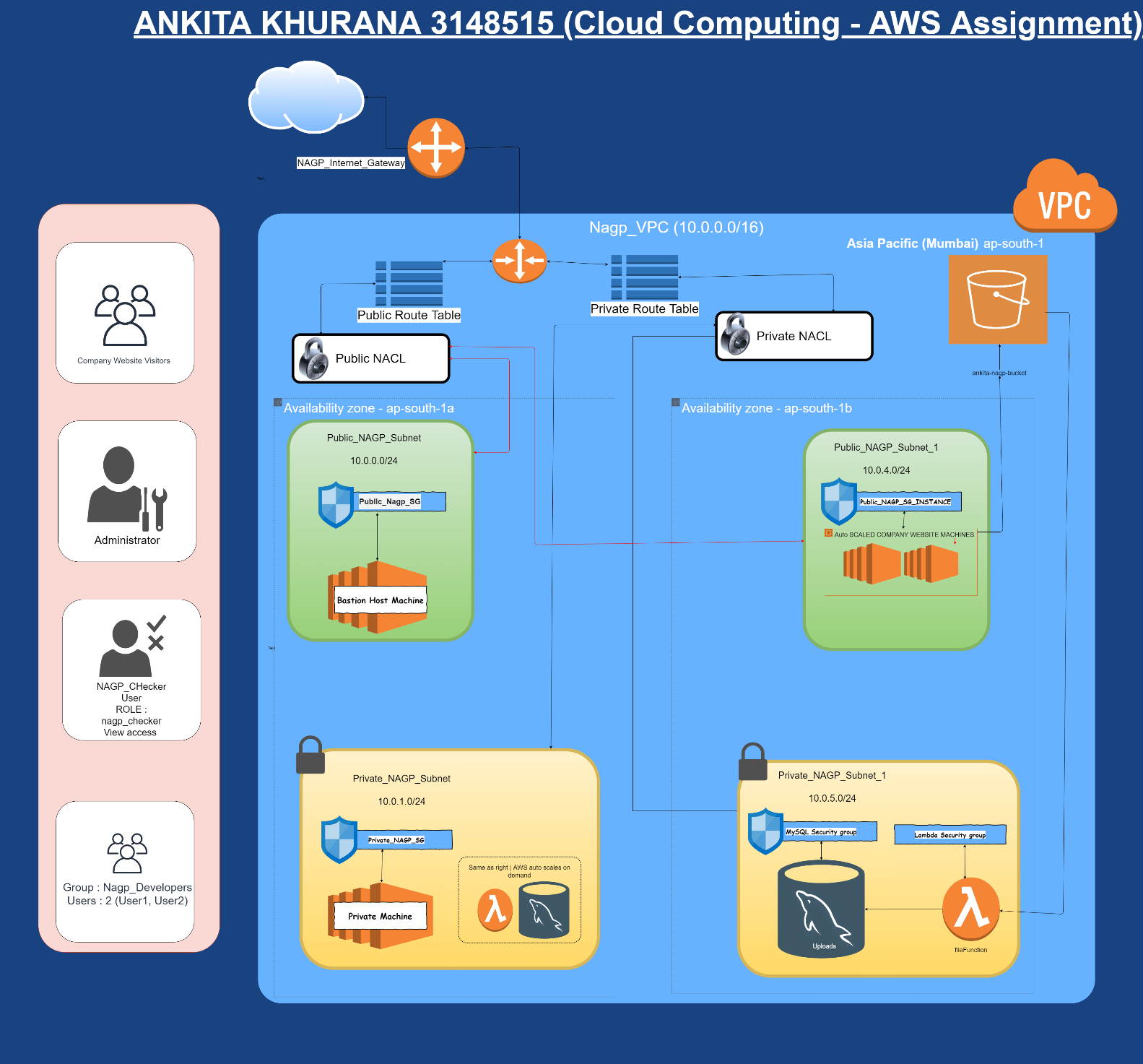
**CLOUD COMPUTING ASSIGNMENT 2**

* **Ankita Khurana**
* **3148515**

**Note:**

1. **As mentioned in the group (No NAT Gateway) is added to the architecture due to non-free service.**

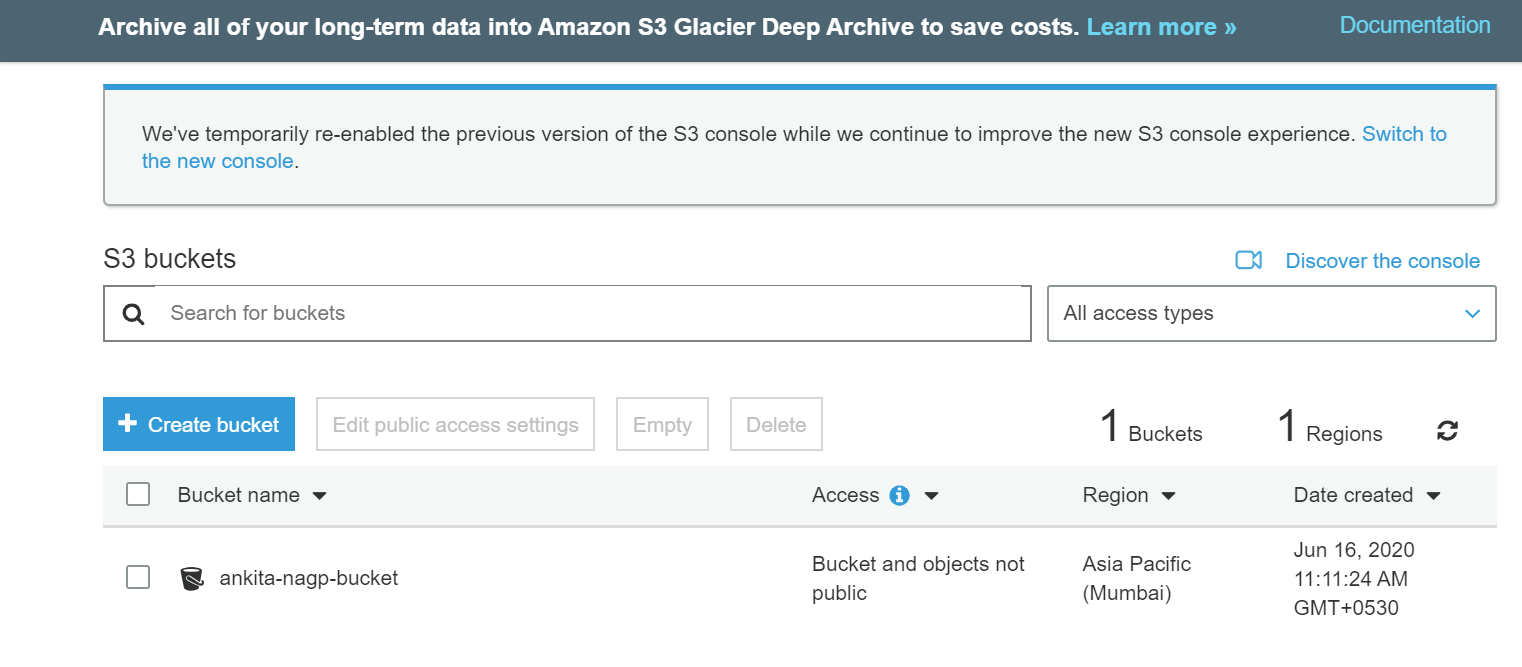


**ARCHITECTURE**

**RESOURCES AND SERVICES used till date for XYZ project**

1. **S3 Bucket: ankita-nagp-bucket**

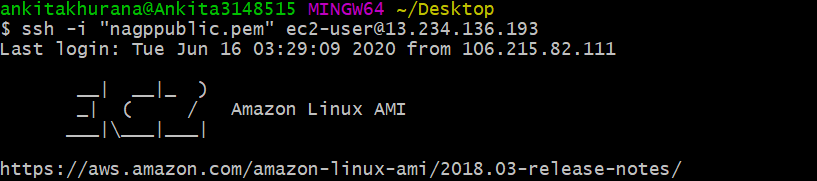
* S3 bucket created in Asia Pacific region to store files uploaded via the company website.
* S3 bucket data is extracted in the company website to view data.
* Any upload of file in the bucket Triggers Lambda function (fileFunction)



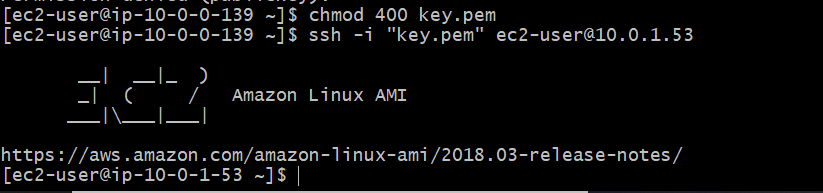
1. **EC2 Machines :** (Bastion host on Public subnet + Private Ec2 on private subnet + Auto Scale company website hosted on machines on Public subnet 1)

* One EC2 machine acts as a Bastion host to connect to the private ec2 of company
* It also acts as an SSH layer to connect the admin to the RDS-MYSQL inside the private subnet
* The AUTO SCALING Machines which hosts the Company Website are present on the public network and load balanced via target group

1. Logging into Public Bastion Host



b. Logging into Private Machine from Bastion Host



**# Auto Scaling – user data**

#!/bin/bash

sudo yum update -y

sudo yum install git -y

cd /home/ec2-user

git clone https://github.com/AnkitaKhurana/fileuploader.git

sudo chmod a+rwx fileuploader

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

sudo chmod a+rwx /.nvm

. /.nvm/nvm.sh

nvm install node

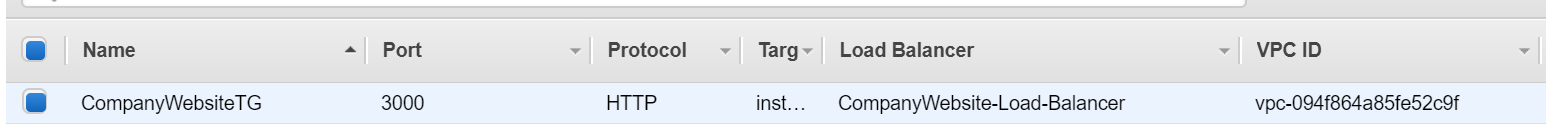
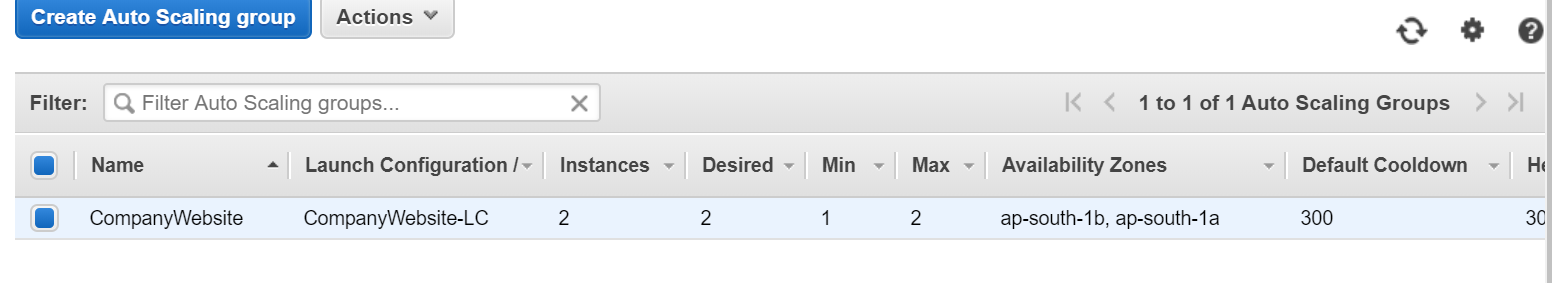
cd fileuploader

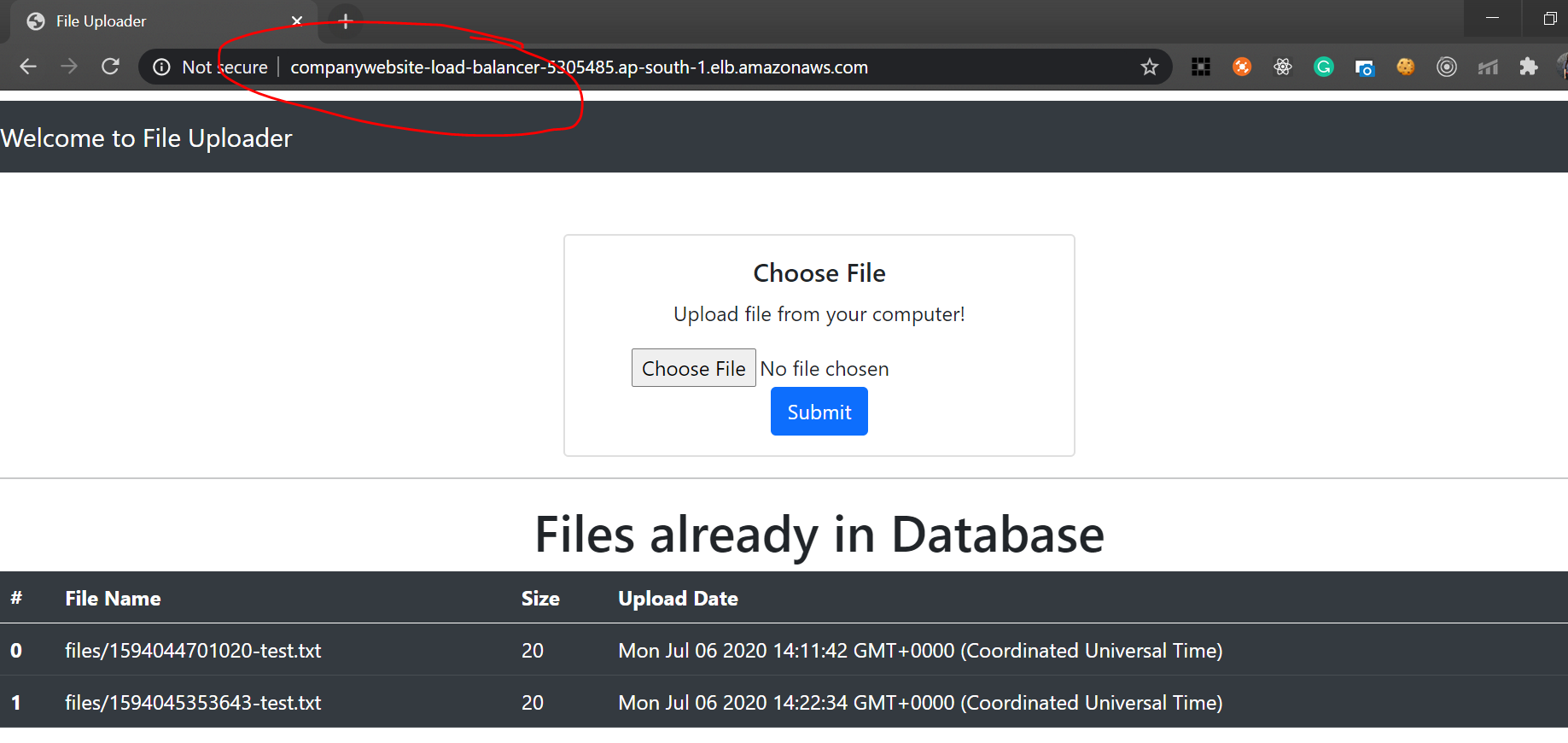
npm install

npm start

1. **Target Group and Load Balancer || Launch Configuration and Auto Scaling group :**

* Target group listens to the port 3000 and load balancer loads between the EC2 machines instances created via autoscaling group and ports to 80.
* The auto scaling group SCALES between (min: 1 , max : 2 ) machines and scale when the “number of bytes in> 10000”.

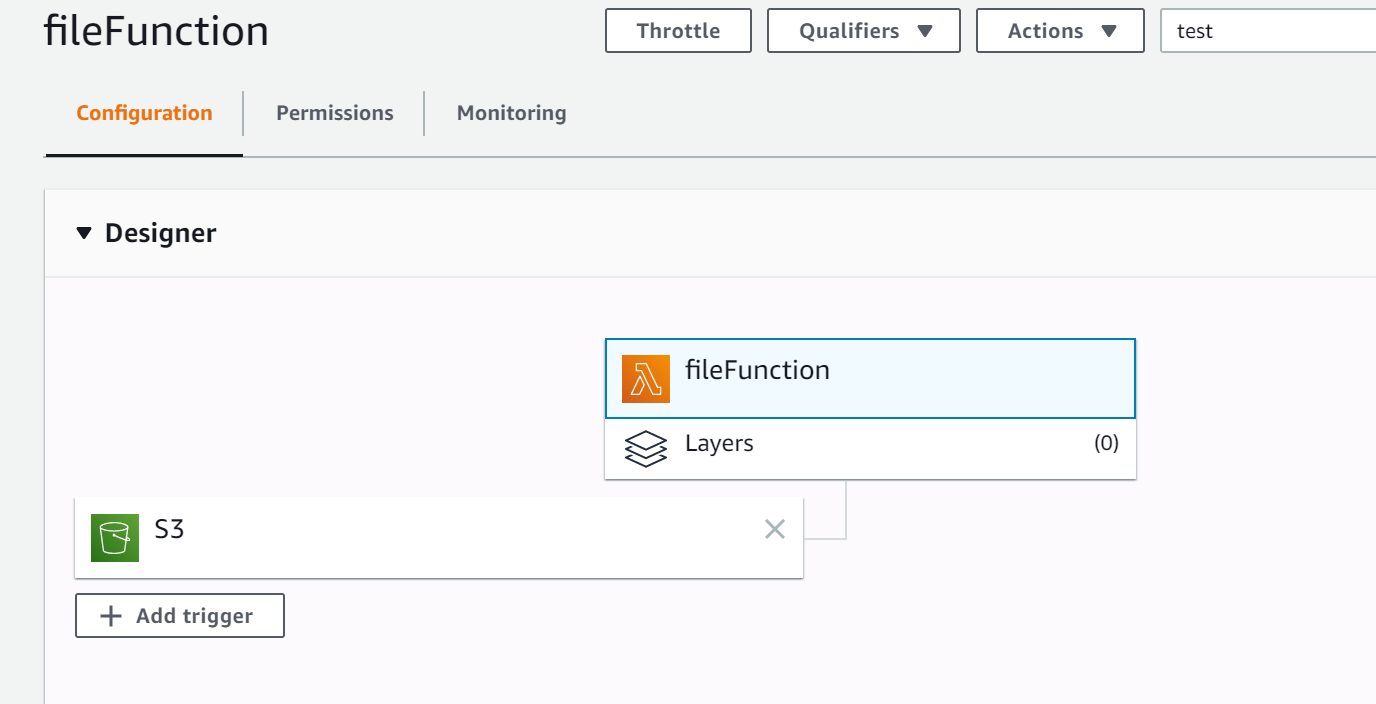
 



**Code in :** https://github.com/AnkitaKhurana/fileuploader.git

1. **Lambda Function : fileFunction**

* Any upload of file in the bucket (ankita-nagp-bucket)Triggers Lambda function (fileFunction)
* Responsible for adding file size and entry of the uploaded file in the private RDS



**Code in Lambda function:**

**var AWSXRay = require('aws-xray-sdk-core')**

**var captureMySQL = require('aws-xray-sdk-mysql')**

**var mysql = captureMySQL(require('mysql2'))**

**const username = process.env.databaseUser**

**const password = process.env.databasePassword**

**const host = process.env.databaseHost**

**const AWS = require('aws-sdk')**

**var s3 = new AWS.S3();**

**exports.handler = async(event, context) => {**

**var bucket = event.Records[0].s3.bucket.name;**

**var key = event.Records[0].s3.object.key;**

**var size = event.Records[0].s3.object.size;**

**var connection = mysql.createConnection({**

**host: host,**

**user: username,**

**password: password,**

**database: 'data'**

**});**

**connection.connect();**

**let date = new Date().toISOString().slice(0, 19).replace('T', ' ');**

**var result;**

**let query = "INSERT INTO myfiles(filename,size, upload) values('" + key +"',"+size+ ",'" + date + "');";**

**connection.query(query, function (error, results, fields) {**

**if (error) throw error;**

**console.log("Ran query: " + query);**

**for (result in results)**

**console.log(results[result])**

**});**

**return new Promise((resolve, reject) => {**

**connection.end(err => {**

**if (err)**

**return reject(err)**

**const response = {**

**statusCode: 200,**

**body: "Saved to RDS"**

**}**

**resolve(response)**

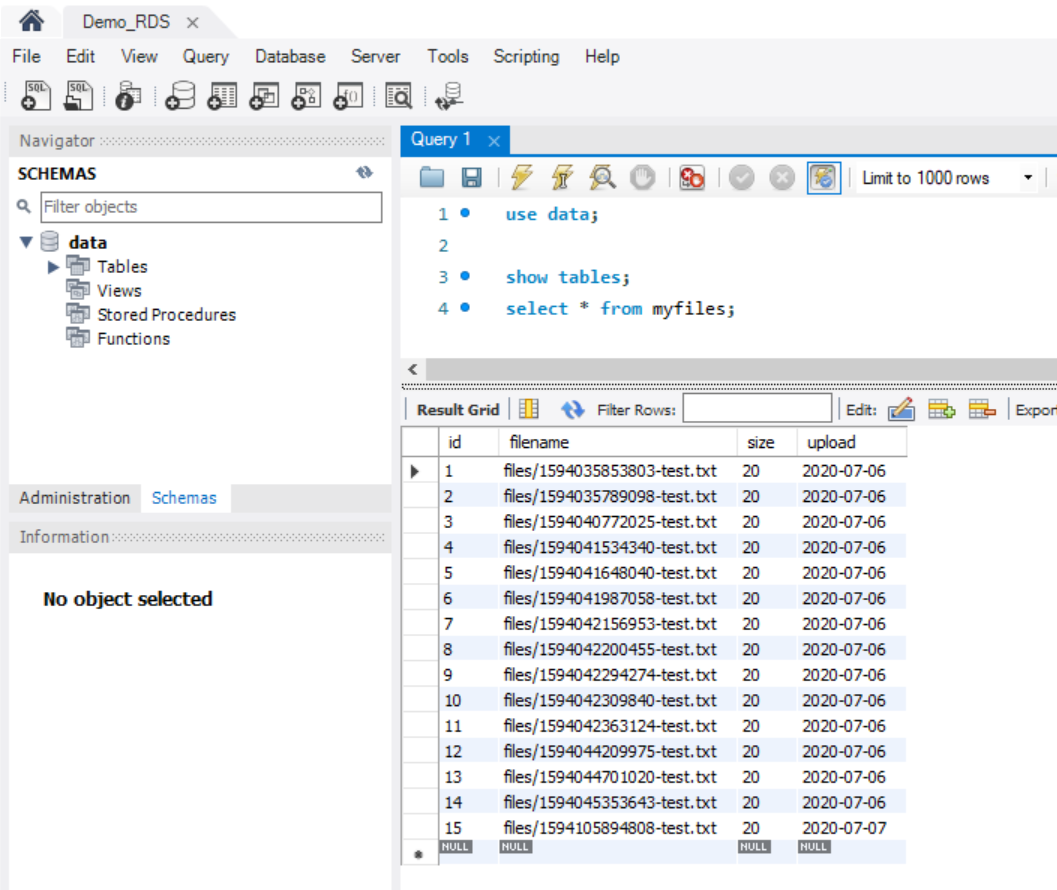
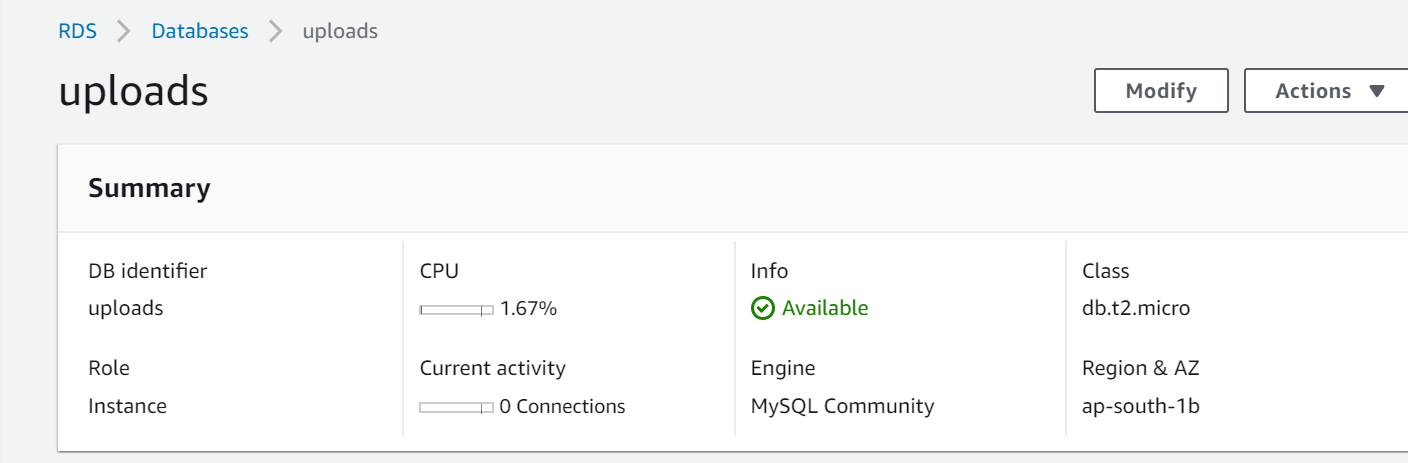
**})**

**});**

**}**

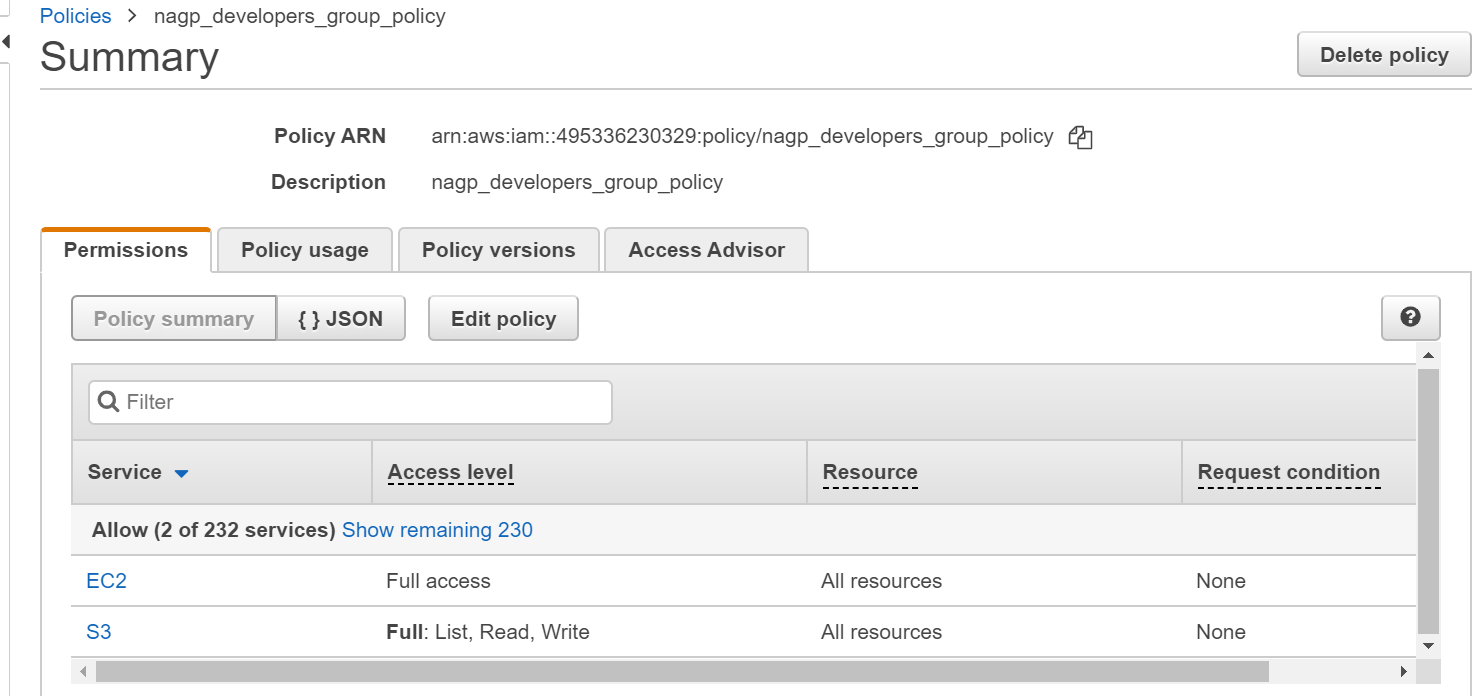
1. **RDS – MySQL : uploads :**

* Contains company sensitive data (here which text file was uploaded in the S3 Bucket and the size)

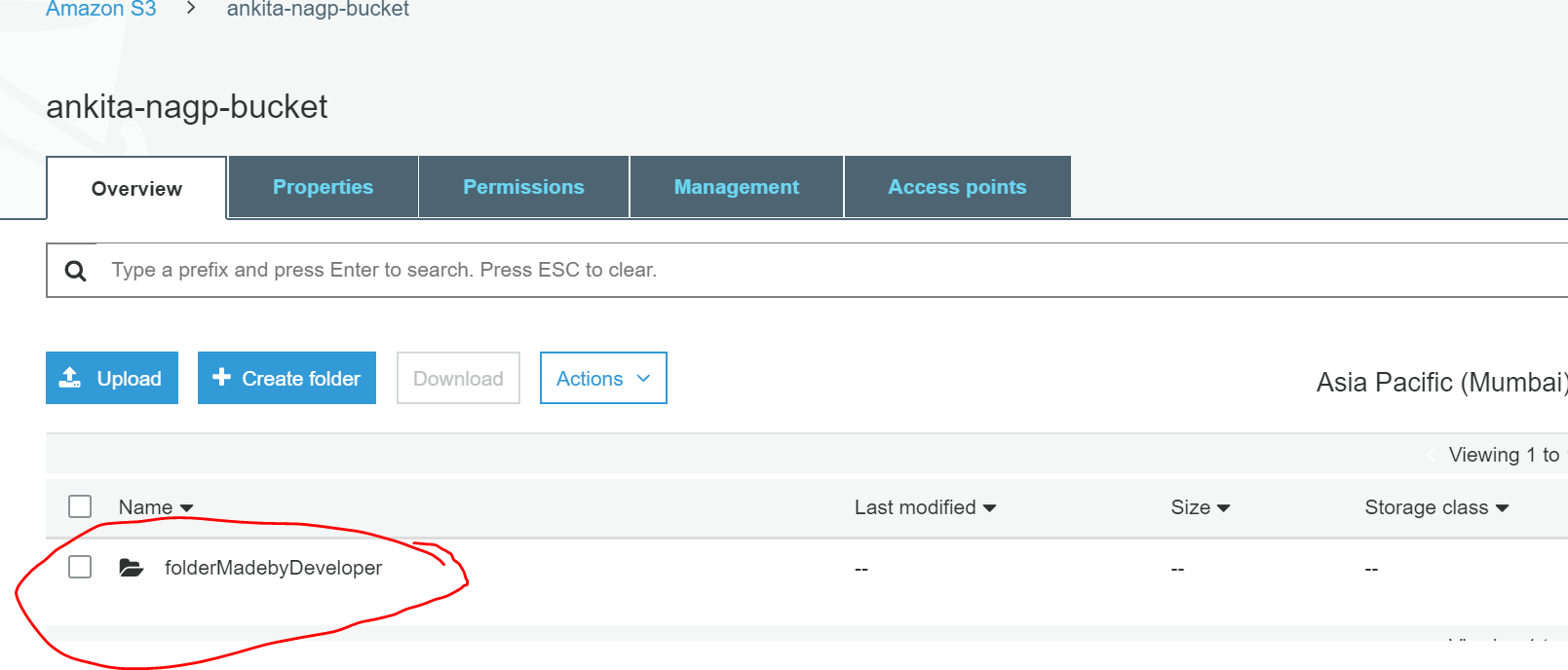


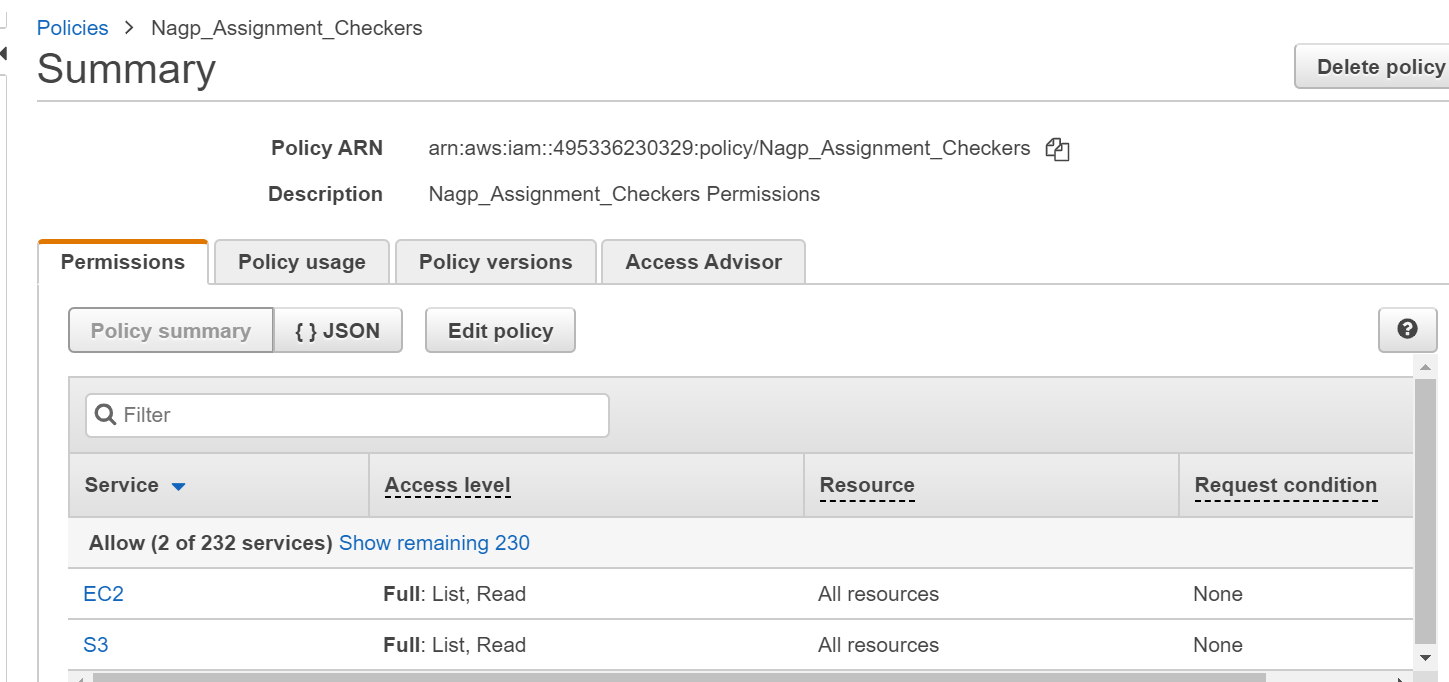
**POLICIES**

1. Policy for DEVELOPERS (R/W on S3 Buckets + Full access (Create etc) on EC2)



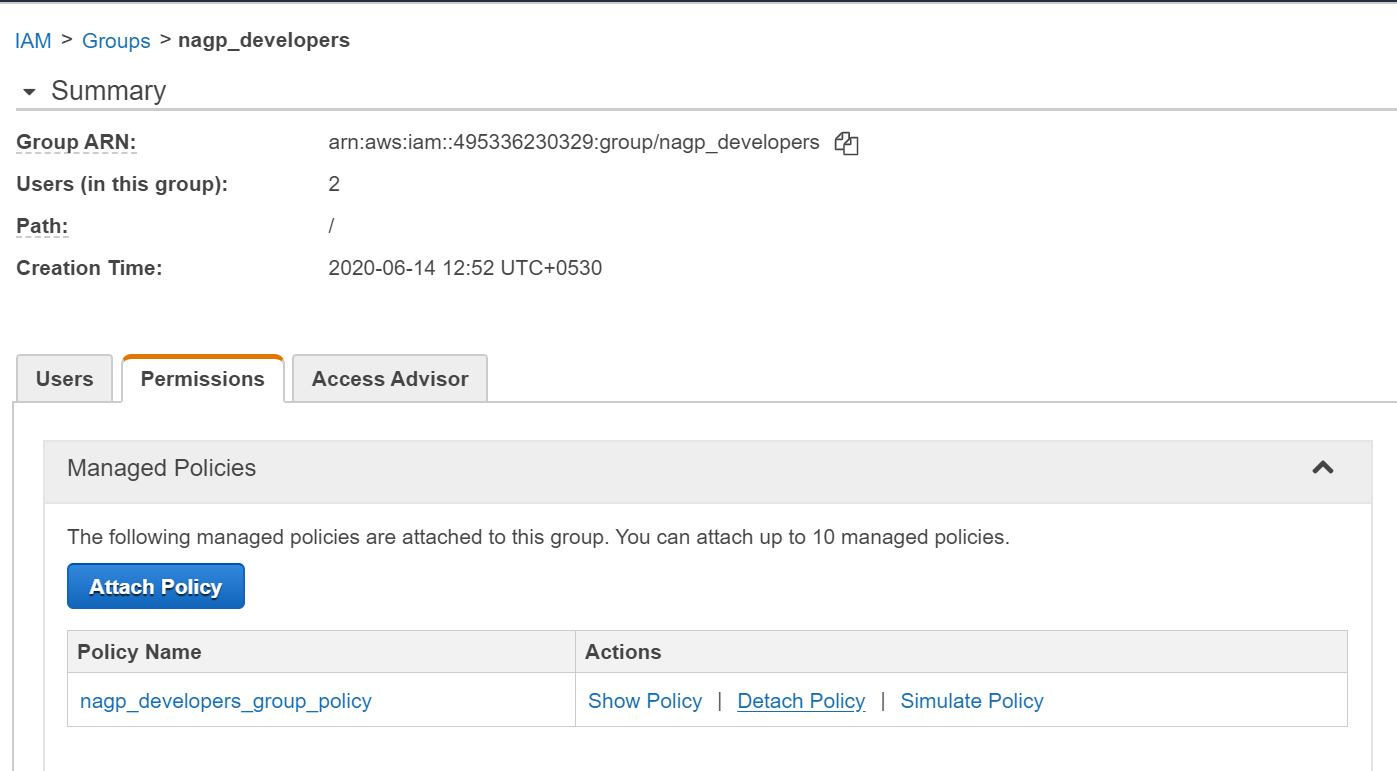
Example: On logging from a user in developer group (Writing in Bucket)



1. Policy for roles/users who wants to have min permissions (Assignment checkers)

**GROUPS**

1.Developer group (For giving access to Services for development, two users made under this group)

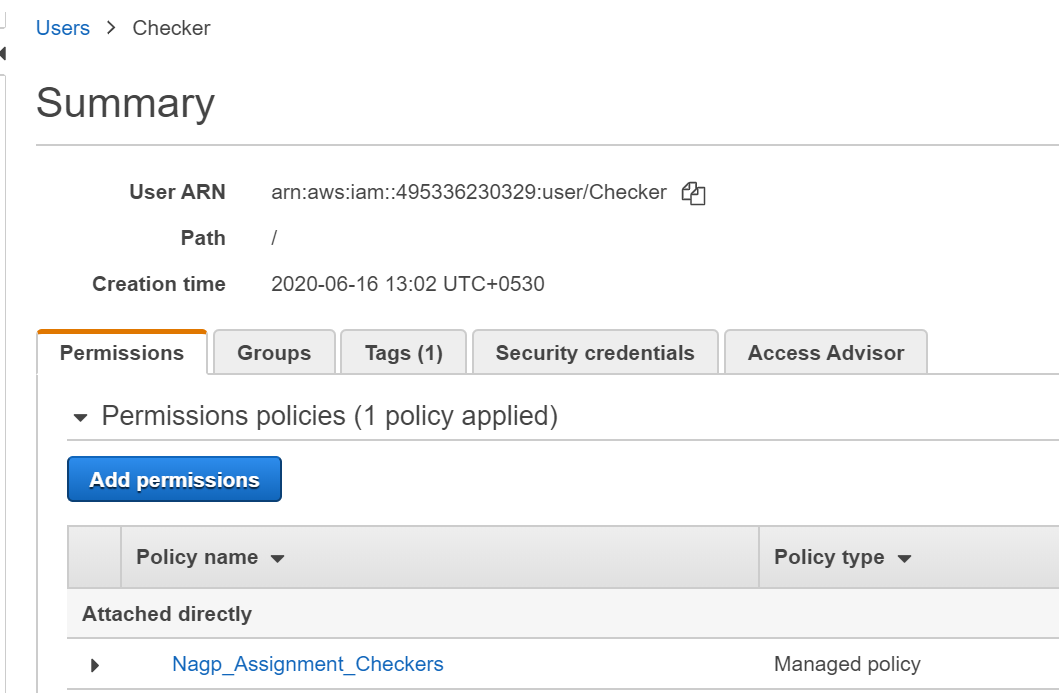


**USERS**

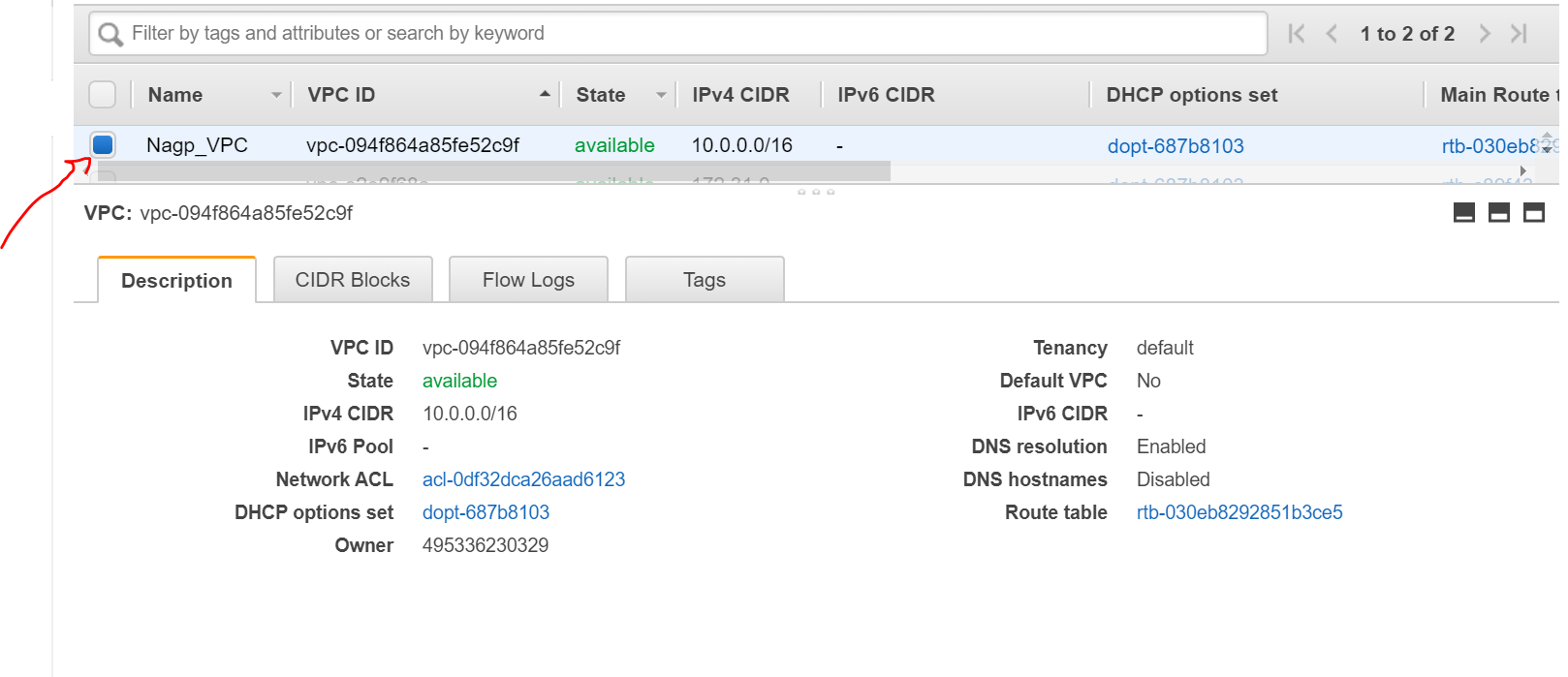
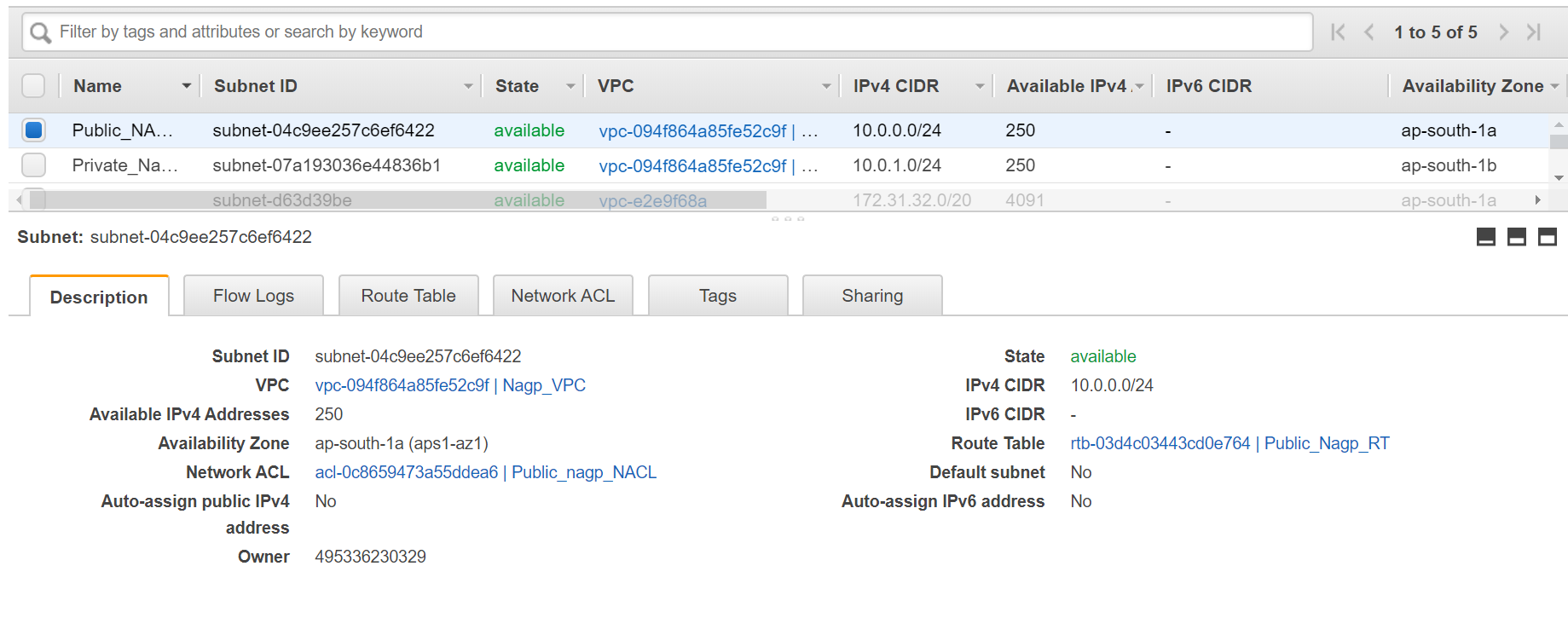
1. Developers user accounts made for the development team

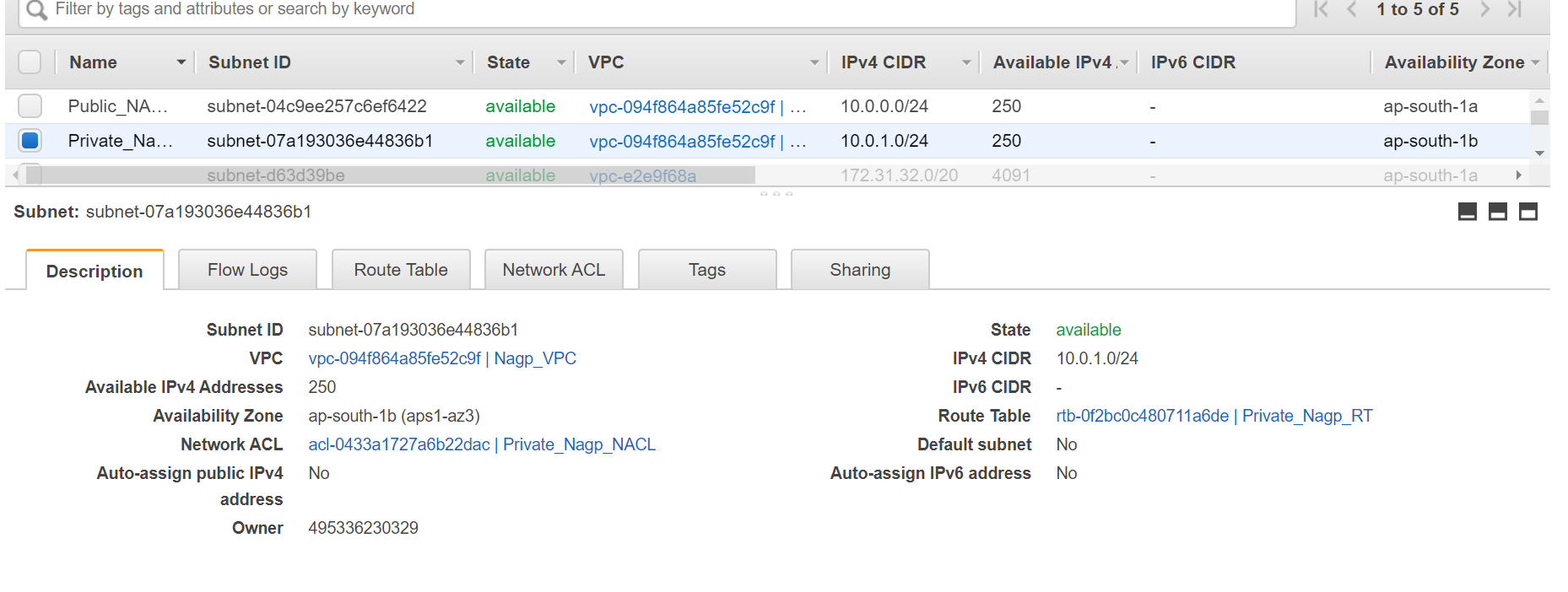


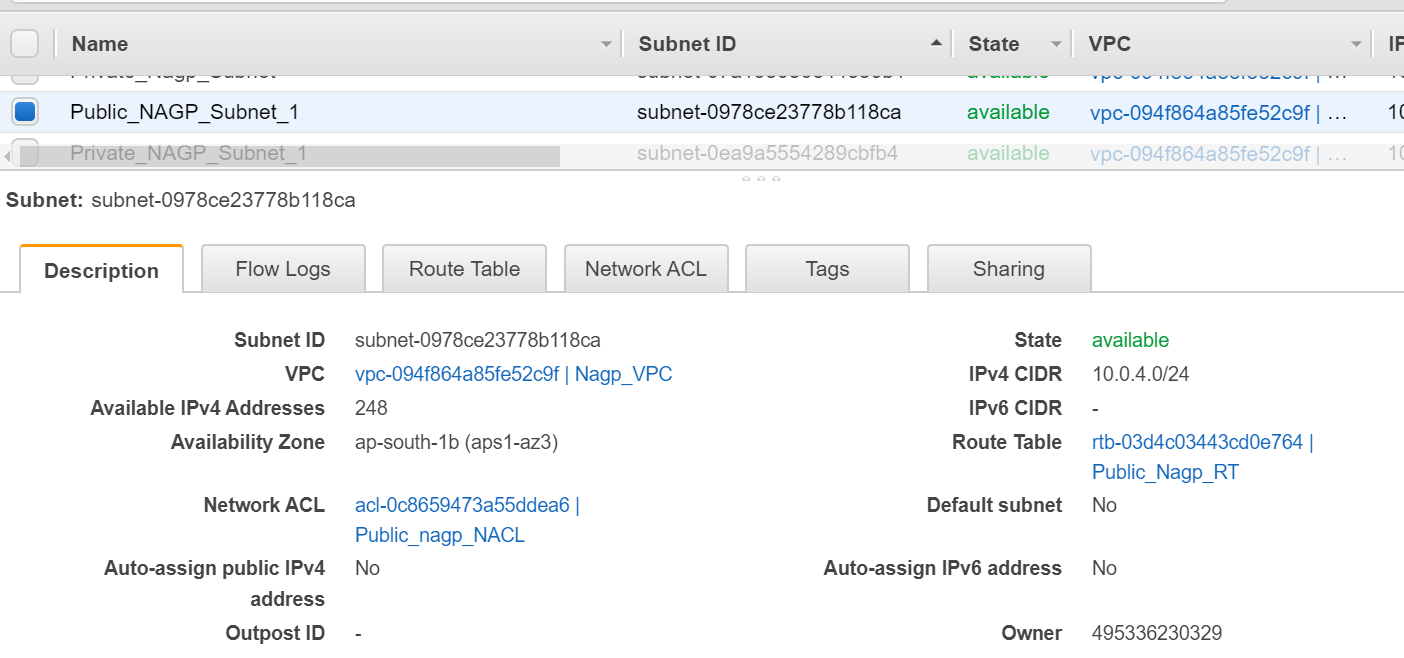
1. Single checker user account made for temp access



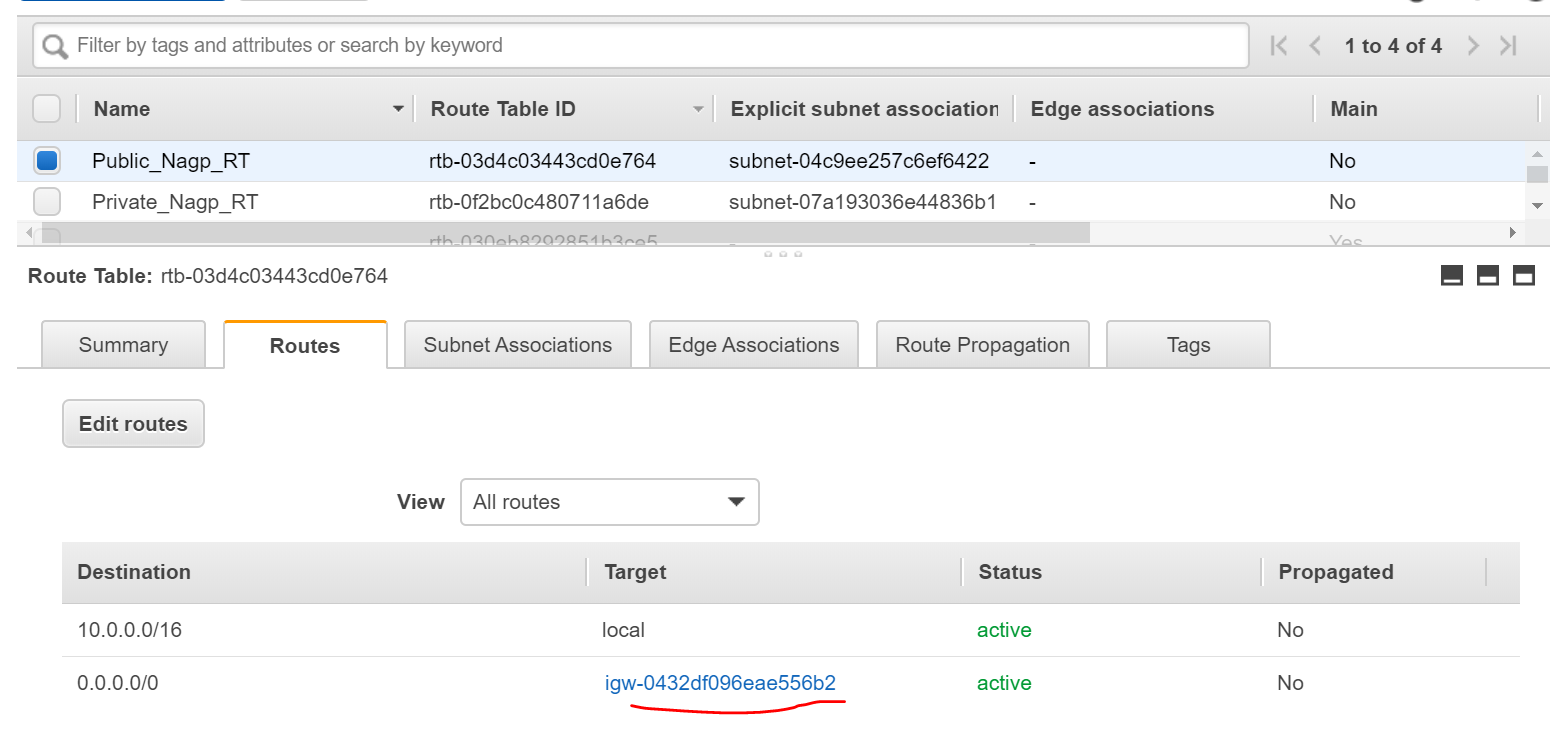
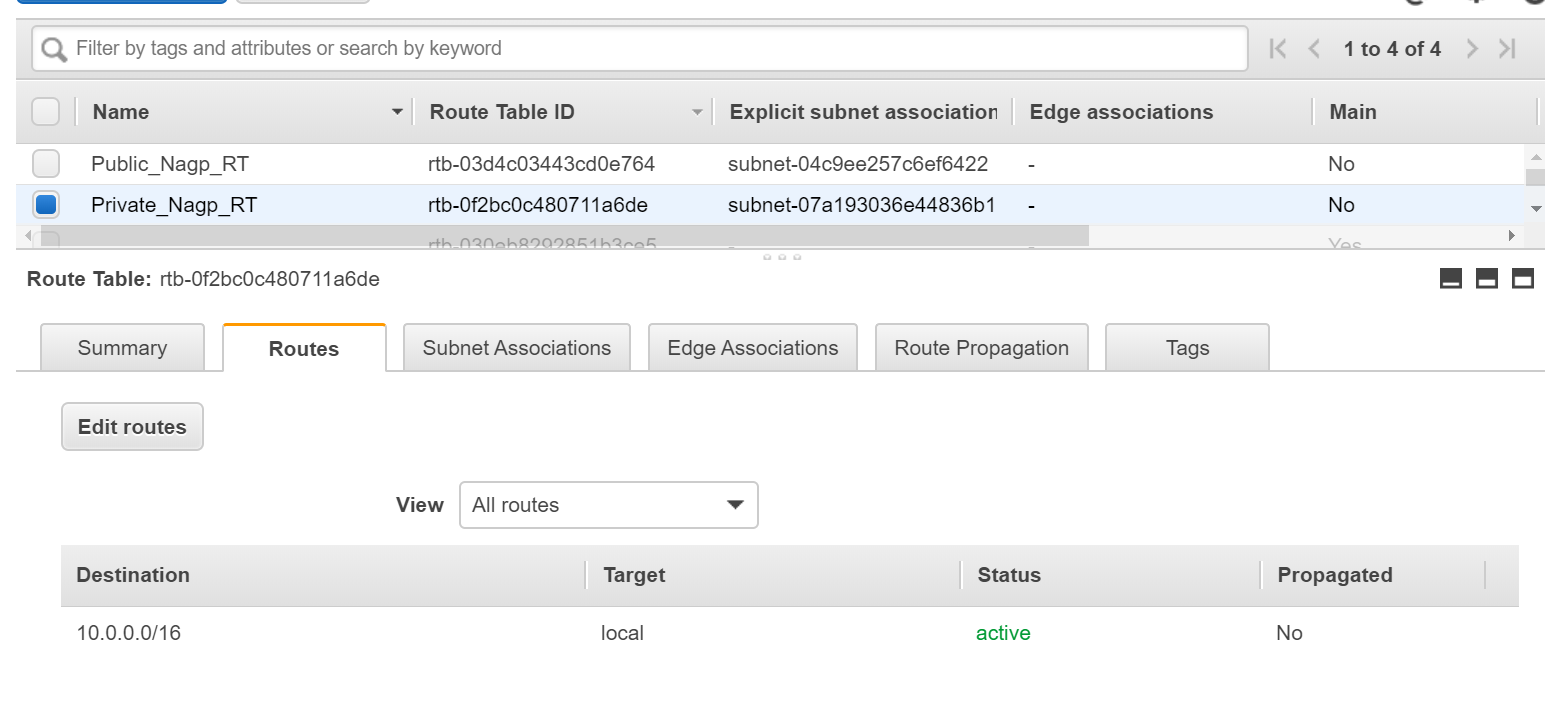
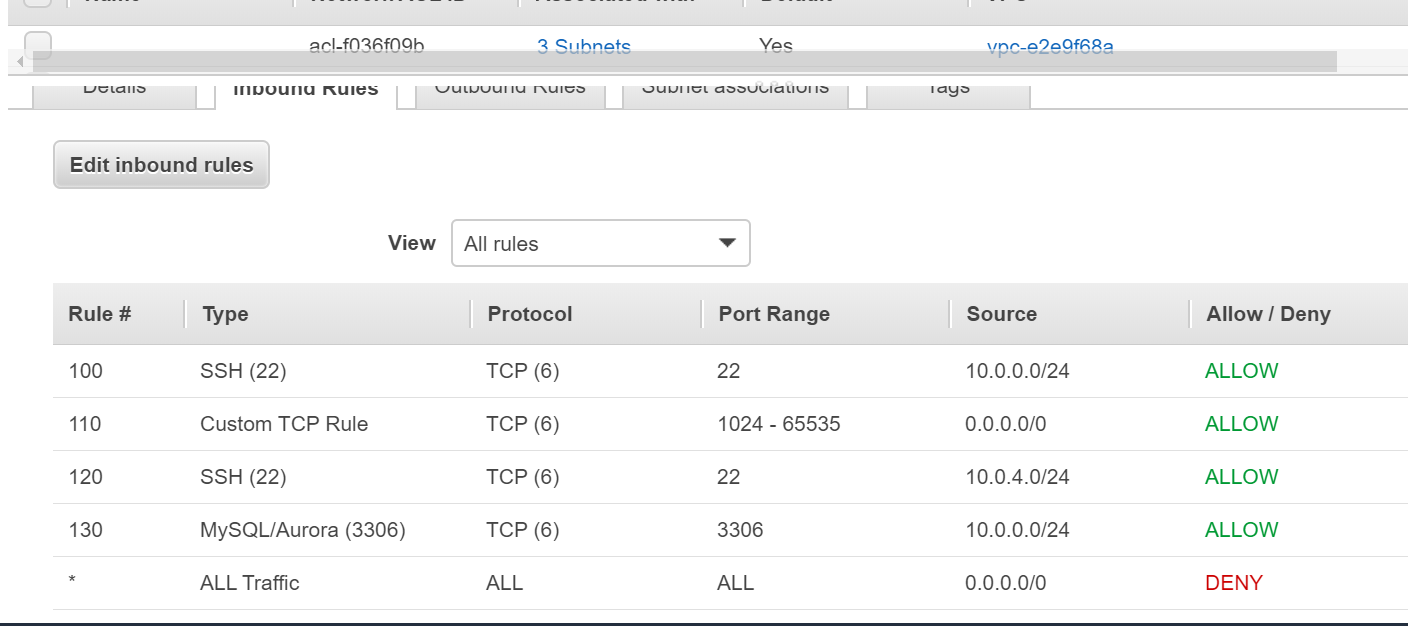
**VPC COMPONENTS**

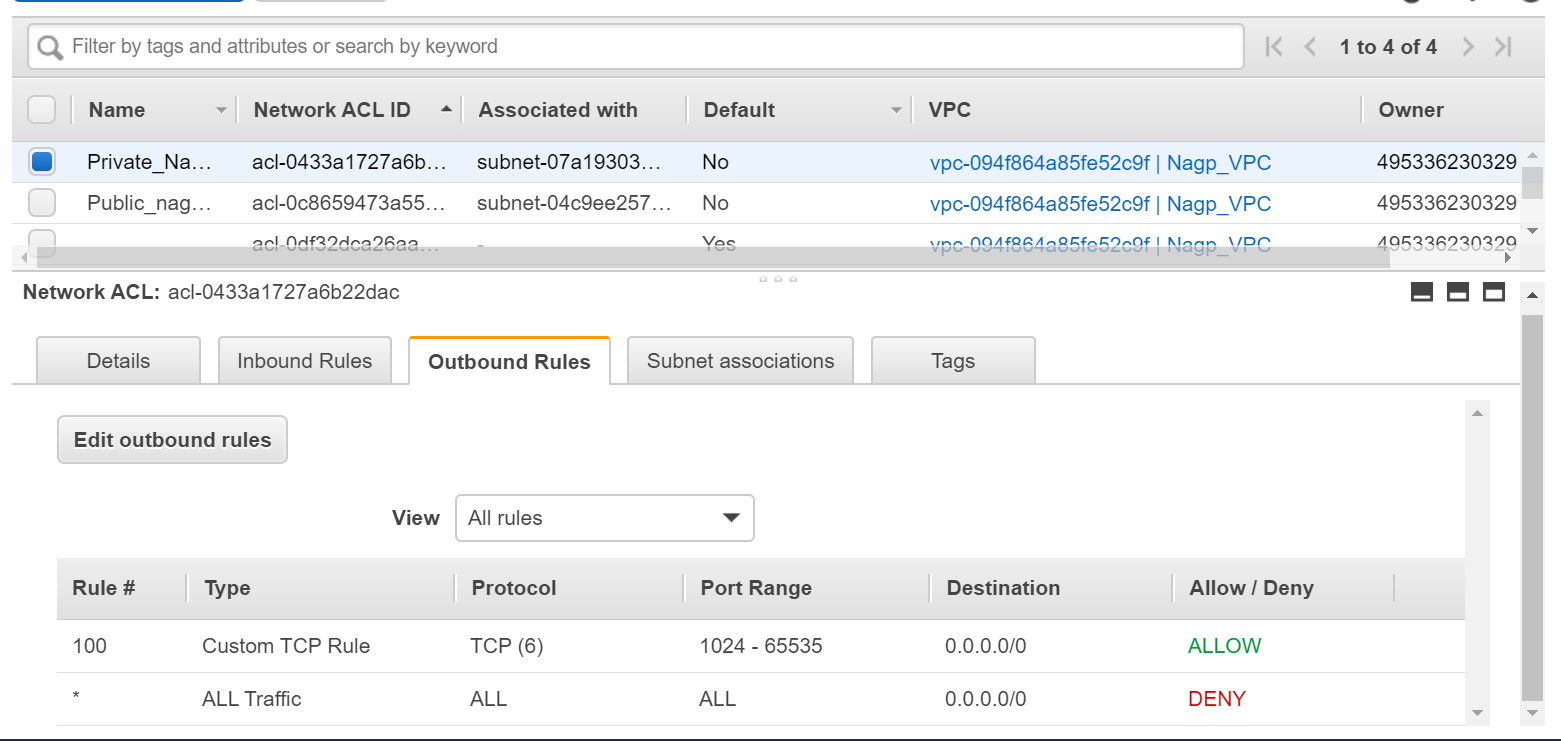
1. Nagp\_VPC : 10.0.0.0/16
2. Public\_Nagp\_Subnet : 10.0.0.0/24
3. Private\_Nagp\_Subnet: 10.0.1.0/24



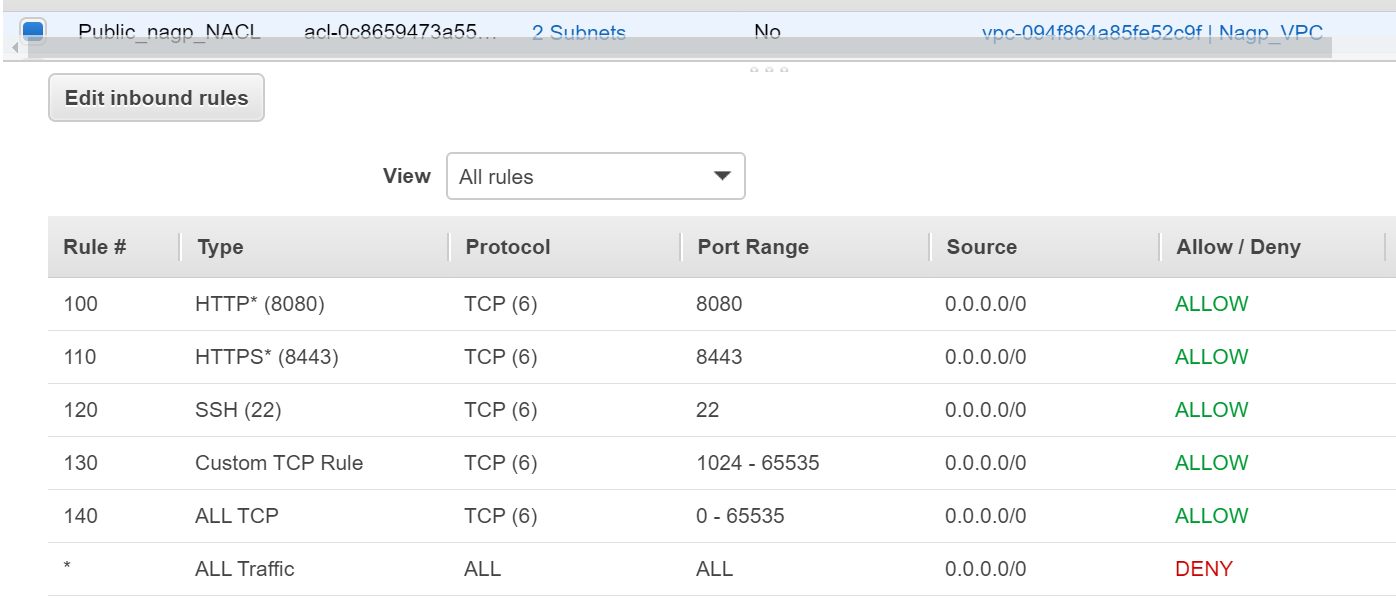
1. Public\_Nagp\_Subnet\_1 : 10.0.4.0/24
2. Private\_Nagp\_Subnet\_1 : 10.0.5.0/24

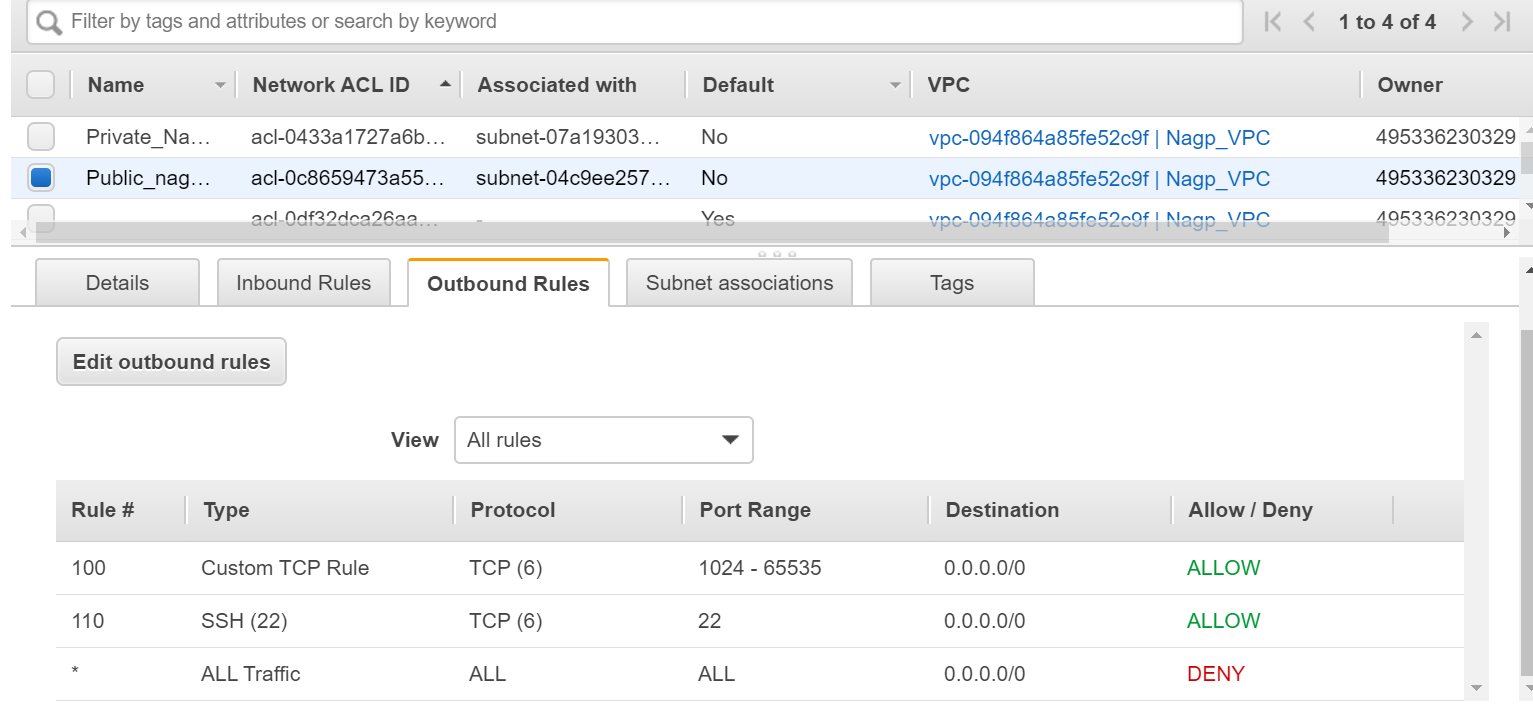
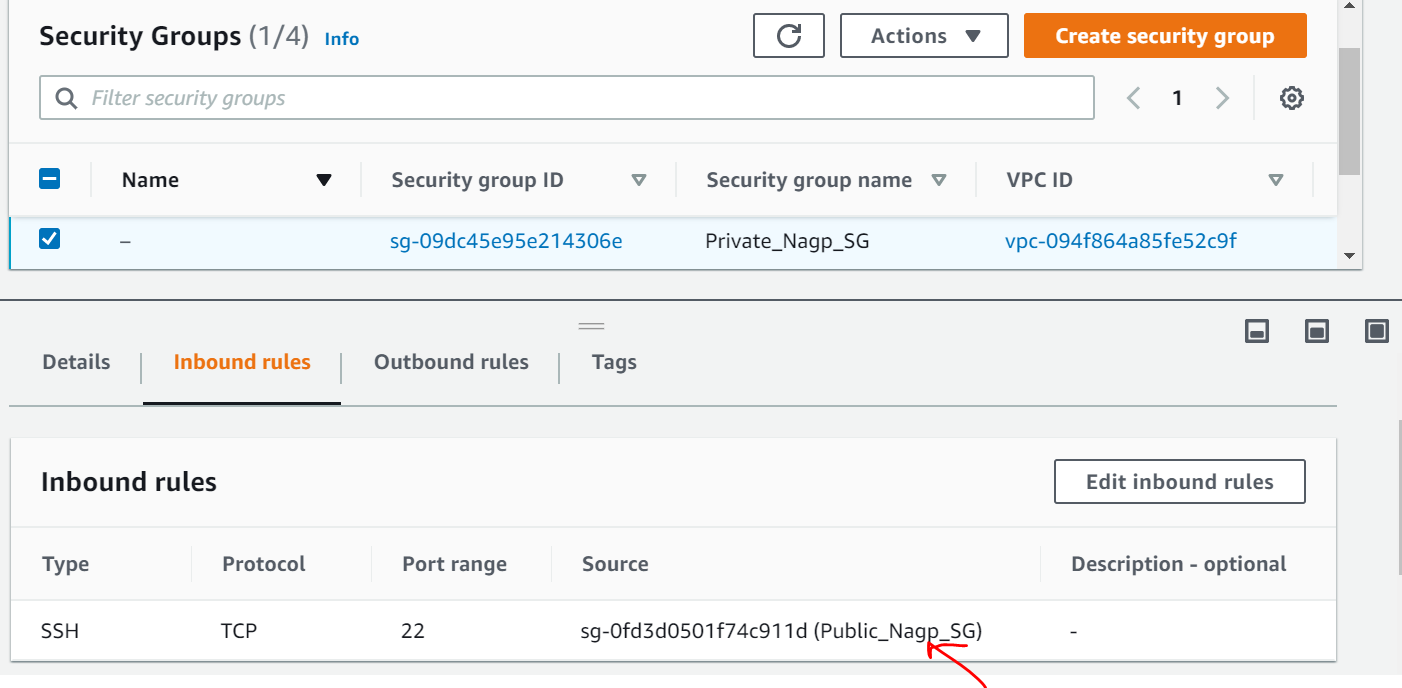
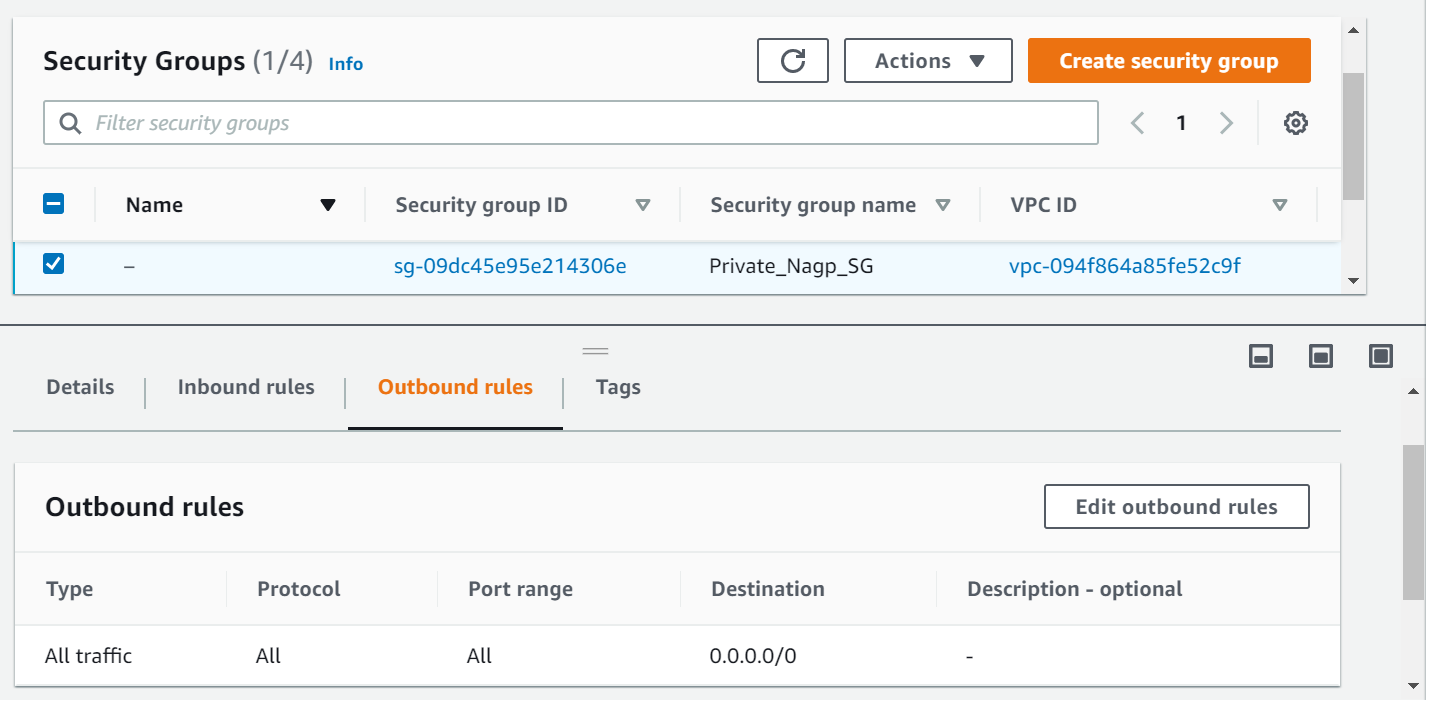
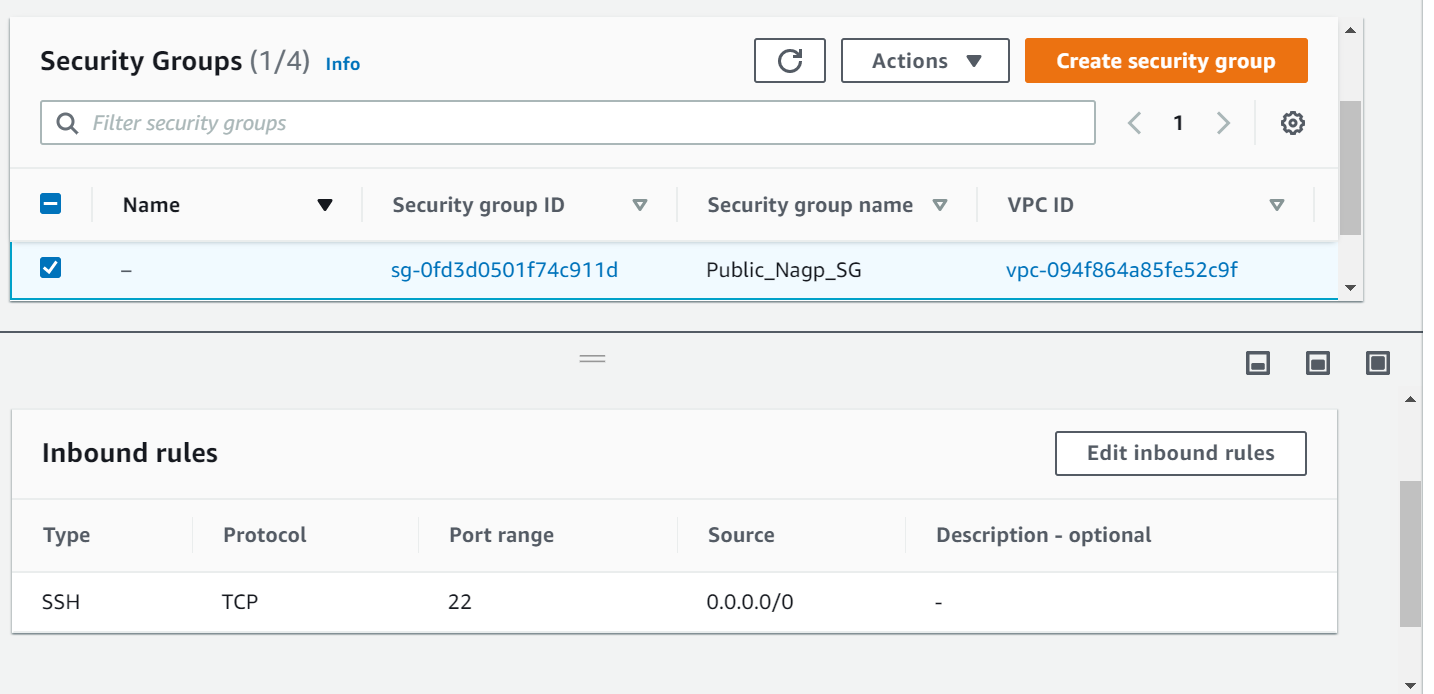
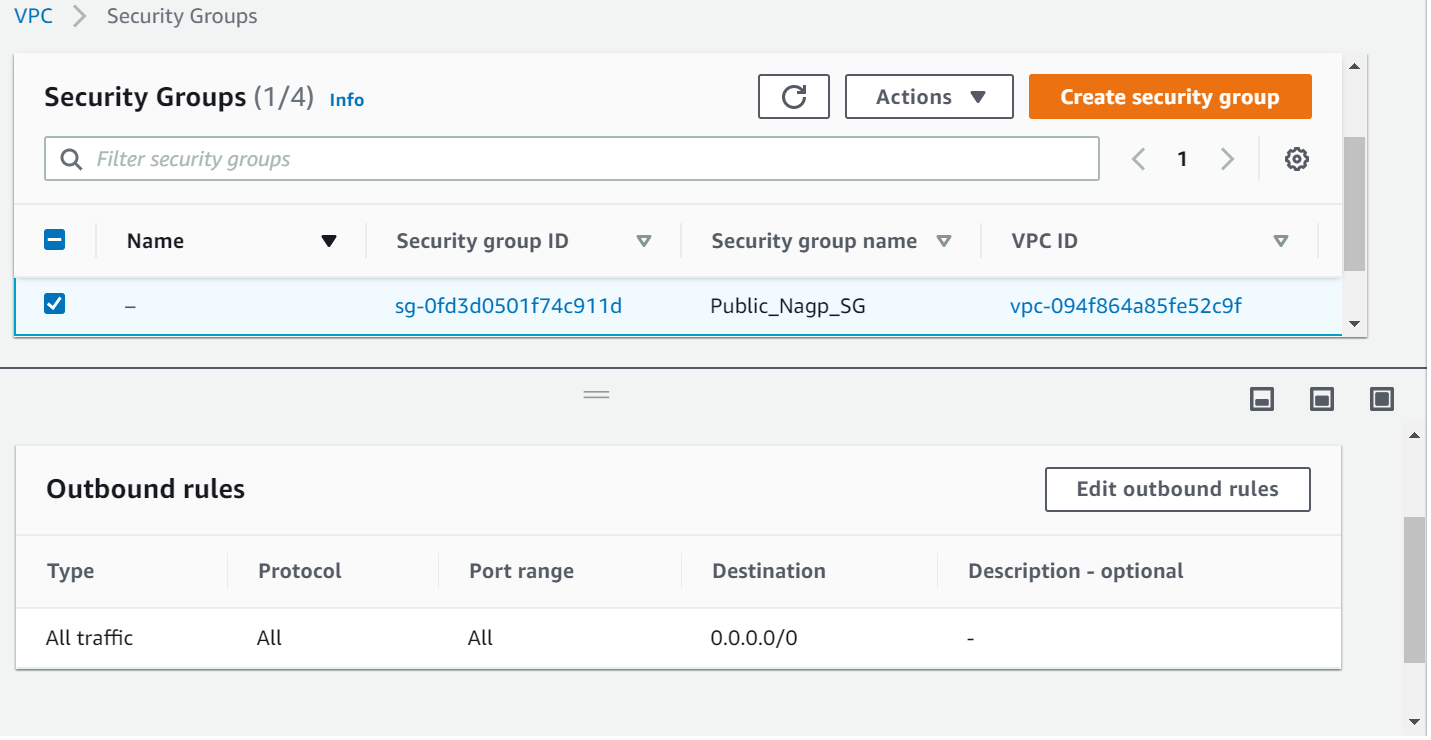


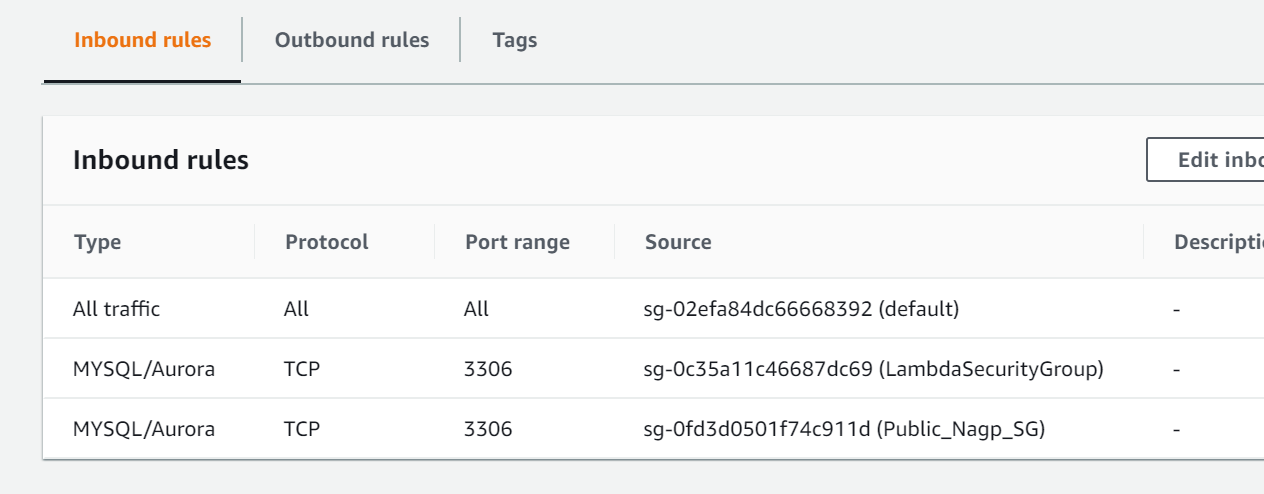
1. Public\_nagp\_RT (Route table) 
2. Private\_nagp\_RT (Route table) 
3. Private\_nagp\_NACL : inbound rules 
4. Private\_nagp\_NACL : outbound rules

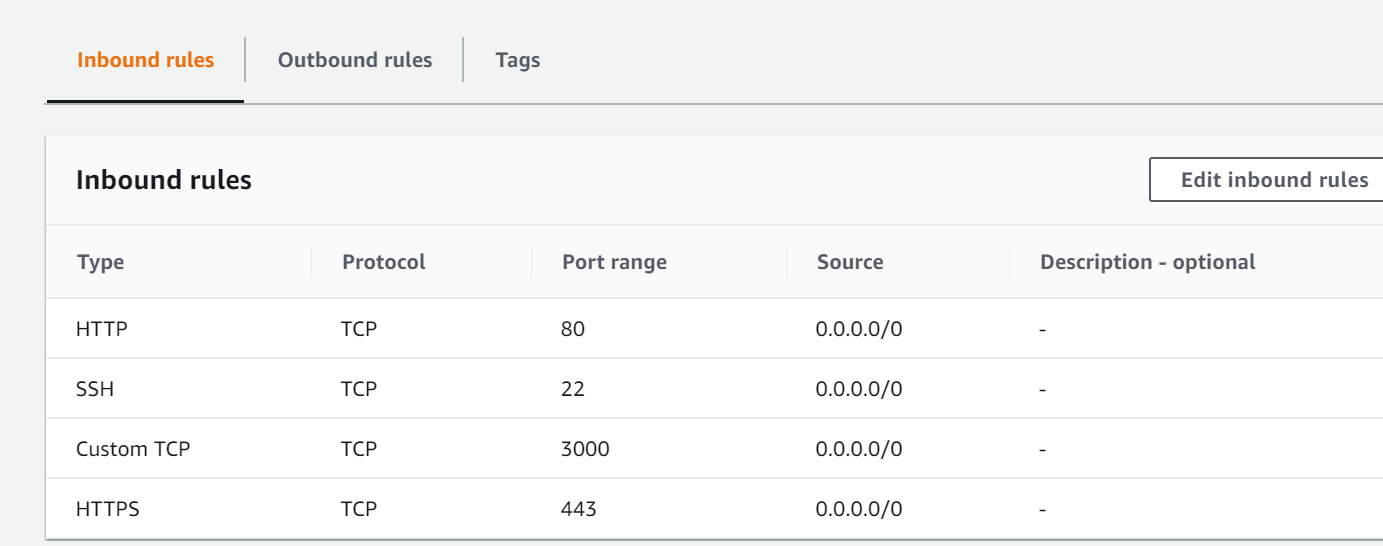


1. Public\_nagp\_NACL : inbound rules



1. Public\_nagp\_NACL : outbound rules
2. Private\_Nagp\_SG (Security Group) : inbound rules
3. Private\_Nagp\_SG (Security Group) : outbound rules 
4. Public\_Nagp\_SG (Security Group) : inbound rules 
5. Public\_Nagp\_SG (Security Group) : outbound rules
6. MySQL Security Group



1. Auto Scaling EC2 Security Group 
2. Internet Gateway 