

# **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**

<b>Abstract.....</b>	ii
<b>1 Introduction.....</b>	1
<b>2 Cloud Infrastructure.....</b>	2
<b>3 Architecture Diagram.....</b>	3
<b>4 Microservices Architecture Overview.....</b>	4
4.1 Scalability.....	4
4.2 Performance.....	4
4.3 Security.....	5
4.4 Cost Optimization.....	5
<b>5 Implementation.....</b>	6
5.1 Setting Up the Application Layer.....	6
5.2 Microservices Layer Configuration.....	11
5.3 Database Layer Configuration.....	17
5.4 User Interface.....	21
<b>5 Conclusion.....</b>	22

# **LIST OF FIGURE**

3.1	Proposed Architecture Diagram	3
-----	-------------------------------	---

# **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.

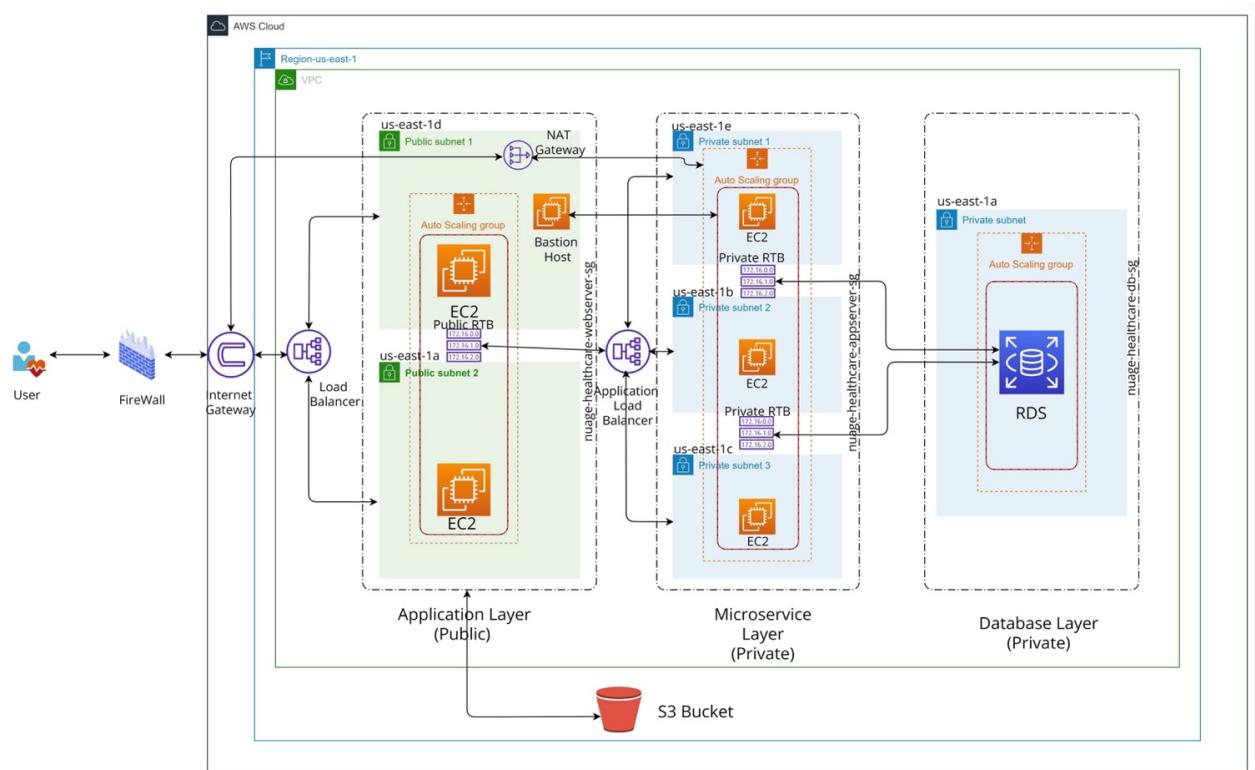
### **4. Microservices Layer (Private):**

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.

The image contains two screenshots of the AWS VPC console. The top screenshot shows the 'Your VPCs' page with a single VPC listed: 'nuage-healthcare-vpc' (VPC ID: vpc-09678fd6b7f62a2f2). The bottom screenshot shows the detailed configuration page for the same VPC, highlighting the 'Details' tab which displays various network settings like CIDR ranges, route tables, and DNS resolution. Both screenshots also show the 'Resource map' tab, which provides a visual representation of the VPC's structure, including subnets, route tables, and network connections.

**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.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 address
nuage_healthcare_public1	subnet-0c9b317288f8326bc	Available	vpc-09678f66b7f62a2f2	172.31.0.0/20	-	4090
nuage_healthcare_subnet_private1	subnet-d17ecf6eaed0d2e0	Available	vpc-09678f66b7f62a2f2	172.31.64.0/20	-	4091
nuage_healthcare_subnet_private2	subnet-00b0cc03952b4045	Available	vpc-09678f66b7f62a2f2	172.31.16.0/20	-	4090
nuage_healthcare_public2	subnet-00b0cc03951f24b40	Available	vpc-09678f66b7f62a2f2	172.31.48.0/20	-	4091
	subnet-0d4184d4ff5ed824	Available	vpc-09678f66b7f62a2f2	172.31.32.0/20	-	4091
	subnet-0db4483161512d44	Available	vpc-09678f66b7f62a2f2	172.31.80.0/20	-	4091

Subnet ID	Subnet ARN	State	IPv4 CIDR
subnet-0c9b317288f8326bc	arn:aws:ec2:us-east-1:891376917927:subnet/subnet-0c9b317288f8326bc	Available	172.31.0.0/20

Flow logs section shows no matching resources found.

Subnet ID	Subnet ARN	State	IPv4 CIDR
subnet-017ee96ea00d2e00	arn:aws:ec2:us-east-1:891376917927:subnet/subnet-017ee96ea00d2e00	Available	172.31.64.0/20

Route table section shows a route to destination 172.31.0.0/16 via target local with gateway ip-026c709021df5cd6.

- 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.

The screenshot shows the AWS VPC NAT gateways console. The main table displays one entry:

Name	NAT gateway ID	Connectivity...	State	State message	Primary public IP...	Primary private IP...	Primary network...	VPC
public-NAT-1	nat-061b6f5590319a038	Public	Available	-	3.225.95.131	172.31.6.206	eni-0c4d77a0310977a6	vpc-09678f6d

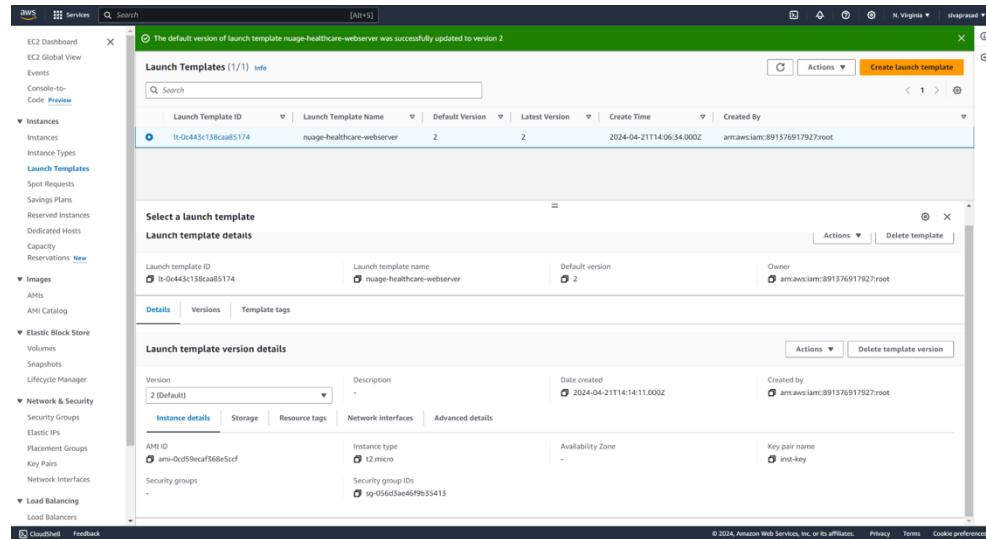
The sidebar on the left includes sections for Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services), NAT gateways (Peerings connections), Security (Network ACLs, Security groups), DNS Firewall (Rule groups, Domain lists), and Network Firewall (Firewalls, Firewall policies, Network Firewall rule groups).

The screenshot shows the AWS VPC route tables console. The main table displays one entry:

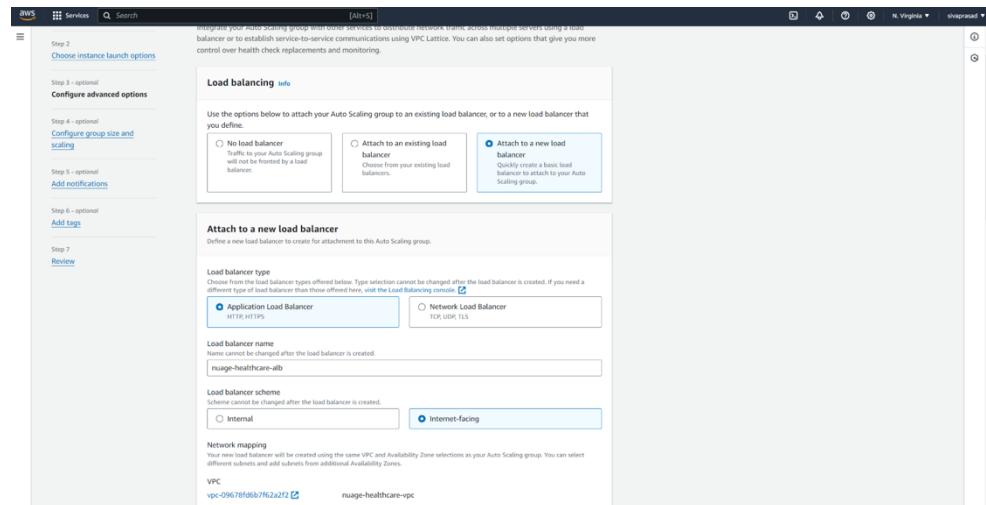
Route table ID	Main	Explicit subnet associations	Edge associations
rtb-02df3e80a6c5a9459	Yes	subnet-0c9b317288f8326bc / nuage_healthcare_public	-

The sidebar on the left includes sections for Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services), NAT gateways, and Network Firewall.

- 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.



**Load balancer: nuage-healthcare-alb**

**Details**

- Load balancer type: Application
- Status: Active
- VPC: vpc-09678f6db7f62a2f2
- IP address type: IPv4
- Scheme: Internet-facing
- Hosted zone: Z155XQDTRQ7X7K
- Availability Zones: subnet-0c9b3172888336bc (us-east-1d), subnet-003c0bc5905f24b40 (us-east-1e), subnet-003c0bc5905f24b40 (us-east-1f)
- Date created: April 21, 2024, 16:34 (UTC+02:00)
- Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:891376917927:loadbalancer/app/nuage-healthcare-alb/320a4bf609d87a97
- DNS name info: nuage-healthcare-alb-2137323946.us-east-1.elb.amazonaws.com (A Record)

**Load balancer: nuage-healthcare-alb**

**Resource map**

**Overview**

**Listeners (1)**

- HTTP:80

**Rules (1)**

- Priority default → Forward to target group  
Conditions (If): if no other rule applies

**Target groups (1) Info**

- nuage-healthcare-tg

**Targets (2)**

- i-01cc5fd586c018bfaf Port 80 (Healthy)
- i-09909907397146911 Port 80 (Healthy)

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

**Auto Scaling groups (1) Info**

**Launch configurations**

**Launch template/configuration**

**Instances**

**Status**

**Desired capacity**

**Min**

**Max**

**Availability Zones**

**nuage-healthcare-asg**

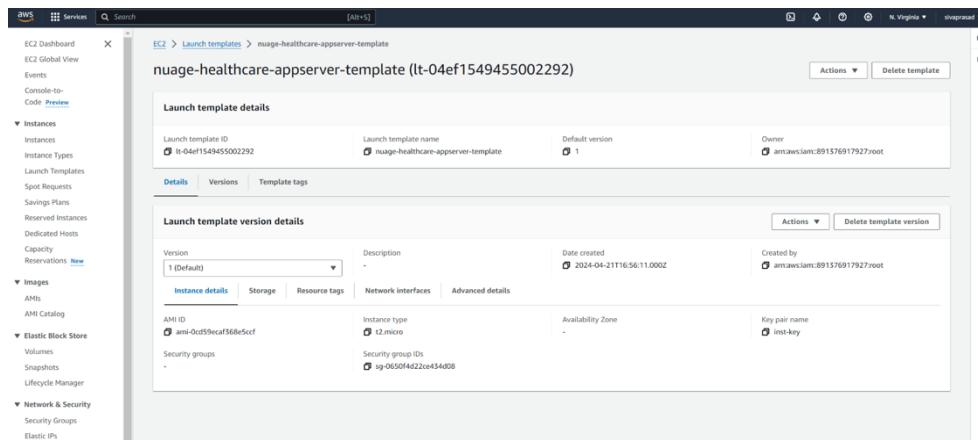
nuage-healthcare-webserver | Version 1 | 2

2 2 5 us-east-1d, us-east-1e

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

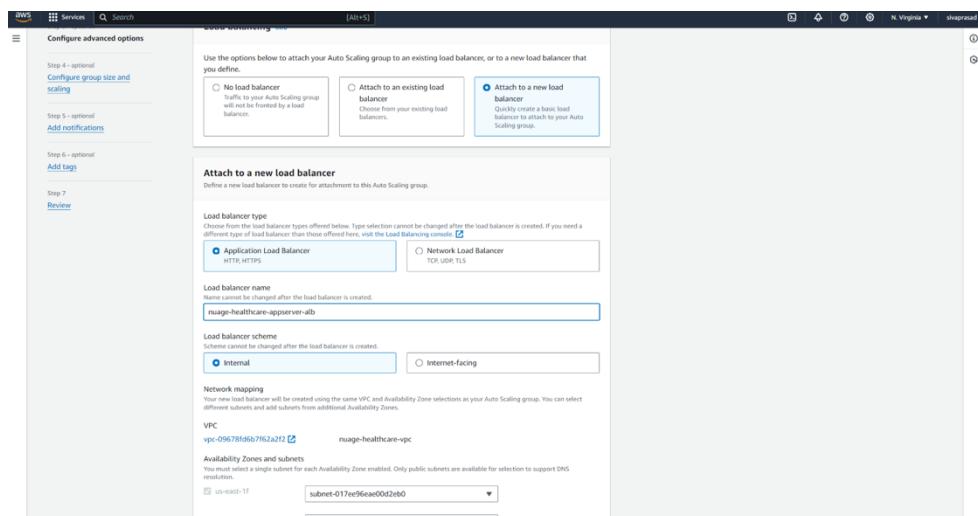
## Part B: Microservices Layer Configuration

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



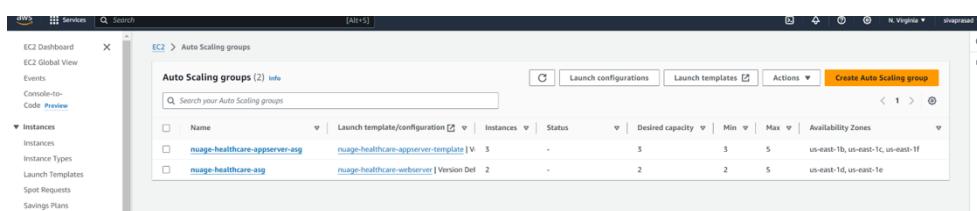
## 2. Application Load Balancer:

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.

The screenshot shows the AWS EC2 Instances page. The instance summary for i-0670ea6a231061d8 (micro5\_layer1) is displayed. Key details include:

- Instance ID:** i-0670ea6a231061d8 [micro5\_layer1]
- Public IPv4 address:** 54.164.182.28 [open address]
- Private IP DNS name (IPv4 only):** ip-172-31-22-5.ec2.internal
- Instance state:** Running
- Instance type:** t2.micro
- VPC ID:** vpc-09678f6b762a2f2 [usage-healthcare-vpc]
- Subnet ID:** subnet-00b4cc03952b404a5 [usage\_healthcare\_subnet\_private2]
- Elastic IP addresses:** 172.31.22.5
- Public IP-4 DNS:** ec2-54-164-182-28.compute-1.amazonaws.com [open address]
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
- Auto Scaling Group name:** usage-healthcare-appserver-asg

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

```
ubuntu@ip-172-31-7-148:~$ ping 172.31.22.5
PING 172.31.22.5 (172.31.22.5) 56(84) bytes of data.
64 bytes from 172.31.22.5: icmp_seq=1 ttl=64 time=1.57 ms
64 bytes from 172.31.22.5: icmp_seq=2 ttl=64 time=0.929 ms
64 bytes from 172.31.22.5: icmp_seq=3 ttl=64 time=0.926 ms
64 bytes from 172.31.22.5: icmp_seq=4 ttl=64 time=0.934 ms
64 bytes from 172.31.22.5: icmp_seq=5 ttl=64 time=0.908 ms
64 bytes from 172.31.22.5: icmp_seq=6 ttl=64 time=0.954 ms
64 bytes from 172.31.22.5: icmp_seq=7 ttl=64 time=1.06 ms
64 bytes from 172.31.22.5: icmp_seq=8 ttl=64 time=0.905 ms
64 bytes from 172.31.22.5: icmp_seq=9 ttl=64 time=1.02 ms
```

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.

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-0ca6d0ead7116fe11	-	IPv4	SSH	TCP	22	46.193.69.68/32	-

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

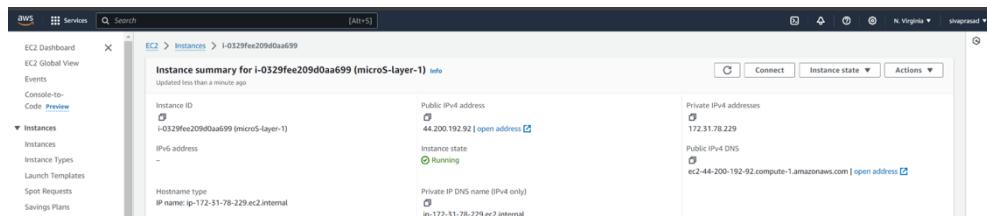
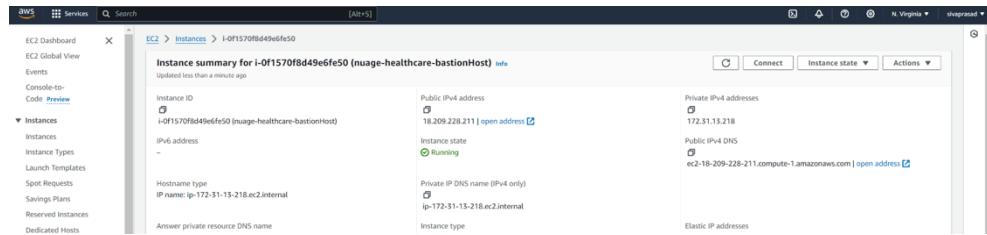
Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-01d040fb7beac32e9	-	All ICMP - IPv4	ICMP	All	-	sg-057d19ab5b0443366	-
sgr-0ca6d0ead7116fe11	-	IPv4	SSH	TCP	22	46.193.69.68/32	-

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

```
puthu@Siva_MINGW64 ~/siva/EPITA/S2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
$ ssh -i "inst-key.pem" ec2-user@ec2-3-215-174-34.compute-1.amazonaws.com
The authenticity of host 'ec2-3-215-174-34.compute-1.amazonaws.com (3.215.174.34)' can't be established.
ED25519 key fingerprint is SHA256:soBkkn1ml2Xqjy2wISeokOnt/wCnhfigQQyxSHhpy.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
warning: Permanently added 'ec2-3-215-174-34.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

# 
# Amazon Linux 2023
# ##
# ##| https://aws.amazon.com/linux/amazon-linux-2023
#/ ,-->
# \~ / 
# \~ / 
[ec2-user@ip-172-31-4-191 ~]$
```

Connecting to private microservice layer via bastion host.



```

$ ssh -A ec2-user@ec2-18-209-228-211.compute-1.amazonaws.com
Last login: Tue Apr 23 19:49:12 UTC 2024 from 163.2.23.2
[ec2-user@ip-172-31-11-218 ~]$ ssh -A ubuntu@ec2-44-200-192-92.compute-1.amazonaws.com
The authenticity of host 'ec2-44-200-192-92.compute-1.amazonaws.com (172.31.78.29)' can't be established.
ED25519 key fingerprint is SHA256:RpxoERp3nB3Mx0SwqEhb0ppjZ8dcduvL0o+T0g.
This key is known by other names/addresses:
  /ssh/known_hosts:1: 172.31.78.29
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-200-192-92.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1055-aws x86_64)

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.
   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

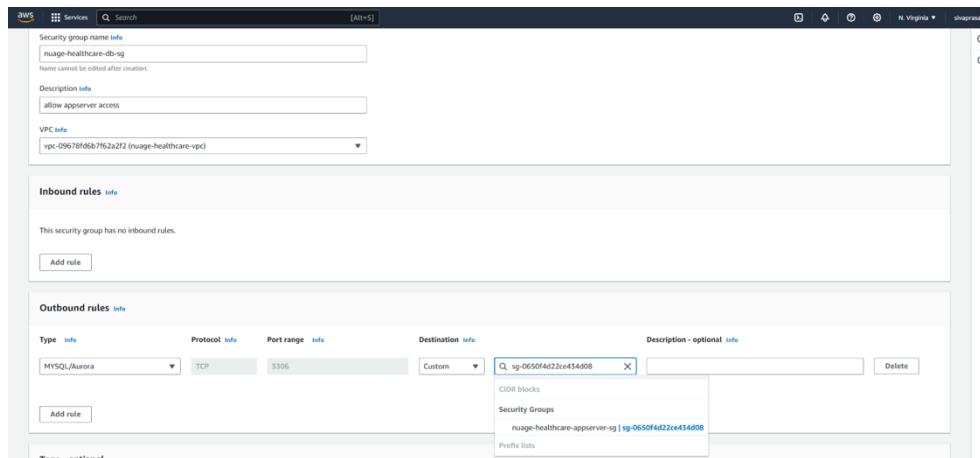
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-78-229:~$ ls
ubuntu@ip-172-31-78-229:~$
```

The Microservices Layer configuration was completed successfully.

## Part C: Database Layer Configuration

- 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.

The screenshot shows the AWS EC2 Security Groups console. A specific security group, 'sg-0650f4d22ce434d08 - nuage-healthcare-appserver-sg', is selected. The 'Details' section shows the security group name, ID, owner, and VPC ID. The 'Inbound rules' tab is active, displaying two entries:

Name	Security group rule...	Type	Protocol	Port range	Source	Description
sgr-088f6683582261e...	-	All ICMP - IPv4	ICMP	All	sg-056d3ee46ff6b354...	-
sgr-0c96c1e730934e80	-	MYSQL/Aurora	TCP	3306	sg-03af6c3dd8f39c01d...	-

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

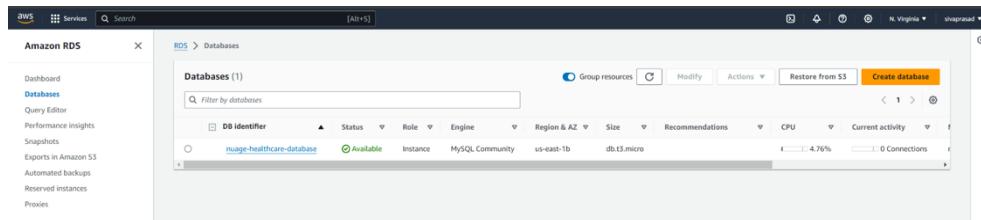
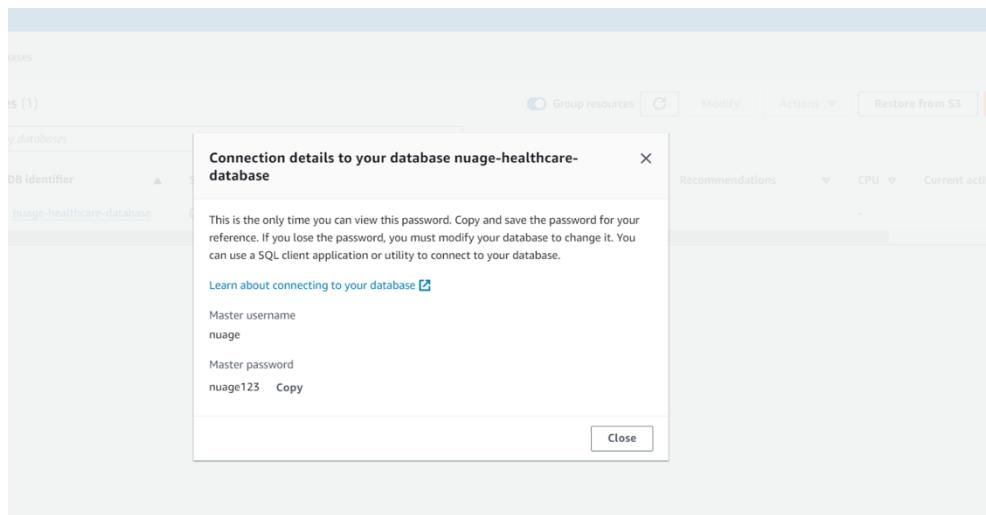
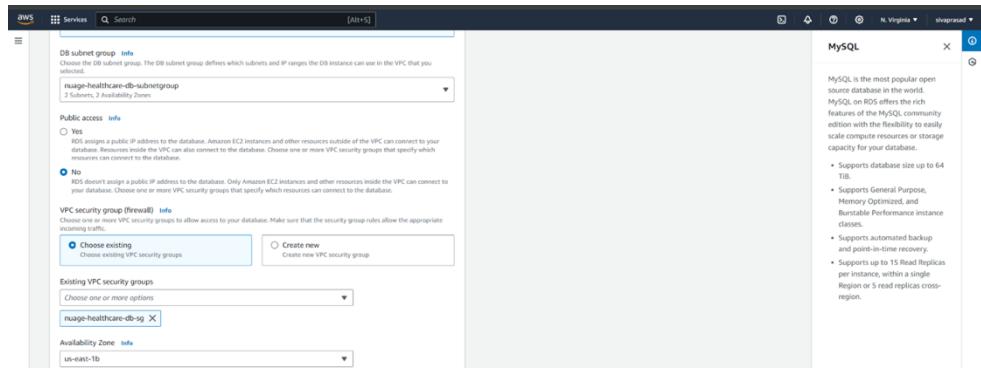
The screenshot shows the AWS RDS Subnet Groups console. A message at the top indicates 'Successfully created nuage-healthcare-db-subnetGroup. View subnet group.' Below, the 'Subnet groups (1)' section lists the newly created group:

Name	Description	Status	VPC
nuage-healthcare-db-subnetgroup	db subnet group	Complete	vpc-09678fd6b7f62a2f2

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

The screenshot shows the AWS RDS MySQL instance creation settings page. The 'DB instance identifier' is set to 'nuage'. Under 'Credentials Settings', the 'Master username' is 'nuage', and the 'Self managed' option is selected. To the right, a sidebar provides information about MySQL:

- MySQL is the most popular open source database in the world.
- MySQL on RDS offers the rich features of MySQL, compatibility with the MySQL client 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.



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

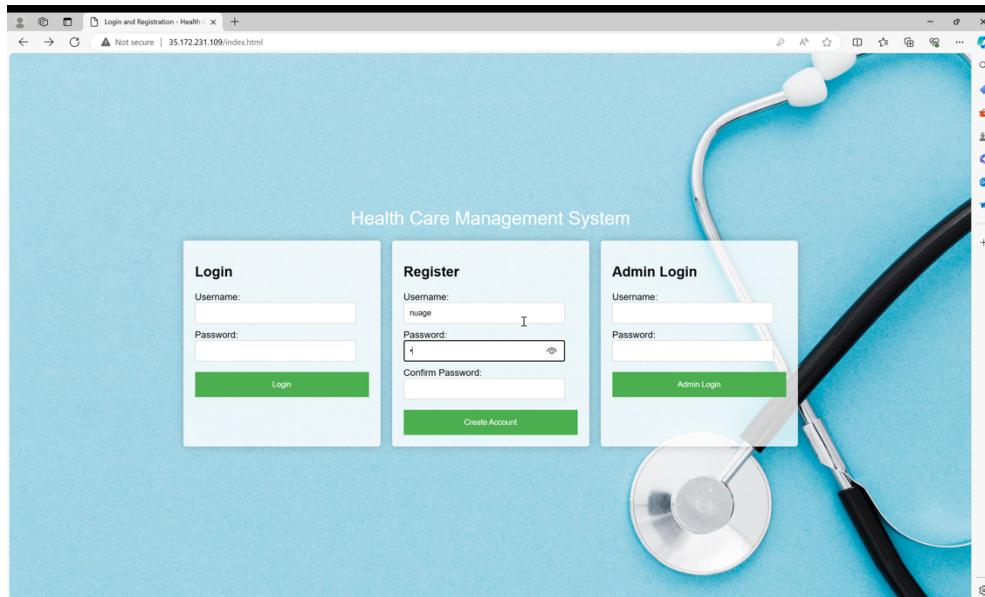
```
siva@SIVA:~/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
$ eval "$!ssh-agent -s"
agent pid 429
siva@SIVA:~/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
$ ssh-add
/c/Users/puthu/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
puthu@SIVA:~/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
$ ssh -i /c/Users/puthu/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete/inst-key.pem ec2-user@ec2-3-215-174-34.compute-1.amazonaws.com
Amazon Linux 2023
Last login: Sun Apr 21 18:22:03 2024 from 46.193.69.68
[ec2-user@ip-172-31-4-191 ~]$ ssh -A ec2-user@172.31.4.191
Amazon Linux 2023
Last login: Sun Apr 21 18:23:58 2024 from 46.193.69.68
[ec2-user@ip-172-31-4-191 ~]$
```

```
puthu@SIVA:~/siva/EPITA/s2/cloud_computing_using_aws/lab2_aws/aws_key_dont_delete
$ ssh -i "inst-key.pem" ec2-user@ec2-3-215-174-34.compute-1.amazonaws.com
Amazon Linux 2023
Last login: Sun Apr 21 19:03:35 2024 from 46.193.69.68
[ec2-user@ip-172-31-4-191 ~]$ mysql -h nuage-healthcare-database.cd6aqueuo9t1.us-east-1.rds.amazonaws.com -P 3306 -u nuage -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 8.0.35 Source distribution
copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> |
```

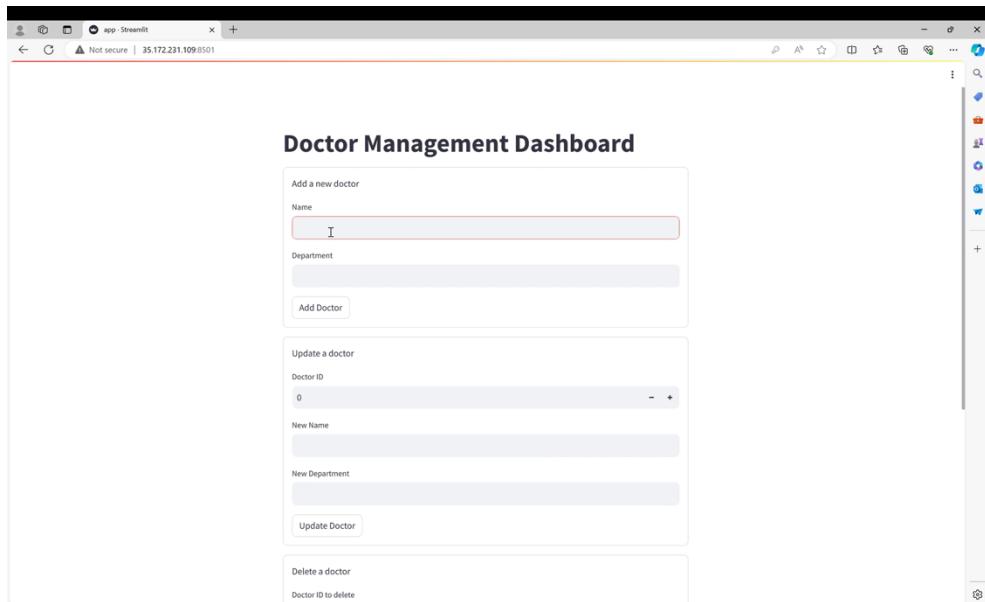
The Database Layer was successfully completed.

## 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.