Challenge 3 Report: Establishing a Full-Stack Application Using Docker

Overview

The main objective of this project was to demonstrate how to configure a full-stack application using Docker. Docker streamlines deployment by creating isolated containers that simplify management, deployment, and scaling of applications. The architecture of the application consisted of three key components: a Node.js application, a MariaDB database, and an Nginx web server.

Preparation Requirements

Tools Required: The project necessitated the use of Docker and Docker Compose, essential for managing containerized applications. Installation links for Docker and Docker Compose can be found on their respective official websites.

Detailed Implementation Steps

1)Environment Setup:

Begin by crafting a .env file populated with crucial environment variables such as DB_USERNAME and DB_PASSWORD.

2) Docker file Construction:

- Node.js Configuration: Establish the Docker file within the Node.js application directory, detailing commands like FROM, COPY, RUN, and CMD.
- Database Setup: Set up the Docker file in the database directory to integrate the init.sql file during the build phase.
- Nginx Setup: Configure the Dockerfile for Nginx and detail the process of setting up the nginx.conf file. Important
- note: Adjust the COPY paths in the Dockerfiles to reflect the actual location relative to the root directory of each Dockerfile.

3)Compose File Configuration:

Construct the docker-compose.yml, defining how each service interacts within the architecture, including specifics on build context, ports, volumes, and dependencies among services.

4)Container Operations:

- Execute docker-compose up --build to build and initiate the containers.
- Verify the operational status of the containers using docker-compose ps.

5)Application Testing:

Validate the functionality by accessing the application via a web browser, ensuring that it performs as expected.

6)Troubleshooting:

Tackle common issues such as path errors in Dockerfile COPY commands and provide solutions for these challenges.

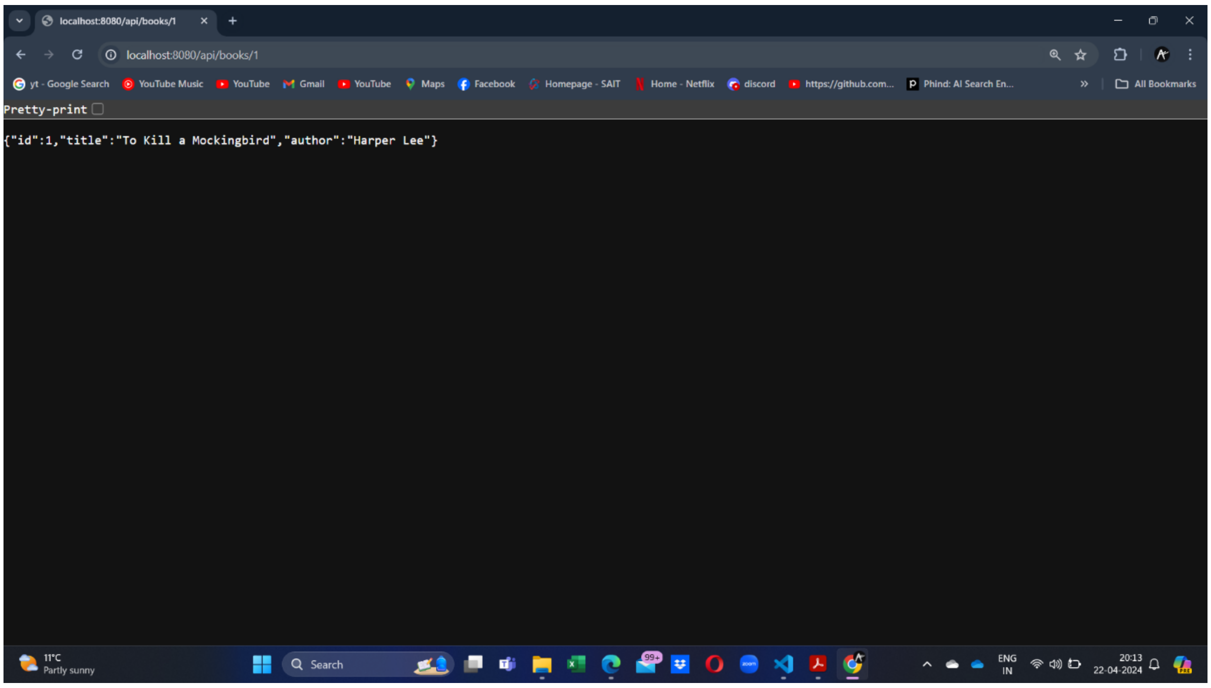
Conclusion

This documentation has encapsulated the process of setting up a full-stack application with Docker, emphasizing the configuration of individual components and their interconnections.

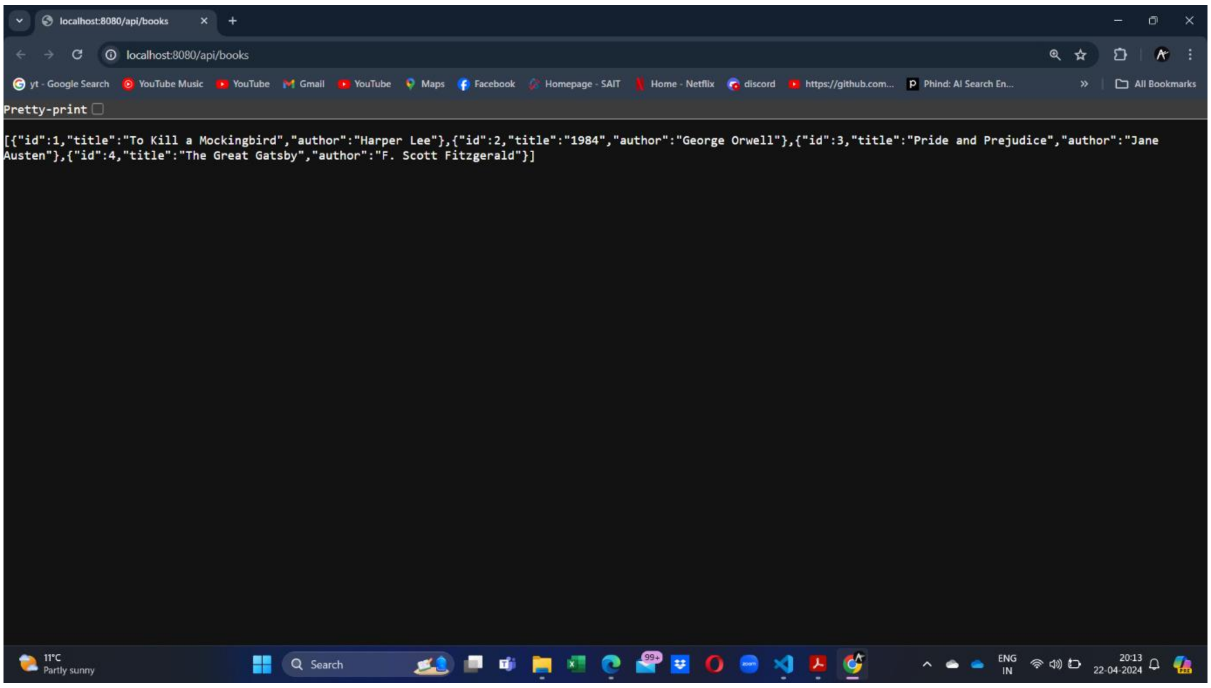
Cited References

Docker Documentation, Docker Docs. [Online] Available: https://docs.docker.com [Accessed: Apr 22,2024]

Screenshots



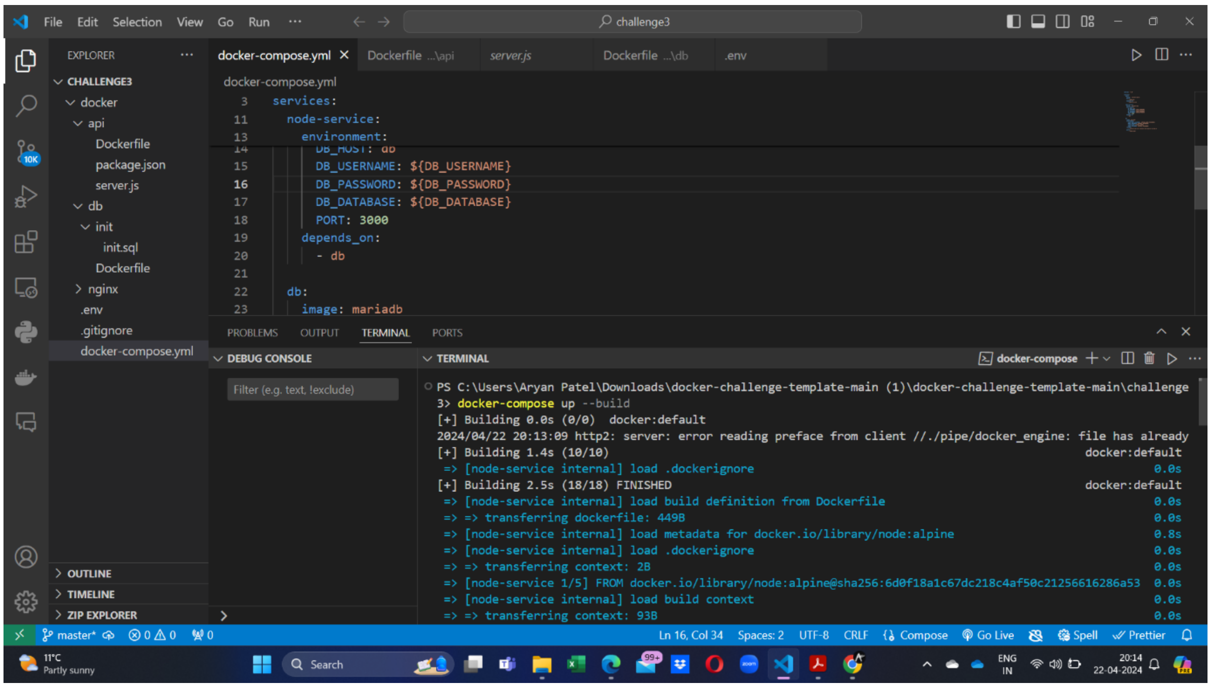
```
{"id":1,"title":"To Kill a Mockingbird","author":"Harper Lee"}
```



localhost:8080/api/books

Pretty-print

```
[{"id":1,"title":"To Kill a Mockingbird","author":"Harper Lee"},{"id":2,"title":"1984","author":"George Orwell"},{"id":3,"title":"Pride and Prejudice","author":"Jane
Austen"},{"id":4,"title":"The Great Gatsby","author":"F. Scott Fitzgerald"}]
```

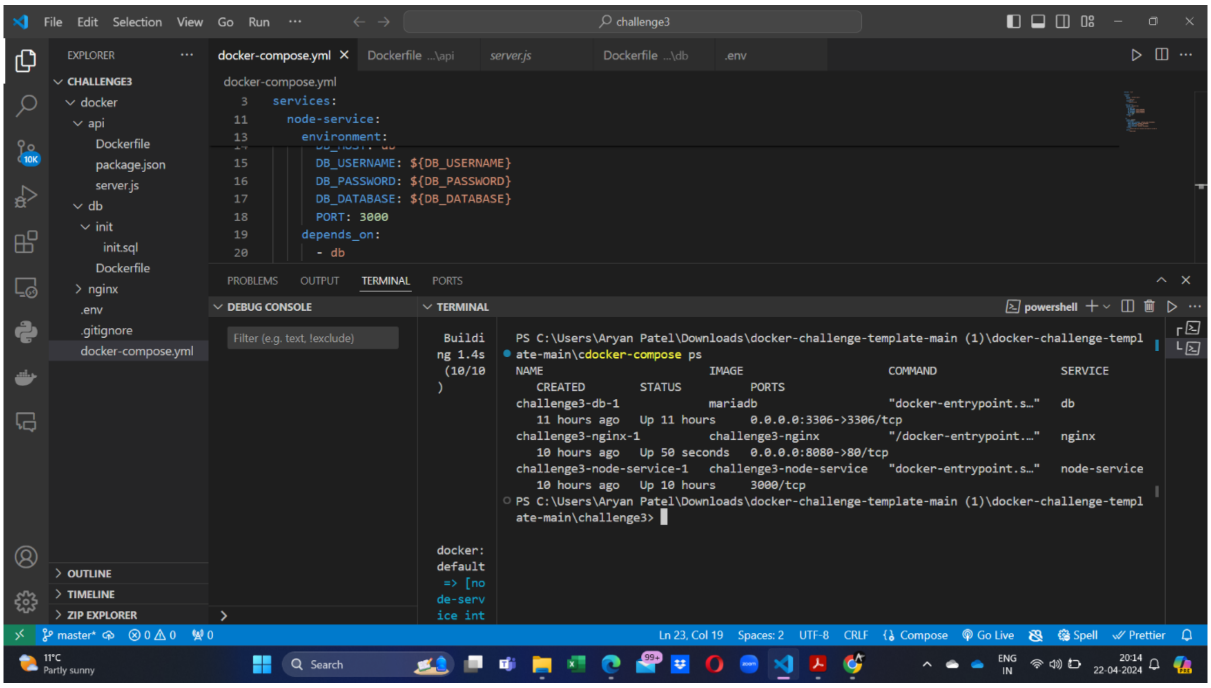


docker-compose.yml

```yaml
services:
  node-service:
    environment:
      DB_HOST: db
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_DATABASE: ${DB_DATABASE}
      PORT: 3000
    depends_on:
      - db

  db:
    image: mariadb
```

```
PS C:\Users\Aryan Patel\Downloads\docker-challenge-template-main (1)\docker-challenge-template-main\challenge
3> docker-compose up --build
[+] Building 0.0s (0/0)  docker:default
2024/04/22 20:13:09 http2: server: error reading preface from client //./pipe/docker_engine: file has already
[+] Building 1.4s (10/10)                                                                    docker:default
 => [node-service internal] load .dockerignore                                                         0.0s
[+] Building 2.5s (18/18) FINISHED                                                           docker:default
 => [node-service internal] load build definition from Dockerfile                                      0.0s
 => => transferring dockerfile: 449B                                                                   0.0s
 => [node-service internal] load metadata for docker.io/library/node:alpine                            0.8s
 => [node-service internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                                        0.0s
 => [node-service 1/5] FROM docker.io/library/node:alpine@sha256:6d0f18a1c67dc218c4af50c21256616286a53  0.0s
 => [node-service internal] load build context                                                         0.0s
 => => transferring context: 93B                                                                       0.0s
```



```yaml
services:
  node-service:
    environment:
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_DATABASE: ${DB_DATABASE}
      PORT: 3000
    depends_on:
      - db
```

```
Buildi
ng 1.4s
(10/10
)

docker:
default
=> [no
de-serv
ice int
```

```
PS C:\Users\Aryan Patel\Downloads\docker-challenge-template-main (1)\docker-challenge-templ
ate-main\cdocker-compose ps
NAME                        IMAGE                       COMMAND                     SERVICE
   CREATED          STATUS          PORTS
challenge3-db-1             mariadb                     "docker-entrypoint.s…"   db
   11 hours ago     Up 11 hours     0.0.0.0:3306->3306/tcp
challenge3-nginx-1          challenge3-nginx            "/docker-entrypoint.…"   nginx
   10 hours ago     Up 50 seconds   0.0.0.0:8080->80/tcp
challenge3-node-service-1   challenge3-node-service     "docker-entrypoint.s…"   node-service
   10 hours ago     Up 10 hours     3000/tcp
PS C:\Users\Aryan Patel\Downloads\docker-challenge-template-main (1)\docker-challenge-templ
ate-main\challenge3>
```