DOCKER CAPSTONE PROJECT

BUILDING AND DEPLOYING A MICROSERVICES APPLICATION

DOCUMENTATION

By:

1. Sinchana
2. Pratyush
3. Akarsh

Introduction:

Overview

This project aims to develop a scalable and containerized micro eCommerce application using **React** for the front-end and **Docker** for containerization. The application will provide a basic yet functional eCommerce platform with features such as product listings, product details, and a shopping cart.

Objectives

- **Frontend Development**: Build a responsive and interactive user interface using React.
- **Containerization**: Package the application into Docker containers for consistent and isolated deployment environments.
- **Scalability**: Ensure the application can be easily scaled and maintained through Docker's containerization.

Technologies Used

- **React**: A popular JavaScript library for building user interfaces. React will be used to create dynamic and responsive components.
- **Docker**: An open platform for developing, shipping, and running applications in containers. Docker will be used to package the React application for consistent deployment across different environments.

1. Prerequisites

Before you start, ensure you have the following installed:

- **Node.js**: Download Node.js
- **Docker**: Download Docker
- **npm**: Comes with Node.js

PROJECT STRUCTURE

The project is divided into two main parts:

1. **Frontend Application**: Built with React, this part includes:
    - **Components**: Reusable React components such as Home, Product, and Cart.
    - **Routing**: React Router for handling navigation within the application.
    - **State Management**: (Optional) Libraries like React Context API or Redux for managing application state.
2. **Docker Configuration**: To containerize the React application, including:
    - **Dockerfile**: Defines the build and runtime environment for the application.
    - **Docker Compose (Optional)**: Manages multi-container Docker applications if additional services are required (e.g., backend API, database).

PROCESS:

Step 1: Define the Project Scope

1. Application Type: E-commerce

2. Microservices*: User authentication, product management, orders and payment processing.

Step 2: Set Up Your Development Environment

1. Docker Installation

2. Docker Compose Installation

3. Version Control Set Up

Step 3: Design the Application Architecture

1. Define Service Boundaries.

2. Technologies.

3. Database Design

Step 4: Develop Microservices

1. Create Repositories
2. Write Code
3. Create Docker files.

Step 5: Containerize the Application

1. Build Docker Images
2. Test.

Step 6: Set Up Docker Compose

1. Create docker
2. Configure Networking
3. Environment Variables

Step 7: Orchestrate with Docker Compose

1. Run Docker Compose
2. Test Integration

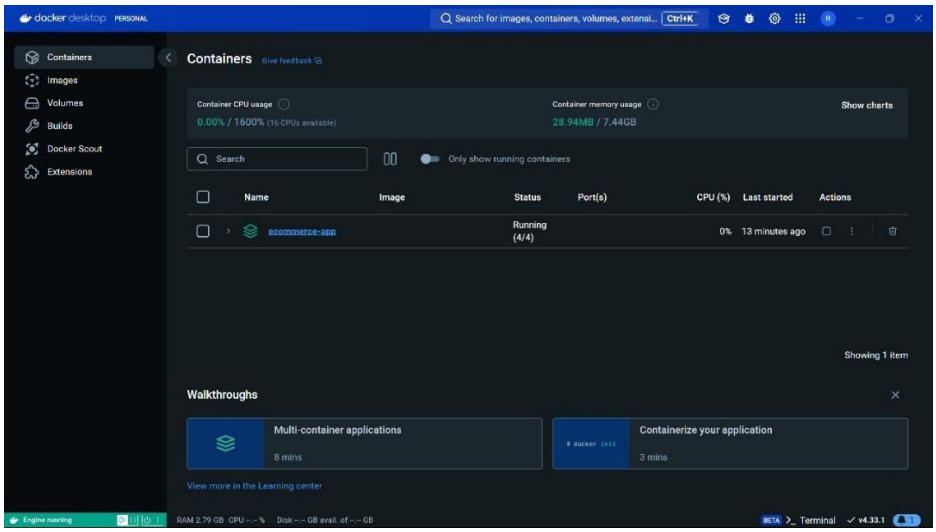
FUTURE ENHANCEMENTS

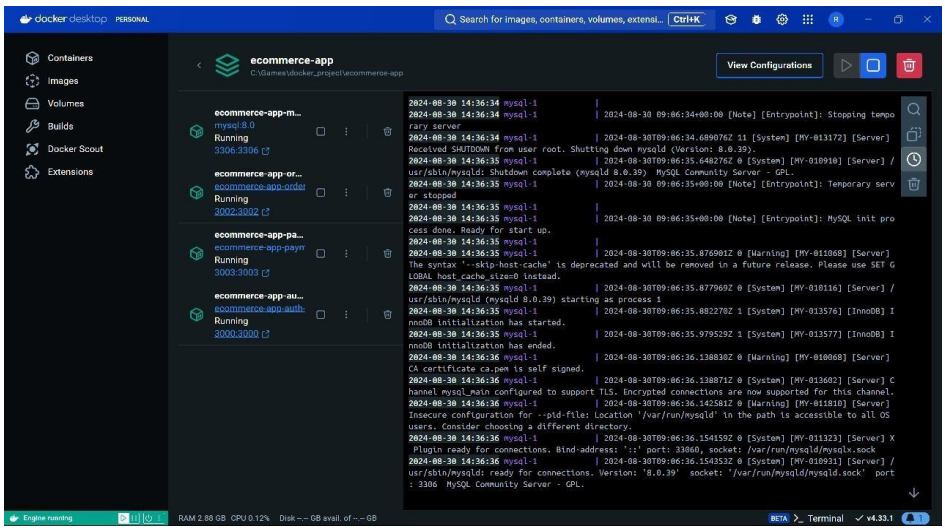
While the core features have been implemented, there are several areas for future enhancement:

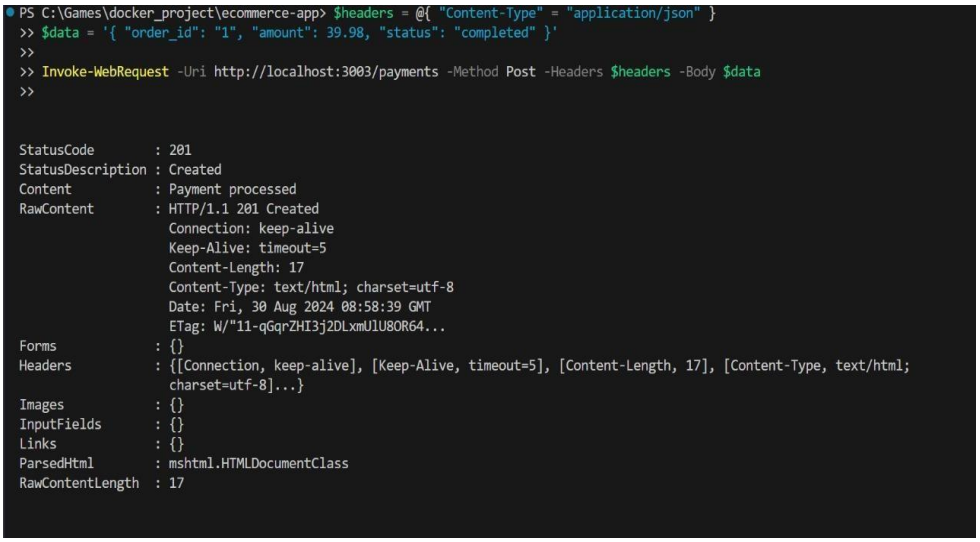
- **Backend Integration**: Connect the front-end application with a backend API to handle data persistence, user authentication, and more complex business logic.
- **Database Integration**: Incorporate a database to manage product information, user data, and transaction history.
- **User Authentication**: Implement user login and registration features to personalize the shopping experience and manage user-specific data.

DOCKER IMAGES:

```
PS C:\Games\docker_project\ecommerce-app> docker-compose ps
time="2024-08-30T14:51:07+05:30" level=warning msg="C:\\Games\\docker_project\\ecommerce-app\\docker-compose.yml: the attribute `versio
n` is obsolete, it will be ignored, please remove it to avoid potential confusion"
NAME                                IMAGE                                COMMAND                  SERVICE           CREATED          STATUS
      PORTS
ecommerce-app-auth-service-1        ecommerce-app-auth-service           "docker-entrypoint.s…"   auth-service      14 minutes ago   Up 14 min
utes   0.0.0.0:3000->3000/tcp
ecommerce-app-mysql-1               mysql:8.0                            "docker-entrypoint.s…"   mysql             14 minutes ago   Up 14 min
utes   0.0.0.0:3306->3306/tcp, 33060/tcp
ecommerce-app-order-service-1       ecommerce-app-order-service          "docker-entrypoint.s…"   order-service     14 minutes ago   Up 14 min
utes   0.0.0.0:3002->3002/tcp
ecommerce-app-payment-service-1     ecommerce-app-payment-service        "docker-entrypoint.s…"   payment-service   14 minutes ago   Up 14 min
utes   0.0.0.0:3003->3003/tcp
PS C:\Games\docker_project\ecommerce-app>
```







Conclusion:

The eCommerce micro application project successfully demonstrates the integration of **React** and **Docker** to build a scalable and easily deployable web application. By following this project, several key objectives have been achieved