**Name: M.Nithiya Ananthi**

**Session : AWS Weekday BC = 2118052505**

#### Course Name: Cloud and DevOps Architect Master’s Course

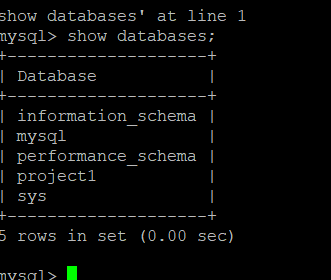
Project 1 - Deploying a Multi-Tier Website Using AWS EC2 Topic: Deploy a Multi-tier website using EC2

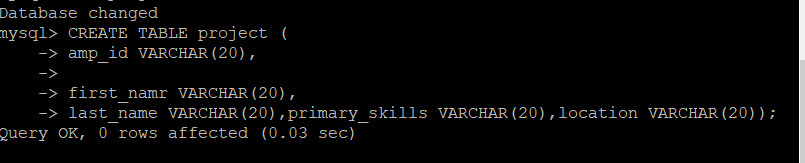
Description: Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

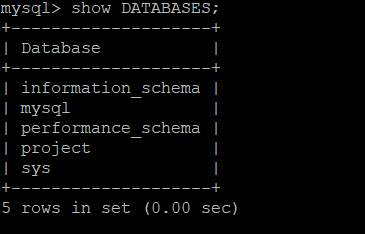
Problem Statement: Company ABC wants to move their product to AWS. They have the following things setup right now:

1. MySQL DB

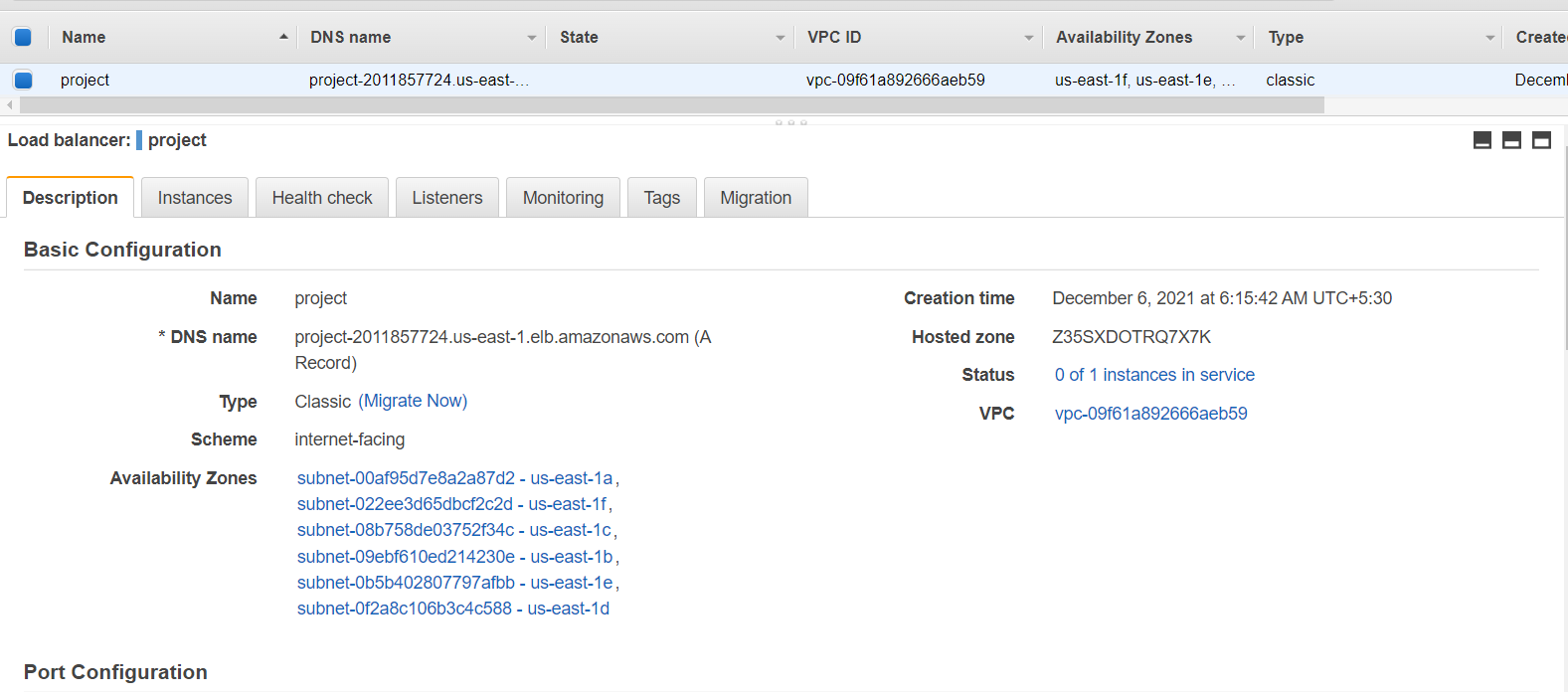




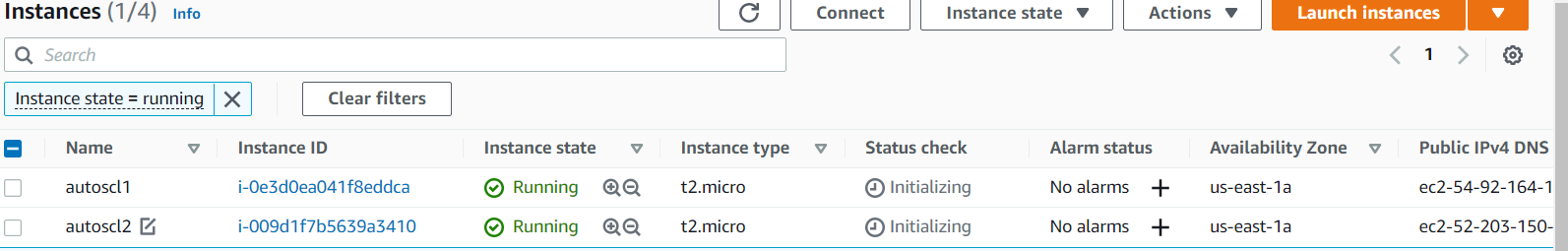




Creating Load balancers:



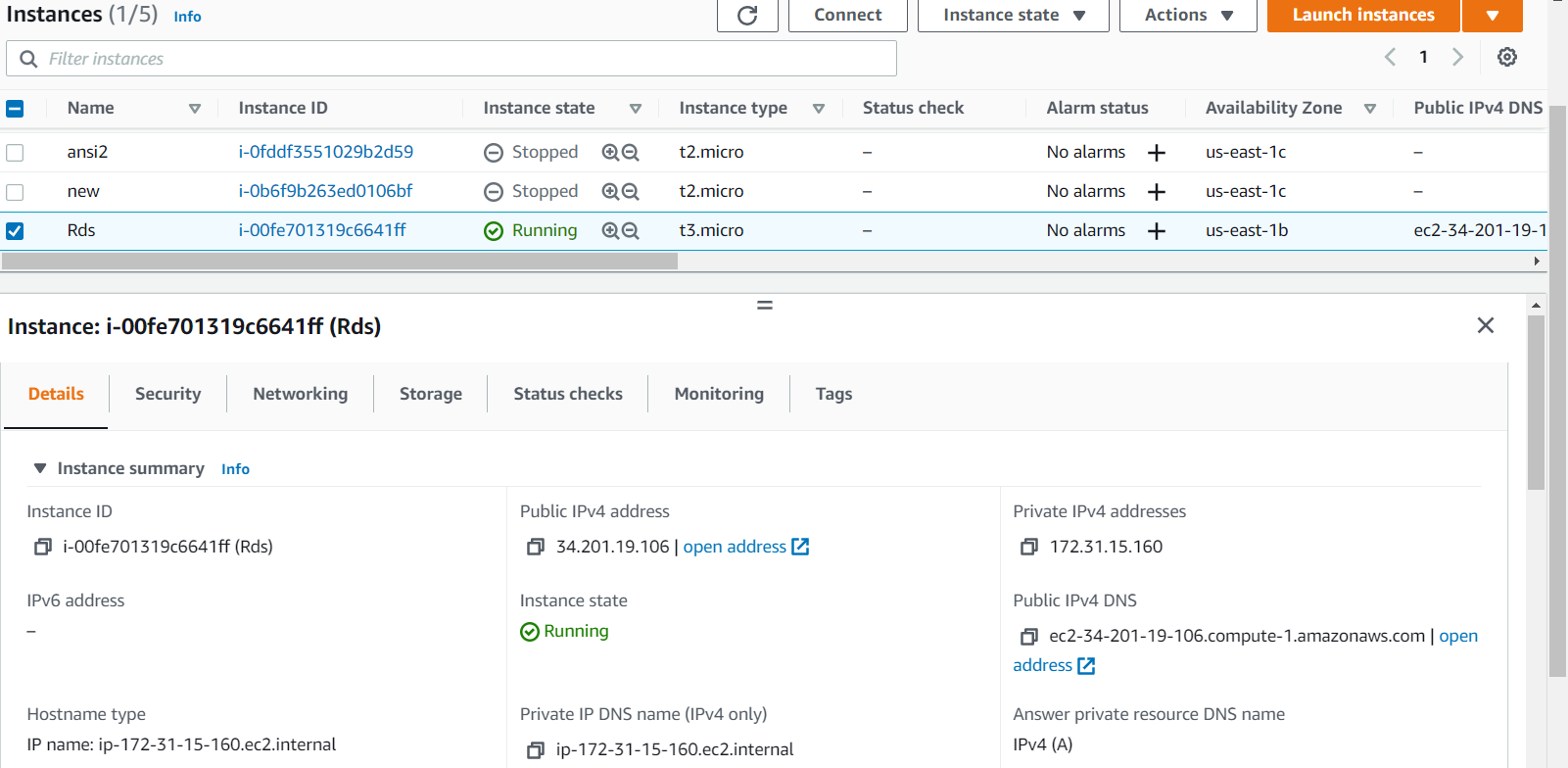
Autoscaling group:



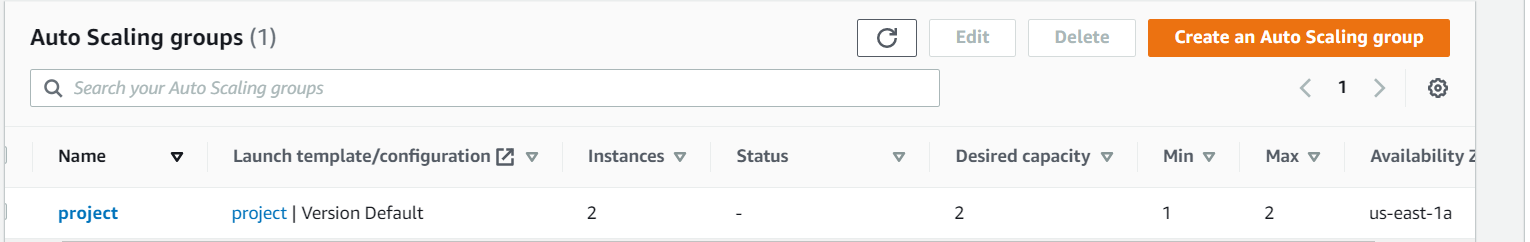
2. Website (PHP) The company wants high availability on this product, therefore wants autoscaling to be enabled on this website.

Steps to solve:

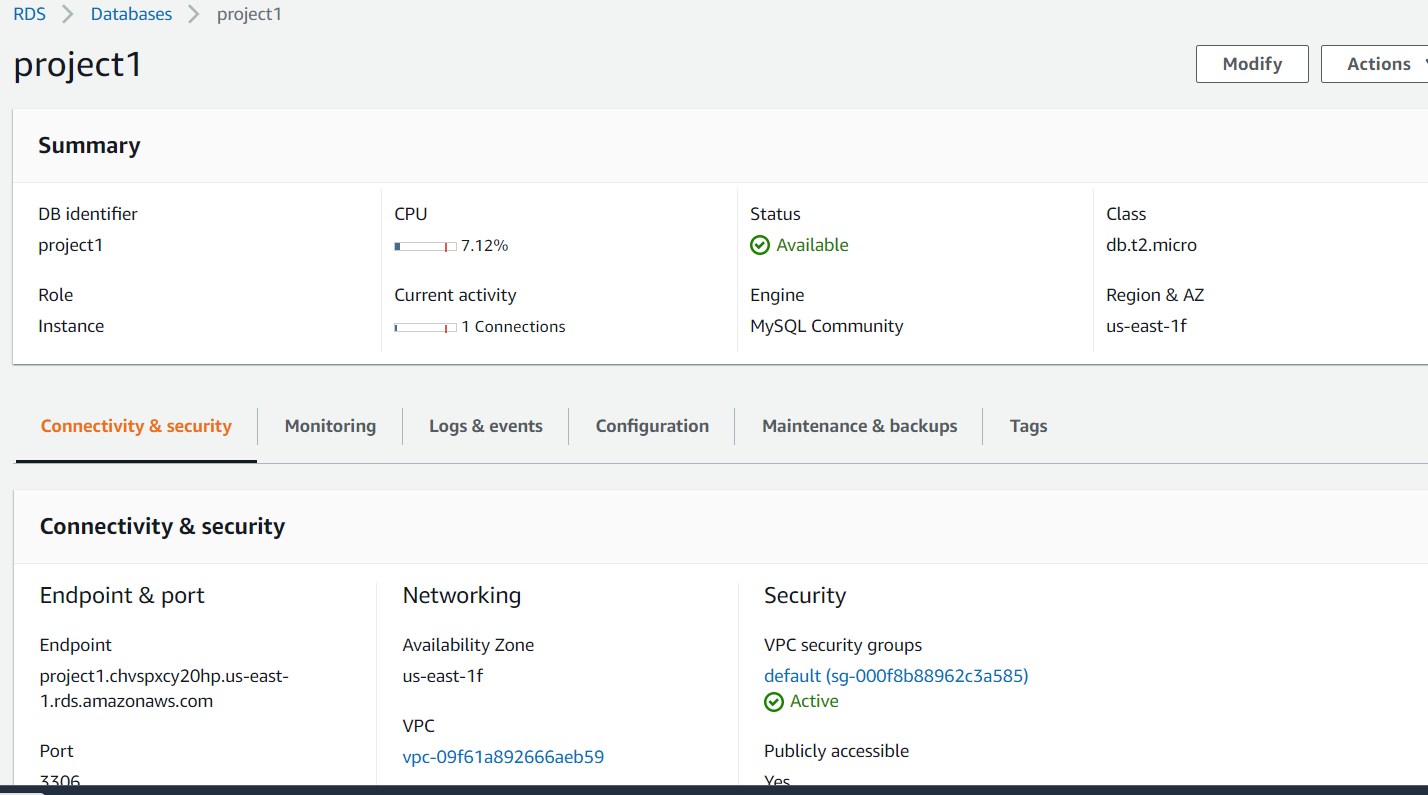
1. Launch an EC2 Instance



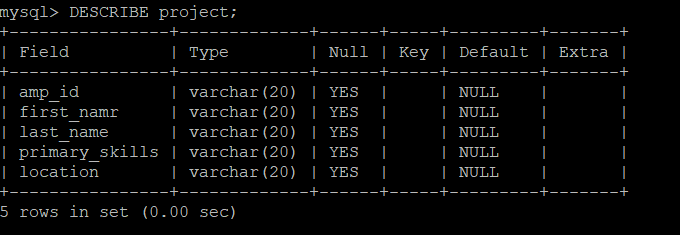
1. Enable Auto Scaling on these instances

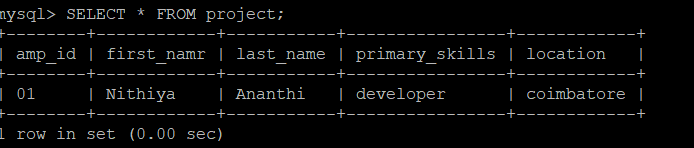
****

3. Create an RDS Instance

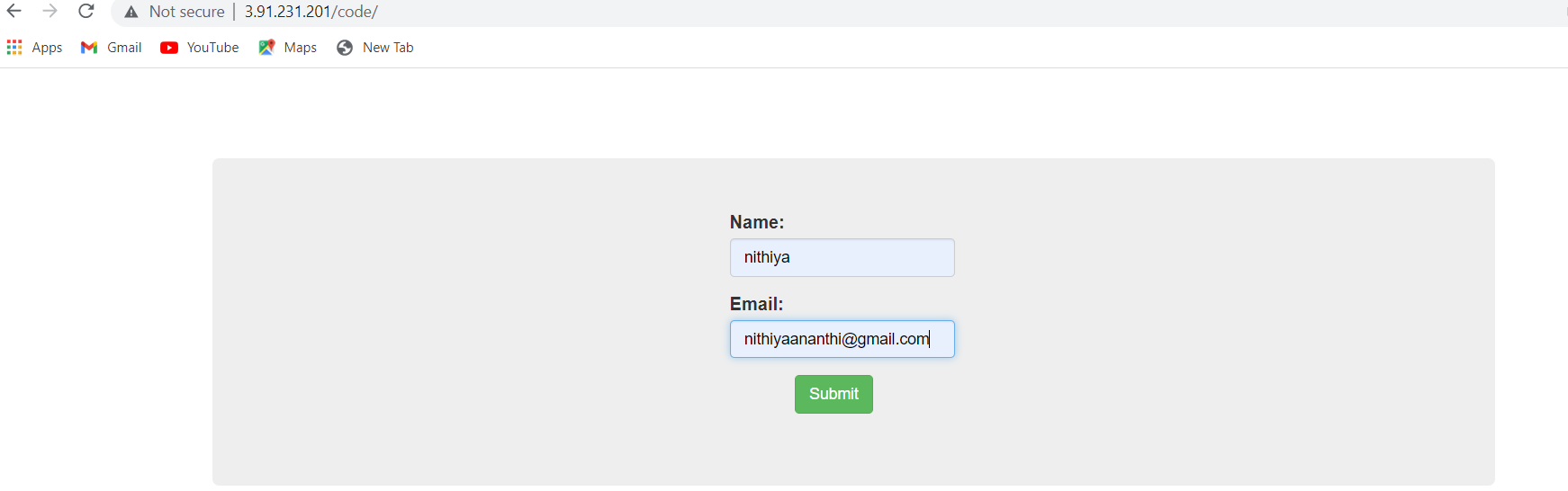


4. Create Database & Table in RDS Instance • Database name: intel • Table name: data • Database password: intel123

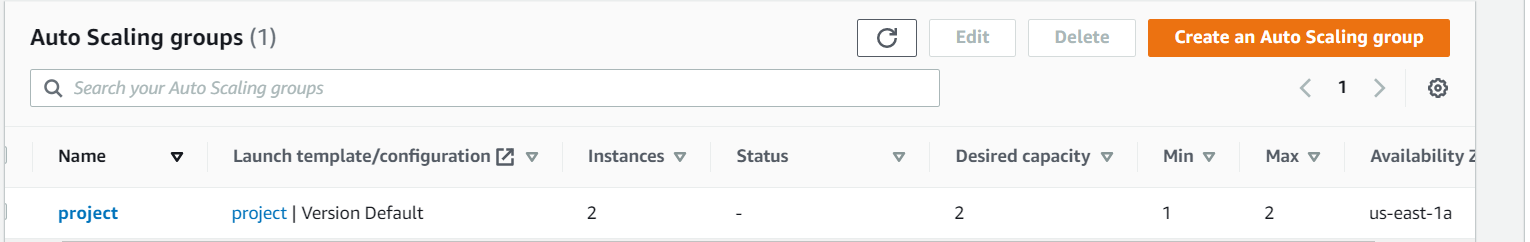




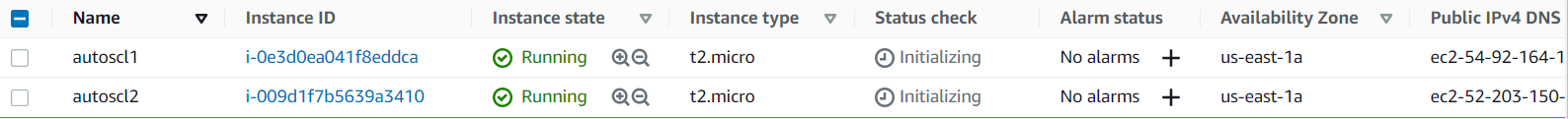
5. Change hostname in website



6. Allow traffic from EC2 to RDS Instance

****

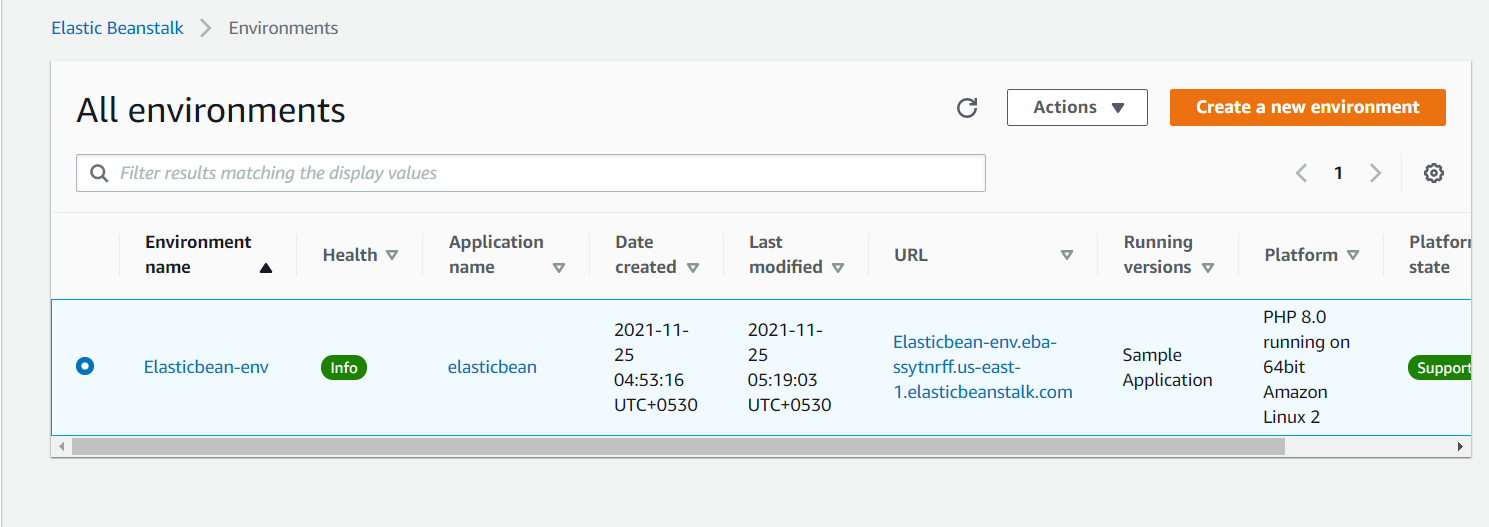
7. Allow all-traffic to EC2 instance

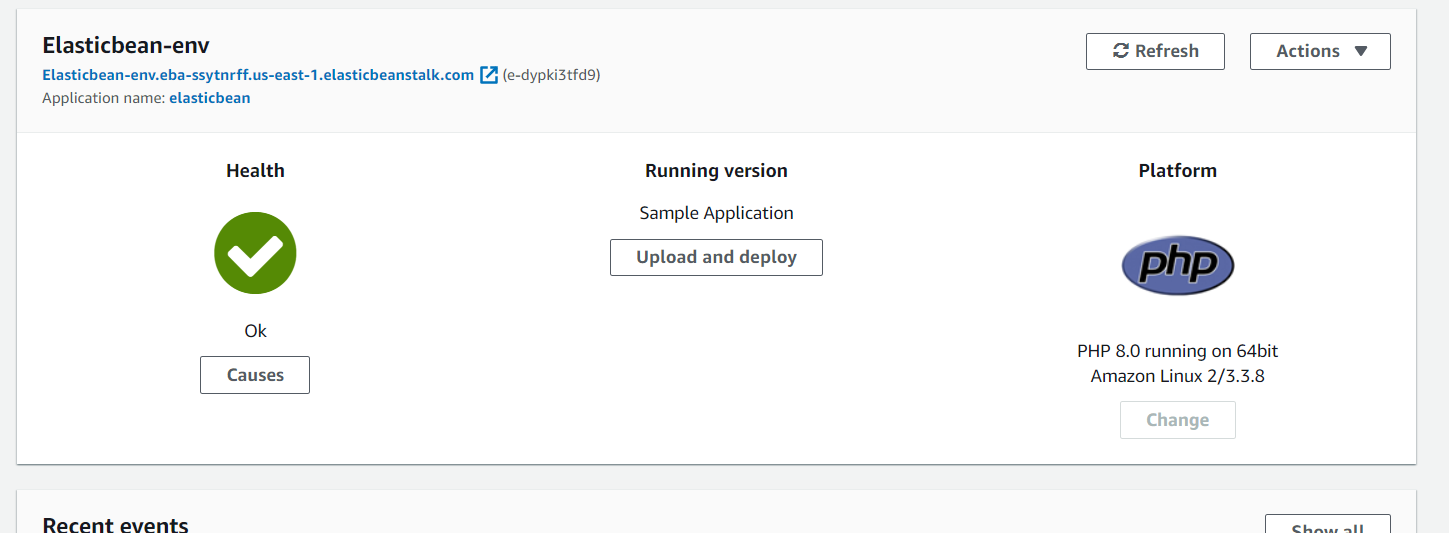


Project 2 – Website Orchestration Project: Website Orchestration Industry: Internet related Problem Statement: How to orchestrate a website with lesser time and higher availability along with autoscaling.

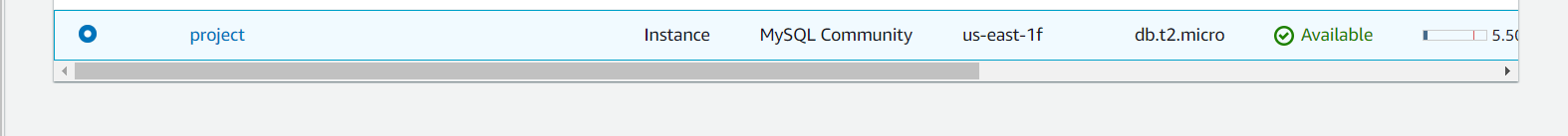
Topics: In this AWS project, you have to deploy a high-availability PHP application with an external Amazon RDS database to Elastic Beanstalk. Running a DB instance external to Elastic Beanstalk decouples the database from the lifecycle of your environment. This lets you connect to the same database from multiple environments, swap one database for another, or perform a blue/green deployment without affecting your database.

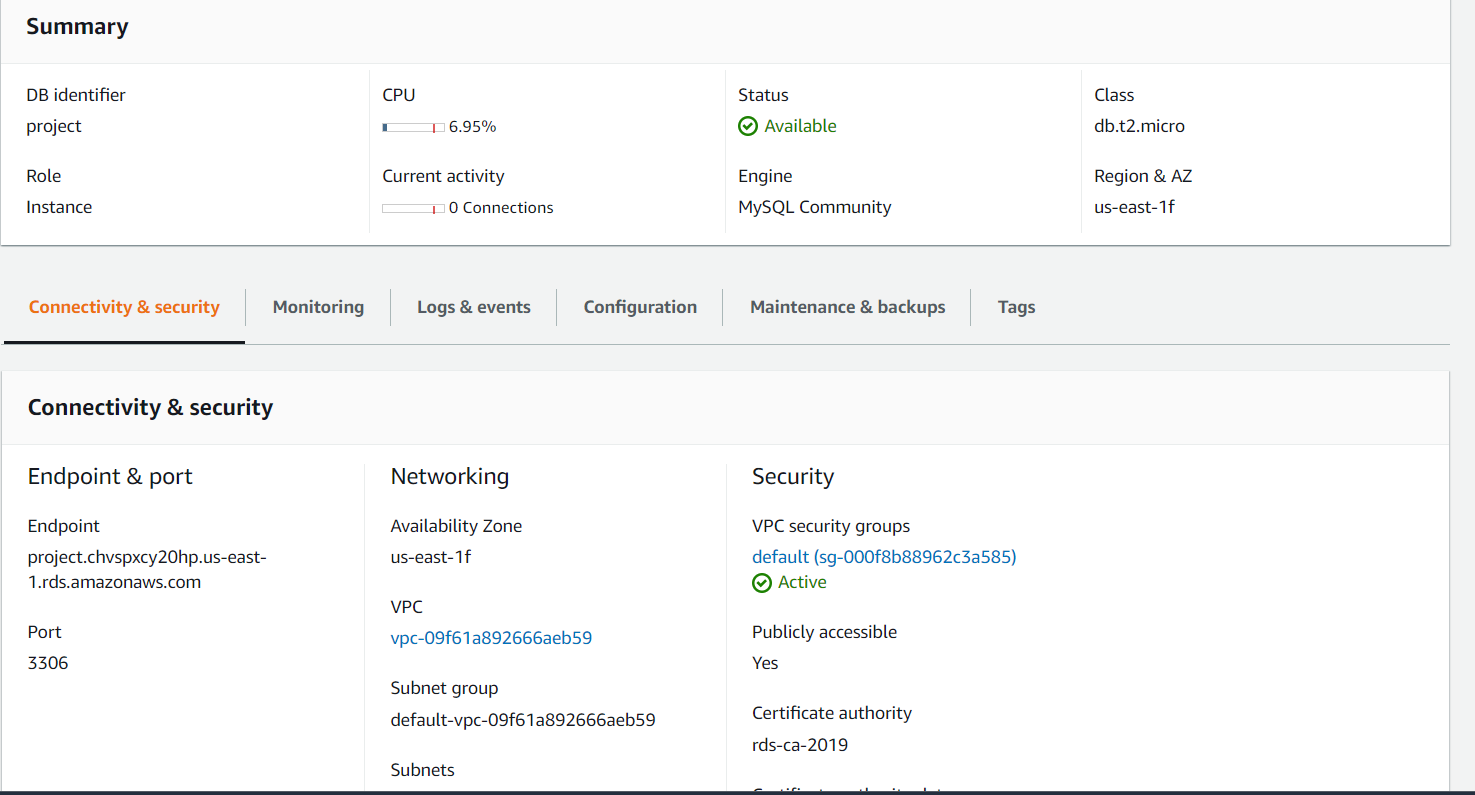
Highlights: Launch a DB instance in Amazon RDS Create an Elastic Beanstalk Environment Configure Security Groups and Scaling

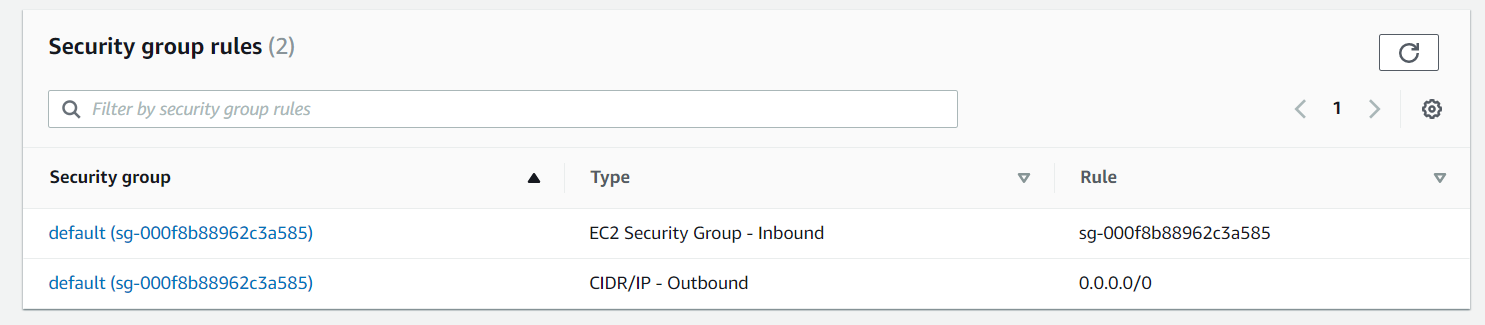




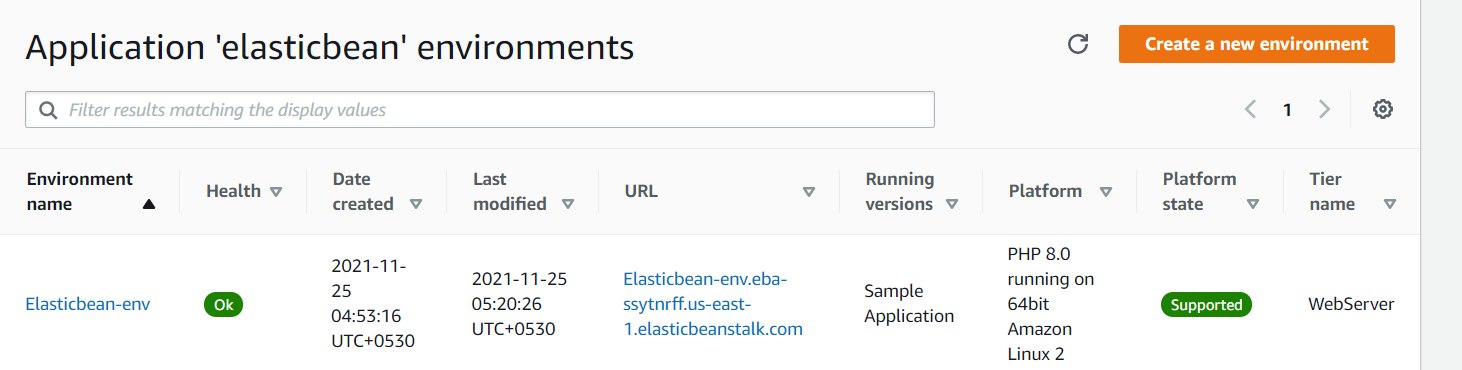
Creating a DB instance:



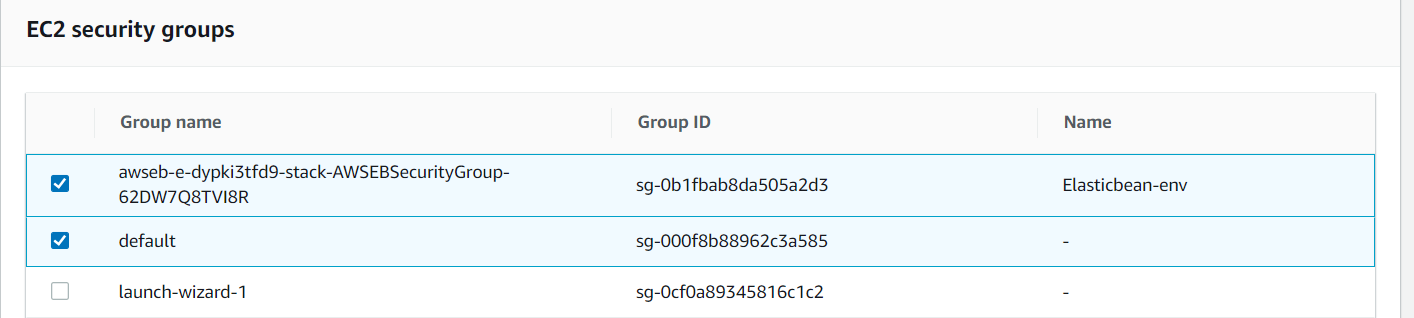


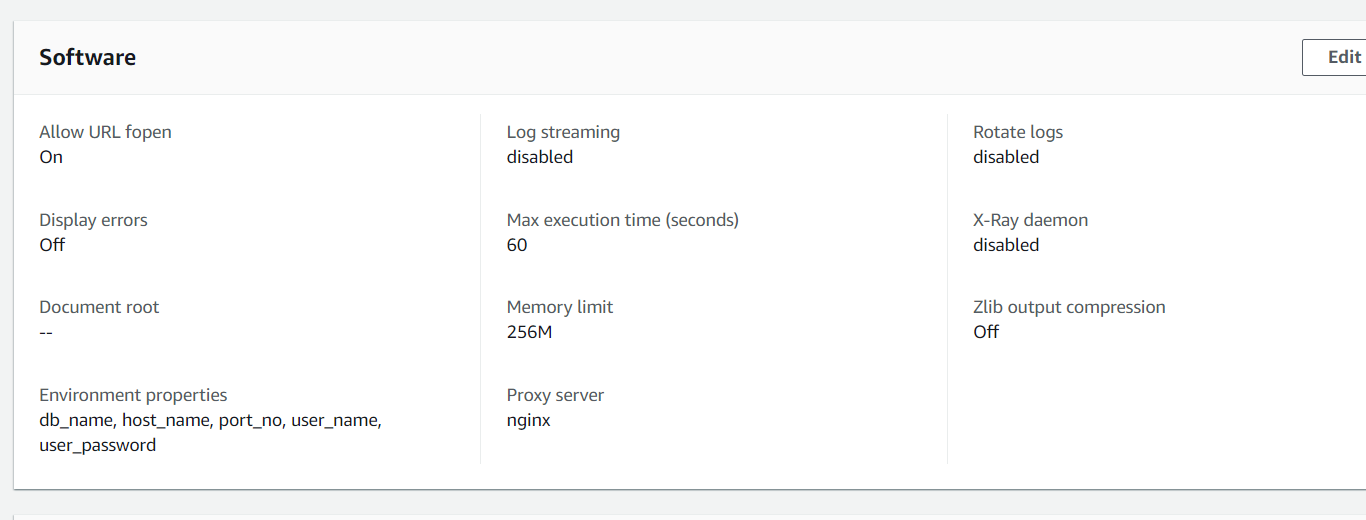


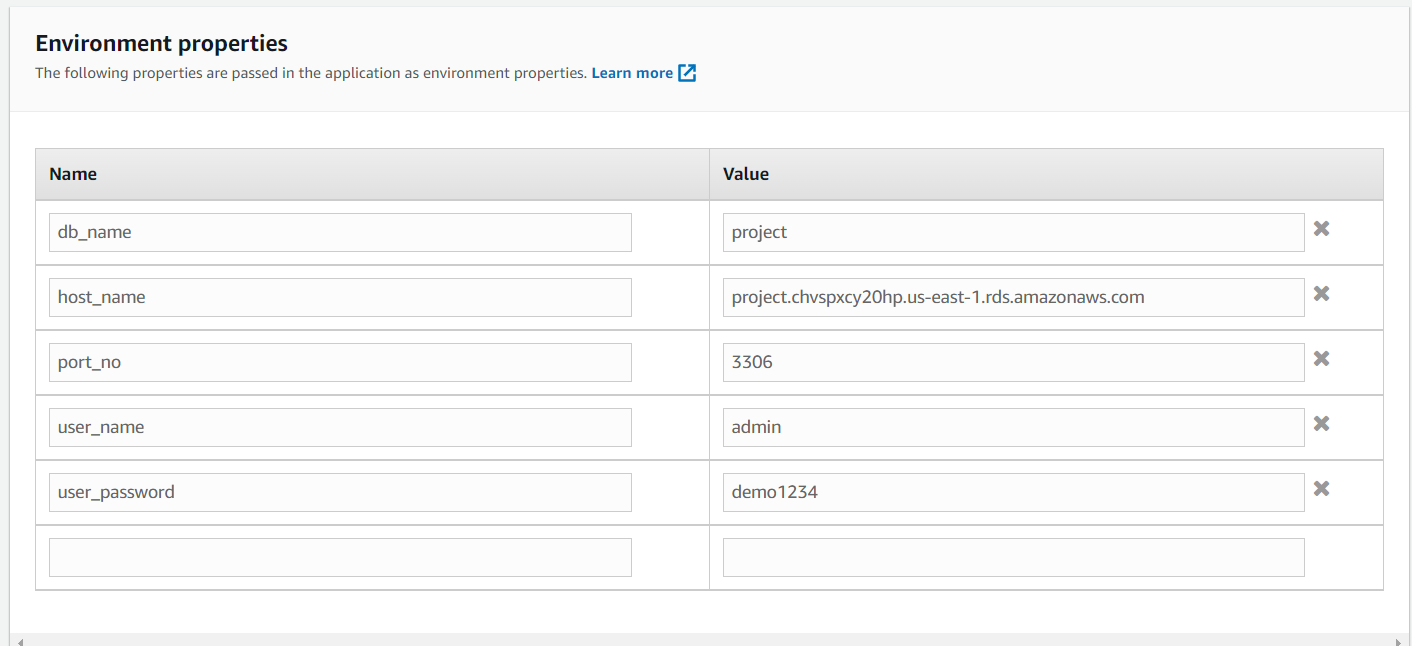
Sample application:

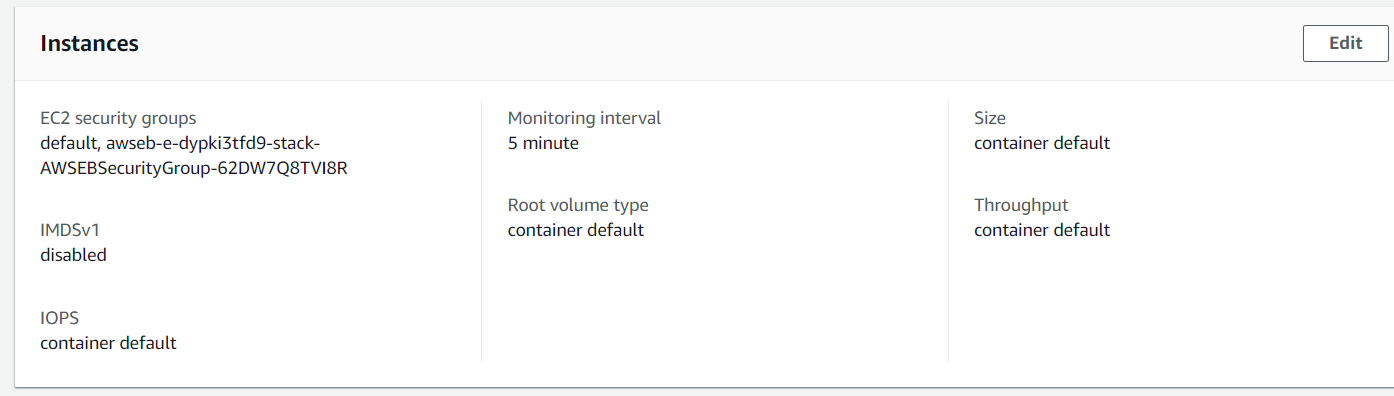


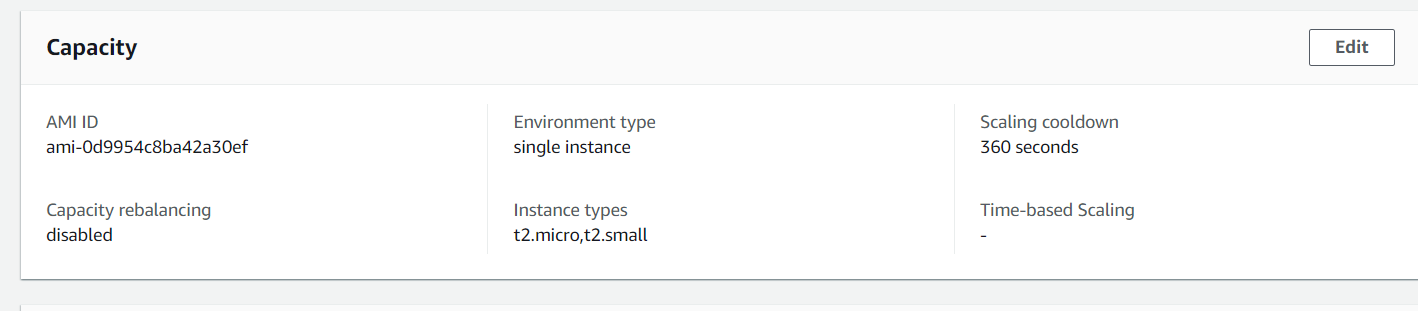
Instances security groups:

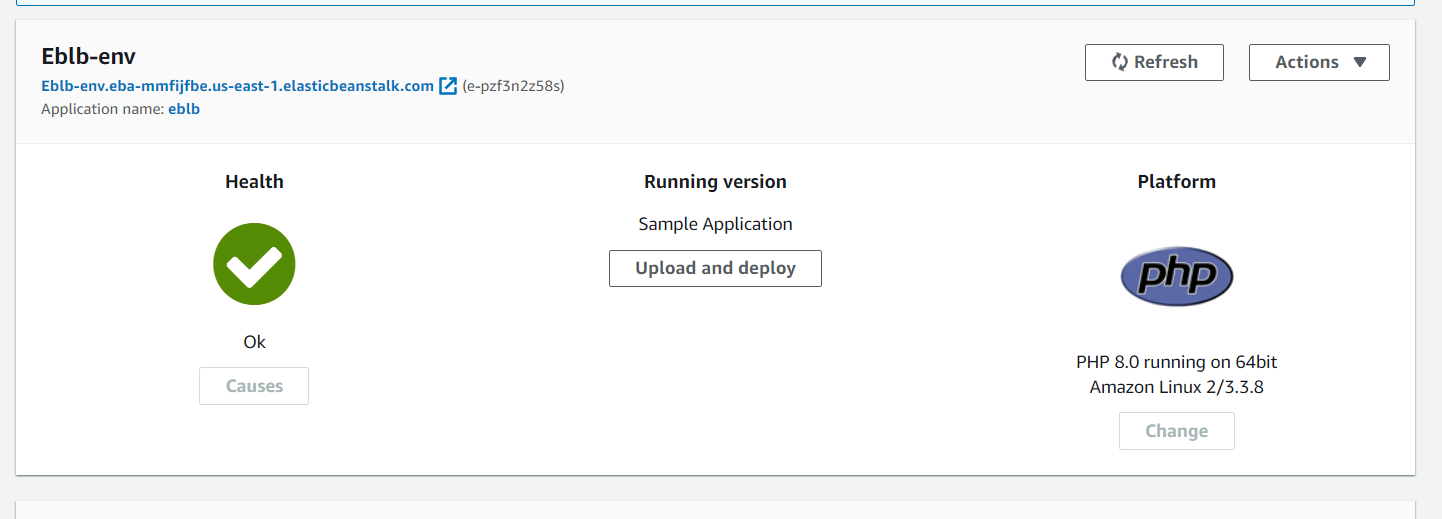




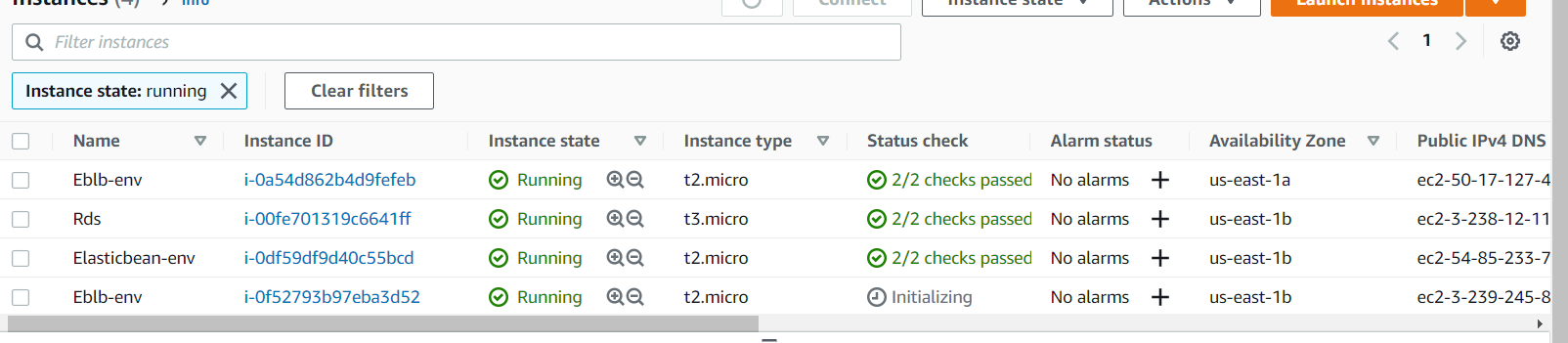


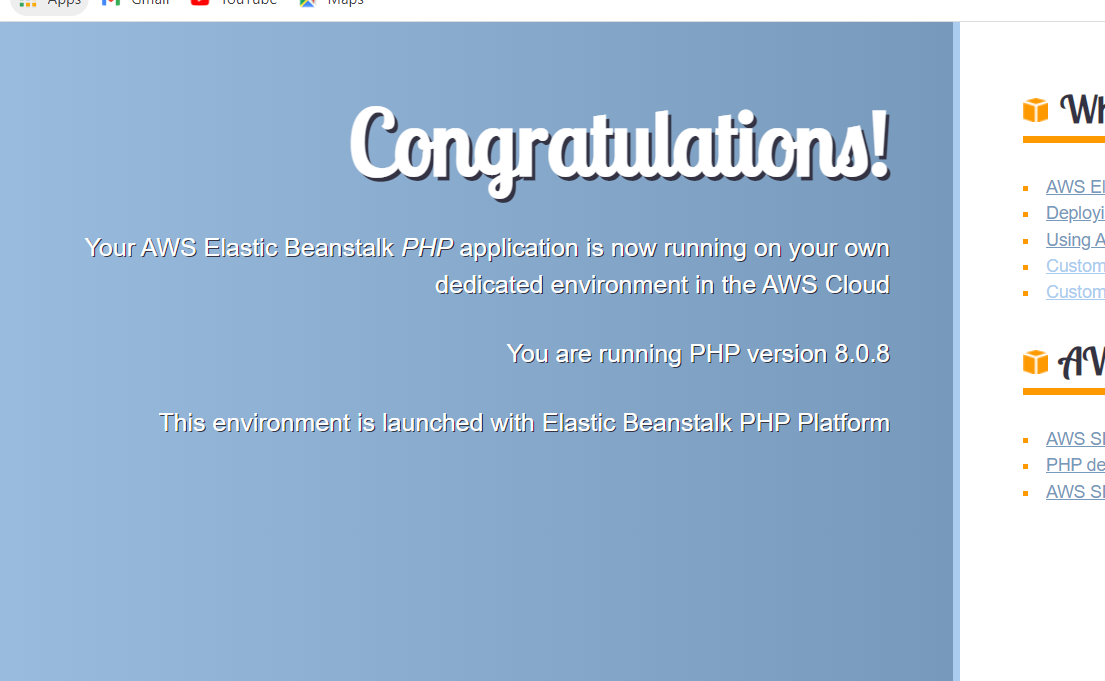






Autoscaling:





Project 3 – Publishing Amazon SNS Messages Privately Project: Publishing Amazon SNS messages privately Industry: Healthcare

Problem Statement: How to secure patient records online and send it privately to the intended party

Topics: In this project, you will be working on a hospital project to send reports online and develop a platform so the patients can access the reports via mobile and push notifications. You will publish the report to an Amazon SNS which keeping it secure and private. Your message will be hosted on an EC2 instance within your Amazon VPC. By publishing the messages privately, you can improve the message delivery and receipt through Amazon SNS.

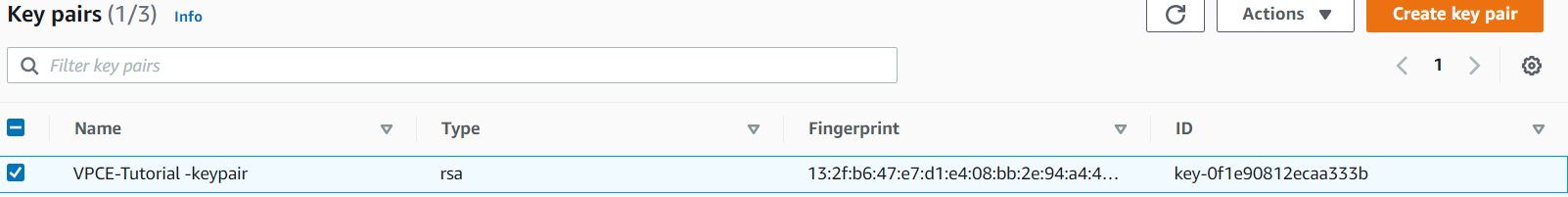
Highlights:

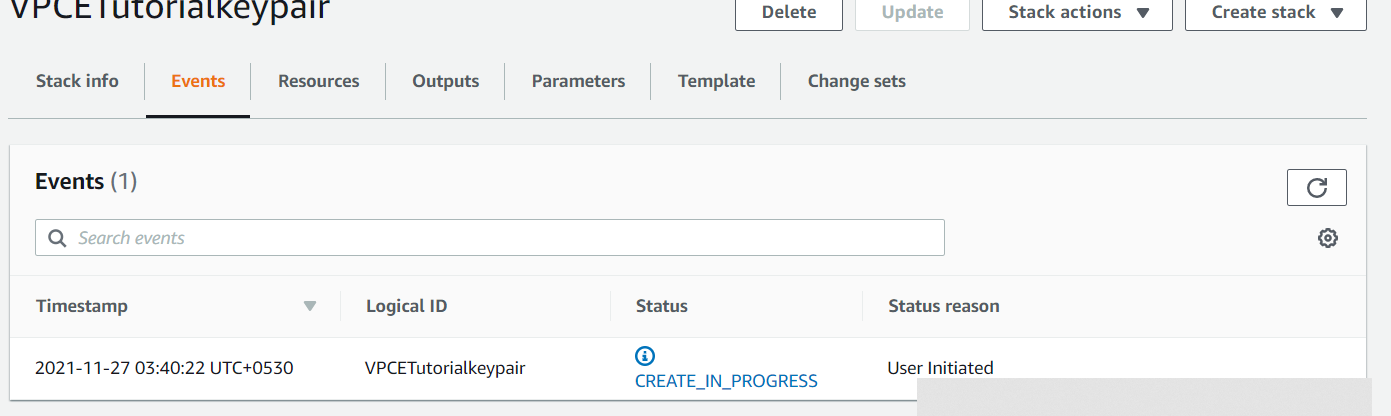
• AWS CloudFormation to create a VPC

• Connect VPC with AWS SNS

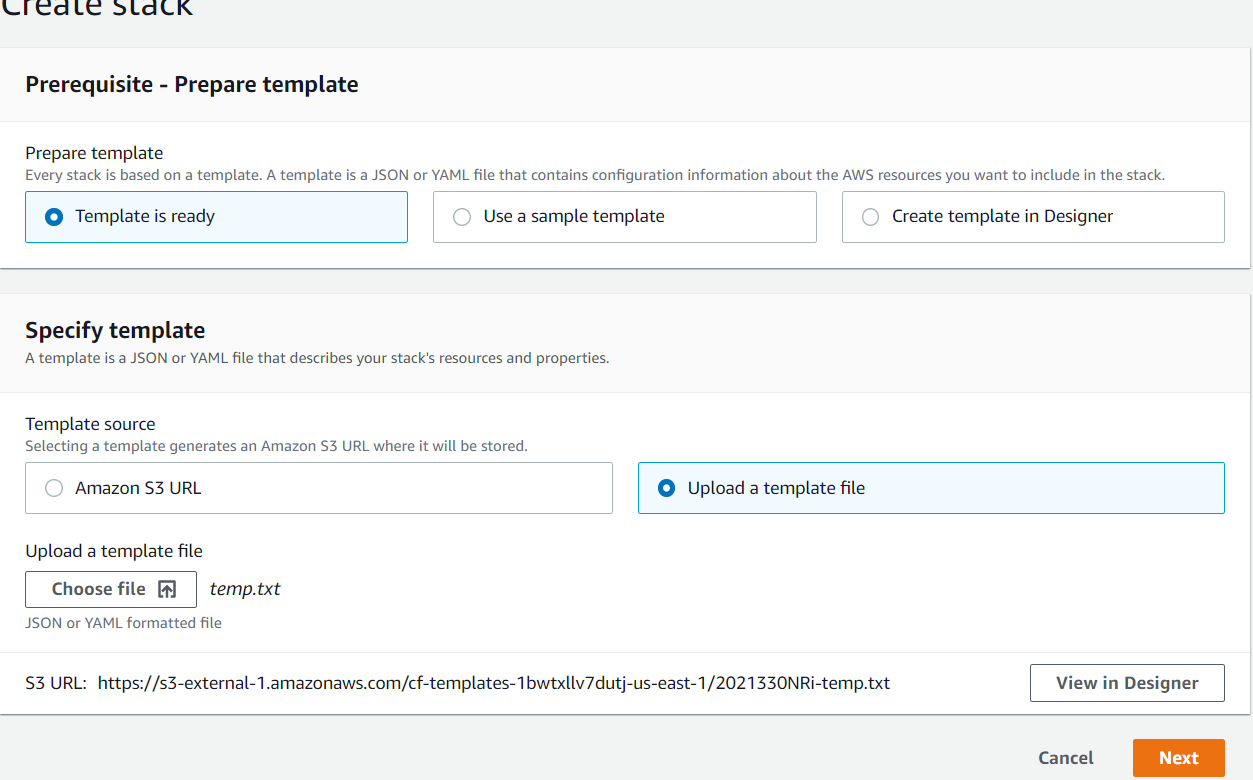
• Publish message privately with SNS

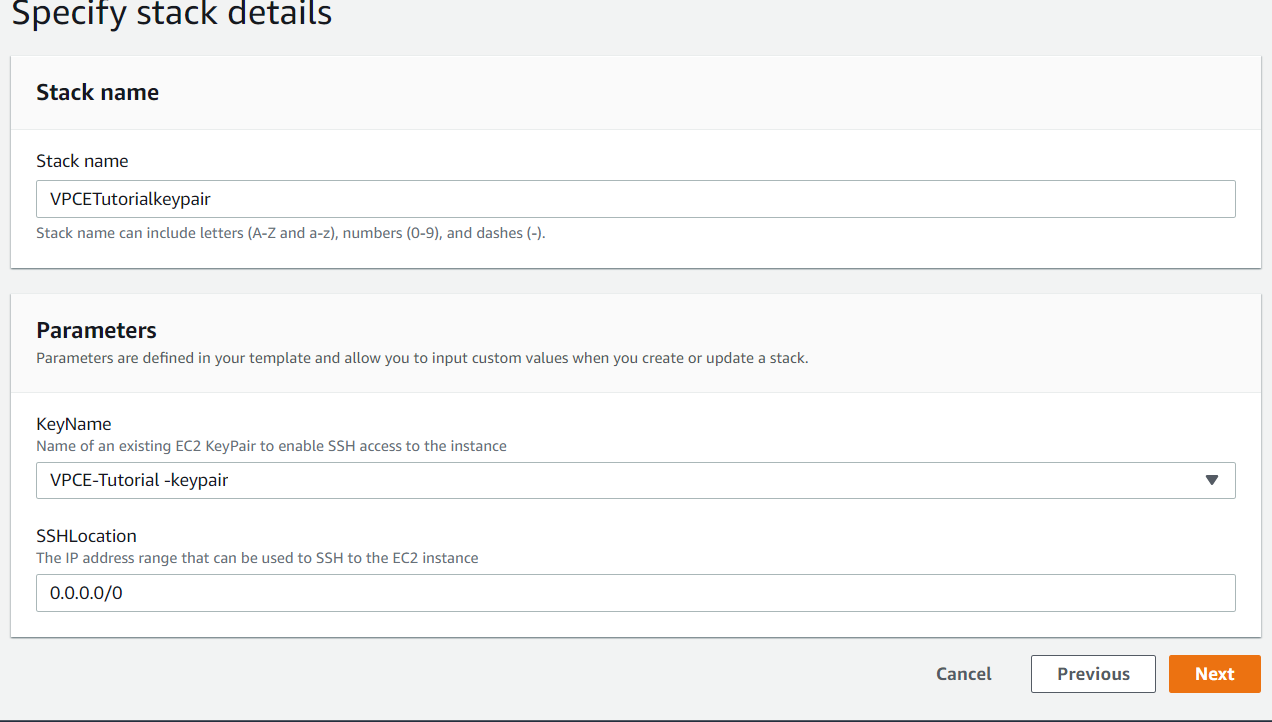
1.create an amazon keypair:

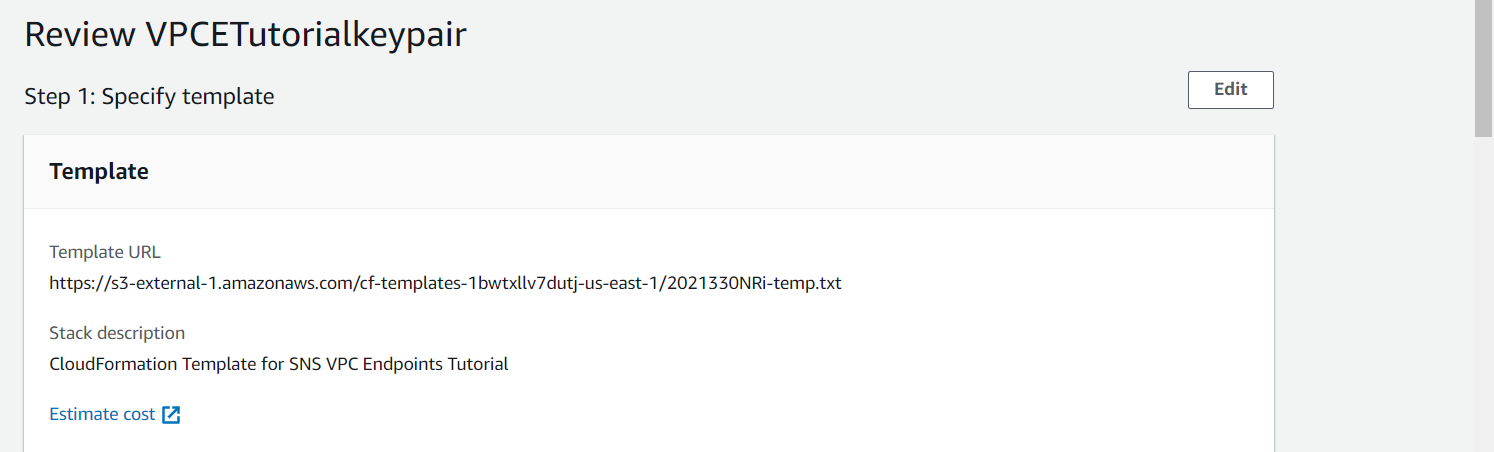


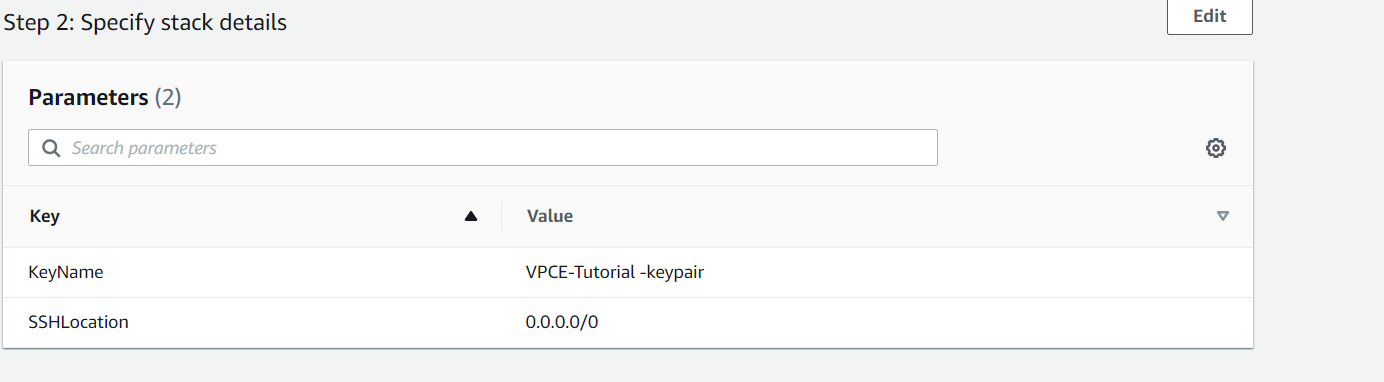


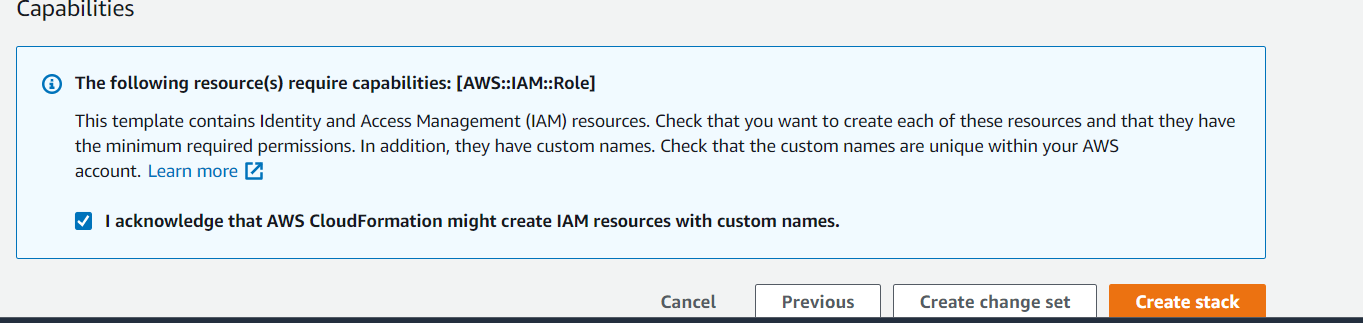
2.create an amazon resource:

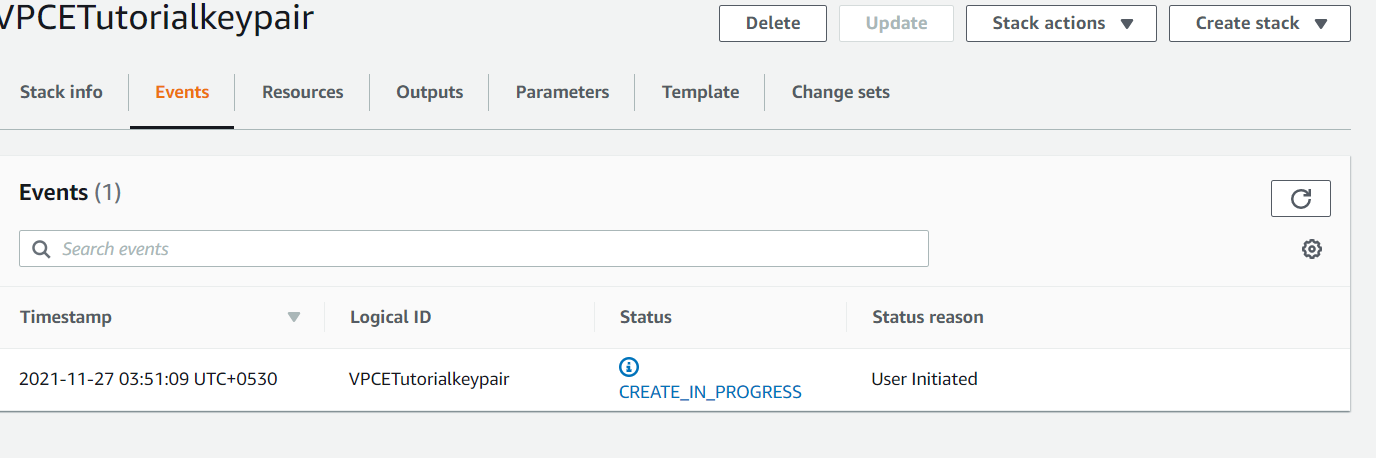


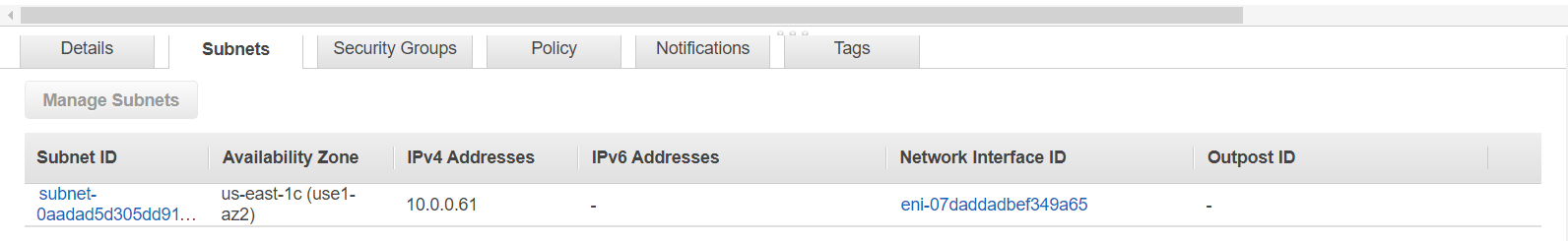


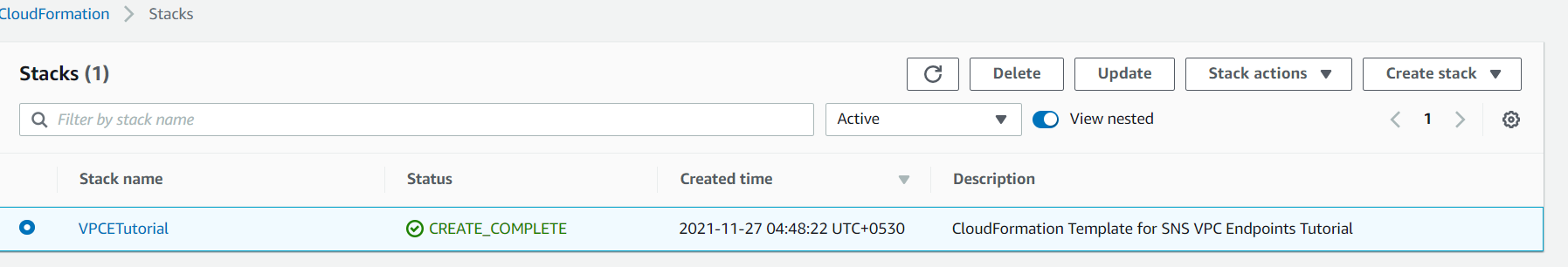




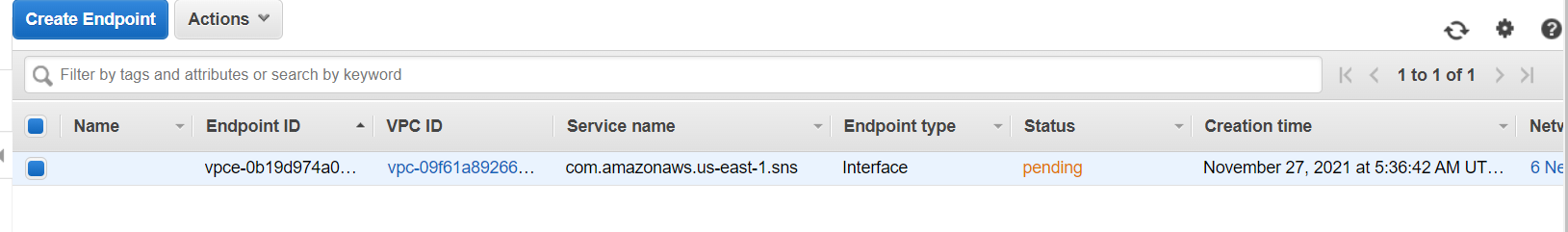




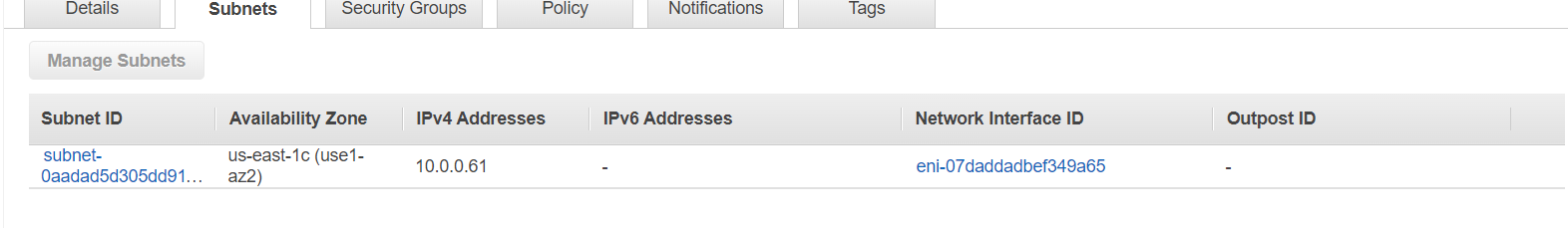




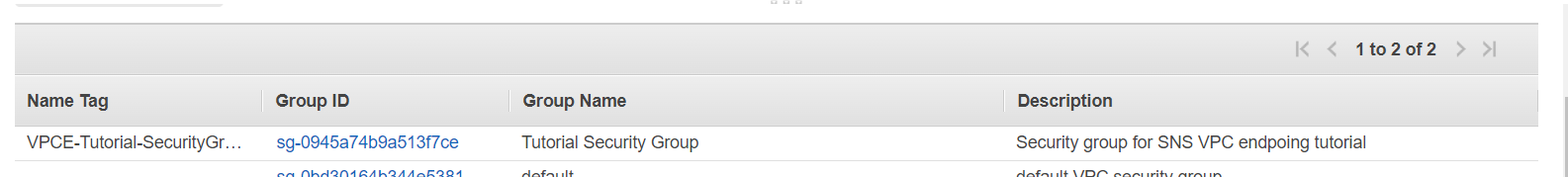
To create an endpoint:



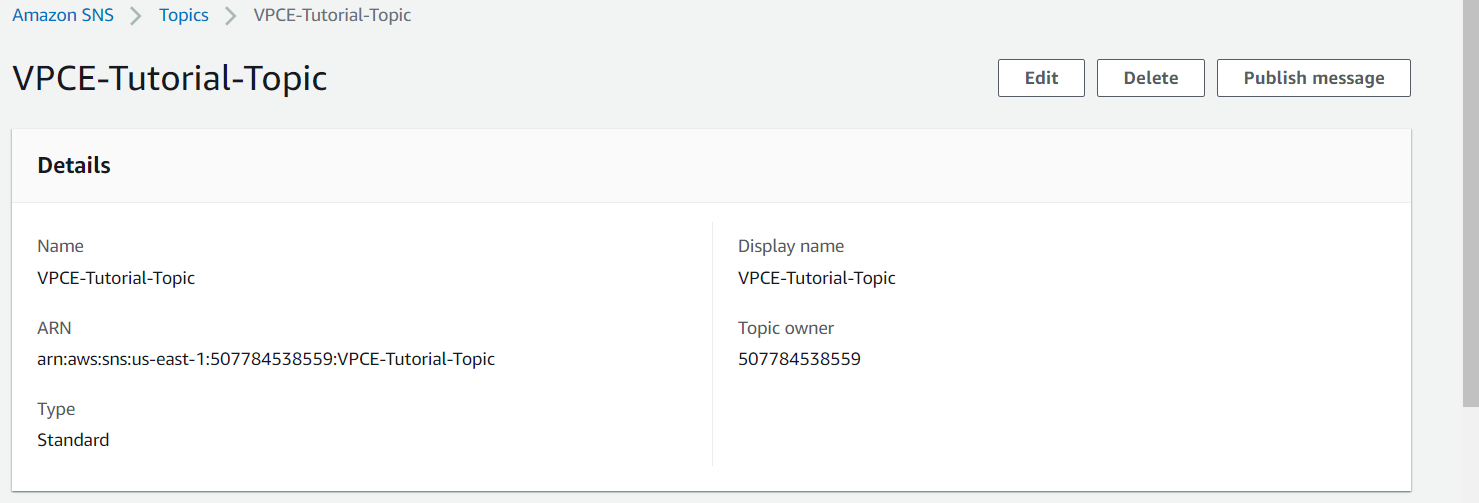
Subnets:

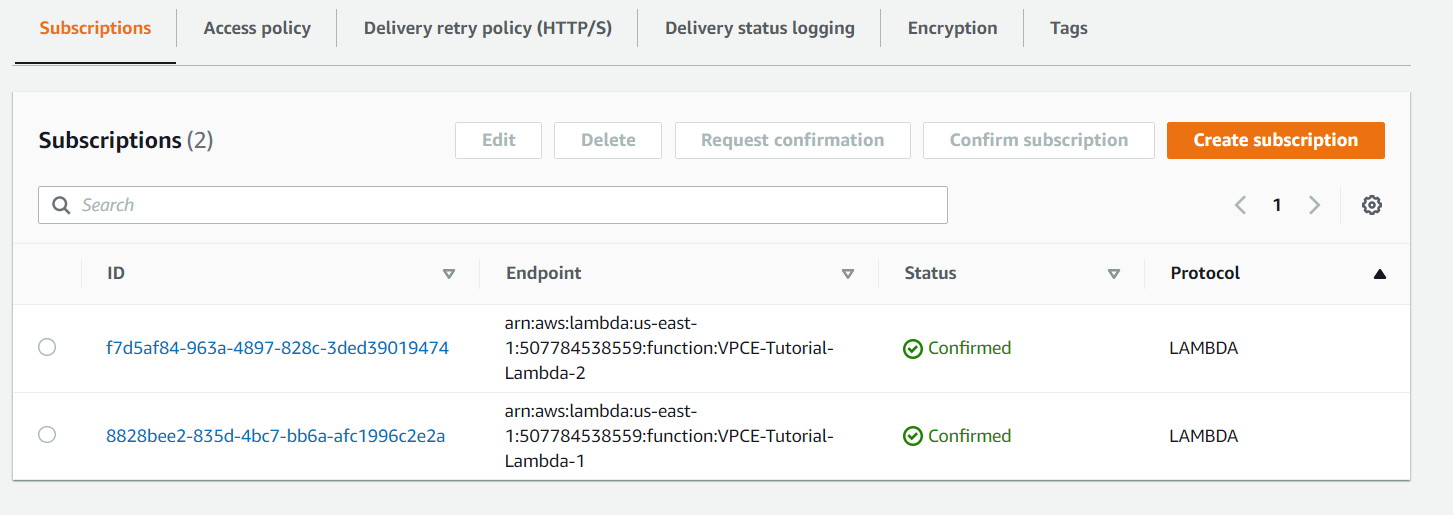


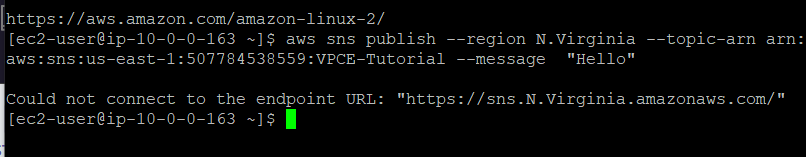
Security group:



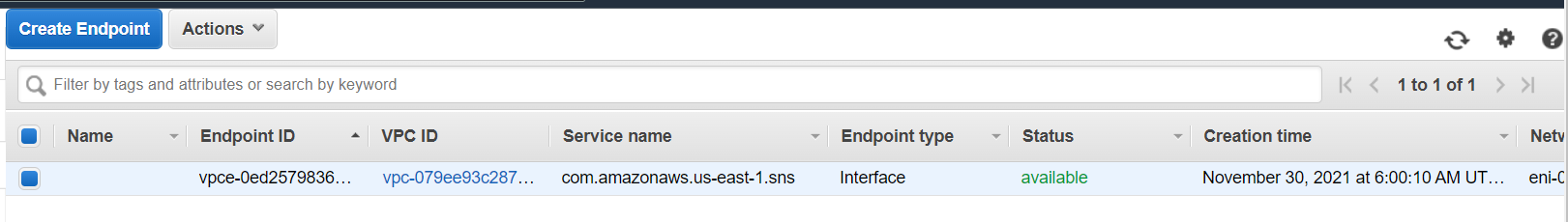
Confirm amazonEC2 lacks internet connection:

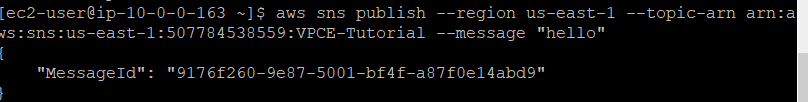


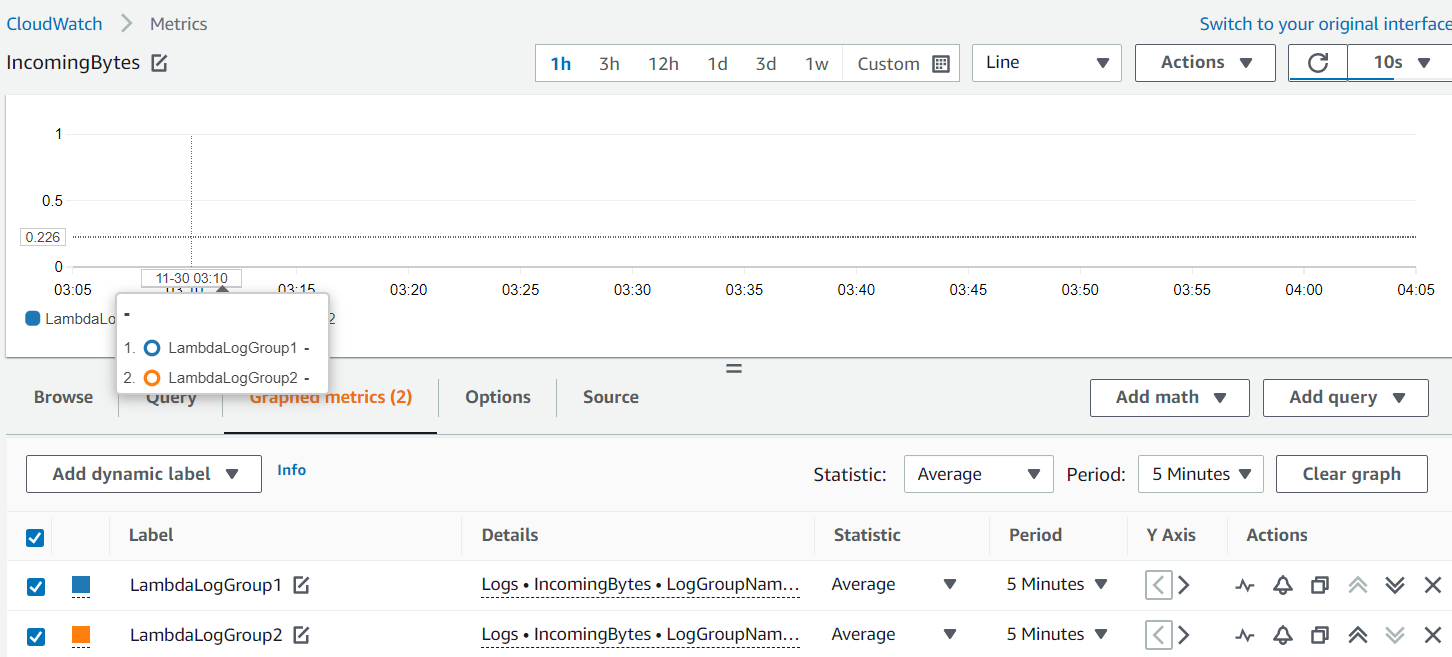




To verify the instance lacks connection to internet:







AWS Case Study:

AWS Case Study Problem Statement: A firm wants to launch their new application. To reduce cost it has been decided that the entire application will be hosted on AWS. It’s a photo and video storage application. This application will store movies and videos, and customers can also upload photos and their custom made videos using this application. To buy videos, movies and wallpapers stored in the application customers have to pay some amount using their credit card. Customer will have to provide their home address and some other personal information as well. Following are some of the requirements:

1. The most important requirement is that the application should be very much secure and none of the servers (front-end UI, back-end application, database) should be available over Internet.

2. Application should handle sudden spike in customer activity without any problem.

3. Customer information should be kept secure as much as possible.

4. It has been decided that 2 types of NoSQL databases would be used to store metadata information about videos and photos. 3 Relational databases would be used for transactional purposes.

5. Performance is vital for the success of the product. Make sure that users get low latent responses wherever possible.

6. Photos and videos should be stored securely at rest.

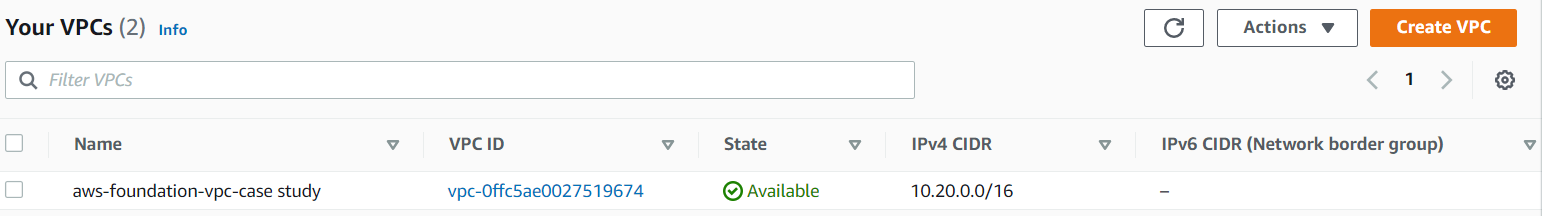
7. Access to AWS resources should be authorized as per business need, e.g. application developers should not have access to database servers. DBAs should not have any access to application servers.

8. Whenever customer uploads a photo/video he or she should get an email notification. 9. Photos/videos older than 2 years should be archived to save cost.

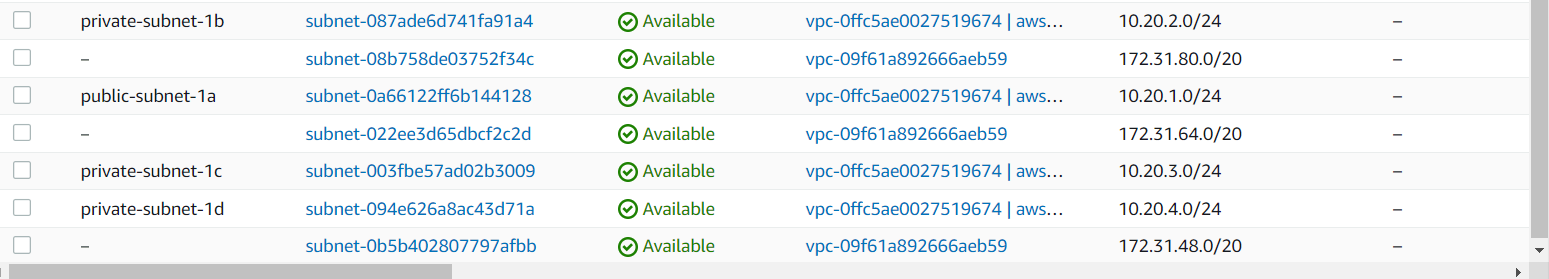
10. Application should be highly available. It should not go down in the event of any data center failure. Disaster Recovery has to be set up to mitigate regional failures as well.

11. Transactions should be as secure as possible. 12. All the SysOps out there configure and use AWS CLI to provision all services.

**Vpc:**

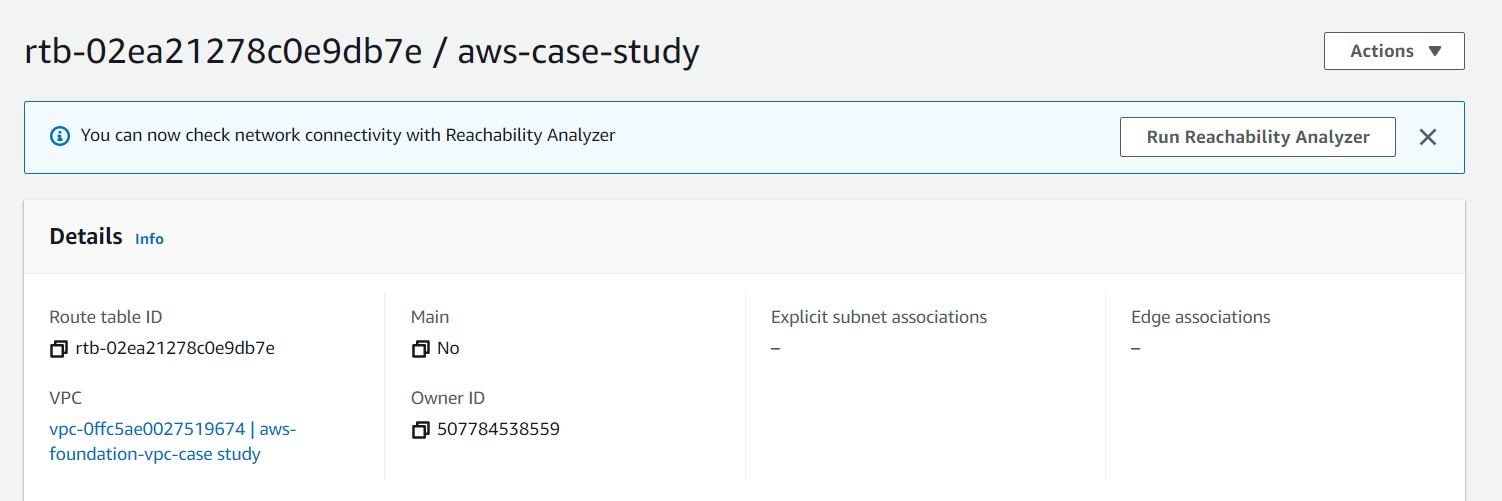
****

**Subnets:**

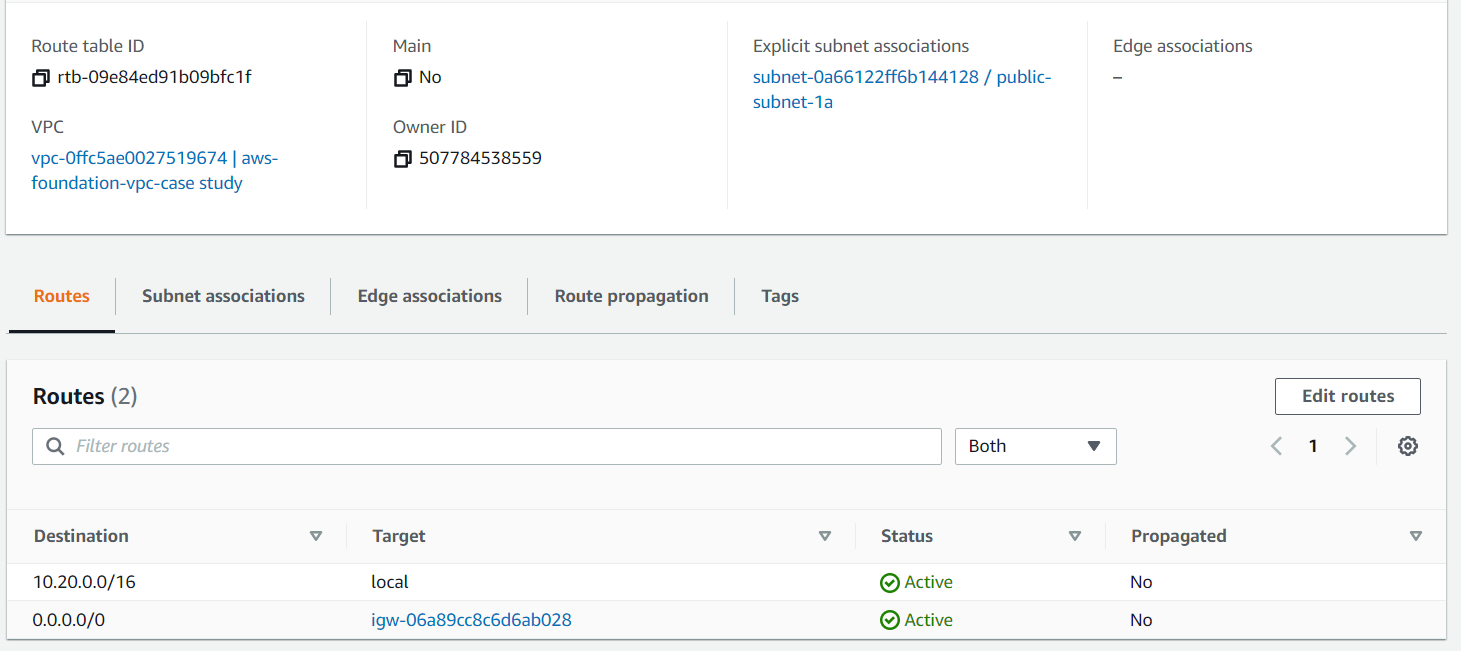
****



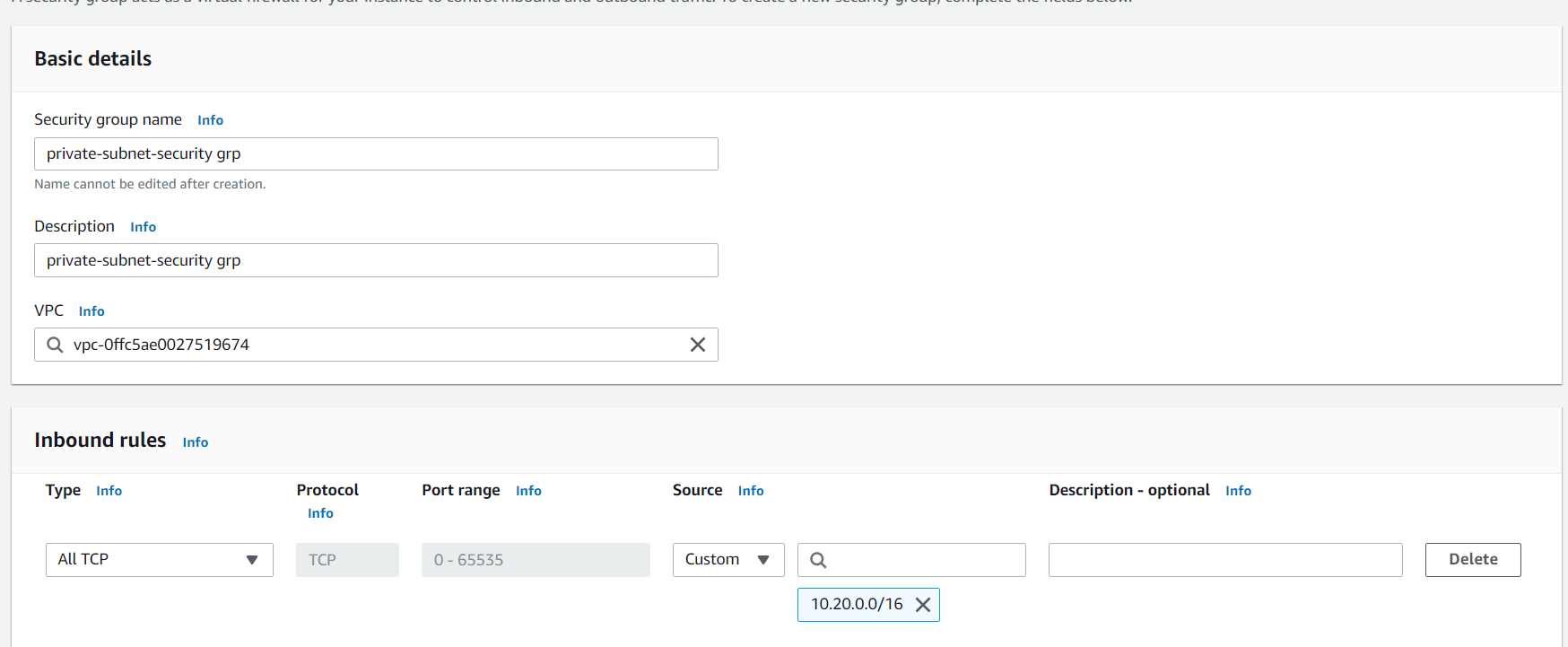
Route table

Public route table:

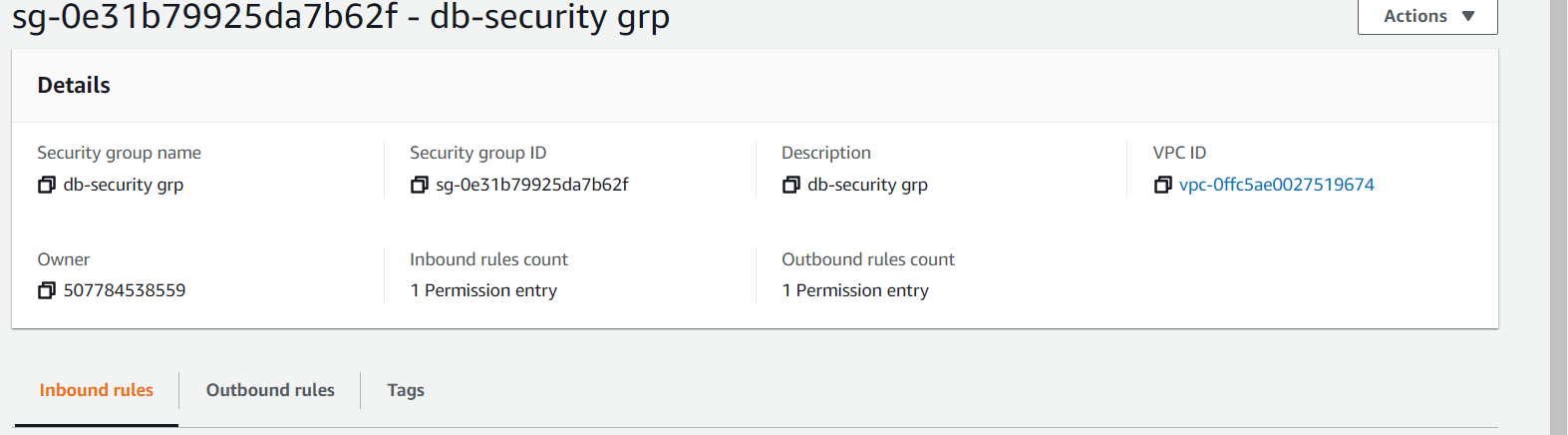
Private route table:



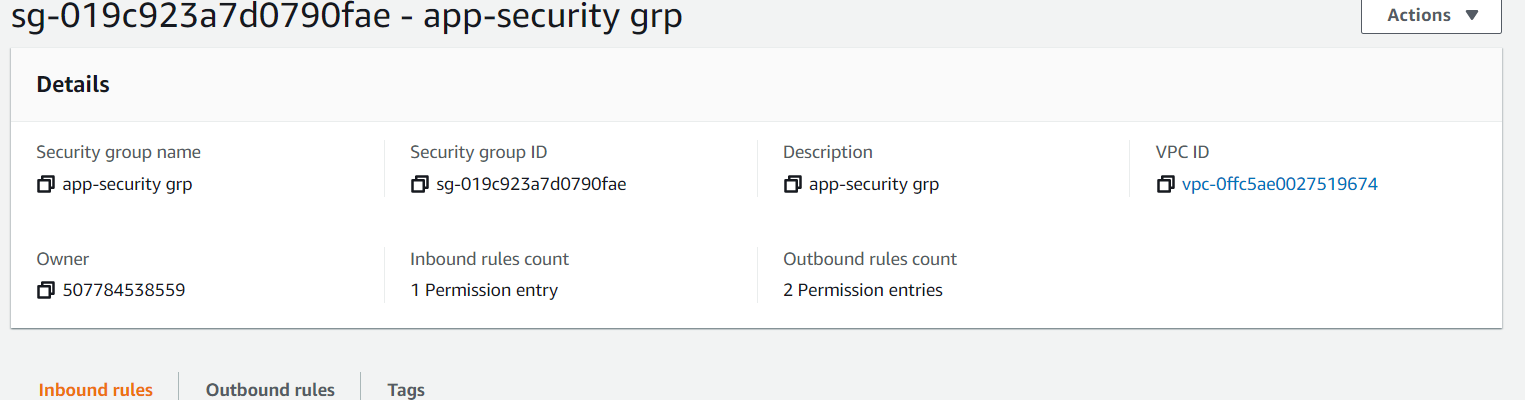
Setting up firewall:



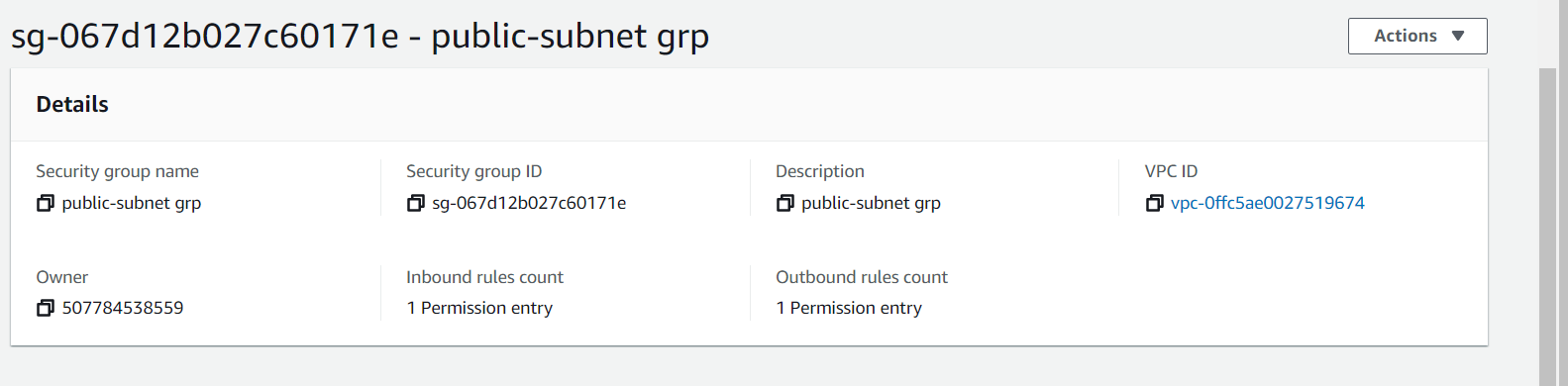
Db security grp



App security grp

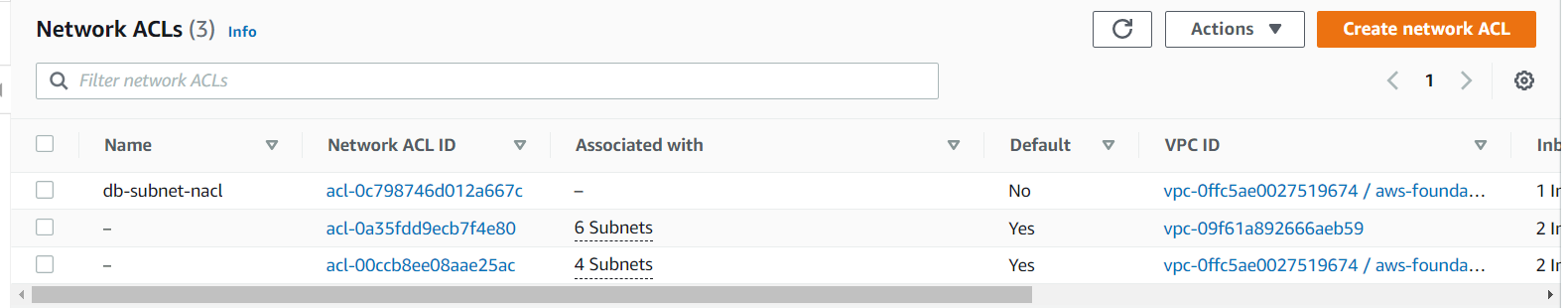


Public subnet grp:

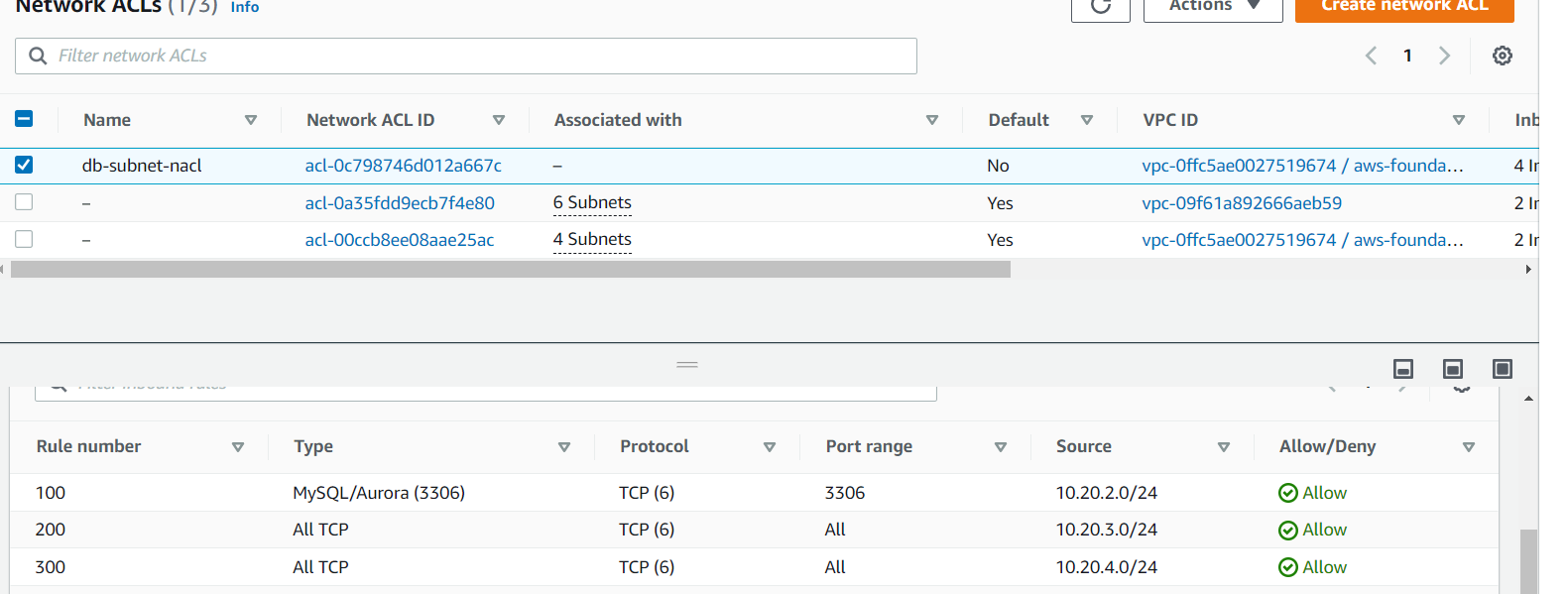


NACL:

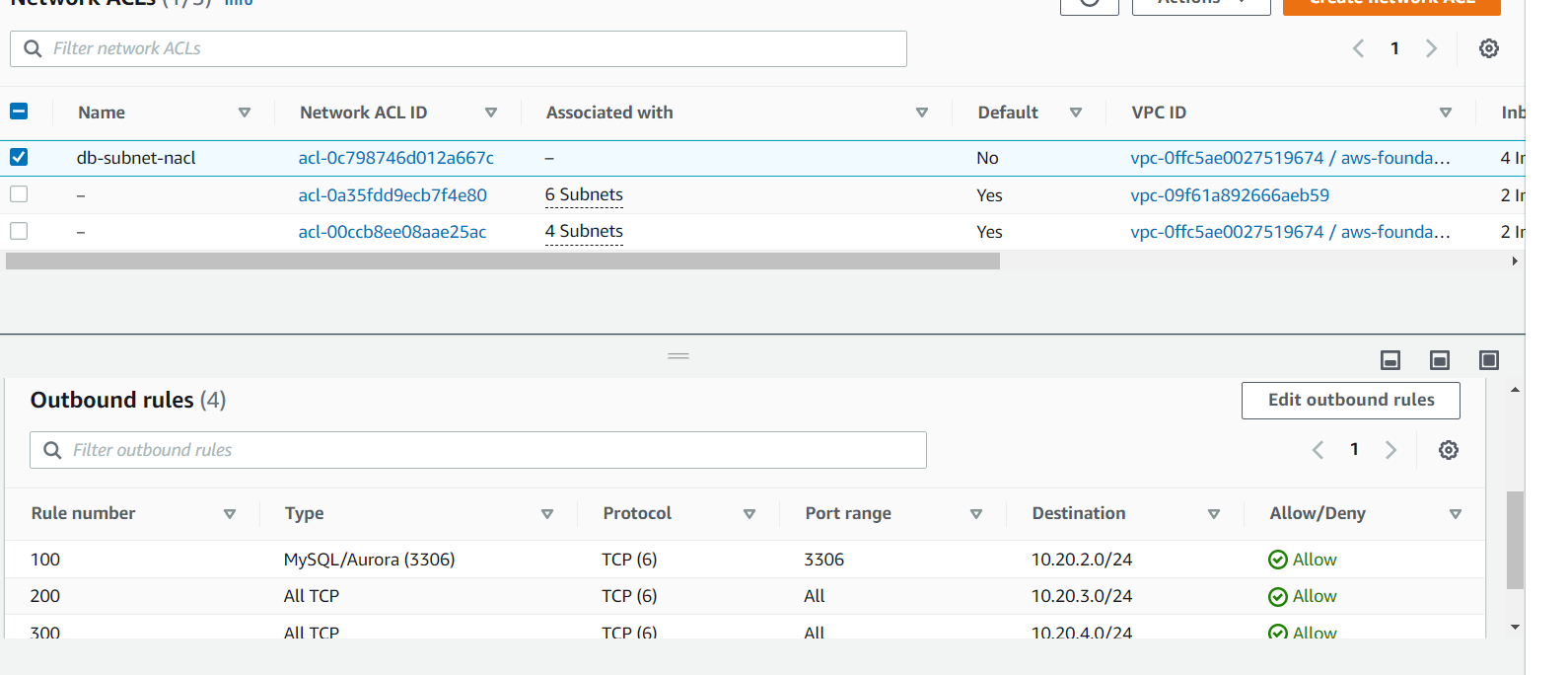
DB subnet:



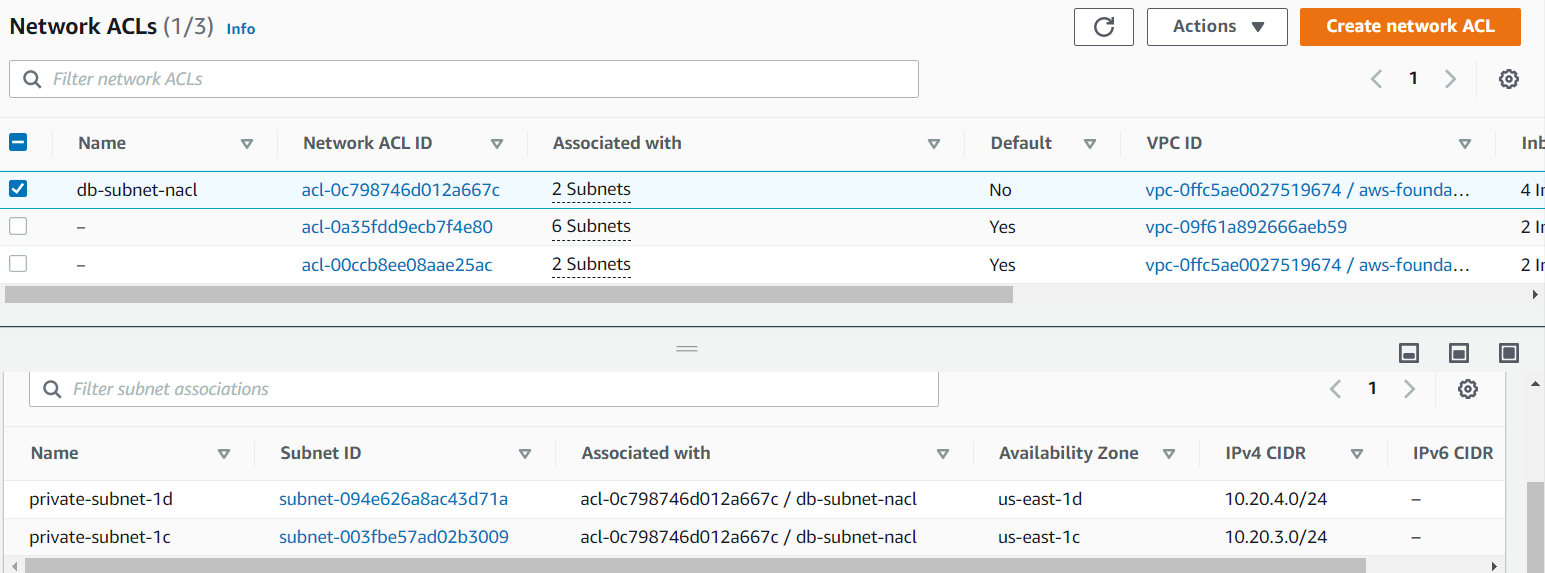
Inbound rules:



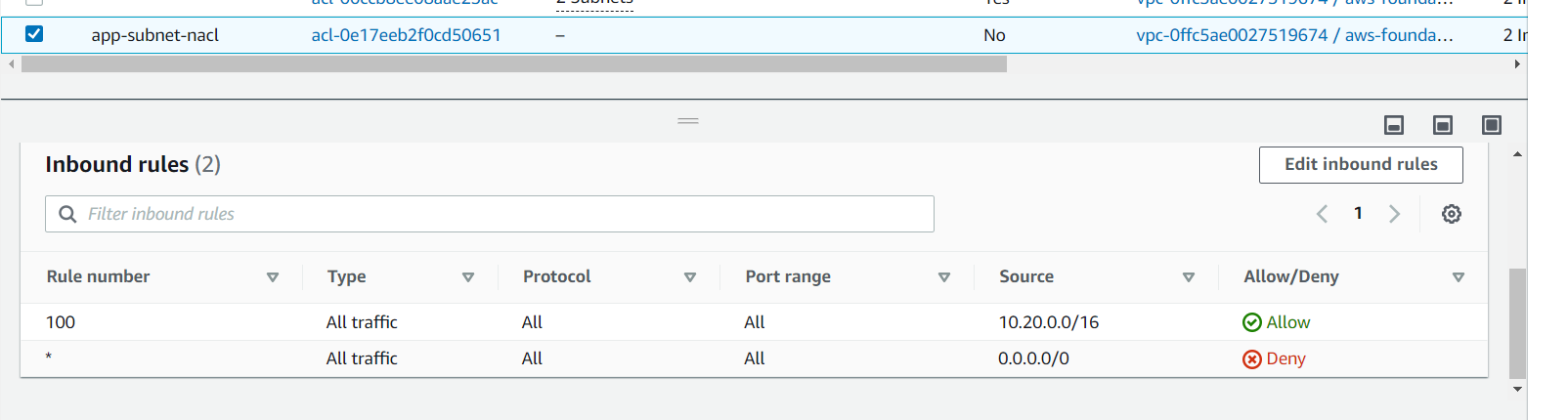
Outbound rules:

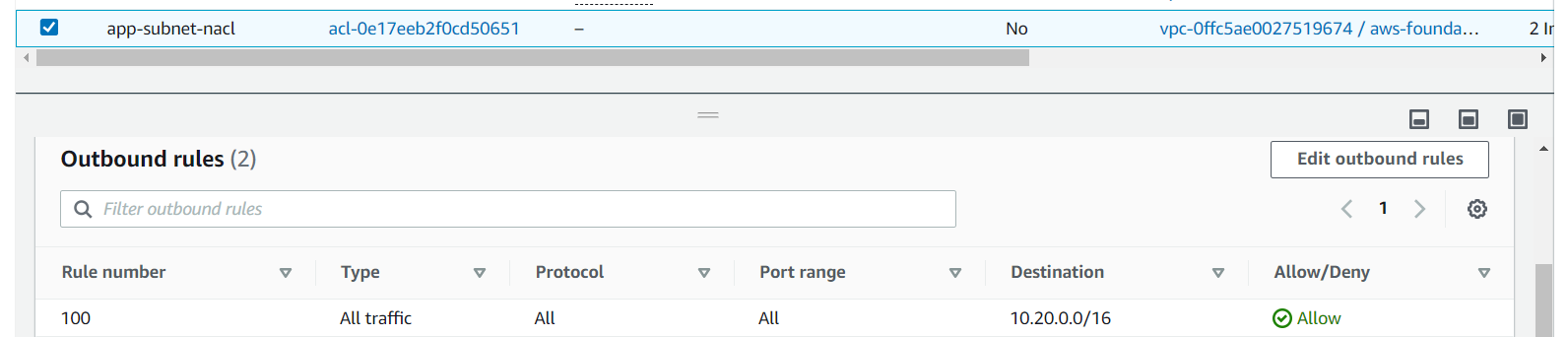


Subnet association:

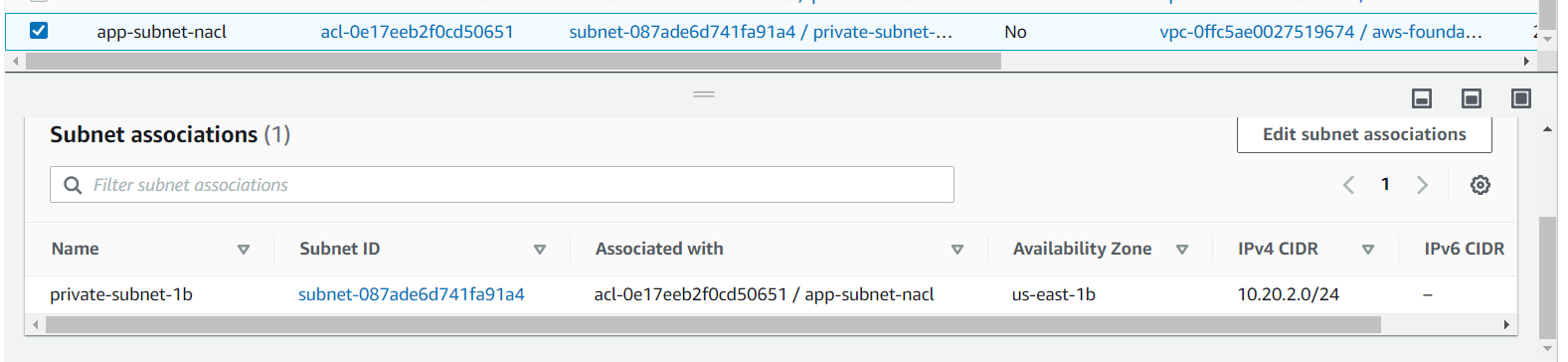


App subnet:

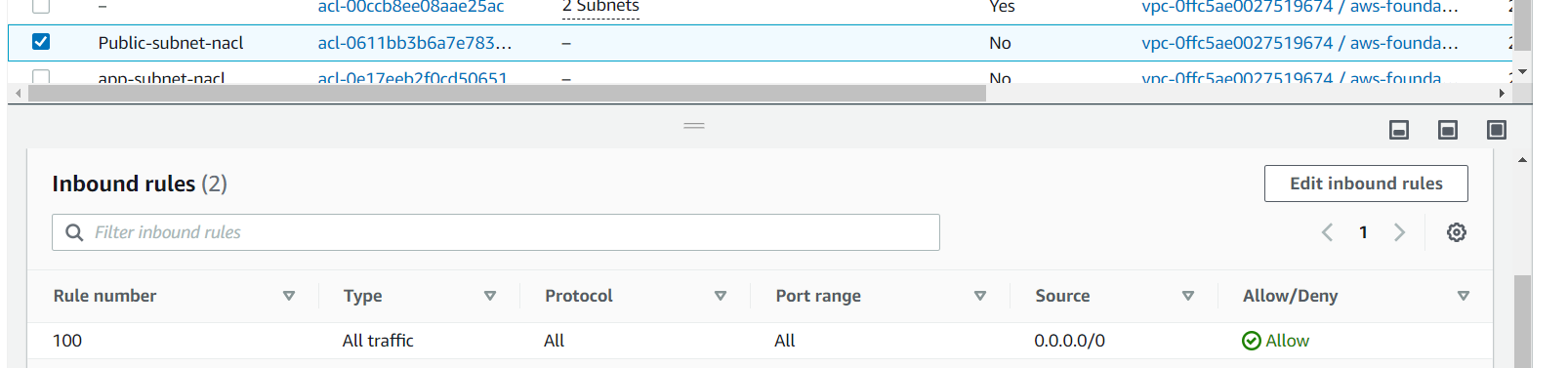


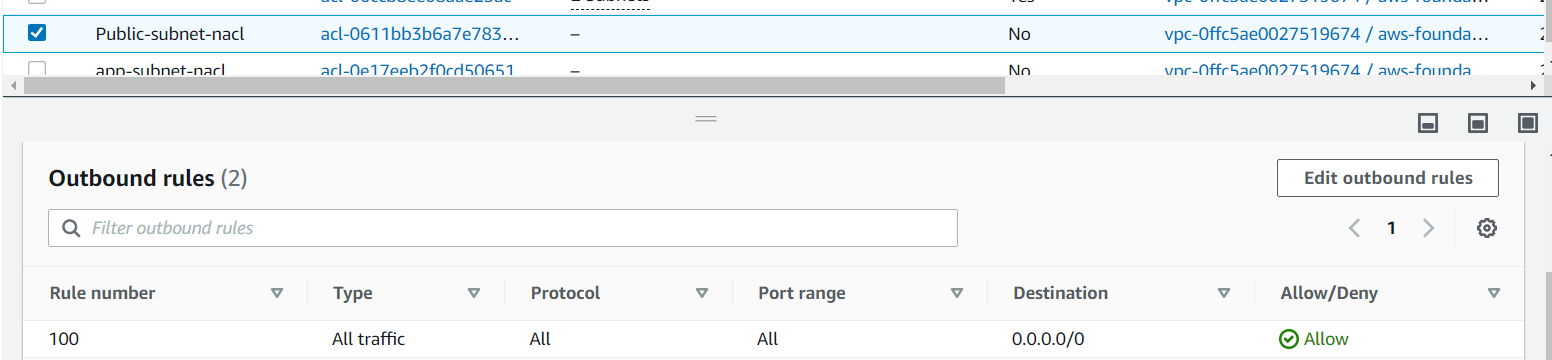


Subnet Association:

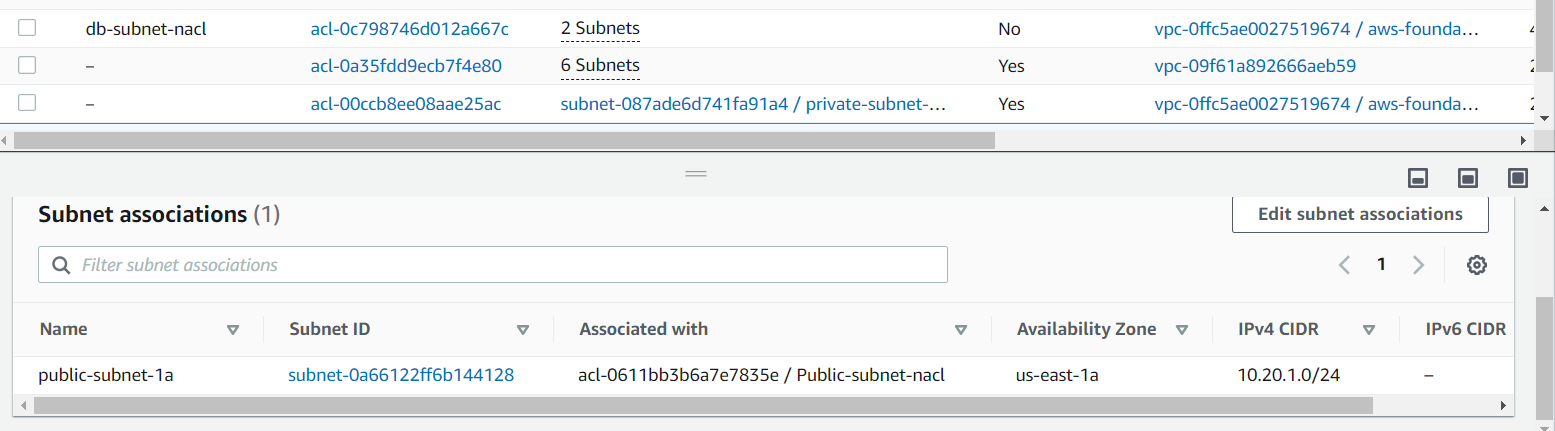


Public subnet:

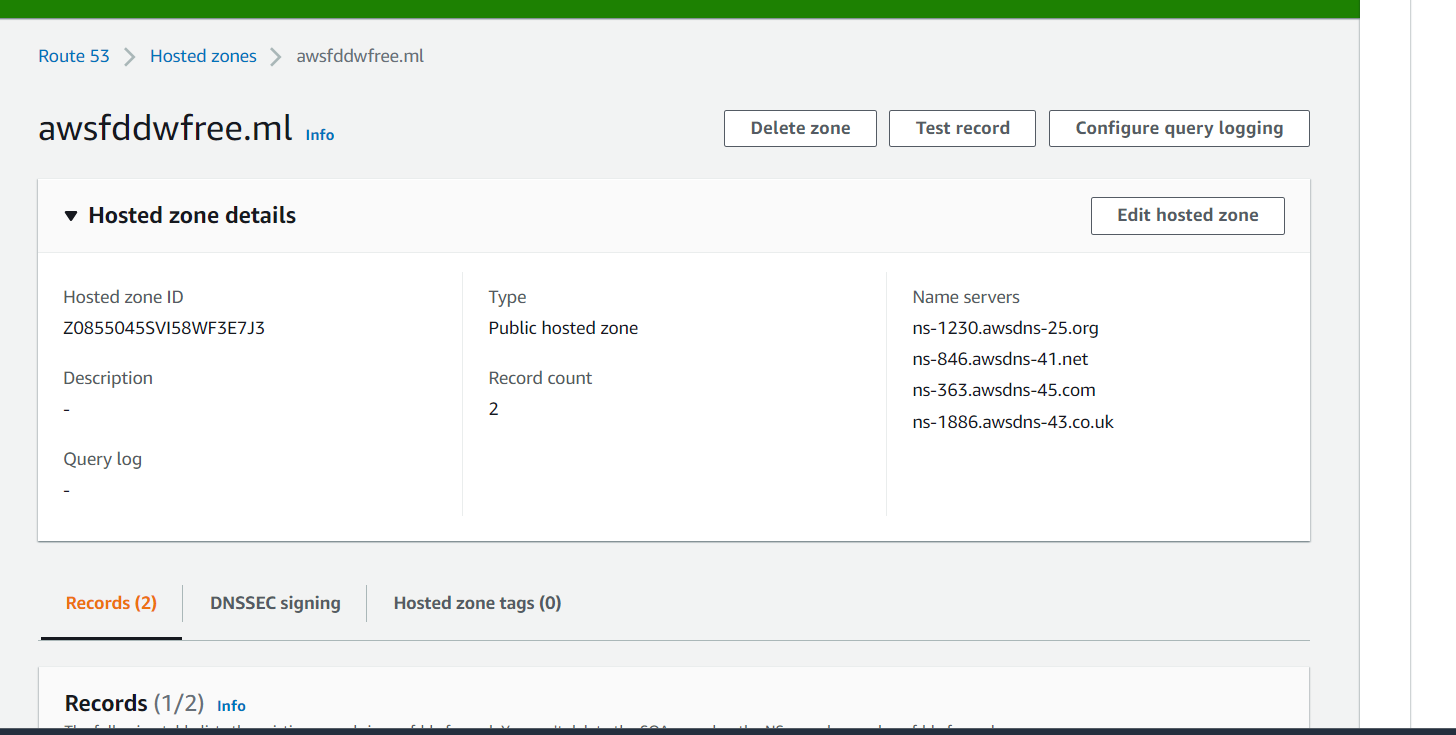




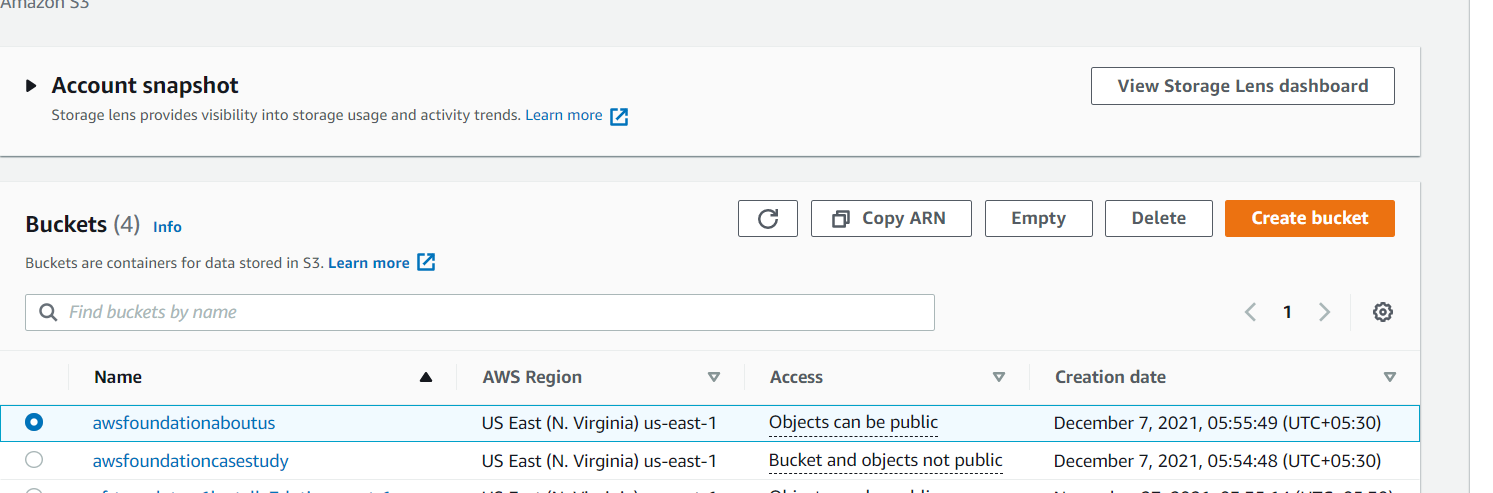
Subnet association:



Route 53: Domain Name setup:

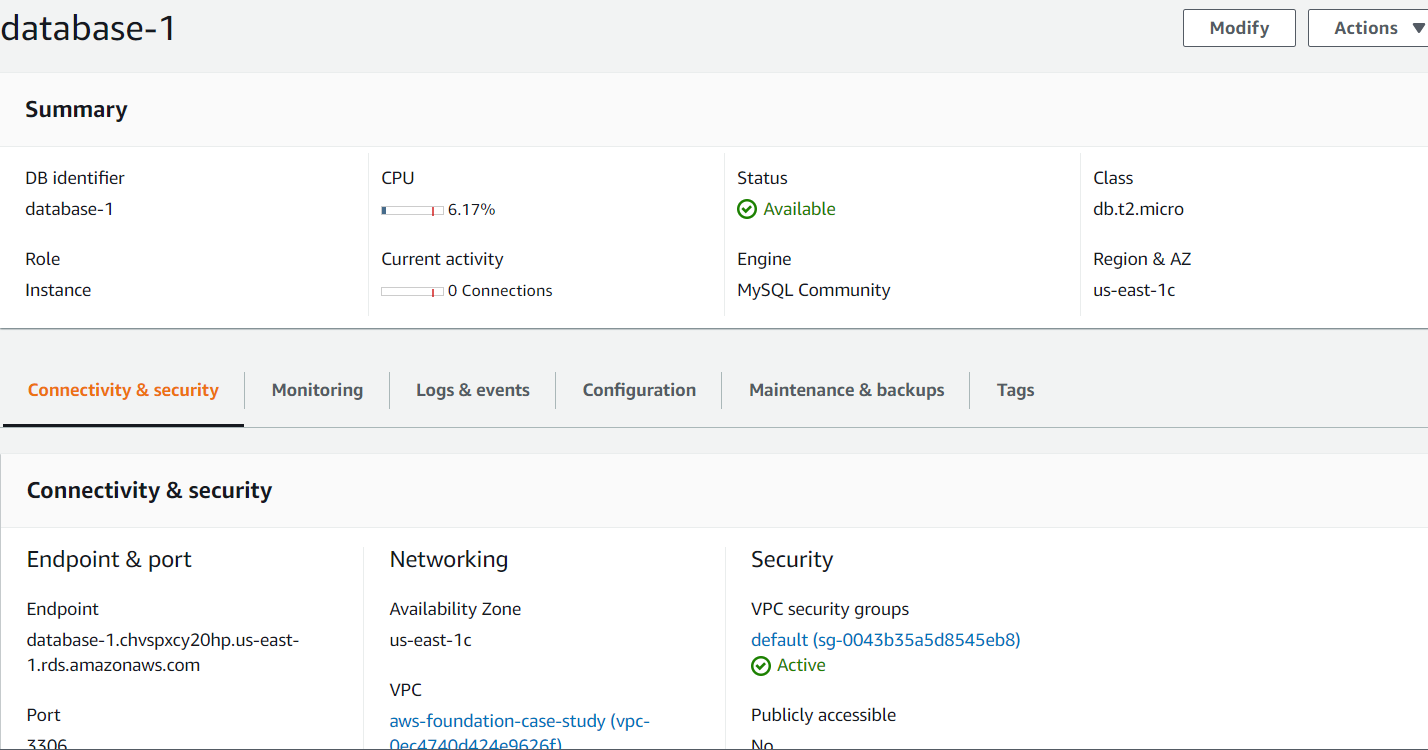


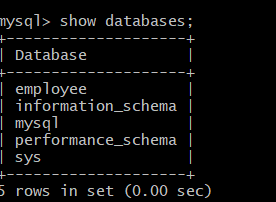
Creation of S3 bucket:

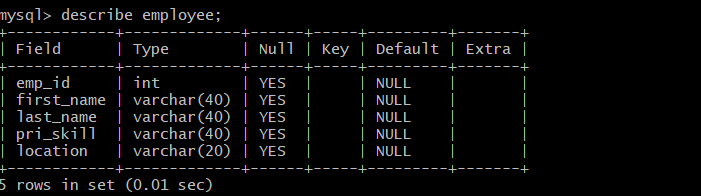




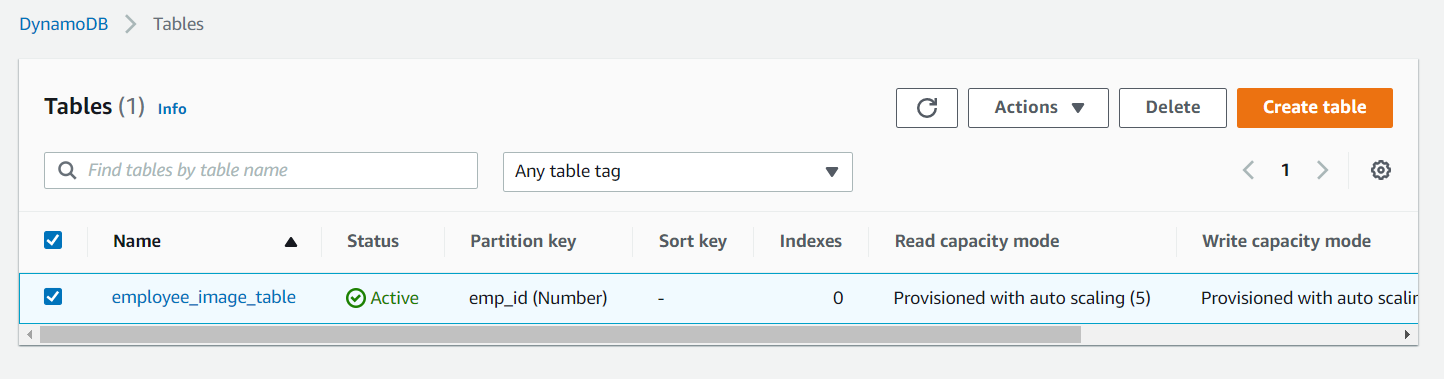
Creation of database:

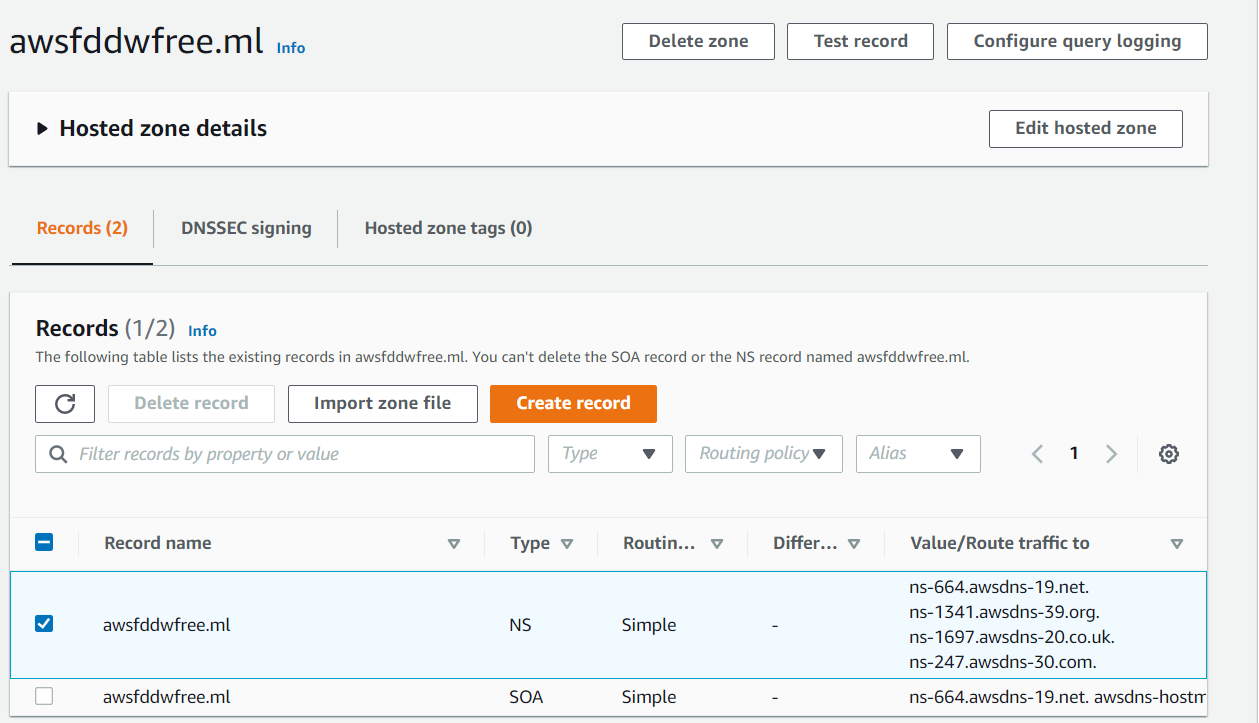




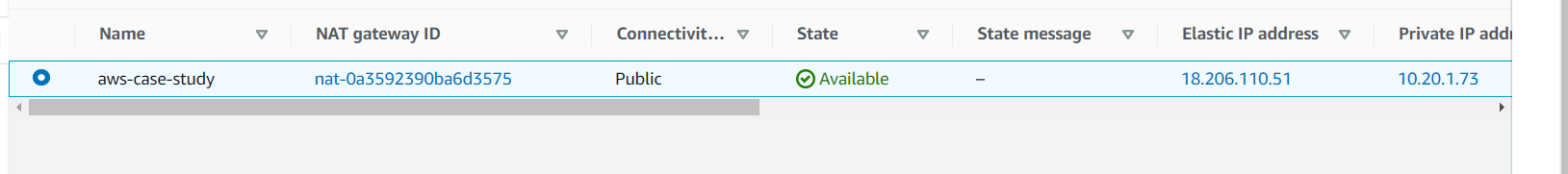


Creation of DynamoDB :

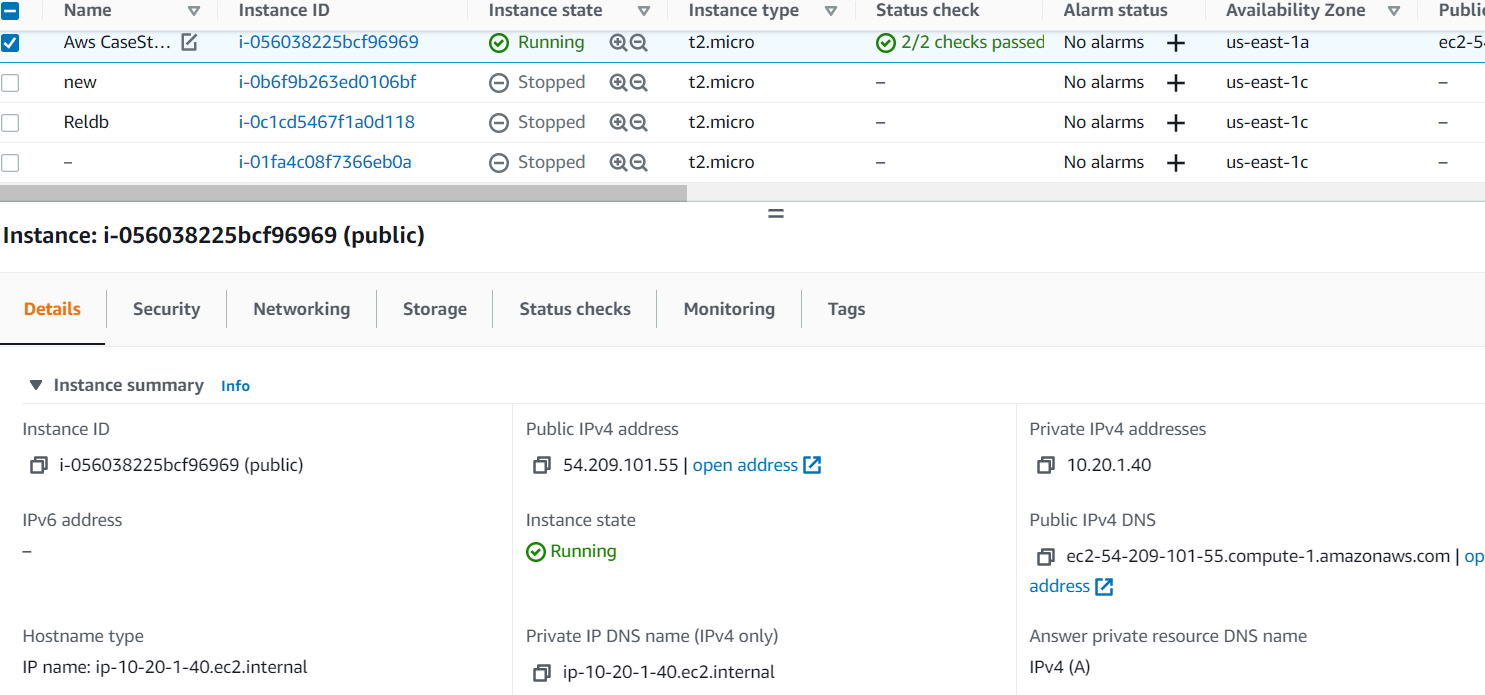




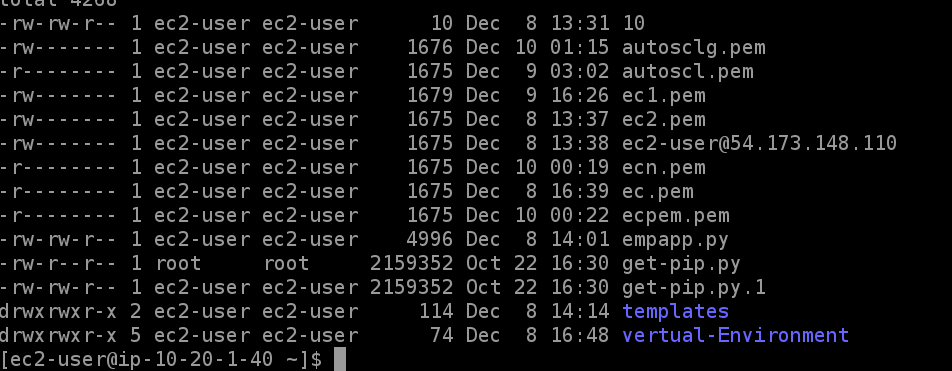
Creation of NAT gateway:



Ec2 Instance:

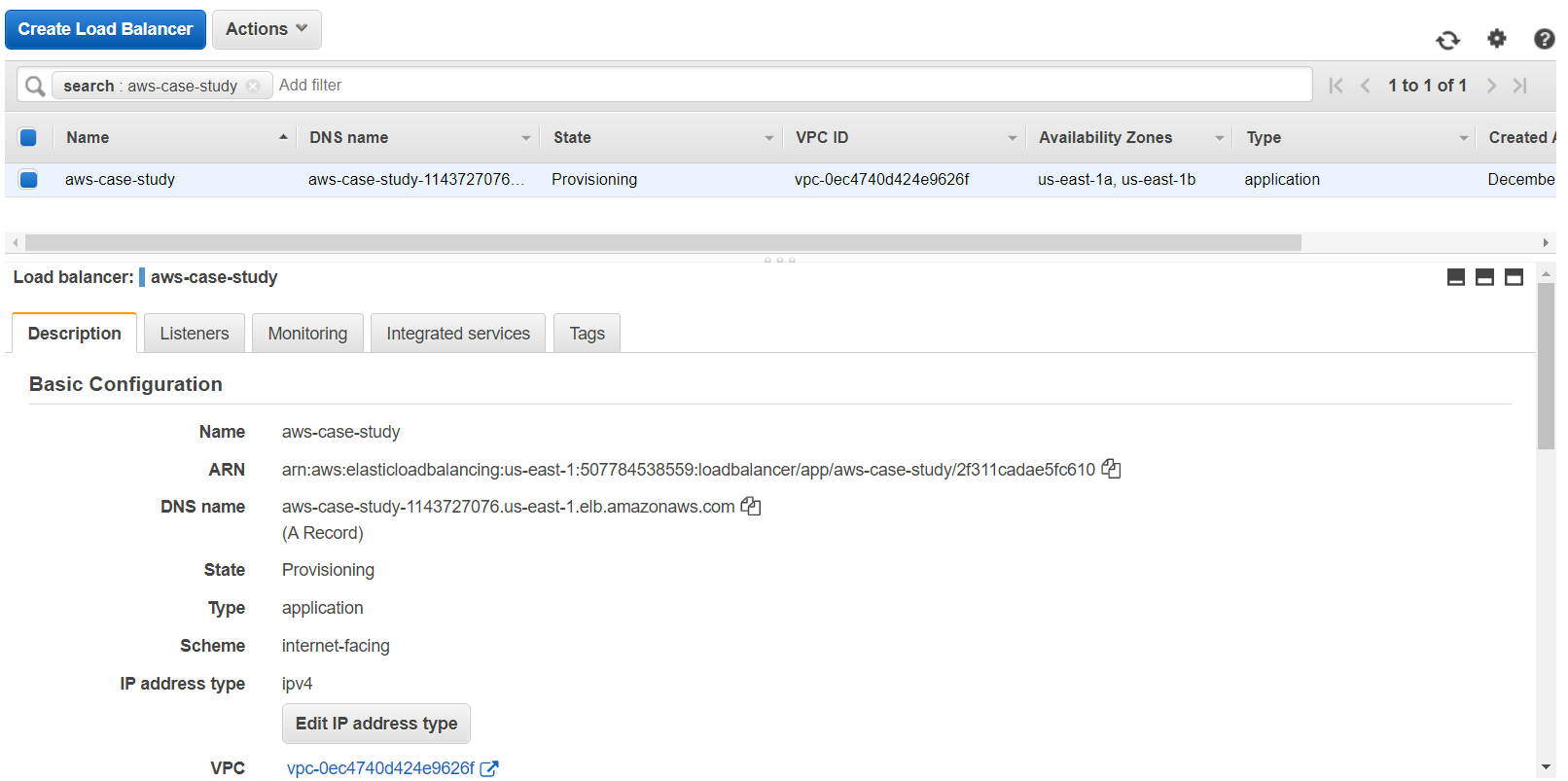


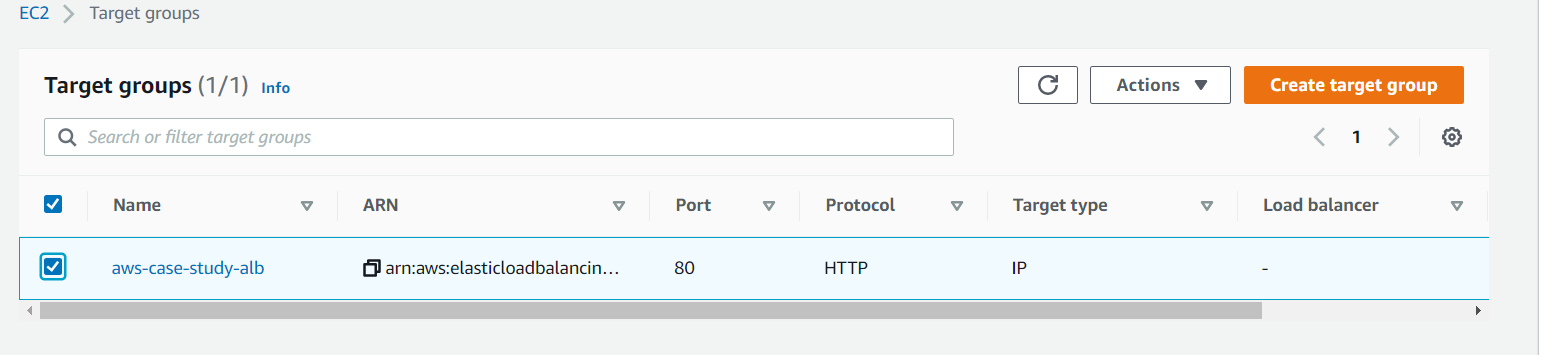
Copying files through Ec2-user :



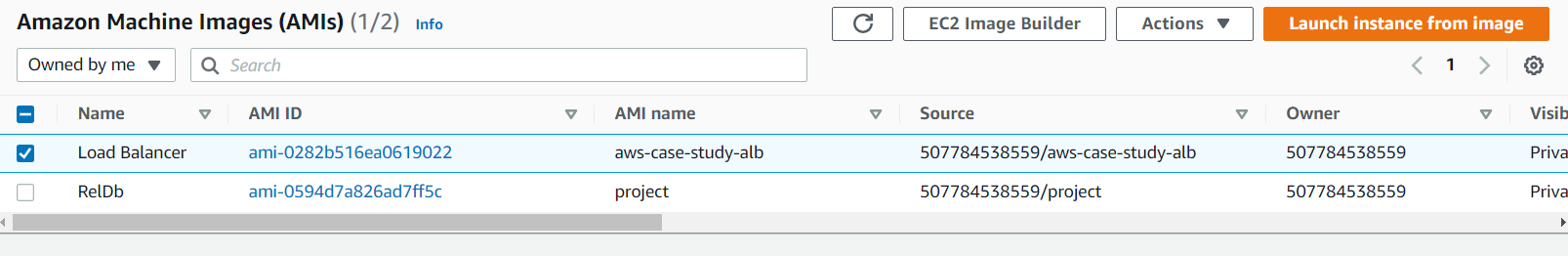
` `

Application Load Balancer:

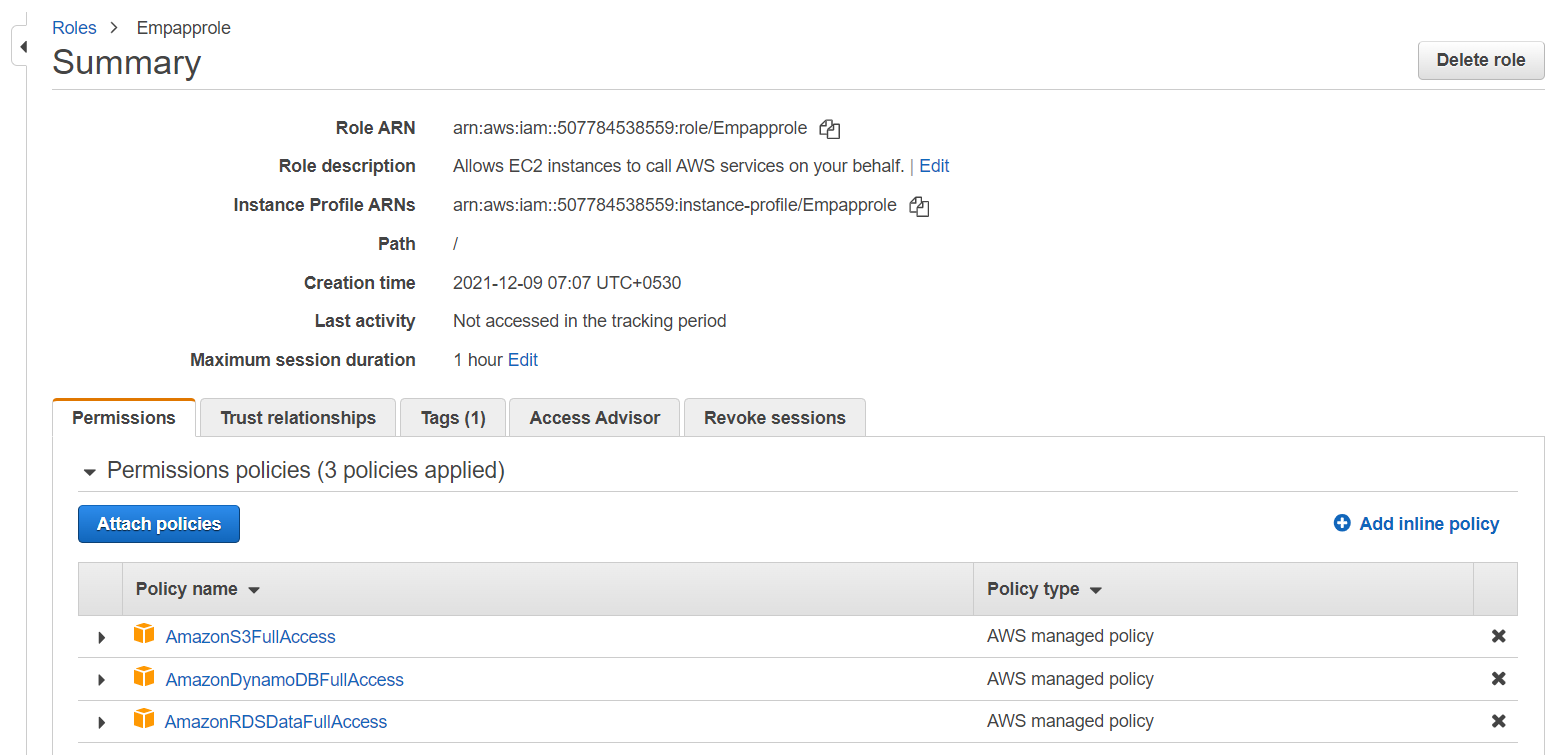




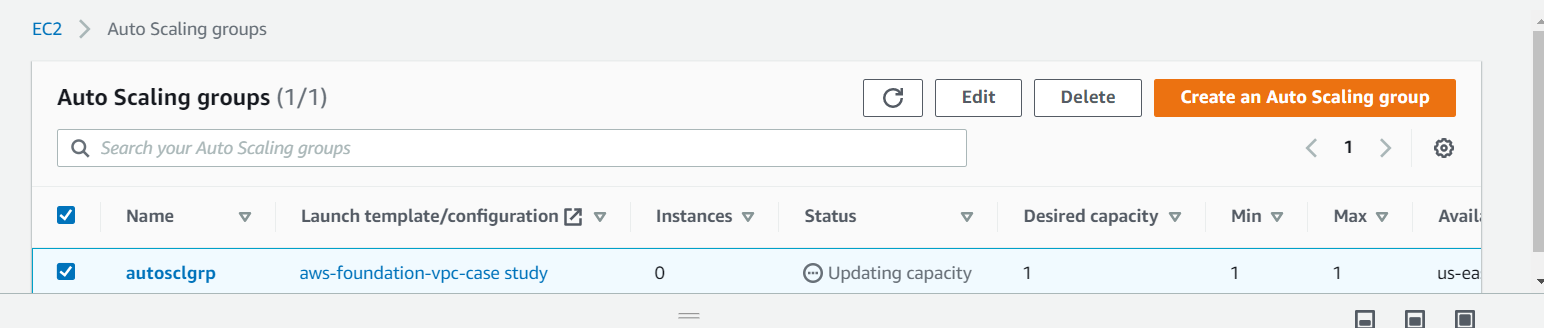
AMI:



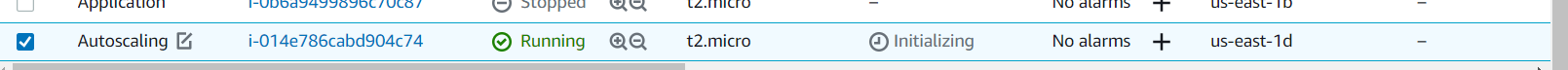
IAM Role:



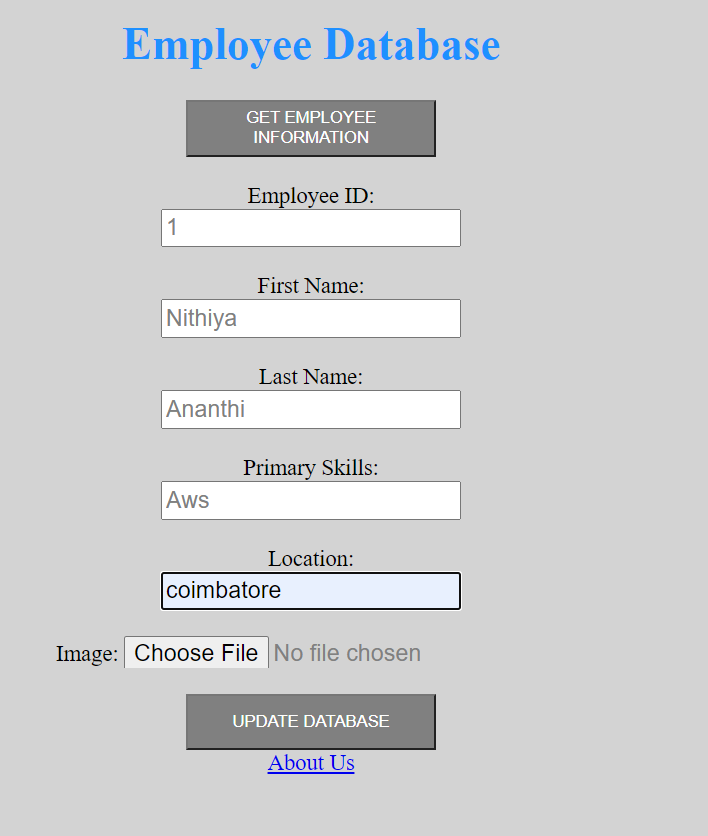
Auto scaling group:



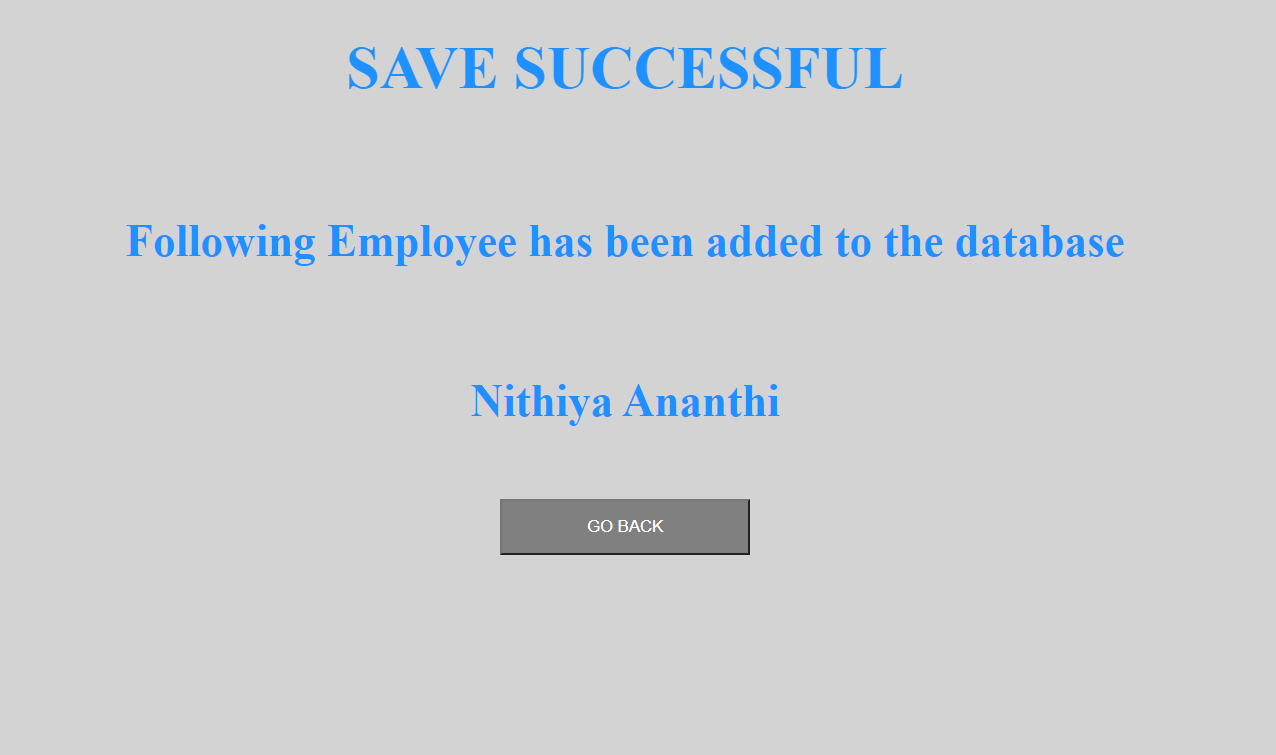
Auto scaling instace:



Add Employee details:

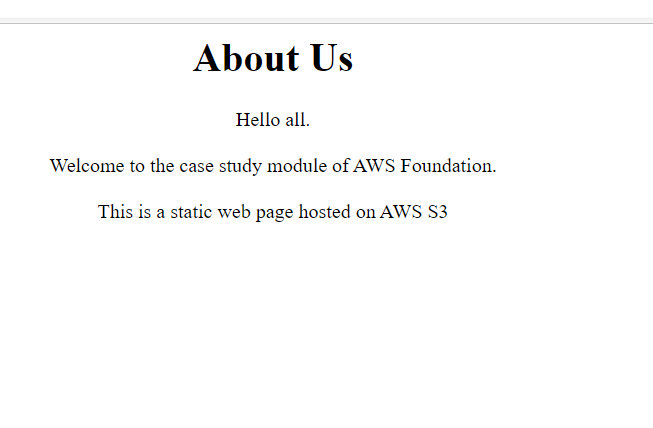


Output of Add employee:





About us:



Creation of SNS topic:

