cb37e5b9a0a186241dd70c5ceab5af7a7a891d3180a4b160  0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 58.07kB                                      0.0s
 => [2/6] WORKDIR /usr/src/app                                            0.5s
 => [3/6] COPY [package.json, package-lock.json*, npm-shrinkwrap.json*, .  0.0s
 => [4/6] RUN npm install --production --silent && mv node_modules ../    17.8s
 => [5/6] COPY . .                                                        0.0s
 => [6/6] RUN chown -R node /usr/src/app                                  0.2s
 => exporting to image                                                    0.4s
 => => exporting layers                                                   0.4s
 => => writing image sha256:dedff3e407396fff1010384d04abd2cc4c75a2af29150  0.0s
 => => naming to docker.io/library/fullstack-todo-list-docker_backend     0.0s
Building frontend
[+] Building 102.3s (11/11) FINISHED                          docker:default
 => [internal] load build definition from dockerfile                      0.0s
 => => transferring dockerfile: 297B                                      0.0s
 => [internal] load metadata for docker.io/library/node:lts-alpine        6.7s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 2B                                           0.0s
 => [1/6] FROM docker.io/library/node:lts-alpine@sha256:41e4389f3d988d2ed  0.0s
 => [internal] load build context                                         0.0s
 => => transferring context: 302.23kB                                     0.0s
 => CACHED [2/6] WORKDIR /usr/src/app                                     0.0s
 => [3/6] COPY [package.json, package-lock.json*, npm-shrinkwrap.json*, .  0.1s
 => [4/6] RUN npm install --include dev                                  65.9s
 => [5/6] COPY . .                                                        0.0s
 => [6/6] RUN chown -R node /usr/src/app                                 26.6s
 => exporting to image                                                    2.8s
 => => exporting layers                                                   2.8s
 => => writing image sha256:95cd0c543b487b754ea6e64060ba9d1dcf313c8d6183b  0.0s
 => => naming to docker.io/library/fullstack-todo-list-docker_frontend    0.0s
Creating mongodb_container ... done
Creating fullstack-todo-list-docker_backend_1 ... done
Creating fullstack-todo-list-docker_frontend_1 ... done
Attaching to mongodb_container, fullstack-todo-list-docker_backend_1, fullstack-todo-list-docker_frontend_1
```

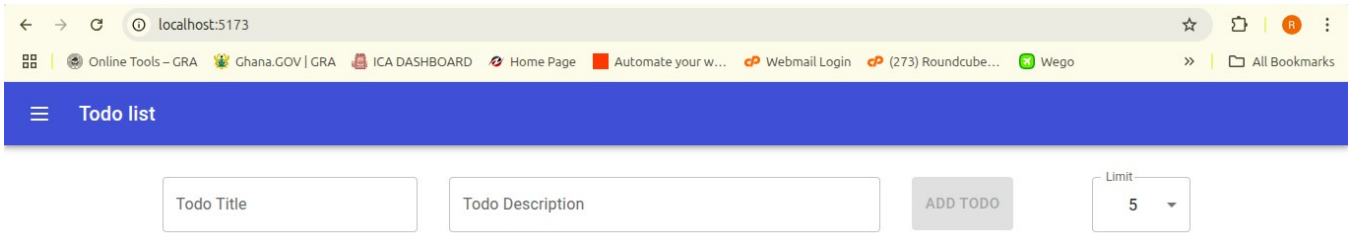
Start the Containers:

Run the containers

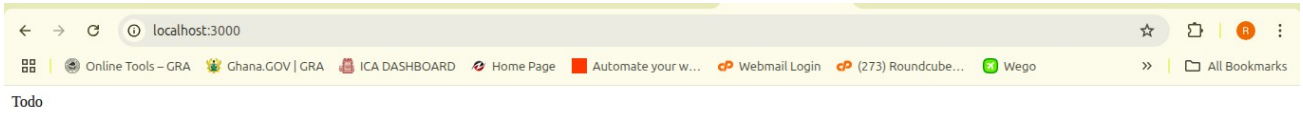
```
edem@edem-ThinkPad-T450:~/FullStack-ToDo-List-Docker$ docker-compose up
Starting mongodb_container ... done
Starting fullstack-todo-list-docker_backend_1 ... done
Starting fullstack-todo-list-docker_frontend_1 ... done
Attaching to mongodb_container, fullstack-todo-list-docker_backend_1, fullstack-todo-list-docker_frontend_1
```

Access the Application:

Frontend: Open your browser and navigate to http://localhost:5173.



Backend: You can access the backend API through http://localhost:3000.



Stop the Containers:

To stop the containers, run: