Healthcare Management System Using Cloud Computing

Big Data Infrastructure & Cloud Computing

Report Submitted By

Anandhu Krishna

Bhargav Ravi

Hardik Kolhe

Sivaprasad Puthumadthil Rameshan Nair

Sravanya Nediyedath Arun

Under the guidance of

Dr. Badre Bousalem

EPITA - School of Engineering and Computer Science

Master of Science MSc

FALL 2023

ABSTRACT

This report outlines the design and implementation of a cloud-based Healthcare Management System on Amazon Web Services (AWS). The system features a hybrid cloud model with a user interface and backend layer of microservices, providing high scalability, security, and cost-effectiveness. Utilizing AWS services such as Amazon RDS and Amazon S3, the system achieves high availability and durability while managing patient data and healthcare services. The architecture is divided into public and private subnets, ensuring secure and efficient data management. Auto-scaling groups and load balancers enable dynamic adjustment to demand, ensuring high availability and performance. The successful implementation of the application, microservices, and database layers demonstrates the system's capability to handle business logic and core functions effectively. This robust solution addresses healthcare organizations' challenges, delivering high-quality healthcare services efficiently and reliably.

CONTENTS

LIST OF FIGURE

1. INTRODUCTION

Our web application is designed with a cloud-based architecture and deployed on the robust infrastructure provided by Amazon Web Services (AWS). This architecture consists of a user interface and a backend layer, the latter of which is deployed as microservices. This approach allows for efficient, flexible, and agile development and deployment processes. The application manages a variety of functions such as user Registration, Login, Administrative tasks, Patient Registration, and Doctor Management, catering to a broad range of healthcare system requirements.

By leveraging the comprehensive suite of AWS services, we ensure the application is highly scalable, secure, and cost-effective. This includes taking advantage of AWS's advanced capabilities such as auto-scaling, serverless computing, and various security services to monitor and enhance the application's performance. Our focus is on delivering a seamless, reliable, and efficient user experience while optimizing resource utilization and maintaining robust data security and privacy standards.

2. CLOUD INFRASTRUCTURE

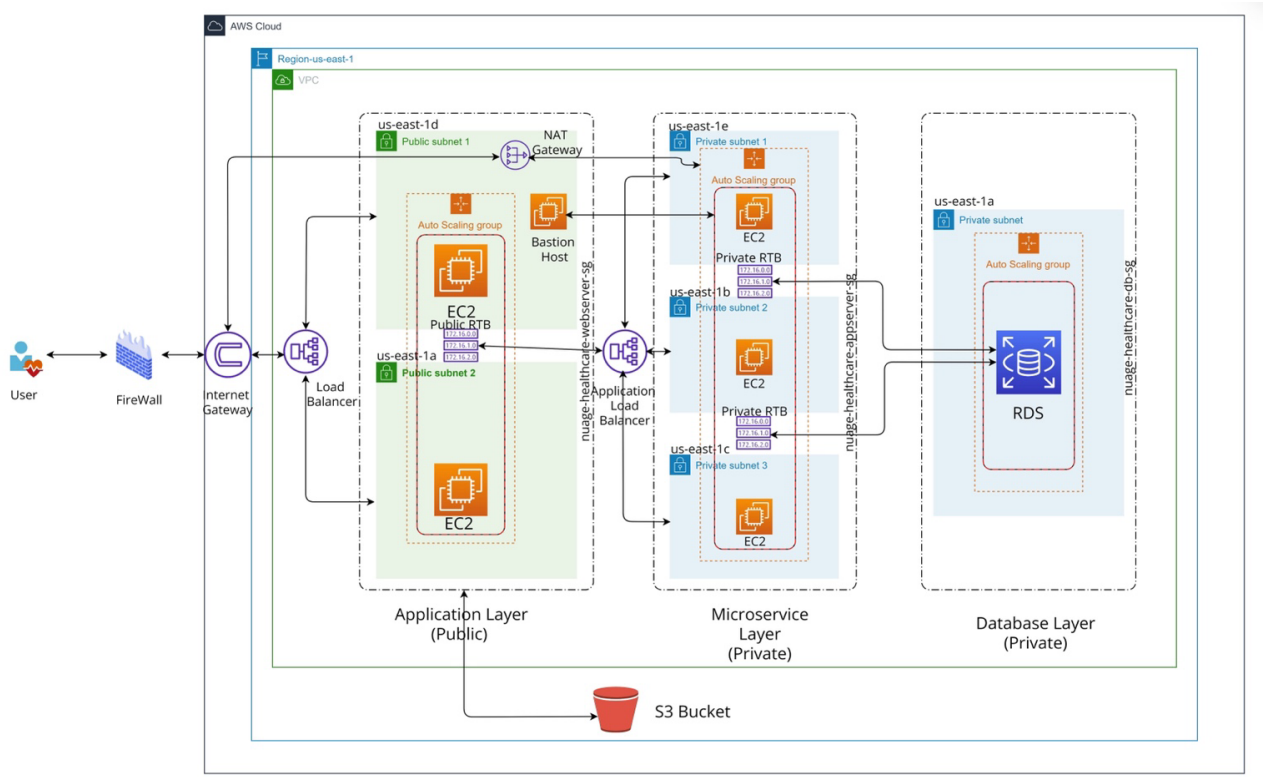
1. **Hybrid Cloud Approach:**
   - **Approach:** We have adopted a hybrid cloud model, merging Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) to craft our application architecture. This blend allows us to tailor our infrastructure for optimal performance.
   - **Benefits:** This approach offers us remarkable flexibility, scalability, security, and cost-effectiveness to meet our application's requirements.
2. **Integration with AWS:**
   - **Amazon RDS:** We leverage Amazon Relational Database Service (RDS) to efficiently manage our databases, ensuring our architecture remains scalable, highly available, and durable.
   - **Amazon S3:** Our architecture benefits from utilizing Amazon S3 buckets for storing static files and other data, achieving high durability, availability, and security.
3. **Application Layer (Public):**

   Our application layer is situated in a public subnet, encompassing components such as the user interface, firewalls, and internet gateways. It serves as the point of access for users engaging with our healthcare system application.
4. **Microservices Layer (Private):**

   We deploy the microservices layer in a private subnet to house the application's core functionalities. This layer spans multiple microservices hosted on EC2 instances, where it efficiently handles business logic and core application functions, facilitating communication between the application and database layers.
5. **Database Layer (Private):**

   Our database layer resides in a private subnet, providing enhanced security and isolation. We use Amazon RDS for database management to expertly handle application data. This layer manages and stores data securely, remaining isolated from the public internet.

3. ARCHITECTURE DIAGRAM



(Fig: 3.1 Architecture Diagram)

4. MICROSERVICES ARCHITECTURE OVERVIEW

A. Scalability

- **Auto Scaling:**

  The architecture leverages autoscaling groups to dynamically adjust the number of EC2 instances based on predefined scaling policies or custom metrics. It provides elasticity by automatically scaling the capacity of the application tier in response to changes in demand. AWS-managed services such as Amazon RDS for the database tier and Amazon S3 for storage are designed to scale seamlessly to accommodate growing data volumes and request rates which simplifies management by offloading the responsibility of scaling infrastructure to AWS. It also adds, removes, or replaces EC2 instances across multiple availability zones based on need or changing demand. It reduces the impact of system failures and improves the availability of our applications.

- **Load Balancer:**

  Elastic Load Balancer automatically distributes incoming traffic across multiple EC2 instances and availability zones. ELB automatically scales its capacity to handle varying levels of traffic, ensuring that the application remains responsive under load spikes. ELB benefits in Improved Performance, Enhanced Availability & Centralized Traffic Management.

B. PERFORMANCE

Virtual Private Cloud (VPC) along with private and public subnets are the components used for organizing and securing the network infrastructure. The VPC templates are used to the integration with AWS services like EC2, Auto Scaling, and Elastic Load Balancing. Public subnets are subnets within the VPC that have a route to the Internet Gateway (IGW), allowing instances within the subnet to have direct access to and from the internet if they have public IP addresses. Private subnets are subnets within the VPC that do not have a route to the Internet Gateway (IGW), making them inaccessible from the internet. Route tables are used to determine where network traffic is

directed within the VPC. Each subnet is associated with a route table that defines the rules for routing traffic. Amazon RDS is used for managing the relational database tier of the architecture.

Amazon S3 is utilized for storing static assets, such as images, videos, or user uploads. ELB is used to distribute incoming traffic across multiple EC2 instances. NAT gateway is created to allow instances in private subnets to access resources outside of the VPC and the internet. We used Bastion hosts to provide a single point of entry into your private network, for reducing the exposure of the private instances to the public internet.

C. SECURITY

Firewalls are deployed in public and private subnets to control inbound and outbound traffic. A bastion host serves as a secure entry point for administrators to access resources in the private subnet. A virtual private cloud (VPC) is a logically isolated section of the AWS cloud where you can launch your own resources in a virtual network that you define. The architecture utilizes both private and public subnets. Public subnets house resources that are accessible over the internet, while private subnets contain resources that are only accessible from within the VPC.

D. COST OPTIMIZATION

- **Auto Scaling Groups:**

  ASGs helped to optimize costs by automatically adjusting the number of EC2 instances in response to the demand pattern of our application which ensured that we were only paying for the resources we needed.

- **Database Optimization:**

  AWS RDS instance was chosen as our database layer which is appropriately sized for our workload.

- **Reserved Instances:**

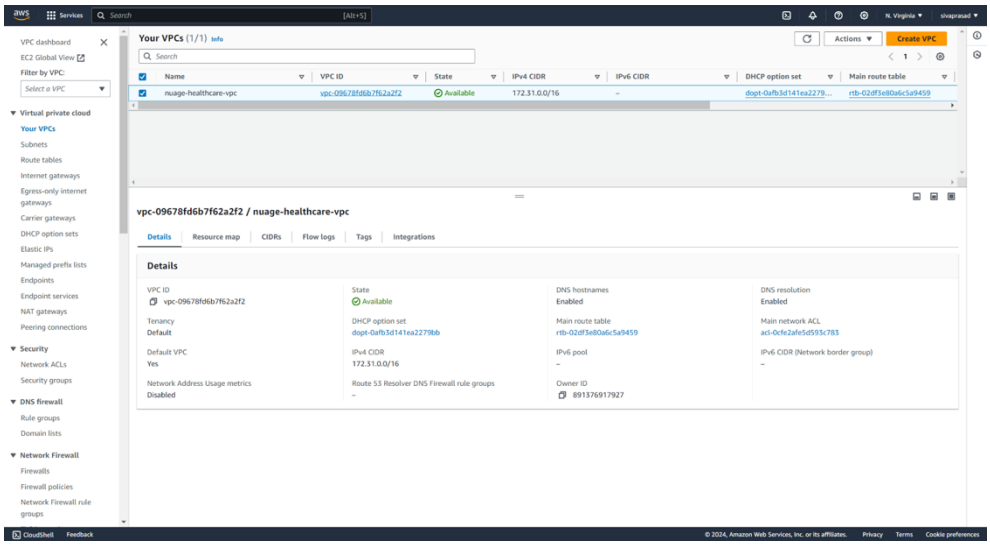
  We have predictable, steady-state workloads, and purchasing Reserved Instances leads us for significant cost savings over On-Demand pricing.

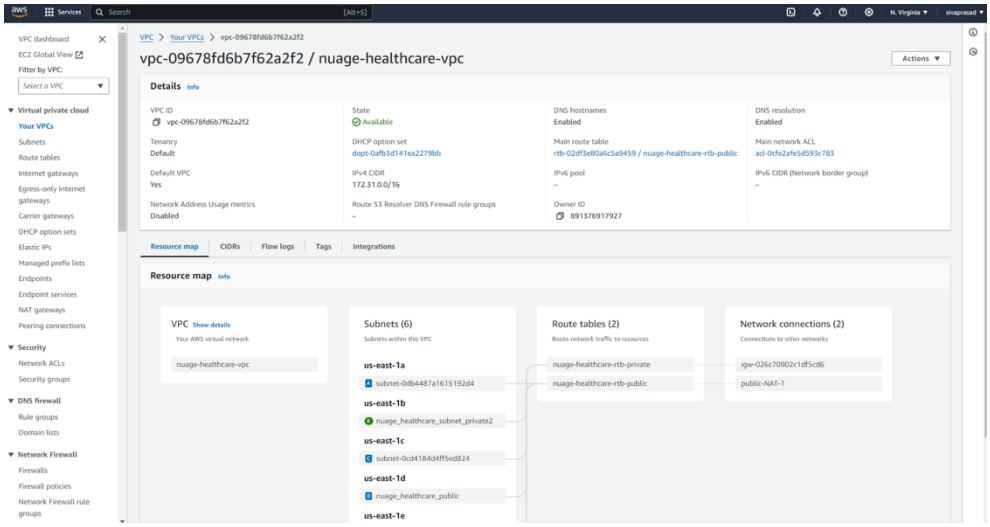
5. IMPLEMENTATION

The implementation of a cloud-based Healthcare Management System that leverages Amazon Web Services (AWS) infrastructure. The implementation process is divided into three key layers: Application, Microservices, and Database.

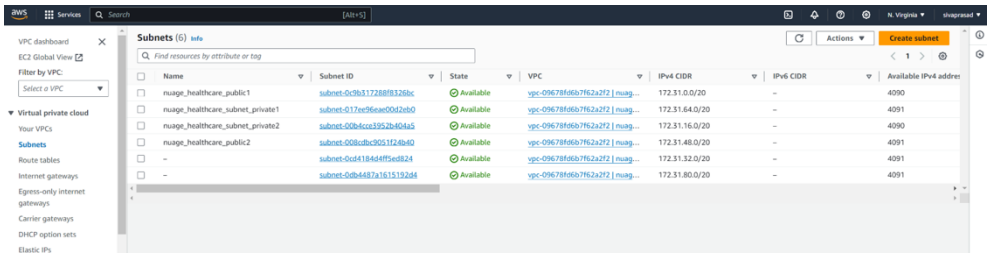
Part A: Setting Up the Application Layer

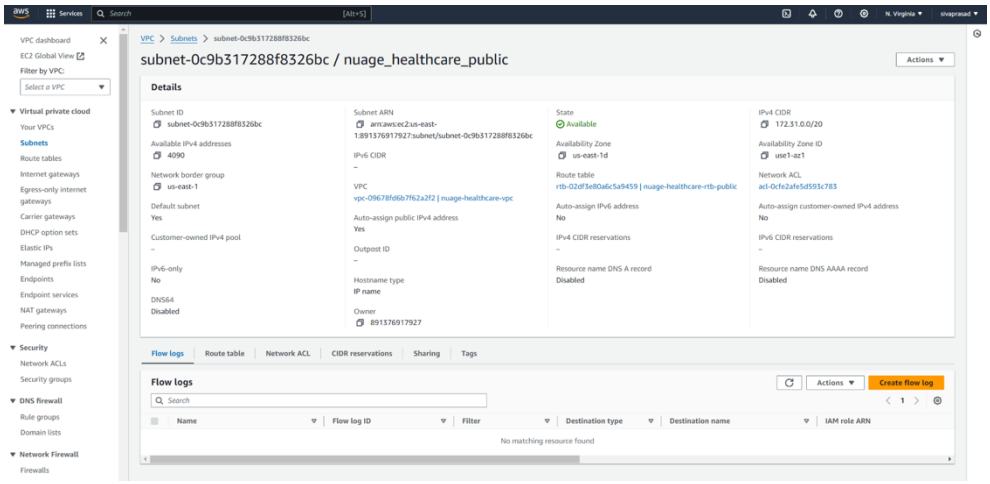
1. **VPC Creation**: A Virtual Private Cloud (VPC) was established to manage the virtual networking environment and provide isolated networking for the application.

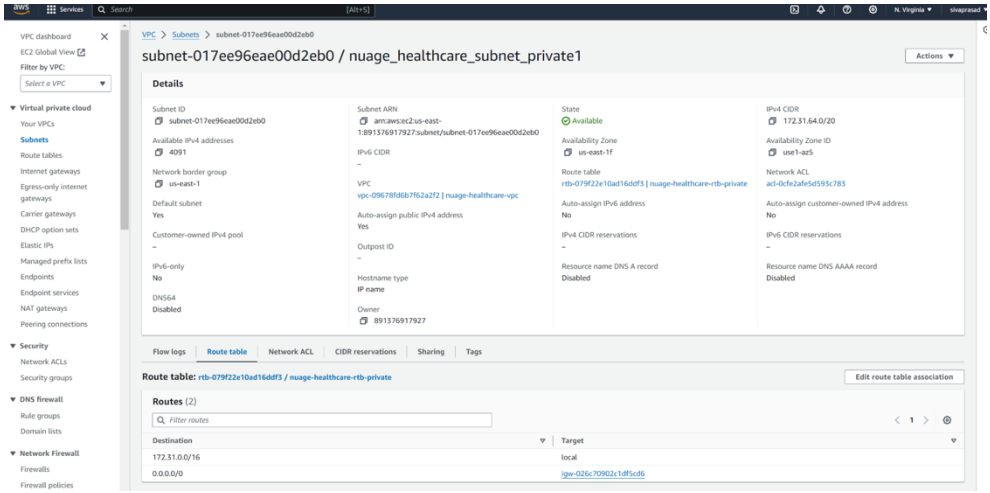




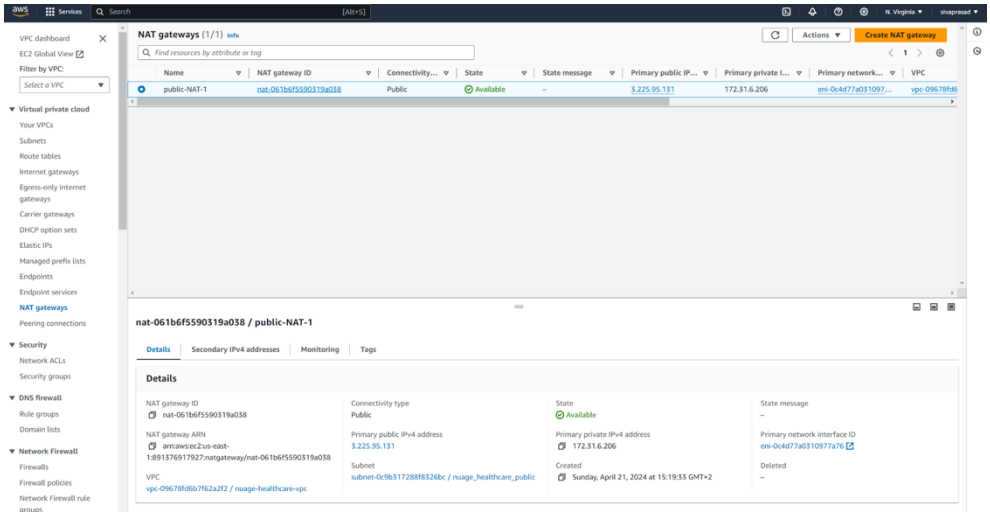
2. **Subnets Configuration**: In the main VPC, public and private subnets were created. One public subnet houses the EC2 instance for the application layer, which contains the user interface (UI). Two private subnets host the microservices and database EC2 instances.

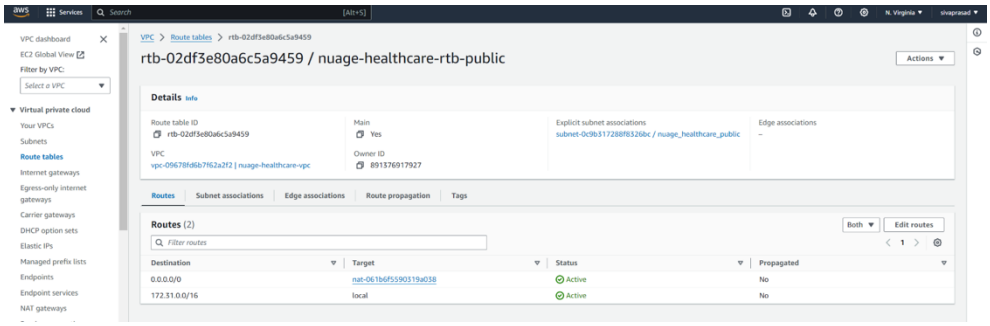




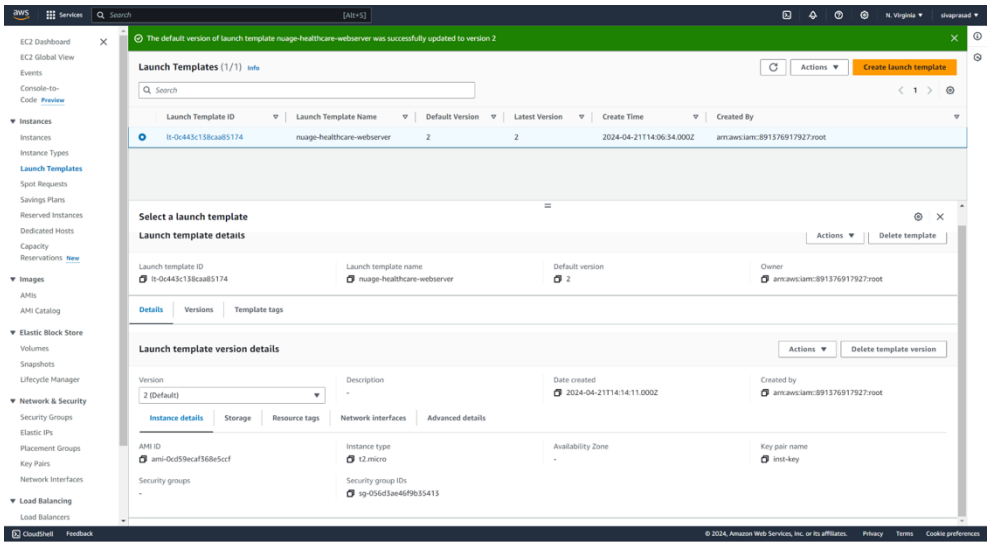


3. **NAT Gateway**: A NAT gateway was set up to allow private subnets to access resources outside the VPC and connect to the internet. A new route was added to this NAT gateway.

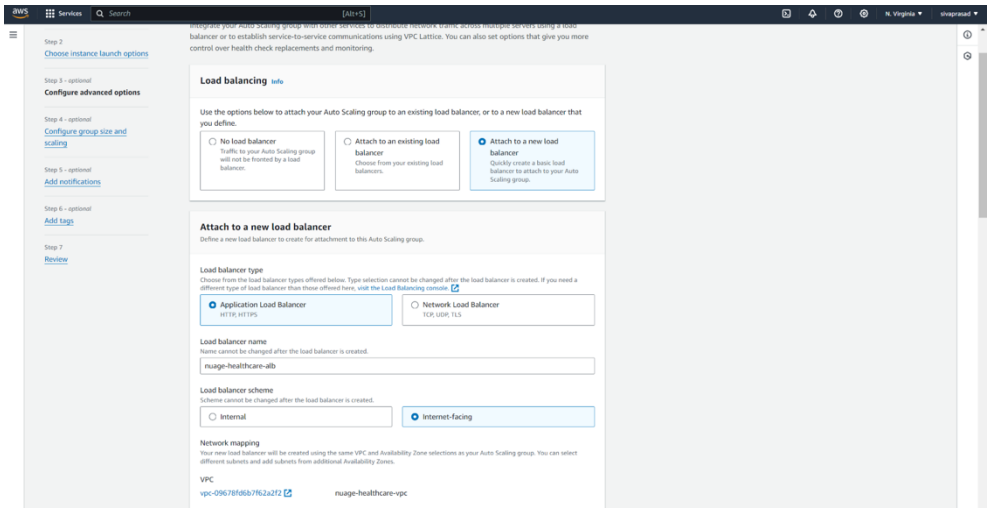


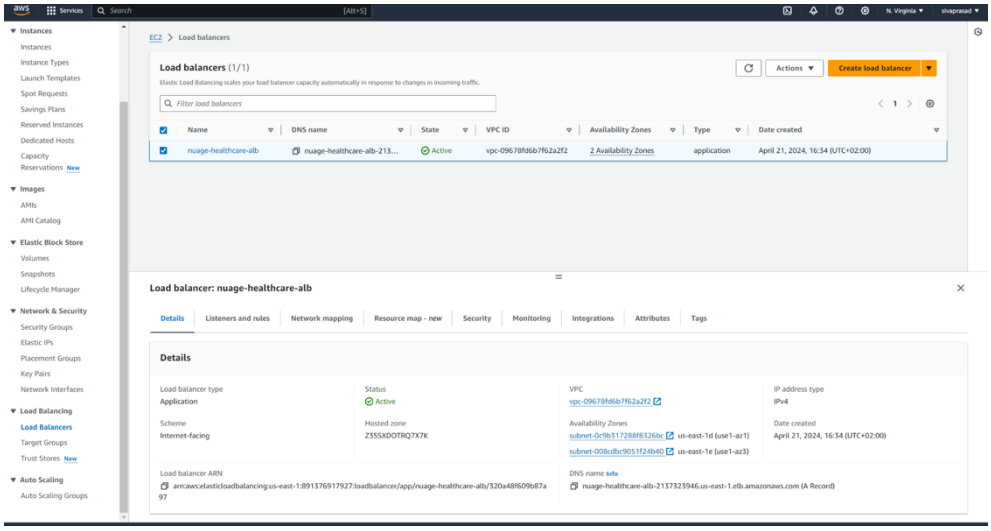


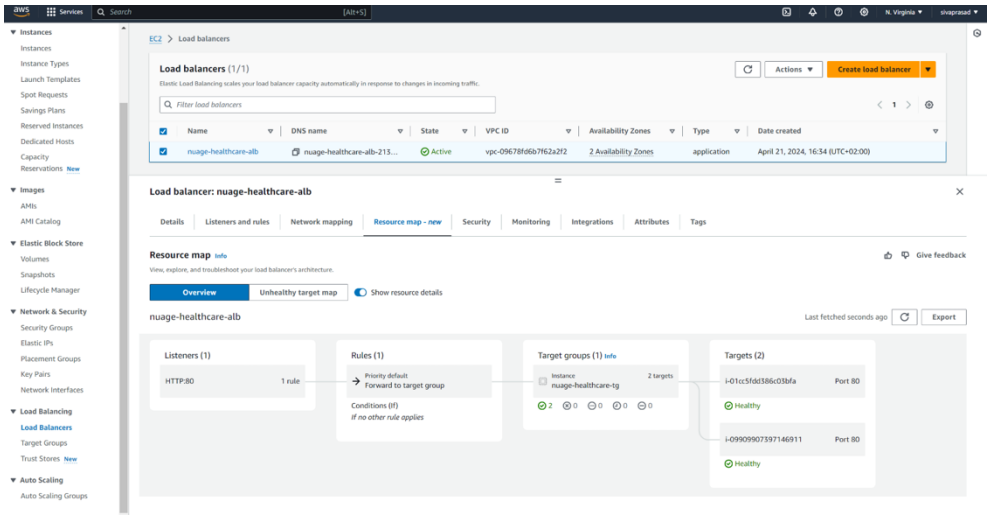
4. **Web Server Launch Template**: A web server launch template was created, which is used by an Auto Scaling Group (ASG) to dynamically launch EC2 instances. A new security group with inbound SSH, HTTP, and HTTPS rules was created for this purpose.



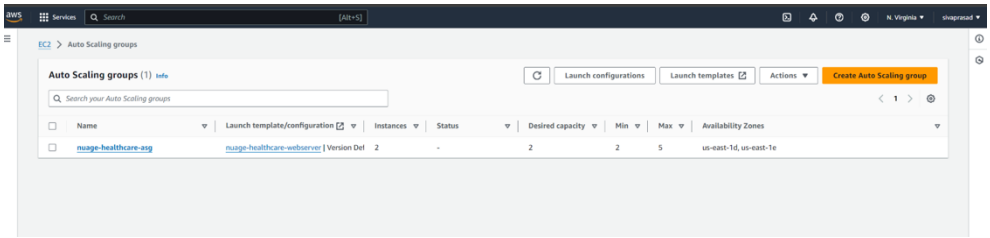
5. **Load Balancer Configuration**: A load balancer was set up to distribute incoming traffic and enhance application availability.

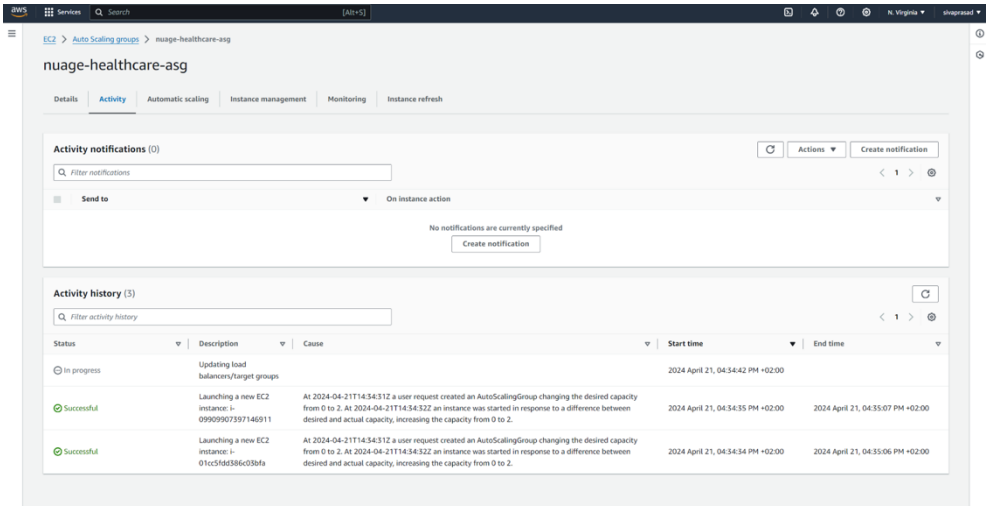






6. **Auto Scaling Group**: An auto-scaling group was configured to automatically adjust the number of running instances based on demand.

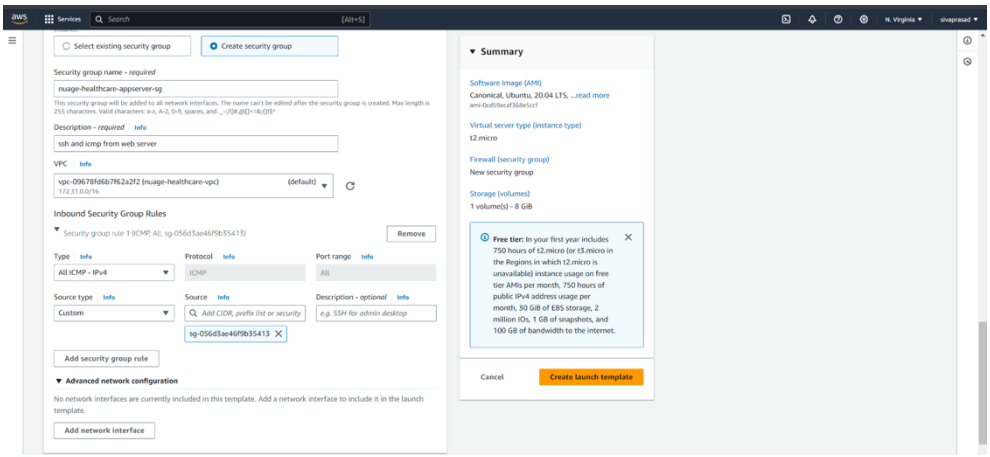


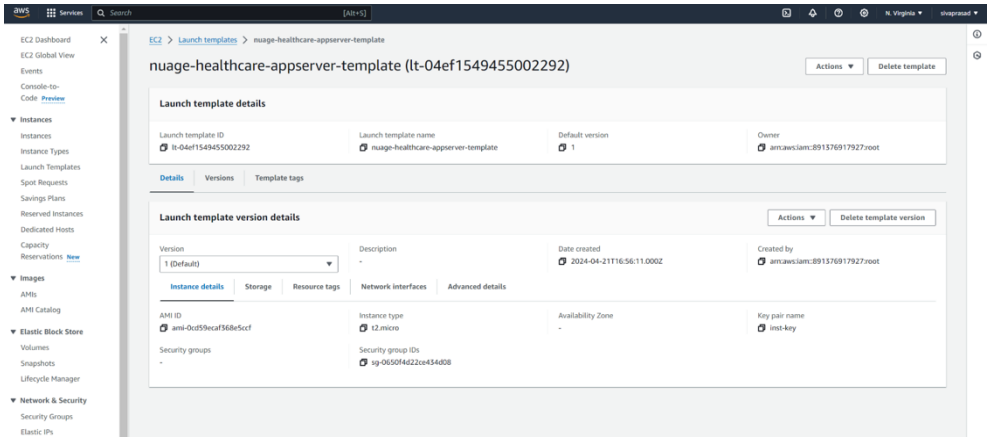


Successfully setting up the Application Layer resulted in the UI being up and running.

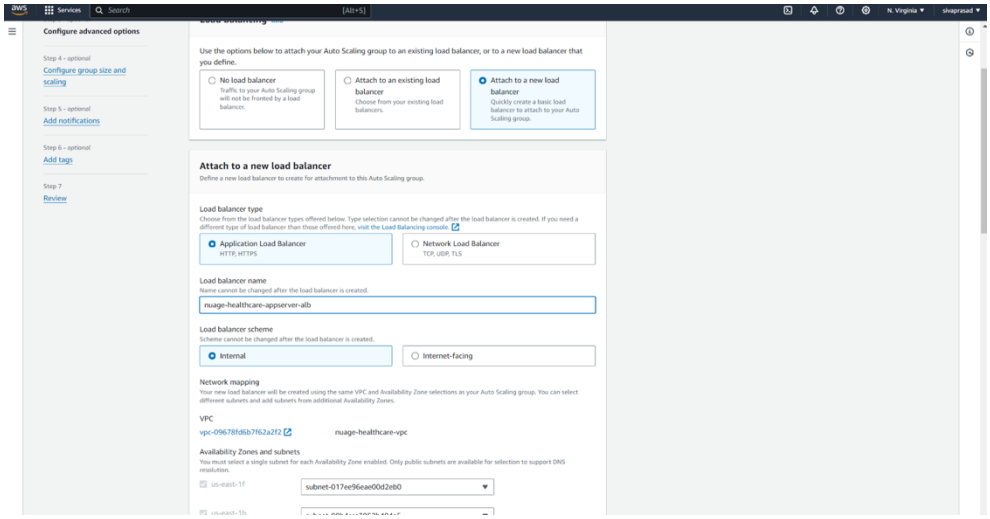
Part B: Microservices Layer Configuration

1. **Launch Template Creation**: A launch template was defined to specify the types of EC2 instances to be used for backend services.

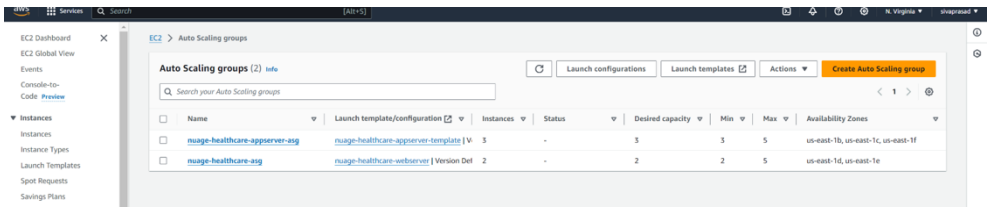


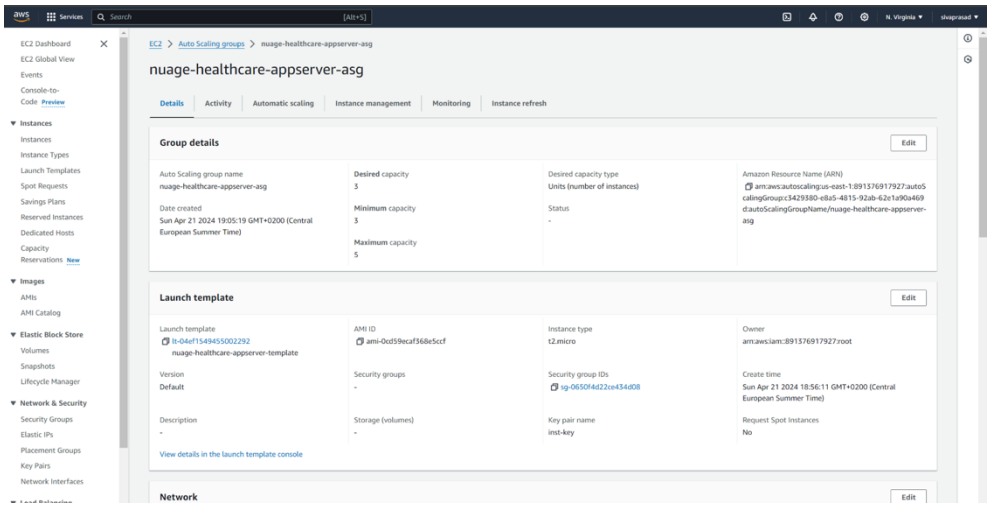


2. **Application Load Balancer**: An application load balancer was established to distribute traffic among the microservices.

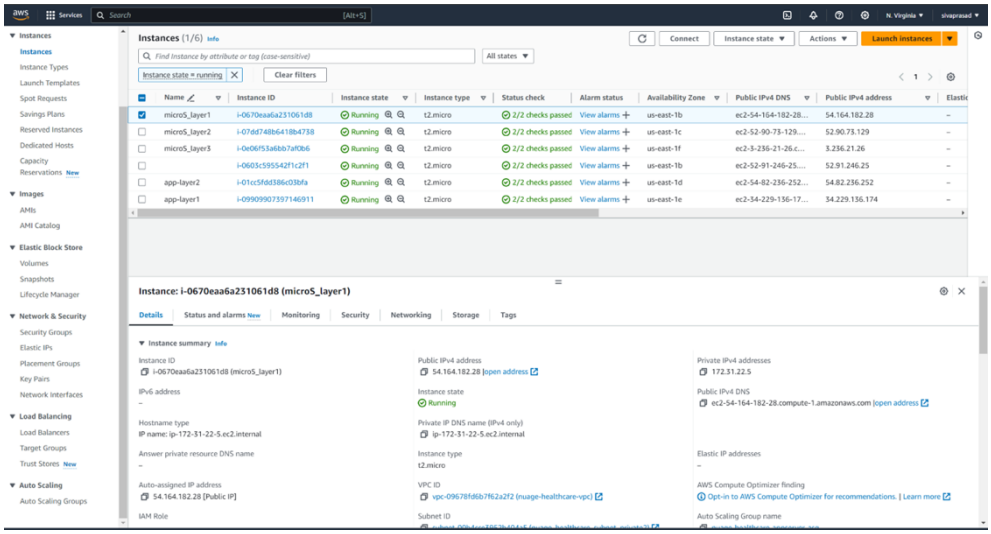


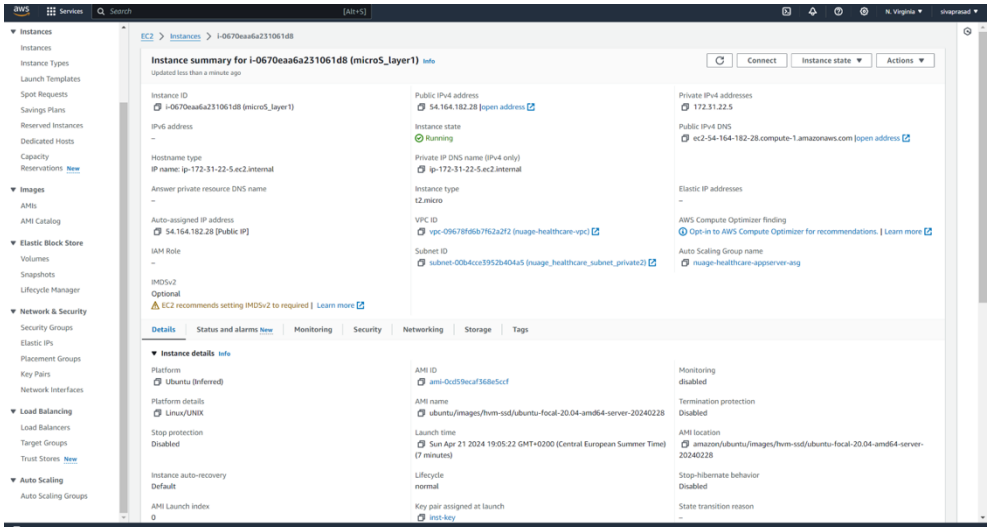
3. **Auto Scaling Group**: An auto-scaling group was set up to manage the private EC2 instances for the microservices layer.



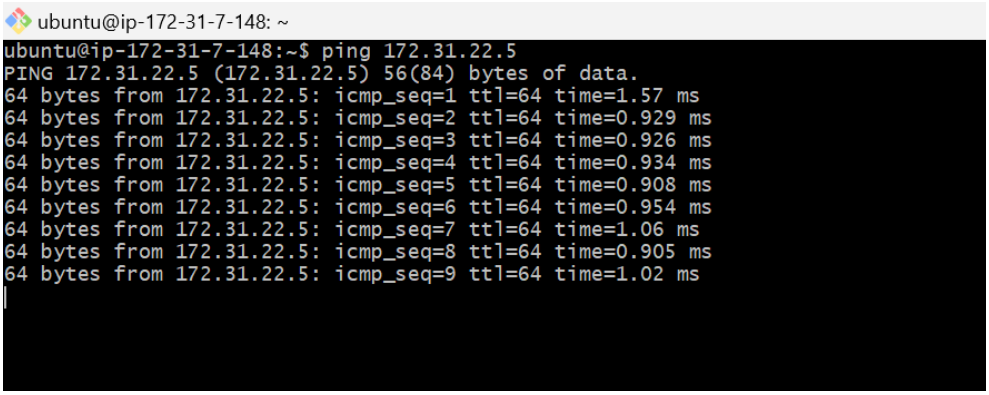


4. **Private EC2 Instances**: Three private EC2 instances for the microservices layer were launched.



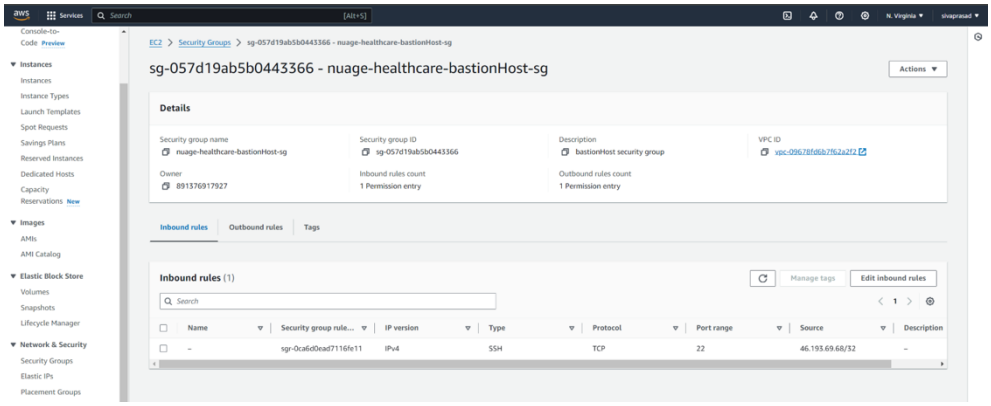


5. Now let's ping the private IP address of the instances from app layer EC2 instance.

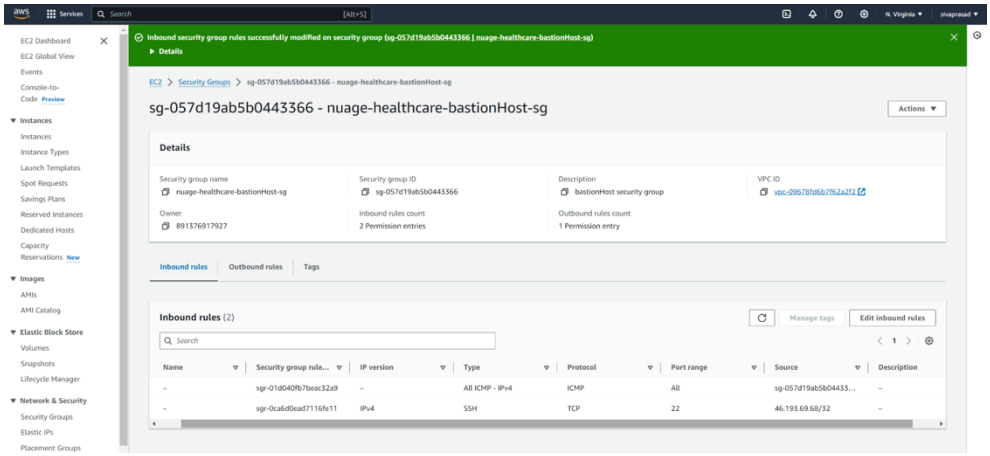


As seen in the terminal we successfully pinged the app server and received a response.

6. **Bastion Host Creation**: A bastion host was established as a secure entry point to access the private network from the public network, protecting the microservices layer.

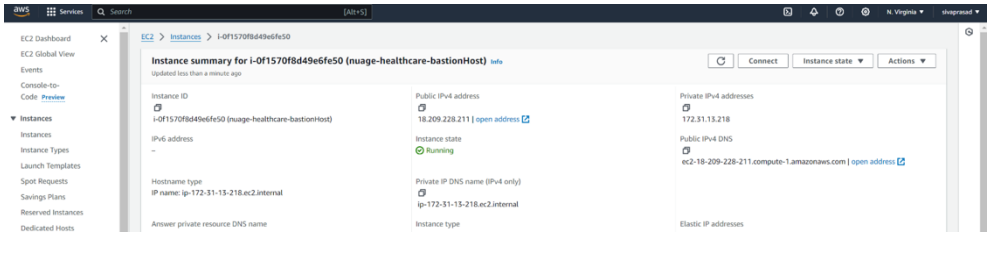


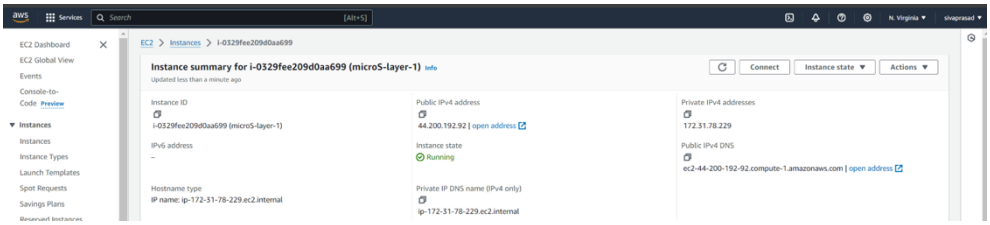
Editing inbound rules for the created security group to make sure we are allowing SSH access only from the bastion host server.

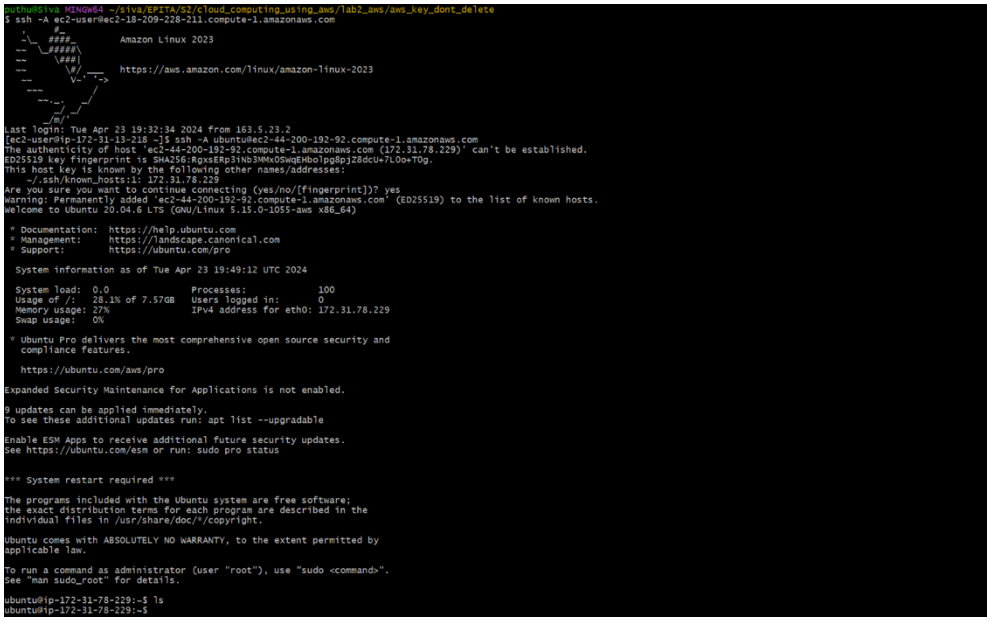


7. **Logging into Bastion Host**: A successful connection to the private microservices layer was achieved via the Bastion Host.

Connecting to private microservice layer via bastion host.



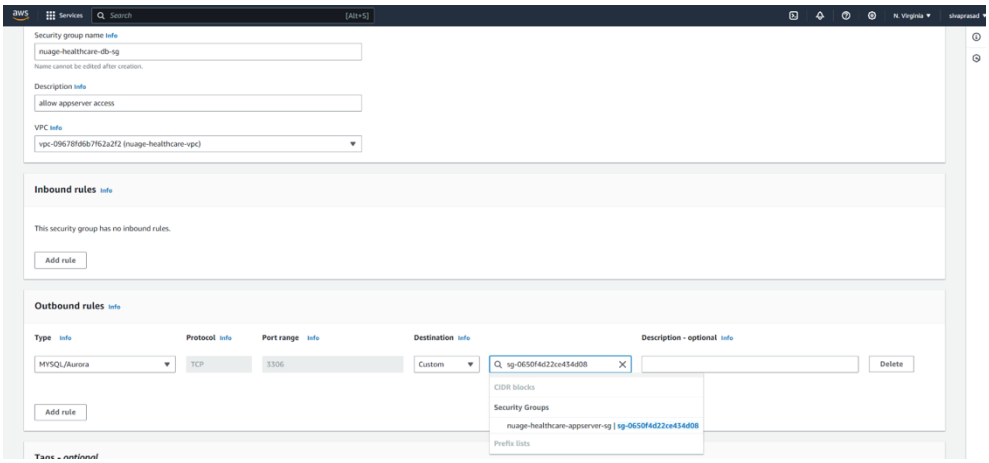




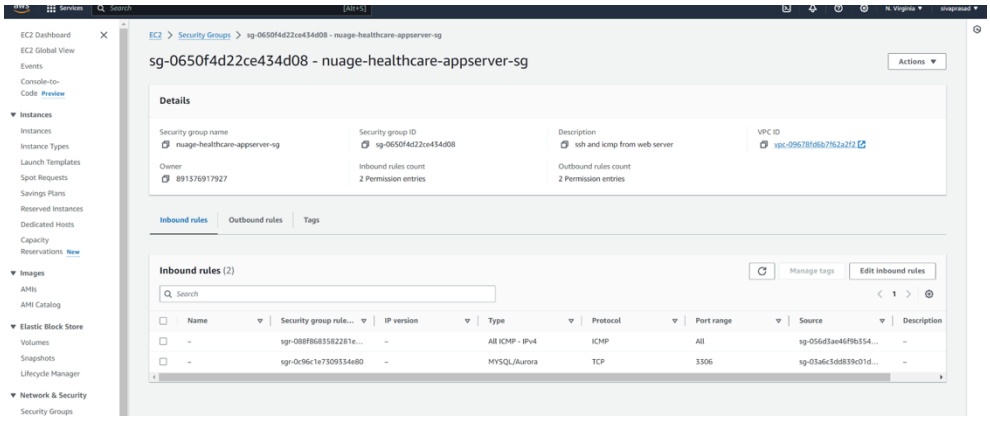
The Microservices Layer configuration was completed successfully.

Part C: Database Layer Configuration

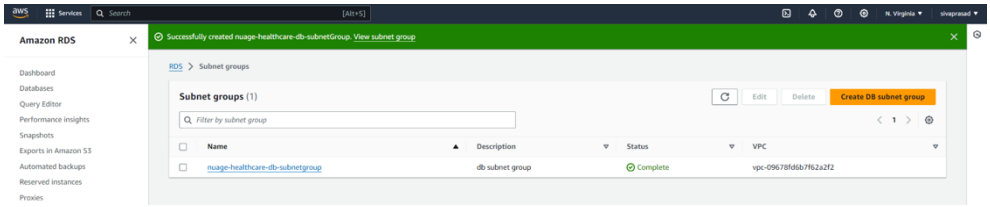
1. **Database Security Group**: A security group was created for the database, with updated inbound and outbound rules to control traffic.



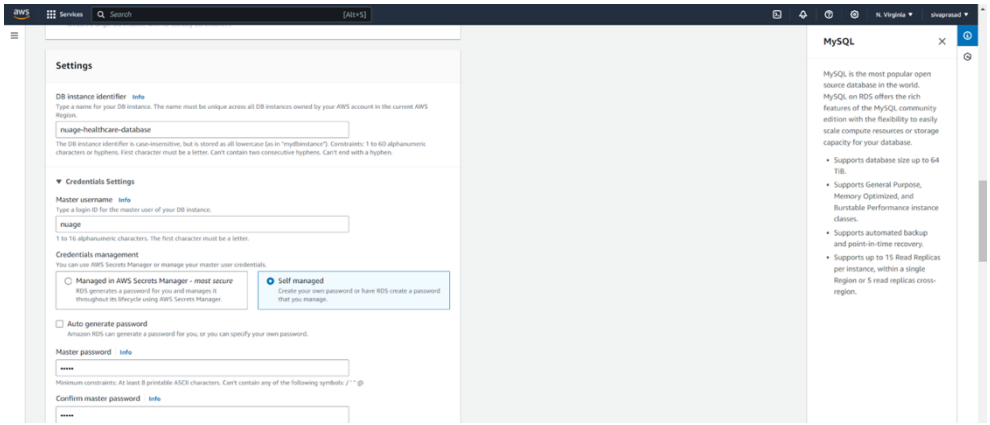
Update inbound and outbound rules of app server also.

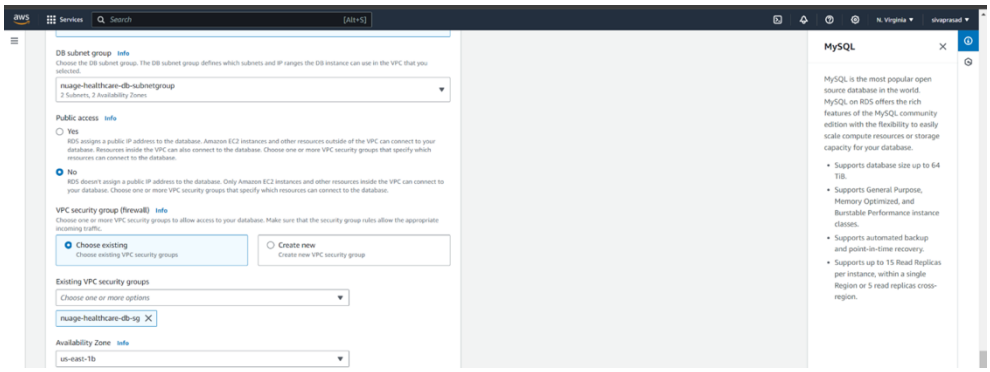


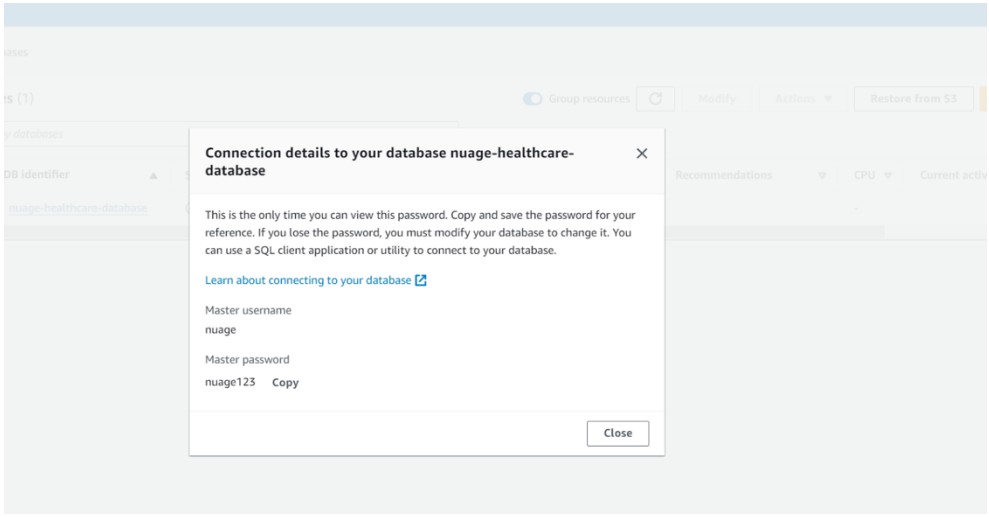
2. **Database Subnet Group**: A database subnet group was established for the RDS database.

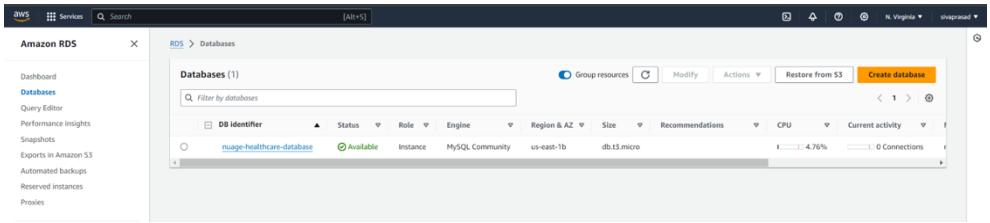


3. **RDS Database Creation**: An Amazon Relational Database Service (RDS) database was created with specified identifiers and credentials.

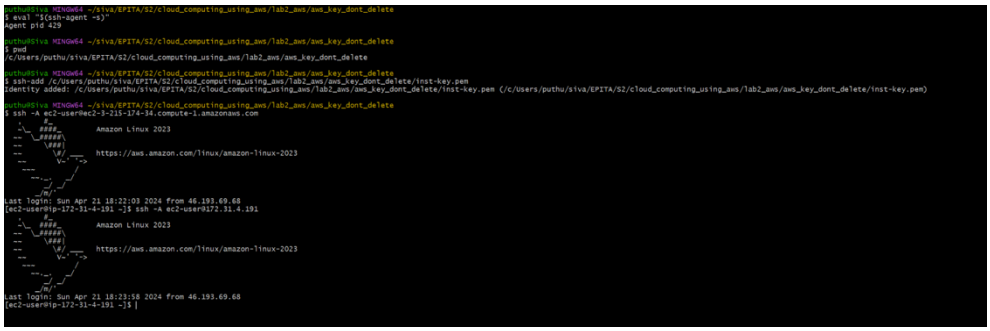


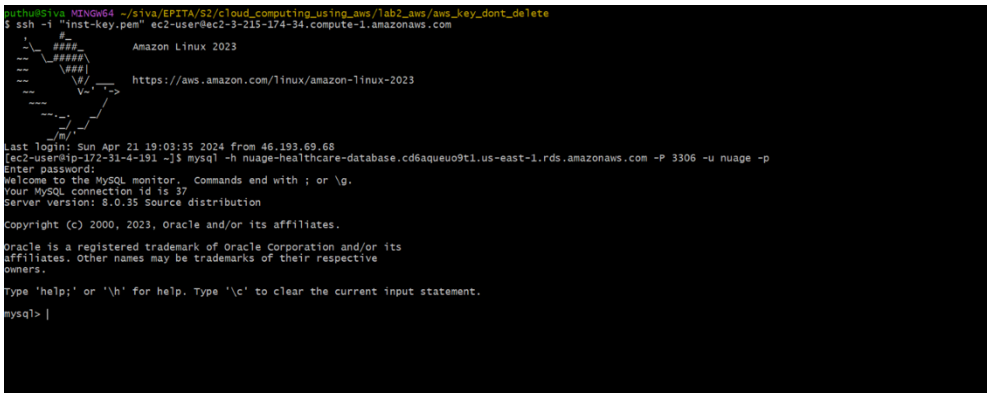

DB subnet group Info
nuage-healthcare-db-subnetgroup
Public access Info
Yes
No
VPC security group (firewall) Info
Choose existing
Create new
Existing VPC security groups
nuage-healthcare-db-sg
Availability Zone Info
us-east-1b
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.


Connection details to your database nuage-healthcare-database
This is the only time you can view this password. Copy and save the password for your reference. If you lose the password, you must modify your database to change it. You can use a SQL client application or utility to connect to your database.
Learn about connecting to your database
Master username
nuage
Master password
nuage123 Copy
Close


Amazon RDS
Dashboard
Databases
Query Editor
Performance Insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
RDS > Databases
Databases (1)
Group resources
Modify
Actions
Restore from S3
Create database
Filter by databases
DB identifier
Status
Role
Engine
Region & AZ
Size
Recommendations
CPU
Current activity
nuage-healthcare-database
Available
Instance
MySQL Community
us-east-1b
db.t3.micro
4.76%
0 Connections

4. **Connecting to the Database**: Connection to the database was established by connecting to the bastion host and then using MySQL commands.

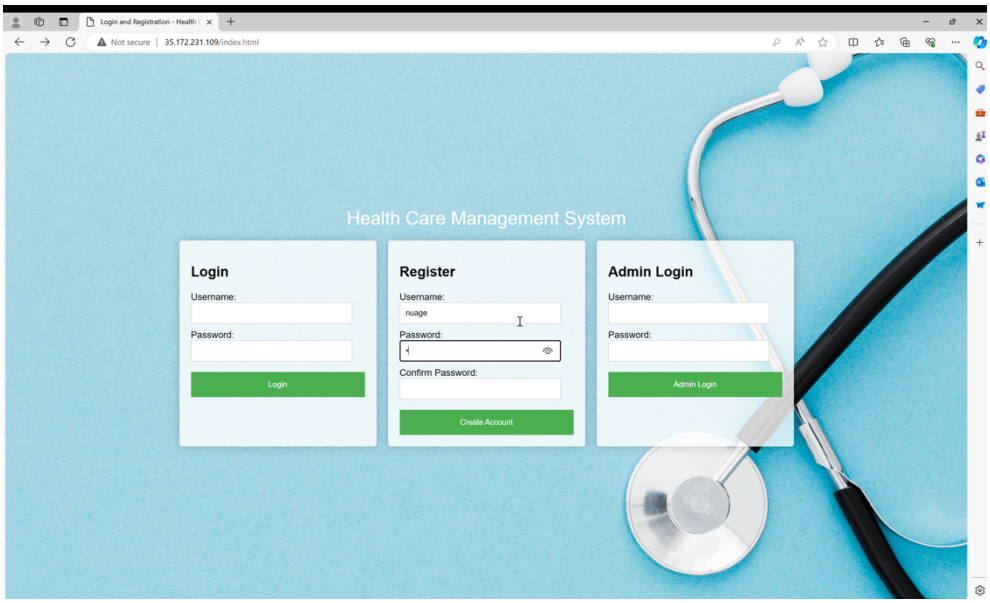




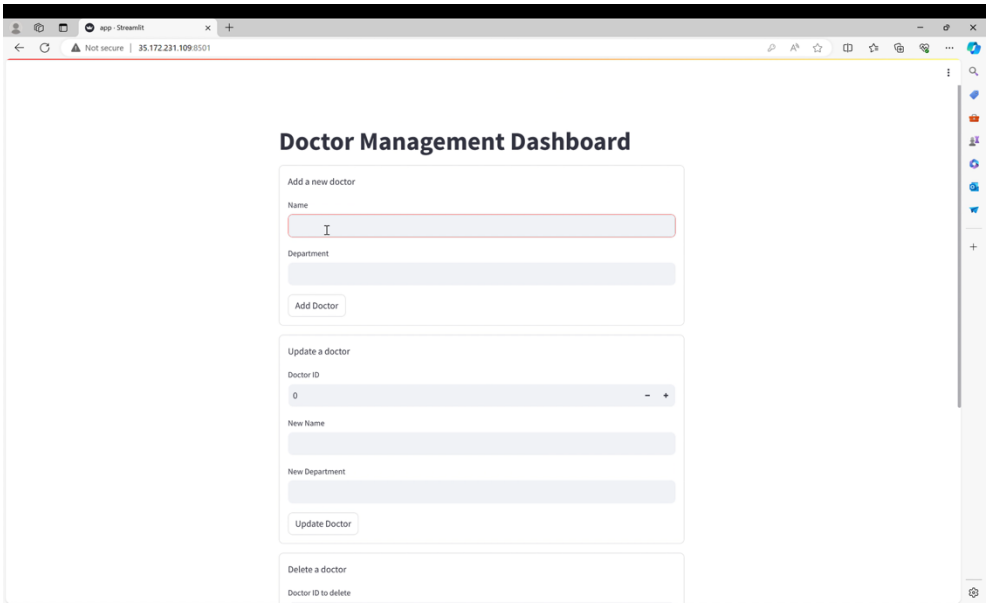
The Database Layer was successfully completed.

Part D: USER INTERFACE

1. Health Care Management System:



2. Doctor Management Dashboard:



6. CONCLUSION

The implementation of the cloud-based Healthcare Management System on Amazon Web Services (AWS) has demonstrated a robust and efficient solution to the challenges faced by healthcare organizations in managing patient data and delivering effective healthcare services. The architecture, comprising a user interface and a backend layer deployed as microservices, ensures high scalability, security, and cost-effectiveness.

The hybrid cloud model, integrating Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), provides remarkable flexibility, scalability, and security. The use of AWS services such as Amazon RDS and Amazon S3 ensures high availability, durability, and security of our data. The system's architecture, divided into public and private subnets, provides a secure and efficient environment for managing patient data. The use of auto-scaling groups and load balancers ensures the system can dynamically adjust to changes in demand, ensuring high availability and improved performance.

The successful implementation of the application, microservices, and database layers demonstrates the system's ability to handle business logic and core application functions effectively, facilitating seamless communication between the application and database layers.

The cloud-based Healthcare Management System provides a comprehensive solution to the challenges faced by healthcare organizations, ensuring efficient management of patient data, robust security, and cost optimization. The system's design and implementation ensure its reliability, effectiveness, and compliance with healthcare industry standards, making it an ideal solution for delivering high-quality healthcare services.