CLOUD ARCHITECTURE REPORT ON:

DEPLOYING A MUSIC STREAMING PLATFORM ON AWS

By: MOHAMMED NOMAN MOHAMMED ARIF

LinkedIn: https://www.linkedin.com/in/furkhan67/
GitHub: https://github.com/furkhan67/codes/

ABSTRACT

There are many Music Streaming platforms available in the market, each providing different features to their customers. But each of them has their own subscription plans. In order to avoid these different subscription plans, this project aims to create a cloud architecture to deploy a platform for Music Streaming which combines all the best features of these existing platforms and provide customers with a single platform which has the best of all worlds. The goal of this project is to create and deploy the Music Streaming Application using the best cloud practices and create a Cloud Architecture which is secure, highly available, reliable and sustainable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Music streaming platforms are the go-to way of listening to music nowadays. The popularity of these music streaming platforms has increased nowadays mainly due to the handheld devices such as mobiles and tablets getting more powerful to run the required applications and the internet easily accessible to everyone. The main problem with these platforms is that there are multiple players in this market and each of them has paid subscriptions to use them.

This project takes all the features of these platforms and create a single platform offering all these features. This project aims to build a Cloud Architecture Framework which is secure, reliable and sustainable, for the Music Streaming Platform to be deployed using the best cloud practices.

In the following section, the project plan is discussed which gives an idea of the entire process of this project. Section 3 explains the Cloud Architecture System in detail and finally the Section 4 gives an analysis of the project, what work is done and what could have been done better.

2. PROJECT PLAN

The following table shows the entire process of the project which includes each activity done, summary of that activity and time taken by that activity:

Sr.	Activity	Brief Summary	Time Required
1	Project Outline	In this stage, a structure is created of what tasks needs to be done to complete the project successfully.	10 Days
2	Requirement Gathering	All the Functional and Non-Functional requirements are collected at this stage which are essential for the application to operate.	10 Days
3	Selection of Cloud Platform	The cloud platform is selected based on the availability time of the Cloud Provider, Cost of the system after implementation, availability of required compliances and regulation policies etc., also the requirements collected in the previous stage plays a huge role in Selection of Cloud Platform.	05 Days
4	System Architecture Design	In this stage, System Architecture is designed according to the requirements and the Cloud Provider which is selected. System Architecture shows how we will use the services offered by Cloud Provider to run the application on the cloud.	15 Days
5	System Deployment	The Application is deployed after creating the System Architecture which was designed in the last stage using the cloud services provided by the selected Cloud Provider.	04 Days
6	System Testing	After deploying the Application, it is tested to check the reliability, availability, scalability, and security.	10 Days
7	Monitoring & Maintenance	Throughout it's life cycle the system will be monitored and maintenance will be done if any new problem arises.	-

Table 1: Project Plan

Figure representing progress of the Project:

ID	Name	Start Date	End Date	Duration	Progress %
1	Project Outline	Mar 22, 2022	Mar 31, 2022	10 days	100
2	Requirement Gathering	Apr 01, 2022	Apr 10, 2022	10 days	100
3	Selection of Cloud Platform	Apr 11, 2022	Apr 15, 2022	5 days	100
4	System Architecture Design	Apr 16, 2022	Apr 25, 2022	10 days	100
5	System Deployment	Apr 26, 2022	Apr 29, 2022	4 days	100
6	System Testing	Apr 30, 2022	May 09, 2022	10 days	40

Table 2: Project Plan and Current Progress

Gannt Chart representing timeline of each activity and the time taken:

ID	Name	Duration
1	Project Outline	10 days
2	Requirement Gathering	10 days
3	Selection of Cloud Platform	5 days
4	System Architecture Design	10 days
5	System Deployment	4 days
6	System Testing	10 days

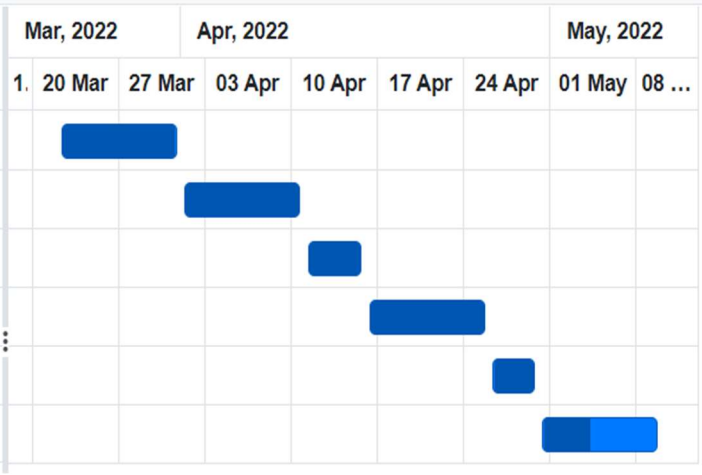

Figure 1: Gannt Chart of Project Plan

3. CLOUD ARCHITECTURE

3.1. REQUIREMENTS GATHERING:

There are two types of Requirements which are essential for any application to operate:

- **Functional Requirements**: In simple terms, Functional requirements states "What a system should do."
- **Non-Functional Requirements:** In a similar way, Non-Functional Requirements states "What a system should be".

Table listing both functional and Non-Functional Requirements:

Sr.	Functional Requirements	Non-Functional Requirements
1	Users should be able to register their accounts using their emails.	The UI of the player should be user friendly and include all the basic features.
2	Artists should have a different account from the normal users, which allows them to publish their tracks.	The Architecture should be robust and should be able to scale in and out depending on the traffic.
3	Users should be able to create a playlist, and the playlist should be easily shareable with others.	The compliances advised by the local Government should be followed/respected.
4	The Music library should have options to sort the albums by: Artist, Genre, Date etc	
5	There should be a Chart List which include all the songs which are trending.	

Table 3: Requirements

3.2. CLOUD SOLUTION:

There are many ways in which this project can be deployed on the cloud. Some of them are:

- Running a **container inside a Virtual Machine,** this is the option giving developer the most amount of granularity and control over the cloud infrastructure after it is deployed.
- Using **Container Orchestration Services,** this is the option, which is rather easy for deployment, cost significantly lower than running containers in a Virtual Machine and provides decent amount of granularity and control over the cloud infrastructure after it is deployed.
- Using **fully managed service to run containers**, this is the option which is easy to deploy as well as cost significantly lower than the above solutions, but the caveat is you lose control over the cloud infrastructure once deployed.

For this project we have chosen the last solution as using fully managed service has far greater advantages to overlook, and the only limitation the service is, once deployed, we lose some control over the resources as a fully managed resource means it can scale up and down according to the demand.

3.3. SELECTION OF TYPE OF CLOUD PLATFORM:

There are many Cloud Providers which provide excellent solutions for our purpose, but we are going to go with Amazon Web Services (AWS), as AWS is hands-down the biggest player in this industry offering more services than any other cloud provider. The main reason for this decision is AWS provides the perfect set of services to deploy and maintain our application easily.

3.4. CHOICE OF DATA CENTRE AND STANDARDS:

As the users will be from all around the world, the system should be reachable from all round the world with less latency. Hence, our System will include datacentres like Singapore to cover the Asia Pacific shard, Frankfurt to cover EMEA shard and US East to cover both Americas shard.

AWS complies with the local rules, so we don't need to worry about the local compliances. The only thing we need to do is to keep the user data of users from a continent in the datacentre which belongs to same continent to comply with the Data Protection Policies.

3.5. SYSTEM ARCHITECTURE DESIGN:

For understanding the architecture, first have a look at all services which we are going to use:

- **Availability Zones:** Availability Zones are the physical Datacentres, each geographic location(region) has multiple isolated Availability Zones to provide Disaster Recovery, prevent datacentre failures etc.
- **Virtual Private Cloud (VPC):** Virtual Private Cloud (VPC) is an Amazon service which allows to deploy the AWS services in a virtual network that can be defined by the user. It acts as a traditional network in a private datacentre, but with AWS services.
- **Network Address Translation (NAT) Gateway:** As the name suggests, it is a Network Address Translation service, which is used to connect resources in a Private Subnet with the resources on the internet, the services outside VPC cannot initiate a connection with these Private subnet resources.
- **Subnets:** Subnet is a range of IP addresses in a VPC, AWS resources can be launched in a subnet. There are 2 main types of Subnets namely Public and Private. Public Subnet has direct access to the Internet, whereas Private subnet do not, instead it is connected to a NAT Gateway.
- **Route Tables:** Route Tables are a set of rules which defines where the network traffic from your subnet/resources is directed.
- **Security Groups:** A Security Group controls the traffic that is allowed to receive or leave the resource, basically a virtual Firewall.
- **AppRunner:** AWS AppRunner is a fully managed service which allows developer to focus on the development of the application and takes care of scaling, load balancing, automatic/scheduled building and deployment of the application.
- **DynamoDB:** It is a fully managed NoSQL Database service in AWS, which supports key-value pairs and document data structures.

Our system is deployed in 3 regions. Each region will be deployed with a VPC which will have Public and Private subnets in 2 availability zones for redundancy and backup as shown in the figure below.

This figure represents the VPC containing the application hosted using Amazon AppRunner service deployed in public subnets in 2 separate availability zones. AppRunner will be deployed with a configuration of minimum 1 container (during non-peak hours) and it can scale up to 10 containers (during peak-hours). Each container will have 1 vcpu and 2gb of memory. During a day 3 hours are considered as peak hours, and the remaining hours as non-peak hours.

Similarly, an Amazon DynamoDB cluster containing 2 nodes (primary and secondary) are deployed in private subnets in 2 availability zones, the secondary DynamoDB node will be a live backup of the primary DynamoDB node. The reason we chose DynamoDB as our Database is that it is a NoSQL based fully managed database service. As the database cluster is deployed in private subnets, it is secure and has no direct access to public internet.

We have created 2 Security Groups for hosting AppRunner and DynamoDB instances which will allow specific requests required by the services and block all the other requests.

Similarly, VPC containing AppRunner and a DynamoDB cluster containing 2 nodes are deployed in each of our selected 3 regions mentioned earlier.

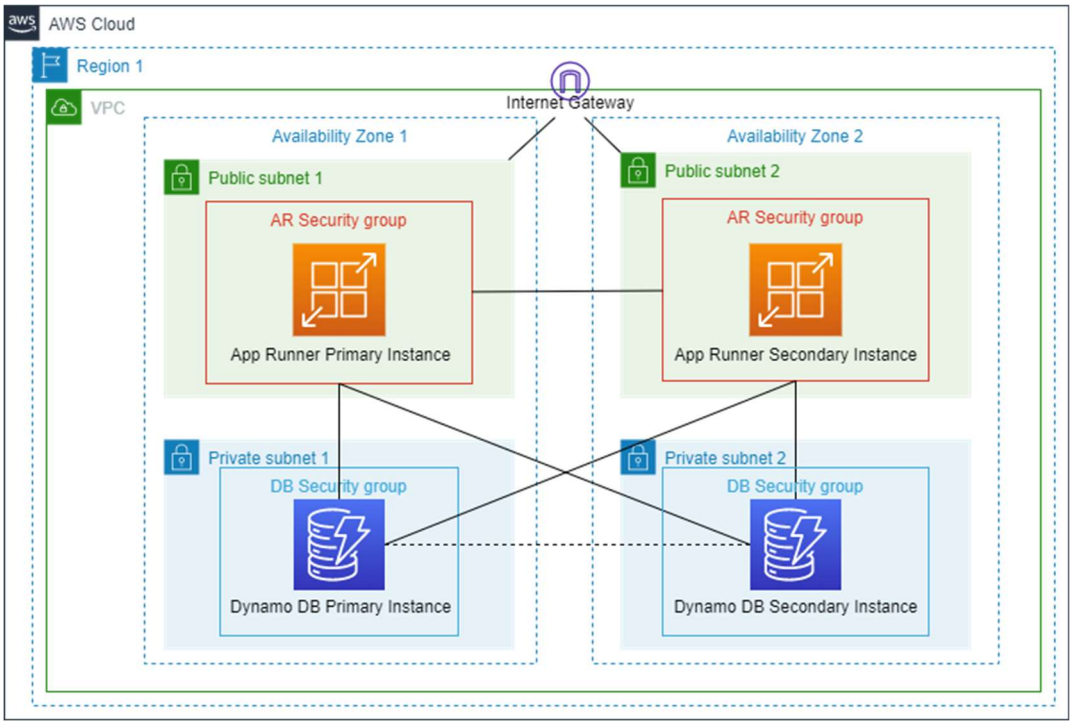

Figure 2: System Architecture Design for each Region

3.6. IMPLEMENTATION:

To successfully implement our Cloud Architecture, we need to do the following:

- **Create VPC:**

  Step 1: In the AWS Dashboard, search for VPC and select it.

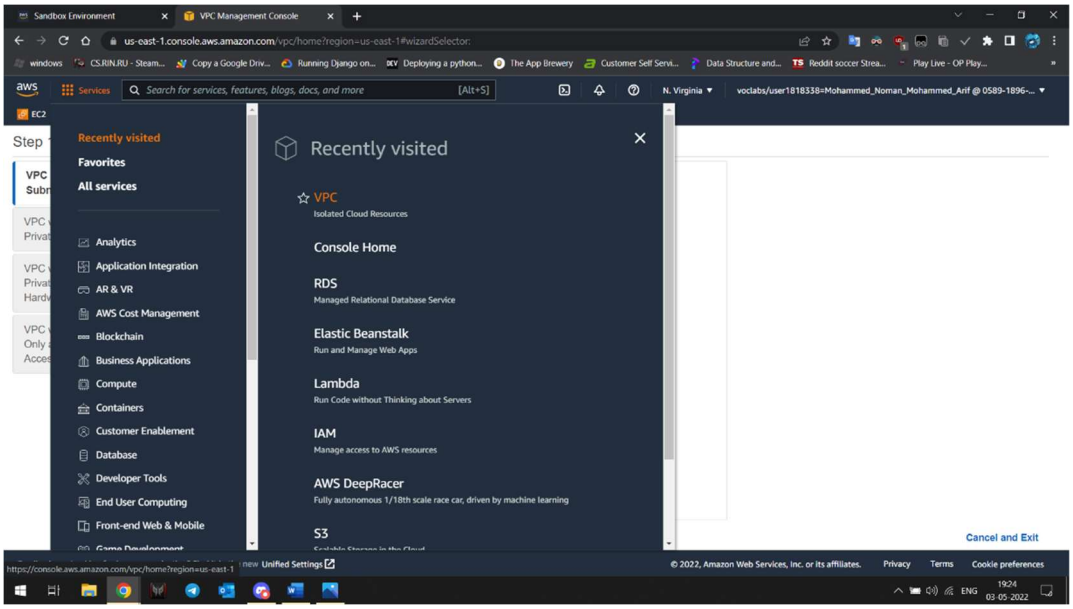

Figure 3.1.1: Creating VPC

Step 2: Select Launch VPC wizard

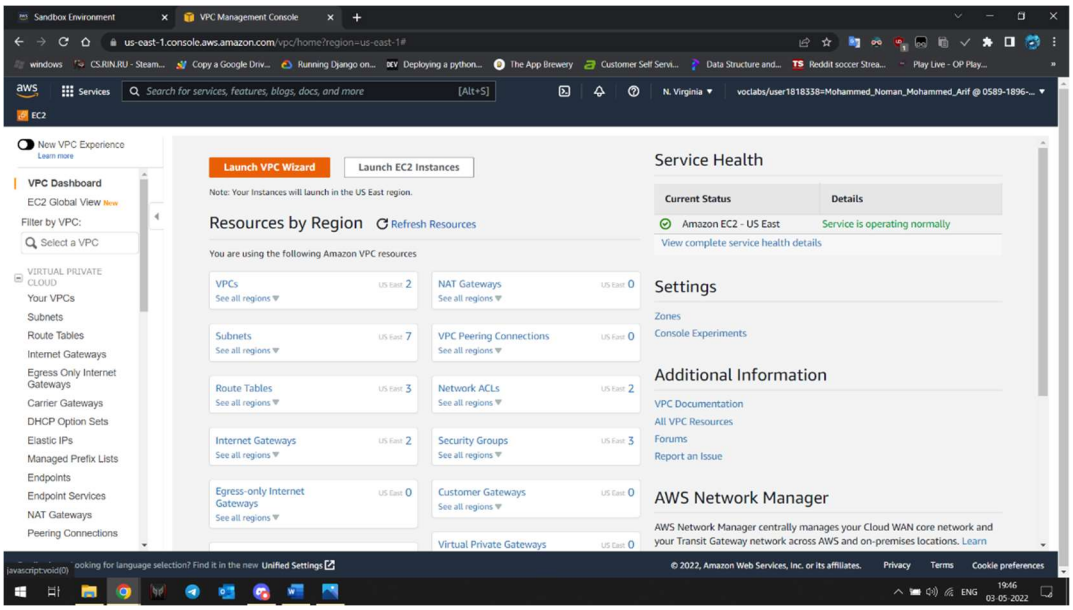

Figure 3.1.2: Creating VPC

Step 3: Select second option, “VPC with public and private subnets”

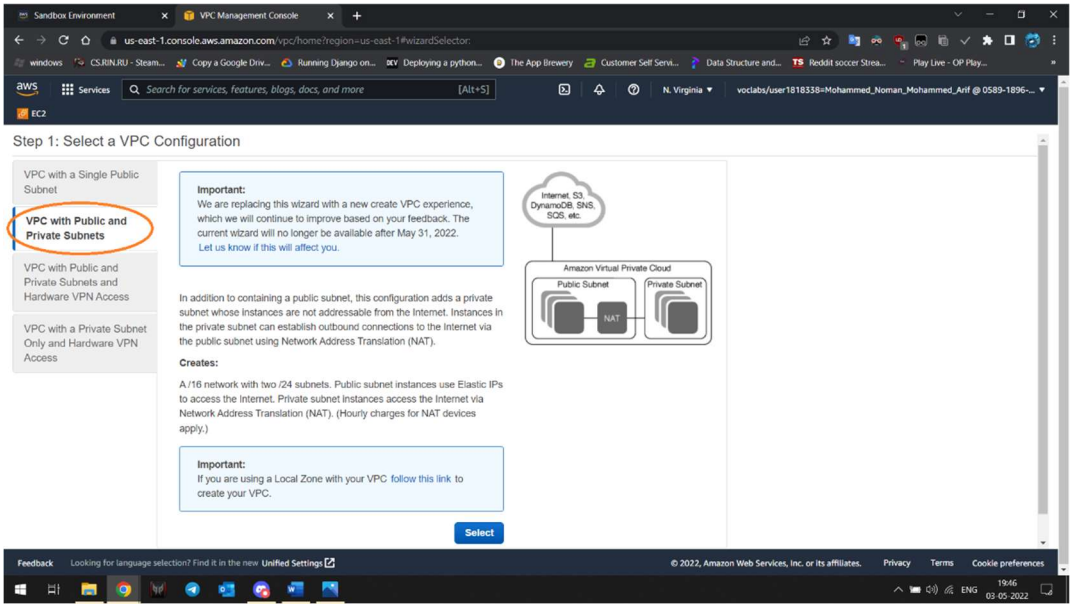

Figure 3.1.3: Creating VPC

Step 4: Configure the VPC settings (make sure to select an availability zone for both subnets and note it down, as we will need it to create two more subnets in different zone than this one), Then select create VPC and wait for the service to be deployed.

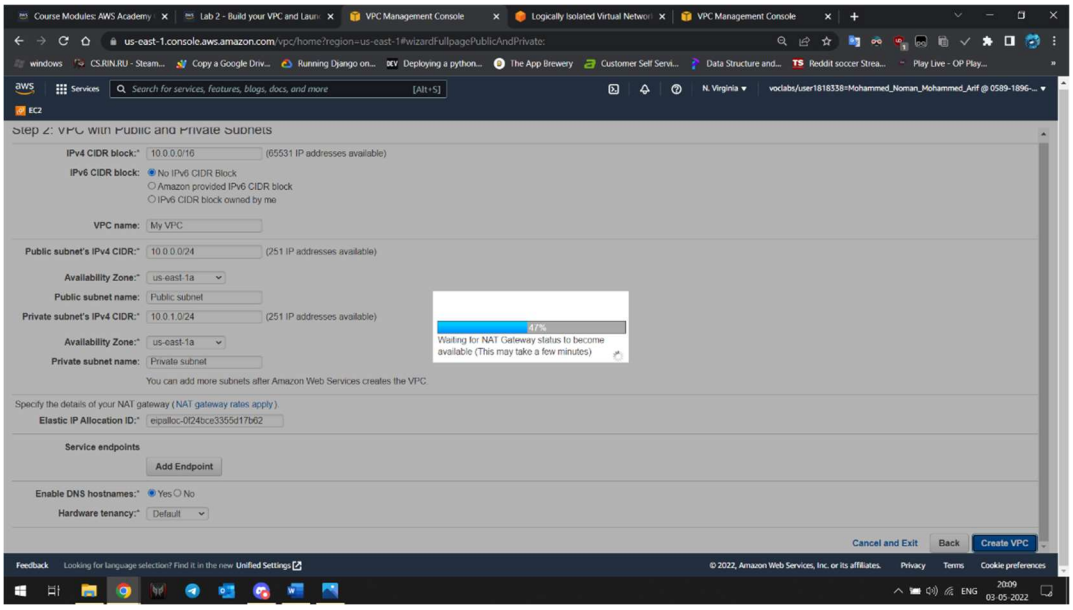

Figure 3.1.4: Creating VPC

And the VPC is deployed successfully!

- **Create Additional Subnets:**

Step 1: Go to subnets and select Create subnet.

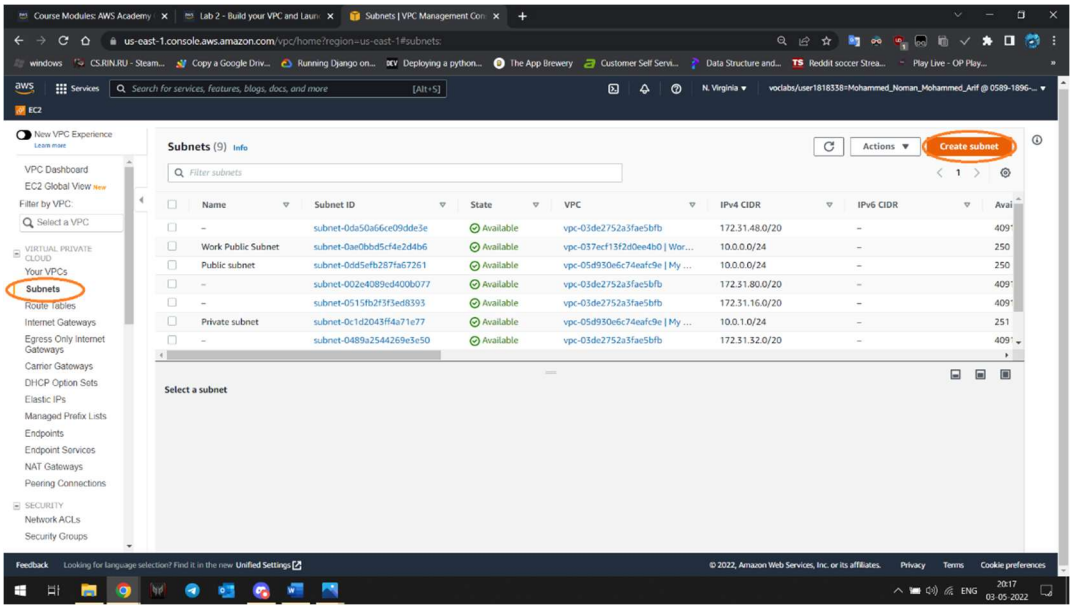

Figure 3.2.1: Create Additional Subnets

Step 2: Configure the VPCs as shown below to create a public and a private subnet, and then select create. This will create the subnets.

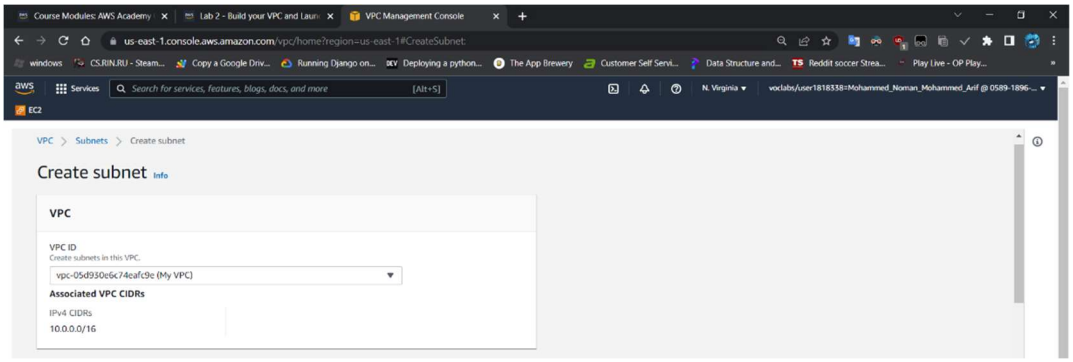

Figure 3.2.2: Create Additional Subnets

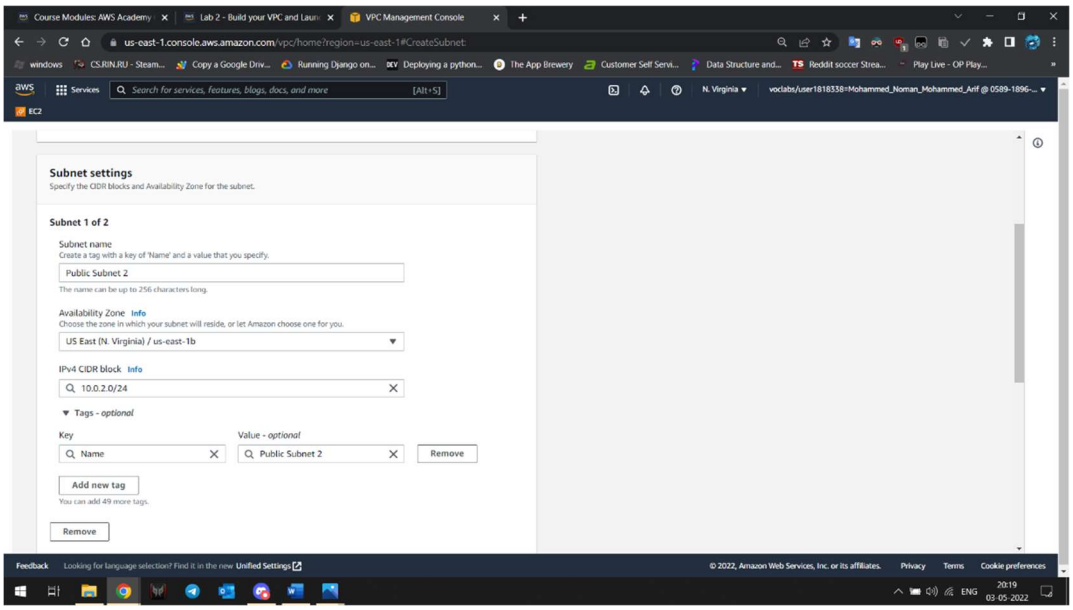

Figure 3.2.3: Create Additional Subnets

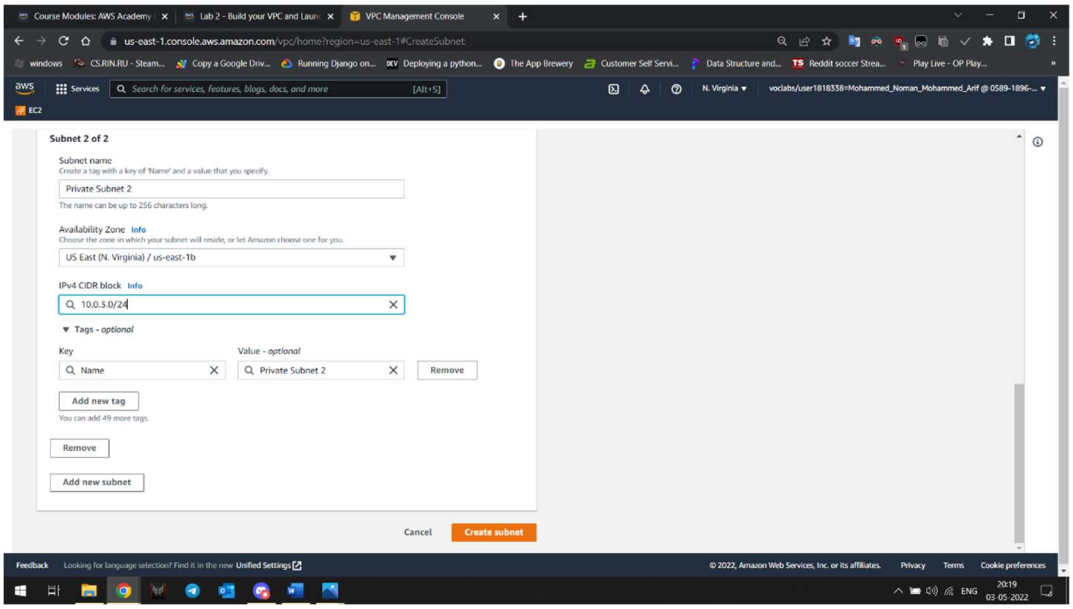

Figure 3.2.4: Create Additional Subnets

Required subnets are created.

- **Assign Route Tables to Subnets:**

  Step 1: Go to Route Table Subsection in VPC Blade, select the route table with 'VPC: My VPC' and 'Main: Yes', and rename it to Private Route Table. This Route Table does not allow the subnets associated with it to have public access, instead it routes through the NAT Gateway as shown below.

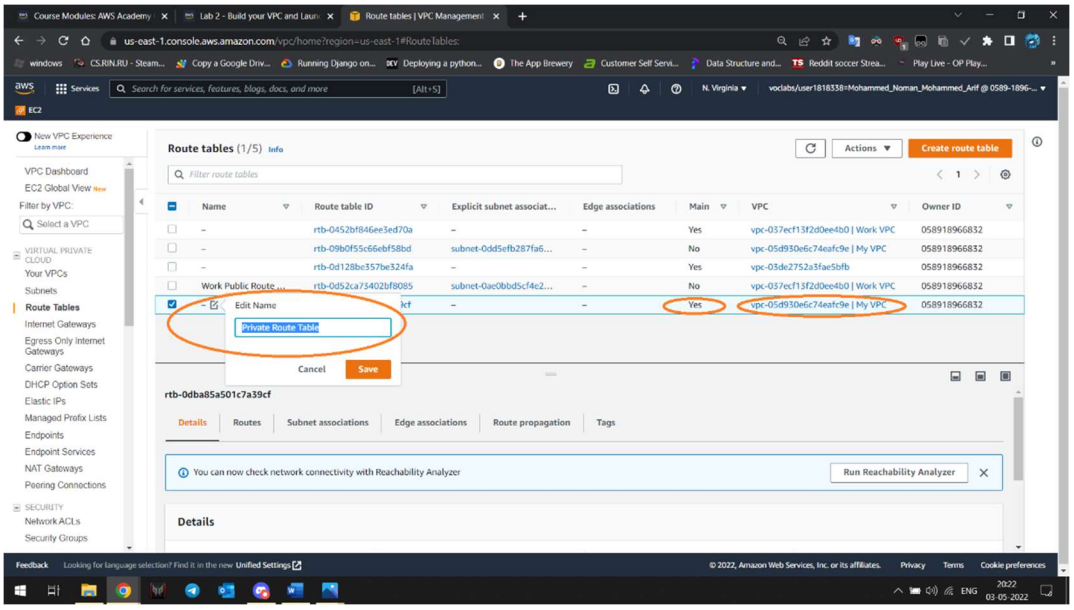

Figure 3.3.1: Assign Route Tables

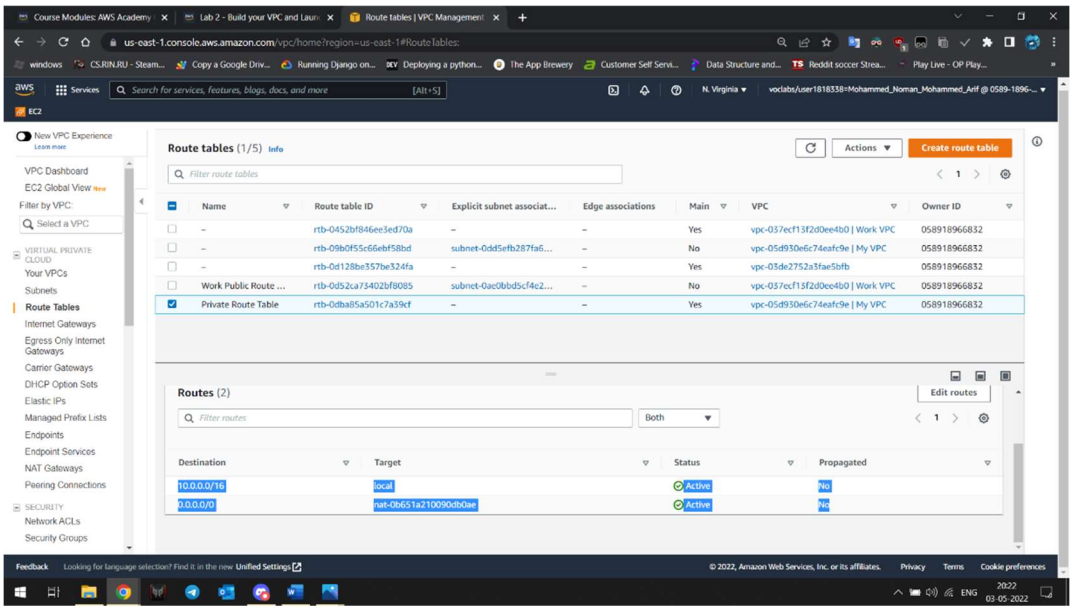

Figure 3.3.2: Assign Route Tables

Step 2: Attach both Private Subnets to this route Table as shown below:

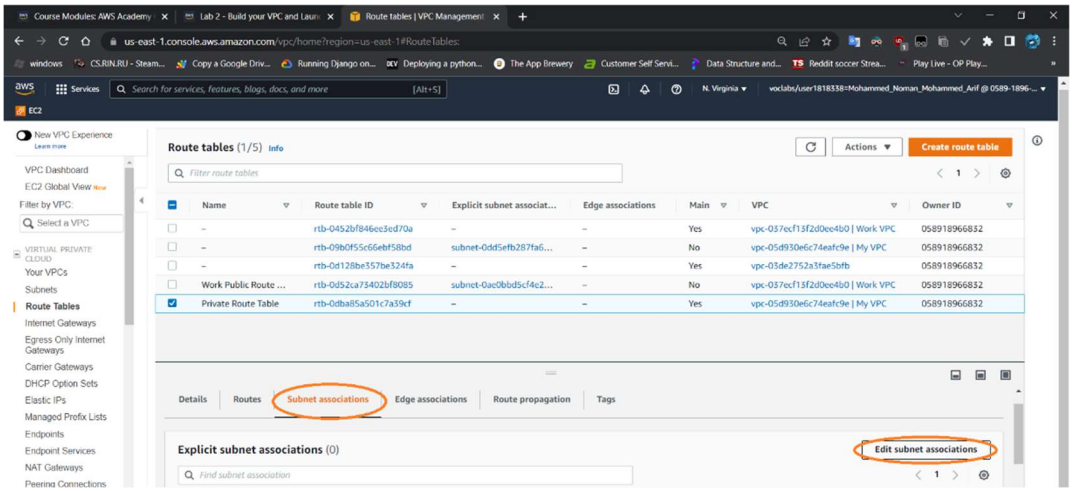

Figure 3.3.3: Assign Route Tables

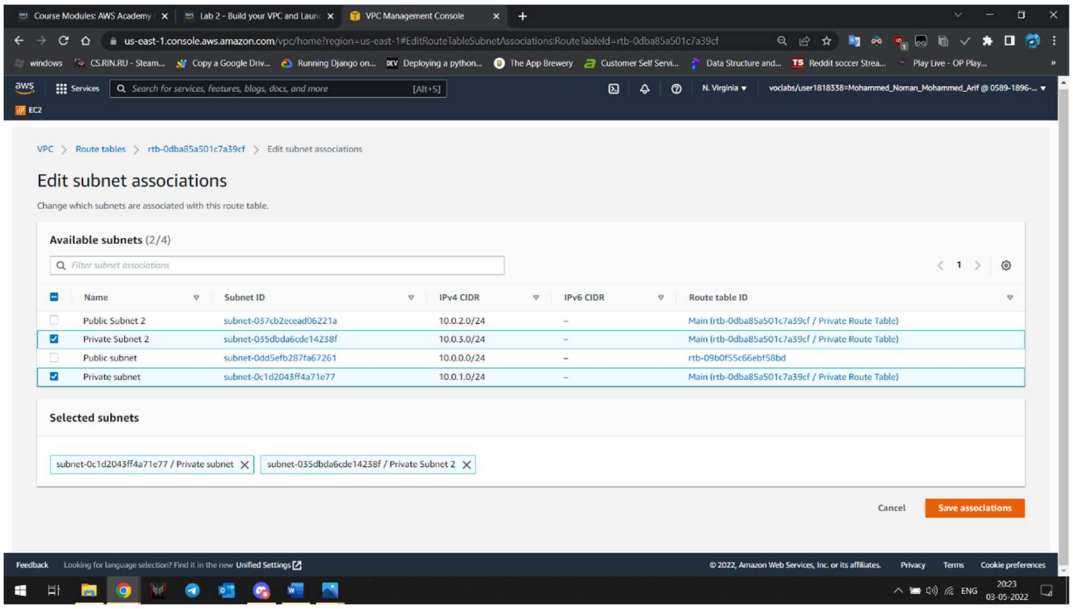

Figure 3.3.4: Assign Route Tables

Step 4: Similarly, Rename Route Table with ‘VPC: My VPC’ and ‘Main: No’ to Public Route Table, and attach both Public Subnets to it.

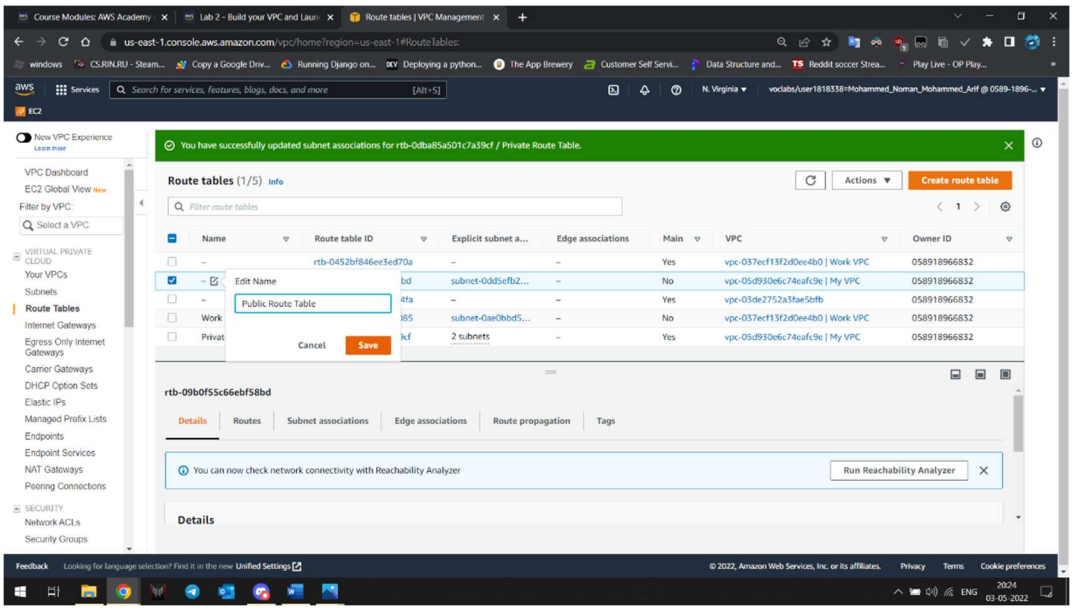

Figure 3.3.5: Assign Route Tables

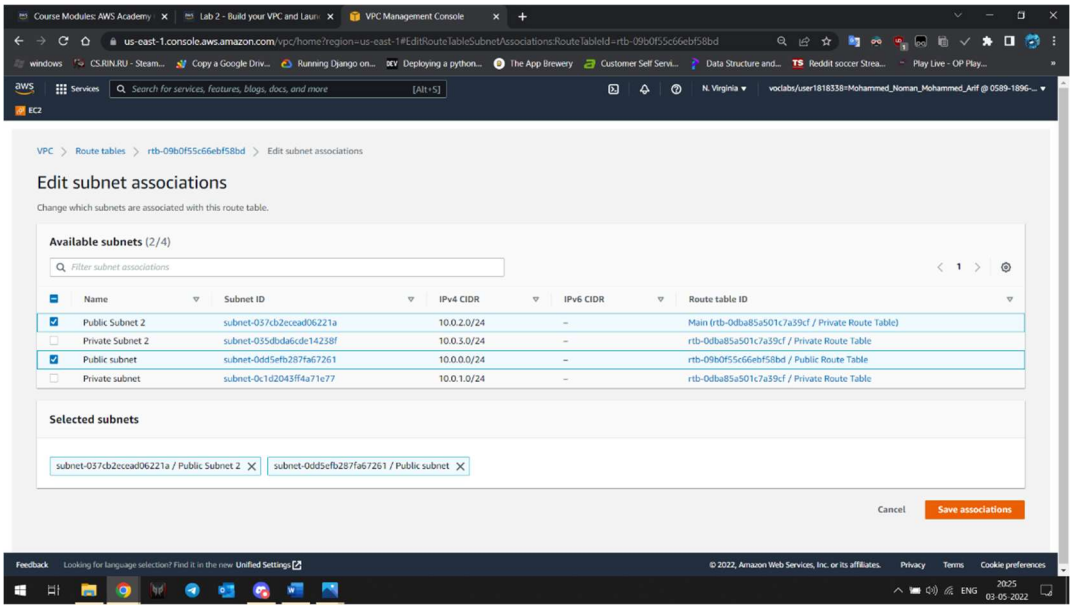

Figure 3.3.6: Assign Route Tables

And now, the Route Tables are attached to both Public and Private Subnets.

- **Security Groups:**

Step 1: Go to Security Groups sub-menu and select ‘Create security group’.

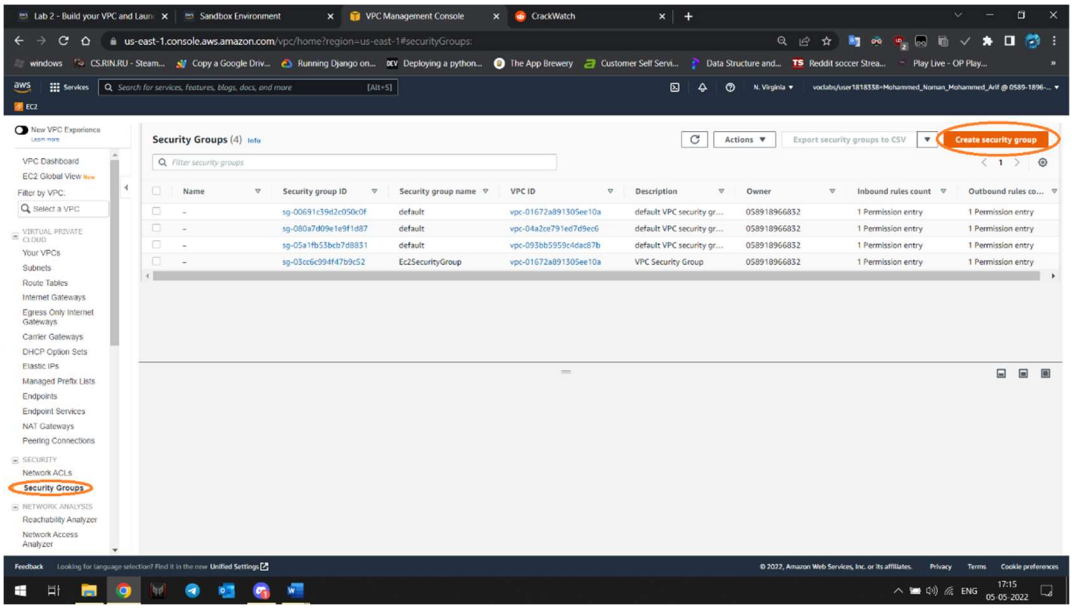

Figure 3.4.1: Creating Security Groups

Step 2: Create AR Security Group using the following configuration and allow HTTP requests for our API:

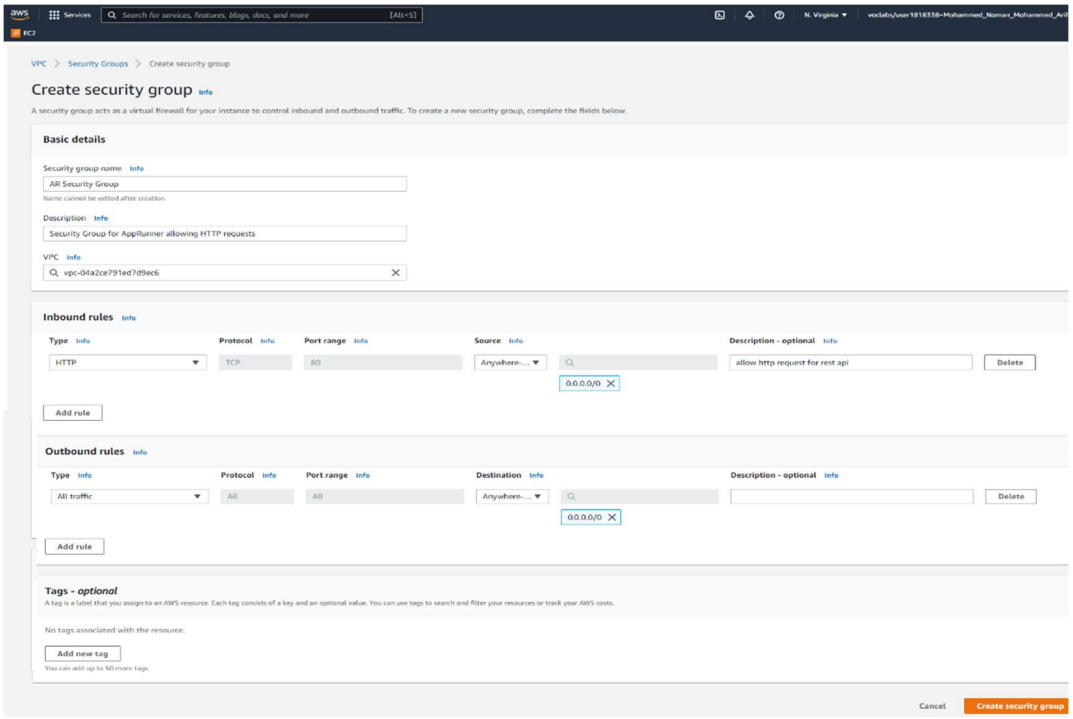

Figure 3.4.2: Creating AR Security Group

Step 3: Create DB Security Group using the following configuration and allow TCP inbound requests for our DynamoDB cluster:

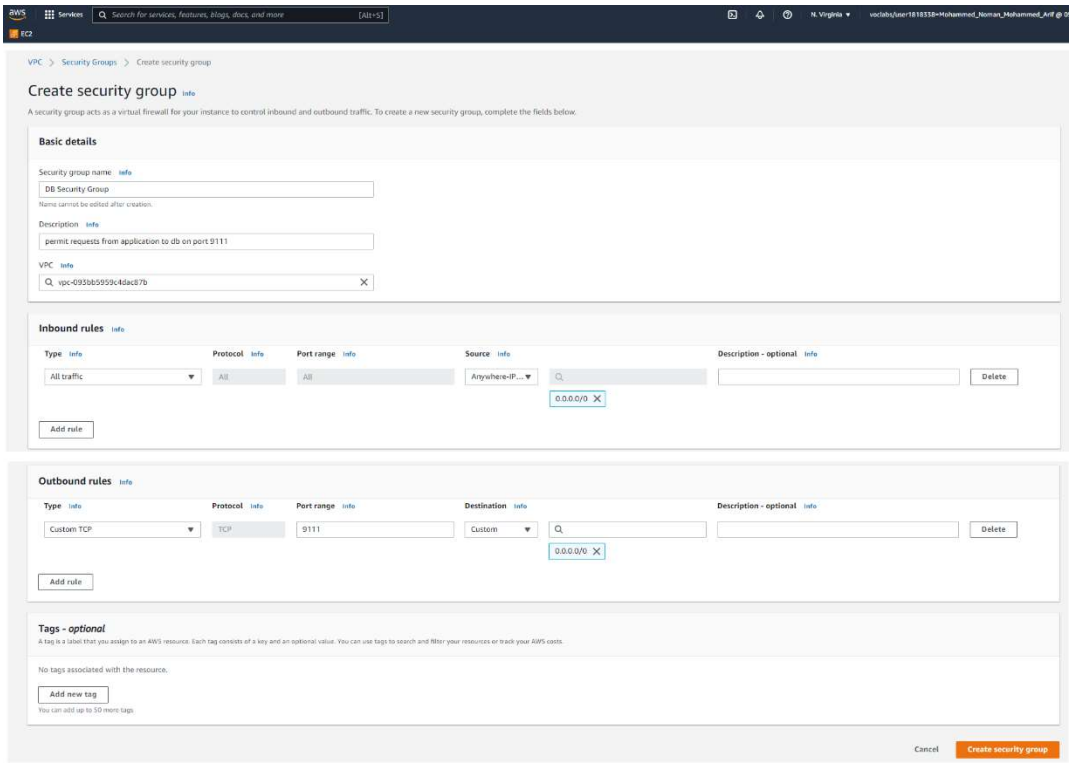

Figure 3.4.3: Creating DB Security Group

Both the Security Groups are created and ready to use.

- **DynamoDB Cluster:**

Step 1: Go to DynamoDB Dashboard and select 'Create DAX Cluster' and configure the cluster as shown below:

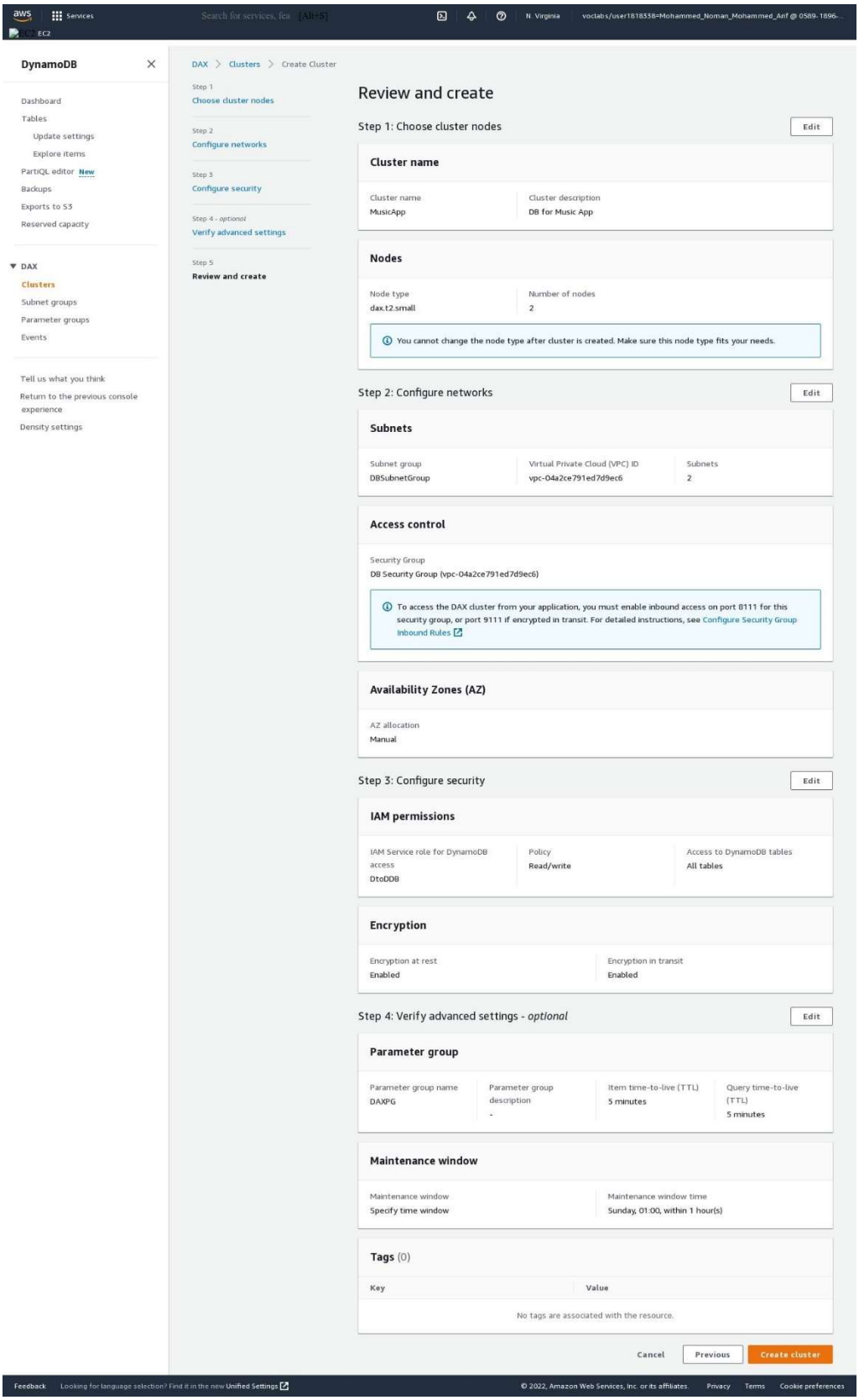

Figure 3.5.1: Creating DynamoDB Cluster

- **AppRunner instances:** AppRunner instance will be deployed in both our public subnets which are deployed in 2 separate availability zones for redundancy. Go to AppRunner Dashboard, select 'Create an AppRunner Service' and configure the AppRunner as follows:

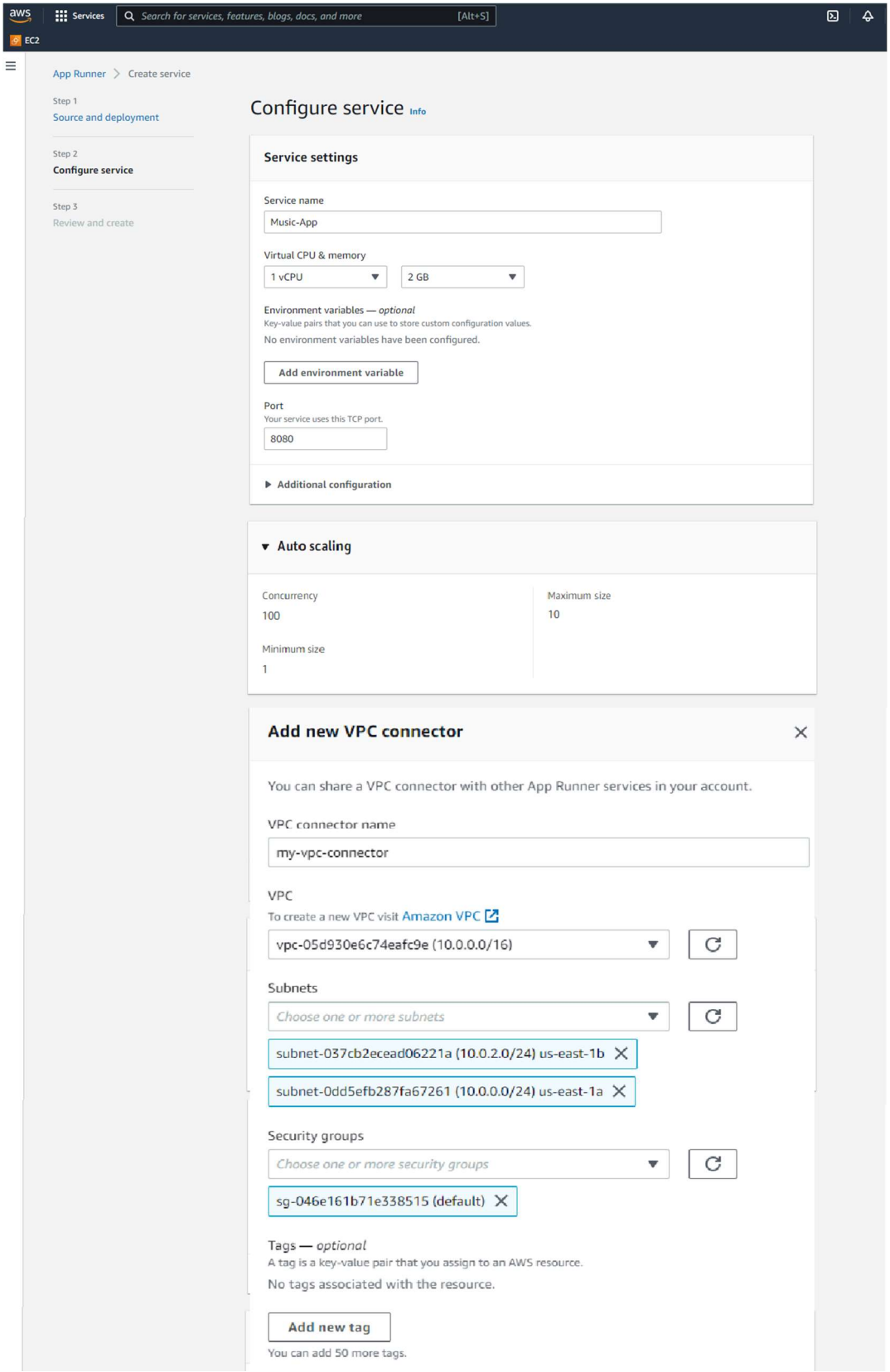

Figure 3.6.1: Create AppRunner

3.7. COSTING:

Following are the services, and the estimated usage of these services that we are going to deploy in each of our selected 3 regions:

Sr.	Service	Estimated Usage
1	Amazon Virtual Private Cloud (VPC)	1
2	NAT Gateway	1
3	NAT Gateway Data Transfer	1000GB/month
4	AWS AppRunner	1
5	AWS DynamoDB	1

Table 3: Cost Estimation

The above services when calculated in the AWS Cost calculator, rounds up to 239.33 USD per month or 2871.96 USD per annum.

The calculations are as follows:

Figure 3.7.1: AppRunner Costing

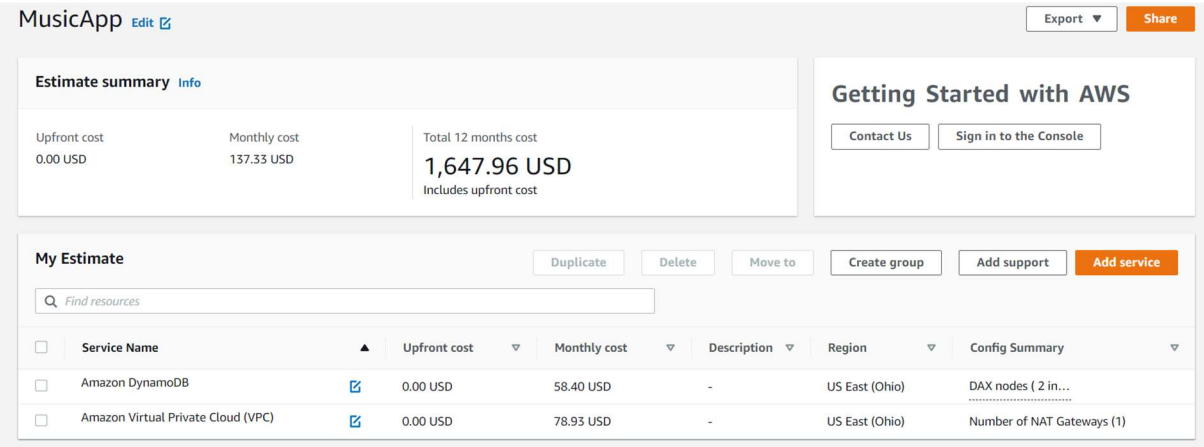

Figure 2.7.2: Costing

4. ANALYSIS AND REFLECTION

Hence, we have created an architecture to host our application on AWS cloud using AWS AppRunner. The architecture created is elastic, highly available, secure and fault tolerant. Whereas the cost of the whole architecture is also very reasonable as we are charged for only what we are using.

As far as the maintenance is concerned, AppRunner and DynamoDB, both are fully managed services and can scale in and out as per the change in amount of demand/traffic. Maximum Cap of container instances can also be set to avoid bills getting over the budget.

Three regions are selected to meet the demand over the globe, further, as the demand increases with time, the application can be deployed in those regions which are demanding more resources.

5. REFERENCES

1. https://aws.amazon.com/application-hosting/benefits/
2. AWS Calculator link
3. https://www.luminis.eu/blog/cloud-en/a-first-impression-of-aws-app-runner/
4. https://aws.amazon.com/apprunner/faqs/
5. https://aws.amazon.com/apprunner/pricing/
6. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.create-cluster.console.configure-inbound-rules.html