NP00575-NP000563-Np0006111-CSVC-Group.docx

by DHAN BAHADUR KARKI .

Submission date: 07-Nov-2023 05:24PM (UTC+0800)

Submission ID: 2220451480

File name: 43784_DHAN_BAHADUR_KARKI_._NP00575-NP000563-Np0006111-CSVC-Group_410863_1492156733.docx (22.34M)

Word count: 5776

Character count: 31142

GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT097-3-3-CSVC

CLOUD INFRASTRUCTURE AND SERVICES

NP3F2304IT

HAND OUT DATE: AUGUST 2023

HAND IN DATE: OCTOBER 2023

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 Submit your assignment at the administrative counter.

2 Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing).

3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.

4 Cases of plagiarism will be penalized.

5 The assignment should be bound in an appropriate style (comb bound or stapled).

6 Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.

7 You must obtain 50% overall to pass this module.

Acknowledgement

In order to complete our assignment, there were lot of helping hands and guidelines provided by various individuals and personalities, for which we are greatly thankful and would like to express our tremendous gratitude.

Firstly, we would like to express our heartfelt thankfulness towards **Asia Pacific University (APU)** and **Lord Buddha Education Foundation (LBEF)** for providing us with this outmost opportunity to participate and execute this wonderful assignment along with the possibility to gain theoretical as well as practical concepts of the provided knowledge.

Also, we would like to extend our profound gratitude to our module leader, **Mr. Manish Bhattarai,** for providing us with all the resources and knowledge required to complete our assignment. This assignment would be almost impossible to complete without his provided guidelines, constant support, active participation, and outmost encouragement.

Furthermore, we would thank all our faculty members as well as our friends who have been directly as well as indirectly in contact with us and helped with to convey the important knowledge and information which benefitted us in completion and outcome of this assignment.

we have been through several sites, books, pdfs, and journals for collecting information and conducting investigations to get the best conclusion and analysis for this assignment.

Yours Sincerely,

Aryan Maharjan (NP000563)

Dhan Bahadur Karki (NP000575)

Sujan Shrestha (NP000611)

Table of Contents

1 Introduction 1

1.1 Background 1

1.2 Problem Statement 2

1.3 Objective 2

1.4 Project Scope 3

2 Project Plan 3

3 Proposed Cloud Architecture 4

4 Cost Estimation 6

5 Implementation Flowchart 7

6 Implementation 8

6.1 VPC 8

6.1.1 Subnet 8

6.1.2 Internet Gateway 9

6.1.3 NAT Gateway 9

6.1.4 Security Groups 18

6.2 Load Balancer 20

6.3 Auto Scaling Group 29

6.4 Amazon RDS 38

6.5 AWS Systems Manager Parameter Store 48

6.6 Data Migration 52

6.6.1 Migrate data to RDS instance 52

7 Testing High Availability 58

8 Conclusion 61

9 References 62

Table of Figures

Figure 1: AWS 1
Figure 2: Gantt Chart 3
Figure 3: AWS Architecture 4
Figure 4: AWS Resource Cost Estimation 6
Figure 5: Implementation Flowchart 7
Figure 6: Preview of VPC 8
Figure 7: List of Subnets 9
Figure 8: "Example IGW" Internet Gateway 9
Figure 9: Creating first NAT gateway 10
Figure 10: Creating first NAT gateway 11
Figure 11: Preview of first NAT gateway 11
Figure 12: Routes in a Private Route Table 1 12
Figure 13: Adding NAT gateway to the route 12
Figure 14: Private Route Table 1 after adding NAT gateway route 13
Figure 15: Creating second NAT gateway 13
Figure 16: Creating second NAT gateway 14
Figure 17: Second NAT gateway preview 14
Figure 18: Creating New Private Route Table 15
Figure 19: Preview of Private Route Table 2 15
Figure 20: Adding second NAT to newly created Private Route Table 16
Figure 21: After adding route 16
Figure 22: Private Route Table Subnet Association 17
Figure 23: Associating Private Subnet 2 to Private Route Table 2 17
Figure 24: After Subnet Association 18
Figure 24: List of Security Groups 19
Figure 25: Traffic flow between different security groups 19
Figure 25: Load balancer types 20
Figure 26: ALB’s basic configuration 21
Figure 27: ALB's Network Mapping 21
Figure 28: ALB's Security group 22

Figure 29: Target groups' target type 23
Figure 30: Target group basic configuration 23
Figure 31: Target group advanced health check settings 24
Figure 32: Register Target screen 25
Figure 33: Review of registered targets 25
Figure 34: "SRO-TG" Target group preview 26
Figure 35: Adding Listener and Routing 26
Figure 36: Load balancer summary 27
Figure 37: Successful creation of load balancer 27
Figure 38: Available Load balancers 28
Figure 39: Available Active Load balancer 28
Figure 40: Choosing Launch Template 29
Figure 41: Choosing network environment for ASG 30
Figure 42: Attaching Load balancer 31
Figure 43: ASG Health checks 31
Figure 44: Group Size configuration 32
Figure 45: Notification options 33
Figure 46: ASG tags 33
Figure 47: Review ASG 34
Figure 48: ASG Capacity Updating Status 35
Figure 49: Launching 2 new instances 36
Figure 50: Successful creation of 2 new instances after status check 36
Figure 51: Home page of web application 37
Figure 52: Query page 38
Figure 53: Subnet Groups 39
Figure 54: Creating Subnet groups 40
Figure 55: Subnet selection in database subnet group 41
Figure 56: Successful creation of subnet group 41
Figure 57: RDS Engine options 42
Figure 58: RDS Deployment options 43
Figure 59: RDS settings 43

Figure 60: RDS Instance and Storage Configuration 44
Figure 61: RDS Connectivity 45
Figure 62: RDS security group 45
Figure 63: RDS additional configuration 46
Figure 64: Creating state of RDS 46
Figure 65: RDS instance created 47
Figure 66: “sro-db” RDS database preview 47
Figure 67: “sro-db” RDS database preview 2 48
Figure 68: AWS SSM Parameter Store Portal 49
Figure 69: Storing database Endpoint 49
Figure 70: Storing database username 50
Figure 71: Storing database password 50
Figure 72: Storing database’s name 51
Figure 73: List of parameters 51
Figure 74: Preview of Bastion Host 52
Figure 75: Connecting to Bastion instance 53
Figure 76: Bastion Terminal 53
Figure 77: Bastion Terminal creating file 54
Figure 78: Capstone Project SSH key’s text 54
Figure 79: Setting read permission to .pem file 55
Figure 80: Connecting to private instance 55
Figure 81: Allowing traffic on port 22 from Bastion SG 56
Figure 82: SSH to private instance 56
Figure 83: Running migration command in the private instance 57
Figure 84: Query web page after database migration 57
Figure 85: Terminating one of EC2 instance 58
Figure 86: List of EC2 instances 59
Figure 87: Target group status 59
Figure 88: Healthy Targets in Target Group 60

1 Introduction

Amazon Web Services is a cloud computing service provided by Amazon which has been a boon to numerous companies and business and has directly averted their problems. This has revolutionized the traditional procedure of resources utilization such as physical use of storage systems along with installation of personal power system for the devices and processing power for the use of software installation and usage as well. This modern venture of cloud computing has directly benefitted with ease in cost reduction, scalability, easy management, and flexibility in the system (Buyya, 2020).

Figure 1: AWS

There are numerous services which are provided by the Amazon Web Services (AWS) which include Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), and Infrastructure-as-a-Service (IaaS) availability. These multiple options have their own benefits based on the company's requirement and help them with a wide range of necessary tools and solutions. Therefore, cloud computing using AWS has been a popular choice for all companies, businesses, software developers, both governmental and non-profit organizations, educational institutes, as well as confidential business and individuals, with a wide range of coverage across the world (Ranjan, 2021).

1.1 Background

The Example Social Research Organization provides necessary services to the international community of social science researchers a useful internet platform. Through an easy-to-use query system, users may get vital global development statistics on the website, including life expectancy statistics for each nation in the globe and other socioeconomic indices. The website

was first hosted by a for-profit hosting business, thanks to the efforts of Shirley Rodriguez, a committed researcher in the organization. But as the website grew in popularity, more people visited it, which caused performance problems.

Additionally, there was a worrying security event involving attempted ransomware assaults. So, the website and database were moved to Amazon Web Services (AWS), where they are housed on an Amazon Elastic Compute Cloud (EC2) instance inside a public subnet in addition to a MySQL instance. Shirley did this because she realized she needed a more secure and scalable hosting option (Garfinkel, 2020).

AWS-based configuration was made to comply with security and dependability best practices in response to these difficulties. The completion of the implementation, website security, and confirming that the query page efficiently provides data were needed to be focused on. By improving the website's functionality and fortifying its defenses against any security breaches, this program hopes to better provide the research community with invaluable data resources.

1.2 Problem Statement

The website which helps social science scholars get the necessary information related to world development is posing serious problems for the Example Social Research Organization. The website's current infrastructure—hosted by a for-profit company—has been attacked with ransomware recently and is unable to handle the increased amount of traffic. This means that the company must move to Amazon Web Services (AWS) for improvement in the website's scalability, security, and performance. To ensure that the website continues to effectively satisfy researchers' data requirements while reducing future security threats, it is required to test the implementation on AWS and resolve security problems (Gupta & Mittal, 2022).

1.3 Objective

Based on the above problem statement, the solution to be implemented will have the following objectives:

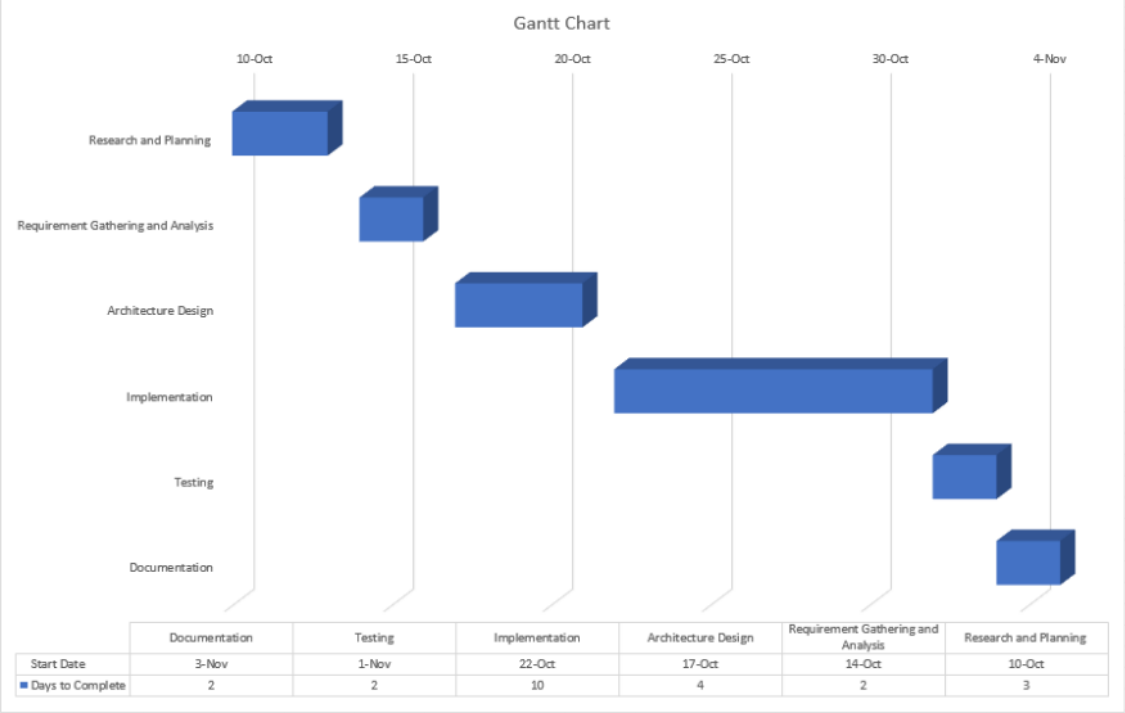
- To create and implement highly secure, highly available and highly scalable AWS deployment architecture

- To design and deploy a secure hosting environment for the MySQL database and create secure access controls
- To implement load balancer and auto scaling for high availability as well as configure PHP website to allow anonymous access for web users

1.4 Project Scope

The project's main scope is to provide secure access to the website to both the root users and visitors with the resources optimized using an EC2 instance. Along with that, to ensure cost-effective and responsive web services, the system's availability, securely managing data, and implementing automatic scaling is also focused on.

2 Project Plan



	Documentation	Testing	Implementation	Architecture Design	Requirement Gathering and Analysis	Research and Planning
Start Date	3-Nov	1-Nov	22-Oct	17-Oct	14-Oct	10-Oct
Days to Complete	2	2	10	4	2	3

Figure 2: Gantt Chart

3 Proposed Cloud Architecture

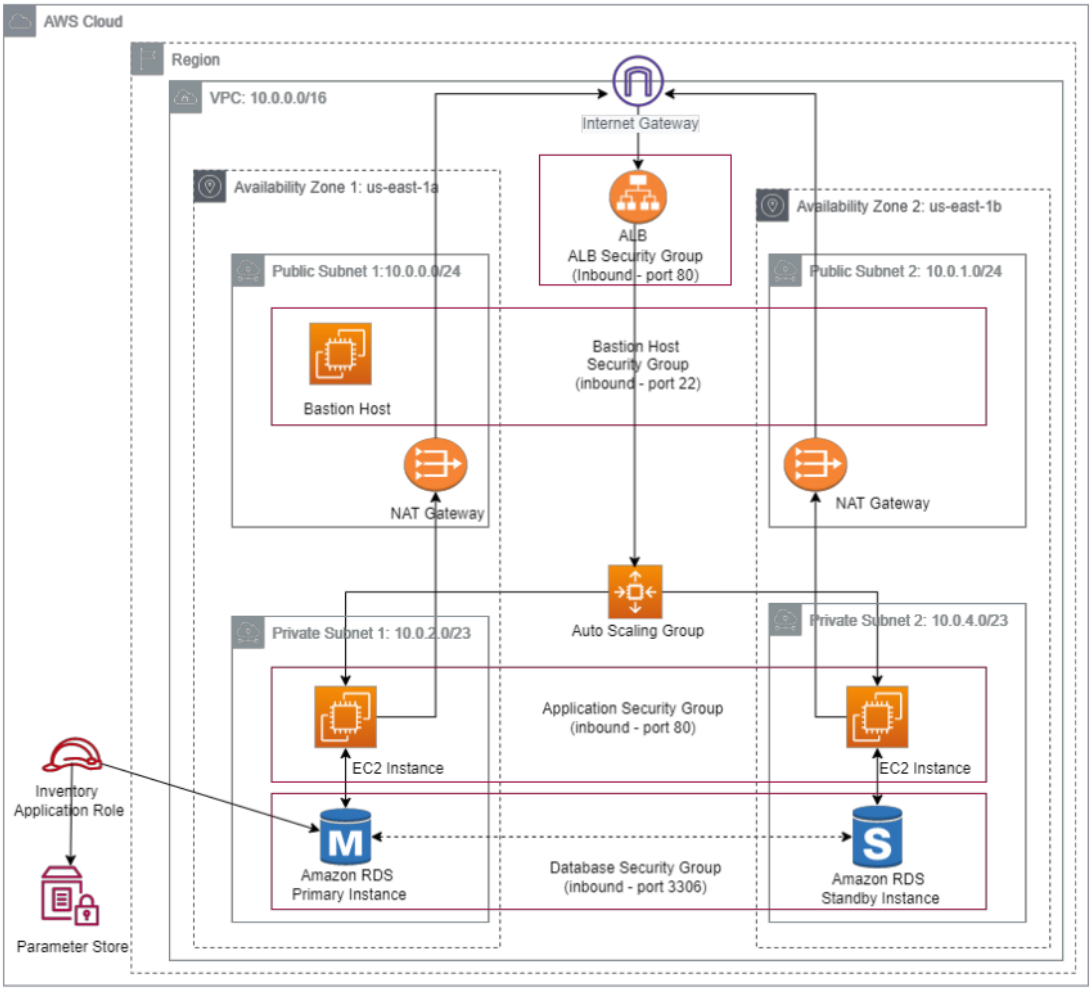


Figure 3: AWS Architecture

The above cloud architecture is a two-tier architecture which consists of the different web applications such as NAT gateways, load balancer, bastion hosts, along with private subnet. The following tier shows the application of the code instance on an Amazon RDS database instance. The load balancer used in this application helps to transport the incoming traffic within the application tier instances. The used bastion hosts are utilized for accessing the private subnet which is available outside the application. The used NAT gateway helps to access the internet for the application tier instances. The use of this implementation helps to improve the security of the

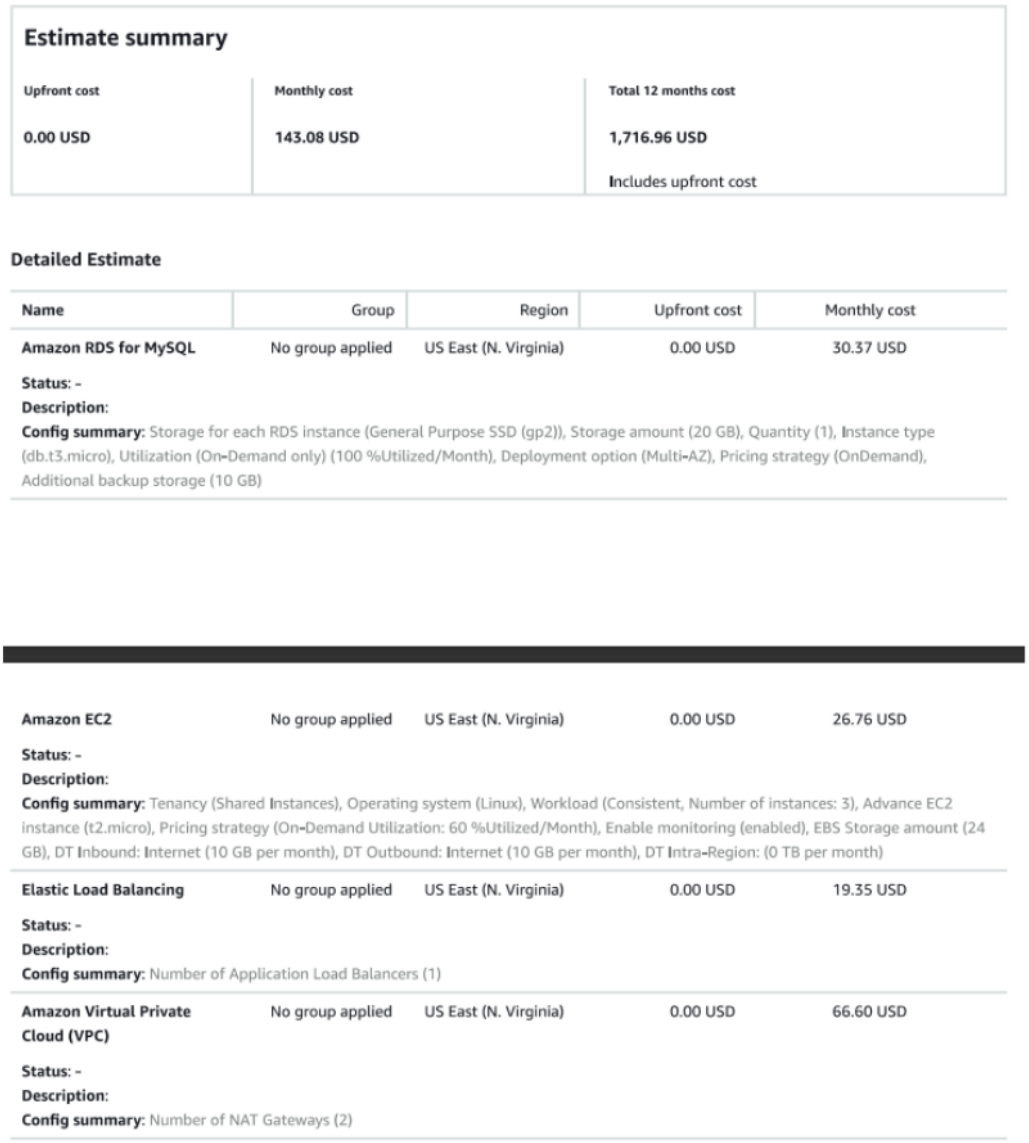
application. In the above architecture, the used private subnet of IP (10.0.0.0/25) is used to contain the application tier instances and their database instances which is inaccessible by the public IP address. Along with that, the application has auto scaling group utilized which allows the availability of necessary tier instances to run for the traffic load to handle. From the architecture, the use of Inventory Application Role or IAM is utilized for the provision of necessary permissions to the application tier instances for them to access the Parameter Store, which is a necessary way that helps to store necessary parameters such as authentication keys and API keys. All these EC2 instances are used to run the application code in the system. Also, the incoming traffic of the system is authorized by the utilized Application Security Group which is the inbound port 80. Along with that the Amazon RDS Standby Instance which is a duplicate instance of the Primary Instance which is used in case of the Primary Instance failure (Porald, 2021).

Overall, the above AWS cloud architecture diagram is well designed and has a secured web application architecture with the use of the following components:

- Load Balancer (ALB)
- Bastian Hosts
- NAT Gateways
- Private subnet (10.0.0.0/25)
- Auto Scaling Group
- Inventory Application Role
- Parameter Store
- EC2 instances
- Application Security Group (Inbound Port 80)
- Amazon RDS Primary Instance
- Database Security Group (Inbound Port 3306)
- Amazon RDS Standby Instance

Hence, the above architecture provides a secured and well-designed two-tier AWS cloud architecture with the use of different application tier and a database tier, with load balancers, bastion host, NAT gateways, and a private subnet to host the web application (Varia, 2022).

4 Cost Estimation



Estimate summary

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	143.08 USD	1,716.96 USD Includes upfront cost

Detailed Estimate

Name	Group	Region	Upfront cost	Monthly cost
Amazon RDS for MySQL	No group applied	US East (N. Virginia)	0.00 USD	30.37 USD
Status: - **Description:** **Config summary:** Storage for each RDS instance (General Purpose SSD (gp2)), Storage amount (20 GB), Quantity (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand), Additional backup storage (10 GB)				
Amazon EC2	No group applied	US East (N. Virginia)	0.00 USD	26.76 USD
Status: - **Description:** **Config summary:** Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 3), Advance EC2 instance (t2.micro), Pricing strategy (On-Demand Utilization: 60 %Utilized/Month), Enable monitoring (enabled), EBS Storage amount (24 GB), DT Inbound: Internet (10 GB per month), DT Outbound: Internet (10 GB per month), DT Intra-Region: (0 TB per month)				
Elastic Load Balancing	No group applied	US East (N. Virginia)	0.00 USD	19.35 USD
Status: - **Description:** **Config summary:** Number of Application Load Balancers (1)				
Amazon Virtual Private Cloud (VPC)	No group applied	US East (N. Virginia)	0.00 USD	66.60 USD
Status: - **Description:** **Config summary:** Number of NAT Gateways (2)				

Figure 4: AWS Resource Cost Estimation

It's necessary to take into account the expenses of hosting our program on Amazon Web Services (AWS) when we begin our capstone project. AWS provides a range of customizable pricing choices to suit our project's requirements. Cost optimization and expenditure minimization may be achieved by comprehending the price model and choosing the appropriate pricing choice for the project. The project involves EC2 services, AWS support, NAT Gateway, and RDS services. We have supplied an expected cost allocation of the services utilized in the project. It's crucial to take into account the expenses of hosting our software on Amazon Web Services (AWS) when we begin our capstone project (Huihong, 2022).

5 Implementation Flowchart

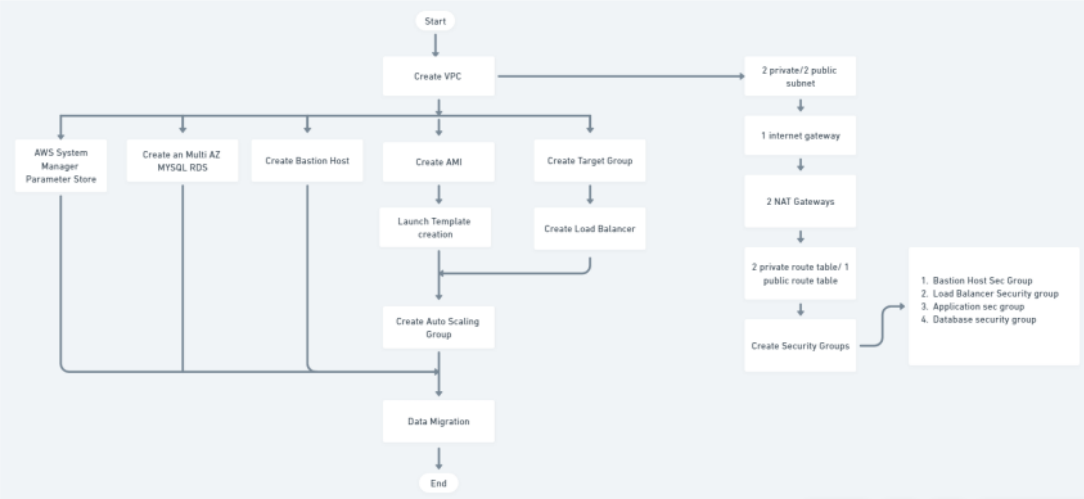


Figure 5: Implementation Flowchart

The process of creating an incredibly scalable and accessible website on AWS is depicted in the flowchart above. Creating a VPC with two private and two open subnets, a web portal, and a NAT portal is the first step in the process. At that moment, to provide secure access to the VPC, a Bastion Have is created inside the open subnet. Additionally, an EC2 occasion that is already in place and created with the site application is used to create an AMI. After that, a dispatch layout is created using the AMI to show the site instances' arrangement options, security gather, and event type. Next, using the Dispatch Format, an Auto Scaling Bunch is created to scale the site occurrences up or down in response to requests. To provide high availability, an RDS MySQL database is created in a Multi AZ configuration for the website. The database is designed with a

security group to restrict access to it. A Stack Balancer is set up inside the open subnet to make the site publicly accessible. For the Stack Balancer to allow web activity, a security group is created. Finally, a target group is created to register the load balancer's site occurrences. Based on the target group, the Stack Balancer will then communicate activities to the site instances (Zhong, 2021).

6 Implementation

6.1 VPC

A VPC is a virtual network that is exclusive to your Amazon Web Services (AWS) account. It is logically isolated from other AWS Cloud virtual networks (Beach, Armentrout, Bozo, & Tsouris, 2019). In the architecture, VPC is created to launch AWS resources such as Amazon EC2, Amazon RDS etc. into the VPC.

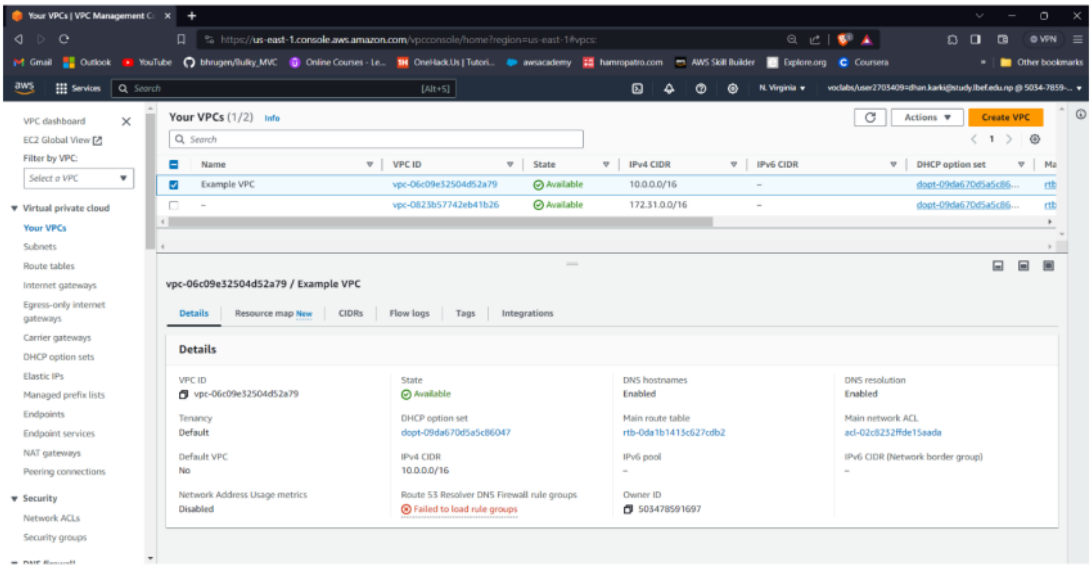


Figure 6: Preview of VPC

6.1.1 Subnet

A subnet represents a subset of IP addresses within the VPC and it's the designated space where AWS resources can be deployed. For resources that require internet access, selecting a public subnet is the appropriate choice while a public subnet is the preferred option of resources that need to be isolated from the internet.

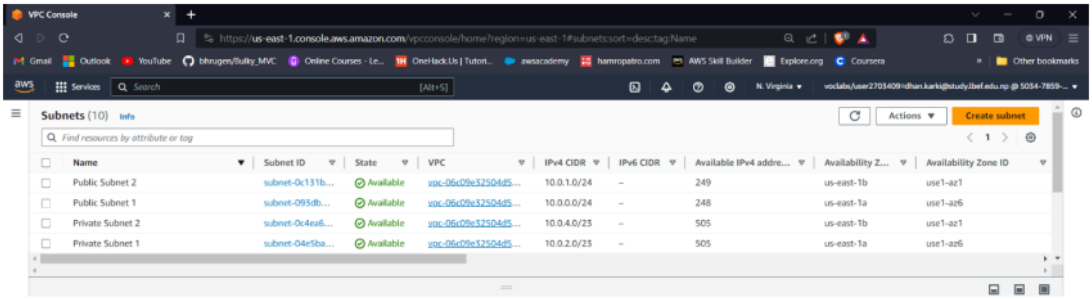


Figure 7: List of Subnets

6.1.2 Internet Gateway

An internet gateway is a critical component of a VPC architecture built for high availability, scalability, and redundancy. It is used with a purpose to provide a reliable communication between instances inside a VPC and the rest of the internet without imposing any availability or bandwidth constraints.

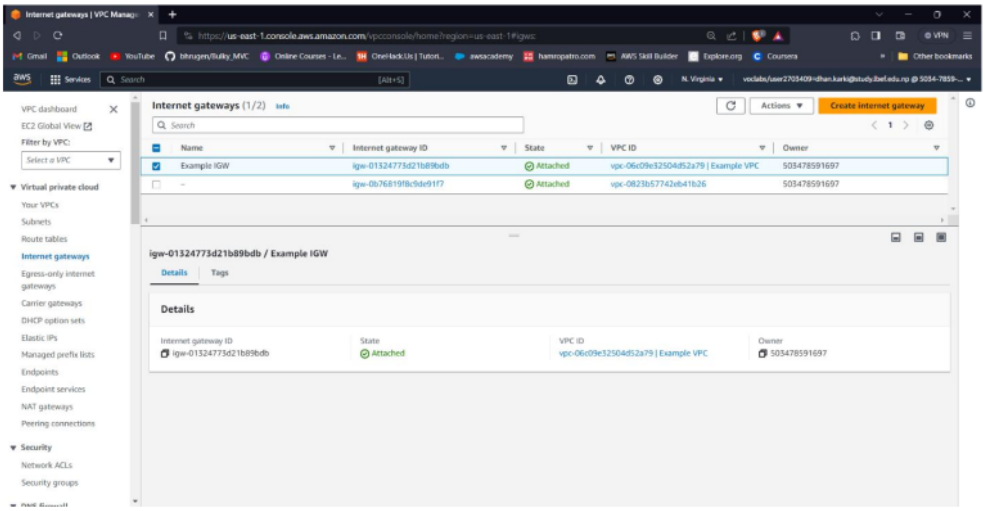


Figure 8: "Example IGW" Internet Gateway

6.1.3 NAT Gateway

In the architecture, the application servers operate on a private subnet. When these servers need to connect to the internet, such as for downloading data, their requests are routed through a Network Address Translation (NAT) gateway located in a publicly accessible subnet. The NAT

gateway serves as a means for instances lacking public IP addresses to gain internet access (AWS, 2022).

6.1.3.1 Creating First NAT gateway

The first NAT gateway is created on the Public Subnet 1, and it will allow application server running in the Private Subnet 1 to have internet access.

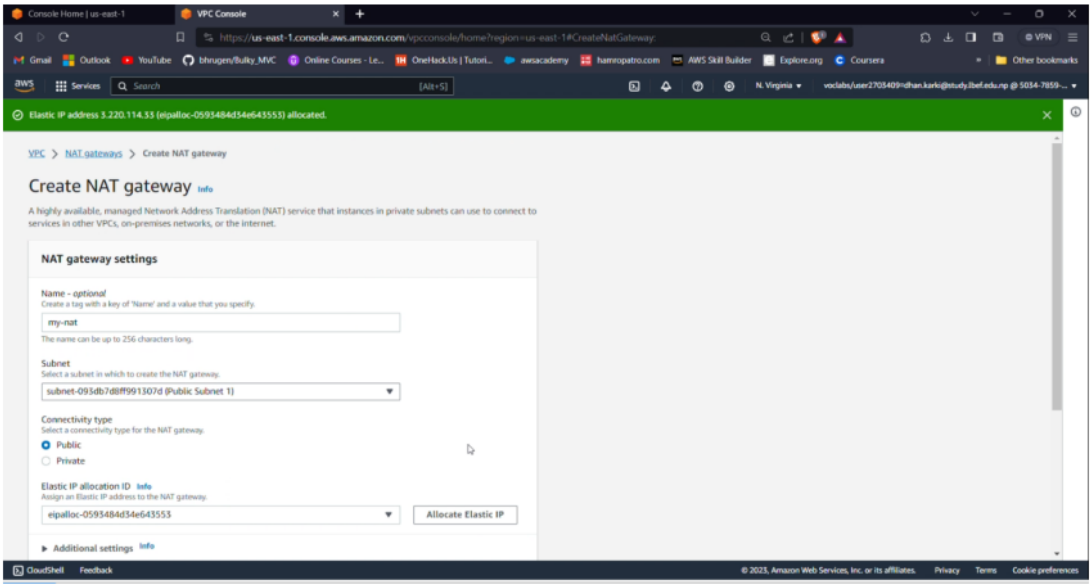


Figure 9: Creating first NAT gateway

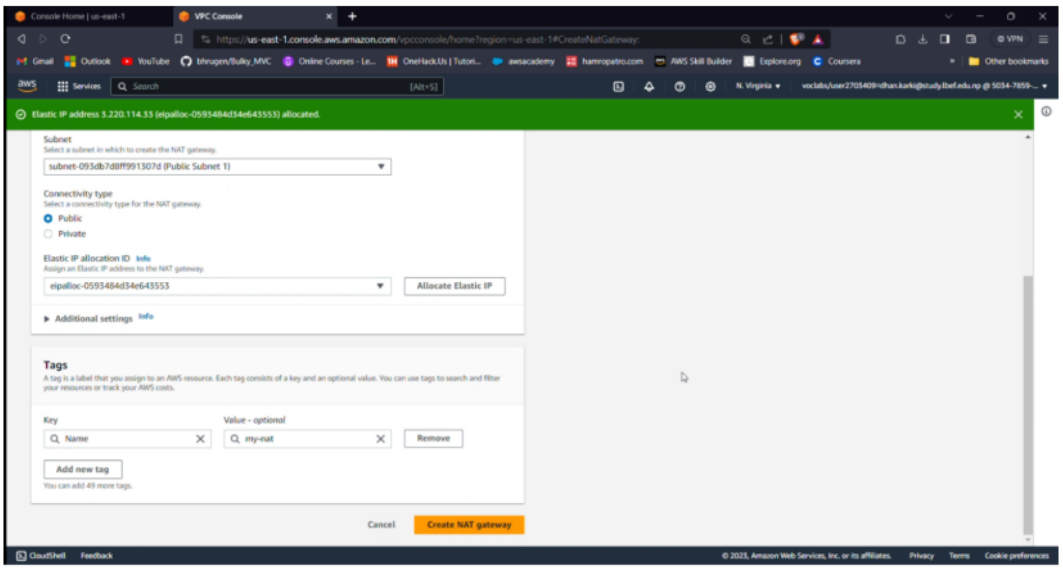


Figure 10: Creating first NAT gateway

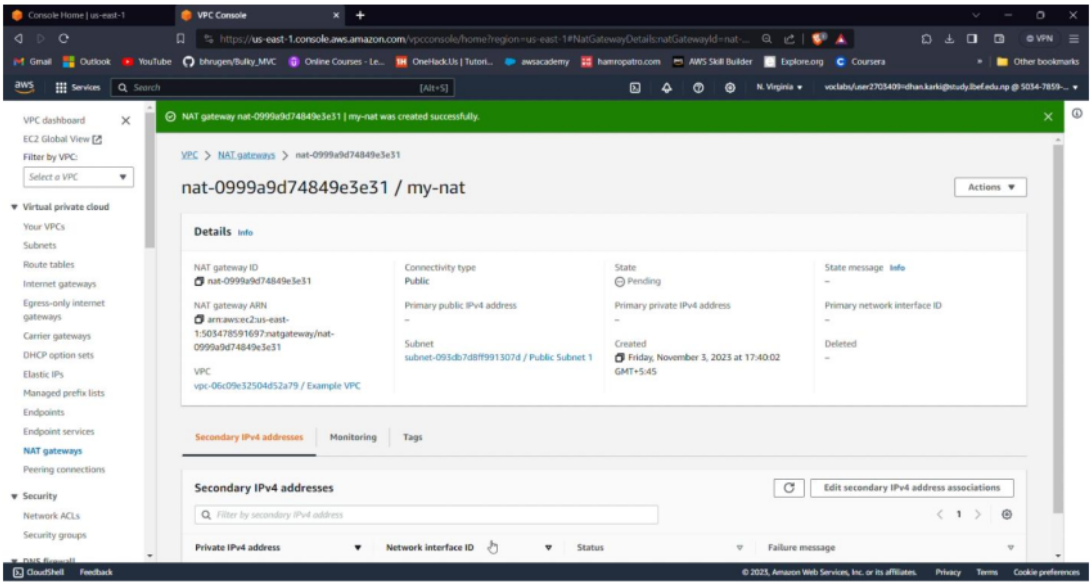


Figure 11: Preview of first NAT gateway

To add NAT gateway to the route table, Private Route Table 1 needs to be modified.

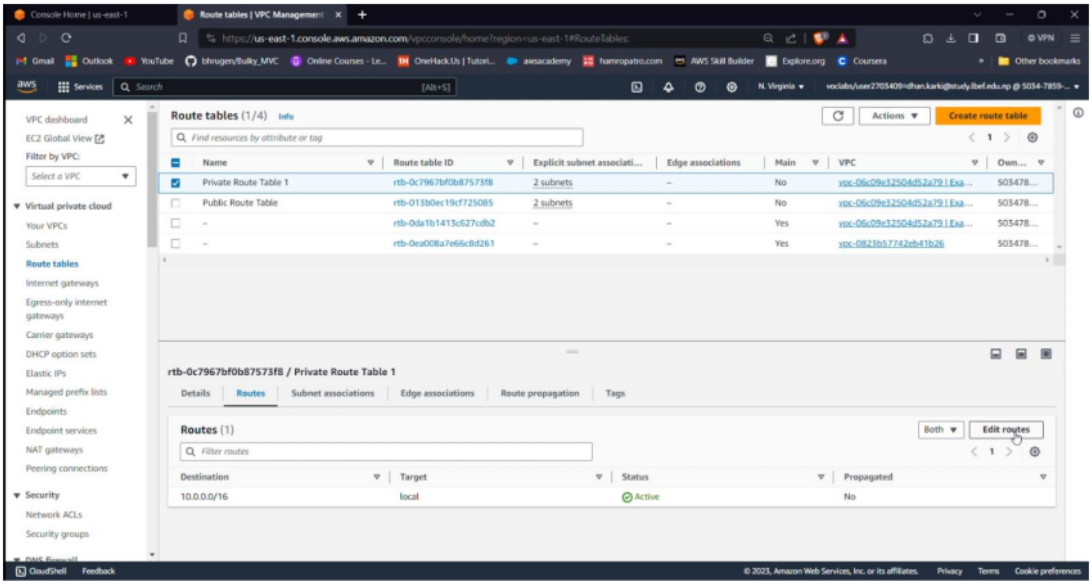


Figure 12: Routes in a Private Route Table 1

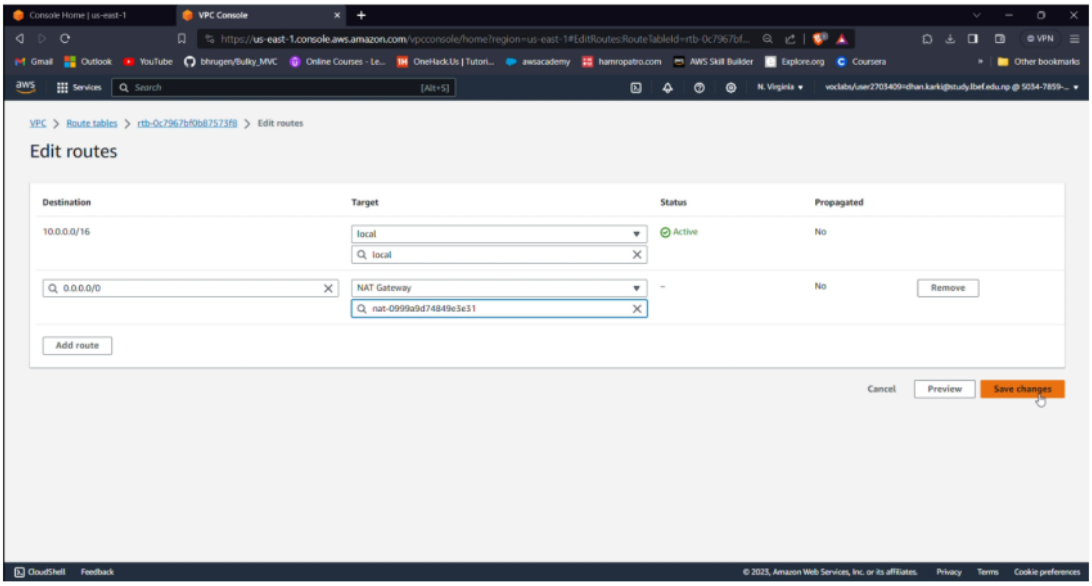


Figure 13: Adding NAT gateway to the route

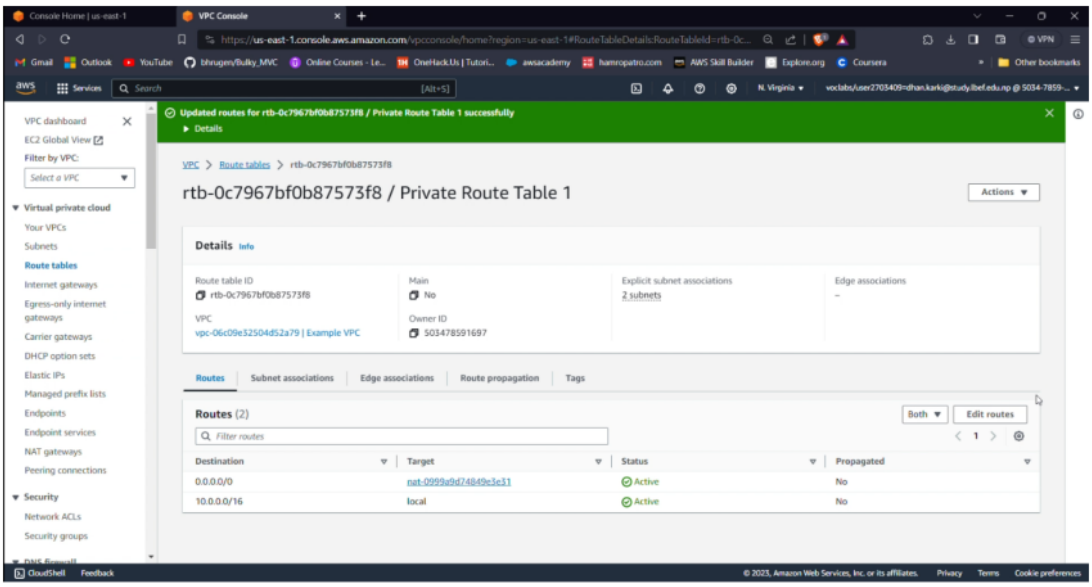


Figure 14: Private Route Table 1 after adding NAT gateway route

6.1.3.2 Create 2nd NAT gateway

If the first Availability Zone fails, the application servers will not be able to communicate with the internet. So, to ensure high availability, another NAT gateway needs to be launched in the other Availability Zone i.e., us-east-1b.

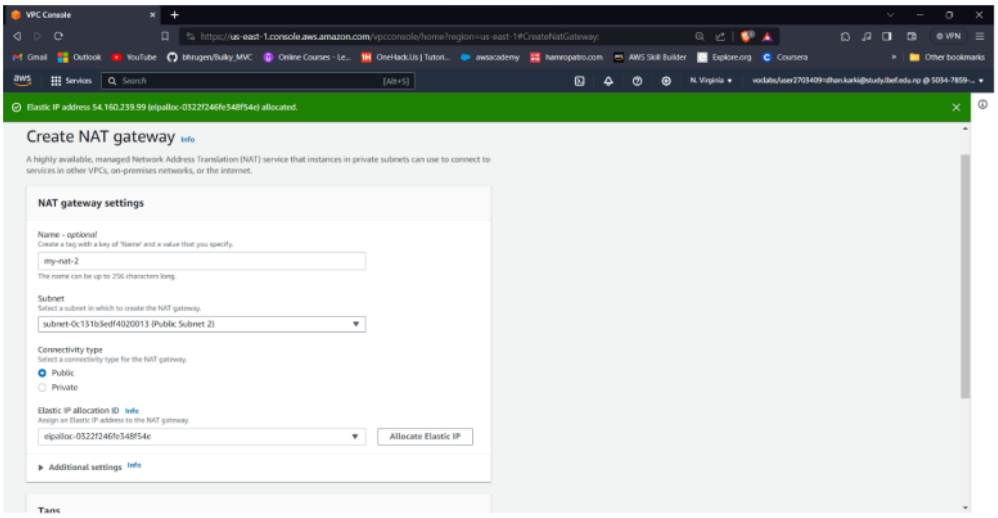


Figure 15: Creating second NAT gateway

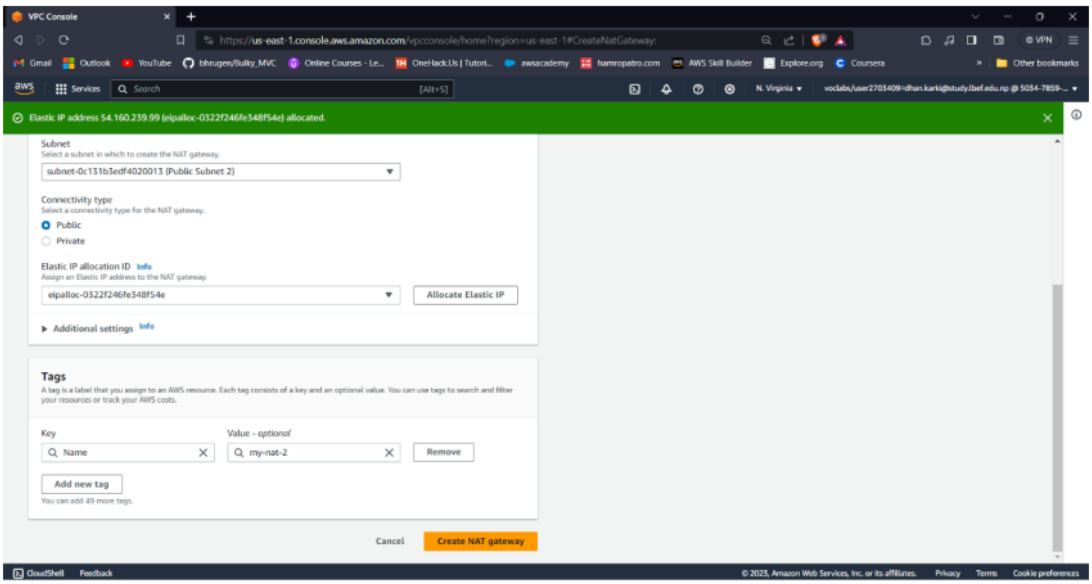


Figure 16: Creating second NAT gateway

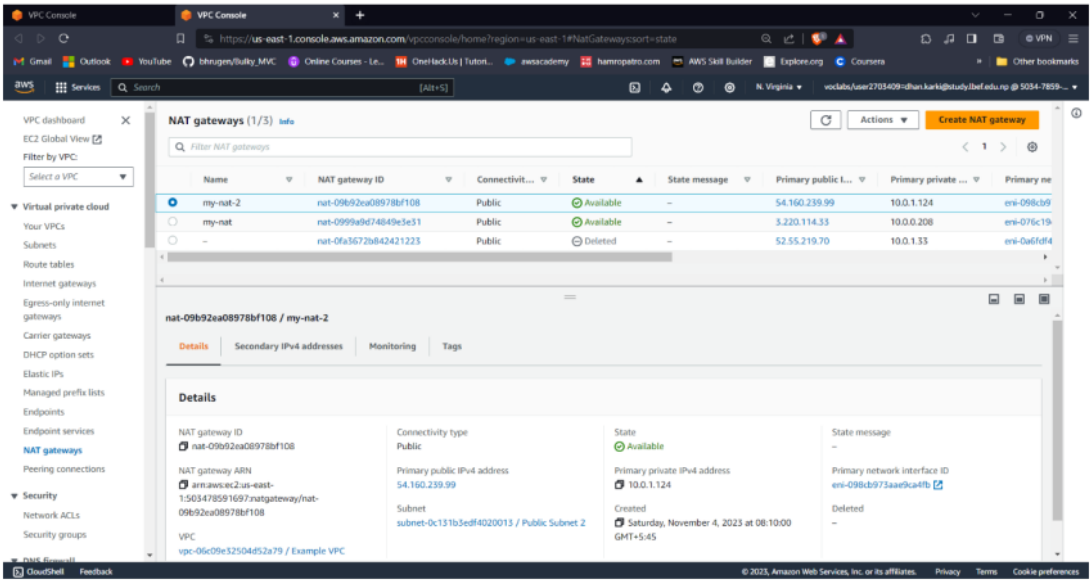


Figure 17: Second NAT gateway preview

Next, to control the routing of network traffic in the second AZs within VPC, Private Route table 2 needs to be created.

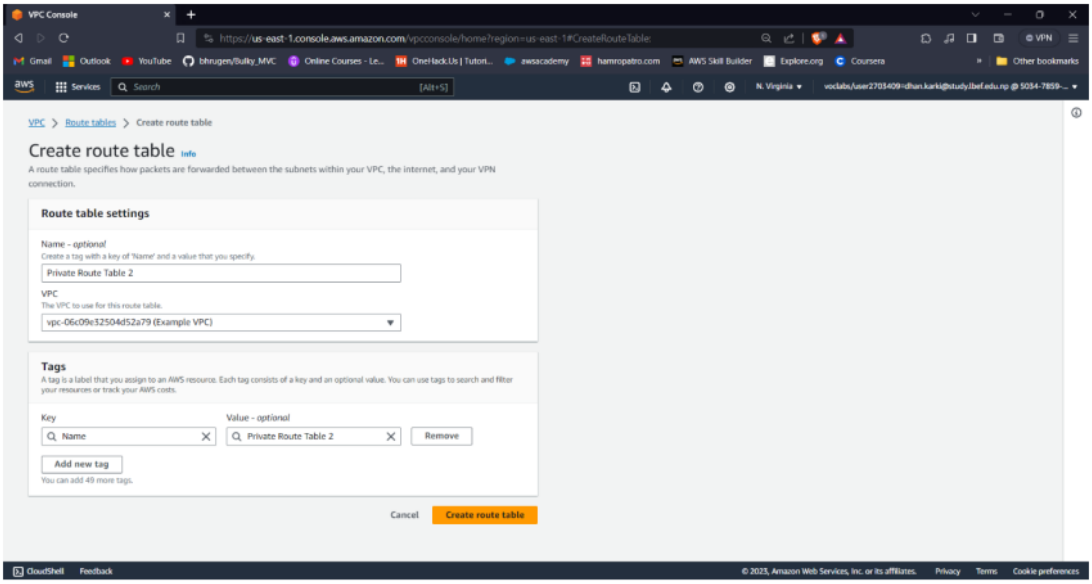


Figure 18: Creating New Private Route Table

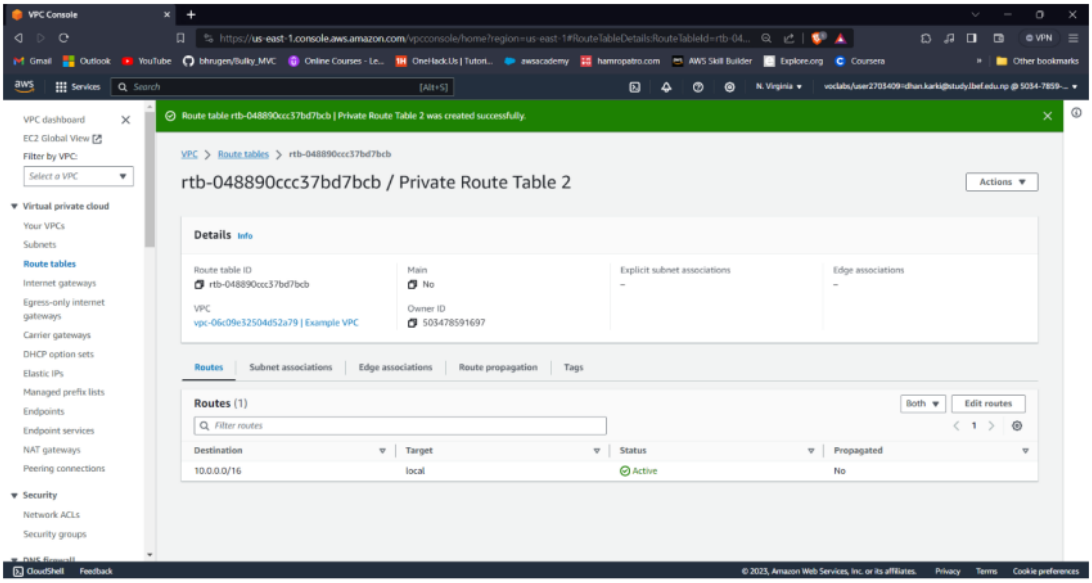


Figure 19: Preview of Private Route Table 2

Next add, edit a route to send internet-bound traffic through the new NAT gateway i.e., my-nat-2.

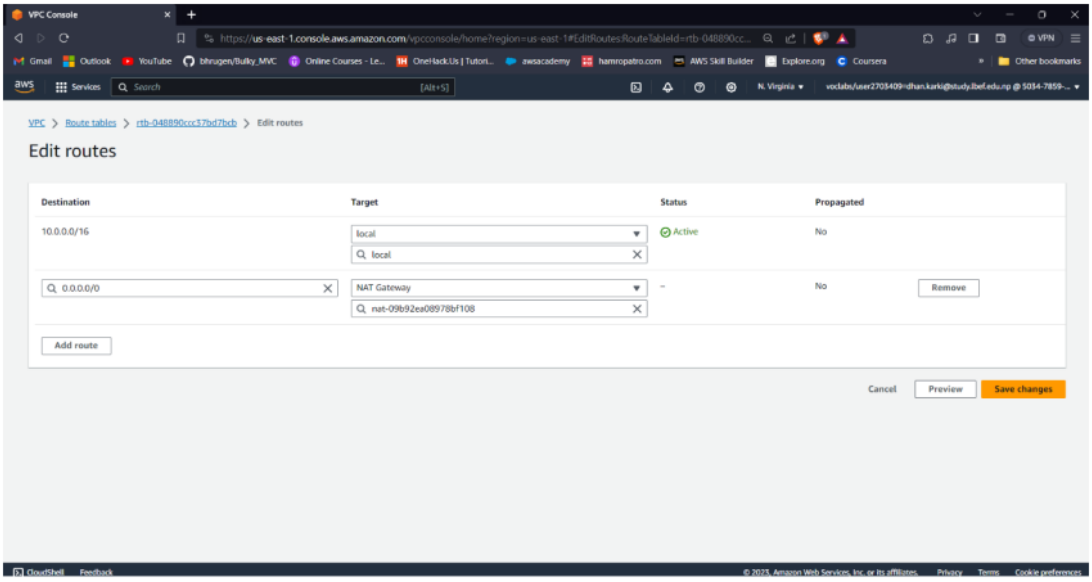


Figure 20: Adding second NAT to newly created Private Route Table

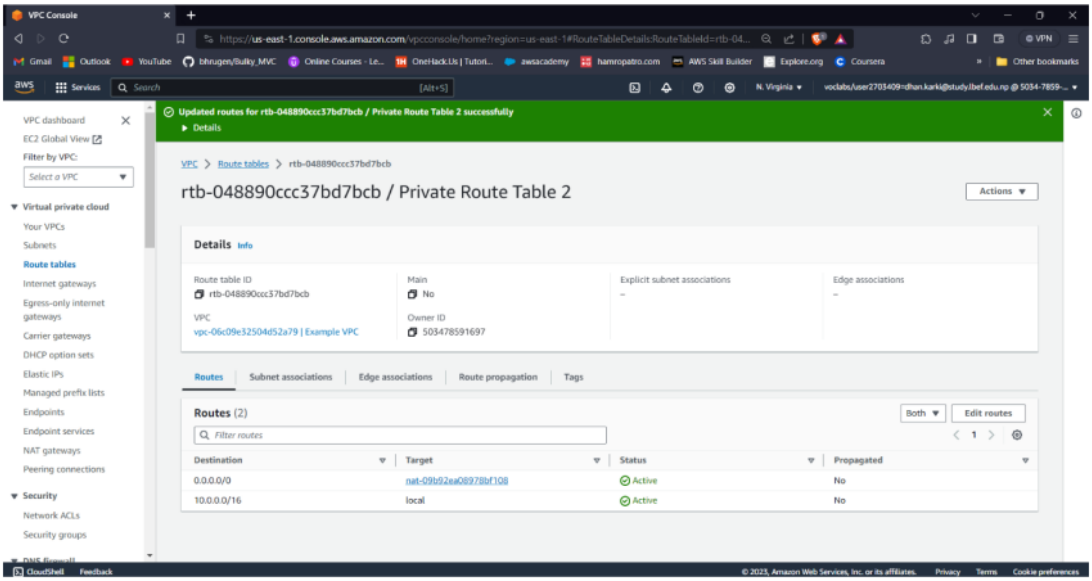


Figure 21: After adding route

To send internet-bound traffic from Private Subnet 2 to the NAT gateway that is in the same (or us-east-1b) Availability Zone, subnet Private Subnet 2 needs to be associated in the Private Route table 2.

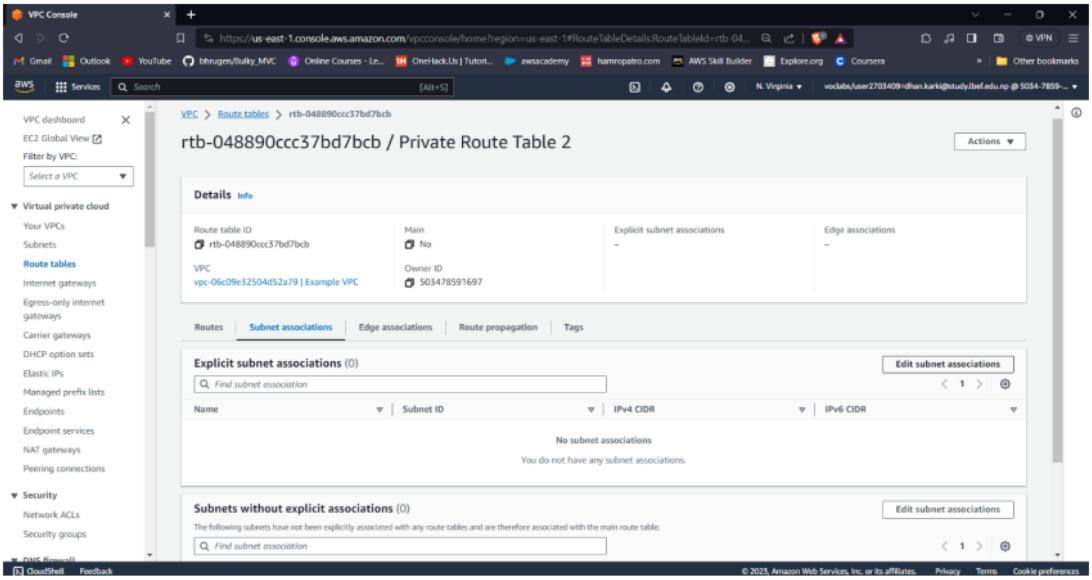


Figure 22: Private Route Table Subnet Association

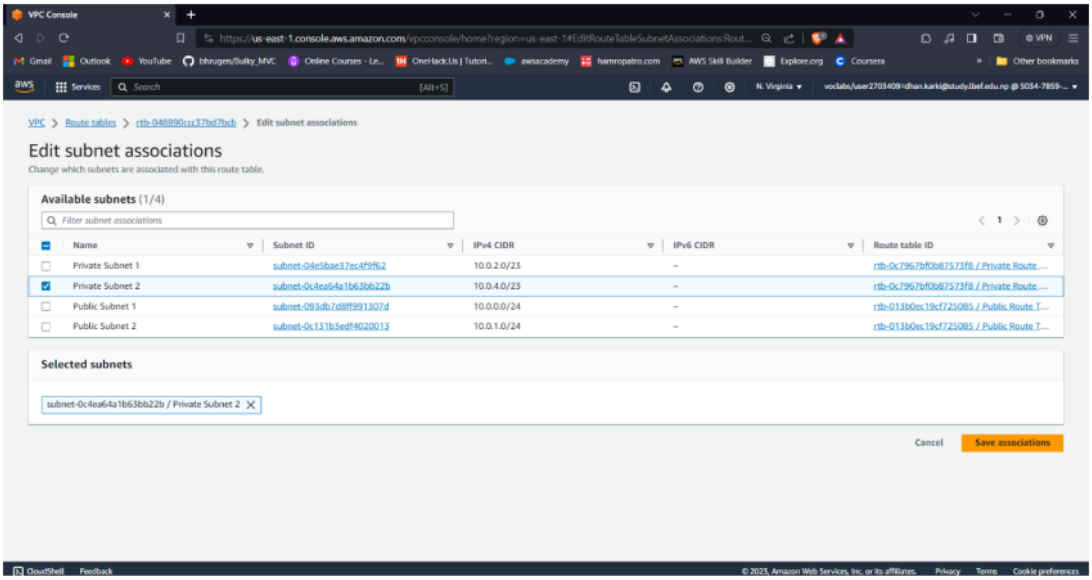


Figure 23: Associating Private Subnet 2 to Private Route Table 2

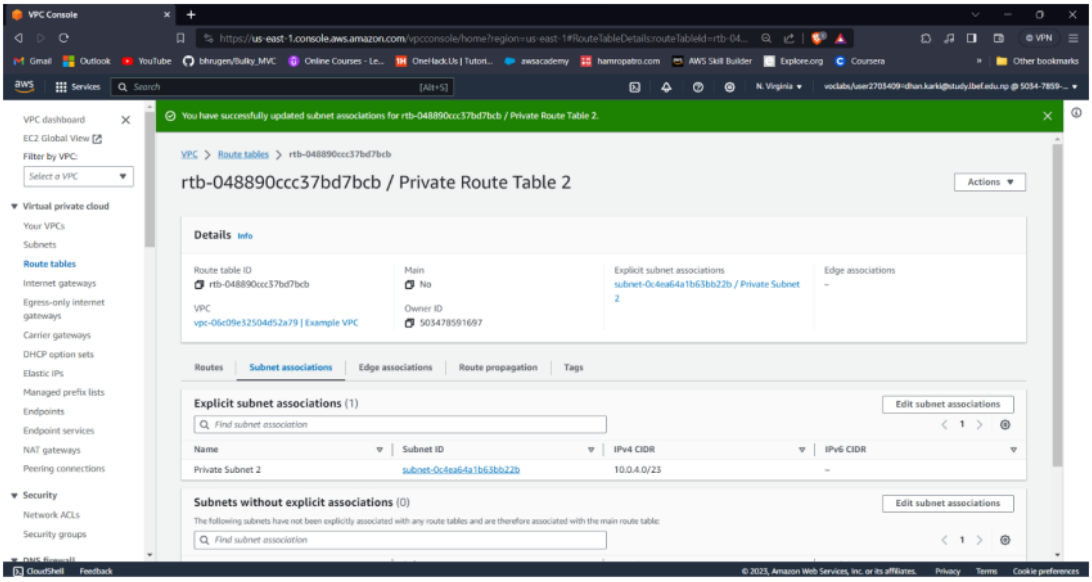


Figure 24: After Subnet Association

Now, NAT gateways are highly available. A failure in one Availability Zone will not impact traffic in the other Availability Zone.

6.1.4 Security Groups

Security groups are an important aspect of safeguarding AWS infrastructure since they limit network access and adhere to the concept of least privilege. It acts as virtual firewalls, regulating traffic to and from AWS services. These security groups enable defining rules for incoming and outgoing traffic, such as the sources, ports, and protocols that are permitted. They work on the "deny by default" approach, which means rules must be explicitly established to allow incoming connections.

There are four security groups created to achieve the security based on architecture:

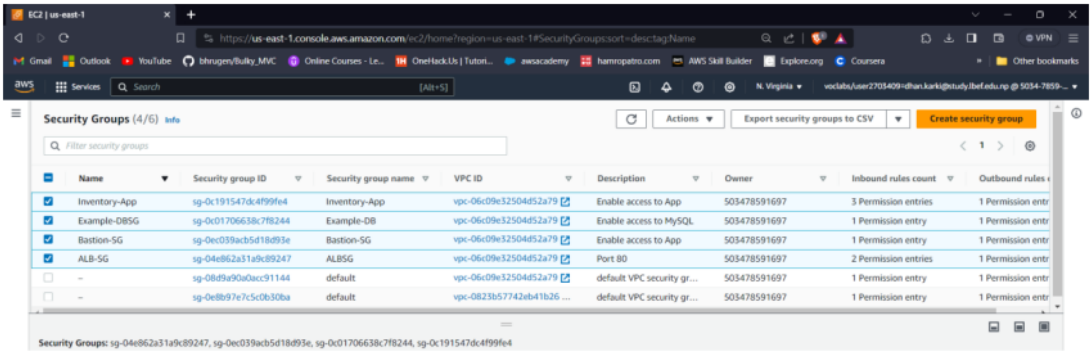


Figure 25: List of Security Groups

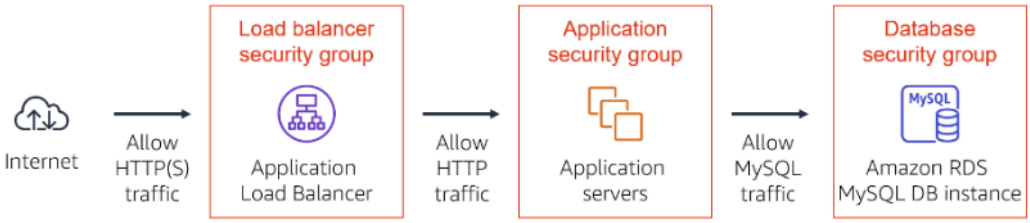


Figure 26: Traffic flow between different security groups

1. Application Security Group (Inventory-APP)

In this security group, Inbound traffic from Application Load Balancer (ALB) is authorized on ports 80 and 443 (HTTP and HTTPS) to provide external access to the application while outbound traffic rules are left as default. Also, Inbound rules is added to permit bastion host security group access the SSH port 22 for data migration purposes.

2. ALB Security Group (ALBSG)

Inbound internet traffic is permitted to reach Application Load Balancer (ALB) on ports 80 and 443 (HTTP and HTTPS), allowing users to access our application while outbound is left as default.

3. Database Security Group (Example-DBSG)

Inbound traffic from the Application Security Group (Inventory-APP) is permitted to access the database on port 3306, providing secure connection between the application and the database.

4. Bastion Security Group (Bastion-SG)

This security group's inbound rule permits secure remote access to the bastion server via SSH (port 22) from any IP addresses, enabling administrative operations and troubleshooting while

outbound has default rules allowing server to initiate outbound connection to other resources in the VPC.

6.2 Load Balancer

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs). There are three types of load balancer provided by AWS: Application, Gateway and Network Load balancer. In the architecture, one of its type, Application Load Balancer (ALB) is used since it can be used distribute incoming network traffic over different destinations, such as Amazon EC2 instances, containers, or IP addresses, inside a particular Availability Zone or across many Availability Zones and route HTTP and HTTPS (Layer 7) traffic effectively providing extensive functionality for application-level load balancing (Selvan, .M, Kuppuchamy, & Shanthi, 2021).

Creating Load balancer

In AWS, Application load balancer's creation process begins with selecting load balancer type as its name suggests.

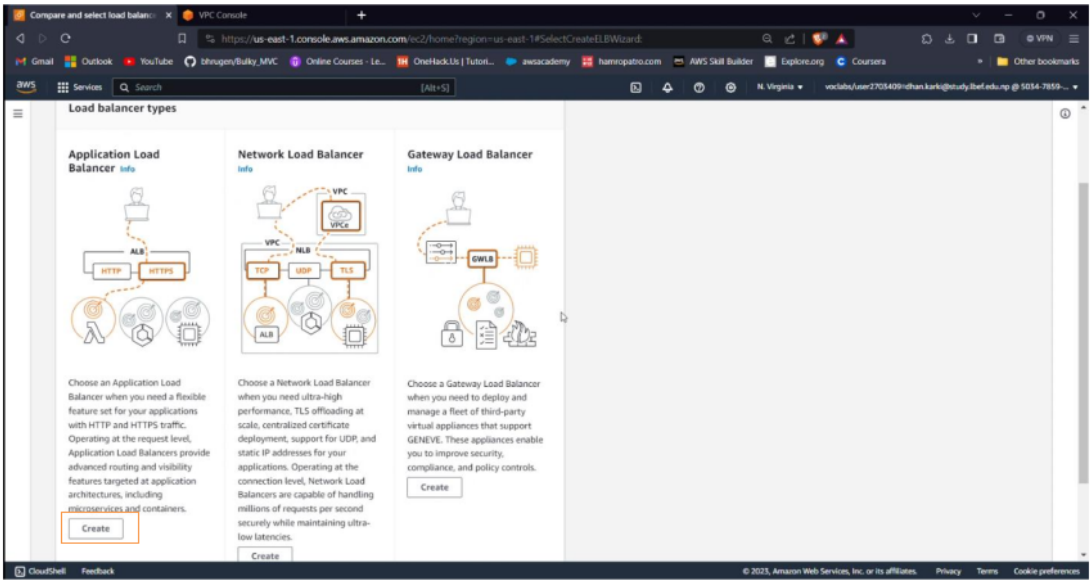


Figure 27: Load balancer types

Then, a name is given to the load balancer.

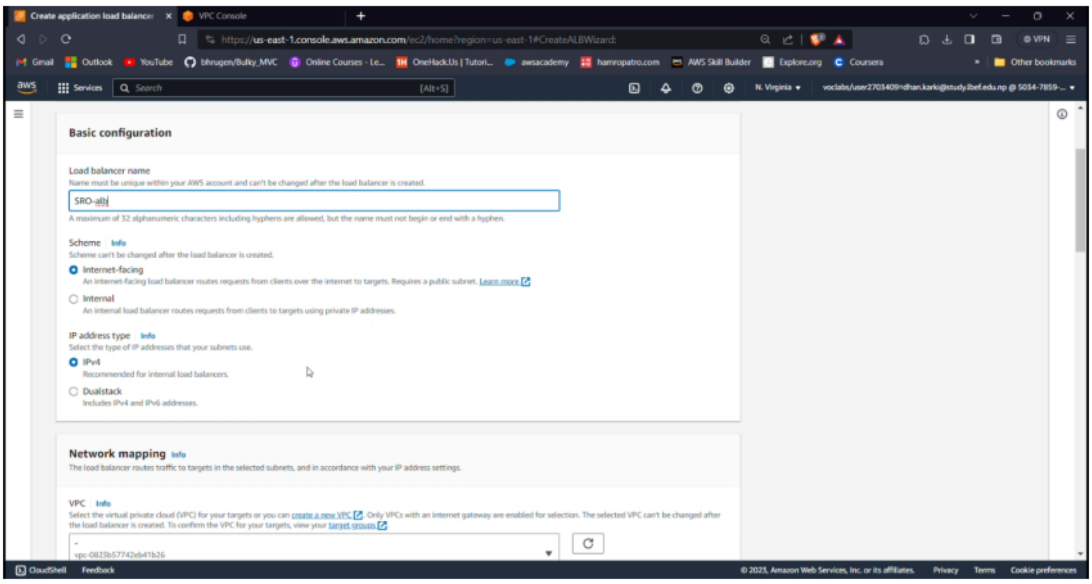


Figure 28: ALB's basic configuration

In the Network Mapping section, "Example-VPC" is selected as the VPC target and two AZs us-east-1a and us-east-1b are selected with Public Subnet 1 and Public Subnet 2 in each of them respectfully.

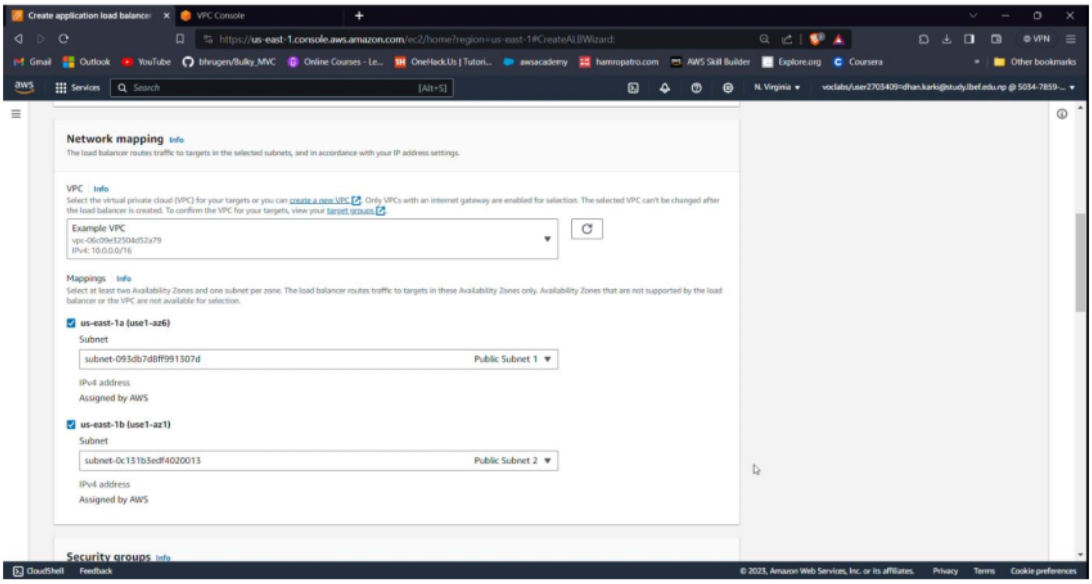


Figure 29: ALB's Network Mapping

Next, the security group named “ALBSG” is selected which has inbound traffic http port 80 and https port 443 configured.

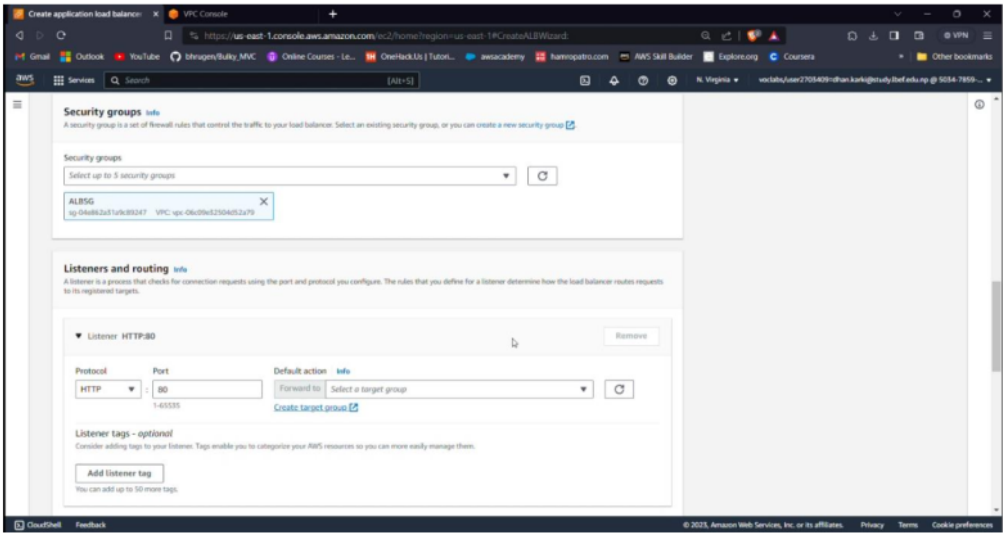


Figure 30: ALB's Security group

Next, a listener needs to define that determines how the load balancer routes request to its registered targets.

To create a target group, the target type as instances is chosen.

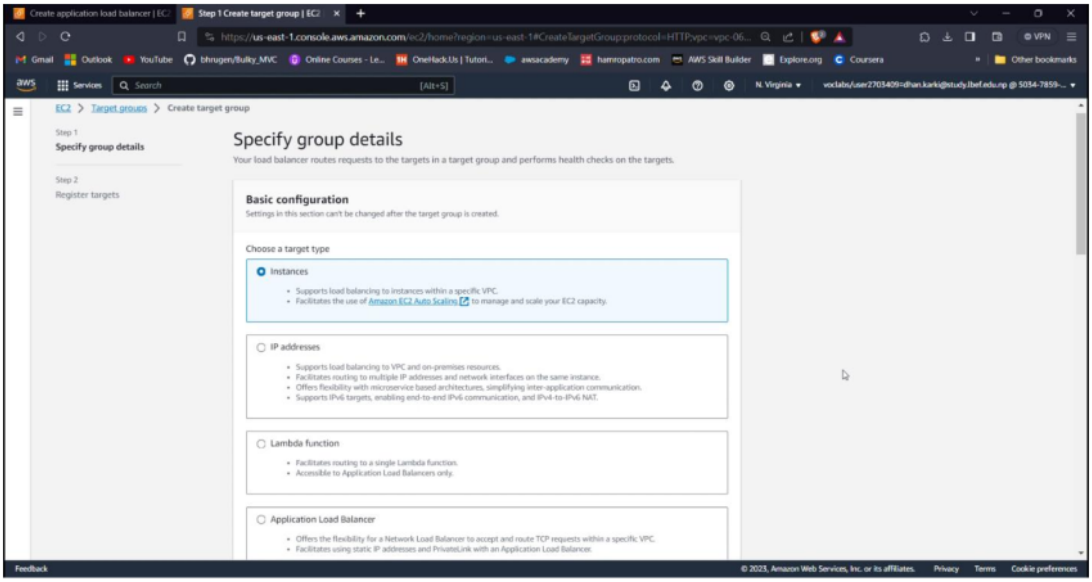


Figure 31: Target groups' target type

Next a target group is named 'SRO-TG' and VPC "Example-VPC" is selected.

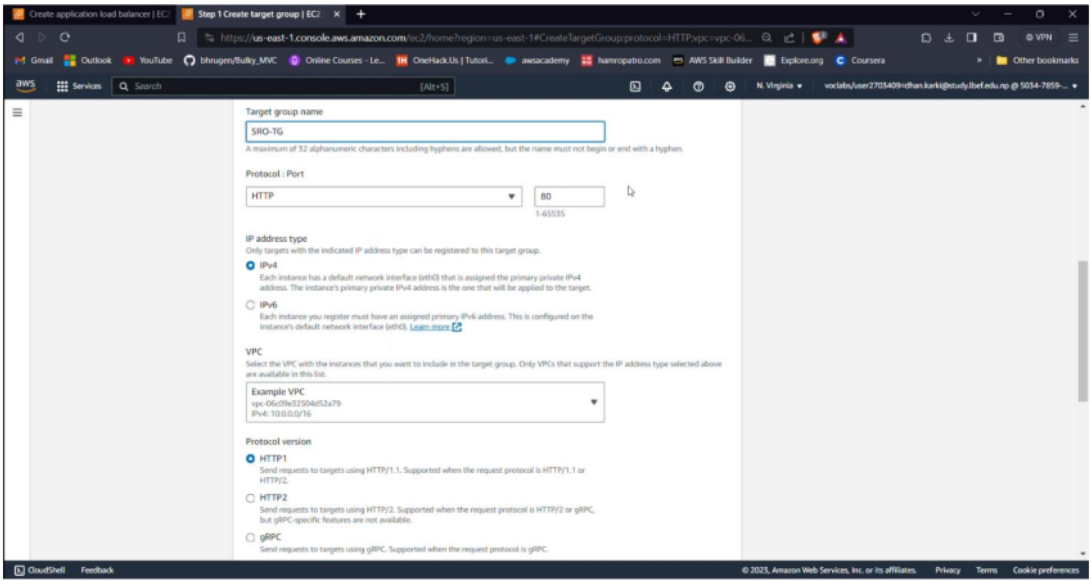


Figure 32: Target group basic configuration

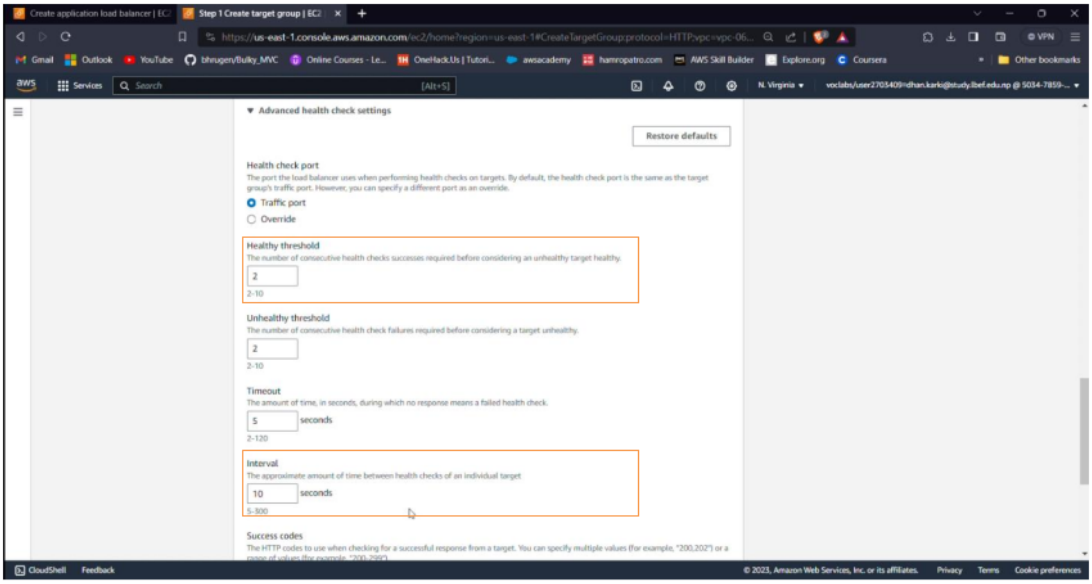


Figure 33: Target group advanced health check settings

The above health check parameters will result in the health check being conducted every 10 seconds, and if the instance replies successfully twice in a row, it will be declared healthy.

In the next step, register targets screen appears and there is no need to register any target (instances) because it will be automatically registered later on. Targets are the instances that will react to load balancer requests.

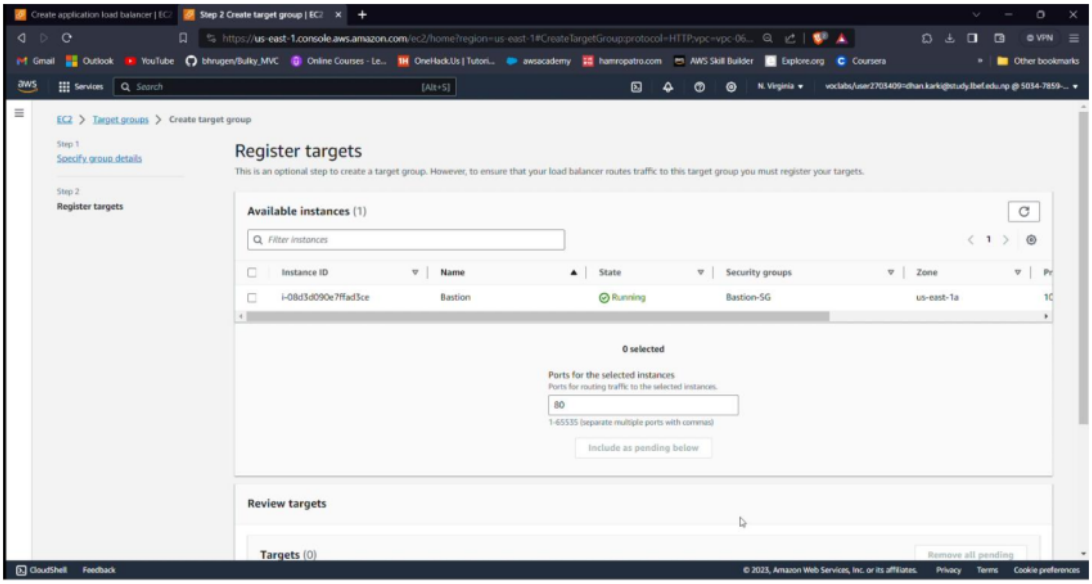


Figure 34: Register Target screen

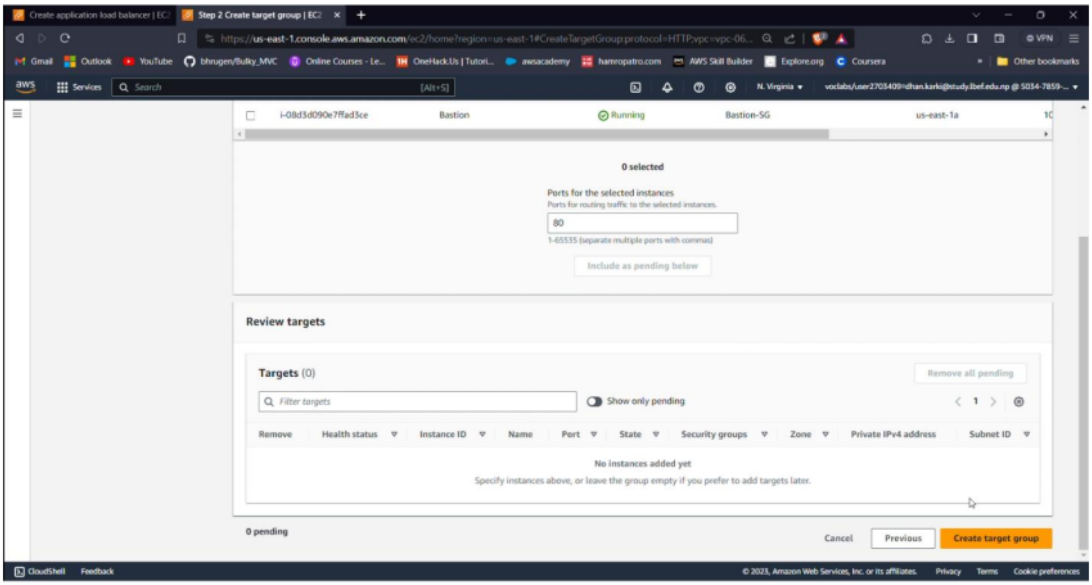


Figure 35: Review of registered targets

Skipping the registering target step and clicking on create target group button, target group is created with name “SRO-TG” and is not associated with any load balancer.

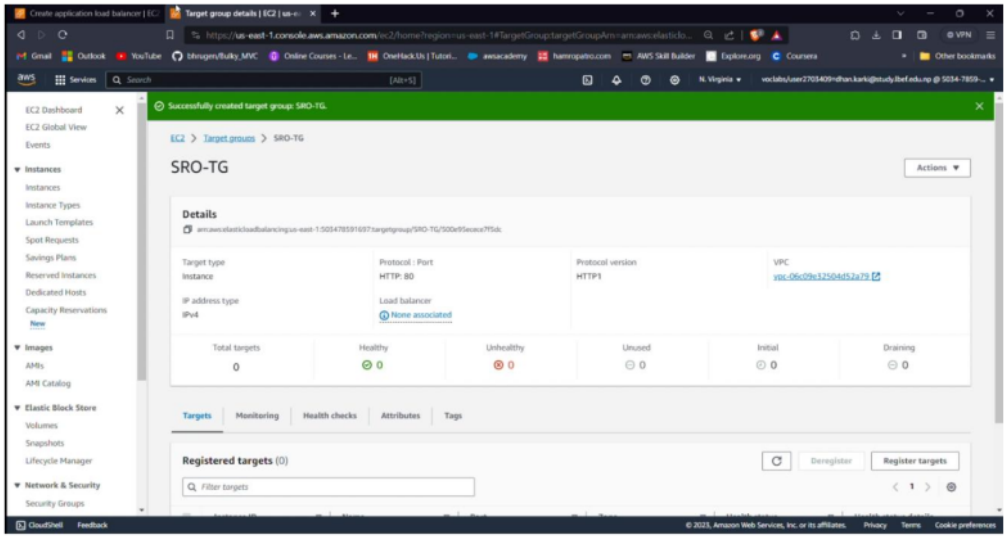


Figure 36: "SRO-TG" Target group preview

Returning to the browser tab where the load balancer has already started defining, the newly created target group is selected.

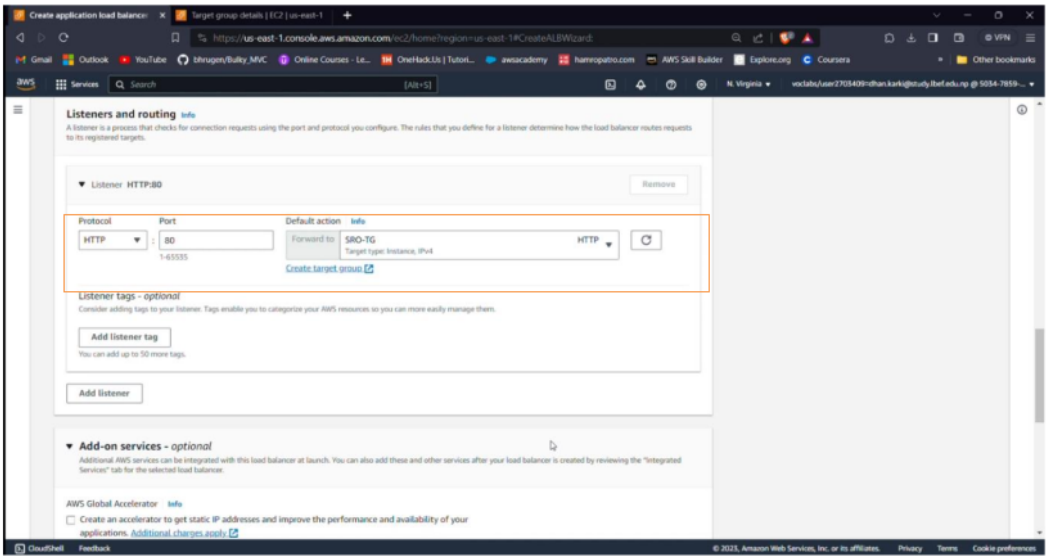


Figure 37: Adding Listener and Routing

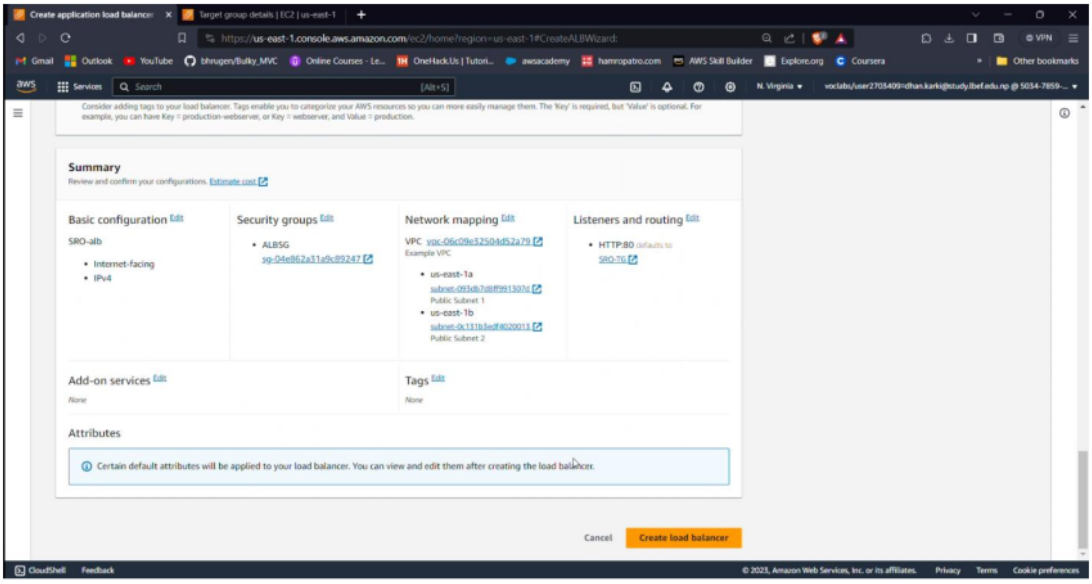


Figure 38: Load balancer summary

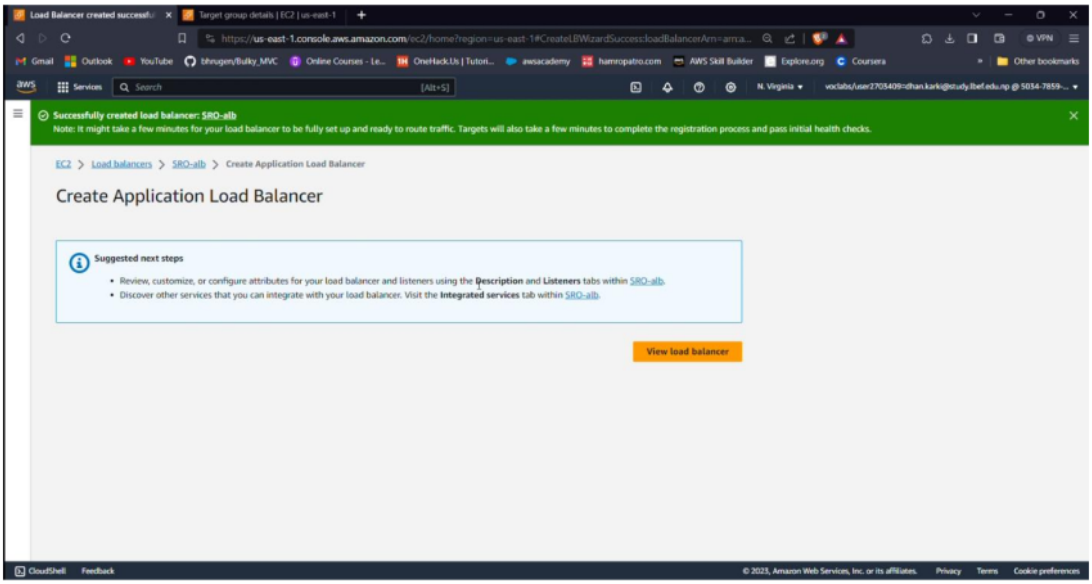


Figure 39: Successful creation of load balancer

After the load balancer configuration is defined, it takes time to create load balancer so it has shown its state provisioning.

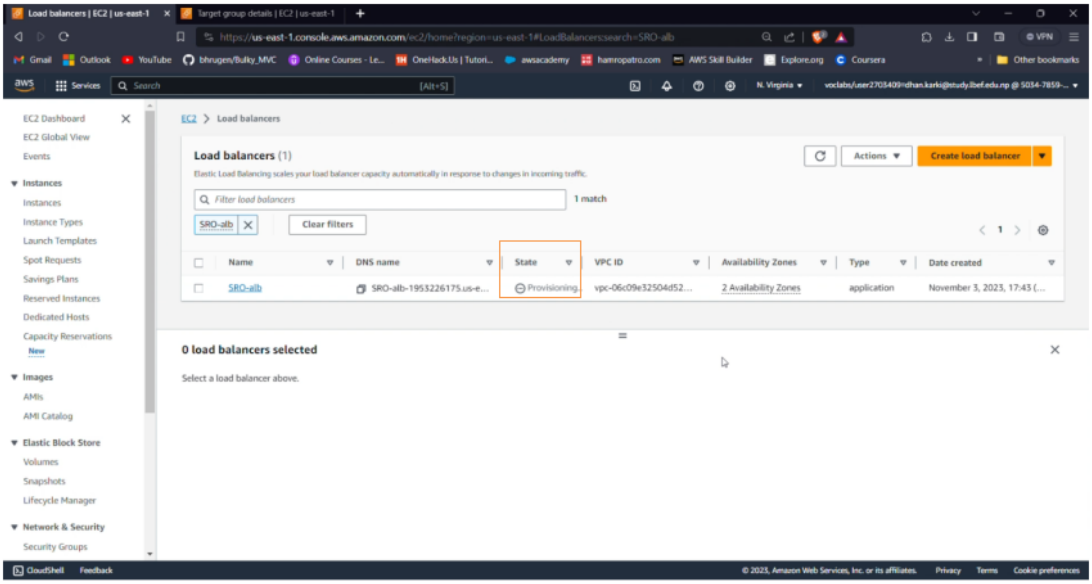


Figure 40: Available Load balancers

After some time, the state of the load balancer will be changed to Active state and then it is ready to be used.

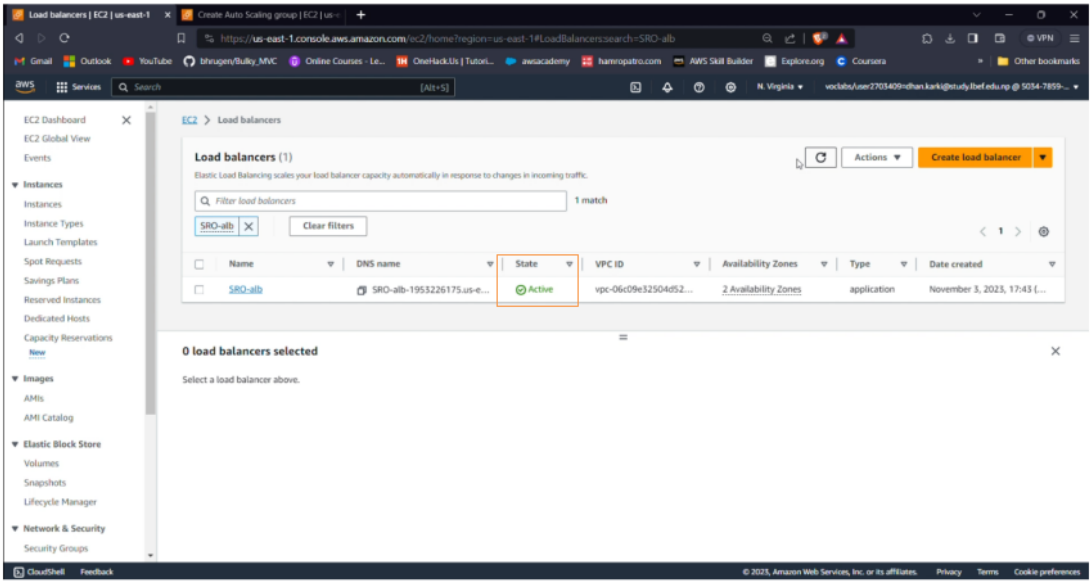


Figure 41: Available Active Load balancer

6.3 Auto Scaling Group

An Auto Scaling Group (ASG) is used to autonomously change server capacity and manage servers. ASG has the capacity to launch the creation of new servers or shut down existing ones in accordance with defined scaling parameters, ensuring that servers stay adaptive and continually available. ASG starts the deployment of a new server by utilizing the corresponding launch template or launch parameters (.M, Selvan, Kuppuchamy, & Shanthi, 2021).

Based on the architecture, the Auto Scaling group has the responsibility for distributing EC2 instances inside private subnets in accordance with a recommended security practice for application deployment. Private subnet instances are not immediately accessible from the internet. Instead, users send their queries to a load balancer, which sends them to the EC2 instances in the private subnets.

Creating Auto scaling Group

To start the process, the name is given to Auto Scaling Group as “SRO-ASG”. The launch template is created at the beginning of the lab session so to obtain it, it is required to click refresh icon on the launch template section. A launch template serves as a blueprint employed by an Auto Scaling group for initiating EC2 instances. It contains information regarding instances including elements like the AMI, instance type, security groups and key pair.

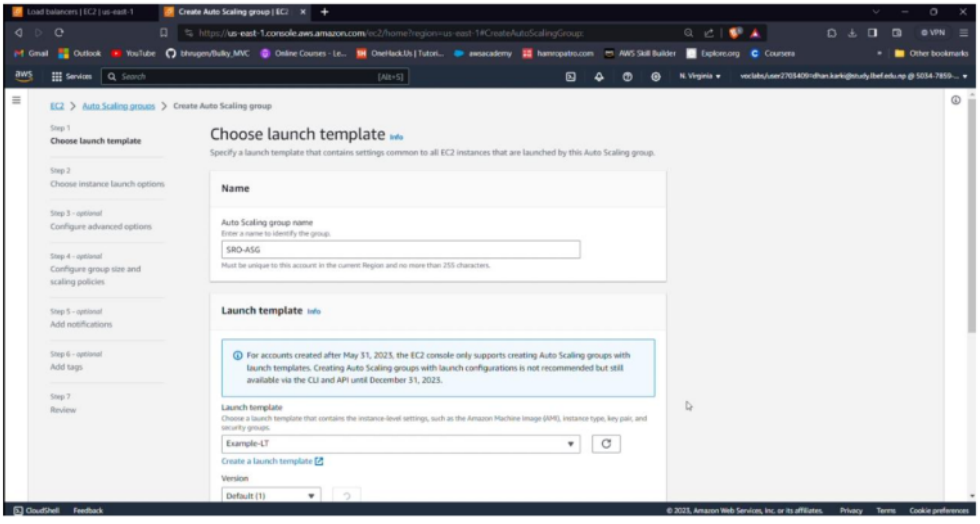


Figure 42: Choosing Launch Template

On clicking next, in the Network Configuration section, VPC “Example-VPC” is chosen and two AZs as “us-east-1a” and “us-east-1b” with private subnet are mapped to each of them respectively.

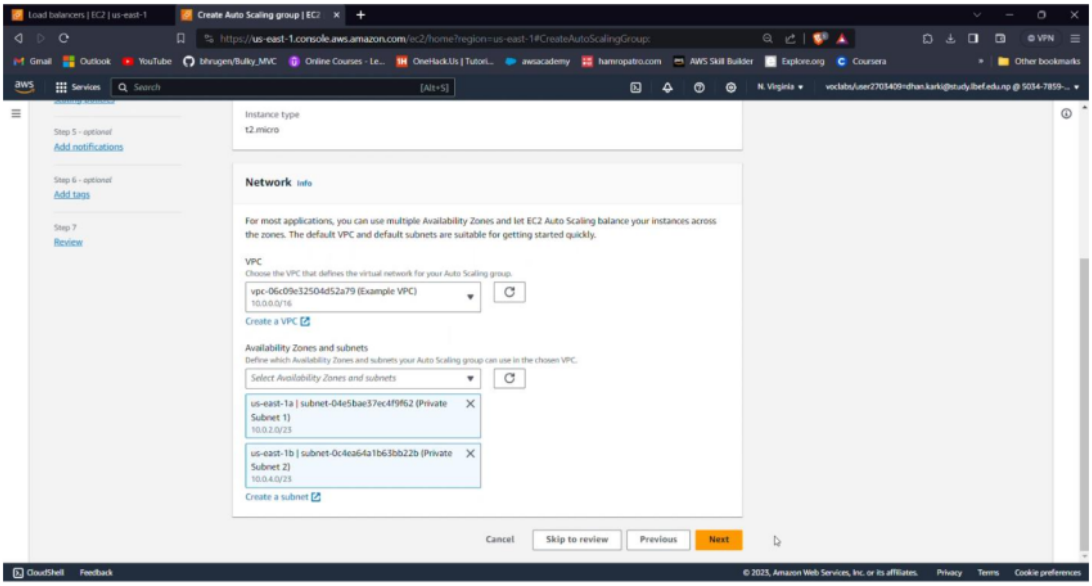


Figure 43: Choosing network environment for ASG

In the load balancer section, an “Attach to an existing load balancer” option is selected and then, “SocialResearch-TG” is selected as an existing target group load balancer. This ensures that the EC2 instances launched with the specified launch template are integrated with the designated target group load balancer and allows for load distribution to the instances created by ASG.

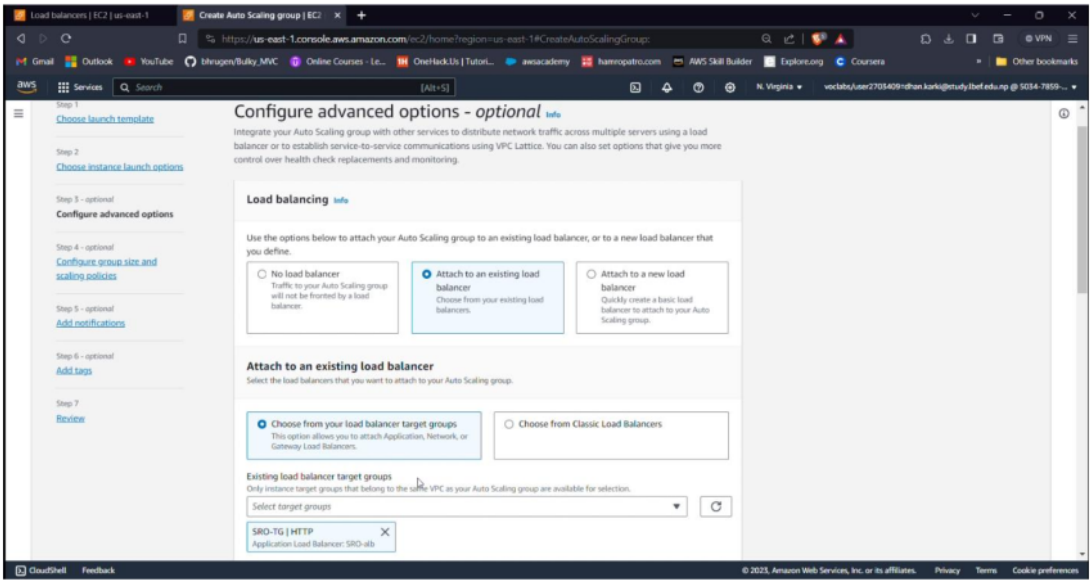


Figure 44: Attaching Load balancer

Additionally, the health check grace period is set to 90 seconds from default 300 so that the newly launched instances boot up and stabilize faster before undergoing health checks.

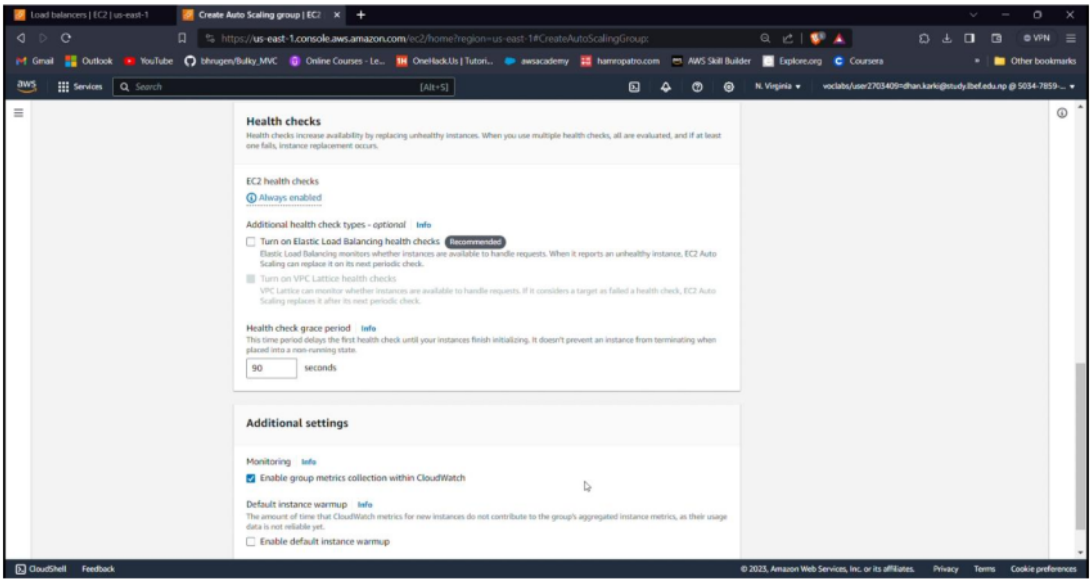


Figure 45: ASG Health checks

Following that, under Group size, desired and minimum capacity is set to 2 and maximum capacity to 4. This implies that the Auto Scaling group will always have two instances. If any instances become unhealthy or are terminated, the group will replace them as soon as possible to guarantee that there are always two instances operating. If the application receives varying loads of traffic, the group tries to increase the instances from 2 but not more than 4.

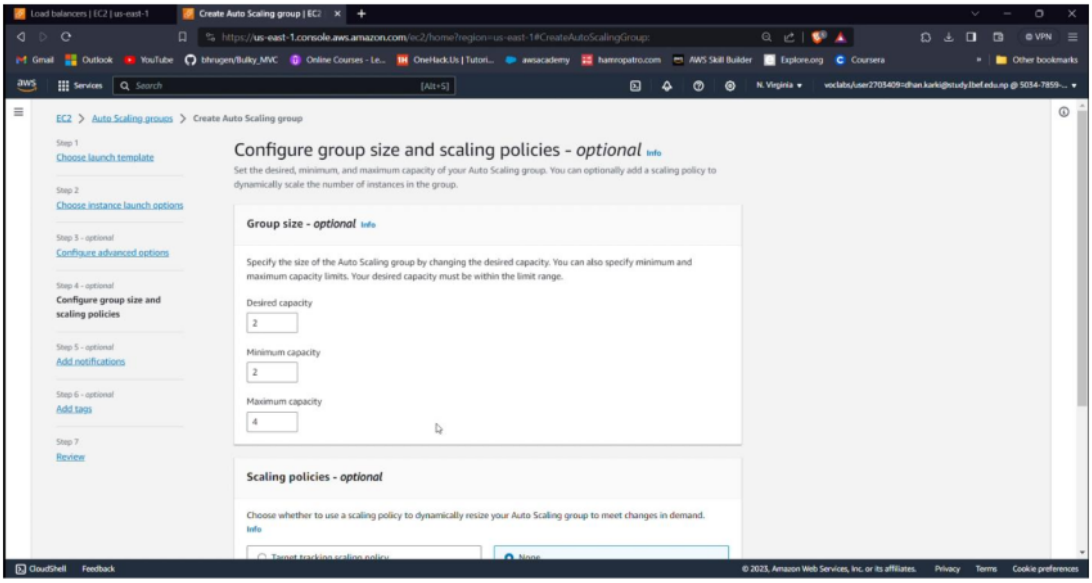


Figure 46: Group Size configuration

No notifications need to be added at this time as per the requirement.

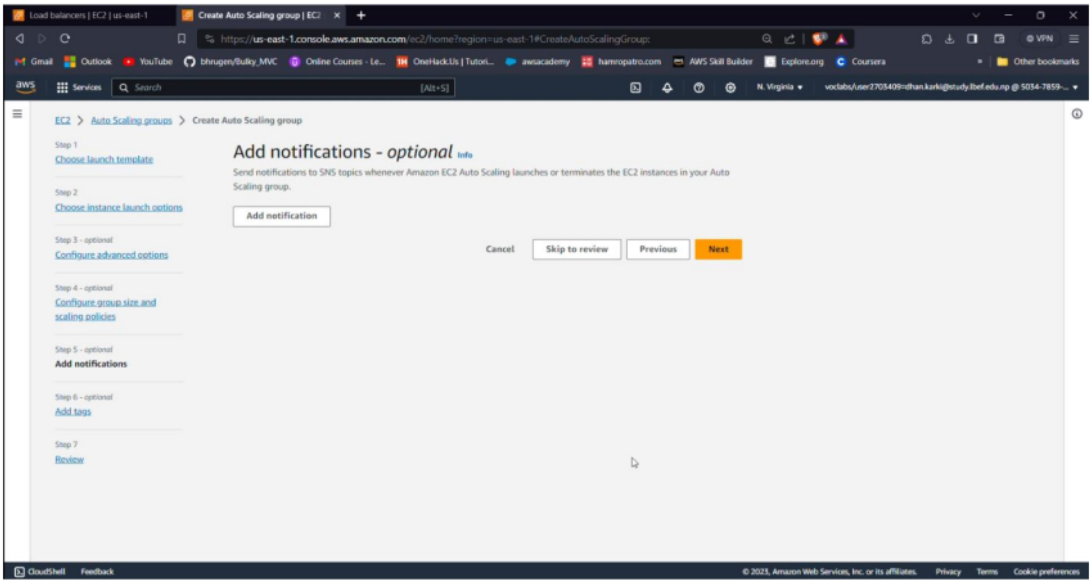


Figure 47: Notification options

Also, not adding any tags at the moment but it's generally considered a best practice to include tags when configuring AWS services.

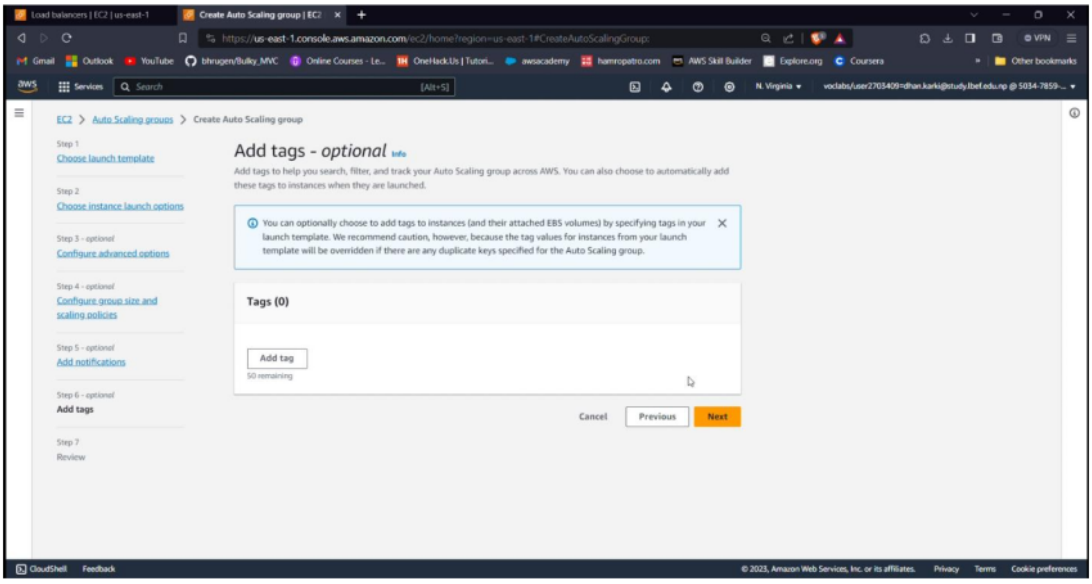


Figure 48: ASG tags

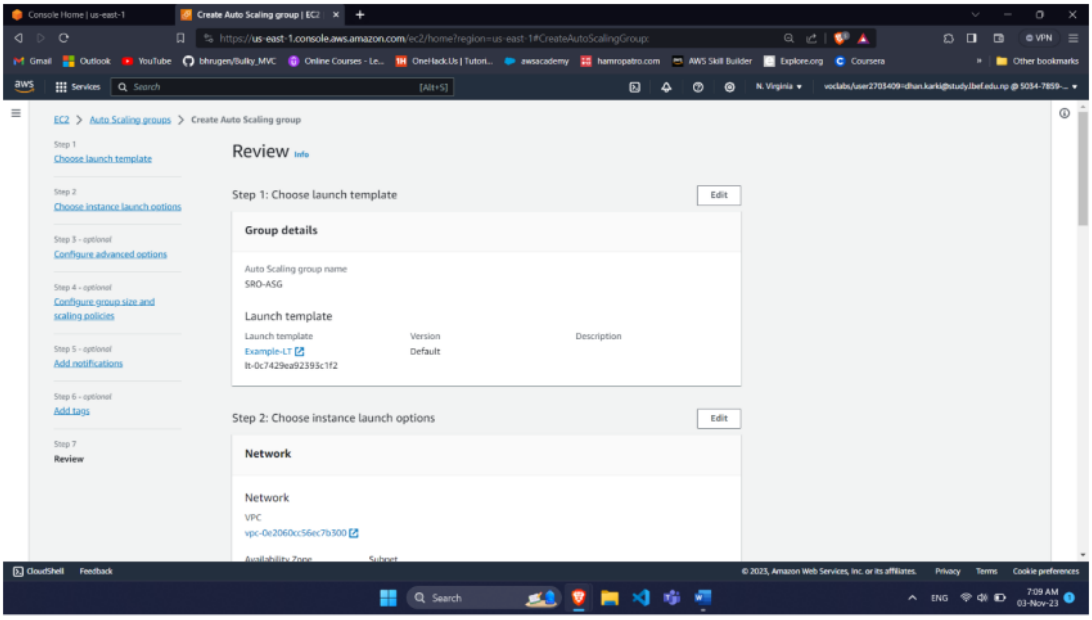


Figure 49: Review ASG

The current state of "SRO-ASG" is "updating capacity". This is because the target capacity has been set to 2, and as a consequence, the ASG is trying to achieve this capacity by launching two EC2 instances.

Here, you can see two ec2 instances initializing in 2 availability zones: us-east-1a and us-east-1b with the name ExampleAPP added.

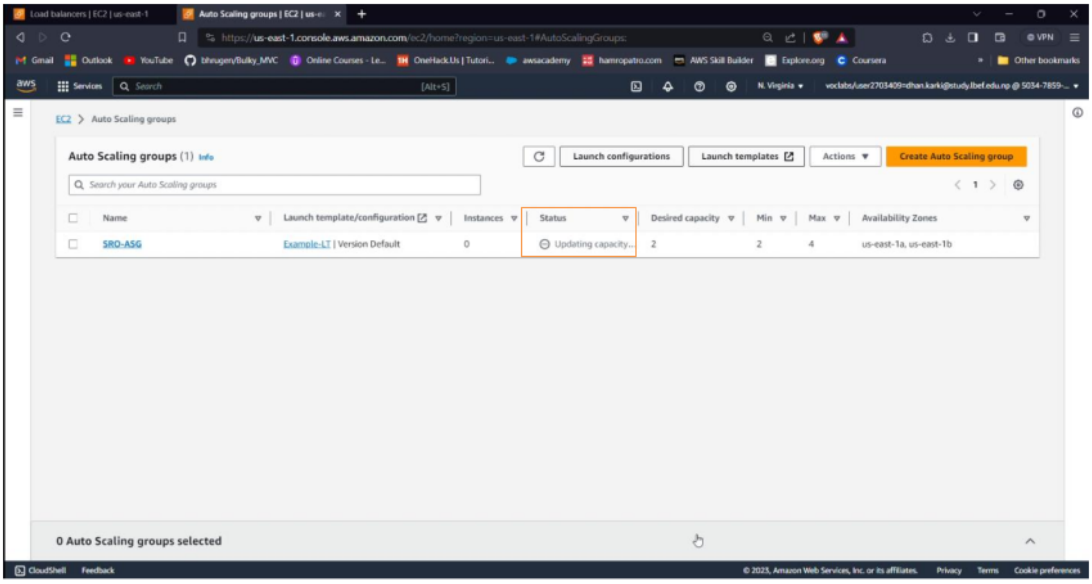
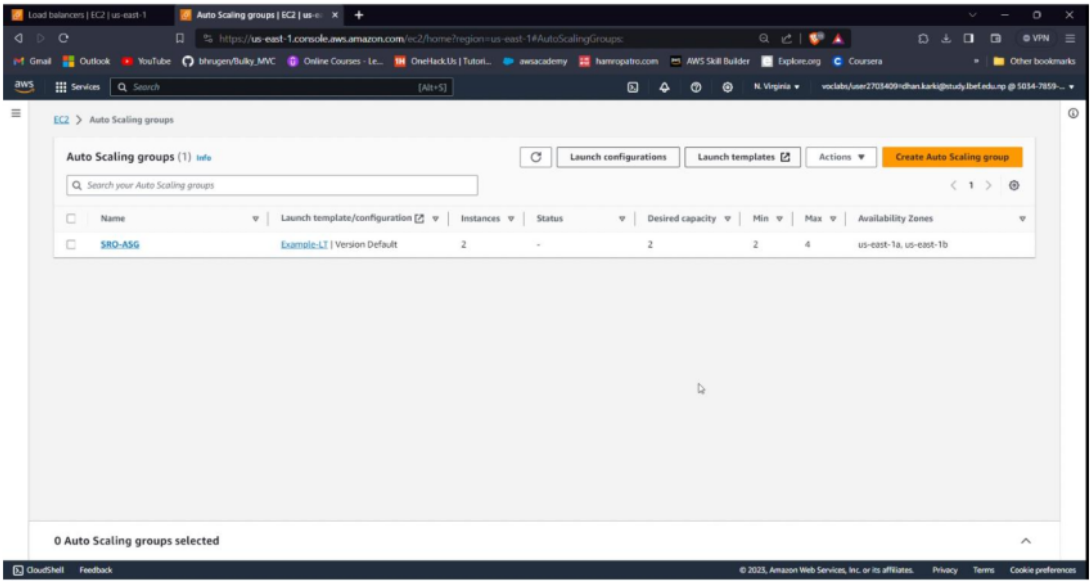


Figure 50: ASG Capacity Updating Status



When looking at the list of instances, two ec2 instances are in the process of initializing across two availability zones: “us-east-1a” and “us-east-1b” with the name “ExampleAPP” and are available after some time.

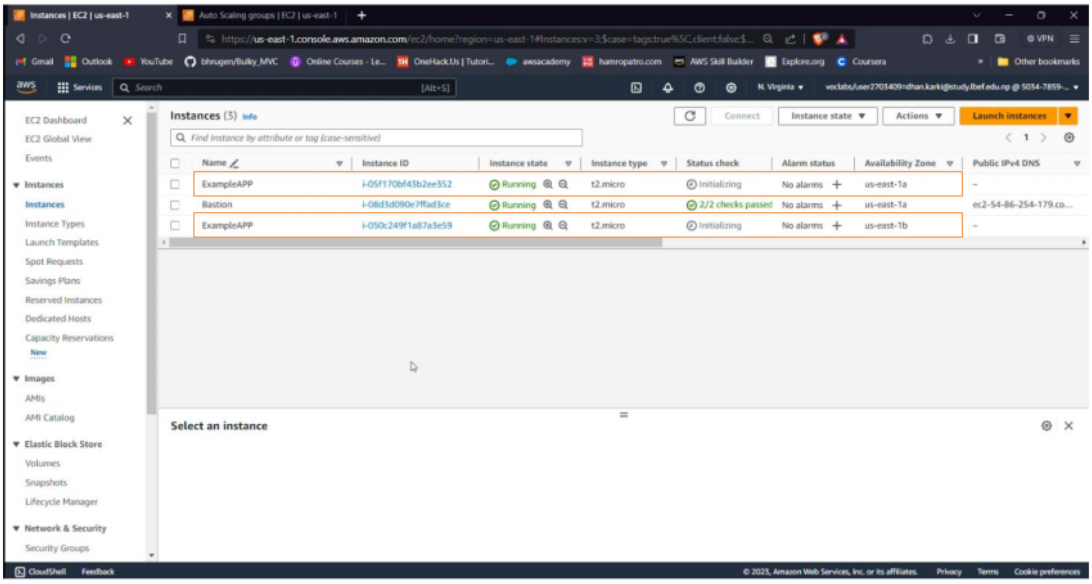


Figure 51: Launching 2 new instances

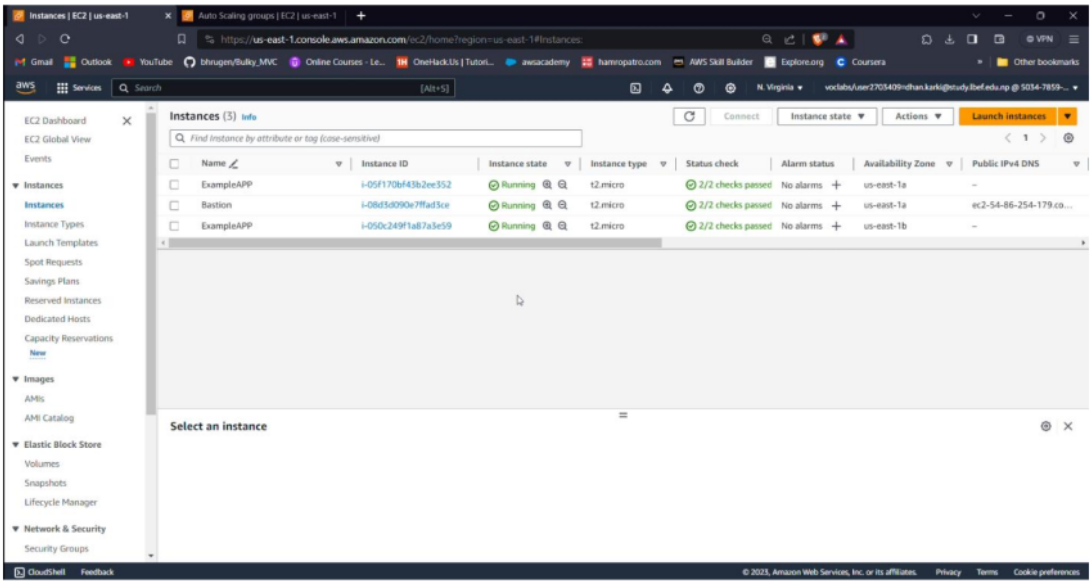
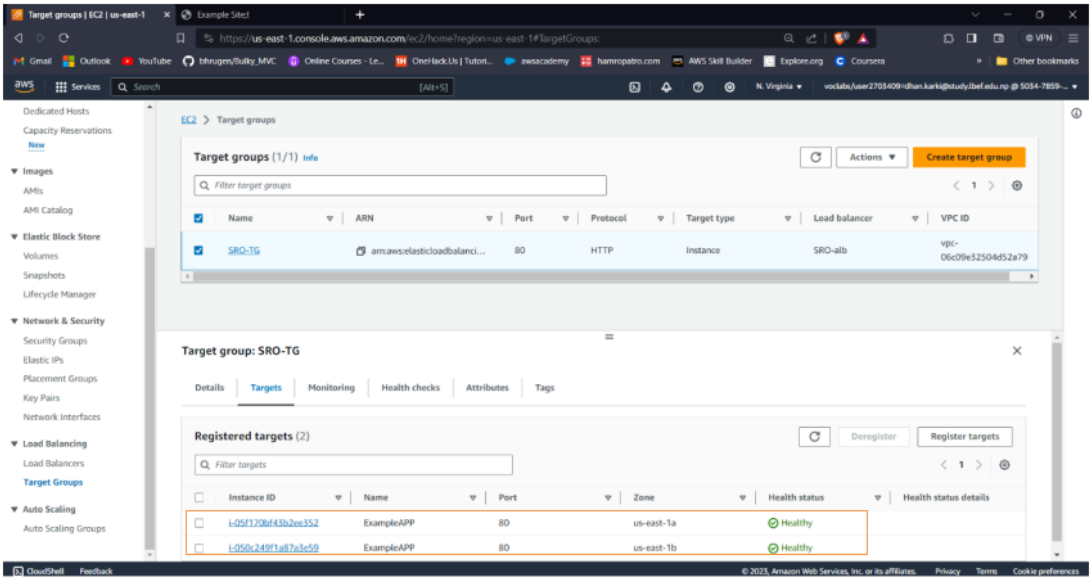


Figure 52: Successful creation of 2 new instances after status check

Furthermore, when examining the instances within the target group, we can find two healthy ec2 instances which means that the load balancer can now send traffic to these instances.



Now, it's time to verify the functionality of the application through the load balancer. After copying the DNS name of the "SRO-alb" load balancer and pasting it into the web browser, webpage is running confirming that the application is operational and running as expected.

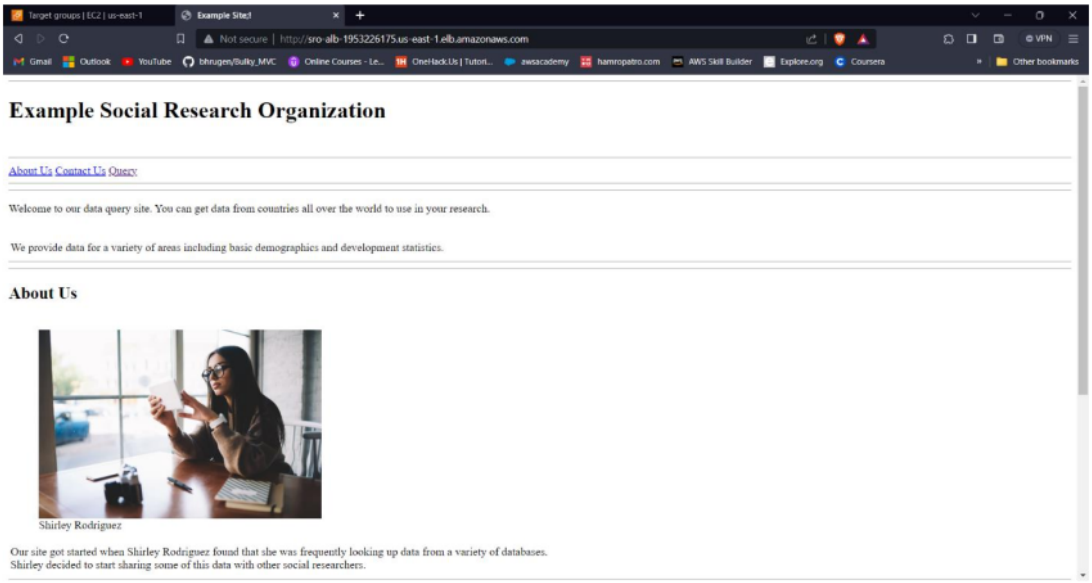


Figure 53: Home page of web application

Navigating to the Query hyperlink, it'll open up below page.

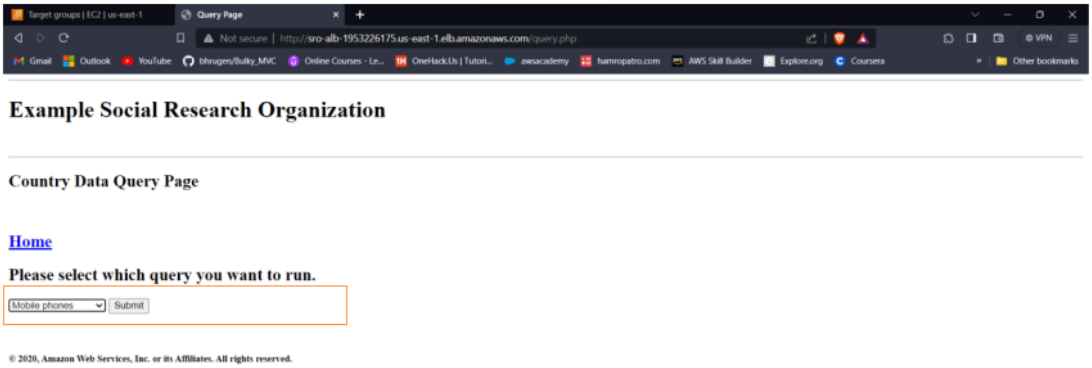
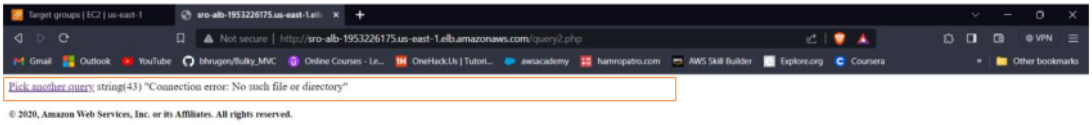


Figure 54: Query page

When the form is submitted choosing "Mobile phones", the web page returns error message like this:



It is not returning any result because it is not being able to connect to the database.

6.4 Amazon RDS

Amazon RDS, or Amazon Relational Database Service, is a cloud-based service offered by Amazon Web Services (AWS) that makes it easier to set up, operate, and scale relational databases. It supports common database engines like as MySQL, PostgreSQL, SQL Server, Oracle, and MariaDB and includes features such as automatic backups, high availability, and simple scalability to fulfill the demands of a wide range of applications and workloads (Sarkar, 2017).

In the architecture setup, a MySQL RDS database has been used as a core component. This database is essential for data storage and management, acting as a reliable and scalable data repository for the application deployed in the private subnets.

Before creating an RDS instance, it is considered best practice to establish a Database Subnet Group to define the VPC and subnets where the database instances are going to operate, allowing

proper network setup and enhancing high availability by distributing instances across several availability zones. It also assures network security, isolation, and adherence to best practices and standards.

Creating database Subnet groups

There are not any subnet groups yet created for the database.

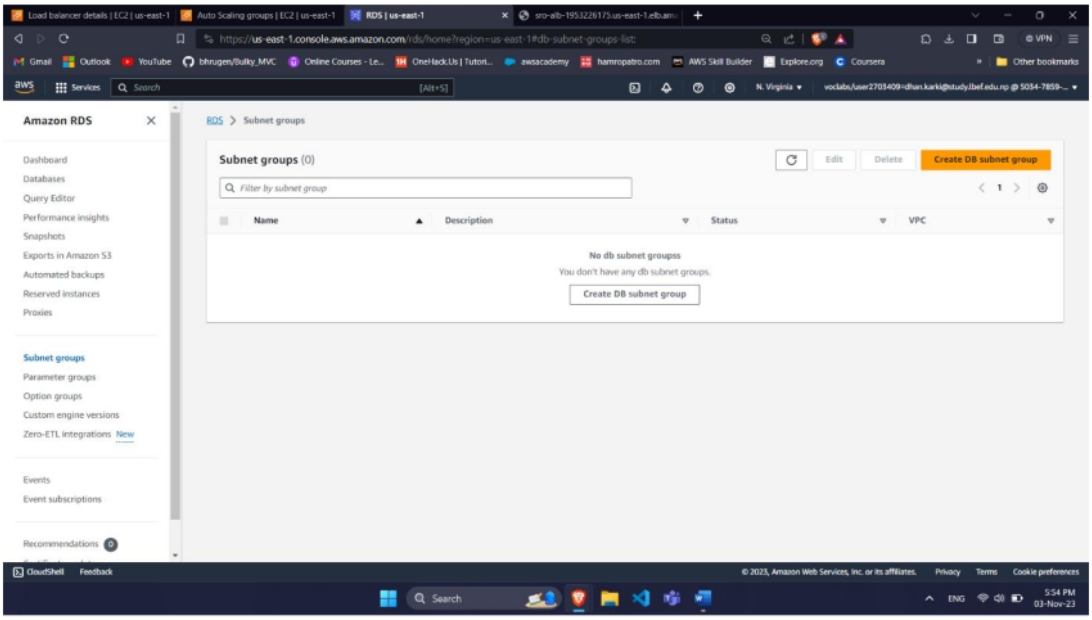


Figure 55: Subnet Groups

To begin the process of creating DB subnet group, the first step is to give a name and description for subnet group.

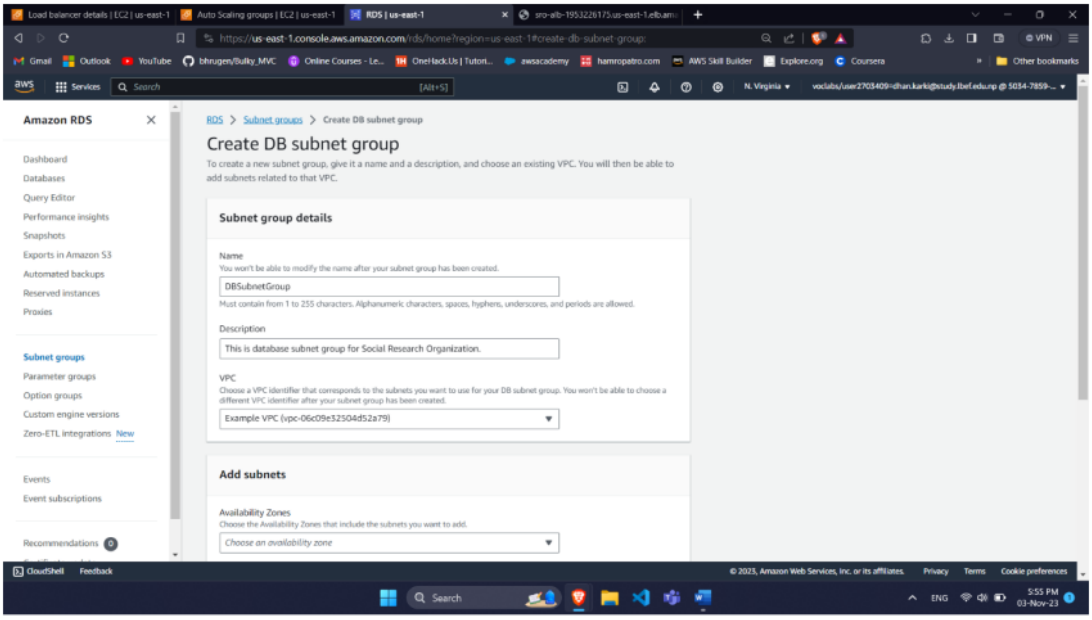


Figure 56: Creating Subnet groups

Then, AZs “us-east-1a” and “us-east-1b” are selected and for the subnet, two private subnet with CIDR block of 10.0.2.0/23 and 10.0.4.0/23 in the subnets section and then a database subnet is created.

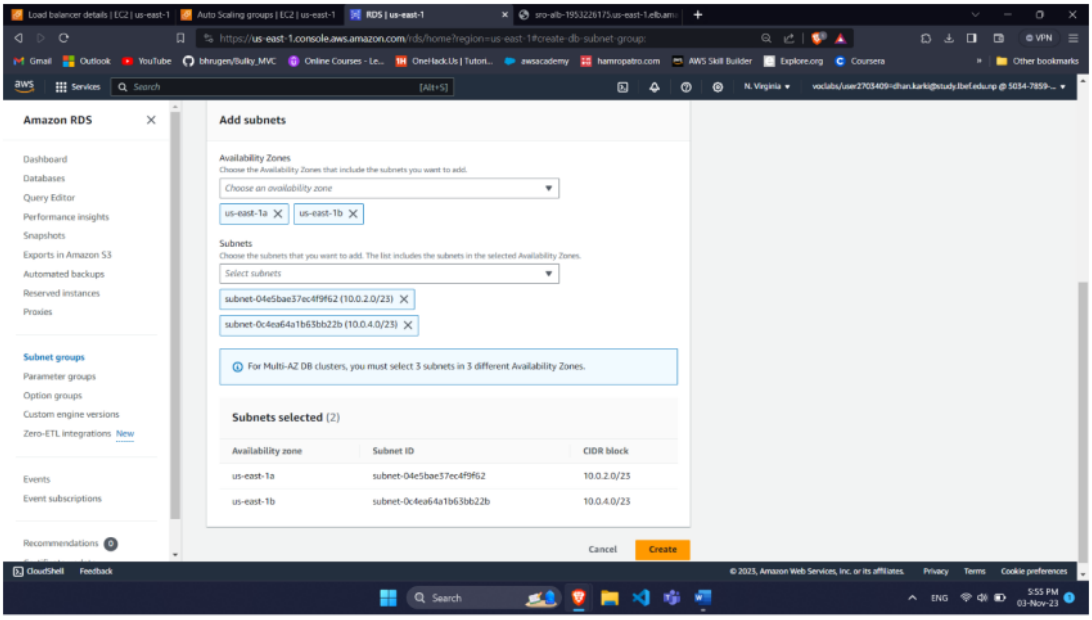


Figure 57: Subnet selection in database subnet group

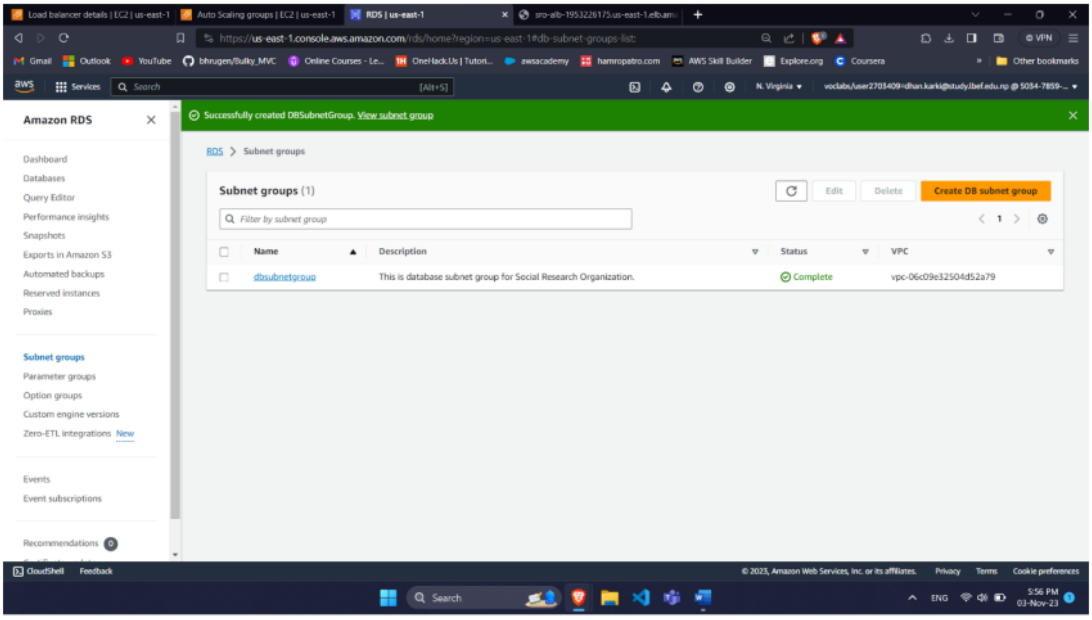


Figure 58: Successful creation of subnet group

Creating an RDS

To begin with creating RDS, select MySQL engine under engine options as shown below.

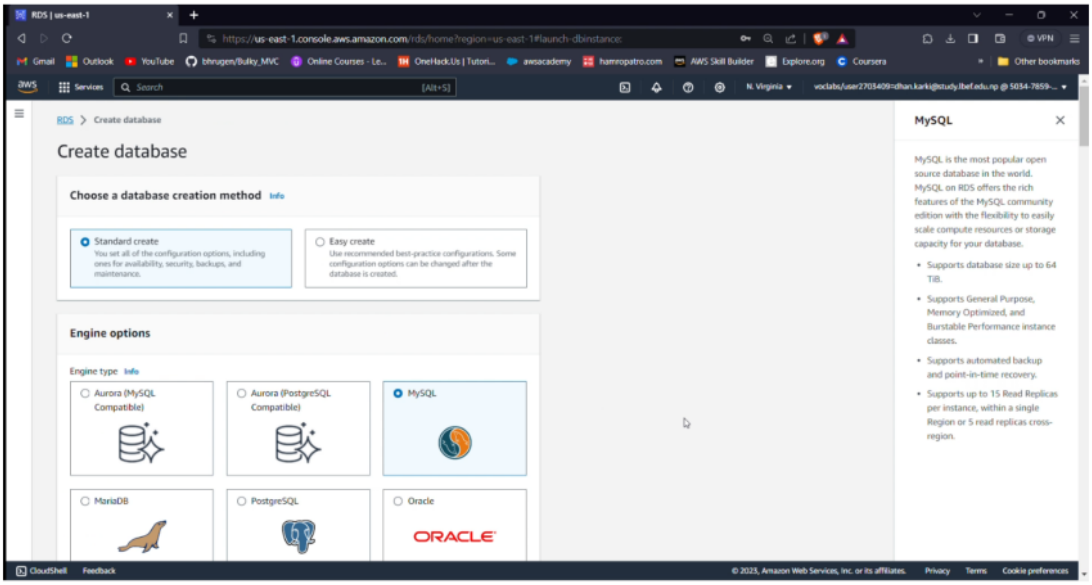


Figure 59: RDS Engine options

Then, a Dev/Test sample template is chose and Deployment options as Multi-AZ DB Instance.

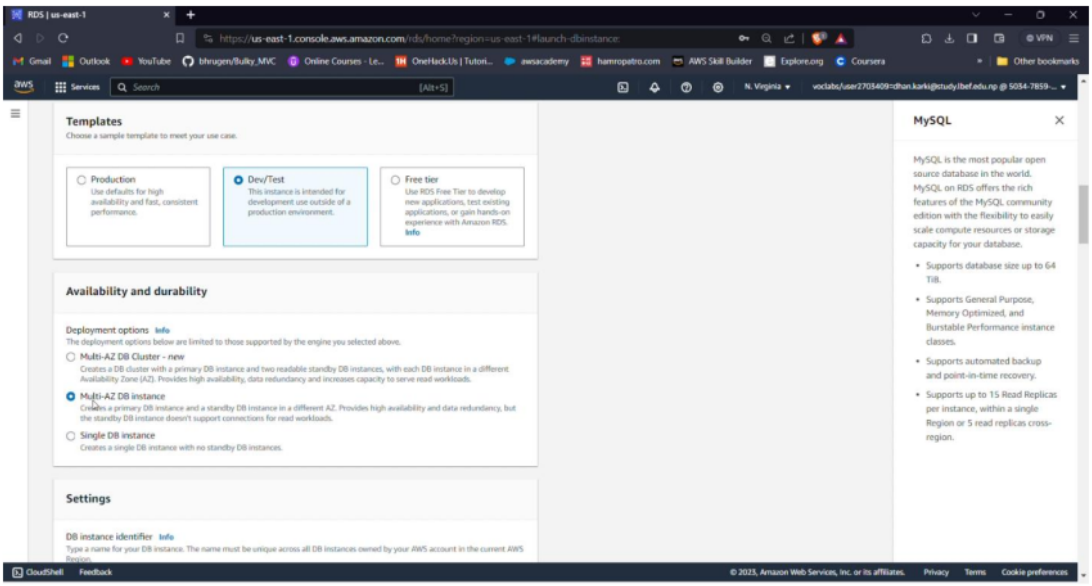


Figure 60: RDS Deployment options

Next, name is given as “SRO-db” to the database instance and under credentials settings which will later on be used to access the created database, username and password are configured.

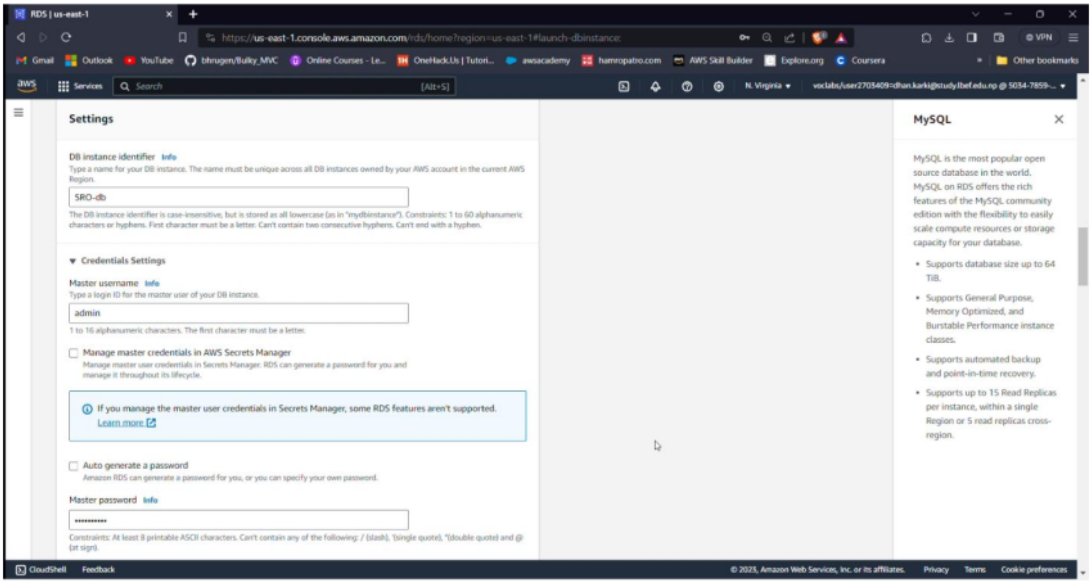


Figure 61: RDS settings

To optimize cost-efficiency for the database, db.t3.micro database instance of burstable classes is chosen. Additionally, gp2 is selected as database storage type and allocated 20 GiB storage.

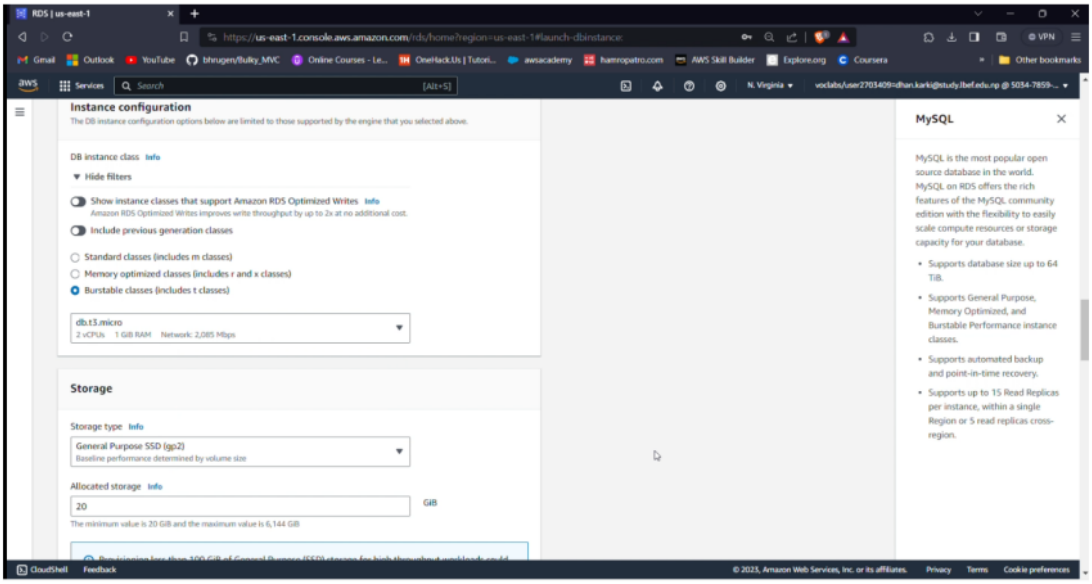


Figure 62: RDS Instance and Storage Configuration

In the connectivity section, Example VPC is taken as Virtual Private Cloud (VPC) and also ensured dbsubnetgroup DB Subnet group is selected.

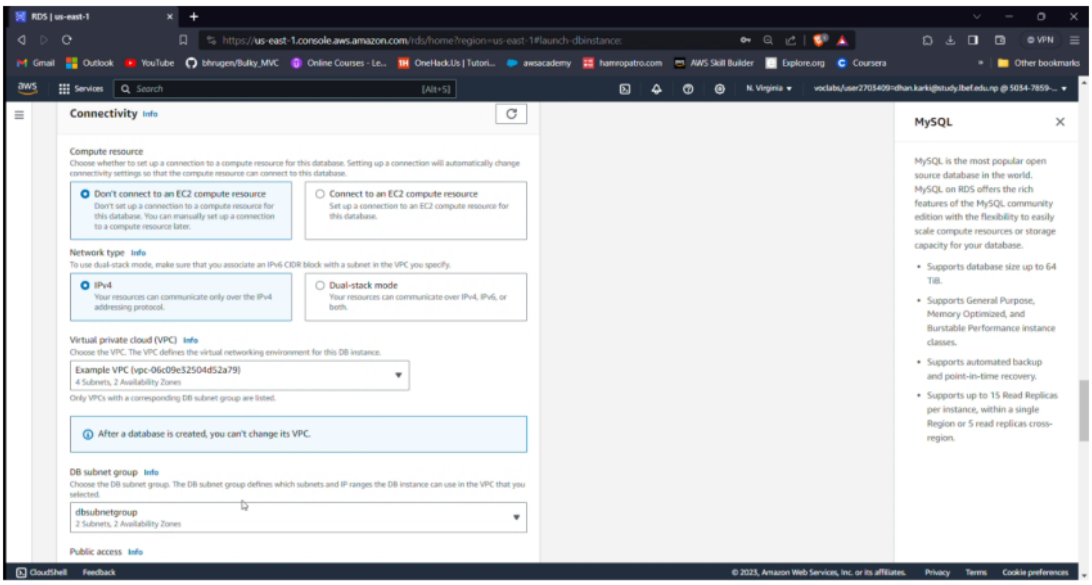


Figure 63: RDS Connectivity

Next, Example-DB security groups is set to existing VPC security groups that allows inbound traffic in port 3306.

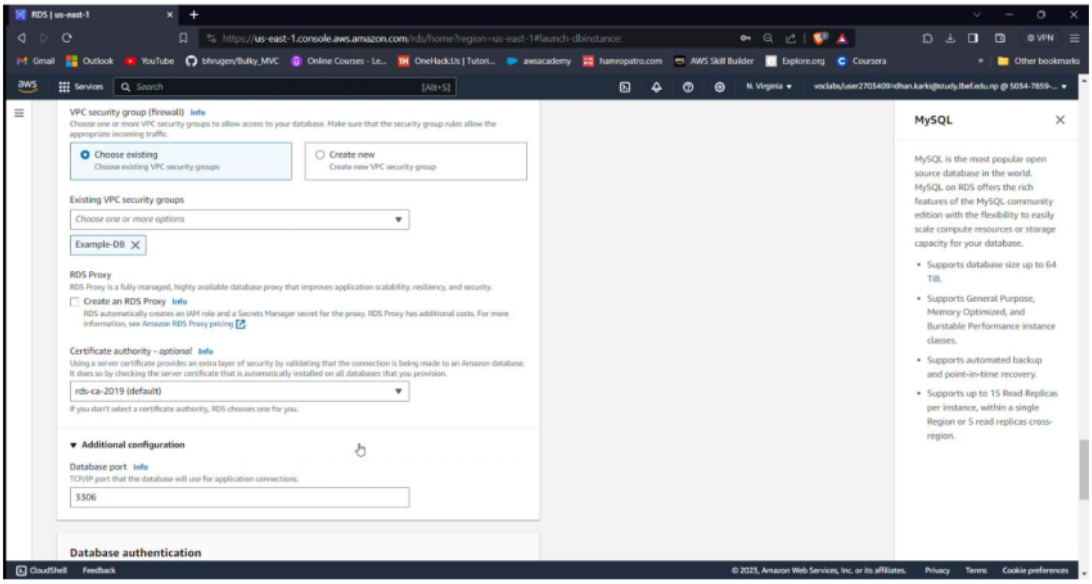


Figure 64: RDS security group

Next, Enhanced monitoring is disabled from Monitoring section and an initial database name as "socialresearchdb" is given.

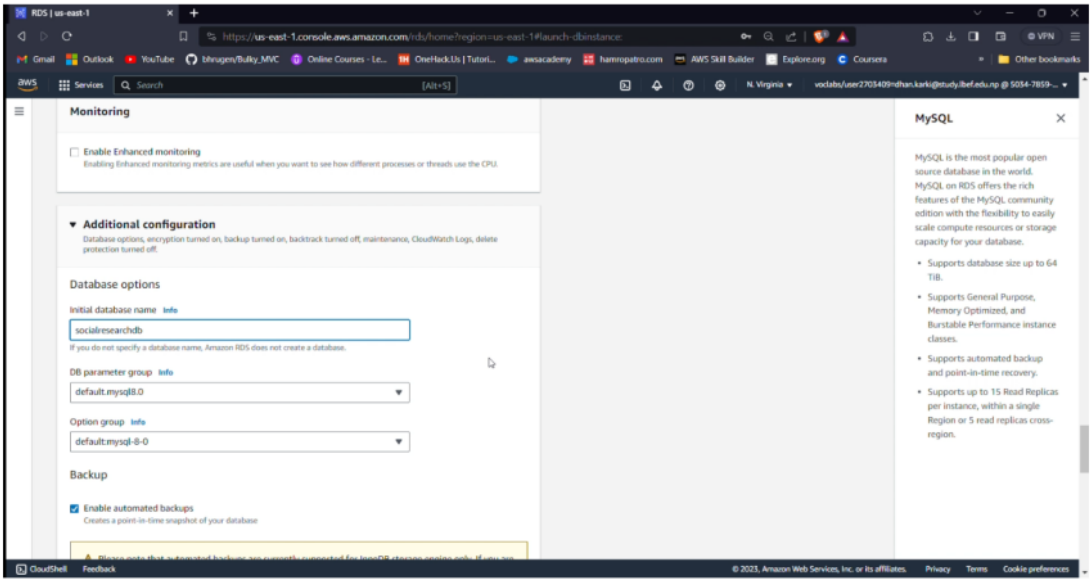


Figure 65: RDS additional configuration

Acting upon the above configuration, now database creation process can be launched which is illustrated below with its status as "Creating".

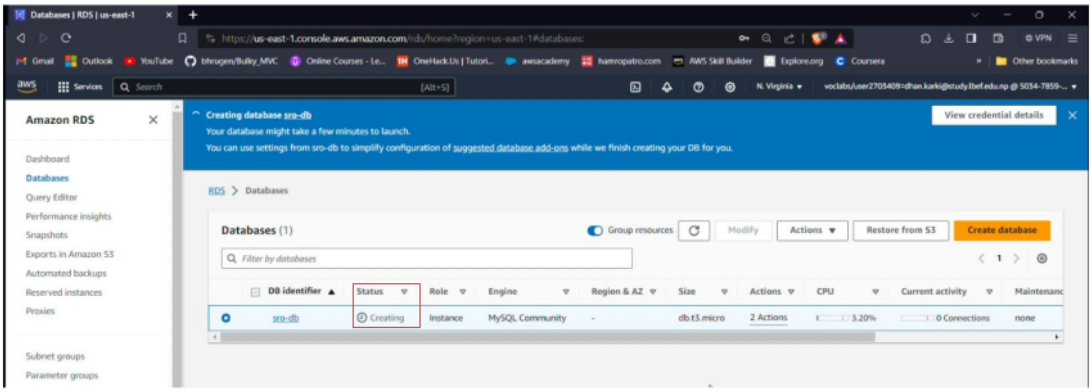


Figure 66: Creating state of RDS

After waiting for at least 4-5 min for its creation, its state will change to modifying which demonstrates successful creation of database.

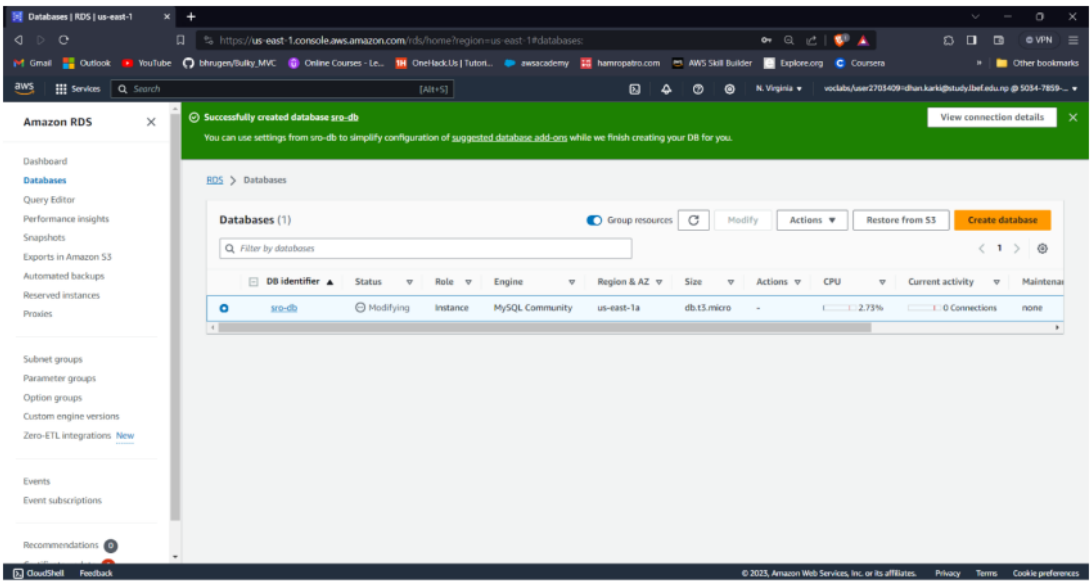


Figure 67: RDS instance created

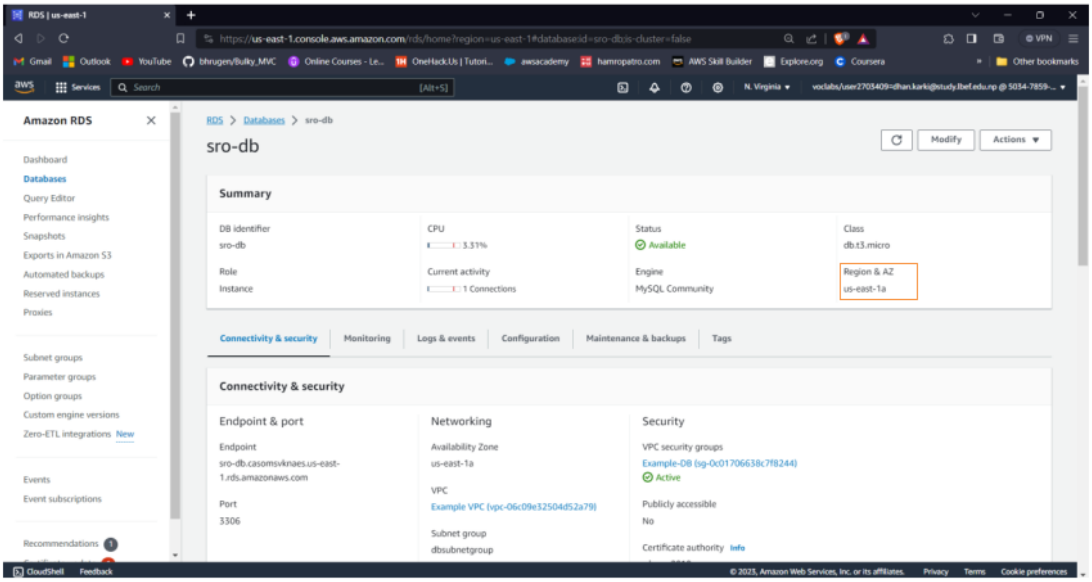


Figure 68: "sro-db" RDS database preview

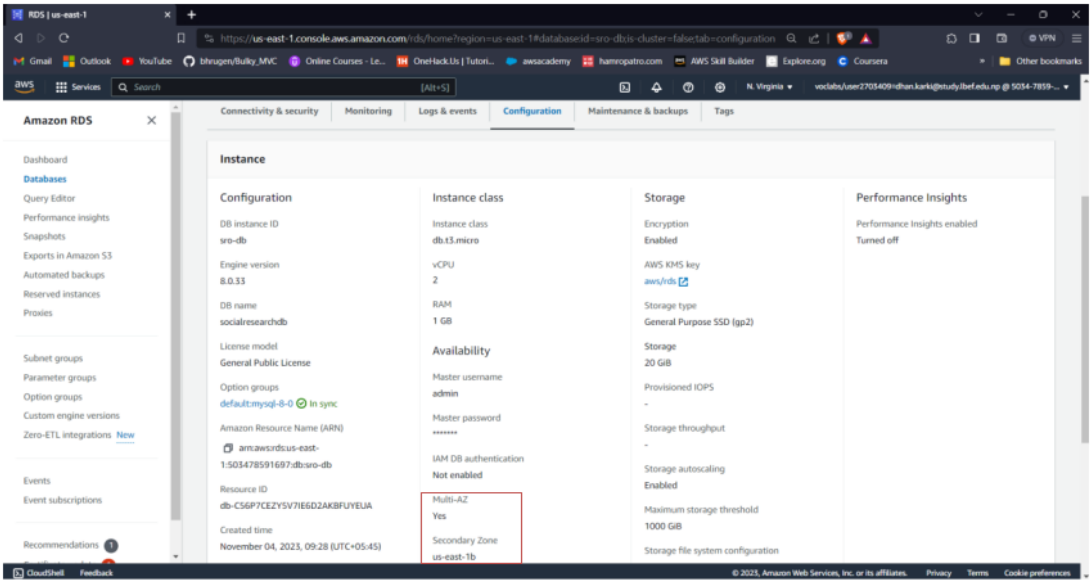


Figure 69: "sro-db" RDS database preview 2

In the above details, there is a primary instance located in first AZs responsible for processing all requests, while standby instances act as a backup and have not been created for now. If the primary instance encounters an issue, then only the standby instance becomes active (will be created on us-east-1b). However, the same DNS name is used by the application to access the database and the connection is automatically rerouted to the currently operational database server making it a highly available database.

6.5 AWS Systems Manager Parameter Store

AWS Systems Manager Parameter Store is a secure and scalable storage solution offered by Amazon Web Services (AWS) designed to store configuration data, secrets, and other information that your applications and services may need. It streamlines the process of securely saving, retrieving, and maintaining configuration data and secrets in a centralized place (AWS, n.d.).

This service has been used in the architecture to store database configuration data such as RDS endpoint, database name, username and password.

Storing database configuration information

In order to store database information in AWS System Manager parameter store, click on parameter store.

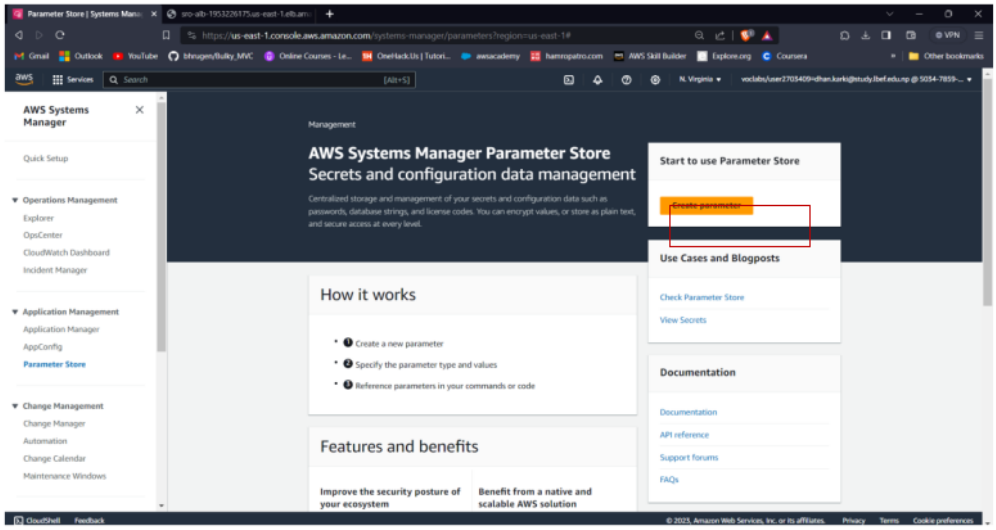


Figure 70: AWS SSM Parameter Store Portal

In the parameter details, the information is filled as follows:

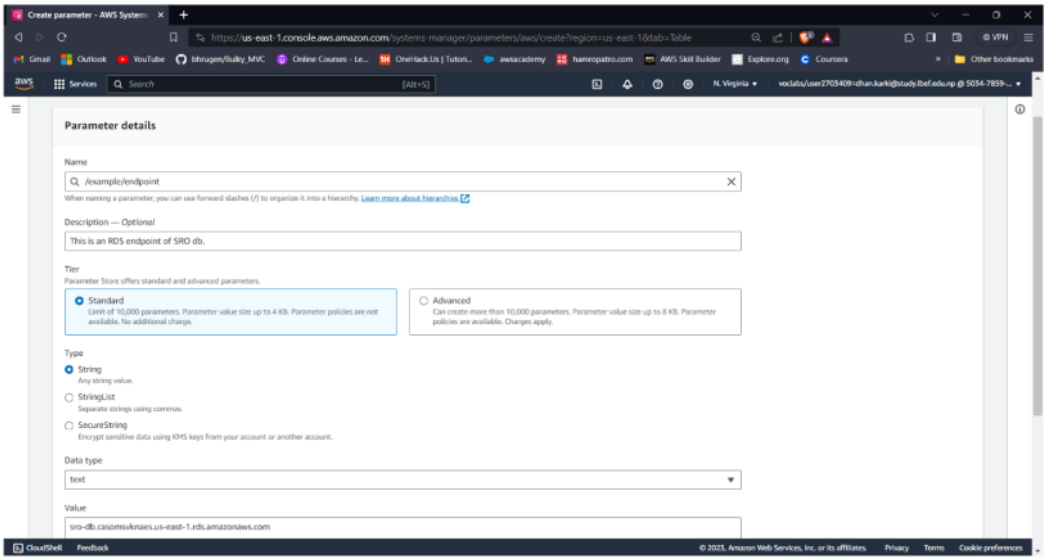


Figure 71: Storing database Endpoint

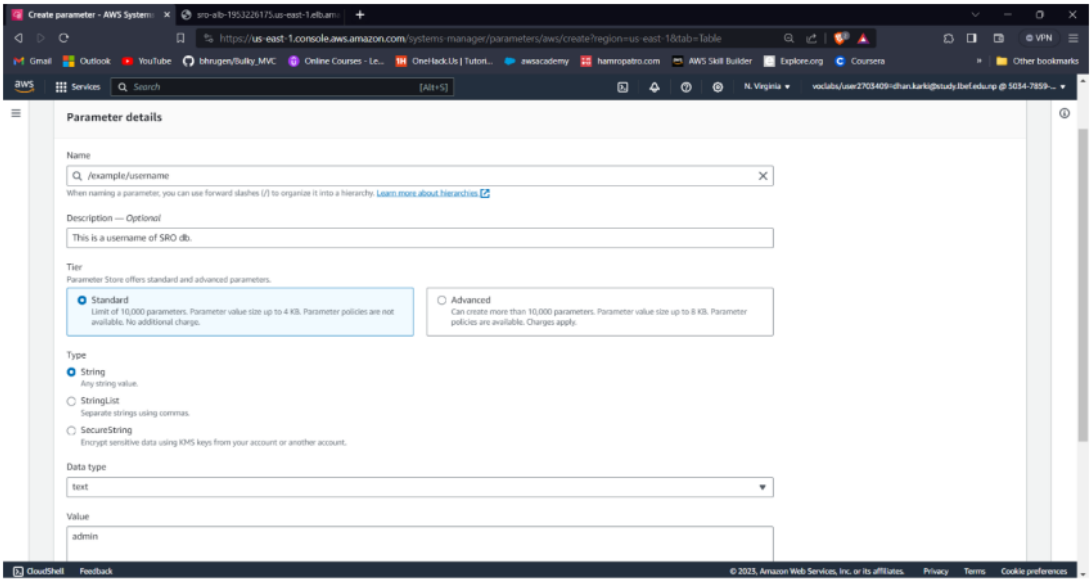


Figure 72: Storing database username

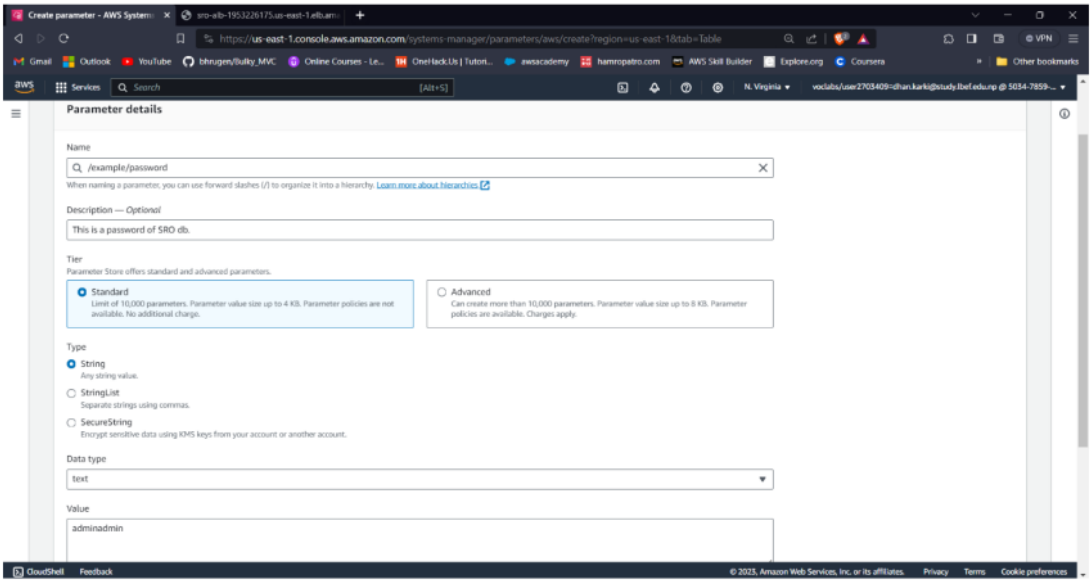


Figure 73: Storing database password

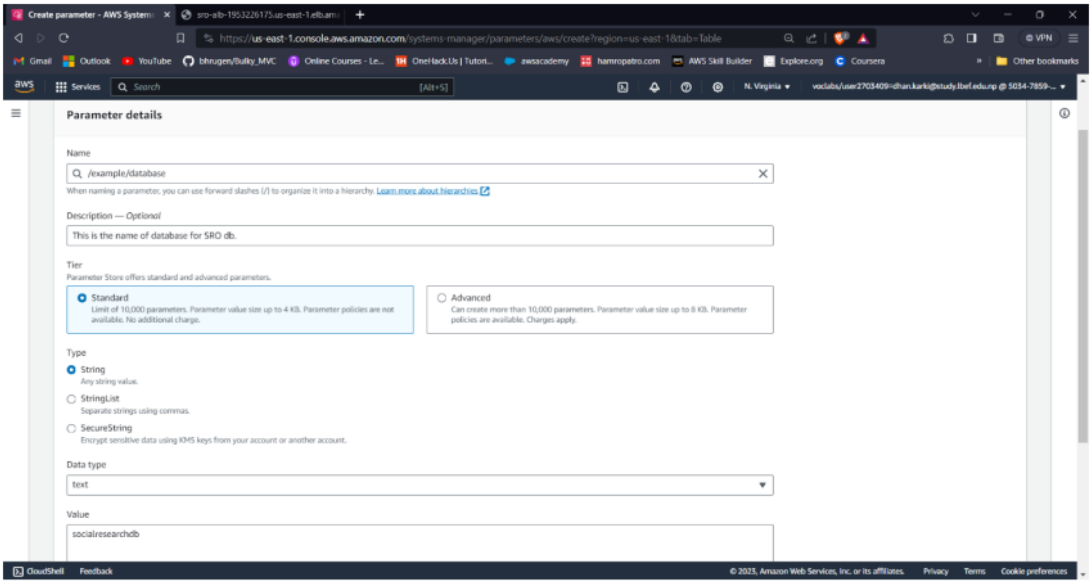


Figure 74: Storing database's name

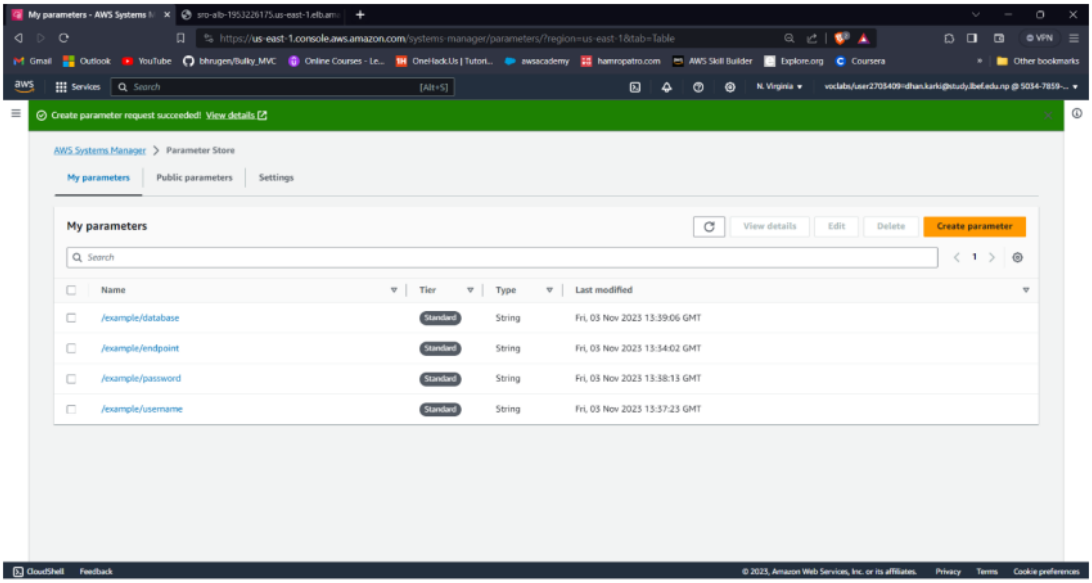


Figure 75: List of parameters

The deployed PHP application can access this parameter through the use of AWS SDK for PHP, which makes it easy to interact with AWS services and then fetch parameters from SSM.

6.6 Data Migration

In order migrate SQL data to RDS instance, we need to SSH through bastion host (i.e., configured in architecture's public subnet) and then to instance in a private subnet from where we can run migration command. A bastion host, sometimes called a leap host or jump server, is a specific type of specialized instance on a network that is intended to provide safe and regulated access to other instances that are located behind a firewall or inside a private network. Here, it serves as an intermediary, providing an extra degree of protection while enabling authorized users to access other network services.

6.6.1 Migrate data to RDS instance

In order to transfer SQL data to an Amazon RDS (Relational Database Service) instance, a bastion host is employed to establish an SSH connection through an EC2 instance, allowing access to a terminal on a bastion server.

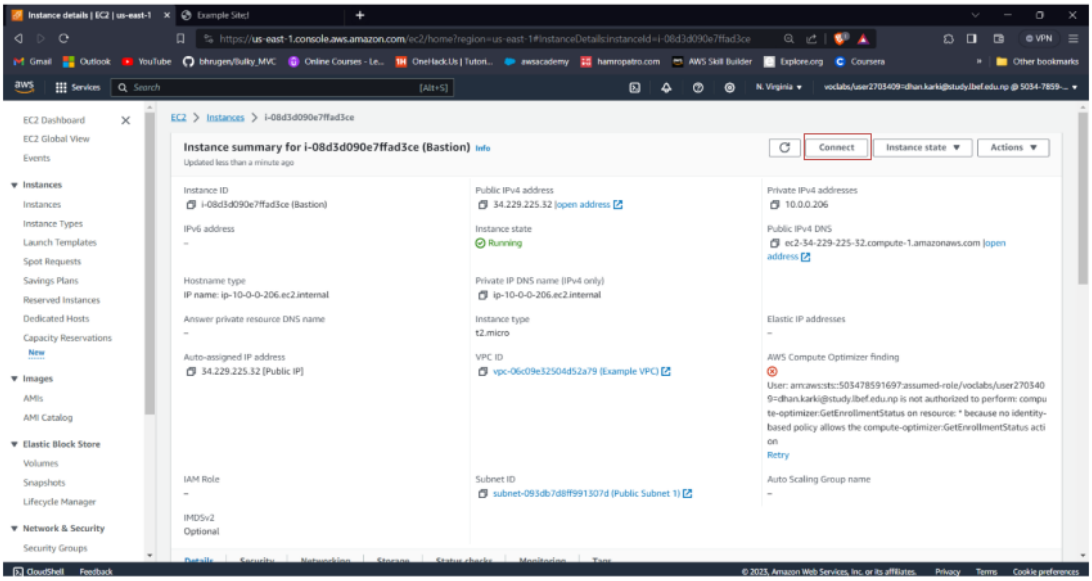


Figure 76: Preview of Bastion Host

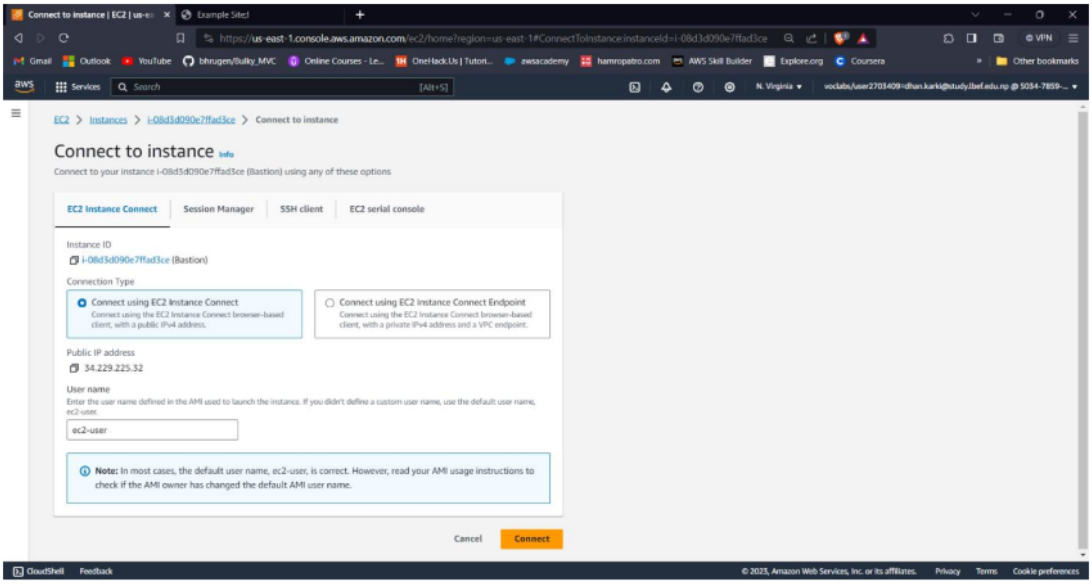


Figure 77: Connecting to Bastion instance

There are not any files in the home directory of bastion server.

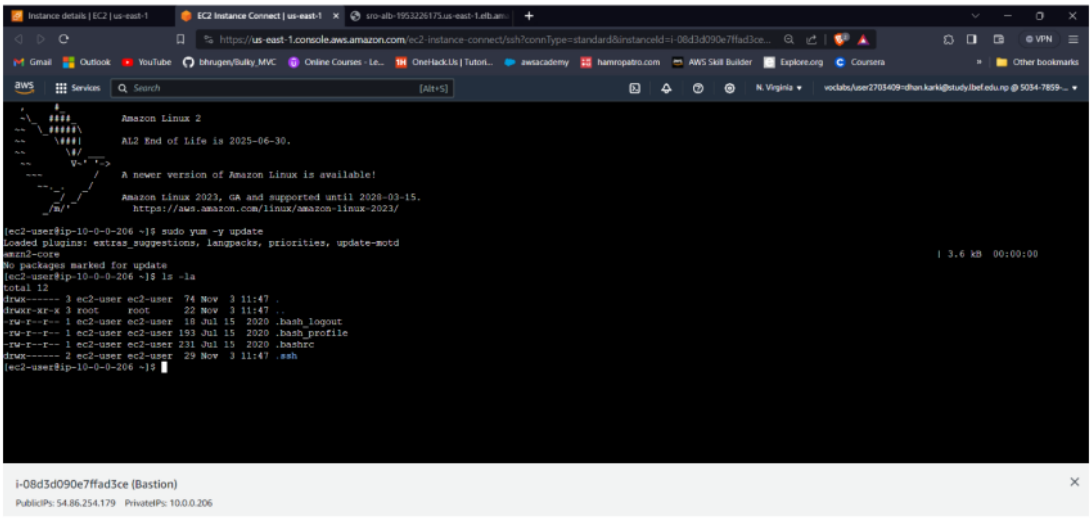


Figure 78: Bastion Terminal

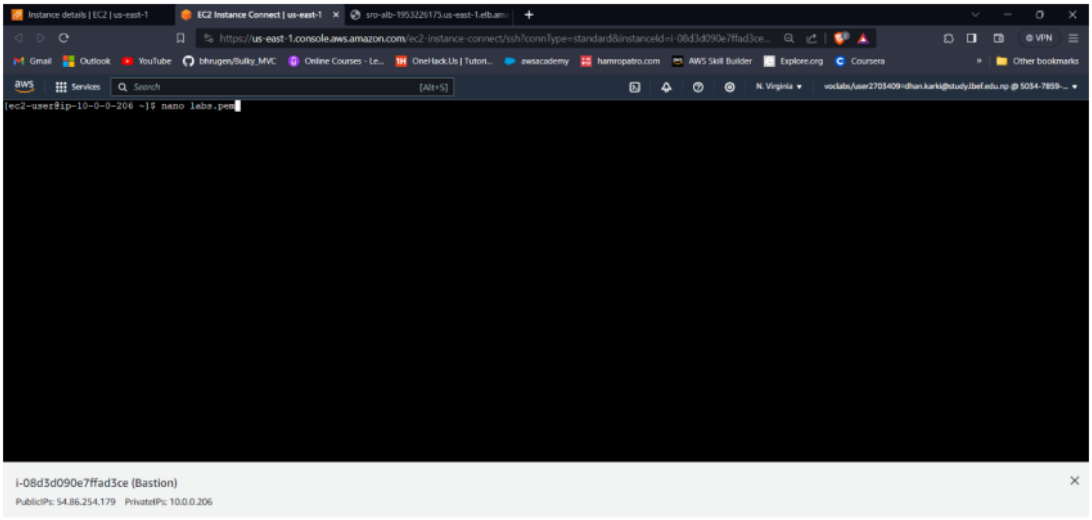


Figure 79: Bastion Terminal creating file

For this assignment, SSH Key's encrypted text is provided by the Capstone Project. user interface. After pasting the copied key text into the CLI and pressing 'Ctrl + X' and then press 'Y', a new file 'labs.pem' is created in the home directory.

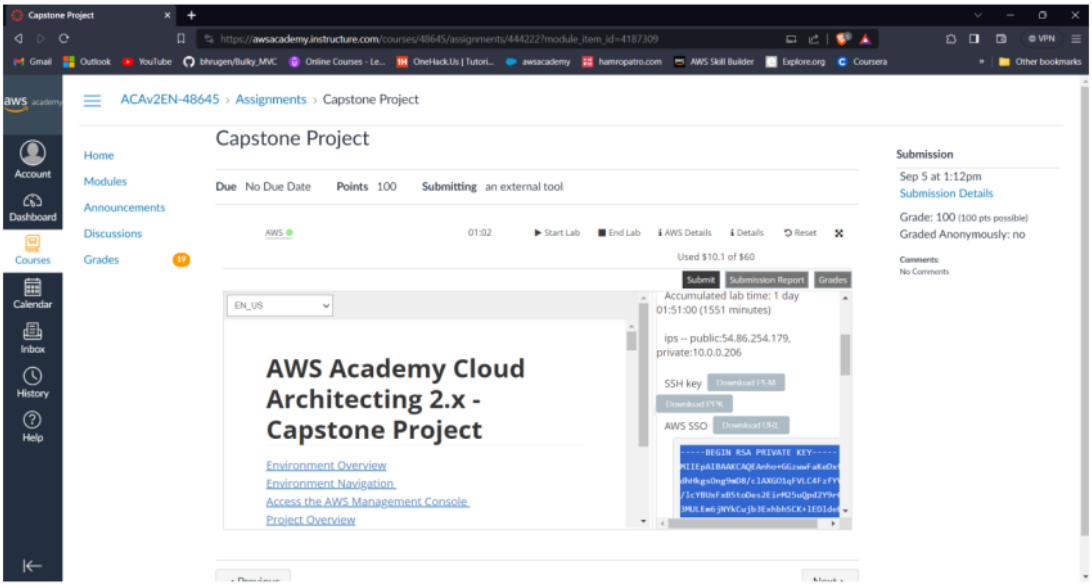


Figure 80: Capstone Project SSH key's text

Then, in order to prevent modifying the .pem file, permission is required to change to the newly created .pem file.

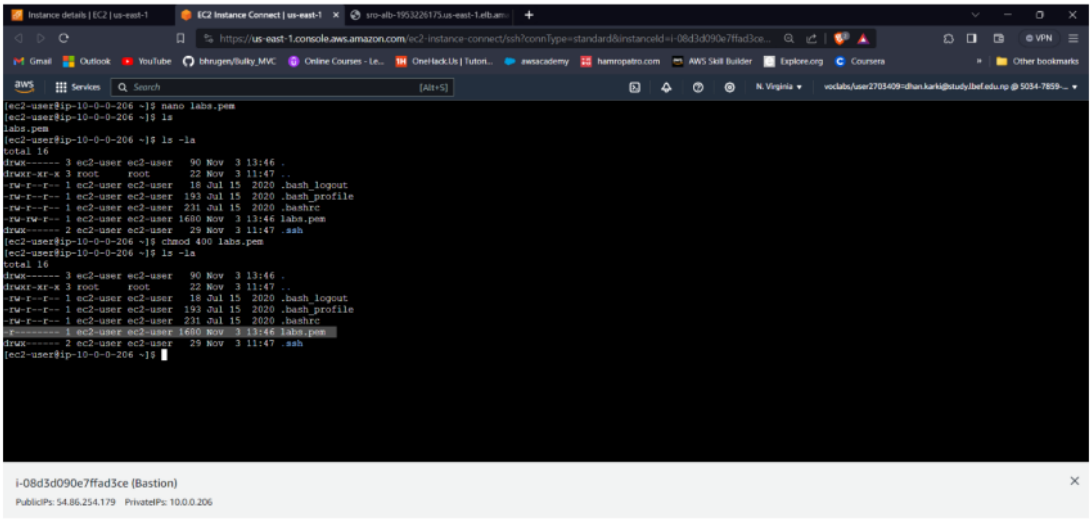


Figure 81: Setting read permission to .pem file

Again, to connect to the private instance, the private Ip of "ExampleApp" ec2 instance is copied as used as below:

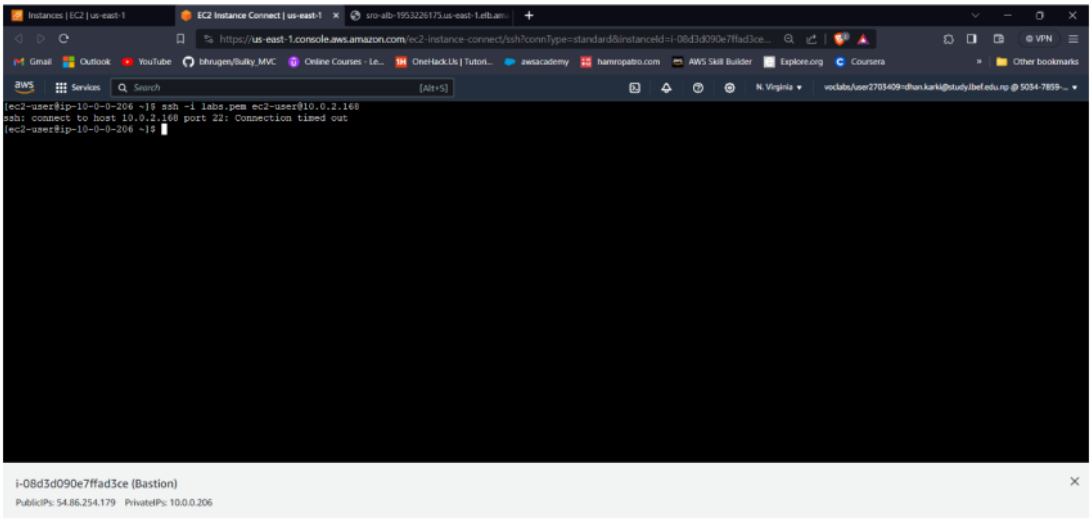


Figure 82: Connecting to private instance

Here, the error message says SSH connection cannot be established to the ec2 instance because the instance's inbound traffic rules in a security group is not configured to allow SSH port. So, it is required to add SSH port inbound rule to "Inventory-App" security group that allows bastion host to connect to instance on a private subnet.

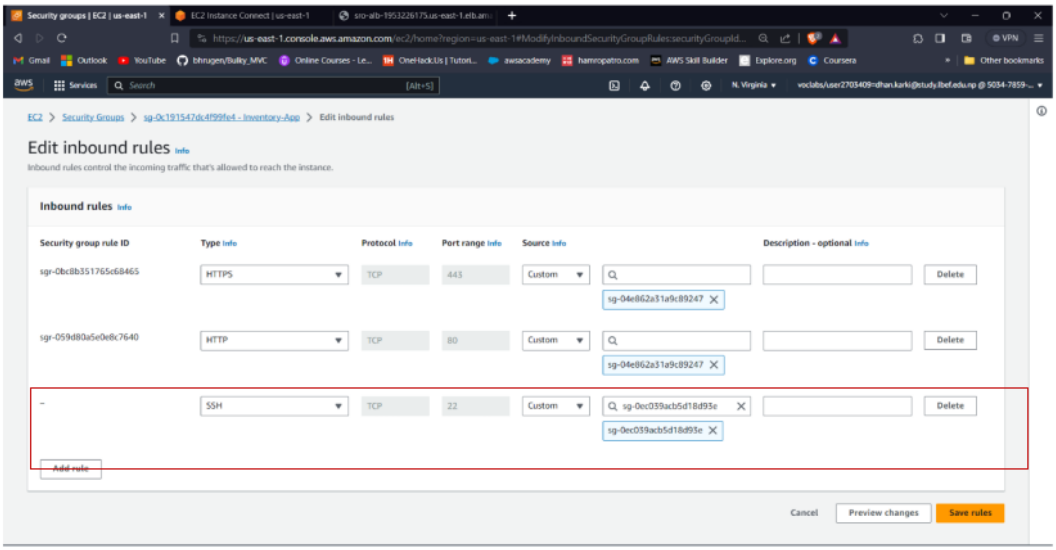


Figure 83: Allowing traffic on port 22 from Bastion SG

At the second attempt, connection can be established successfully.

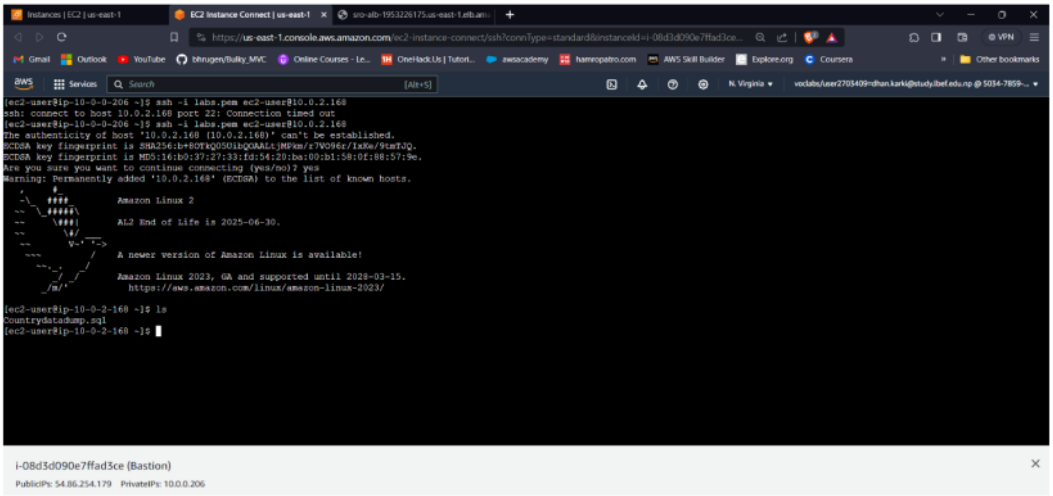


Figure 84: SSH to private instance

Now, the server is ready to migrate data to RDS instance which is shown by the figure below:

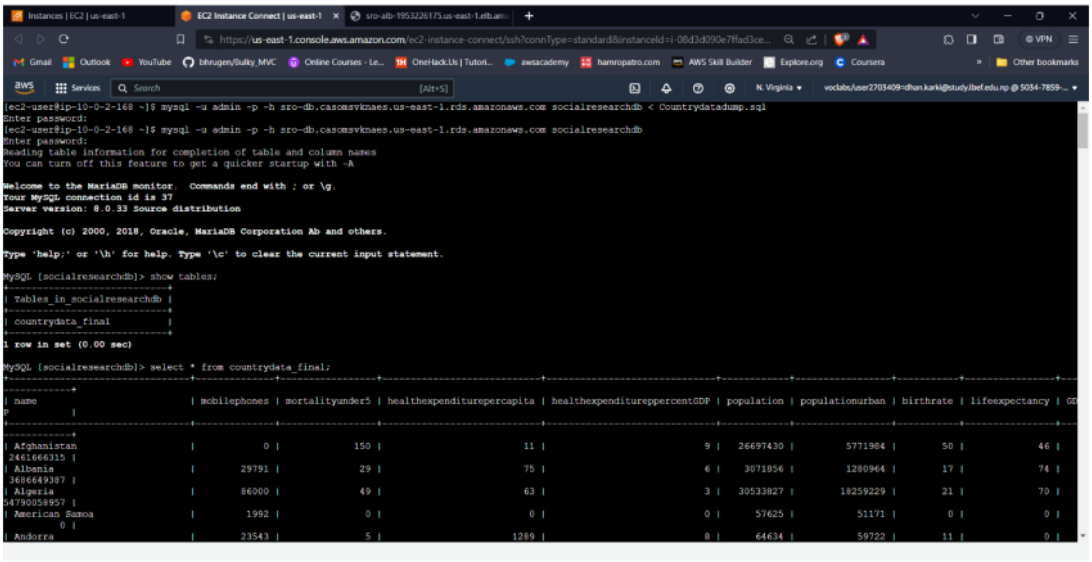


Figure 85: Running migration command in the private instance

This shows the successful migration of databases and again, when trying to access the web page, it can successfully retrieve data from database.

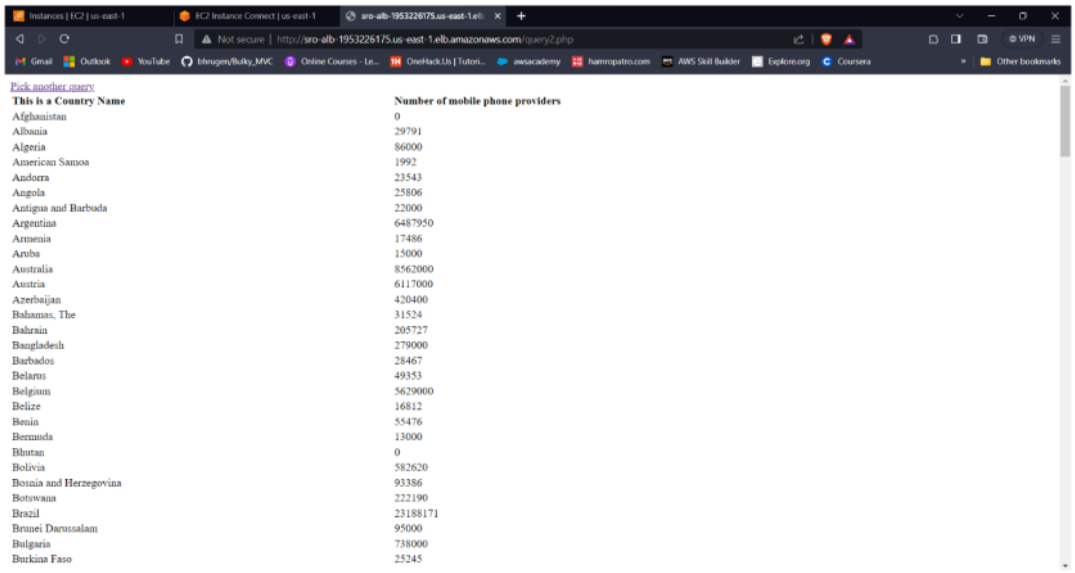


Figure 86: Query web page after database migration

7 Testing High Availability

High availability in AWS is a fundamental part of designing robust and fault-tolerant systems. High availability guarantees that the applications are available and running with little downtime (B, 2023). To test the availability of web applications, we need to terminate one of the instances.

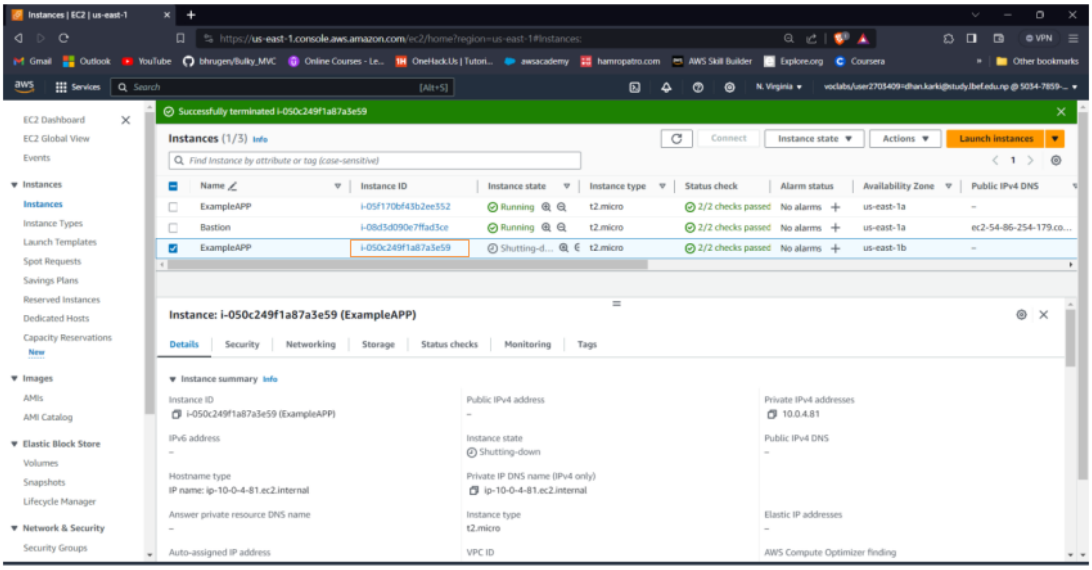


Figure 87: Terminating one of EC2 instance

Shortly after the termination request, the load balancer's health checks will notice that the instance is no longer responding. As a result, the load balancer will autonomously divert all requests to the last active instance, ensuring the application's remains operational. A few minutes later, Amazon EC2 Auto Scaling will also recognize the instance failure. Since it was configured to keep two instances running, Amazon EC2 Auto Scaling will automatically commence the deployment of a replacement instance.

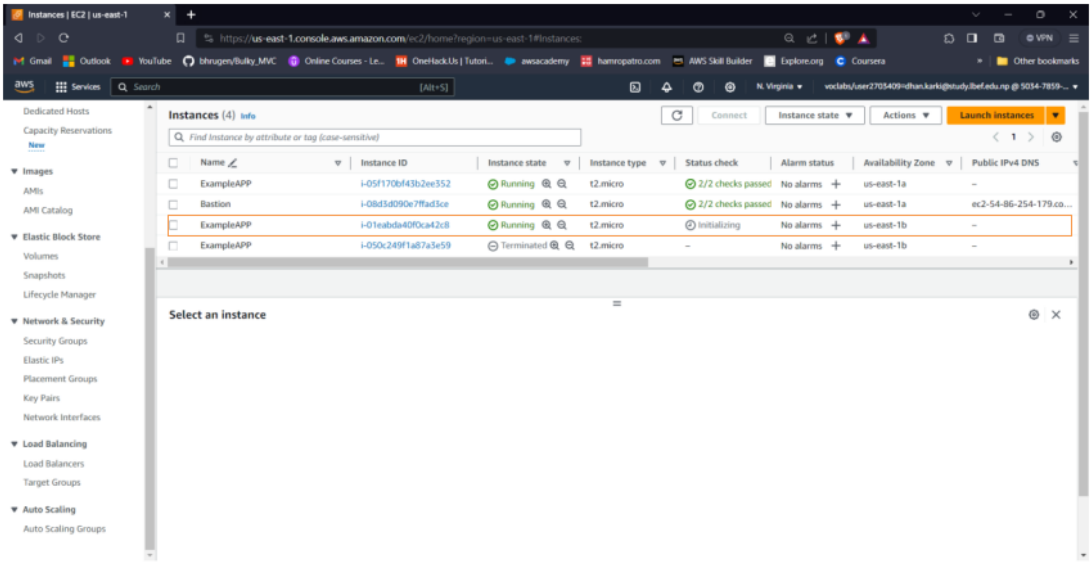


Figure 88: List of EC2 instances

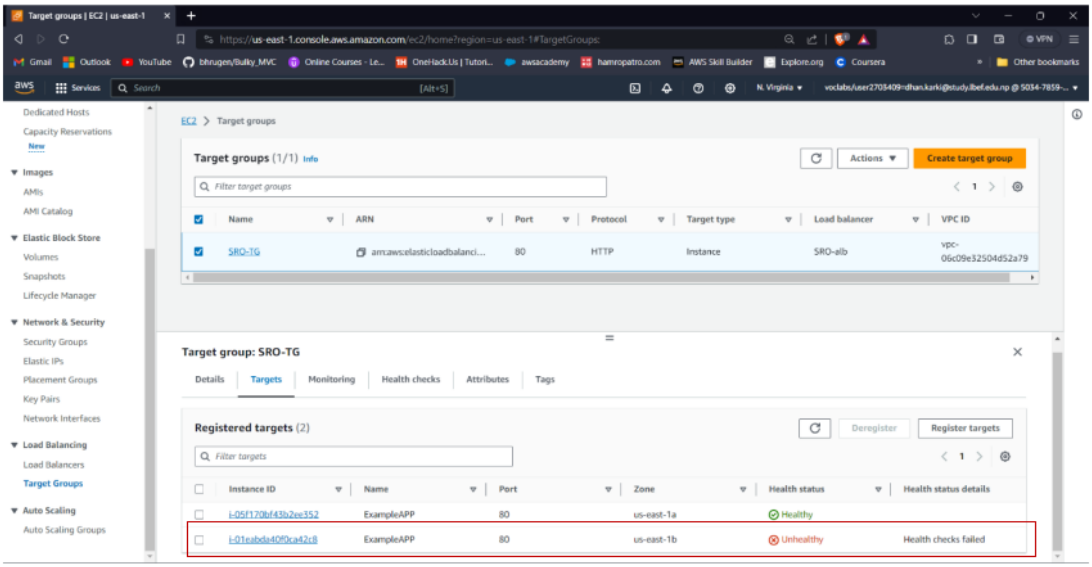


Figure 89: Target group status

After a few minutes, the health check for the new instance should become healthy. The load balancer will resume sending traffic between the two Availability Zones.

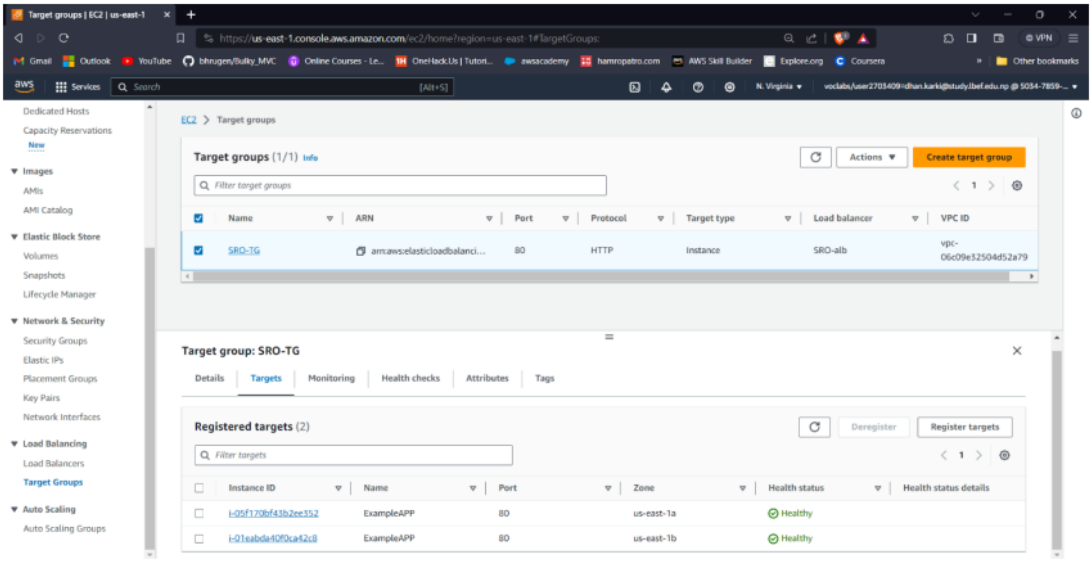


Figure 90: Healthy Targets in Target Group

8 Conclusion

In conclusion, the goal of this project is to improve the security, functionality, and scalability of the Example Social Research Organization's website infrastructure. Important components include optimizing EC2 resources, securing the MySQL database, controlling administrative access, guaranteeing a smooth user experience, attaining high availability with a load balancer, safely storing database information in the AWS Systems Manager Parameter Store, and turning on automatic scaling. These upgrades will provide a more reliable and safer website that is better suited to meet the demands of the organization for research. These steps help to create and deploy a strong, safe, and responsive website that advances the organization's research objectives.

9 References

.M, M., Selvan, S. T., Kuppuchamy, R., & Shanthi, S. (2021, February). Highly scalable and load balanced web server on AWS cloud. *IOP Conference Series Materials Science and Engineering, 1055*, 012113.

AWS. (2022). *https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html*. Retrieved from docs.aws.amazon.com.

AWS. (n.d.). *https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html*. Retrieved from docs.aws.amazon.com: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html

B, S. (2023, June). High Availability and Fault Tolerance in AWS. *International Journal of Innovative Research in Information Security, 09*, 66-72.

Beach, B., Armentrout, S., Bozo, R., & Tsouris, E. (2019). *Virtual Private Cloud.* doi:10.1007/978-1-4842-4850-8_5

Buyya, R. (2020). Cloud Computing: Principles and Paradigms. *Wiley Online Library*.

Garfinkel, S. (2020). An Evaluation of Amazon's Grid Computing Services: EC2, S3, and SQS. *Harvard Library Office for Scholars*.

Gupta, B., & Mittal, P. (2022). A Review on Amazon Web Service (AWS). *European Union Digital Library*.

Huihong, H. (2022). An Approach to Estimating Cost of Running Cloud Applications Based on AWS. *Asia-Pacific Software Engineering Conference*.

Porald, D. (2021). A Distributed Architecture for the Monitoring of Clouds and CDNs: Applications to Amazon AWS. *IEEE Transaction on Network Collection Journal*.

Ranjan, R. K. (2021). A Comparative Study of SaaS, PaaS and IaaS in Cloud. *International Journal of Advanced Research in*.

Sarkar, S. (2017, 03). Security of Amazon Web Services.

Selvan, S. T., .M, M., Kuppuchamy, R., & Shanthi, S. (2021). Highly scalable and load balanced web server on AWS cloud. *IOP Conference Series Materials Science and Engineering, 1055*, 012113.

Varia, J. (2022). Architecting for the Cloud: Best Practices . *Amazon Web Services*.

Zhong, H. (2021). Proposed Solution for the Deployment in Cloud AWS. *Amazon Web Services Journal*.

NP00575-NP000563-Np0006111-CSVC-Group.docx

ORIGINALITY REPORT

10%	4%	3%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Online Education Services Student Paper	1%
2	Submitted to Bradford College, West Yorkshire Student Paper	1%
3	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
4	Submitted to University College London Student Paper	1%
5	Submitted to Swinburne University of Technology Student Paper	<1%
6	ijritcc.org Internet Source	<1%
7	sgd8.ok-em.com Internet Source	<1%
8	Brian Beach, Steven Armentrout, Rodney Bozo, Emmanuel Tsouris. "Pro PowerShell for	<1%

	Amazon Web Services", Springer Science and Business Media LLC, 2019 Publication	
9	www.insight.com Internet Source	<1 %
10	docs.aws.amazon.com Internet Source	<1 %
11	dir.texas.gov Internet Source	<1 %
12	www.readkong.com Internet Source	<1 %
13	Nick Alteen, Jennifer Fisher, Casey Gerena, Wes Gruver, Asim Jalis, Heiwad Osman, Marife Pagan, Santosh Patlolla, Michael Roth. "AWS® Certified Developer Official Study Guide", Wiley, 2019 Publication	<1 %
14	Submitted to TAFE NSW Higher Education Student Paper	<1 %
15	Submitted to Toronto Business College Student Paper	<1 %
16	www.coursehero.com Internet Source	<1 %
17	Submitted to Study Group Australia Student Paper	<1 %

18	technodocbox.com Internet Source	<1%
19	Sara Perrott, Ben Piper. "AWS Certified SysOps Administrator Practice Tests", Wiley, 2020 Publication	<1%
20	pdfcoffee.com Internet Source	<1%
21	static-aliyun-doc.oss-cn-hangzhou.aliyuncs.com Internet Source	<1%
22	coronasdk.tistory.com Internet Source	<1%
23	raw.githubusercontent.com Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

NP00575-NP000563-Np0006111-CSVC-Group.docx

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69