

CS6006 - CLOUD COMPUTING

AWS BASED EMPLOYEE MANAGEMENT SYSTEM

TEAM DETAILS

ADITHYAA V S – 2020103503

MEDONA SHINY J - 2020103539

ABSTRACT:

The Employee Management System (EMS) represents a cutting-edge full-stack project meticulously integrated into the cloud infrastructure provided by Amazon Web Services (AWS). This initiative leverages essential AWS components, such as Amazon RDS for database management, EC2 instances for hosting, and S3 buckets for streamlined storage. The project's inception involves the creation of a database within Amazon RDS, establishing a secure connection to the remote database, and subsequently linking an EC2 instance to host the Spring Boot backend. As the development progresses, the EC2 Spring Boot backend is seamlessly connected to a React app, paving the way for the ultimate deployment of the React app on an AWS S3 bucket. This multi-faceted integration ensures an advanced and interconnected EMS, aligning with contemporary organizational requirements.

The integration process begins with the establishment of a robust foundation in Amazon RDS, ensuring efficient and secure database management. The subsequent steps involve connecting the remote database to the EMS project, fostering a seamless exchange of data critical for HR operations. The integration extends to the deployment of a Spring Boot backend on an EC2 instance, optimizing performance and scalability. The final phase focuses on linking the EC2-hosted Spring Boot backend with a React app, creating a cohesive and responsive user interface. The React app is then deployed on an AWS S3 bucket, providing a scalable and efficient storage solution. This multi-layered integration ensures the EMS operates as a unified, secure, and accessible system for streamlined HR management.

This abstract encapsulates the progressive development stages, emphasizing the project's commitment to leveraging cloud technology for optimal performance, security, and scalability within the realm of HR management.

1. INTRODUCTION

The Employee Management System (EMS) is a transformative solution seamlessly integrated into Amazon Web Services (AWS). Leveraging AWS components like Amazon RDS, EC2 instances, and S3 buckets, the project initiates with creating a database in Amazon RDS and deploying a Spring Boot backend on an EC2 instance. This integration concludes by linking the EC2-hosted Spring Boot backend to a React app, deployed on an AWS S3 bucket. With a focus on efficiency, security, and scalability, the EMS centralizes HR tasks, utilizing cutting-edge cloud technologies to meet modern organizational needs.

Moreover, the EMS extends beyond traditional HR boundaries, leveraging AWS cloud services for precision in database management. With the React app hosted on an S3 bucket, the EMS delivers an intuitive user interface, facilitating effective employee data interaction. This project signifies a technological leap in HR operations, demonstrating a commitment to a unified, accessible, and forward-thinking system that adapts to the evolving landscape of organizational management.

2. OBJECTIVES

The Employee Management System (EMS) project has comprehensive objectives geared towards transforming HR operations and optimizing organizational efficiency. One primary focus is establishing a centralized hub for HR tasks through a robust database within Amazon RDS. This emphasizes optimizing data management, ensuring the confidentiality, integrity, and accessibility of employee information.

Another key objective is leveraging EC2 instances for a scalable Spring Boot backend, creating a resilient infrastructure to handle evolving organizational needs. This includes seamlessly managing fluctuations in user loads and data volumes, ensuring peak performance and reliability, aligning with the dynamic nature of modern organizations.

The EMS project also aims to enhance user interaction by integrating a React app, focusing on a user-friendly interface for intuitive employee data interaction. Deploying this app on an S3 bucket contributes to scalable and efficient storage, ensuring a responsive user experience. These objectives collectively position the EMS as a forward-thinking solution centralizing HR tasks while adapting to the contemporary landscape of organizational management.

3. THEORY

3.1 Cloud Computing:

Cloud computing is a transformative paradigm in computing, redefining how services are delivered by leveraging the internet. Unlike traditional reliance on local servers or personal devices, cloud computing allows users to access computing resources remotely through a network connection. This on-demand model provides scalability, flexibility, and cost efficiency, granting individuals and businesses access to a diverse range of services without the need for extensive local infrastructure. The pay-as-you-go model ensures that users only pay for the resources they consume, fostering collaboration and empowering users to interact with applications from virtually any device with an internet connection.

In the context of the Employee Management System (EMS), our project aligns with the principles of cloud computing, representing a modern computing paradigm. Operating seamlessly over the internet, the EMS prioritizes enhanced scalability, flexibility, and cost efficiency. The EMS project underscores the transformative role of cloud computing services in redefining how HR applications are developed, deployed, and accessed, emphasizing the paradigm shift toward efficient, scalable, and accessible HR management solutions.

Within the EMS project, the pay-as-you-go model inherent in cloud computing becomes particularly significant. This approach allows users to pay only for the resources they consume, aligning with the cost-effective nature of cloud-based services. Security measures implemented in cloud computing ensure the protection of sensitive employee information, emphasizing data integrity and confidentiality. The EMS project, grounded in the principles of cloud computing, places a strong emphasis on efficiency, flexibility, and accessibility for businesses and individuals alike.

Cloud Computing has left an indelible mark on various technologies, and within the EMS project, its principles are harnessed to redefine HR operations. Operating seamlessly over the internet, the project prioritizes scalability, flexibility, and cost efficiency, showcasing the transformative potential of cloud computing in HR management.

3.2 Amazon Web Services (AWS)

Amazon Web Services (AWS) is a leading cloud computing platform that provides a comprehensive suite of on-demand services. Renowned for its scalability, reliability, and cost-effectiveness, AWS empowers businesses and individuals to access a wide array of computing resources without the need for upfront investments in physical infrastructure. From computing power and storage to databases, machine learning, and more, AWS offers a versatile range of services designed to meet the diverse needs of users worldwide. The platform's global infrastructure spans regions and availability zones, ensuring high availability and fault tolerance. With a commitment to security and innovation, AWS has become a cornerstone in the cloud computing landscape, enabling organizations to scale, innovate, and drive efficiency in the digital era.

3.3 Amazon EC2

An Amazon EC2 instance is a virtual server within the Amazon Elastic Compute Cloud (EC2) service, offering scalable and flexible computing resources in the cloud. Users can choose from a variety of instance types tailored to specific workloads, such as compute-optimized, memory-optimized, or storage-optimized configurations. Each EC2 instance is equipped with its own operating system and customizable configurations, allowing users to deploy applications seamlessly. With the ability to easily scale instances EC2 instances provide agility and cost-effectiveness, making them a cornerstone of cloud computing for running diverse workloads efficiently and reliably.

3.4 Amazon S3

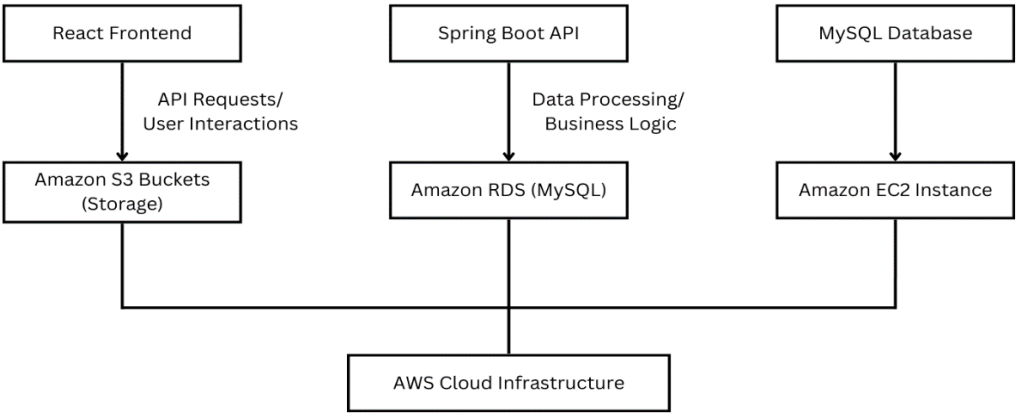
Amazon S3 (Simple Storage Service) is a highly scalable object storage service provided by AWS, allowing users to store and retrieve data, known as objects, in the cloud. It offers virtually unlimited storage capacity and is designed for high durability and availability by storing copies of data across multiple geographically dispersed data centers. S3 provides robust security features, including access control through bucket policies, ACLs, and IAM roles, as well as the option for data encryption in transit and at rest. Each S3 bucket must have a globally unique name, following DNS-compliant conventions. S3 is widely used for various applications,

such as hosting static websites, storing backup data, and serving as a reliable and cost-effective solution for scalable object storage in the cloud.

Our project seamlessly integrates with Amazon Web Services (AWS), specifically leveraging Amazon Elastic Compute Cloud (EC2) and Simple Storage Service (S3) to host, deploy, and store data for the developed Employee Management System (EMS). By utilizing EC2 instances, our project benefits from scalable and reliable computing capacity, optimizing resource utilization and ensuring efficient performance, even during varying workloads. The cloud-based architecture provided by AWS enhances user engagement and accessibility, allowing the EMS to be accessed from anywhere with an internet connection.

AWS and EC2, along with S3, play a crucial role in the hosting and storage infrastructure of our EMS project, providing a secure, flexible, and cost-effective environment. The pay-as-you-go model of AWS ensures cost efficiency, as we only pay for the computing resources and storage consumed, eliminating the need for substantial upfront investments. In essence, our project's integration with AWS, EC2, and S3 represents a modern approach to web development, showcasing the potential of cloud services to revolutionize traditional hosting and data storage models. This collaboration enhances the scalability, reliability, and cost-effectiveness of our EMS platform, aligning with the core tenets of cloud computing.

4. BLOCK DIAGRAM:



5. MODULE DESCRIPTION

5.1 Setting up RDS(AWS):

Setting up Amazon Relational Database Service (RDS) in AWS involves a streamlined process for creating and managing relational databases. Users can easily launch an RDS instance through the AWS Management Console, specifying database engine, version, and other configuration details. RDS supports various database engines such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server, providing flexibility to cater to different application needs. Once the RDS instance is launched, AWS automates routine database tasks like backups, software patching, and scaling to ensure high availability and performance. Users can also configure security settings, such as defining access control through security groups and implementing encryption for data at rest and in transit. With RDS, the complexities of database administration are abstracted, allowing users to focus on application development while AWS manages the operational aspects of the relational database.

5.2 Connecting RDS with Backend- Spring Boot:

Connecting Amazon RDS, utilizing the MySQL engine, with a Spring Boot backend involves configuring the RDS instance with the appropriate security groups and obtaining connection details. These details, including the database endpoint, port, and credentials, must be accurately incorporated into the Spring Boot application's data source configuration. Leveraging frameworks like Spring Data JPA or Hibernate, developers specify the connection parameters within the application's properties, enabling seamless interaction with the MySQL database. Connection pooling can be implemented to optimize performance. With this setup, the Spring Boot backend establishes a connection to the RDS-hosted MySQL database, allowing for efficient data management and retrieval. Regular testing and monitoring ensure the reliability and performance of the database connection, simplifying the overall administration of the system. This integration streamlines database management, allowing developers to focus on building robust applications while AWS handles the intricacies of database administration.

5.3 Creating EC2 Instance

Creating an EC2 (Elastic Compute Cloud) instance in AWS involves accessing the AWS Management Console, navigating to the EC2 dashboard, and selecting the "Launch Instance" option. Users then choose an Amazon Machine Image (AMI), which represents the operating system and software for the instance. Next, users configure the instance details, specifying the instance type, number of instances, and other settings. Additional configurations, such as adding storage through Elastic Block Store (EBS) volumes, setting security groups, and defining key pairs for secure access, are crucial for customizing the instance. Once configured, users review the settings, launch the instance, and monitor its status through the console. This process allows for the quick deployment of virtual servers tailored to specific application requirements.

5.4 Connecting EC2 with Backend-Spring Boot

Connecting an EC2 instance with a Spring Boot backend involves configuring the EC2 environment and integrating it with the Spring Boot application. Initially, users must ensure that the EC2 instance is provisioned with the necessary resources, including the desired operating system and any required software dependencies. Subsequently, developers configure the security groups to allow inbound traffic on the required ports for communication with the Spring Boot backend. In the Spring Boot application, the connection details such as the EC2 instance's public IP address or DNS, port number, and any relevant credentials are configured in the application's properties or configuration files. Once the configurations are in place, the Spring Boot backend can establish a connection with the EC2 instance, enabling seamless interaction between the application and the virtual server. This connection allows the backend to serve requests, process data, and execute tasks on the EC2 instance, providing a scalable and flexible environment for hosting and running Spring Boot applications.

5.5 Integrating Frontend with RDS

Integrating the frontend with Amazon RDS involves establishing a connection between the user interface and the database through the backend. User interactions on the frontend trigger requests sent to the backend, which then communicates with the RDS database to fetch or update data. The backend is configured to

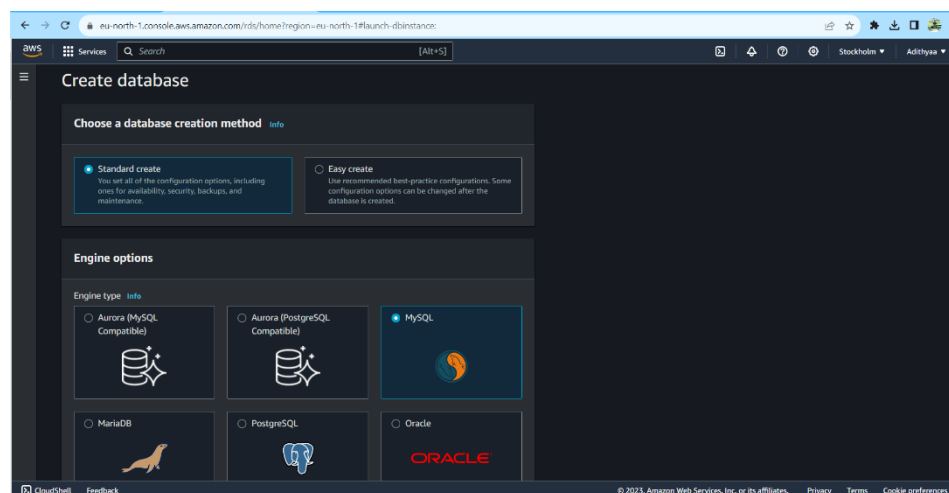
handle these requests, utilizing secure connection details such as the RDS endpoint and database credentials. Security measures, including HTTPS and access controls, are implemented to protect sensitive data during this integration. Optimizing data transfer and minimizing unnecessary queries contribute to the efficiency of the overall integration, ensuring a seamless and secure flow of information between the frontend and the RDS database.

5.6 Deployment of System using Amazon S3:

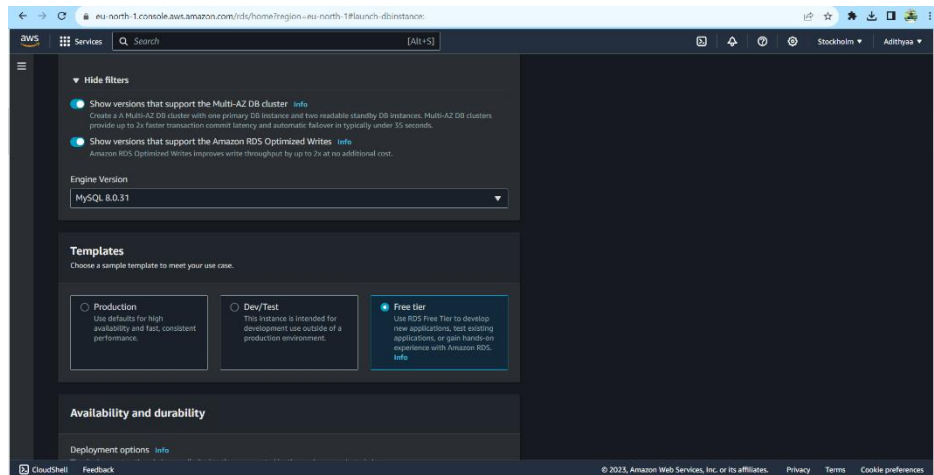
Deploying a system using Amazon S3 involves leveraging the S3 bucket as a scalable and reliable storage solution for hosting application artifacts, static files, or any content constituting the deployed system. First, the necessary files, such as application code, assets, or configurations, are uploaded to the designated S3 bucket. This bucket is configured for static website hosting, allowing it to serve web content directly. Users can enable versioning for the bucket to track changes over time, providing a level of version control. Once the files are stored in the S3 bucket, it generates a unique endpoint URL. This URL becomes the access point for the deployed system, facilitating easy distribution and access. Utilizing Amazon S3 for deployment ensures high availability, scalability, and cost-effectiveness, making it an integral part of modern cloud-based deployment strategies.

6. IMPLEMENTATION

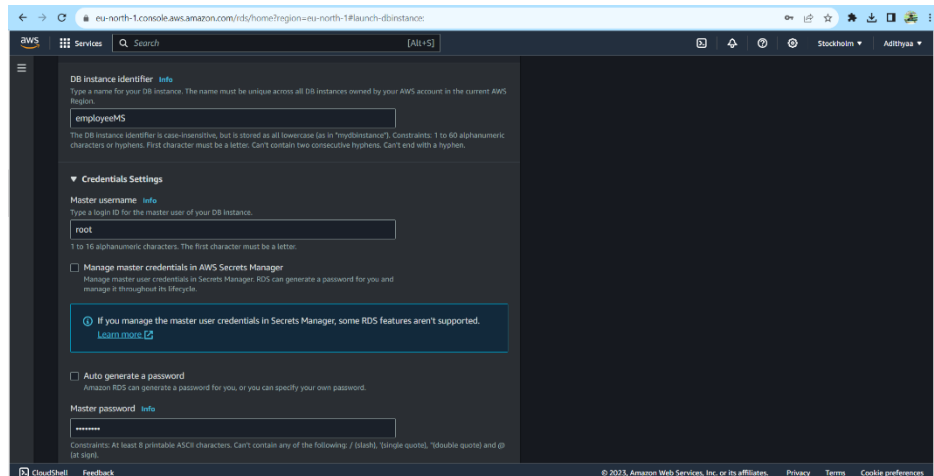
6.1 Creating Database on RDS(AWS)



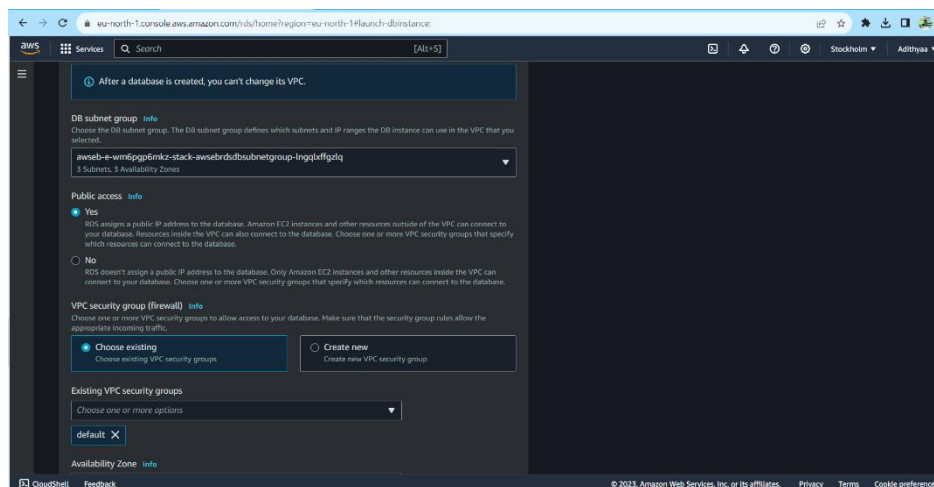
6.1.1 Choosing Engine type



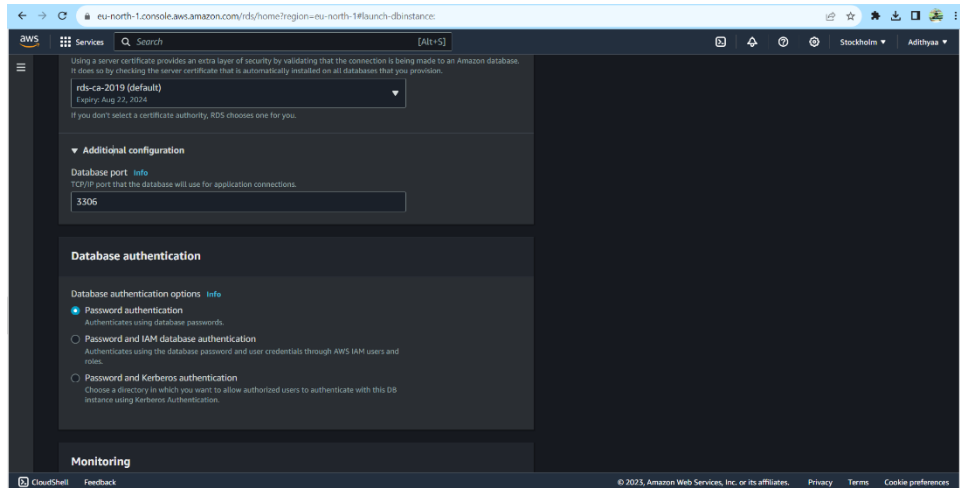
6.1.2 Choosing appropriate Engine version



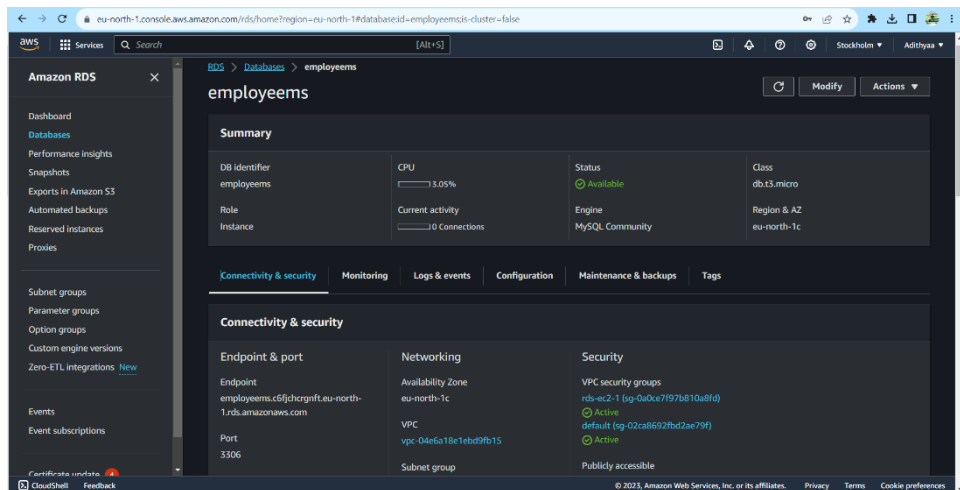
6.1.3 Set Credentials for Database



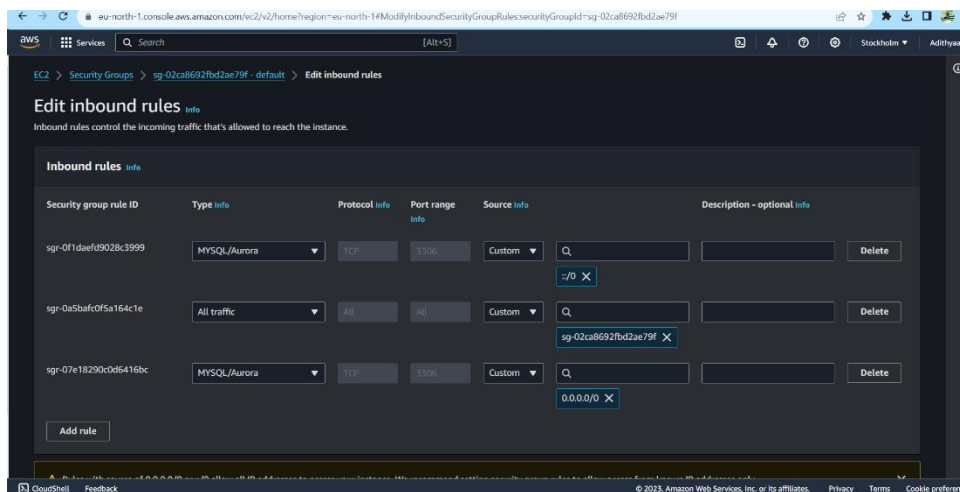
6.1.4 Enable Public Access



6.1.5 Specify Database Port

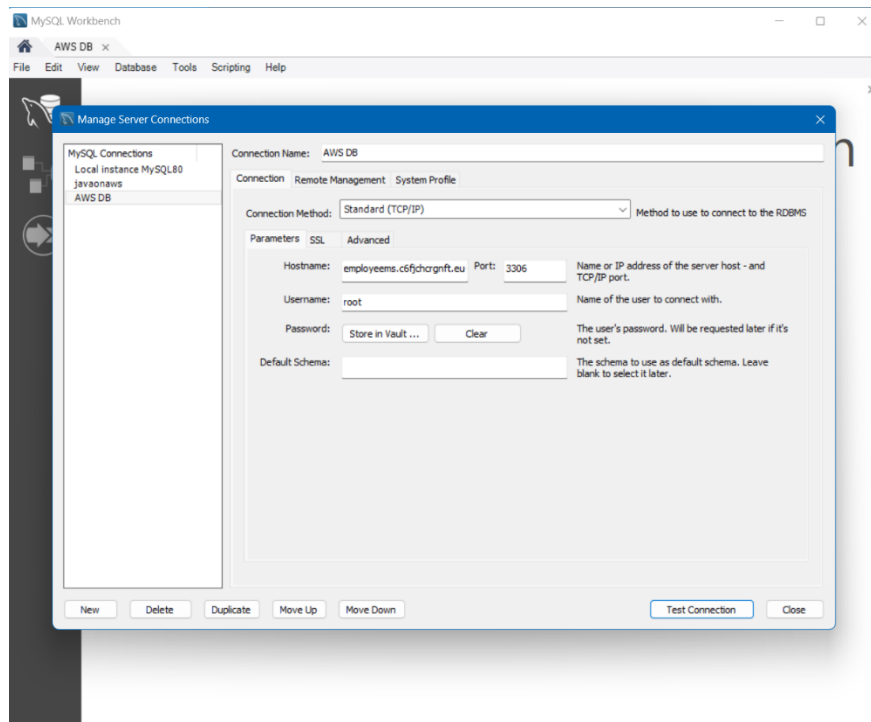


6.1.6 RDS Database Created

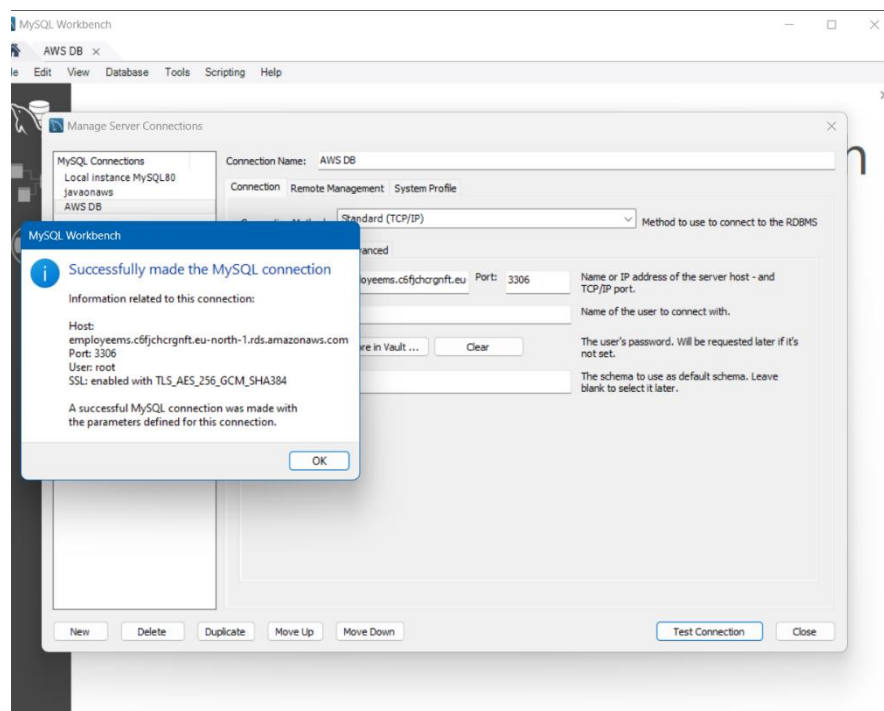


6.1.7 Edit Inbound rules

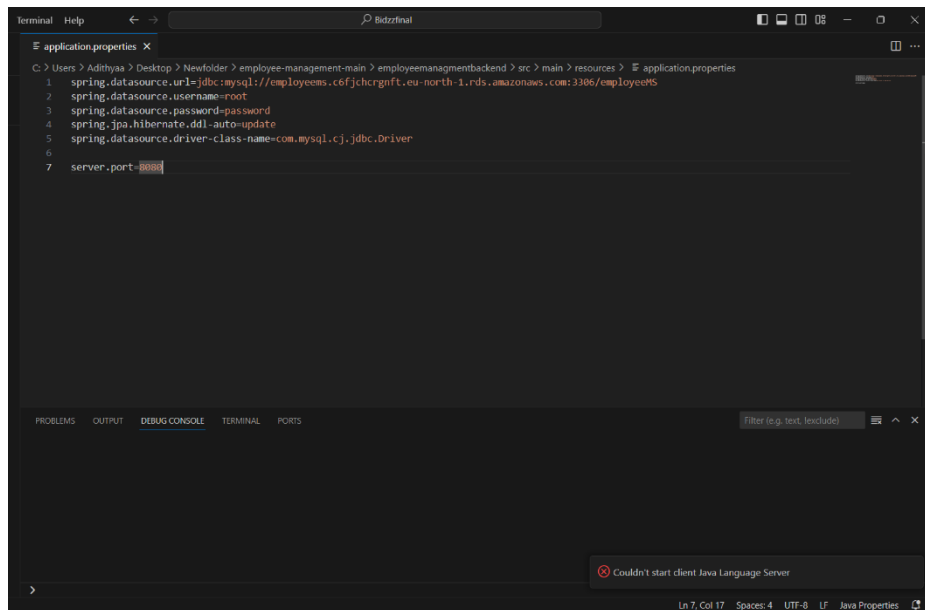
6.2 Connecting RDS with Backend- Spring Boot



6.2.1 Setting up new connection in MySQL



6.2.2 Testing Connection

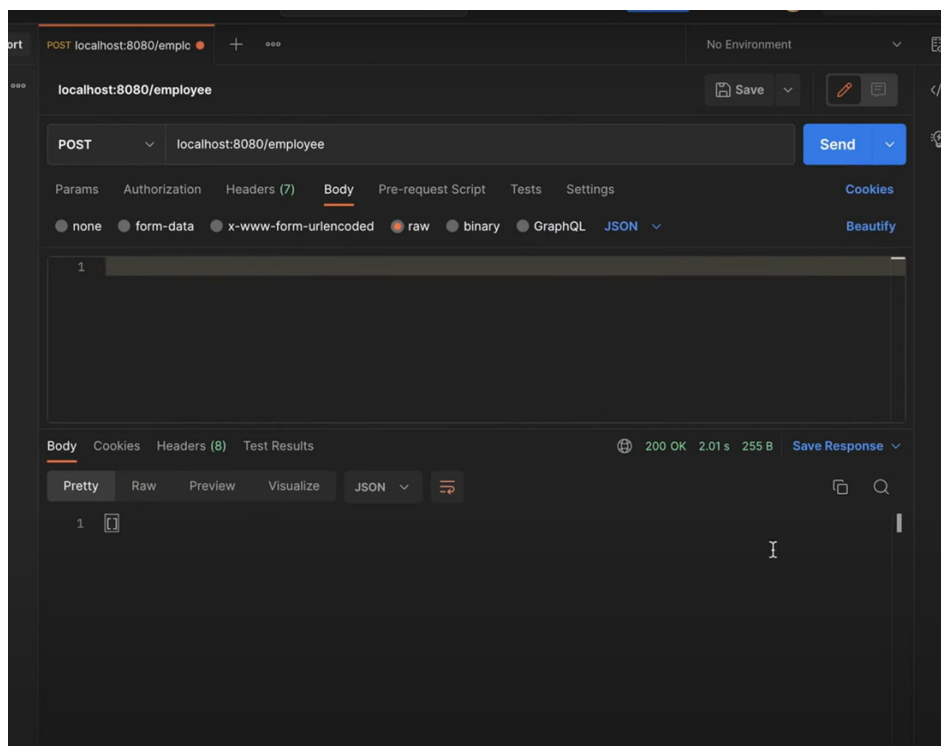


The screenshot shows an IDE window with a file named `application.properties` open. The file contains the following configuration:

```
1 spring.datasource.url=jdbc:mysql://employeeems.c6fjchcrgnft.eu-north-1.rds.amazonaws.com:3306/employeeMS
2 spring.datasource.username=root
3 spring.datasource.password=password
4 spring.jpa.hibernate.ddl-auto=update
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
6
7 server.port=8080
```

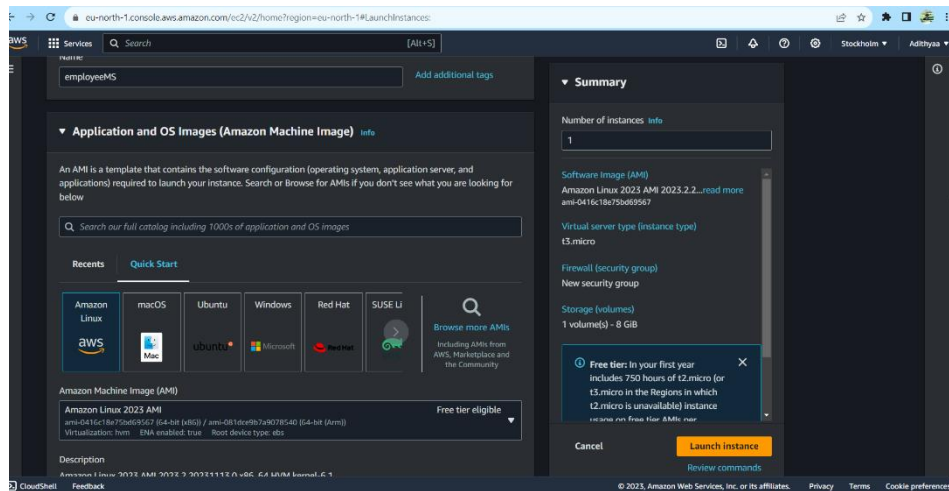
At the bottom of the IDE, a status bar indicates "Couldn't start client Java Language Server".

6.2.3 Changing application.properties

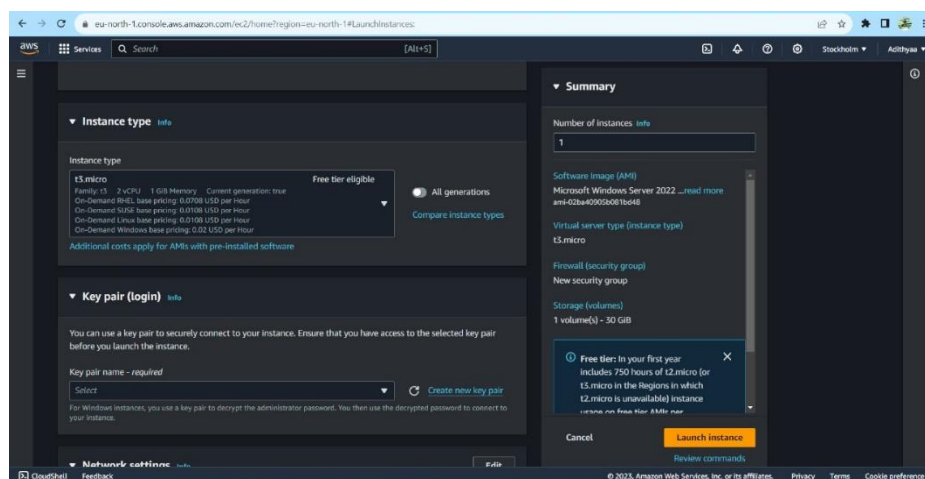


6.2.4 Successfully connecting RDS to Spring Boot

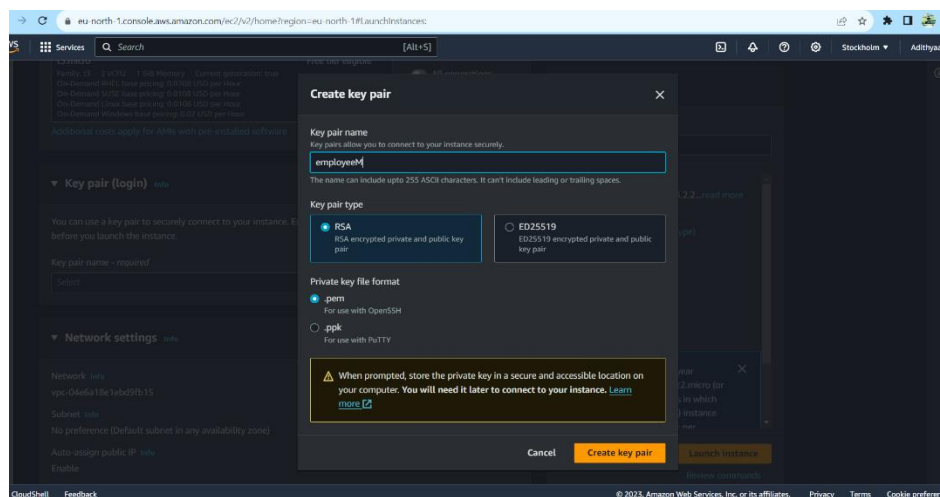
6.3 Creating EC2 Instance



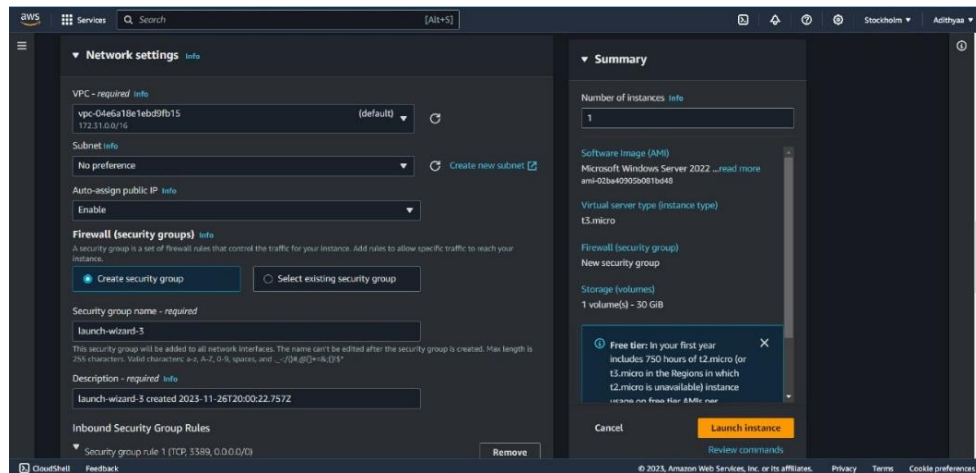
6.3.1 Choosing an AMI



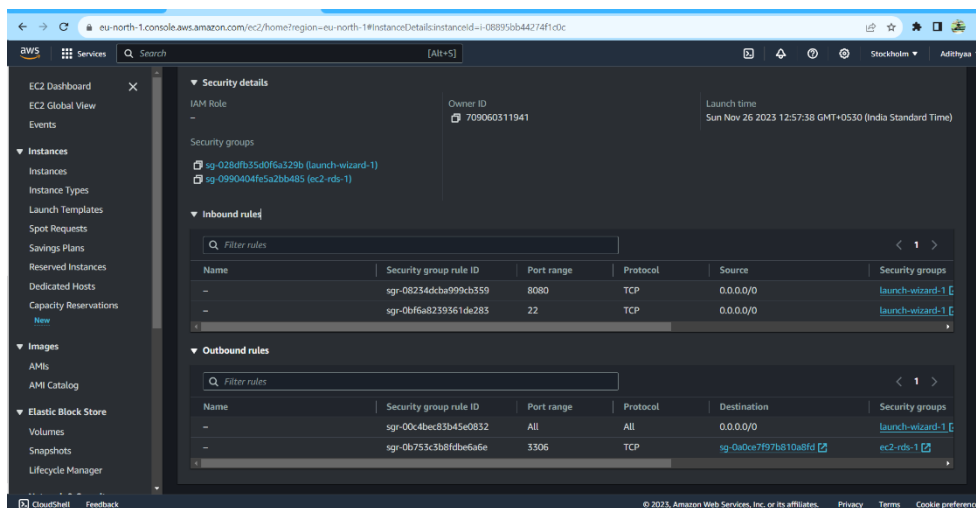
6.3.2 Choosing an Instance Type



6.3.3 Creating key-value pair

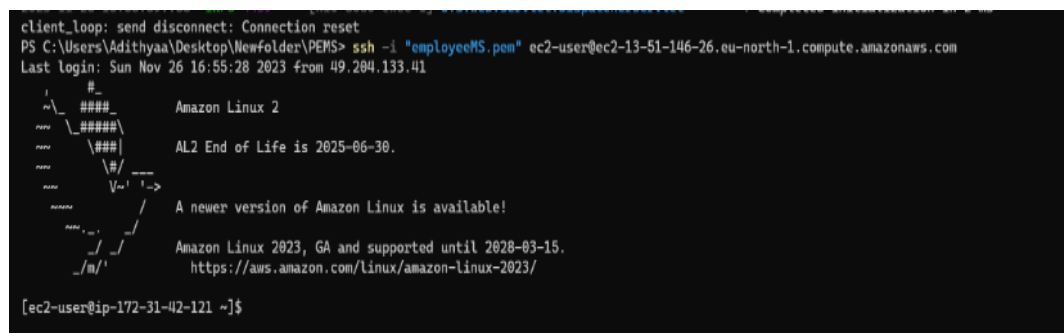


6.3.4 Network Settings

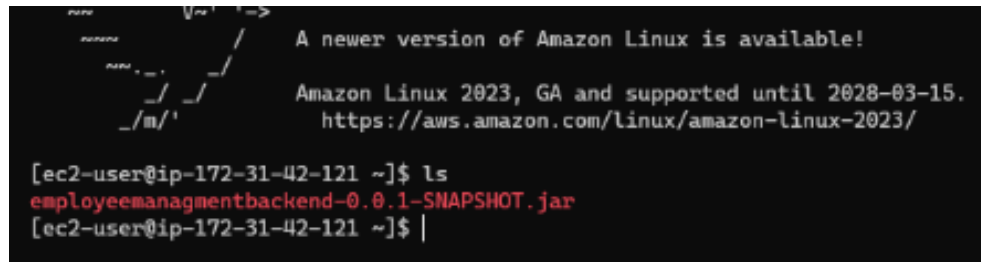


6.3.5 Modify Inbound/Outbound Rules

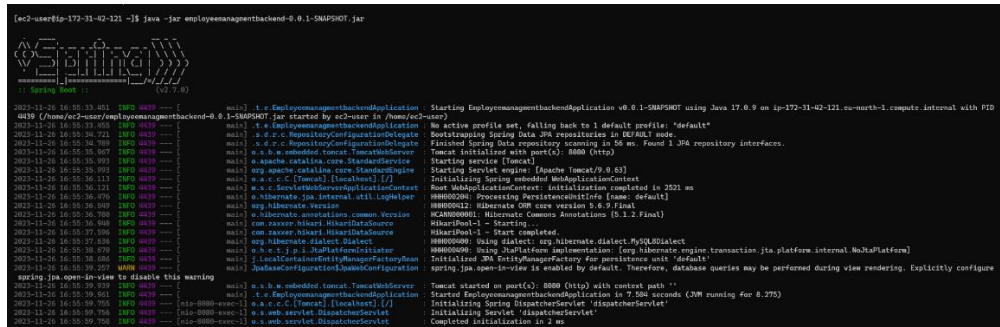
6.4 Connecting EC2 with Backend-Spring-Boot



6.4.1 Connecting to EC2 from local machine

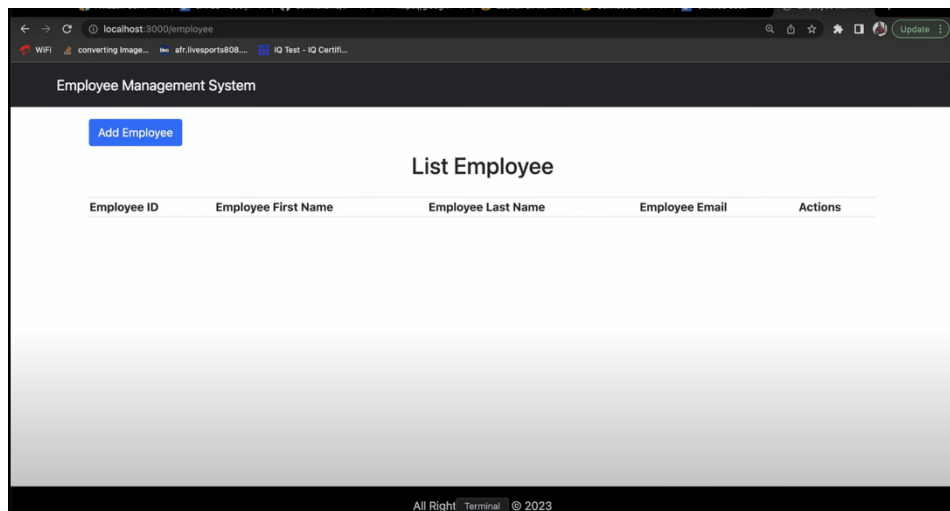


6.4.2 Merging jar file into server

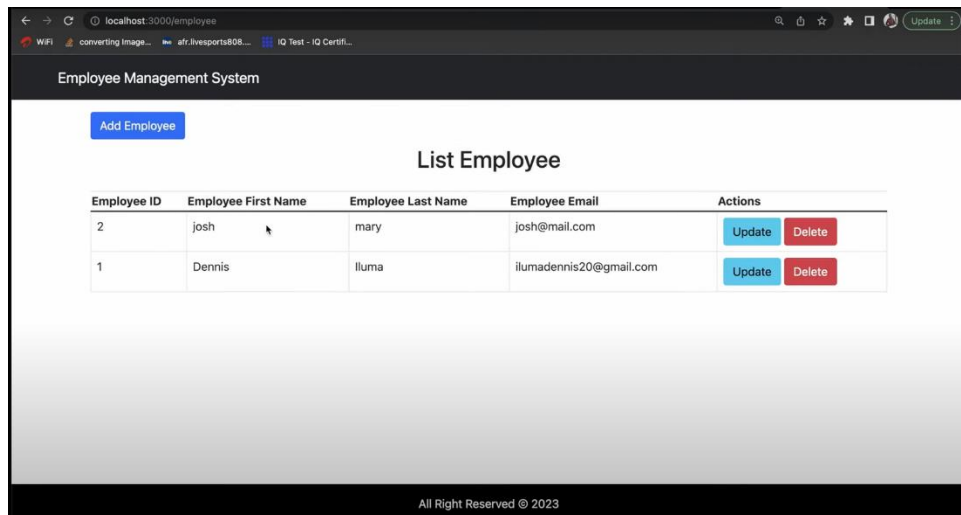


6.4.3 Executing the jar

6.5 Integrating Front-End with Cloud

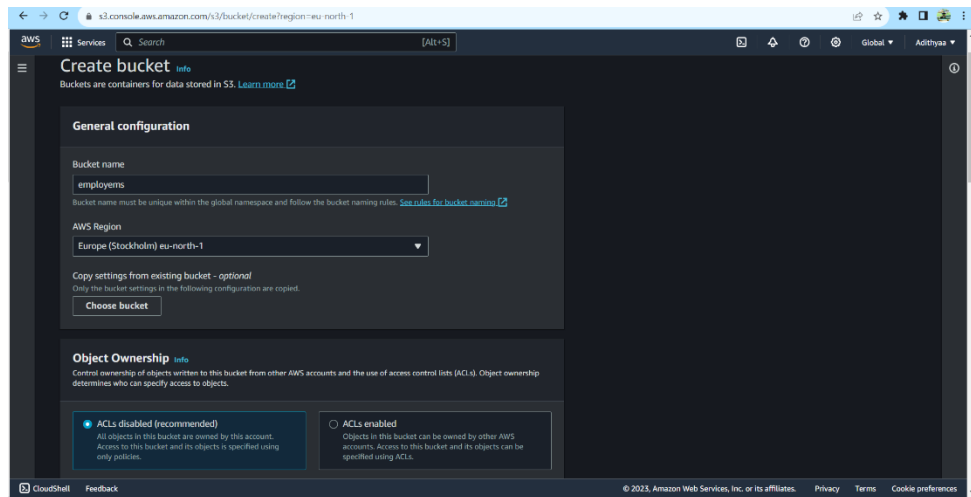


6.5.1 React.js Employee page before connecting to RDS

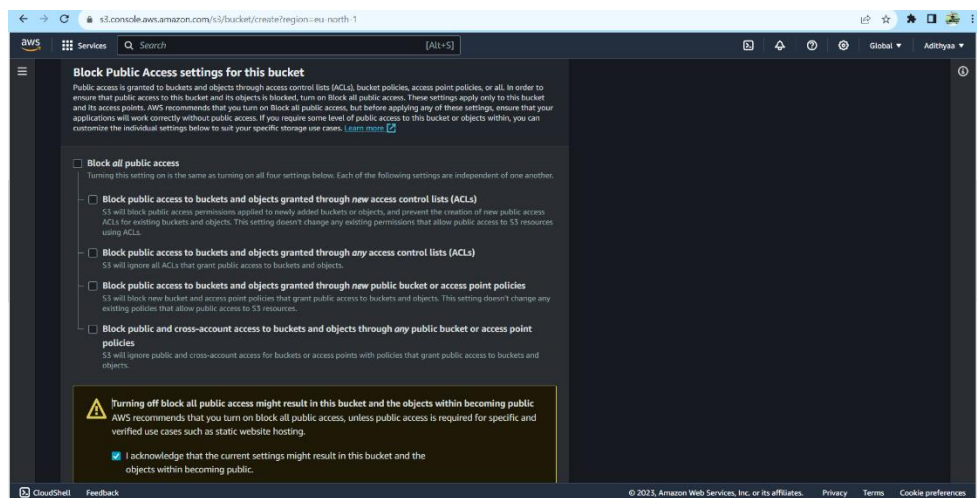


6.5.2 React.js Employee page after connecting to RDS

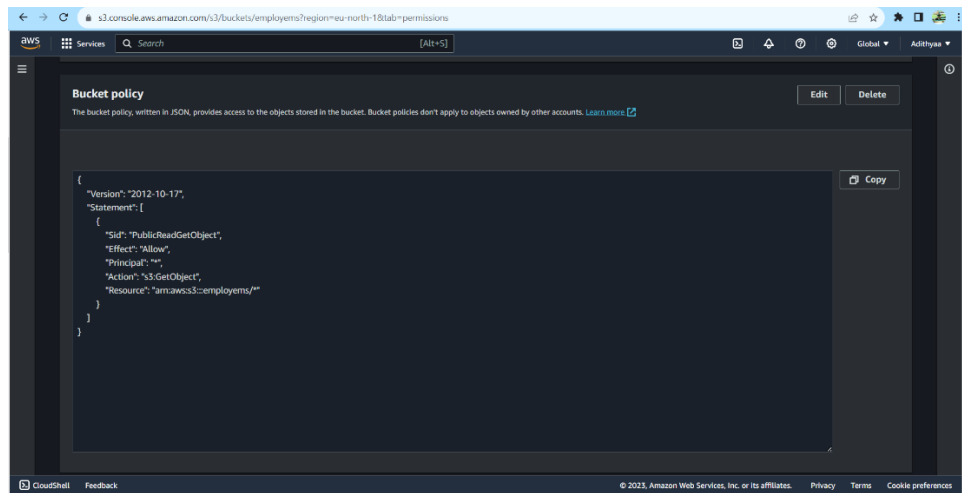
6.6 Deploying React app using amazon S3



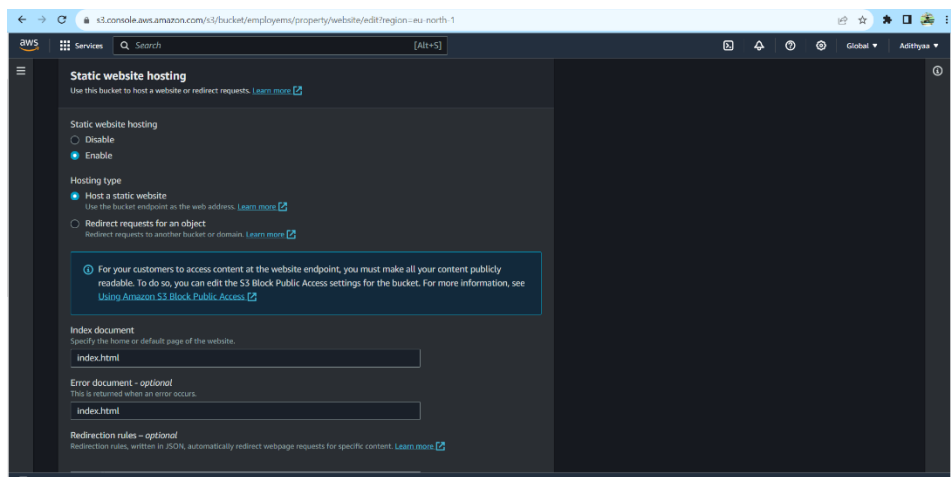
6.6.1 Creating S3 bucket



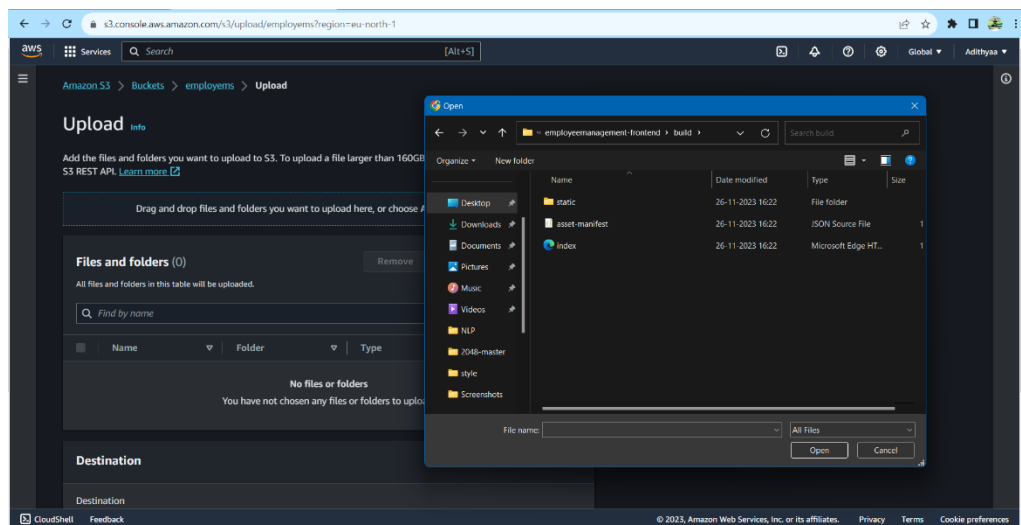
6.6.2 Enable public access



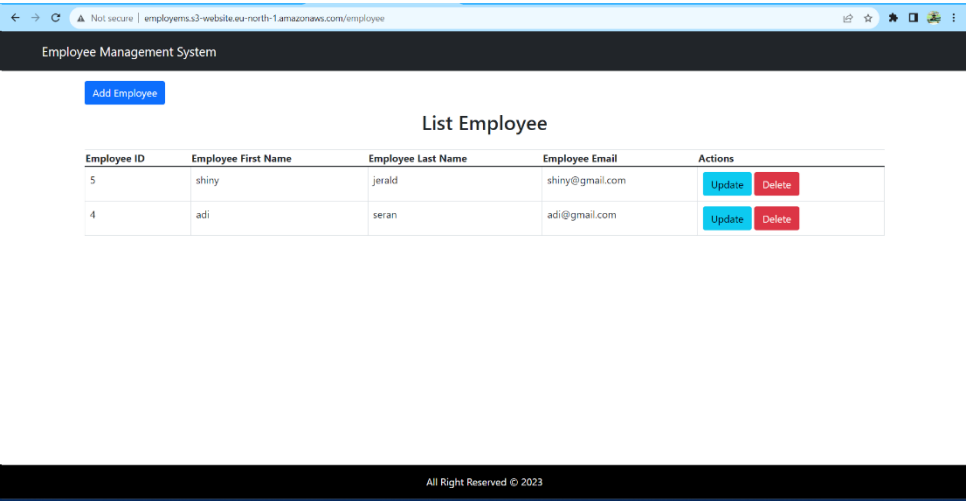
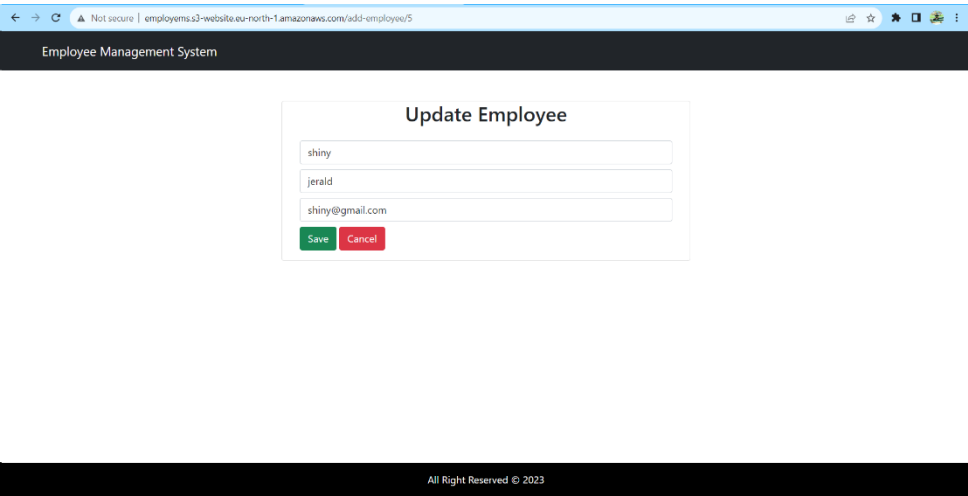
6.6.3 Configure Bucket policy



6.6.4 Enable hosting static website



6.6.5 Upload the build folder creating after running “npm start build”



6.6.6 Deploying system in Cloud

7. APPLICATIONS

The Employee Management System (EMS) developed with CRUD operations offers versatile applications across diverse organizational functions. Primarily, it streamlines HR operations by providing a centralized platform for managing employee data. HR professionals can efficiently perform Create, Read, Update, and Delete operations, ensuring accurate and up-to-date records of employee information. Additionally, the system enhances the employee experience by facilitating self-service functionalities, allowing staff to update personal details, submit leave requests, and access relevant HR information in real-time. Beyond HR, the EMS finds applications in payroll management, simplifying salary calculations and automating payment processes based on the updated employee records.

Furthermore, the system's CRUD capabilities extend to performance management, enabling supervisors to input and update employee performance data seamlessly. This aids in conducting performance reviews, setting goals, and tracking employee achievements. The system's ability to store historical data supports analytics and reporting, allowing organizations to gain insights into employee trends, identify areas for improvement, and make informed decisions. In essence, the EMS with CRUD operations becomes a comprehensive tool for optimizing HR functions, fostering employee engagement, and contributing to data-driven decision-making across various organizational facets.

Moreover, the EMS contributes to organizational transparency and communication. With the ability to update and retrieve employee data in real-time, the system supports seamless internal communication. Managers and employees can access relevant information promptly, fostering a collaborative and informed work environment. Additionally, the system aids in resource allocation by providing insights into workforce distribution, allowing organizations to strategically assign tasks and responsibilities based on employee skills and availability.

In summary, the EMS with CRUD operations delivers a holistic solution that extends beyond basic HR functions. Its applications span compliance management, ensuring legal adherence, and fostering transparent communication within the organization. This multifaceted approach positions the system as a pivotal tool for enhancing efficiency, compliance, and communication across various organizational dimensions.

8. CONCLUSION:

In conclusion, the development and implementation of the Employee Management System (EMS) represent a significant milestone in modernizing and optimizing organizational processes. The system's robust CRUD operations empower HR professionals with a versatile toolset, streamlining employee data management from recruitment to retirement. By facilitating Create, Read, Update, and Delete functionalities, the EMS not only centralizes HR operations but also enhances accuracy and efficiency, laying the foundation for informed decision-making.

The EMS's applications extend beyond traditional HR boundaries, offering a comprehensive solution for compliance management and transparent internal communication. Its capacity to track certifications, licenses, and training records ensures organizations remain compliant with evolving legal and regulatory frameworks. Simultaneously, the system promotes transparency by providing real-time access to employee data, fostering collaborative communication and informed decision-making across the organization.

Looking ahead, the EMS stands as a catalyst for organizational efficiency, data-driven insights, and employee engagement. Its successful integration of CRUD operations, compliance management, and communication functionalities positions the system as an invaluable asset for businesses navigating the complexities of modern workforce management. As organizations evolve, the EMS provides a scalable and adaptable solution, ensuring it remains a cornerstone in fostering streamlined HR processes and contributing to overall operational excellence.