InvestMate: Deployment Manual

Group : Pace Super Kings

FrontEnd (ReactJs)

Installation

1. Audience Definition

- Frontend Developers
- System Administrators
- Full Stack Developers

2. Platform-Specific Deployment Instructions (Min requirements)

- **MacOS**
    - MacOS 10.10
    - RAM size : 4GB
    - Hard disk size : 10 GB Free disk space
    - Browser Chrome, Mozilla, Safari

- **Windows**
    - Window 10 or 11
    - RAM size : 4GB
    - Hard disk size : 10 GB Free disk space
    - CPU : 2 Core 1.7GHz +
    - Browser : Chrome, Mozilla or edge (browserlist)

- **Linux**
    - Ubuntu 16.04
    - RAM size : 4GB
    - Hard disk size : 10 GB Free disk space
    - CPU : 2 Core 1.7GHz +
    - Browser : Chrome, Mozilla
    - Package manager: apt

3. Prerequisite Installation:

- Install Node.js(all platforms)
- Npm

4. Configuration Instructions:

- Create an **.env** file at root/Code/frontend/
    - REACT_APP_LOGIN_ENDPOINT=<your_node_server_url>
    - REACT_APP_IMAGE_DETECTION_SERVER=<your_python_server_url>

5. Installation Scripts:

- cd root/Code/frontend/
- Run npm i in the terminal
- Run npm start

6. Testing and Troubleshooting:

- Use console.log(message) for Logging messages to the browser console
- Test accordingly, After testing

7. Deployment Scripts or Code Snippets:

- cd root/Code/frontend/
- Run npm i in the terminal
- Run npm run build

8. Server:

- Serve the build folder
- (Congrats you are on the production)

Backend Node Server

1. Audience Definition:

- Backend Developers
- System administrators
- Full Stack Developers

2. Platform-Specific Deployment Instructions:

- RAM size : 4GB
- Hard disk size : 15 GB Free disk space
- CPU : Duo Core, 1.7Ghz+

3. Prerequisite Installation:

- **Node.js:** 18.20.2
- **npm:** 10.5.0

4. Configuration Instructions:

- Create an **.env** file at root/Code/backend/node-server
    - PORT=<add-a-port-usually-3001>
    - SECURE_PORT=443
    - NODE_ENV=development
    - DB_USER=
    - DB_PASSWORD=

- DB_HOST=
- DB_NAME=
- JWT_ACCESS_EXPIRES_IN=90000000
- JWT_REFRESH_EXPIRES_IN=2592000000
- JWT_EMAIL_EXPIRES_IN=360000
- TOKEN_SECRET_KEY=
- FNP_API_KEY=
- ALPHAVANTAGE_API=
- S3_ACCESS=
- S3_SECRET=
- S3_REGION=

- For starting the app in production mode, you will need SSL certificates
  - Check out the guide from Lets-encrypt, Digital ocean and certbot
- It is assumed that you have a connection string for a Mongo database
  - If you do not, then install via docker
  - After installing Docker, do the following in the terminal.

```
docker run -d \
 -p 27017:27017 \  # Map container port 27017 to host port 27017
 --name my-mongo \  # Assign a name to the container
 -v /data/db:/data/db  # Optional: Mount a volume for persistent data
 mongo:latest
```

5. Installation Scripts:

- cd root/Code/backend/node-server
- Run npm i in the terminal
- Run npm start

6. API Testing and Troubleshooting:

- Follow the Postman API document for testing APIs to get Started
- Once done, let us proceed to deployment.

7.

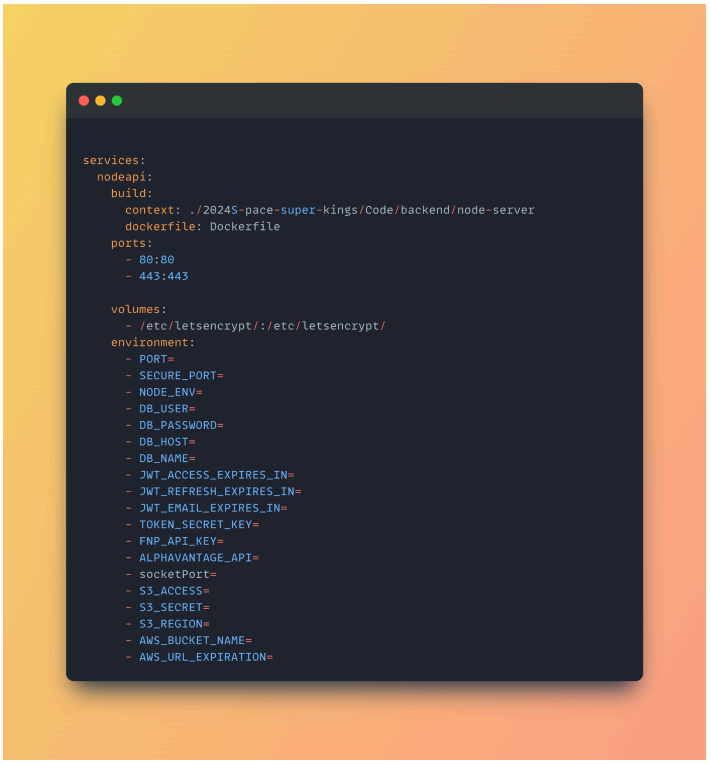
Deployment Configuration instructions

a. Create docker-compose.yml one directory above the root directory
b. Your docker-compose.yml should look like this.

8. Configuration instructions

c. Create docker-compose.yml one directory above the root directory

d. Your docker-compose.yml should look like this.



```
services:
  nodeapi:
    build:
      context: ./2024S-pace-super-kings/Code/backend/node-server
      dockerfile: Dockerfile
    ports:
      - 80:80
      - 443:443

    volumes:
      - /etc/letsencrypt/:/etc/letsencrypt/
    environment:
      - PORT=
      - SECURE_PORT=
      - NODE_ENV=
      - DB_USER=
      - DB_PASSWORD=
      - DB_HOST=
      - DB_NAME=
      - JWT_ACCESS_EXPIRES_IN=
      - JWT_REFRESH_EXPIRES_IN=
      - JWT_EMAIL_EXPIRES_IN=
      - TOKEN_SECRET_KEY=
      - FNP_API_KEY=
      - ALPHAVANTAGE_API=
      - socketPort=
      - S3_ACCESS=
      - S3_SECRET=
      - S3_REGION=
      - AWS_BUCKET_NAME=
      - AWS_URL_EXPIRATION=
```

e. Place the SSL certificates at the right place.

f. Set the NODE_ENV to production

g. In the same parent directory why holds our repository folder, open terminal

h. Run docker compose build

i. After successfully building run docker compose up

j. Check for any errors, and run it in the detached mode docker compose up -d

Backend Python Server (ML model)

1. Audience Definition:

- Backend developers
- System administrators
- Machine learning engineers.

2. Platform-Specific Deployment Instructions:

- RAM size : 8 GB
- Hard disk size : 10 GB Free disk space
- CPU : Duo Core, 2.4Ghz
- GPU is not a must

3. Prerequisite Installation:

- Python 3.6+
- NumPy 1.26.4
- Pandas 2.2.2
- PyTorch 2.0+
- Virtuallenv 20.26.1

4. Configuration Instructions:

- Cd root/Code/backend/pythonserver
- Download this .pt model and keep it in the same folder as app.py (`root/Code/backend/pythonserver`)

5. Deployment Scripts or Code Snippets:

- Make sure that you have virtualenv installed
- Activate the virtual environment source v/bin/activate
- Run pip install -r requirements.txt
- Finally run python app.py

Storage Services

MongoDb(No SQL)

- We have used an atlas service rather than setting up the database ourselves
- We can also host our own database and supply the DB_HOST, DB_PORT, DB_USERNAME and DB_PASSWORD in the .env file in the root of node-server

AWS S3 (Object Storage)

- Update the variables in the node-server folder to the values provided by AWS S3 service for you
    - S3_ACCESS=
    - S3_SECRET=
    - S3_REGION=