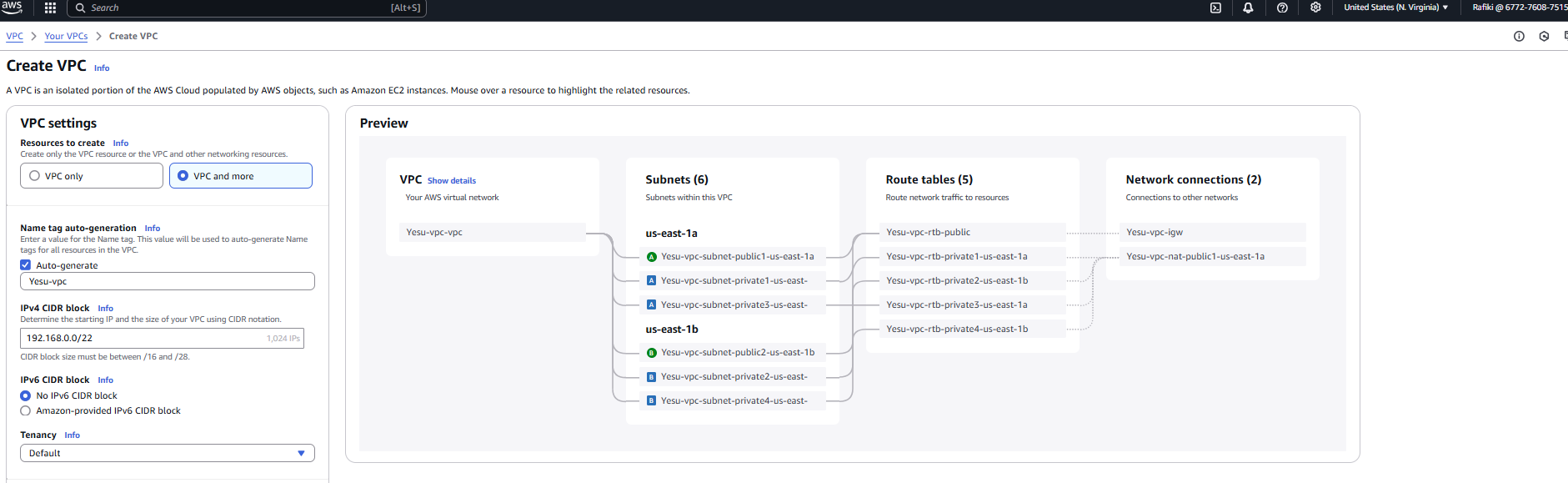
AWS 3 TIER PROJEC T

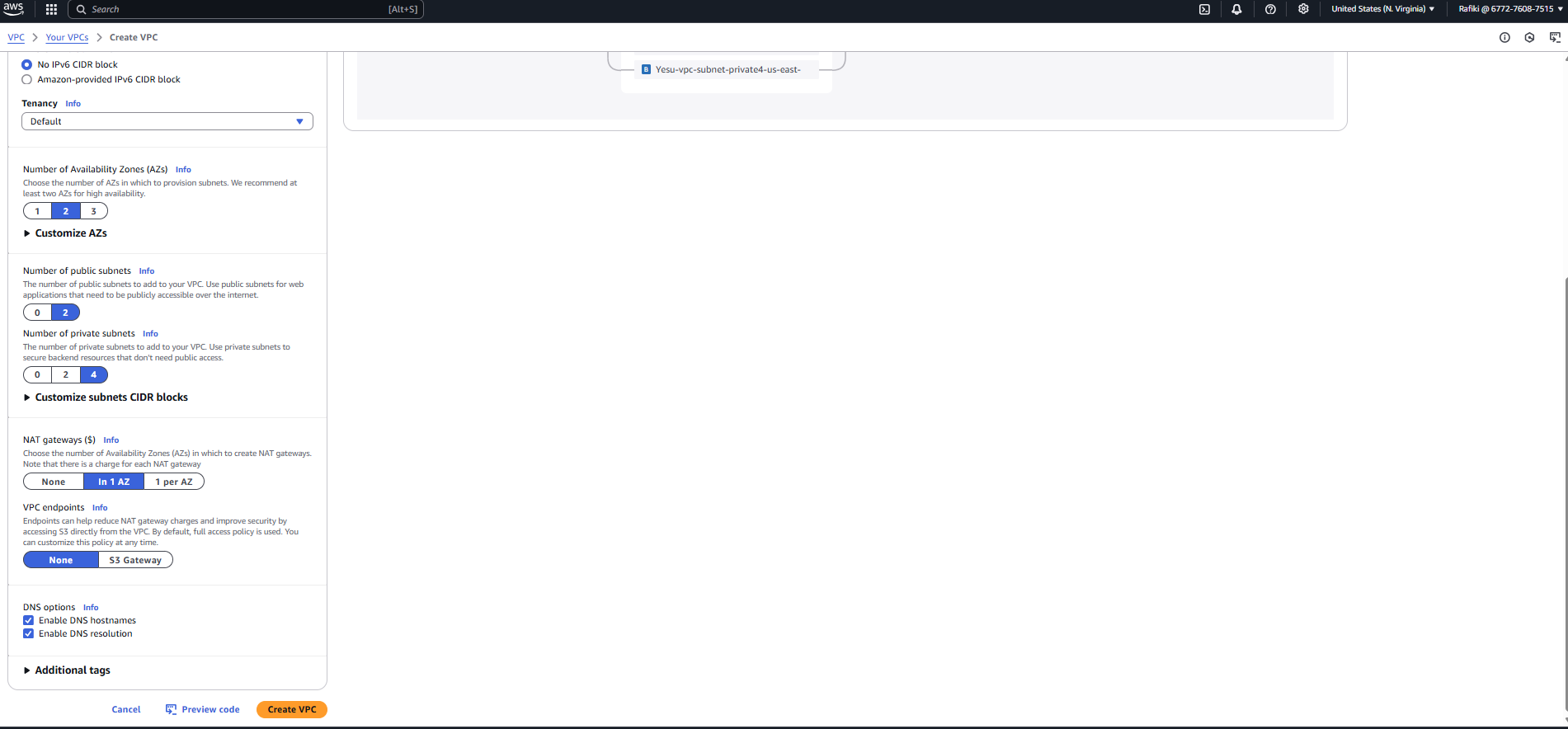
Link to folk project

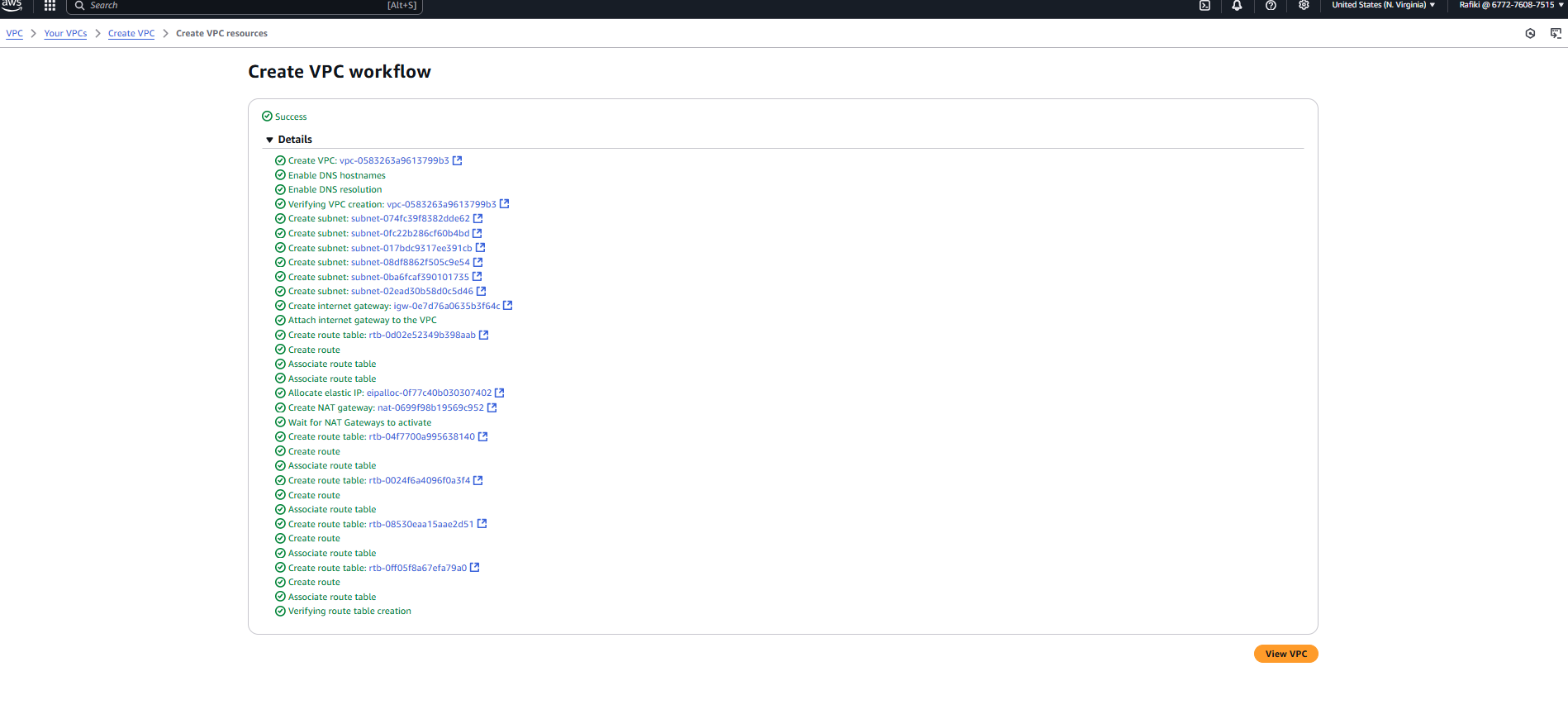
<https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code>

Real time snap shots

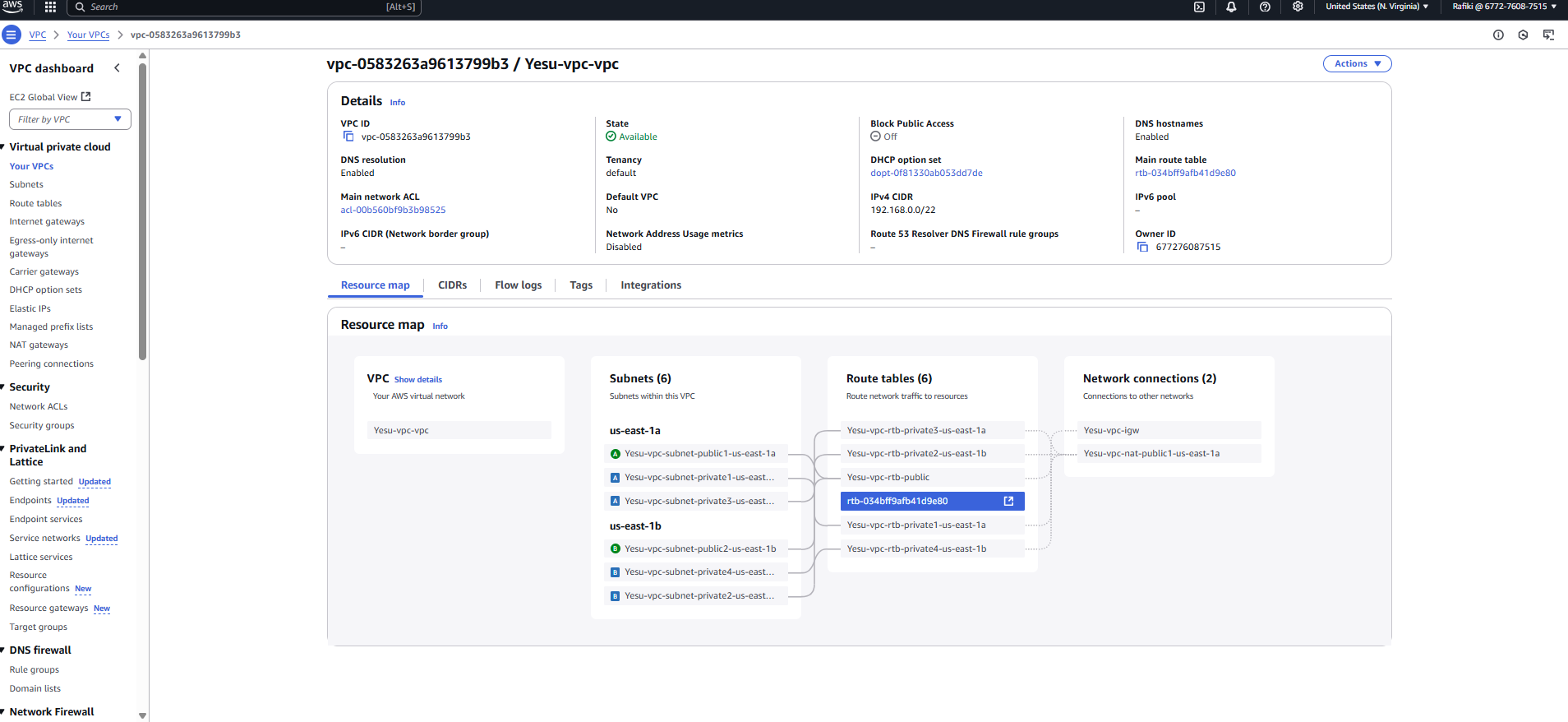
3/10/25



-



View VPC

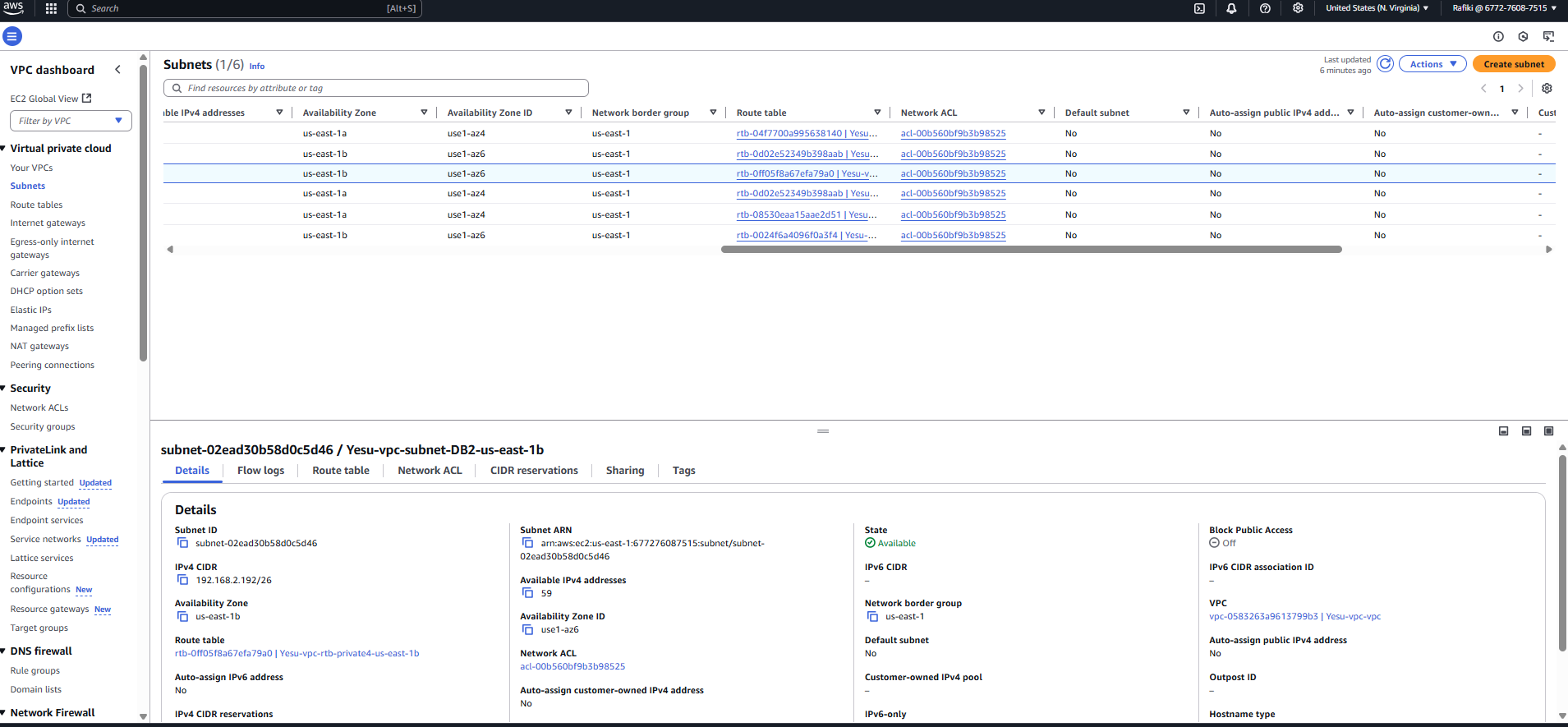


Customize Subnets

Here we are going to edit the names of the subnets to fit their function.

The public subnets will be used for Web layer because we need web connectivity for them.

The private subnets for App layer and Database layer



-Security groups

Next we are going to create security groups. Total five security groups.

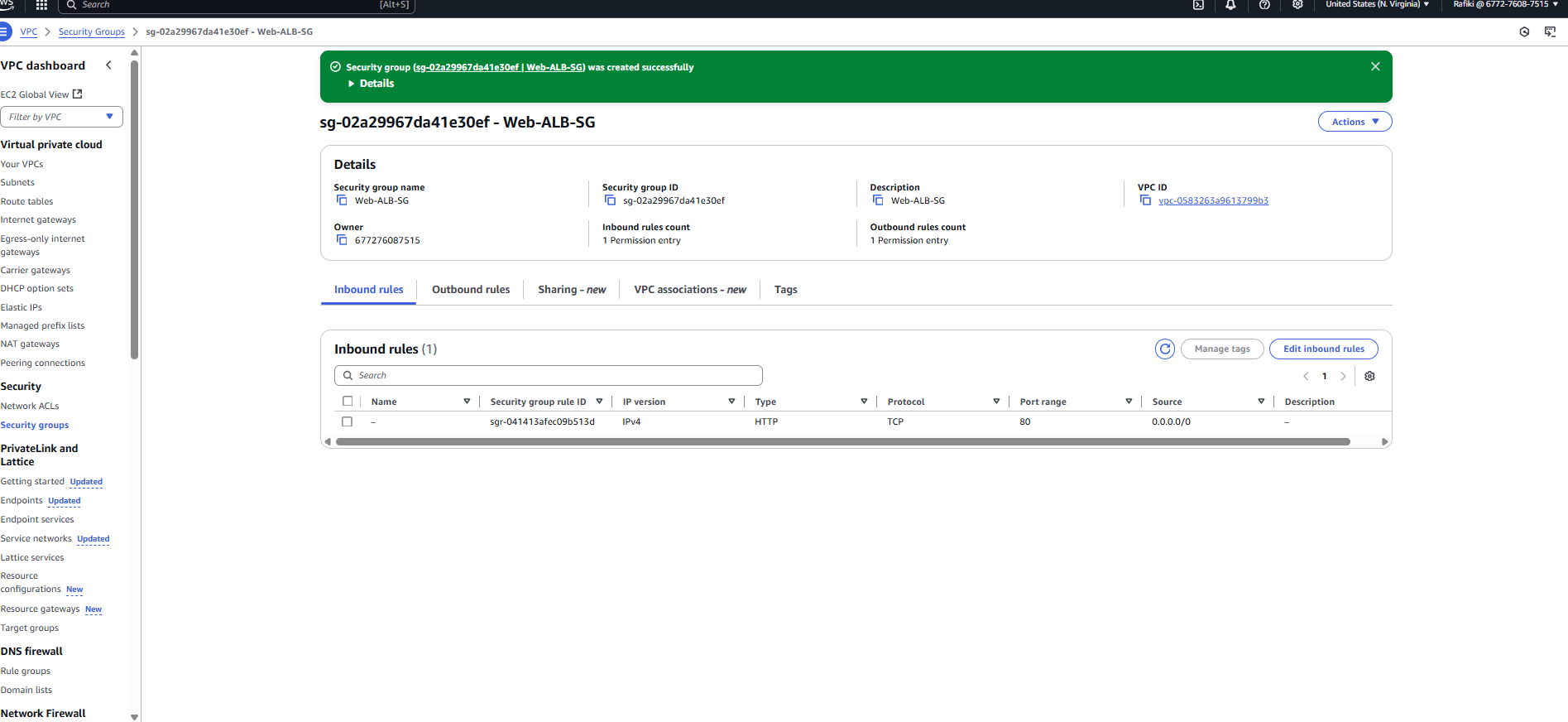
-One for web servers, another for internet facing load balancer( external load balancer) , one for app servers and another one for database servers.

1. Create Security group name -- As Web-ALB-SG , description the same then attache the vpc we created i.e Yesu-vpc

Inbound traffic

http traffic – protocol TCP port range 80 , source type Anywhere –IPV4 ( that means open it to anybody) .

Outbound rules remain the same. Create security group



2)

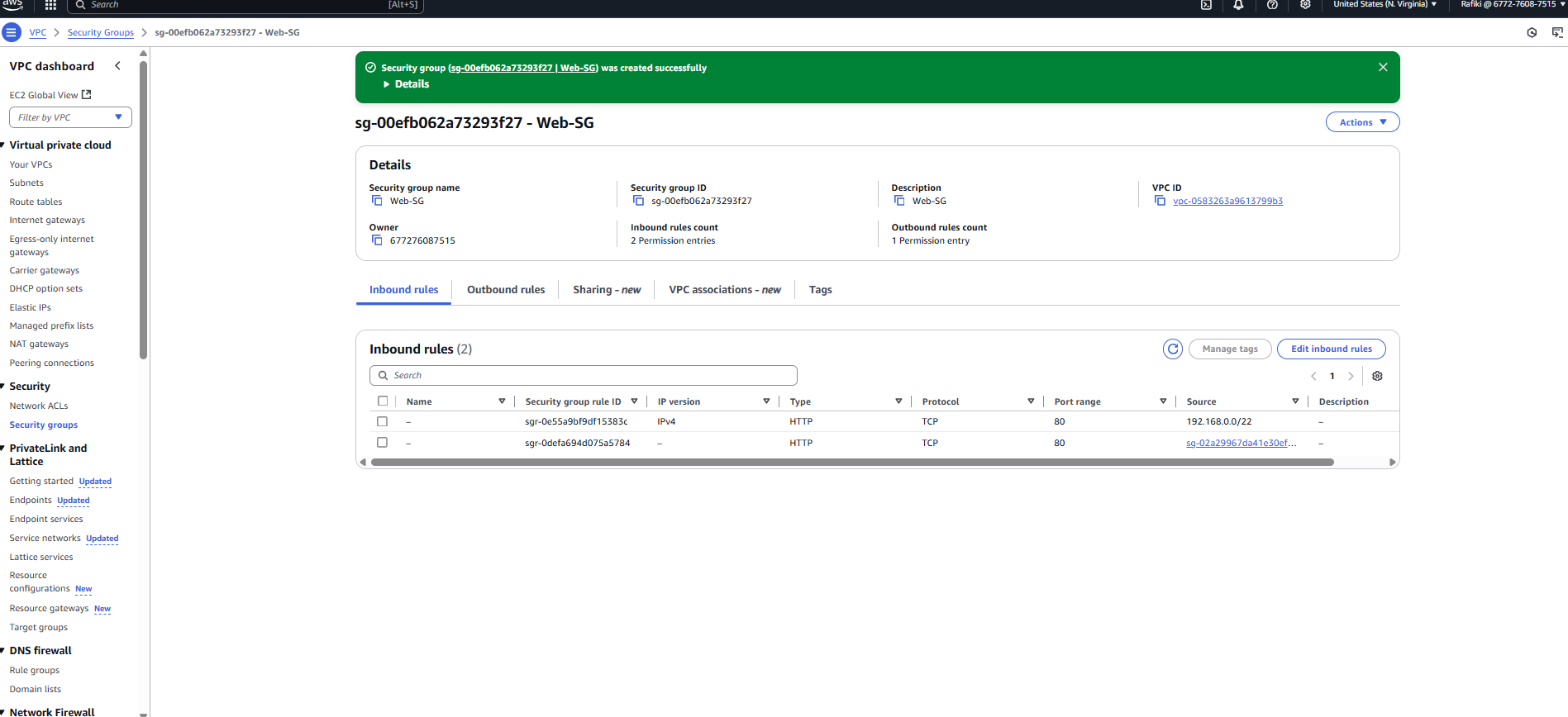
Next security group for the web tier

-Name it Web-SG ,description the same , attached to Yesu-vpc .

-Add inbound rule as HTTP , protocol TCP port 80 . However , source should be custom and attached to Web-ALB-SG security group we created earlier.

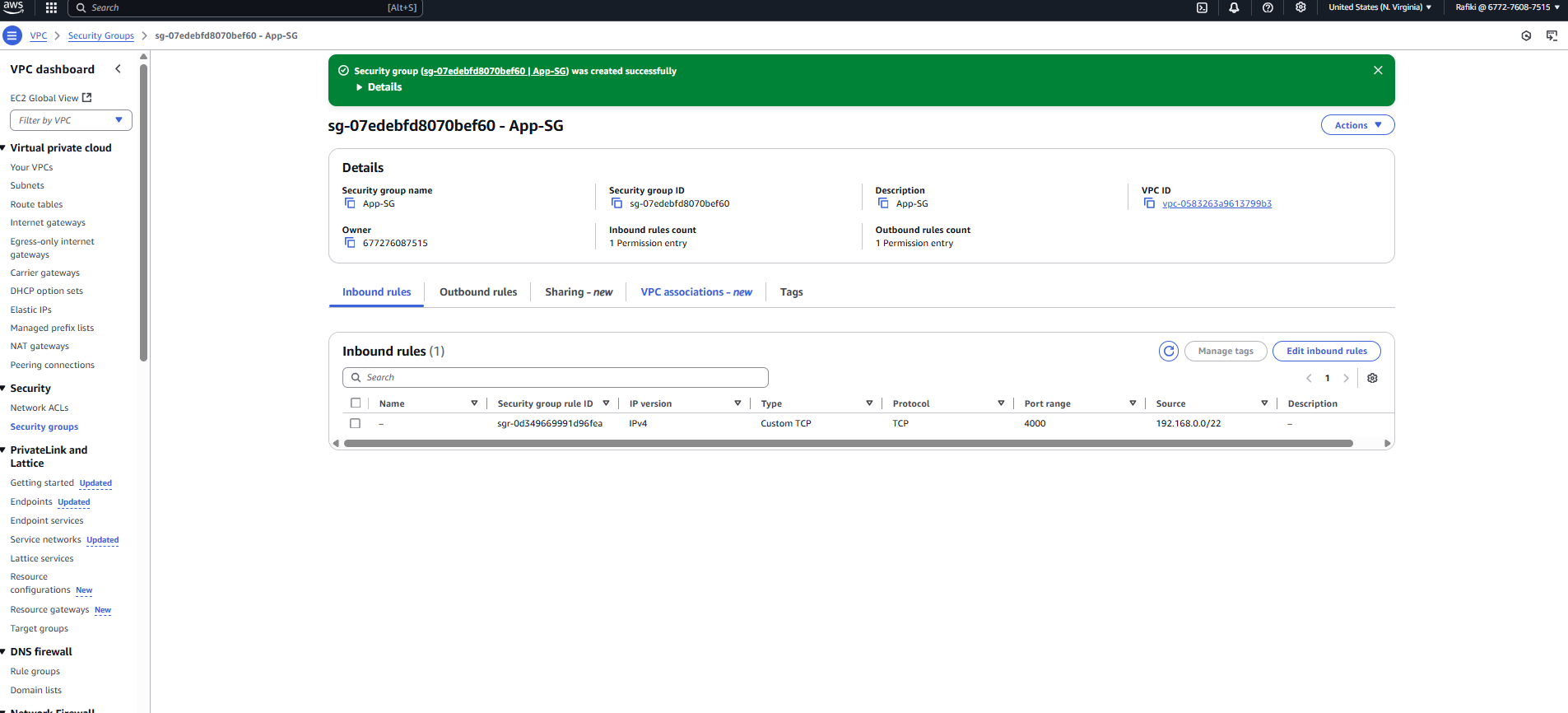
-In addition , we can create another rule with HTTP traffic, protocol TCP , port 80 , custom IP 192.168.0.0/22

IF you recall we used the IP address 192.168.0.0/22 to create vpc CIDR block range.



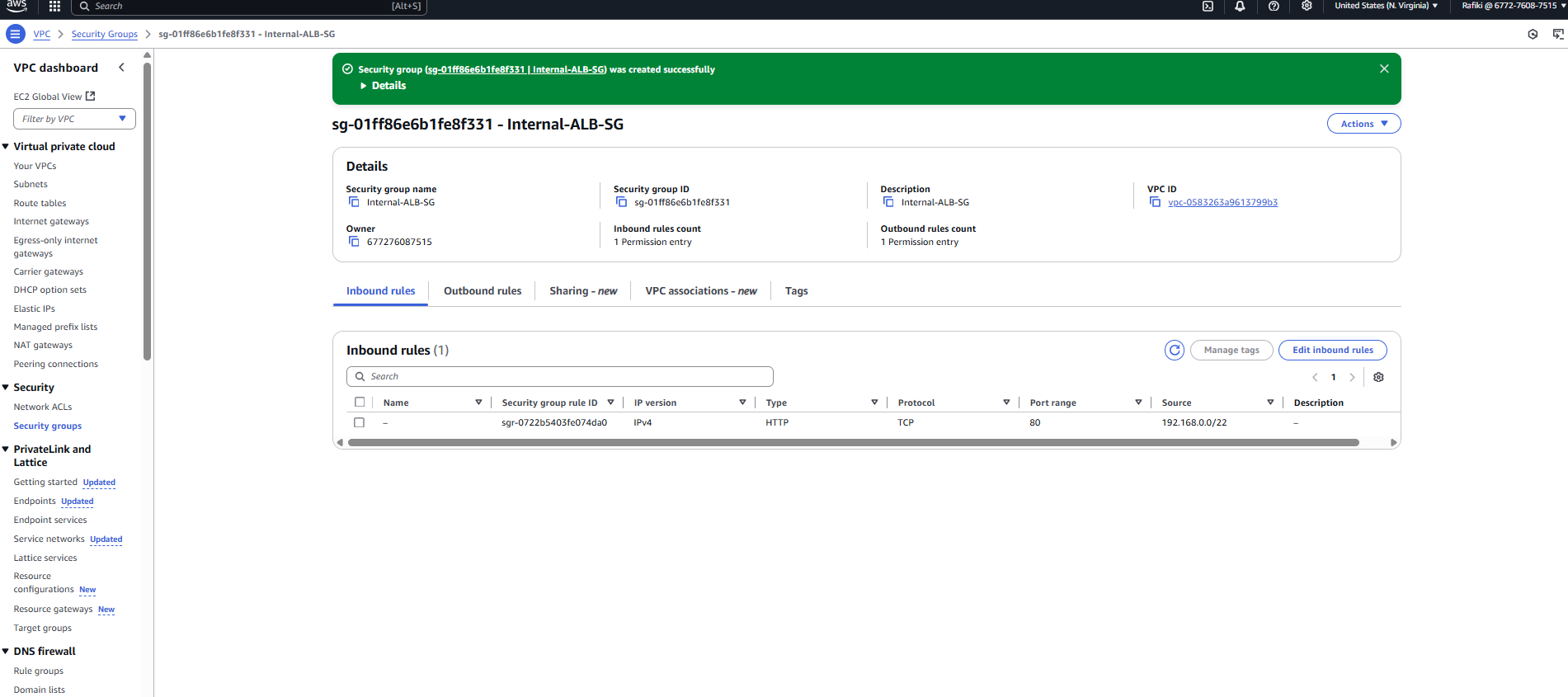
3)Now we need to create security group for private Apps running in the app server. Create security group for the App layer or App tier.

\* React- js based application run Custom traffic TCP , protocol TCP on port range 4000 in the inbound rule and attach to vpc CDIR block 192.168.0.0/22.



4) Create security group for internal load balancer . Name it internal-ALB-SG. Next attach our created vpc Yesu-vpc.

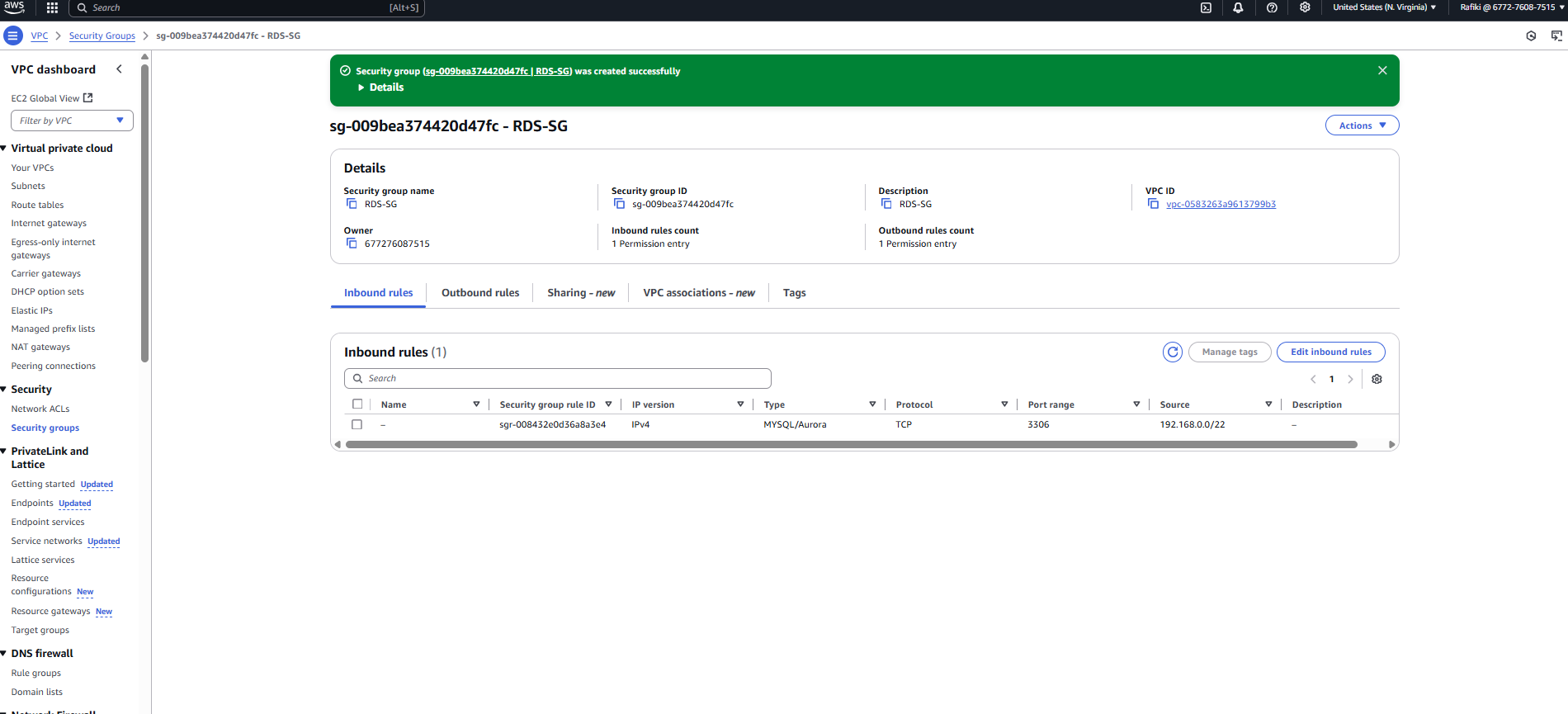
Here we open HTTP traffic, TCP protocol on port 80 , destination custom to 192.168.0.0/22 (vpc CDIR block)



5)

Next create security group for the database . Name it RDS-SG , attached to the vpc I created named Yesu-vpc

Now click on the inbound rule. Here for the traffic is going to be Mysql/aurora . The protocol is TCP and it runs of default port 3306 , source custom is custom network at IP 192.168.0.0/22. Recall this is vpc CIDR block. The resources communicating with the database will be restricted to the private subnets in the App tier.



Note that Web-ALB-SG in internet facing if you examine the inbound rules. We have given the source as anywhere meaning it is public facing network. Anybody from outside can access the application which is running in the web tier.

II

1. The second step is to create the s3 bucket. Reason we need to upload the our application code in the S3 bucket. Why ? To avoid the complexity of uploading the application code directly to our EC2 instances.

The source of our application code may be found in the Github repository using the link provided below.

<https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code>

In my case I cloned the application code on my local machine via git bash. As follows below step by step

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

$ git clone https://github.com/kibush/3TierAWSArchitectureApp.git

Cloning into '3TierAWSArchitectureApp'...

remote: Enumerating objects: 93, done.

remote: Counting objects: 100% (93/93), done.

remote: Compressing objects: 100% (83/83), done.

Receiving objects: 50% (47/93)used 0 (delta 0), pack-reused 0 (from 0)

Receiving objects: 100% (93/93), 369.10 KiB | 14.20 MiB/s, done.

Resolving deltas: 100% (19/19), done.

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

$ ls

3TierAWSArchitectureApp/

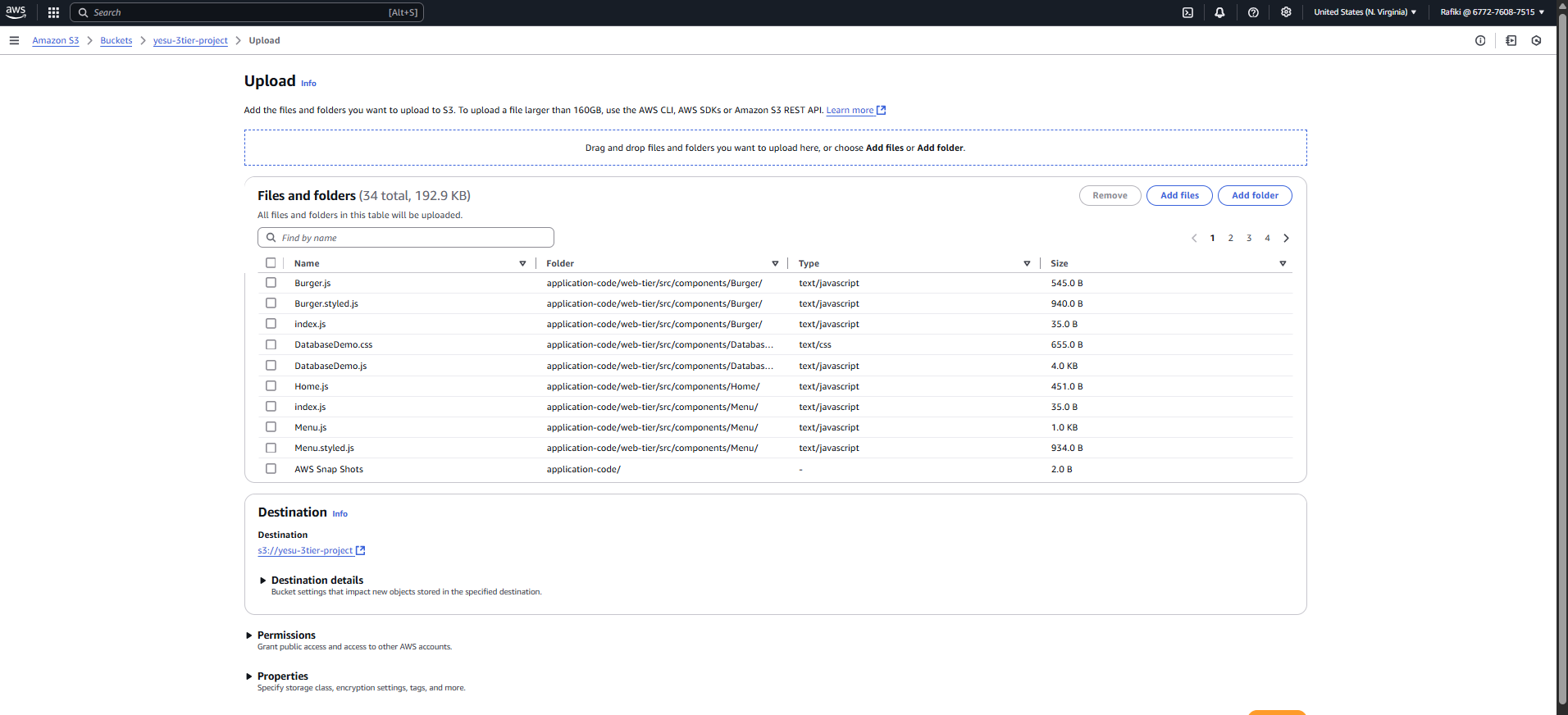
Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

2)

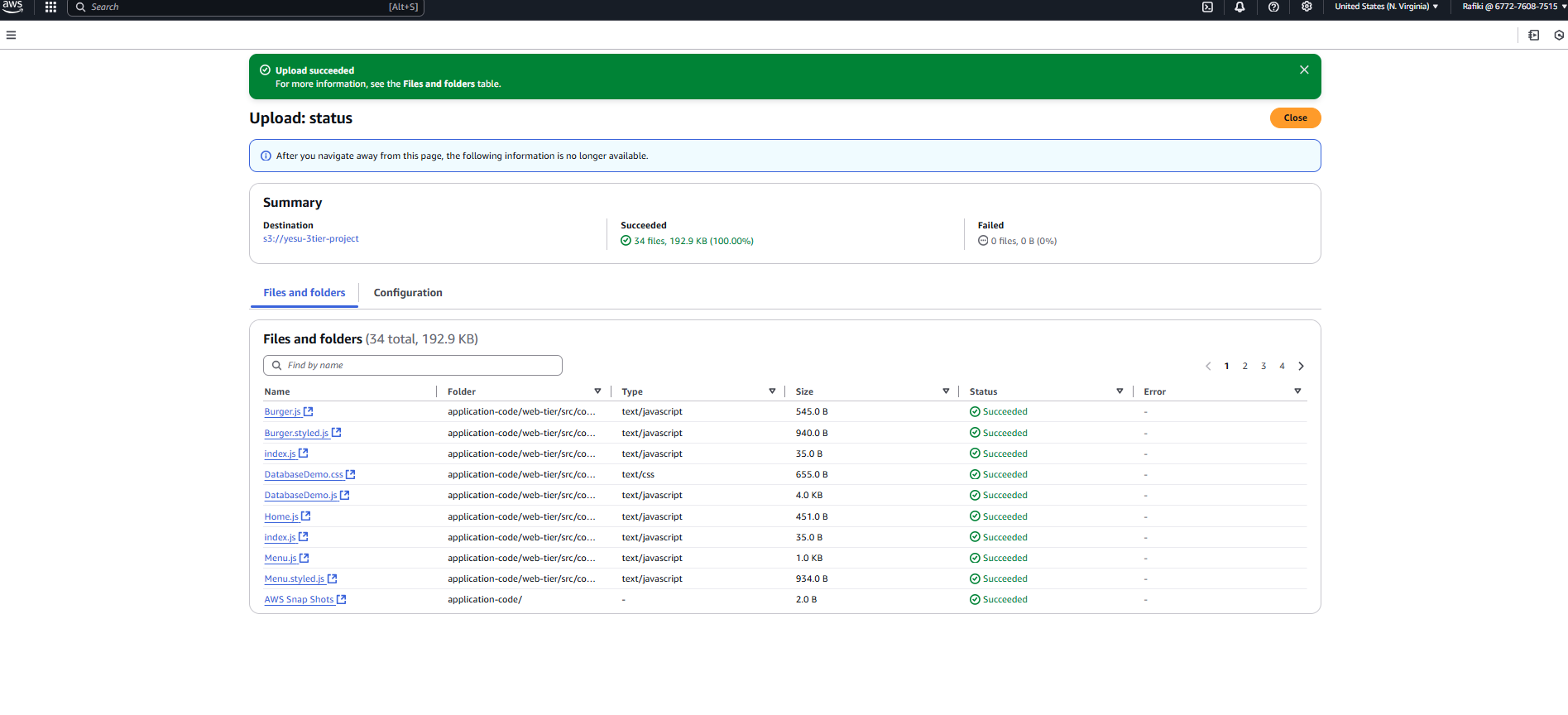
Here I am going upload the entire application folder into the s3 bucket on AWS. Type s3 in the search window to find the s3 service ( Scalable storage in the Cloud). Here a bucket with the default settings.

Note it is important to remember the region of your service . In my case it is US East ( N. Virginia) US-east-1. We will call out bucket name yesu-3tier-project.

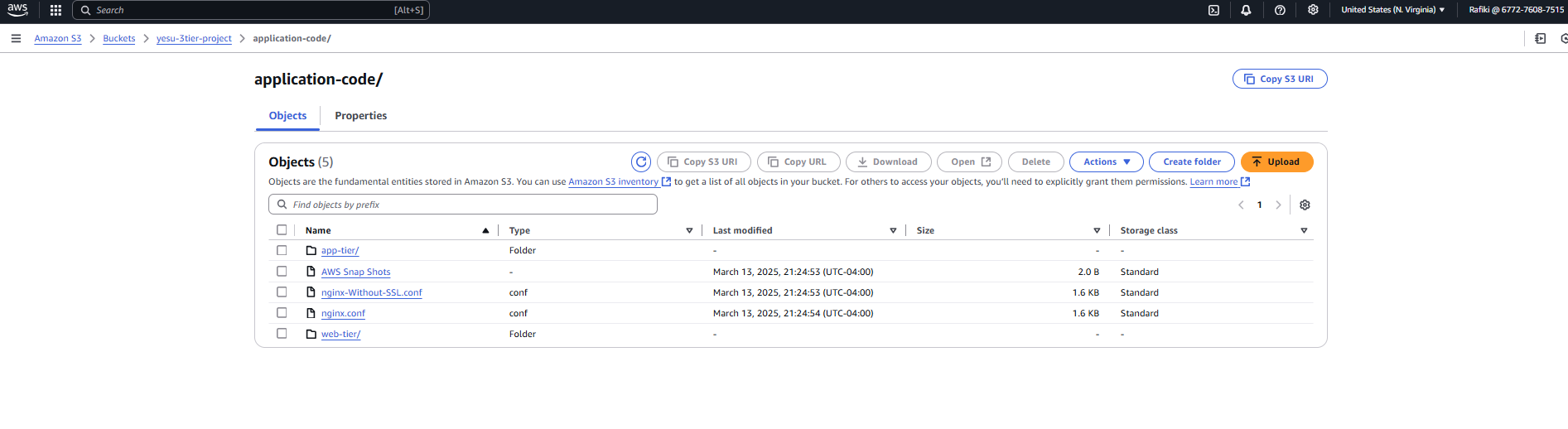
Keep all settings in default mode . Now upload only application code folder. In the bucket you can drag and drop folder directly in the s3 bucket from the local machine.



Click the up load file once you drag and drop file.



Here you can view the application code folder uploaded to s3 bucket.

-

3) Next lets create an IAM role . What is the reason for creating IAM role. Remember we have two EC2 instances in the App tier. In addition they are in the private subnet . Also , the database is also unavailable in the private subnet. It is important to not give these 2 services access to outsiders. So , we are going to close any vulnerability. There may be 2 ways to connect to these two resources i.e. EC2 instances and the RDS. We can use the bastion host which the concept in vpc. That means we create a bastion host in the public subnet and by using the bastion host we will be able to connect to resources in the private subnet. The second way is instead of creating a bastion host an the complexities that come with it we can instead use the AWS service called the systems manager called SSM. That means if we attaché the appropriate role to EC2 instance which are there in the private subnet. Even though the machines are there in the private subnet we access this virtual machines using the SSM agent. So, in our case we use the IAM role to be able to connect to these private layer or tier.

To, define our role we are going to create a policy and attach Amazon EC2 role for SSM.