DeVops project

"Game Deployment using Docker"

Bachelor in Technology

(Computer science and engineering)

SCHOOL OF CSE (LPU)

PHAGWARA

1. ROHIT KAPRI	**REG NO:** 12104800	**ROLLNO:**55

SECTION: KO308

Course code: INT332

Faculty: DR. HARPREET KAUR

LOVELY PROFESSIONAL UNIVERSITY

Phagwara (Punjab)

DECLARATION

This is to declare that this report has been written by me. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I aware that if any part of the report is found to be copied, I will take full responsibility for it.

Rohit Kapri

(Signature of Student)

Name of the Student1: Rohit Kapri

Registration Number: 12104800

Date:20/04/2024

<u>Acknowledgement</u>

I would like to take this opportunity to express my gratitude to everyone who supported me throughout the completion of this individual project, "Creating a Game using Docker and Deploying to AWS Cloud."

I extend my heartfelt appreciation to my mam, Dr Harpreet Kaur, for her guidance, encouragement, and valuable feedback during the development process. Her expertise has been instrumental in shaping the project's direction and ensuring its success.

I am also thankful to Lovely Professional University for providing the necessary resources and infrastructure for me to carry out this project effectively.

I would like to acknowledge the open-source community for developing the technologies and tools that were utilized in this project, particularly Docker and AWS.

Furthermore, I am grateful to my family and friends for their unwavering support, understanding, and encouragement throughout the project's duration.

Lastly, I would like to thank myself for the determination, hard work, and perseverance invested in bringing this project to fruition.

TABLE OF CONTENT

1. INTRODUCTION

The primary objective of this project was to demonstrate the implementation of DevOps practices in the development and deployment of a game application, leveraging Docker containerization and deploying it to the AWS Cloud. DevOps, a combination of development and operations practices, aims to streamline the software development lifecycle by fostering collaboration, automation, and continuous integration and delivery (CI/CD).

In this project, I embarked on the journey of creating a game application from scratch, focusing on incorporating DevOps principles right from the initial development stages to the deployment phase. The chosen game concept served as a practical platform to explore various DevOps tools and methodologies.

The decision to employ Docker for containerization was driven by its ability to encapsulate the game application and its dependencies into portable, lightweight containers, ensuring consistency across different environments and facilitating seamless deployment. Additionally, AWS Cloud was selected as the deployment platform due to its scalability, reliability, and extensive suite of cloud services, enabling easy provisioning and management of resources.

Throughout this report, I will provide detailed insights into the project setup, development process, Docker configuration, AWS deployment, testing procedures, challenges encountered, and future enhancements. By documenting my journey in implementing DevOps practices in this individual project, I aim to share valuable insights and lessons learned that can benefit both aspiring DevOps practitioners and game developers alike.

In today's dynamic software development landscape, DevOps has emerged as a cornerstone methodology for fostering collaboration between development and operations teams, breaking down silos, and accelerating the delivery of high-quality software products. By embracing DevOps principles, organizations can achieve faster time-to-market, improved efficiency, and enhanced reliability of their software applications.

The decision to undertake this project stemmed from a desire to gain hands-on experience in implementing DevOps practices in a real-world scenario. By choosing to develop a game application, I aimed to explore the applicability of DevOps methodologies beyond traditional software development projects and delve into the unique challenges and opportunities presented by the gaming industry.

Throughout the project lifecycle, I adhered to the core tenets of DevOps, including automation, continuous integration, continuous delivery, and infrastructure as code. These principles guided the development process, ensuring that changes to the game application could be seamlessly integrated, tested, and deployed in an efficient and repeatable manner.

Furthermore, by leveraging Docker for containerization, I aimed to encapsulate the game application and its dependencies in isolated, portable environments, thereby simplifying the deployment process and enhancing scalability and resource utilization. The integration of Docker with AWS Cloud provided a scalable and reliable infrastructure for hosting the game application, while also enabling easy provisioning and management of resources through infrastructure as code practices.

In essence, this project represents a practical exploration of DevOps principles in action, demonstrating their effectiveness in streamlining the development and deployment of a game application. Through the documentation of my experiences, insights, and learnings, I hope to contribute to the broader discourse on DevOps and inspire others to embrace its transformative potential in their own projects and endeavors.

2. Scope of the Study

The scope of this study encompasses the entire lifecycle of developing a game application using DevOps practices, with a specific focus on containerization with Docker and deployment to the AWS Cloud. Key components of the study include:

1. Game Development: The study involves conceptualizing, designing, and implementing a game application from scratch. This includes defining game mechanics, creating assets, developing game logic, and ensuring user interaction.

2. DevOps Integration: DevOps principles are integrated into every phase of the project, from initial development to deployment. This includes implementing automation, continuous integration, continuous delivery, and infrastructure as code practices.

3. Containerization with Docker: Docker is utilized for containerizing the game application and its dependencies. The study covers the creation of Docker images, Dockerfile configuration, Docker Compose setup (if applicable), and the orchestration of containers.

4. AWS Deployment: The game application is deployed to the AWS Cloud infrastructure. This involves provisioning AWS resources such as EC2 instances, S3 buckets, RDS databases, and configuring them to host and support the game application.

5. Testing and Quality Assurance: Testing strategies are employed to ensure the reliability, performance, and functionality of the game application. This includes unit testing, integration testing, and end-to-end testing of the Docker containers and deployed application on AWS.

6. Monitoring and Logging: Monitoring tools are implemented to track

the performance and health of the game application and infrastructure on AWS. Logging mechanisms are established to capture and analyze relevant logs for troubleshooting and optimization purposes.

7. Challenges and Solutions: The study identifies challenges encountered during the project and documents the solutions devised to overcome them. This includes addressing issues related to development, deployment, scalability, and reliability.

8. Future Enhancements: Suggestions for future improvements and enhancements to the game application and DevOps processes are provided. This includes exploring additional features, optimizing performance, and refining deployment strategies.

9. Documentation and Reporting: The study encompasses thorough documentation of the project's progress, including the creation of detailed reports, code documentation, and user guides. This documentation serves as a valuable resource for understanding the project's implementation, facilitating knowledge transfer, and enabling future maintenance and updates.

10. Security Considerations: Security measures are integrated into the development and deployment process to ensure the confidentiality, integrity, and availability of the game application and its data. This includes implementing security best practices, such as encryption, access control, and vulnerability scanning, to mitigate potential threats and vulnerabilities.

11. Scalability and Performance Optimization: The study explores strategies for optimizing the scalability and performance of the game application on AWS Cloud infrastructure. This includes scaling resources dynamically based on demand, optimizing resource utilization, and implementing caching and load balancing mechanisms to enhance

responsiveness and user experience.

12. Cost Optimization: Cost optimization techniques are employed to minimize the operational costs associated with hosting and maintaining the game application on AWS Cloud. This includes optimizing resource utilization, leveraging cost-effective AWS services, and implementing cost monitoring and budgeting tools to ensure cost-efficiency without compromising performance or reliability.

13. Continuous Improvement: The study emphasizes the importance of continuous improvement and iterative refinement of the game application and DevOps processes. This involves soliciting feedback from users, monitoring key performance metrics, and iteratively implementing enhancements and optimizations to drive continuous improvement and innovation.

14. Community Engagement: The study encourages community engagement and collaboration by sharing insights, experiences, and lessons learned with the broader DevOps and gaming communities. This includes participating in forums, conferences, and meetups, contributing to open-source projects, and fostering knowledge sharing and collaboration with peers and industry experts.

15. Ethical and Legal Considerations: Ethical and legal considerations are taken into account throughout the project lifecycle to ensure compliance with relevant regulations and ethical standards. This includes safeguarding user privacy and data security, adhering to intellectual property rights, and maintaining transparency and integrity in all aspects of the project.

16. User Experience and Accessibility: The study emphasizes the importance of user experience (UX) design and accessibility considerations in the development of the game application. This

includes designing intuitive user interfaces, optimizing gameplay mechanics for engaging user experiences, and ensuring accessibility for users with diverse needs and abilities. By prioritizing UX design and accessibility, the study aims to create a game application that is enjoyable, inclusive, and accessible to all users, thereby enhancing its overall usability and impact.

By addressing these additional aspects, the study aims to provide a comprehensive and holistic understanding of the development and deployment of a game application using DevOps practices, Docker containerization, and AWS Cloud infrastructure. It seeks to equip practitioners and stakeholders with the knowledge, tools, and insights necessary to successfully undertake similar projects and drive innovation in the field of software development and DevOps.

The scope of the study is limited to the development and deployment of a single game application using DevOps practices, Docker containerization, and AWS Cloud infrastructure. It aims to provide a comprehensive understanding of how these technologies and methodologies can be effectively utilized to streamline the development and deployment process of a real-world application.

3.Existing System

- ## Introduction:

The existing system represents the current state of the software application or platform before any modifications or enhancements are made. Understanding the strengths and weaknesses of the existing system provides valuable insights into areas for improvement and guides the development of the new system.

- ## Existing Software:

This section provides an overview of the software currently in use. It includes details such as the purpose of the software, its functionalities, technologies used, and any limitations or challenges encountered with the existing system.

- ## DFD for Present System:

A Data Flow Diagram (DFD) illustrates the flow of data within the existing system. It visually represents the processes, data stores, and data flows within the system, helping to identify potential bottlenecks, inefficiencies, and opportunities for optimization.

- ## What's New in the System to be Developed:

This section outlines the enhancements, modifications, or new features that will be introduced in the system to be developed. It highlights the improvements and innovations that will address the limitations or shortcomings of the existing system and deliver added value to users.

In the context of the project "Creating a Game using Docker and Deploying to AWS Cloud," the existing system may refer to any previous iterations of the game application or any manual processes used in development and deployment. The DFD for the present system would illustrate the flow of data and processes within the existing game development and deployment workflow. Finally, the section on "What's New in the System to be Developed" would outline the enhancements and innovations introduced in the new game application, such as Docker containerization, AWS deployment, improved gameplay features, and enhanced user experience.

- **Comparison with Existing System:**

A detailed comparison between the existing system and the proposed system provides insights into the improvements and advancements introduced. This comparison covers aspects such as functionality, performance, scalability, security, and usability. It highlights how the new system addresses the limitations of the existing system and delivers added value to users and stakeholders.

- **User Feedback and Issues with Existing System:**

User feedback and issues encountered with the existing system offer valuable insights into areas that require improvement. This feedback can be gathered through surveys, interviews, user testing, or support tickets. By addressing these pain points and incorporating user suggestions, the new system can better meet the needs and expectations of its users.

- **Technological Advancements and Industry Trends:**

Technological advancements and industry trends play a crucial role in shaping the development of the new system. This section discusses the latest innovations, best practices, and emerging technologies relevant to the project domain. By leveraging these advancements, the new system can stay ahead of the curve and remain competitive in the market.

- **Regulatory and Compliance Considerations:**

Regulatory and compliance requirements, such as data privacy laws, industry standards, and security regulations, must be considered when developing the new system. This section outlines the regulatory landscape applicable to the project and how the new system ensures compliance with relevant laws and regulations.

4. Problem Analysis

- **Product Definition:**

The product definition stage involves clearly defining the scope, objectives, and requirements of the project. It includes identifying the target audience, defining the features and functionalities of the product, and establishing success criteria. In the context of the project "Creating a Game using Docker and Deploying to AWS Cloud," the product definition would outline the goals of developing the game application, the target audience (e.g., gamers), and the desired features and gameplay mechanics.

By clearly defining the scope and objectives of the project, the team can ensure alignment between the final product and user expectations, ultimately leading to greater user satisfaction and success in the market.

- **Feasibility Analysis:**

Feasibility analysis assesses the technical, economic, and operational feasibility of the project. It evaluates whether the proposed solution is viable and practical within the constraints of resources, technology, and time. In the case of the project, feasibility analysis would examine the technical feasibility of using Docker for containerization and AWS Cloud for deployment, the economic feasibility of the project within budgetary constraints, and the operational feasibility of integrating DevOps practices into the development process.

Operational feasibility considers the practicality of implementing the solution within the existing organizational framework. By conducting a thorough feasibility analysis, the project team can identify potential challenges and risks early on and make informed decisions about the project's direction and viability.

- **Project Plan:**

The project plan outlines the roadmap for executing the project from start to finish. It includes defining project objectives, identifying tasks and milestones, estimating resource requirements, and establishing timelines. The project plan serves as a blueprint for project management and helps ensure that the project stays on track and meets its objectives within the specified timeframe and budget. In the context of the project, the project plan would detail the tasks involved in developing the game application, implementing DevOps practices, and deploying the application to AWS Cloud, along with associated timelines and resource allocations.

By conducting a thorough problem analysis encompassing product definition, feasibility analysis, and project planning, the project team can lay a solid foundation for the successful execution of the project. It provides clarity on project goals, assesses the viability of the proposed solution, and establishes a roadmap for achieving project objectives.

By breaking down the project into manageable tasks and establishing clear timelines, the project plan helps to ensure that the project stays on track and that deadlines are met. Additionally, the project plan provides a basis for monitoring progress, identifying potential bottlenecks, and making adjustments as needed to keep the project on course. A well-defined project plan is essential for maximizing productivity, minimizing risks, and ultimately achieving project success.

5. Software Requirement Analysis

Introduction:

The software requirement analysis phase is a critical step in the software development lifecycle, where the project team identifies, analyzes, and documents the requirements of the system to be developed. This phase lays the foundation for designing and implementing a solution that meets the needs of the users and stakeholders. By understanding the requirements thoroughly, the project team can ensure that the final product aligns with the expectations and objectives of the project.

General Description:

The general description provides an overview of the system to be developed, including its purpose, scope, and context within the larger organizational or business environment. It outlines the key features and functionalities of the system and describes how it will address the needs and requirements of its users. Additionally, the general description may include information about the technology stack, platform compatibility, and any constraints or assumptions that need to be considered during the development process.

Specific Requirements:

The specific requirements section delves deeper into the detailed functional and non-functional requirements of the system. Functional requirements specify the specific actions and behaviors that the system must support, such as user interactions, data processing, and system functionality. Non-functional requirements define the quality attributes of the system, including performance, reliability, security, usability, and scalability. This section also includes any regulatory or compliance requirements that the system must adhere to, as well as any constraints or dependencies that may impact the development process.

6. Design

- ## System Design:

System design involves translating the requirements gathered during the software requirement analysis phase into a blueprint for the system architecture. This includes defining the overall structure of the system, including its components, modules, and interactions. System design also encompasses decisions regarding the technologies, frameworks, and platforms to be used, as well as the distribution of functionality across different layers of the application. By creating a robust system design, the project team can ensure that the system is scalable, maintainable, and extensible, and that it meets the performance and reliability requirements of its users.

- ## Design Notations:

Design notations are visual representations used to communicate the system design effectively. This may include diagrams such as UML (Unified Modeling Language) diagrams, flowcharts, data flow diagrams (DFDs), entity-relationship diagrams (ERDs), and architectural diagrams. These notations help to convey complex design concepts in a clear and concise manner, facilitating communication and collaboration among project stakeholders. By using standardized design notations, the project team can ensure that everyone involved in the project understands the system architecture and design decisions.

ERDs are useful for modeling the relationships between different entities in a database schema. By employing design notations effectively, the project team can ensure that everyone involved in the project understands the system architecture and design decisions, facilitating communication, collaboration, and consensus among stakeholders.

- ## Detailed Design:

The detailed design phase involves specifying the implementation details of the system, including the algorithms, data structures, database schemas, and interfaces to be used. This includes defining the methods and procedures for implementing each component or module of the system, as well as documenting any design patterns or best practices to be followed. Additionally, the detailed design phase may involve creating prototypes or mockups to validate the design decisions and gather feedback from stakeholders. By creating a detailed design, the project team can ensure that the system is implemented correctly and efficiently, and that it meets the requirements specified during the software requirement analysis phase.

- **Flowcharts:**

In the context of the project "Creating a Game using Docker and Deploying to AWS Cloud," flowcharts can be instrumental in visualizing the various processes involved in the development and deployment pipeline. For example, a flowchart can illustrate the sequence of steps involved in building the game application, containerizing it with Docker, and deploying it to the AWS Cloud. Each step in the process, such as compiling the code, creating Docker images, pushing images to a registry, and provisioning AWS resources, can be represented using appropriate symbols and arrows to indicate the flow of control. Flowcharts help to identify dependencies, decision points, and potential bottlenecks in the pipeline, enabling developers to optimize the workflow for efficiency and reliability.

- **Pseudo code:**

Pseudo code can be used to outline the algorithmic logic and procedural steps involved in the game application's functionality. For example, pseudo code can describe the game mechanics, such as player movement, collision detection, scoring, and level progression. It can also outline the logic for implementing features like game menus, user input handling, and game state management. By writing pseudo code, developers can clarify the structure and behavior of the game application before diving into the actual coding process. Pseudo code helps to break down complex functionality into manageable steps, facilitating the development process and ensuring that the final implementation aligns with the project requirements and objectives.

7. Testing

- **Functional Testing:**

Functional testing verifies that each function of the game application performs as expected according to the defined requirements. It focuses on validating the functionality of the game, including user interactions, game mechanics, and feature implementations. Test cases are designed to cover various scenarios, ensuring that the game behaves correctly under different conditions. Functional testing ensures that players can navigate menus, interact with game objects, progress through levels, and achieve game objectives without encountering any bugs or errors.

- **Structural Testing:**

Structural testing, also known as white-box testing, examines the internal structure of the game application, including the code logic, control flow, and program structure. It aims to uncover errors or defects that may arise from the implementation of the code. Techniques such as code coverage analysis, branch coverage, and path coverage are used to ensure that all parts of the code are exercised and that potential issues are identified and addressed. Structural testing helps to improve code quality, identify coding errors, and ensure that the game application behaves as intended.

- **Levels of Testing:**

The project undergoes testing at multiple levels to ensure comprehensive coverage and validation of its functionality and performance. These levels include:
1. Unit Testing: Testing individual components or modules of the game application in isolation to ensure they function correctly.

2. Integration Testing: Verifying the interactions and interfaces between different components to ensure they work together seamlessly.

3. System Testing: Evaluating the behavior of the entire game application as a whole to ensure it meets the specified requirements and functions as expected.

4. Acceptance Testing: Validating the game against user requirements and expectations to ensure it meets the desired quality standards before release.

Each level of testing serves a specific purpose in ensuring the overall quality and reliability of the game application.

▪ Testing the Project:

Testing the project involves executing test cases designed specifically for the game application developed using Docker and deployed to the AWS Cloud. This includes:

- Testing Docker Containerization: Validating that the game application runs correctly within Docker containers, ensuring consistency and portability across different environments.

- Testing AWS Deployment: Verifying that the game application can be deployed to the AWS Cloud infrastructure and accessed by users reliably and securely.

- Testing Functional Requirements: Ensuring that the game application meets the functional requirements outlined in the project specifications, including gameplay mechanics, user interactions, and feature implementations.

- Testing Non-functional Requirements: Assessing non-functional aspects such as performance, scalability, security, and usability to ensure the game application meets quality standards and user expectations.

By thoroughly testing the project, potential issues and defects are identified and addressed early in the development process, ensuring a high-quality and reliable game application for users.

8. Implementation

- **Implementation of the Project:**

The implementation phase involves translating the design specifications and requirements into actual code and building the game application. Developers write code, integrate components, and perform unit testing to ensure that each module functions correctly. The implementation phase follows the design phase, where system architecture and detailed design specifications are finalized. During implementation, developers adhere to coding standards, best practices, and design patterns to ensure the codebase is maintainable, scalable, and efficient.

- **Conversion Plan:**

The conversion plan outlines the steps and procedures for transitioning from the old system or development environment to the new system developed as part of the project. It includes activities such as data migration, software installation, user training, and deployment strategies. The conversion plan ensures a smooth and seamless transition to the new system with minimal disruption to operations. It identifies potential risks, dependencies, and contingency plans to mitigate any issues that may arise during the conversion process. Stakeholder communication and coordination are essential to ensure all stakeholders are informed and prepared for the transition.

- **Post-Implementation and Software Maintenance:**

After the project is implemented, post-implementation activities and software maintenance are essential to ensure the continued functionality and reliability of the game application. This includes monitoring performance, addressing user feedback, and applying updates and patches as needed. Post-implementation activities also include documentation updates, training, and support for users and administrators. Software maintenance involves fixing bugs, optimizing performance, adding new features, and ensuring compatibility with evolving technologies and platforms. Regular maintenance activities help to ensure the long-term success and sustainability of the game application, providing ongoing value to users and stakeholders.

9. Project Legacy

- ## Current Status of the Project:

The current status of the project "Creating a Game using Docker and Deploying to AWS Cloud" includes the progress made in developing the game application, containerizing it with Docker, and deploying it to the AWS Cloud infrastructure. Key milestones achieved may include completing the game development phase, successfully containerizing the application, and deploying it to AWS Cloud services such as Amazon EC2 or AWS Elastic Beanstalk. The project may be in the final stages of testing and refinement before it is released to users or stakeholders for evaluation.

- ## Remaining Areas of Concern:

Despite the progress made, there may be remaining areas of concern that need to be addressed before the project can be considered complete. These areas may include unresolved technical issues, performance optimization tasks, or usability enhancements. Additionally, any outstanding requirements or user feedback that require further refinement or implementation should be prioritized. It's important to address these areas of concern promptly to ensure the project meets its objectives and delivers value to users.

- ## Technical and Managerial Lessons Learned:

Throughout the project, various technical and managerial lessons have been learned that can inform future endeavors. Technical lessons may include insights into the effectiveness of using Docker for containerization, best practices for deploying applications to AWS Cloud, and challenges encountered in game development and DevOps integration. Managerial lessons may include strategies for effective project planning, communication among team members, and resource allocation. Documenting these lessons learned enables the project team to improve processes, avoid common pitfalls, and drive success in future projects.

10. User Manual: Game Deployment Guide

▪ Introduction:

Welcome to the user manual for our game deployment process using Docker and AWS Cloud! This guide provides comprehensive instructions on how to deploy the game application to the AWS Cloud infrastructure using Docker containers. Whether you're a developer, system administrator, or DevOps engineer, this manual will walk you through the deployment process step-by-step.

▪ Table of Contents:

1. Prerequisites:

Before you begin the deployment process, ensure you have the following prerequisites in place:
- Access to an AWS account with appropriate permissions to create and manage resources.
- Docker installed on your local machine for containerization of the game application.
- Basic knowledge of Docker commands and AWS services such as EC2 and S3.

2. Docker Setup:

Follow these steps to set up Docker for containerization:
- Install Docker on your local machine by downloading and running the installer from the official Docker website.
- Verify the installation by running the "docker --version" command in your terminal or command prompt.
- Pull the game application Docker image from the Docker Hub repository or build the image from the source code using Dockerfile.

3. AWS Setup:
Prepare your AWS environment for deploying the game application:
- Create an AWS EC2 instance to host the Docker containers running the game application.
- Set up an AWS S3 bucket to store any static assets or resources required by the game.
- Configure security groups and network settings to allow inbound traffic to the EC2 instance.

4. Deploying the Game Application:
Deploy the game application to the AWS Cloud using Docker:
- Copy the Docker image of the game application to the EC2 instance.
- Run the Docker container on the EC2 instance, exposing the necessary ports for accessing the game.
- Configure environment variables and settings required by the game, such as database connections or API endpoints.

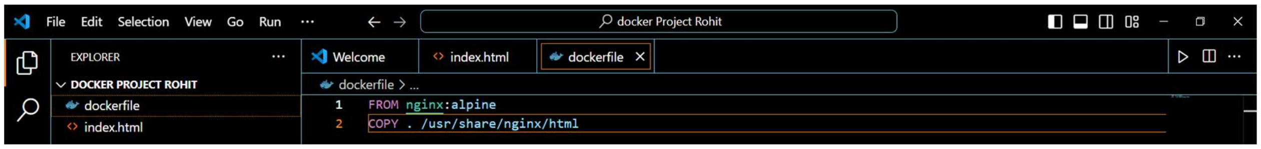
5. Accessing the Deployed Game:
Once the game application is deployed, you can access it using the public IP address or domain name of the EC2 instance. Open a web browser and navigate to the URL to start playing the game. Ensure that any firewall rules or security settings allow access to the game application.

6. Troubleshooting and Support:
If you encounter any issues during the deployment process, refer to the troubleshooting section of this manual for assistance. You can also seek support from our technical team or community forums for further help and guidance.

Thank you for using our game deployment guide! We hope you find it helpful in deploying the game application to the AWS Cloud with Docker. If you have any feedback or questions, please don't hesitate to contact us. Happy gaming!

<u>11.Source Code(Snapshot)</u>

<u>Step 1: Creating a Dockerfile</u>



```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

<u>Step 2:Creating Docker Image From Dockerfile</u>

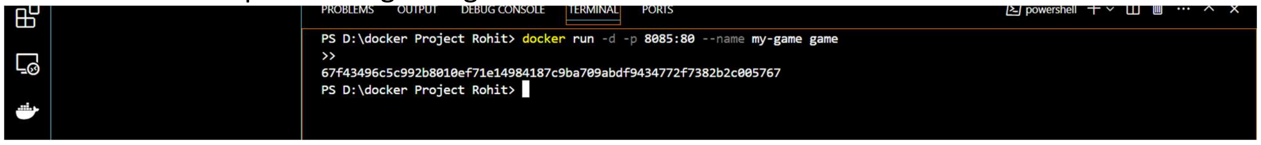
Open a terminal in the directory containing your game files and Dockerfile. Run the following command to build your Docker image:

- **docker build -t guess-game .**

<u>step 3:Run Docker Container using Image</u>

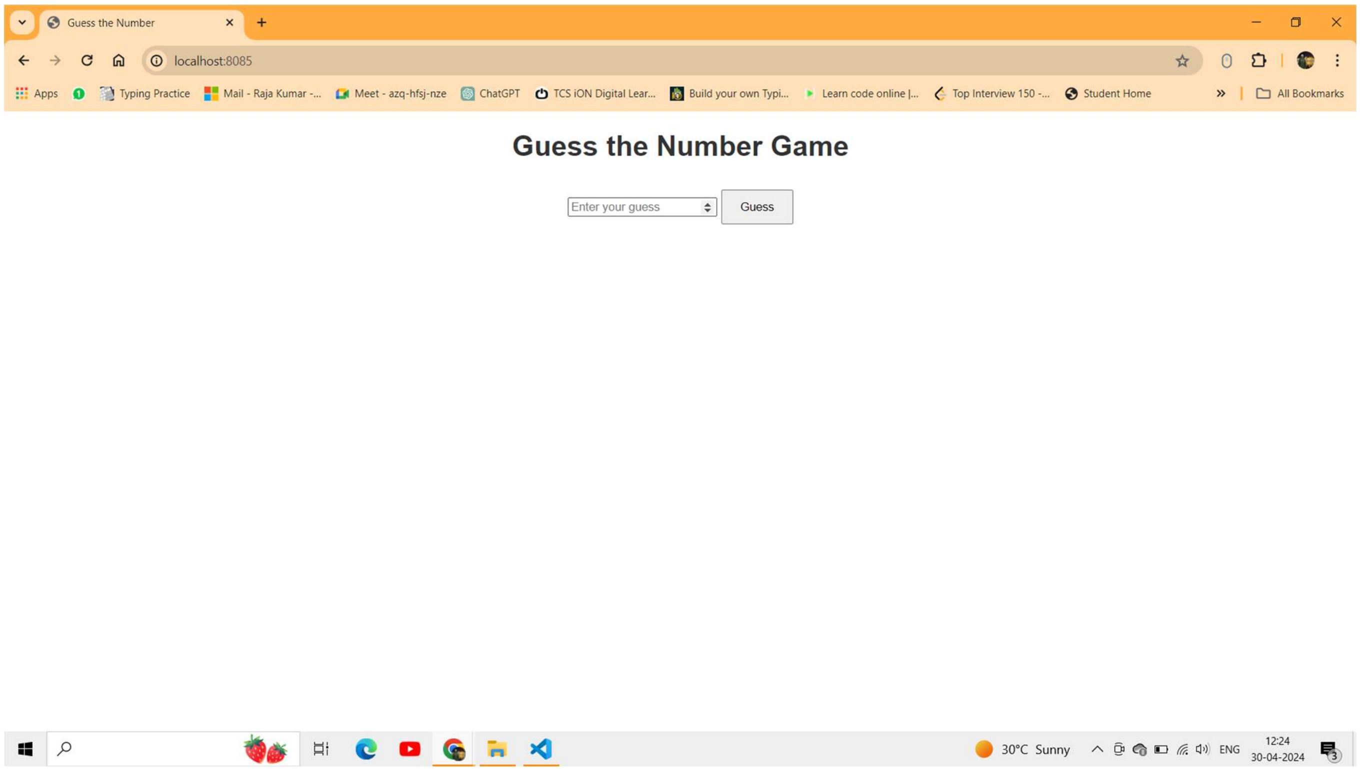
Once the image is built, you can run a container based on that image using the following command:

- docker run -d -p 8085:80 guess-game



```
PS D:\docker Project Rohit> docker run -d -p 8085:80 --name my-game game
>>
67f43496c5c992b8010ef71e14984187c9ba709abdf9434772f7382b2c005767
PS D:\docker Project Rohit>
```

<u>Step 4: Runing localhost:8085</u>



Guess the Number Game

12.Bibliography

1. https://docs.docker.com
2. google.com
3. youtube.com