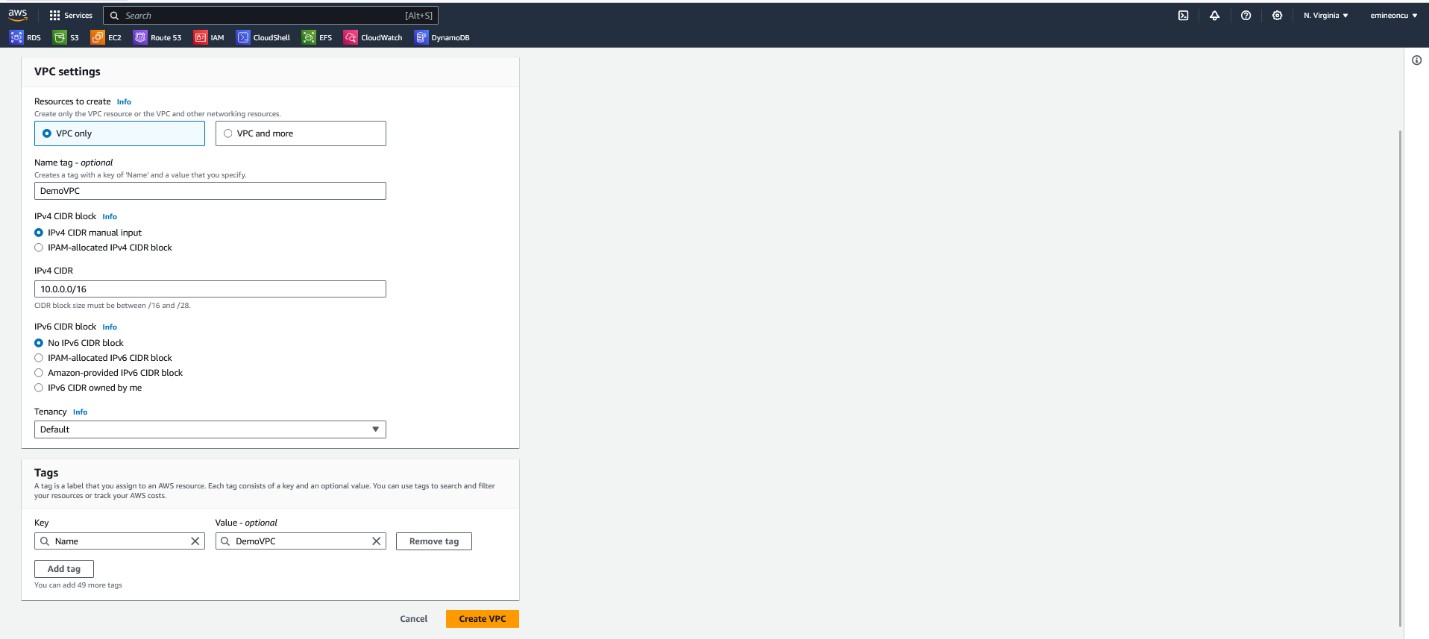
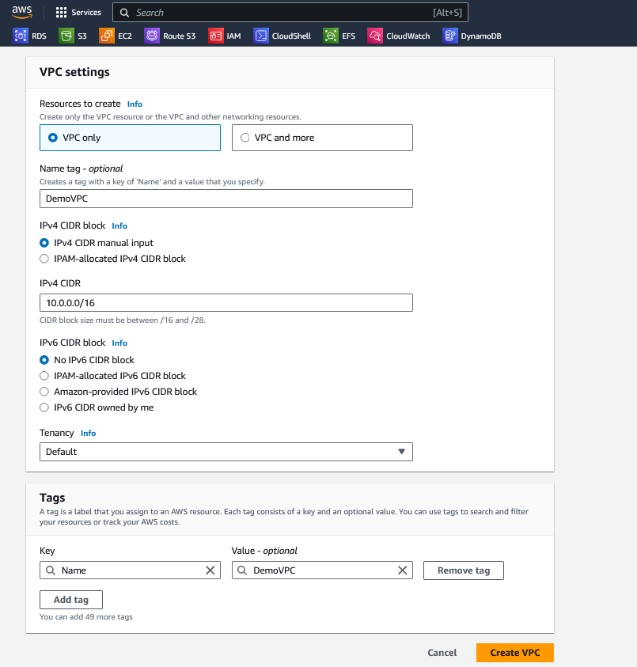
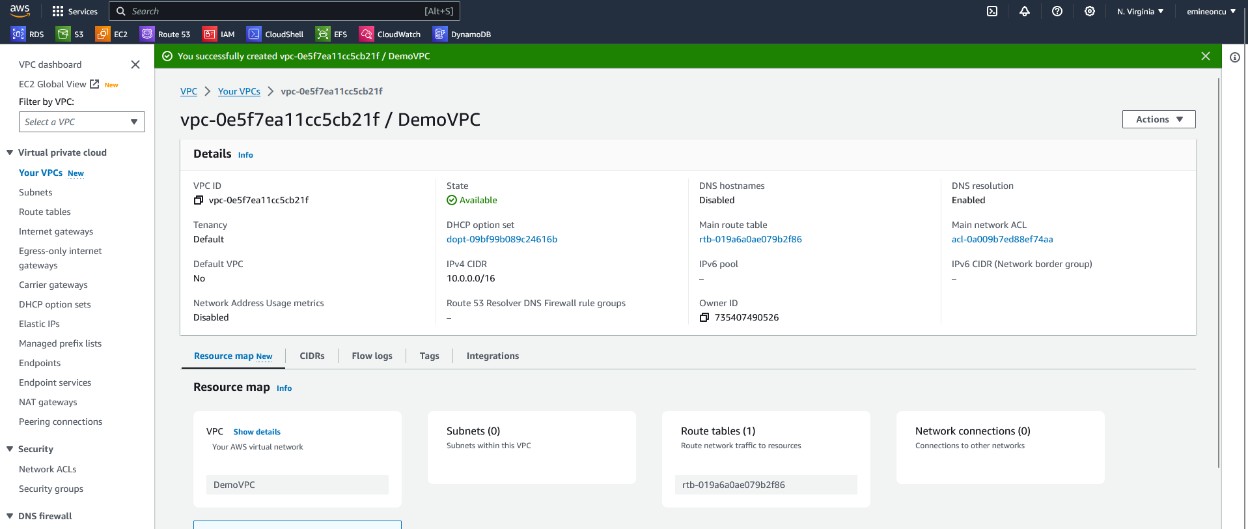
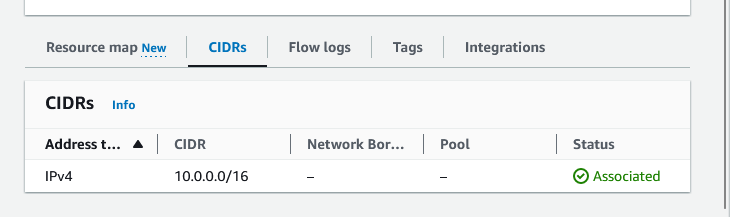
# VPC / VPC Settings /Create a VPC

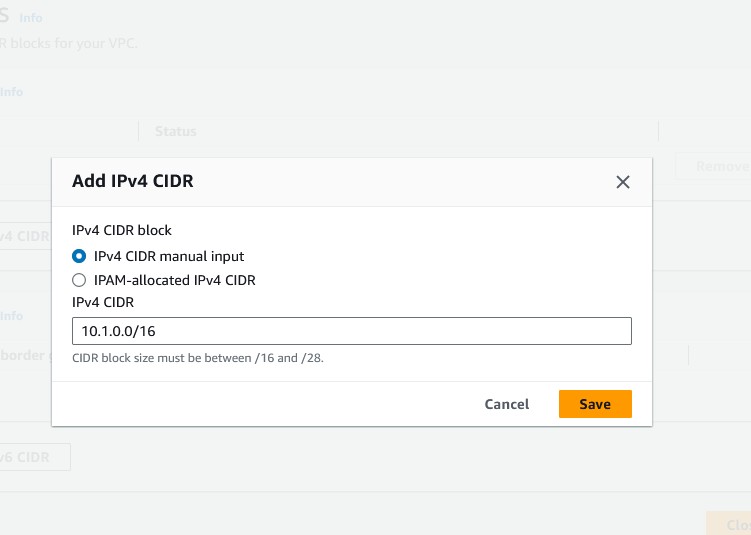


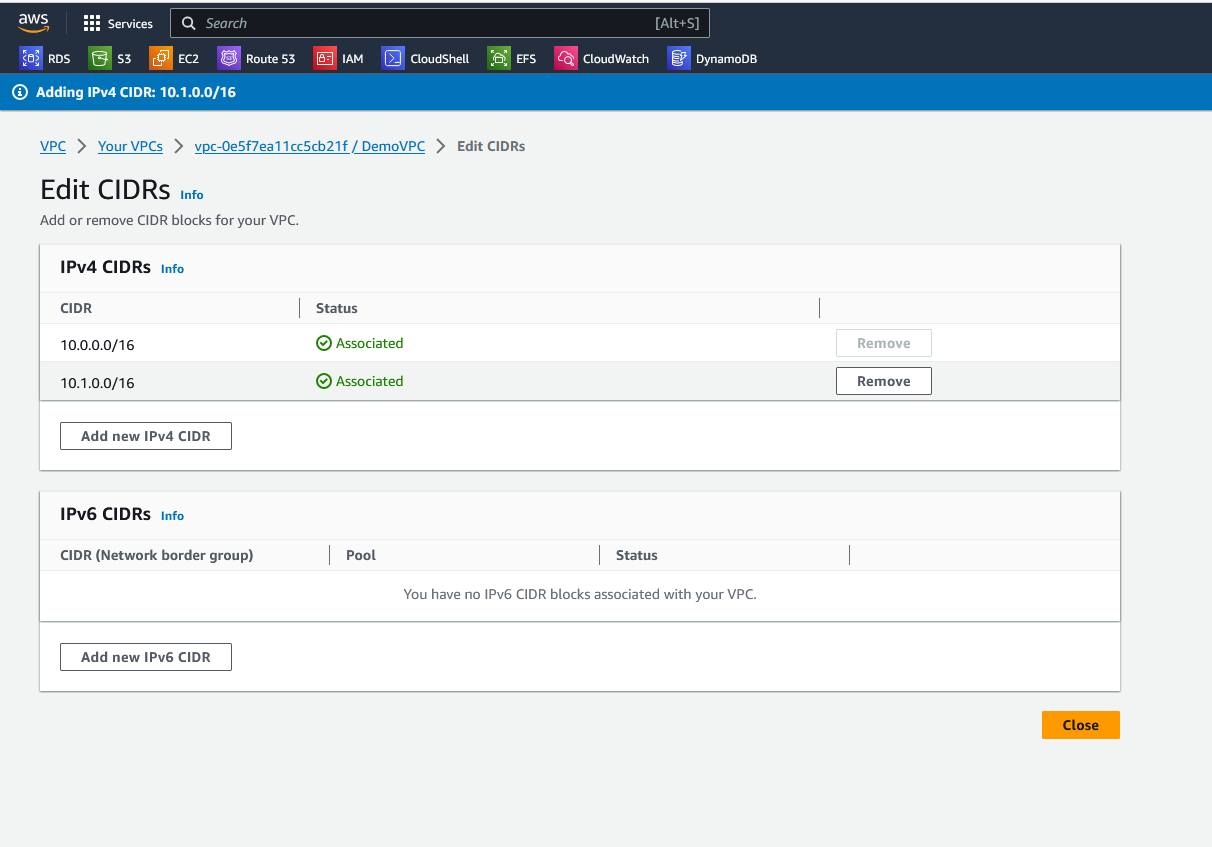


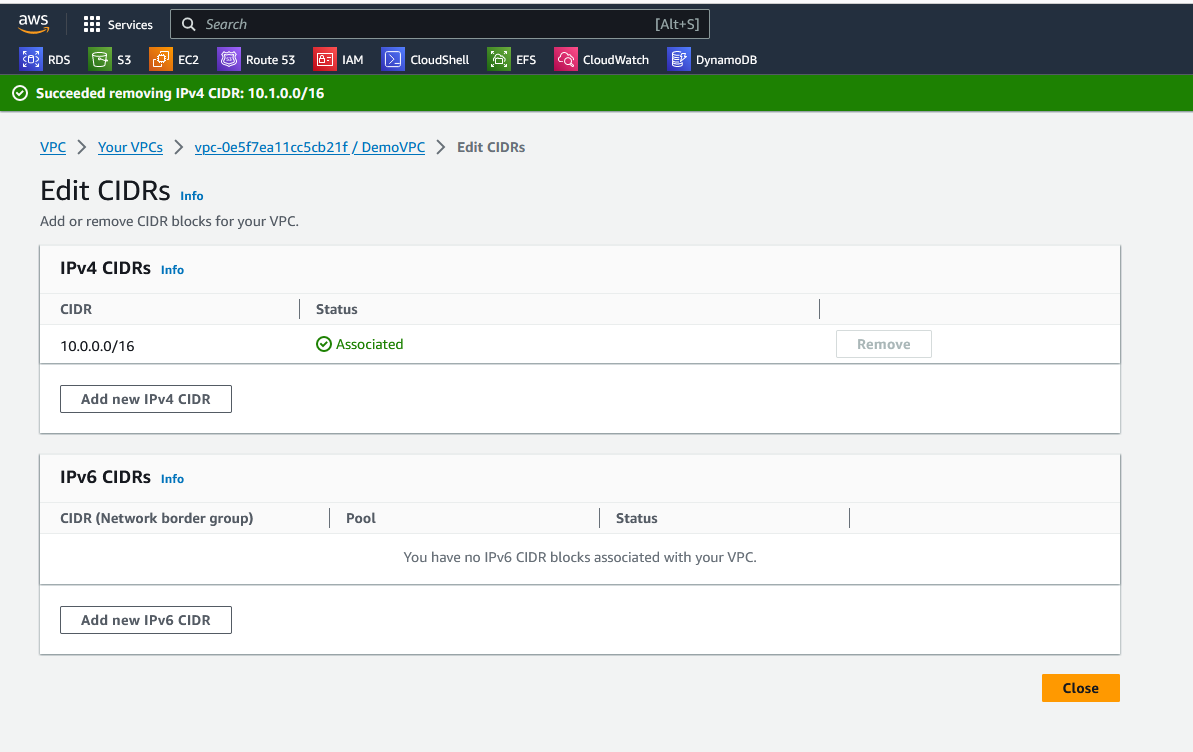


* + **We can add up to 5 IPV4 CIDRs for your VPC.**

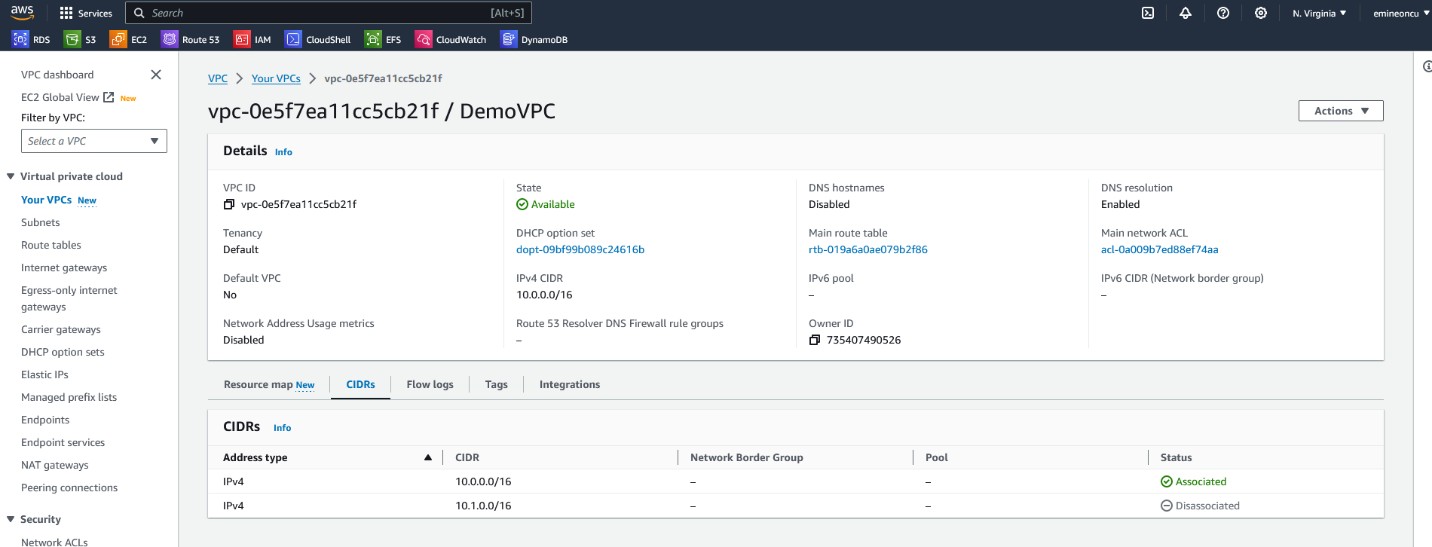


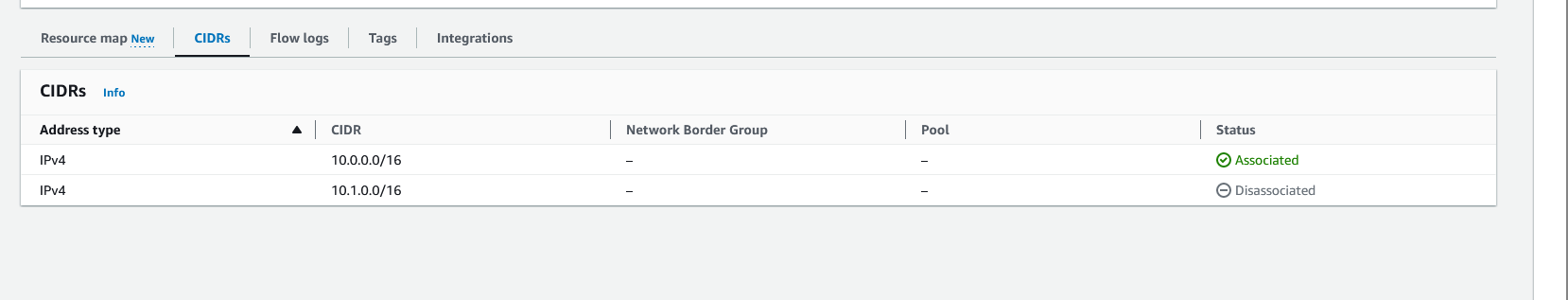




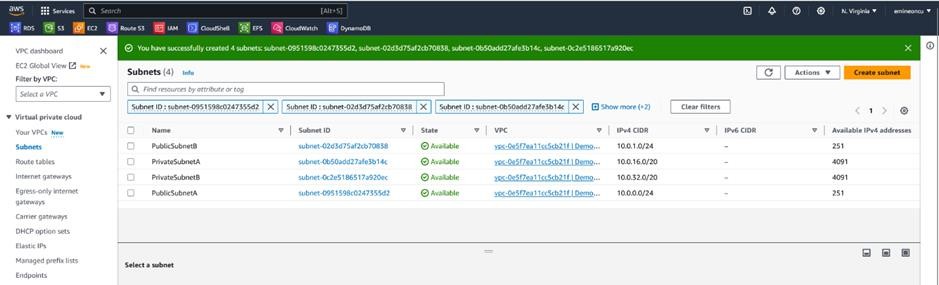


# Removed extra CIDR.



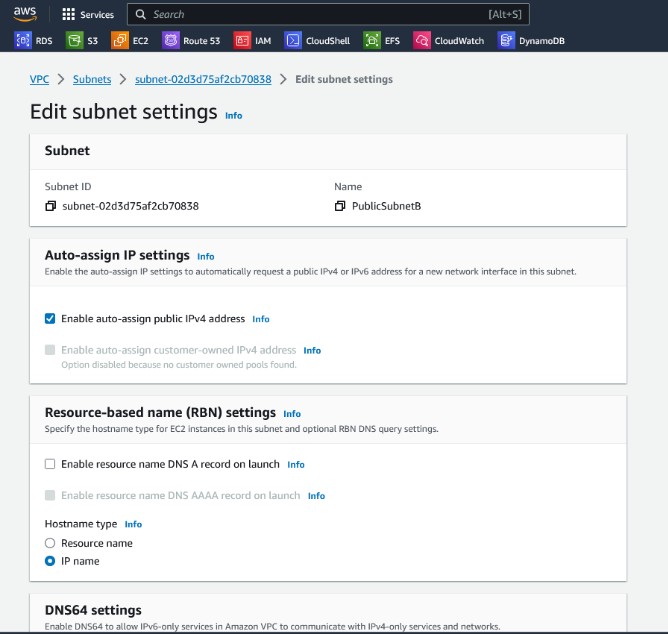
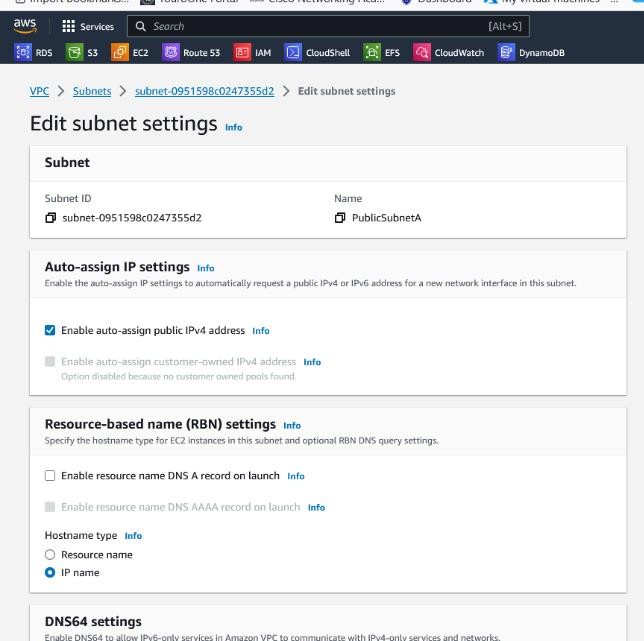


* + **Creating 2 Public and 2 Private Subnets**

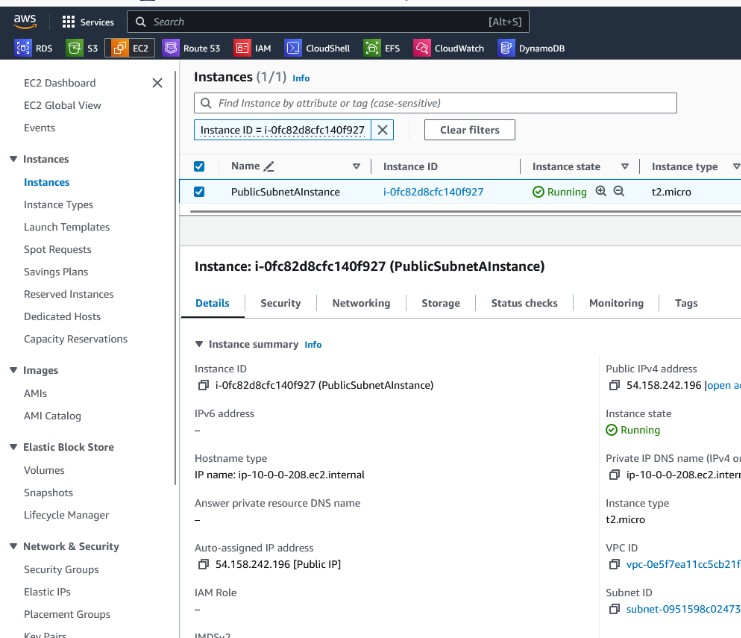


To make SubnetA public, we need to do a couple of steps

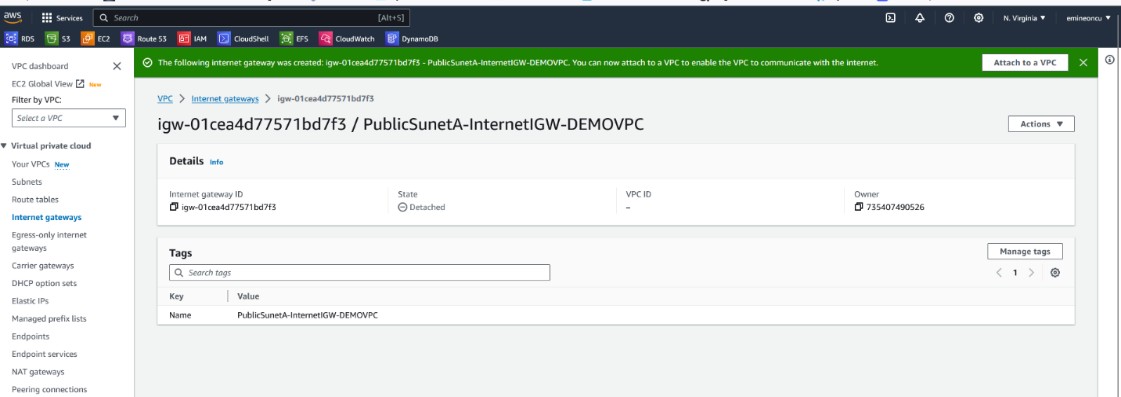
* + - **Enable Public IPV4 address for it**



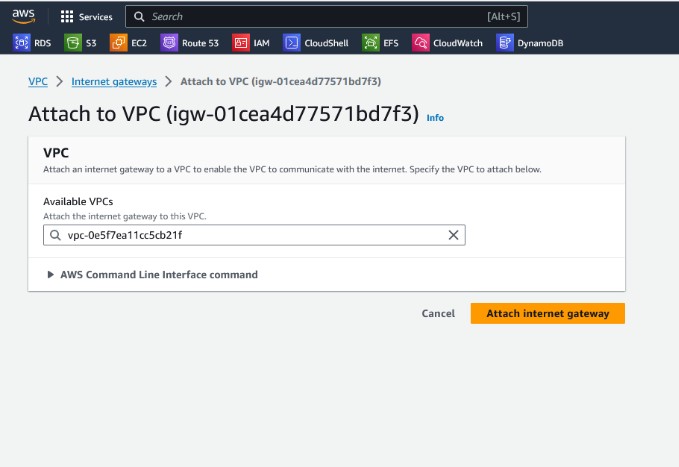
* + - **We create an EC2 instance in our Public Subnet within the DemoVPC.**



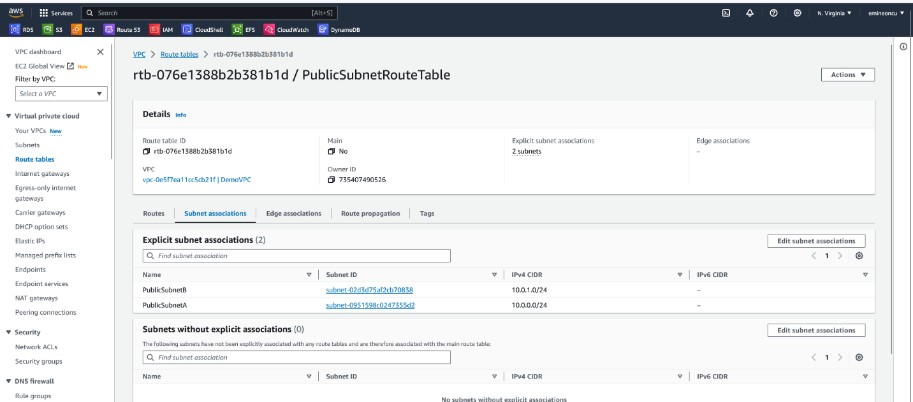
# Create an Internet Gateway



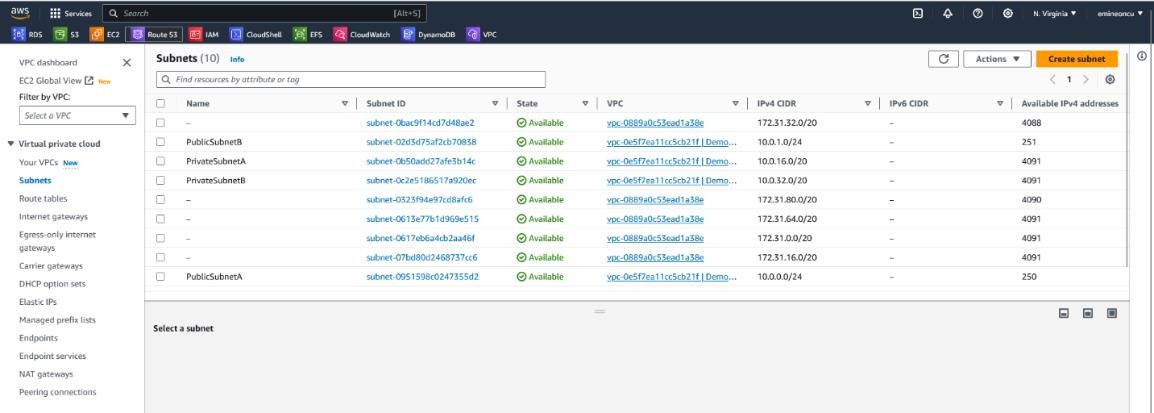
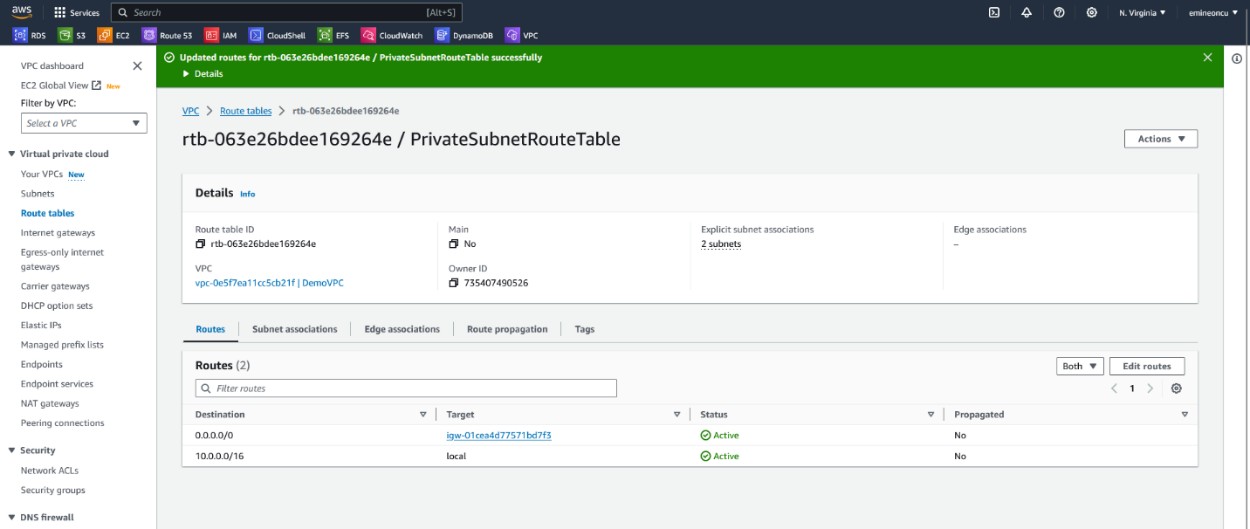
* **Attached the IGW to the DemoVPC**



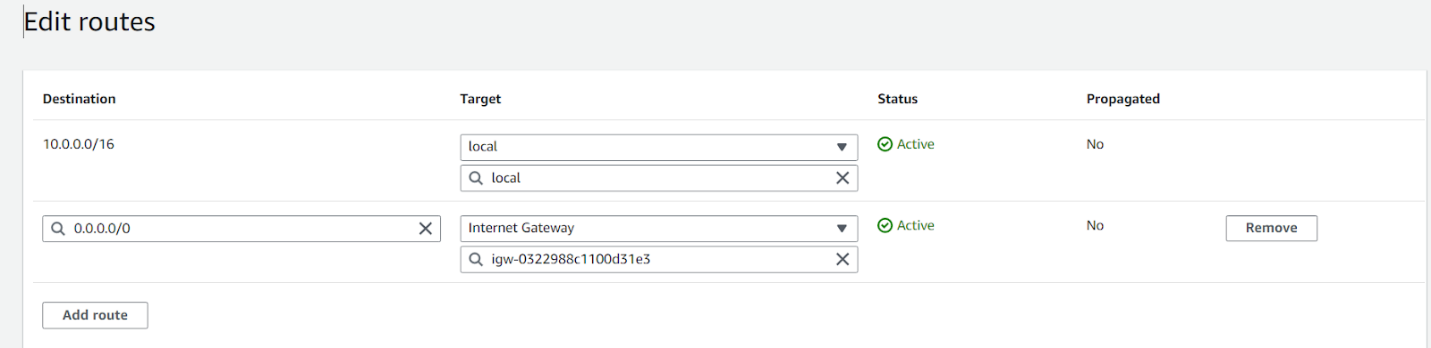
* **We now associate our Public Subnet Route Table with Public Subnets**

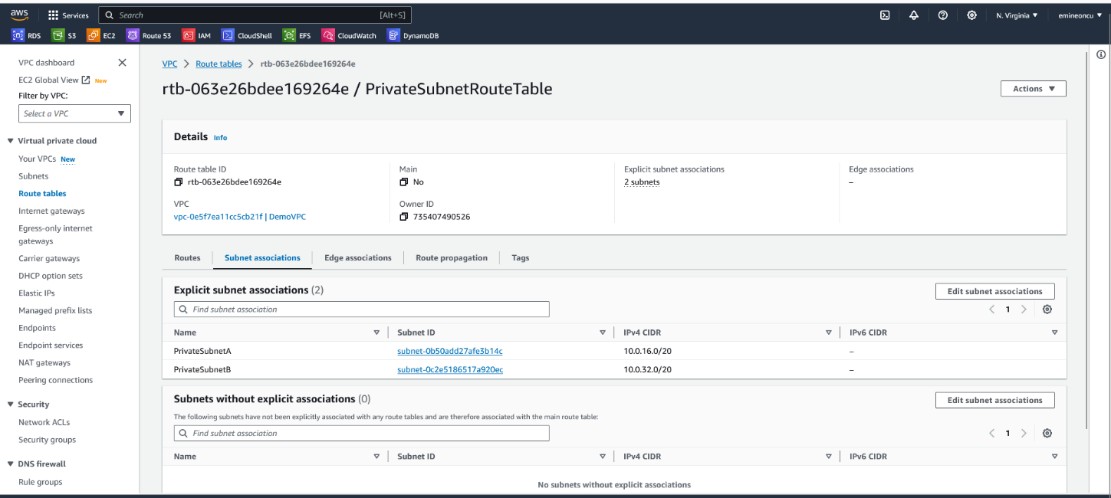


* **We associate the Private Subnet Route Table the same way.**

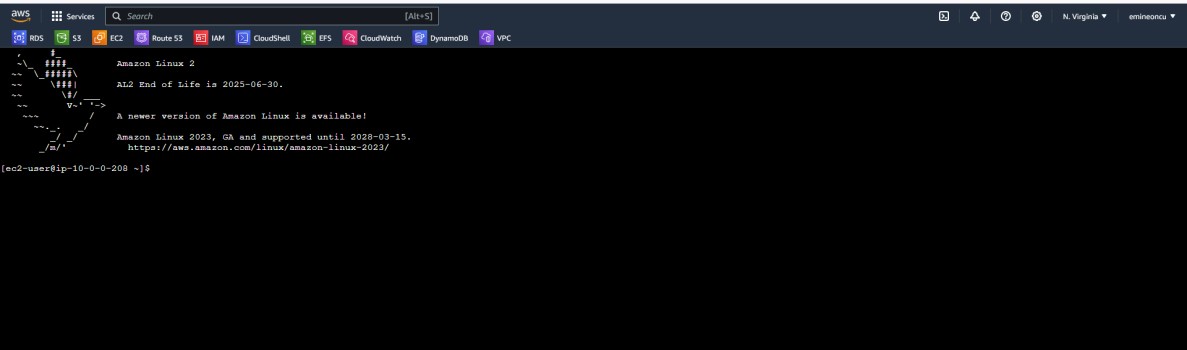


* **Both the public subnets will now have access to the internet via the Internet Gateway.**

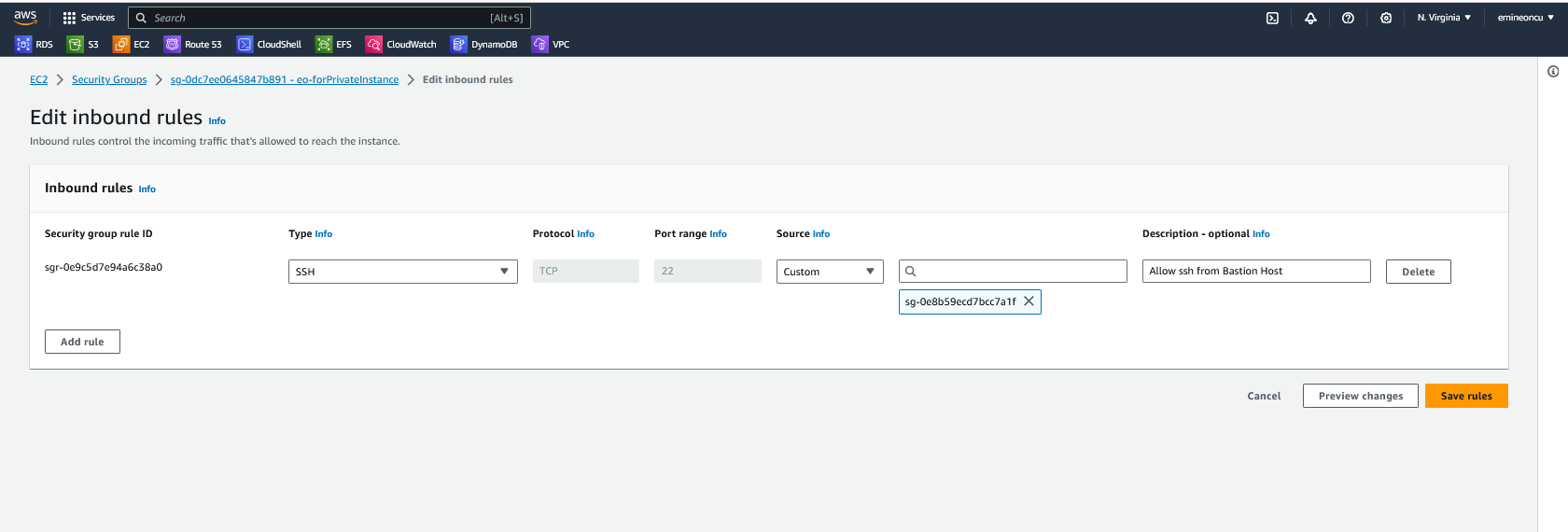


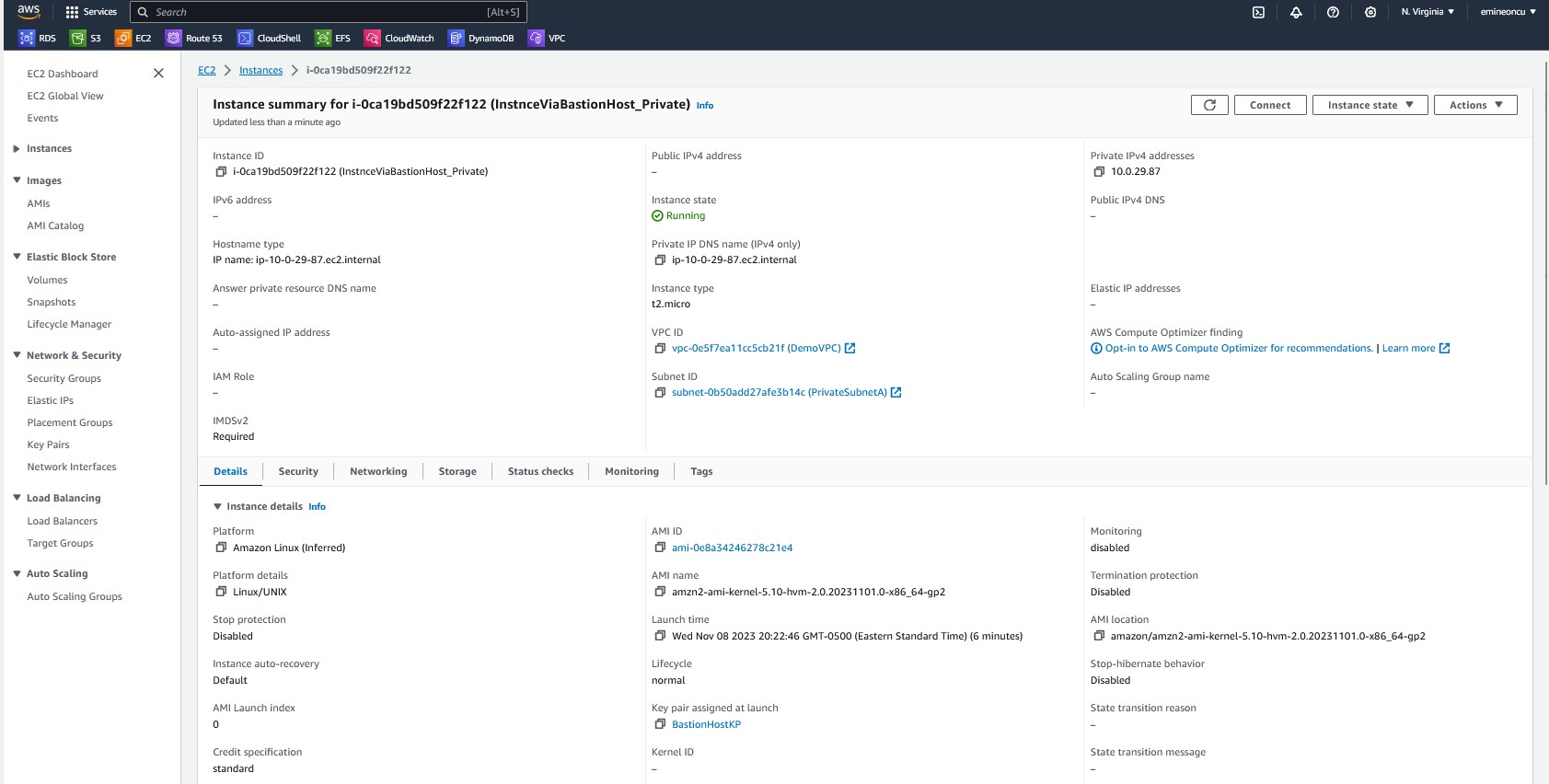


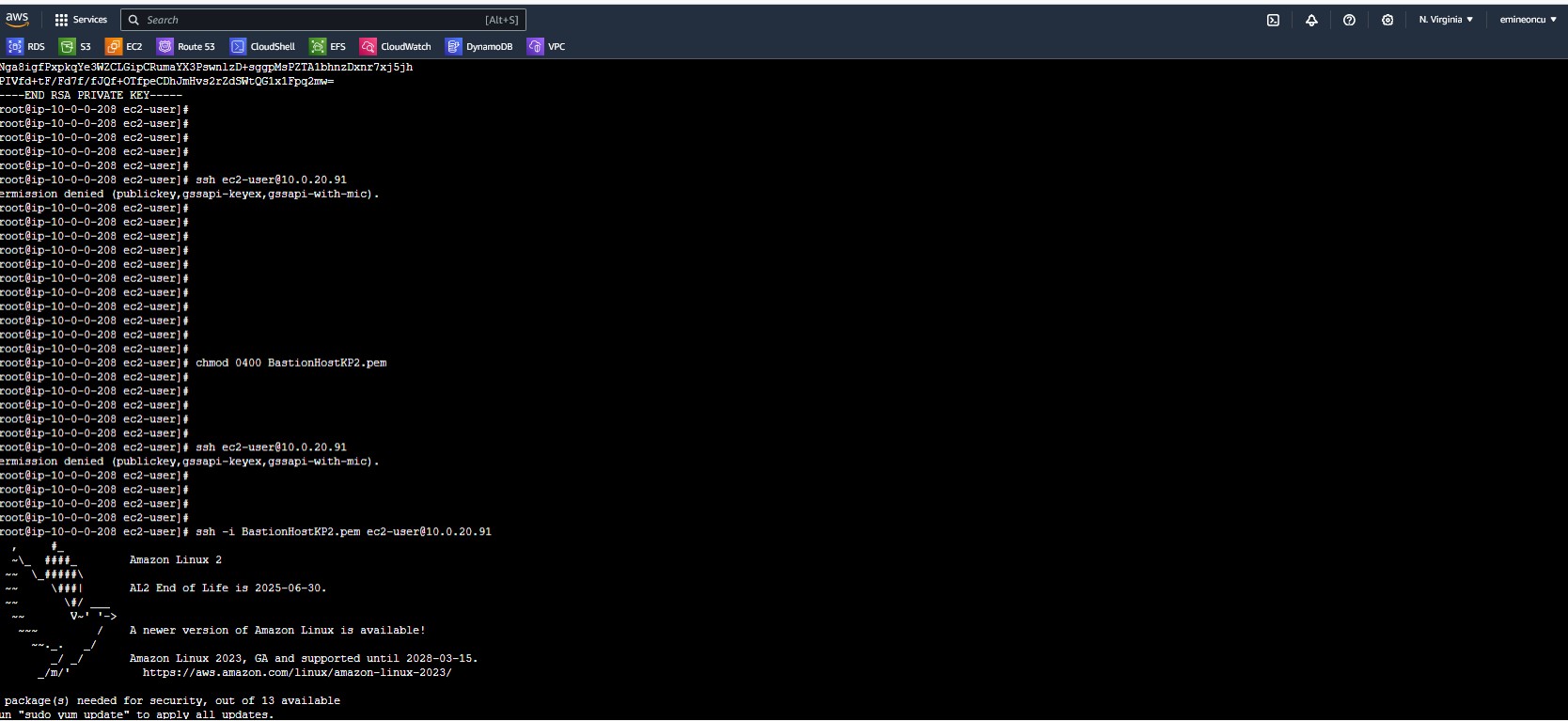
* **EC2 instance can now connect to the internet.**

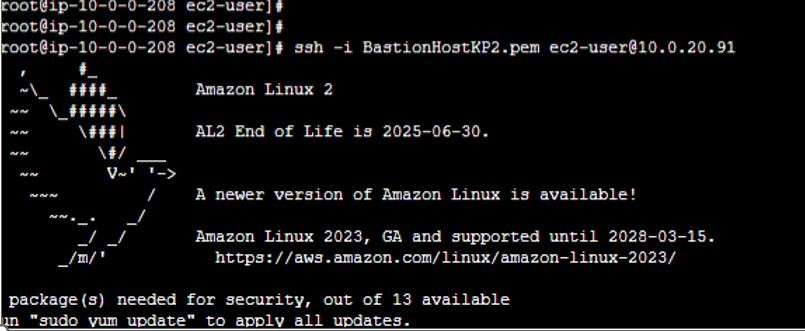


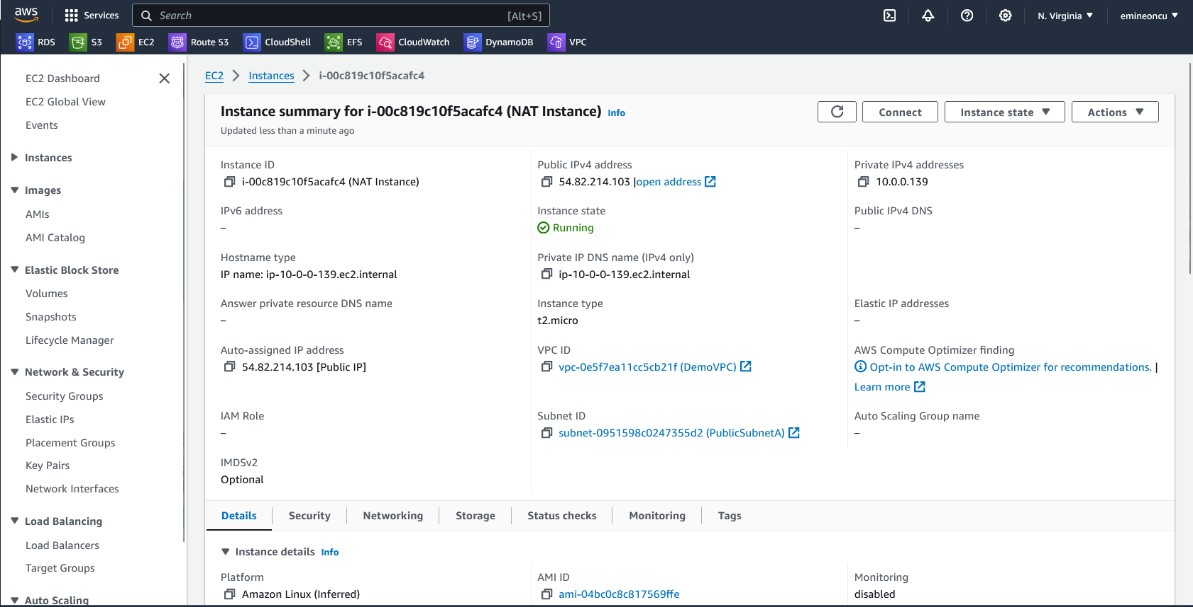
* **Created Bastion Host to provide SSH access to the EC2 instance in a private subnet. Only accessible via the security group of Bastion Host public instance.**

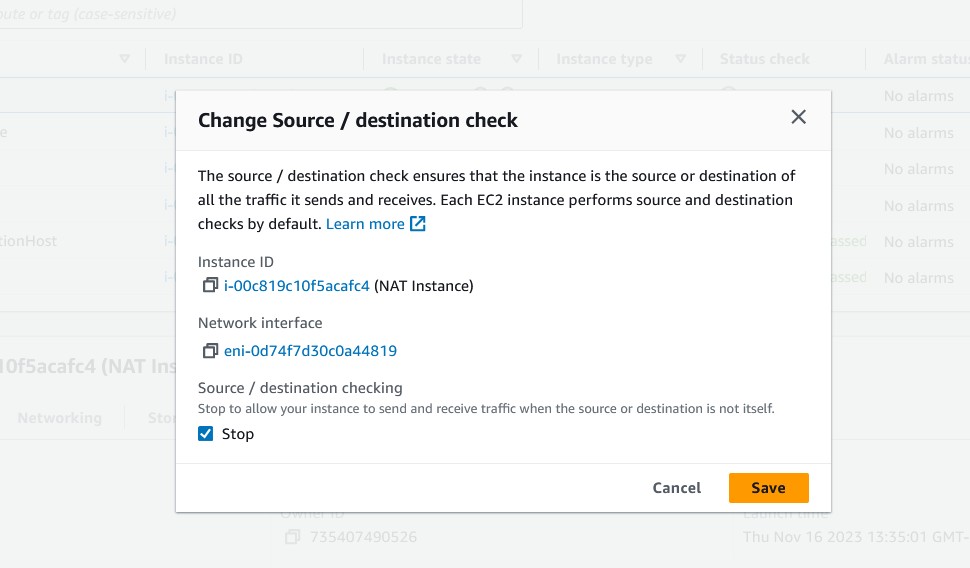






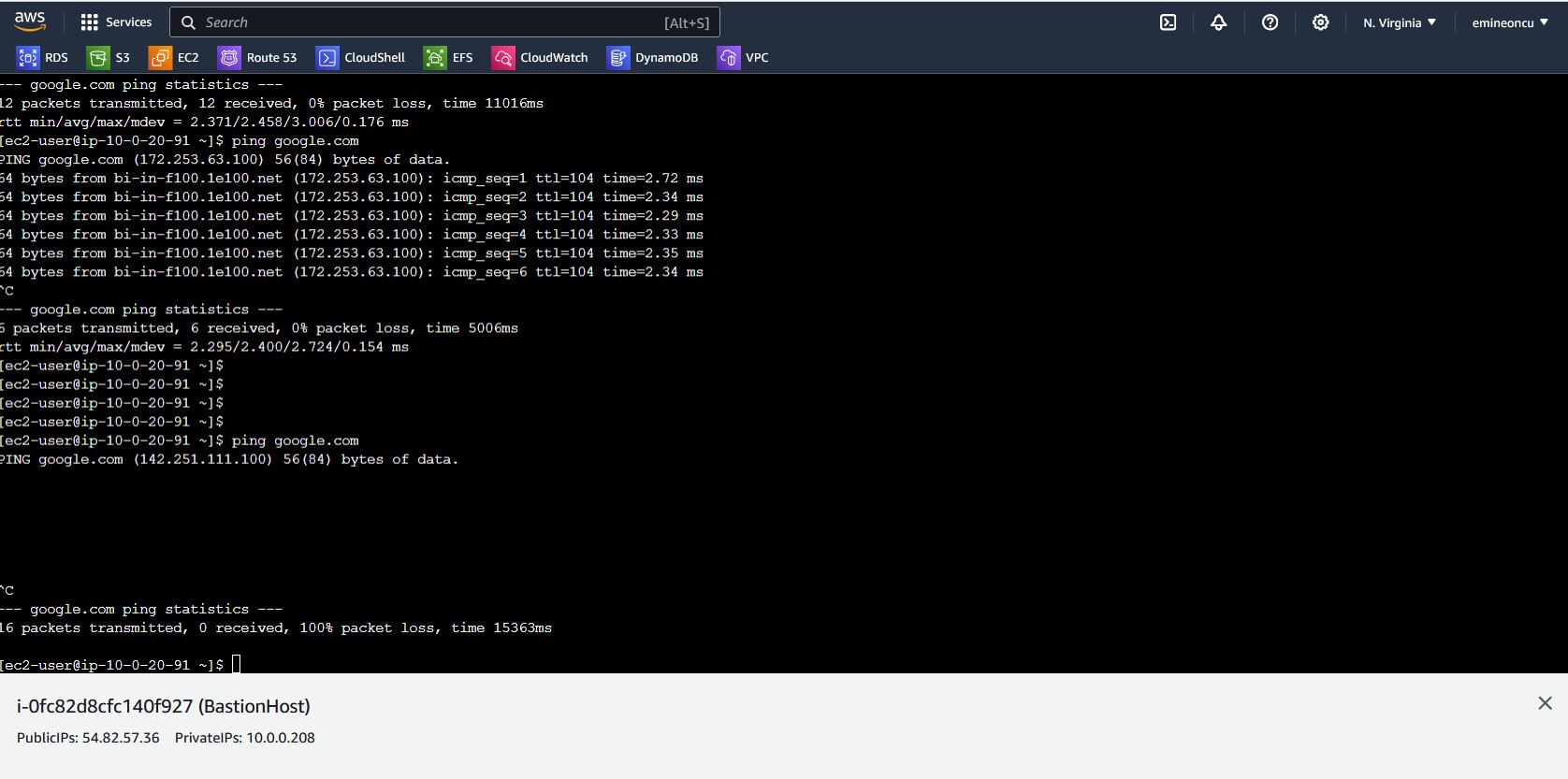


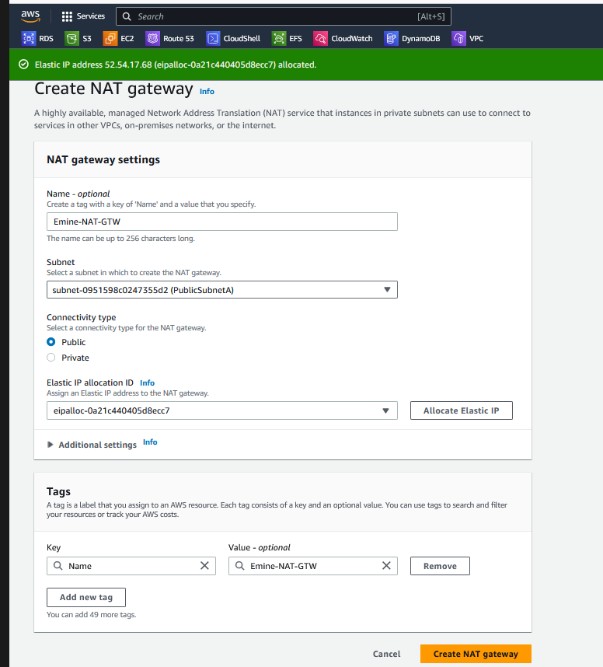
* **Able to SSH into the Private EC2 instance via Bastion Host. Creating a NAT Instance**

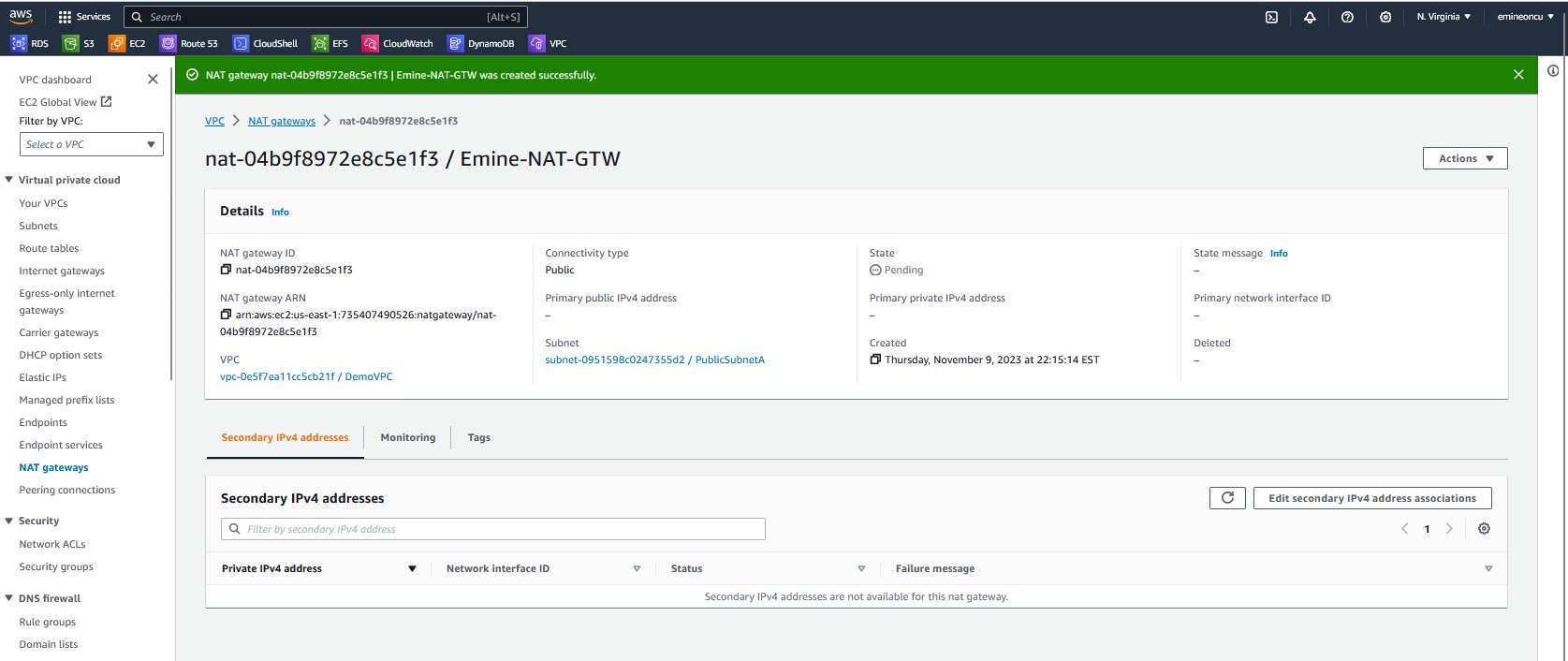


* **Accessed Private Instance via Bastion Host and then our route back to the outside world was**

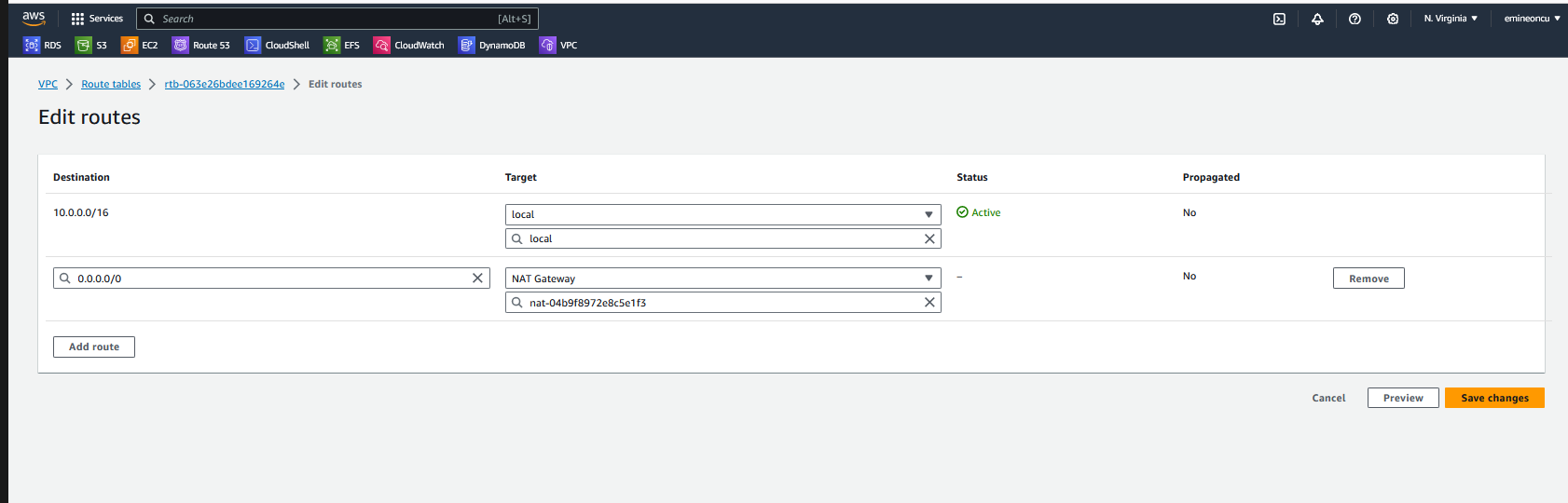
routed back using NAT Instance. The ICMP protocol for “ping” was enabled on the NAT Instance.

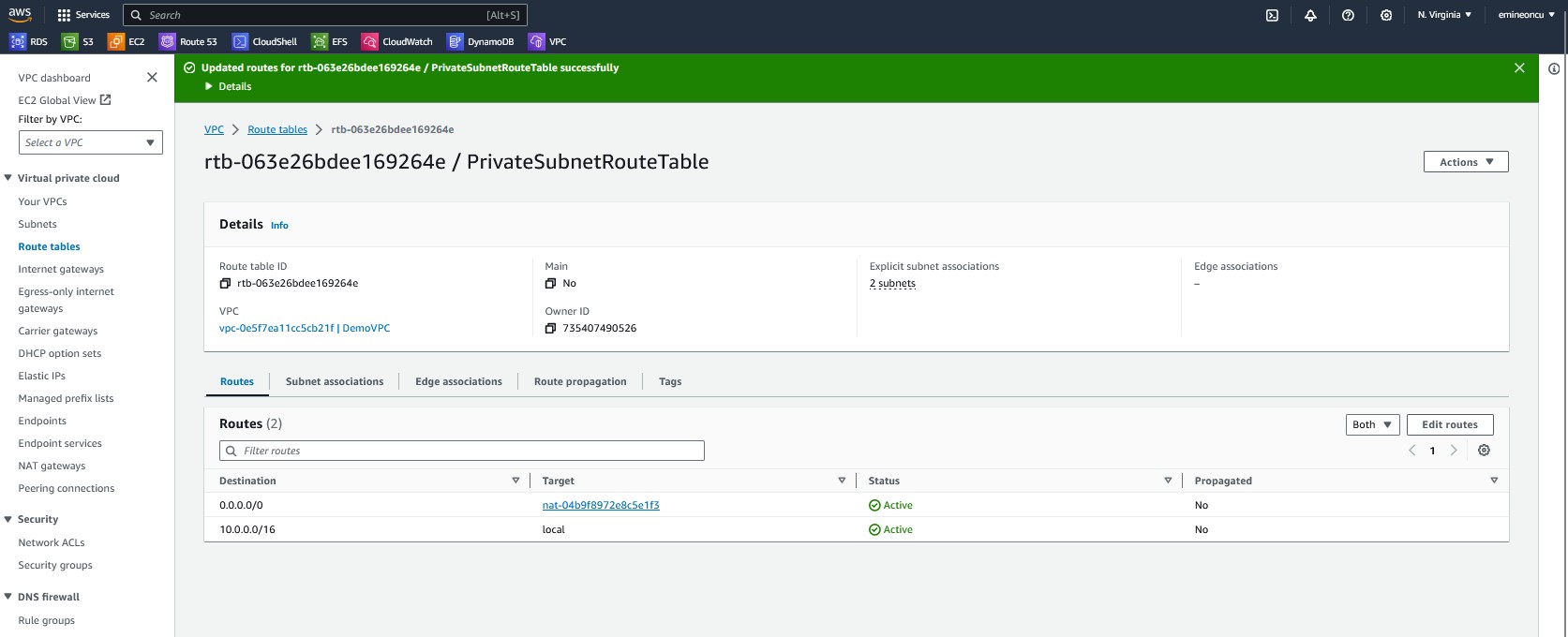


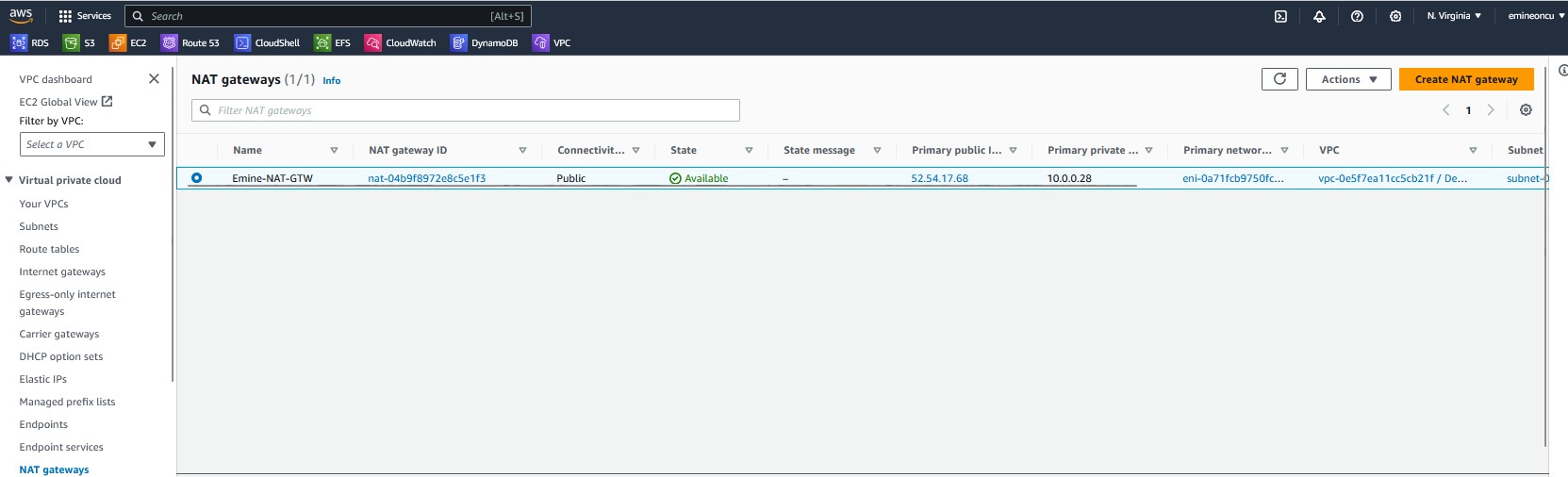
* **We are able to provide access to the public internet for our private instance without creating a public IP address.**
* **NAT Instances are now obsolete, and we will be using NAT Gateways in AWS going forward.**
* **Creating NAT Gateway**
  + **Creating a NAT Gateway.**
* **Created a NAT Gateway**



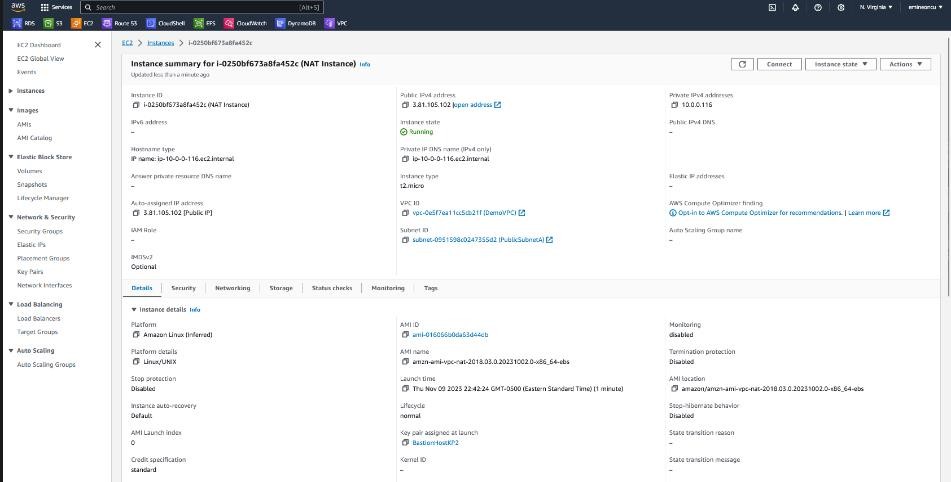
* **Adding route for the NAT GTW from the Private Route Table**







* **Creating NAT Instance, using an AMI which was specifically a NAT Instance AMI from AWS AMI Communities**

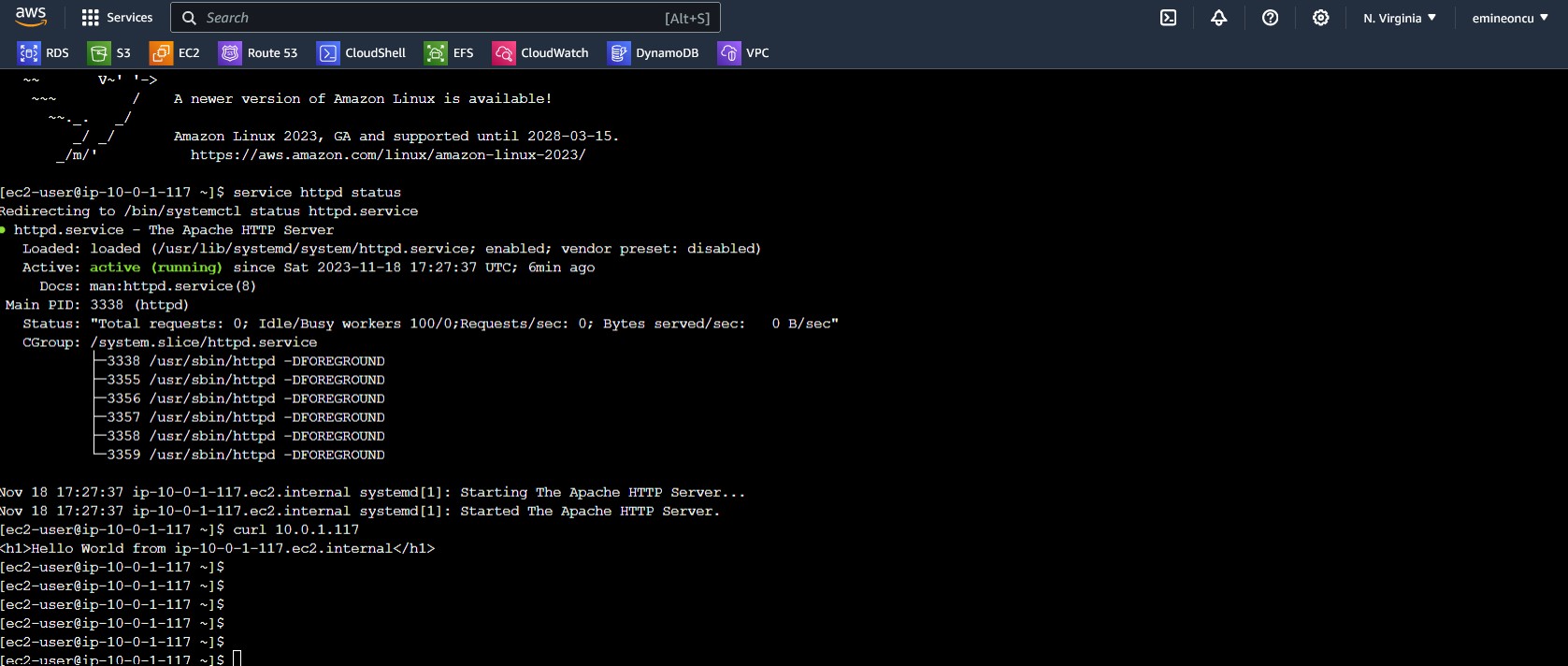


9.4 VPC Peering / VPC Endpoints / VPC LOGS WITH ATHENA

* **VPC peering section: If workloads within different VPCs need to communicate with each other, we**

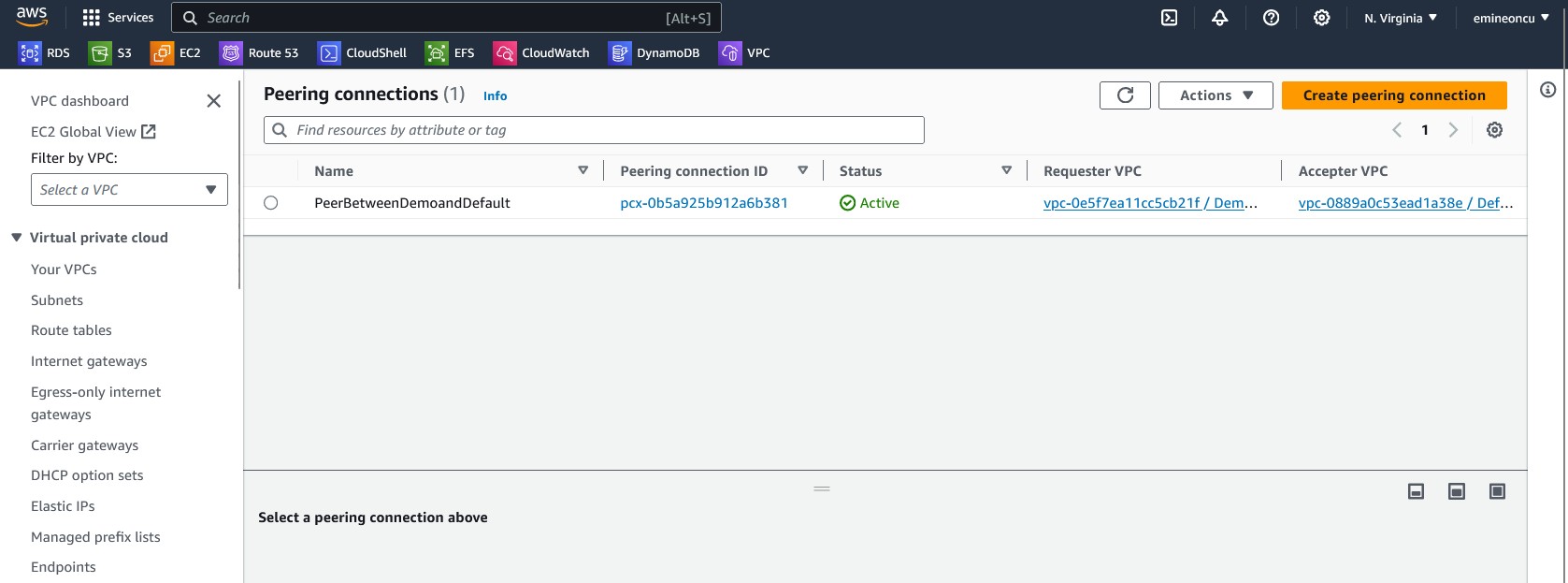
need to do VPC peering…

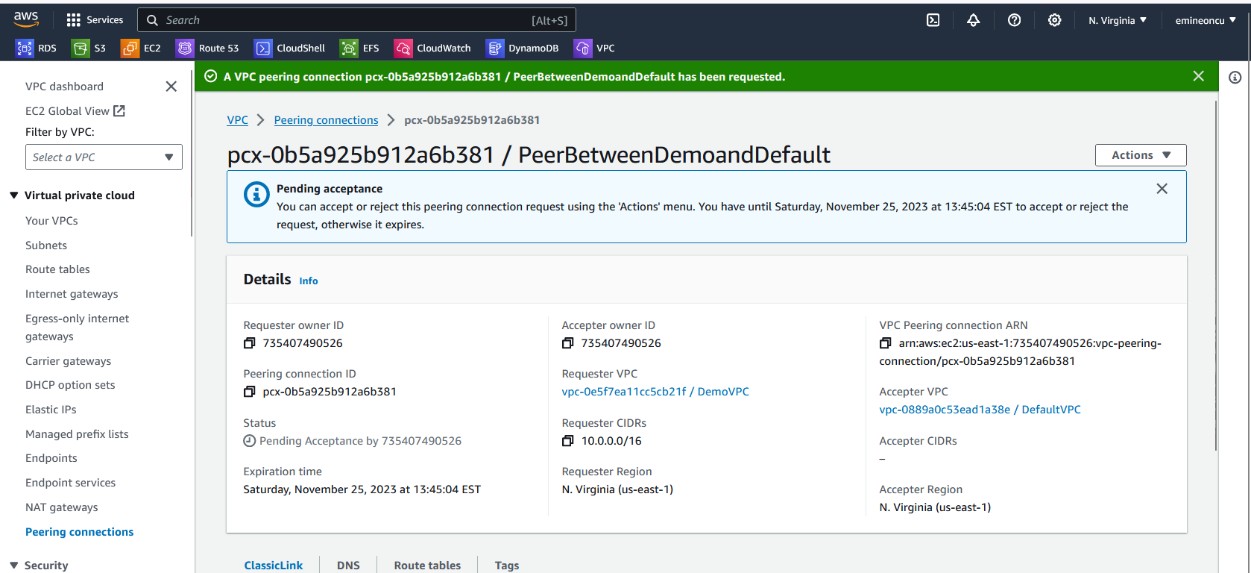
* **Launched an Apache web server on an EC2 instance within DemoVPC. Using curl command to retrieve content of that web page**.

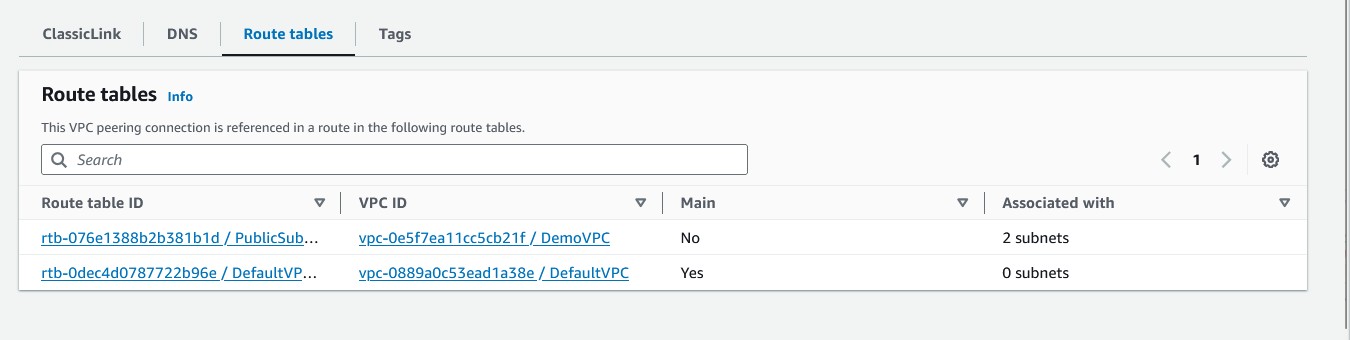


# No connection before VPC Peering

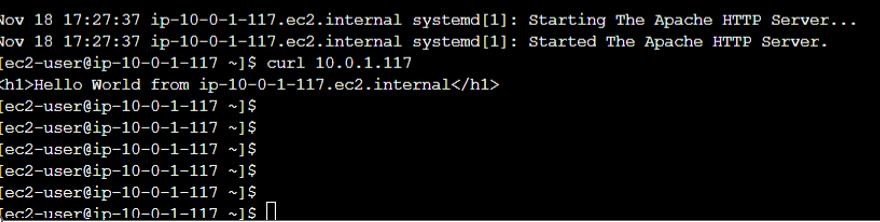








* + **Established connection. Screenshots of route tables and VPCs.**

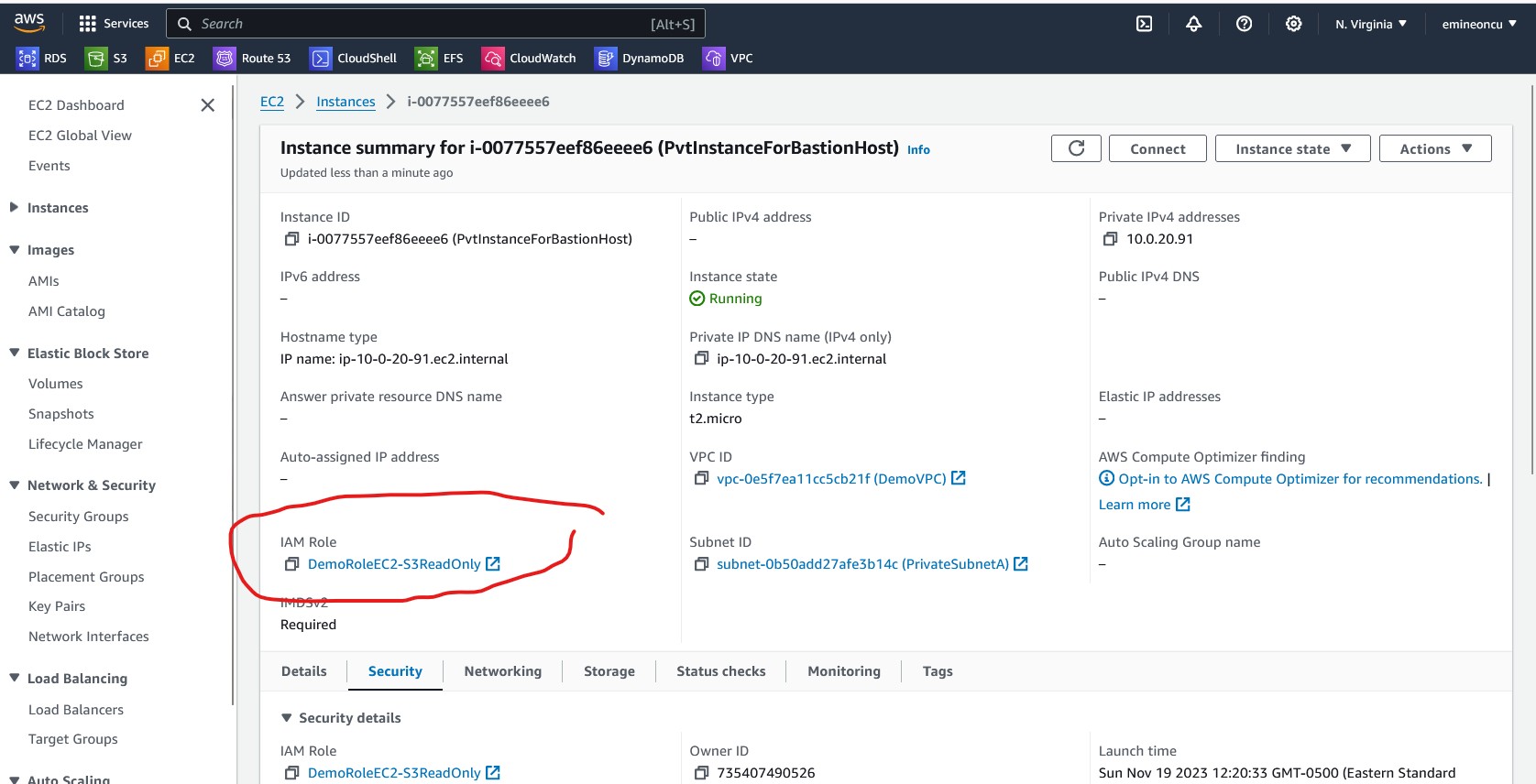


VPC ENDPOINTS

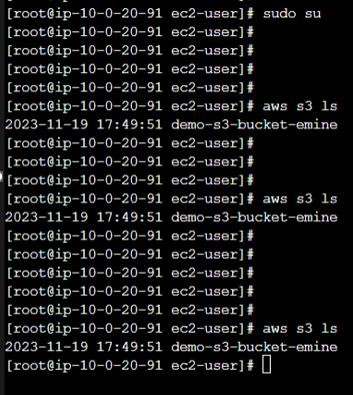
* + **Creating a role for Private EC2 Instance to access S3 bucket**



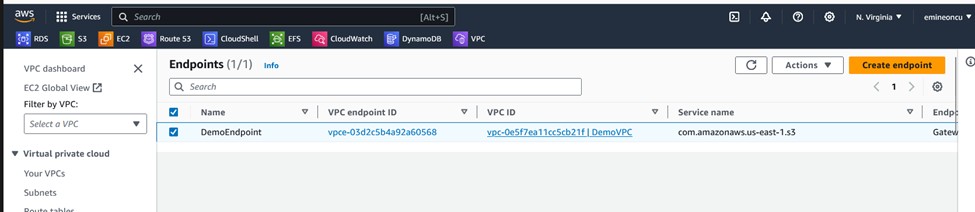
* + **Attached the newly created IAM Role to the Private EC2 instance:**

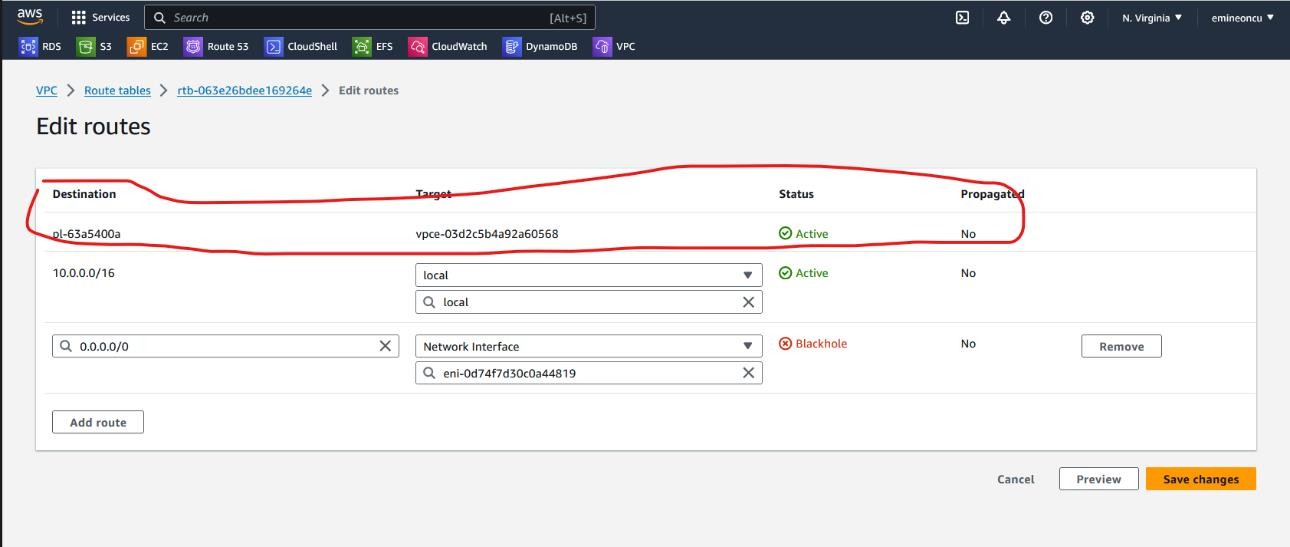


* + **Able to access the S3 bucket from private EC2 instance via NAT instance.**



* + **Created VPC Gateway Endpoint**



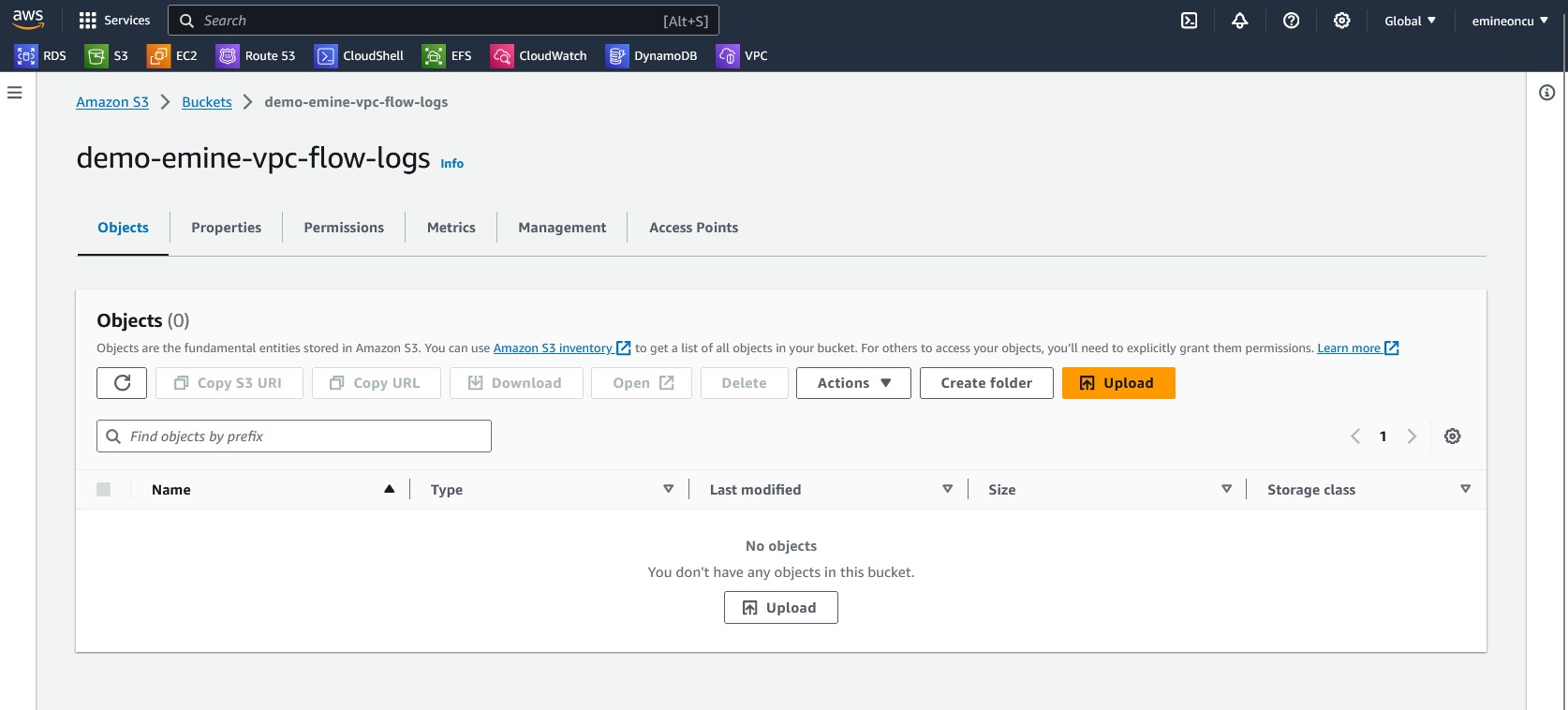


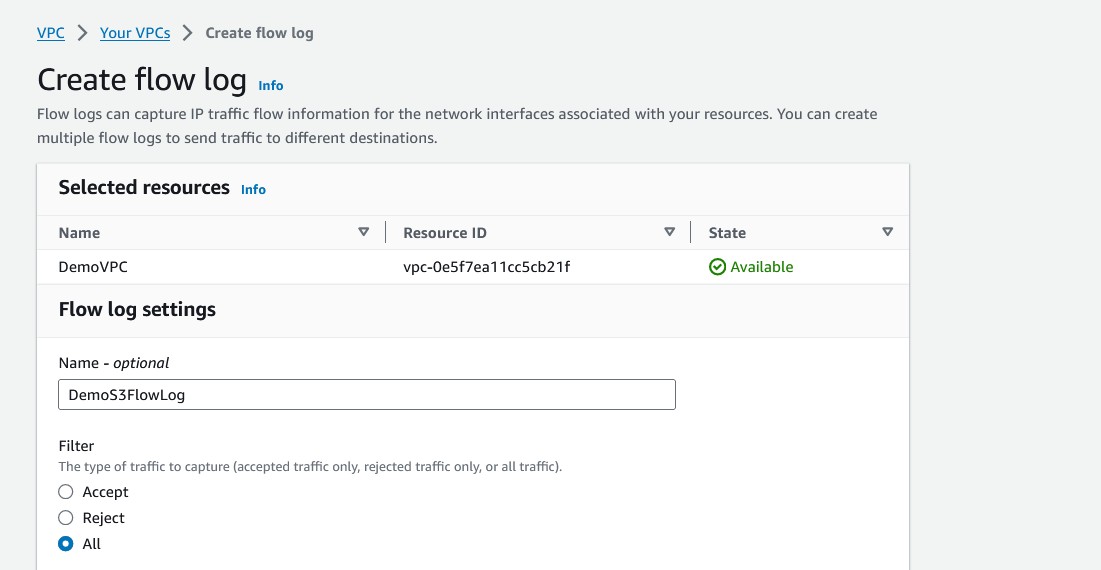
* + **S3 bucket is now accessible from private EC2 instance**



VPC LOGS WITH ATHENA

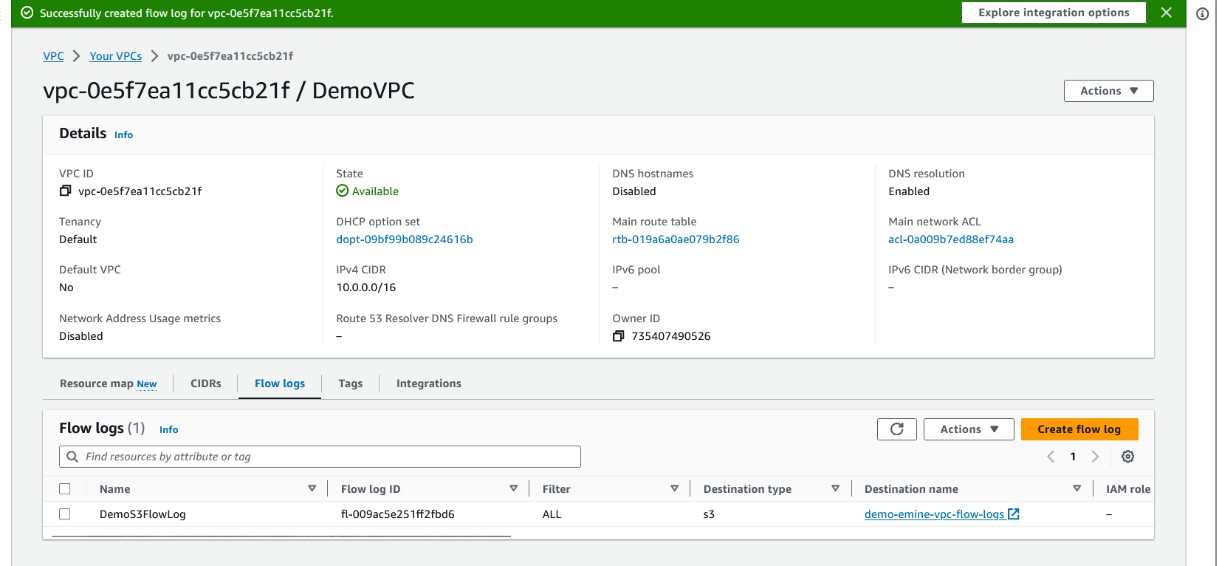
* + **Created an S3 bucket for VPC flow logs.**



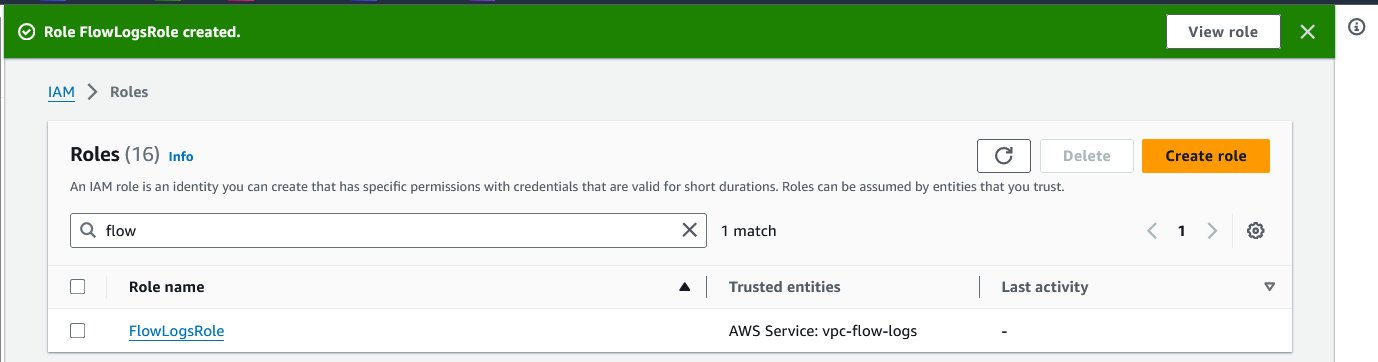
* + Added ARN number S3. (I just created up there)
* Creating flow log.



* + **Created an S3 bucket for VPC flow logs.**



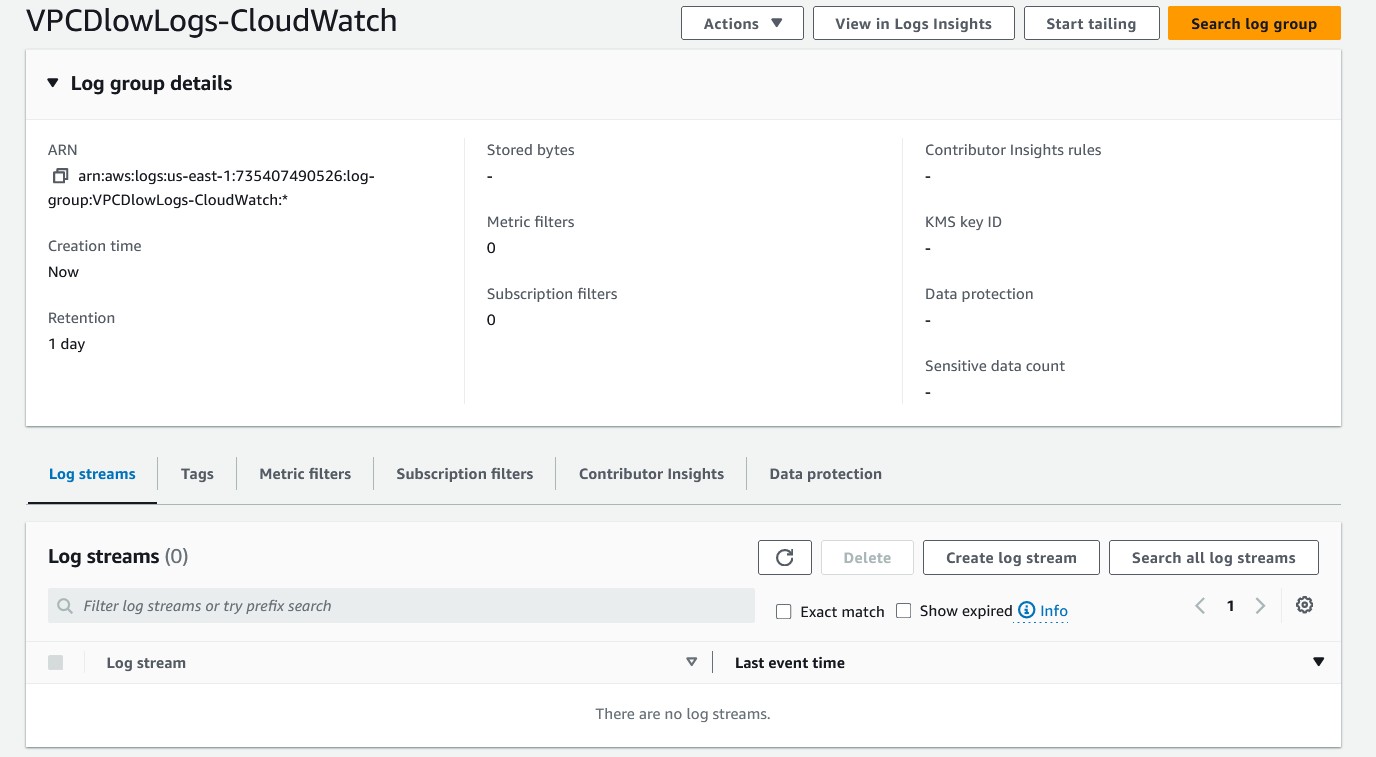
* + **“FlowLogsRole” Created**



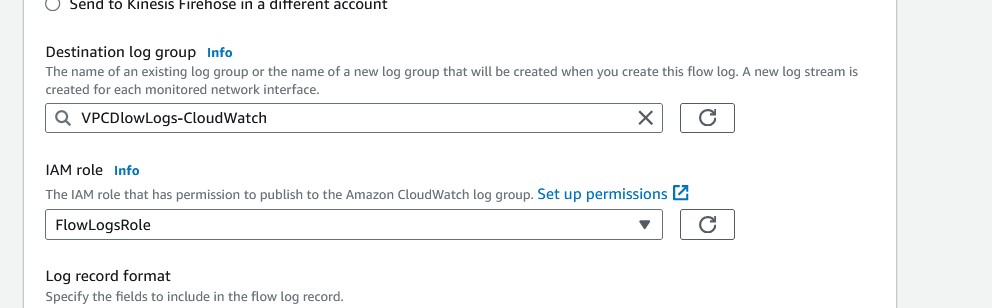
* + **Creating another FlowLog and adding “role” into this flow log.**



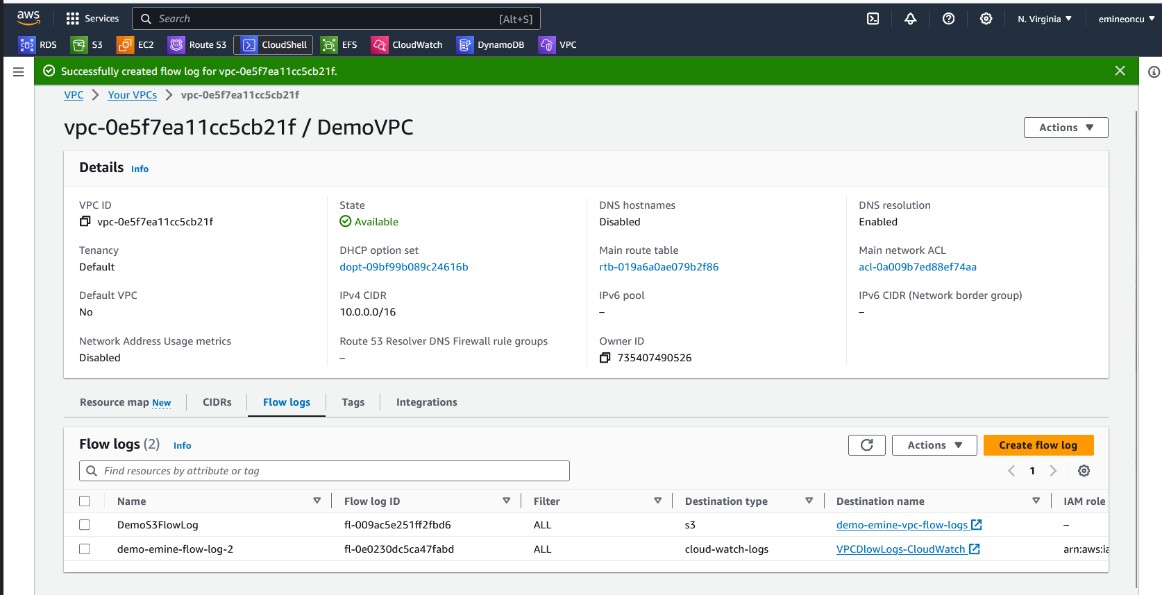
* + **We created Cloud Watch – log group.**



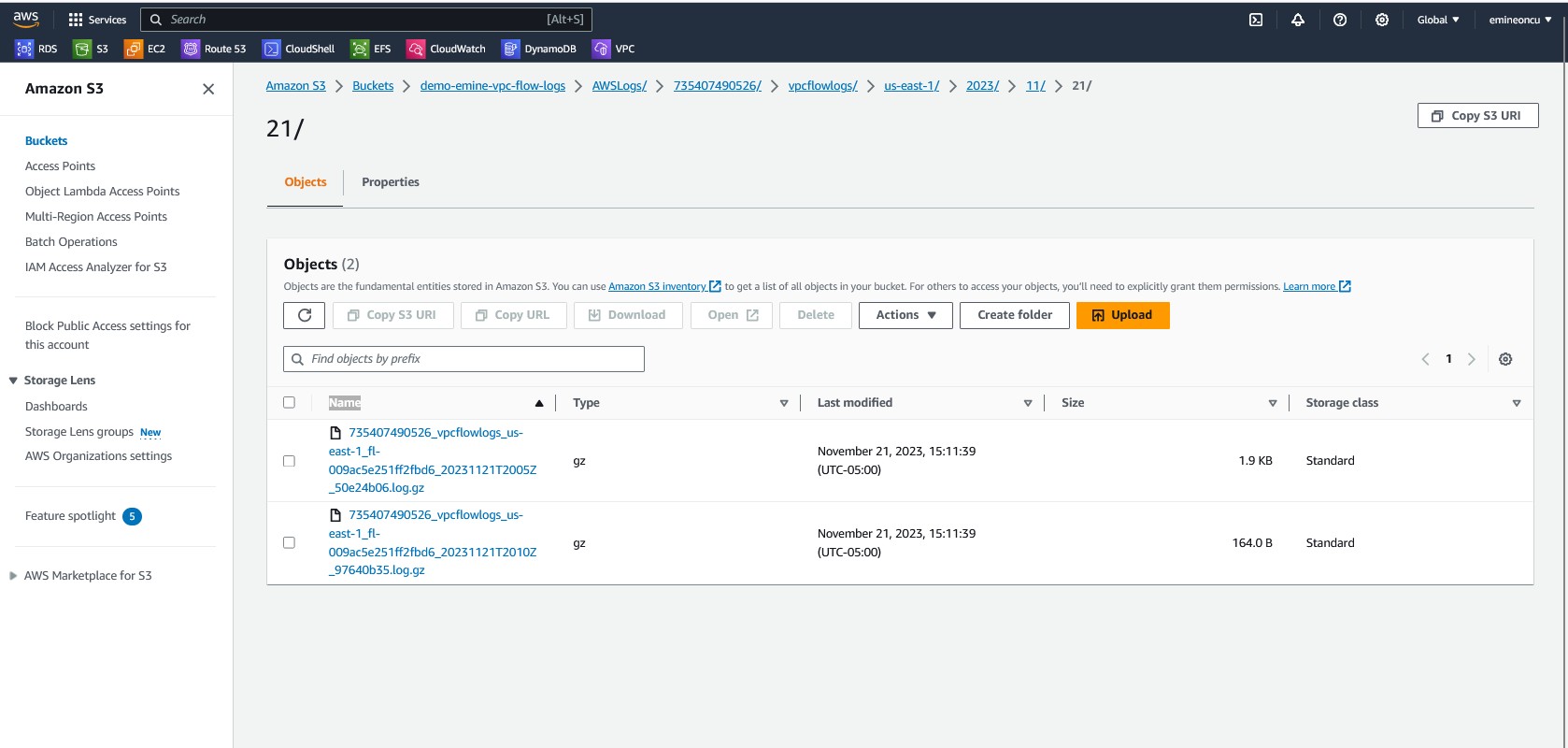
* + **Added into the Flow log creating step.**



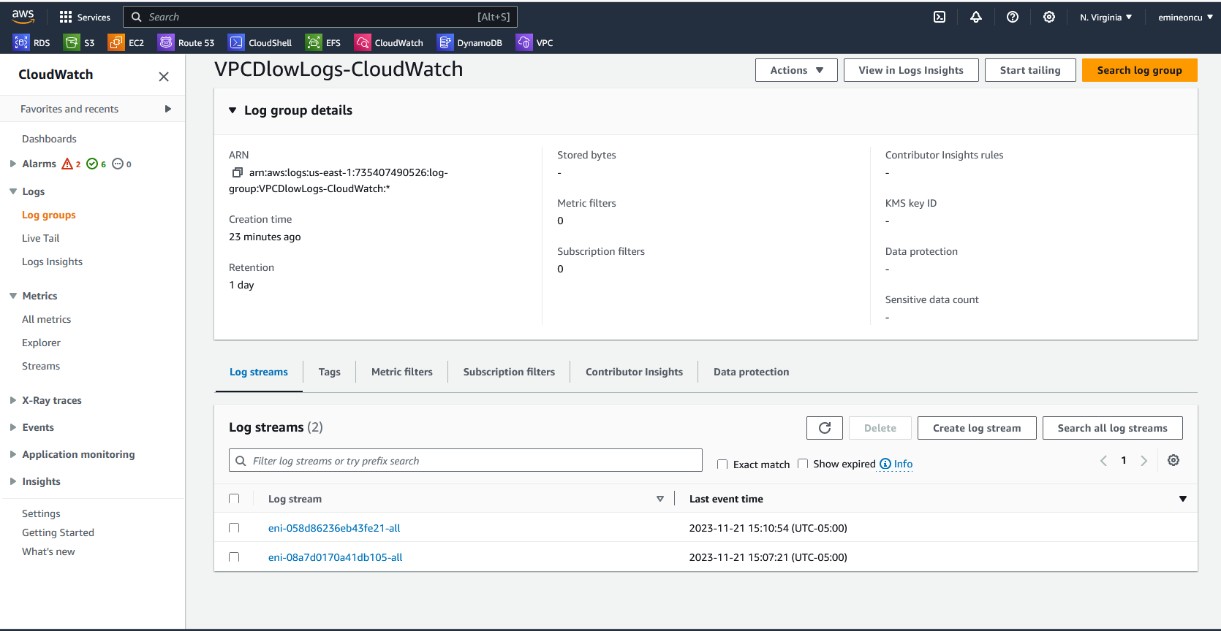
* + **Created both the flow logs 1] VPC Flow log capturing flow between EC2 and S3 and 2] Cloud Watch Flow log, capturing flow to CloudWatch.**



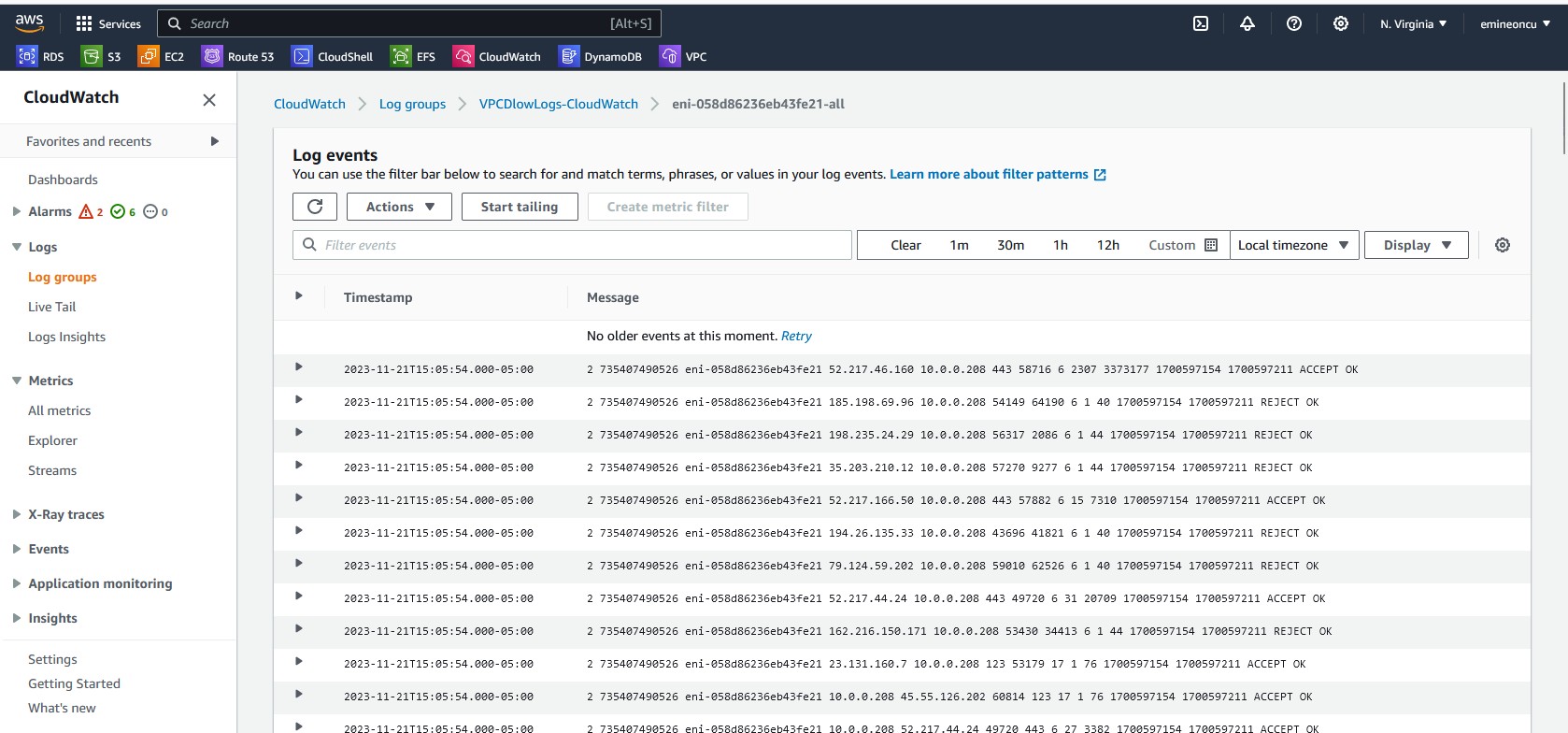
* + **VPC Flow logs are being captured when S3 bucket is accessed from private EC2 instance:**



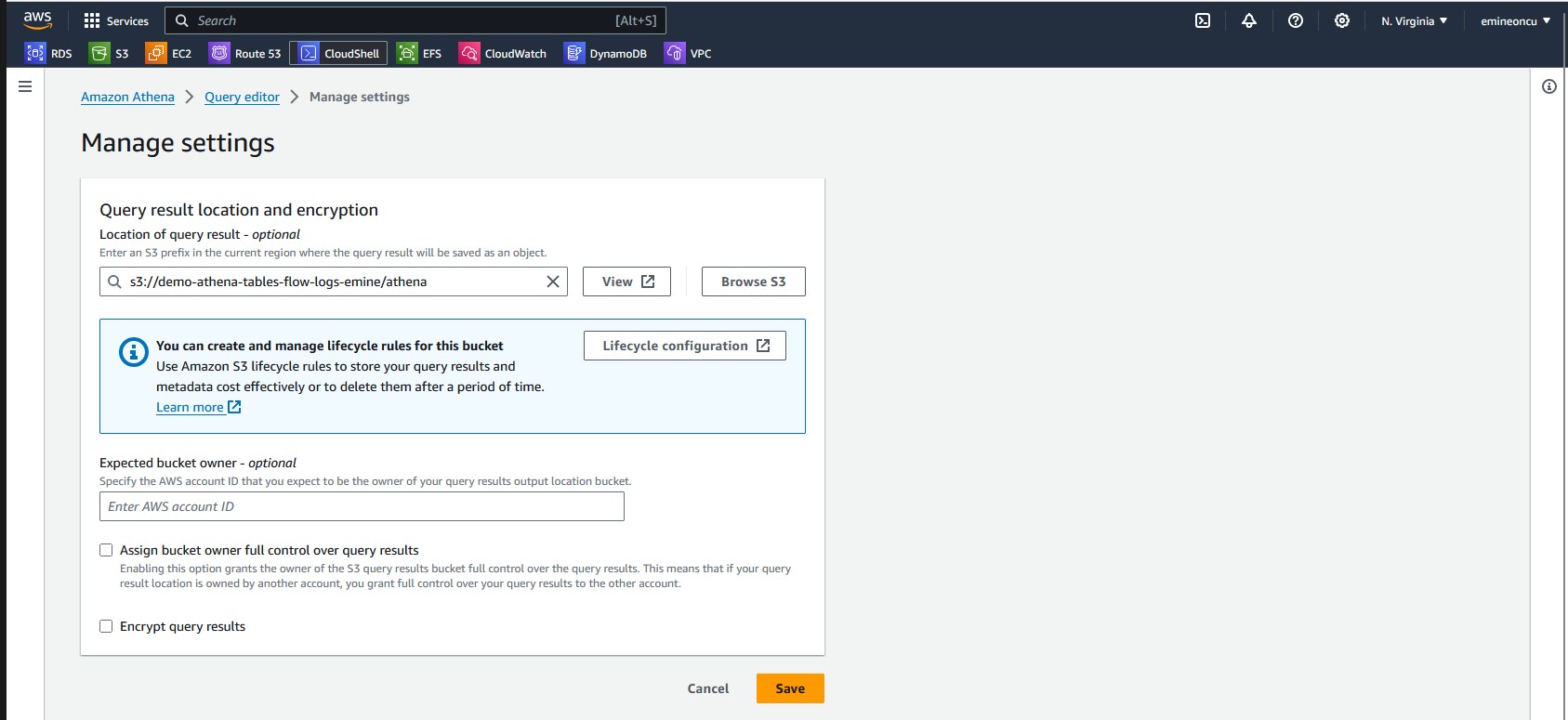
* + **Cloud Watch logs are now generated as well as seen below:**



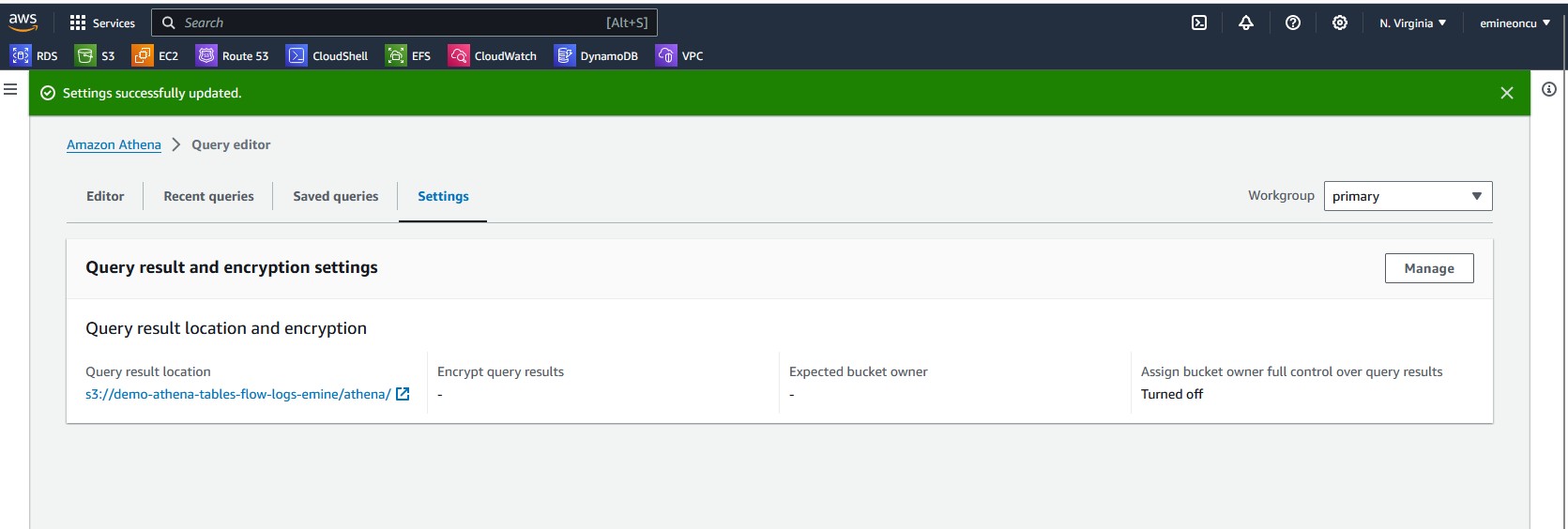
* + **There are a few Accepts and few Rejects, which we can see in the Cloud Watch Logs:**



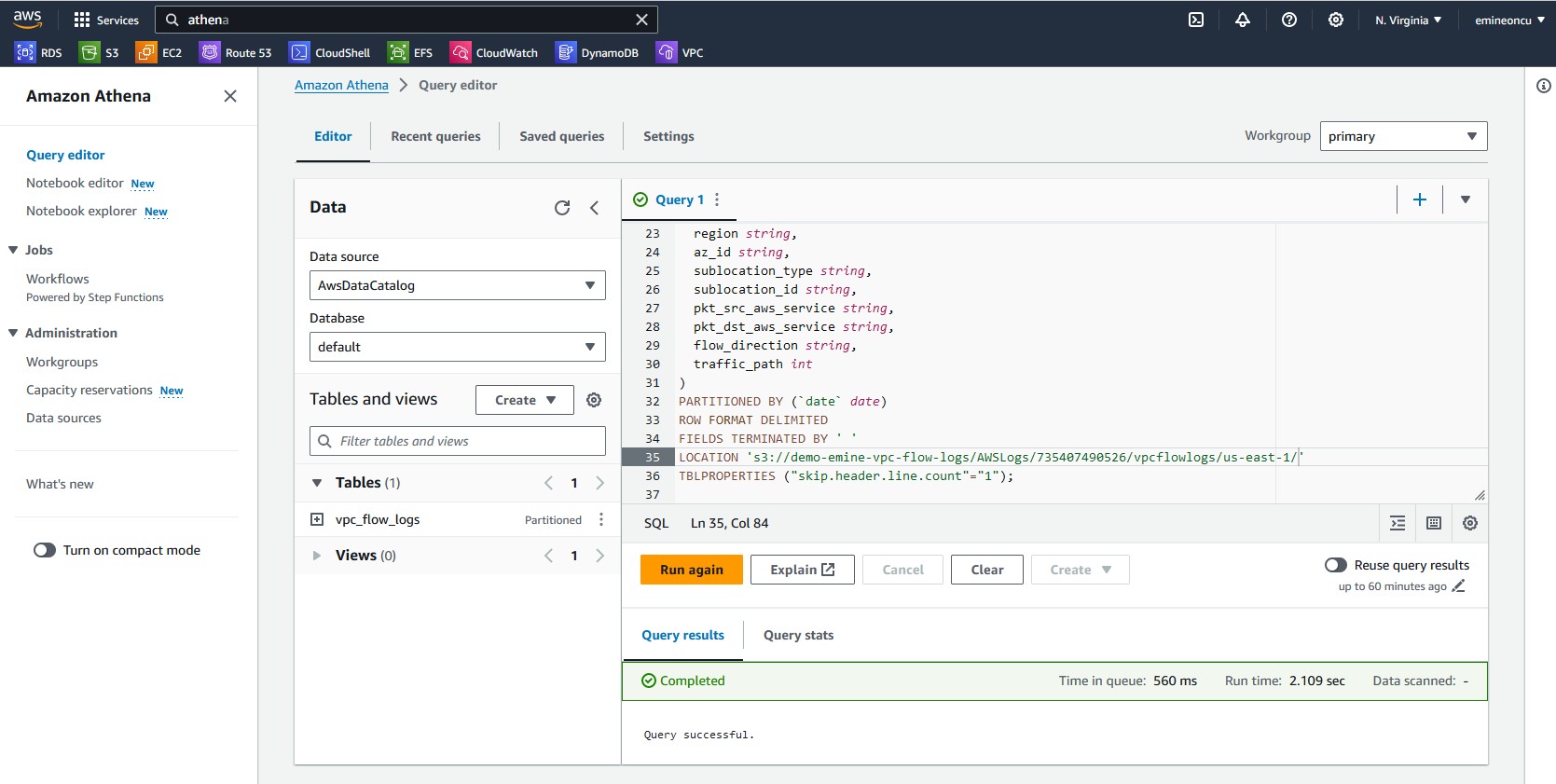
* + **Athena:**
  + **Creating database in Athena to query S3 bucket for flow logs:**

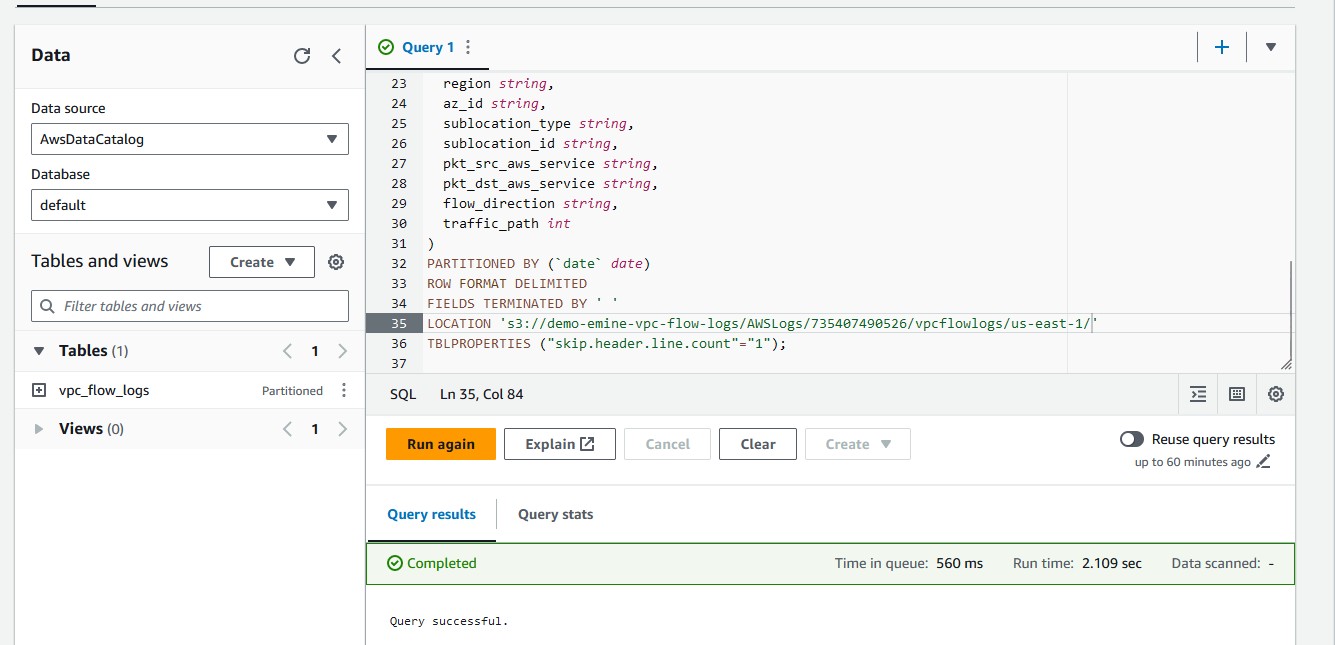


ARN NUMBER: arn:aws:s3:::demo-athena-tables-flow-logs-emine CHANGED: s3://demo-athena-tables-flow-logs-emine/athena



* + **Created a database table to store the VPC flow logs.**

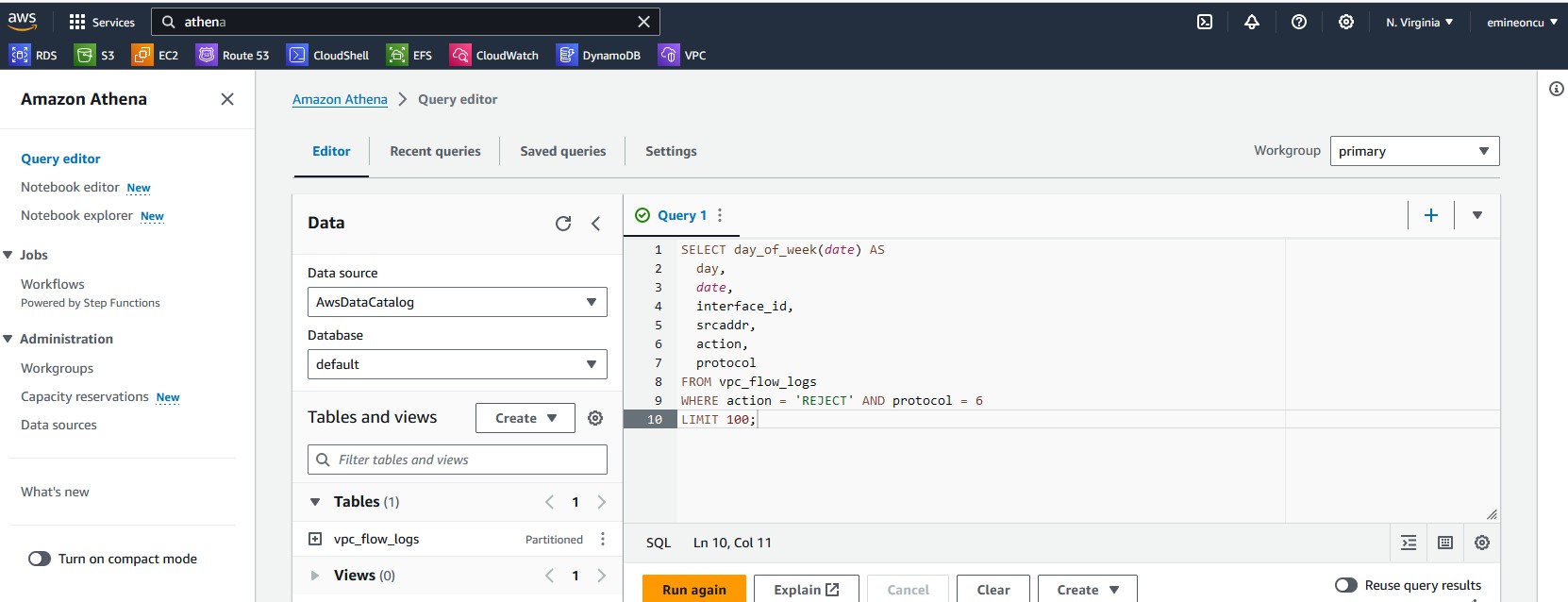


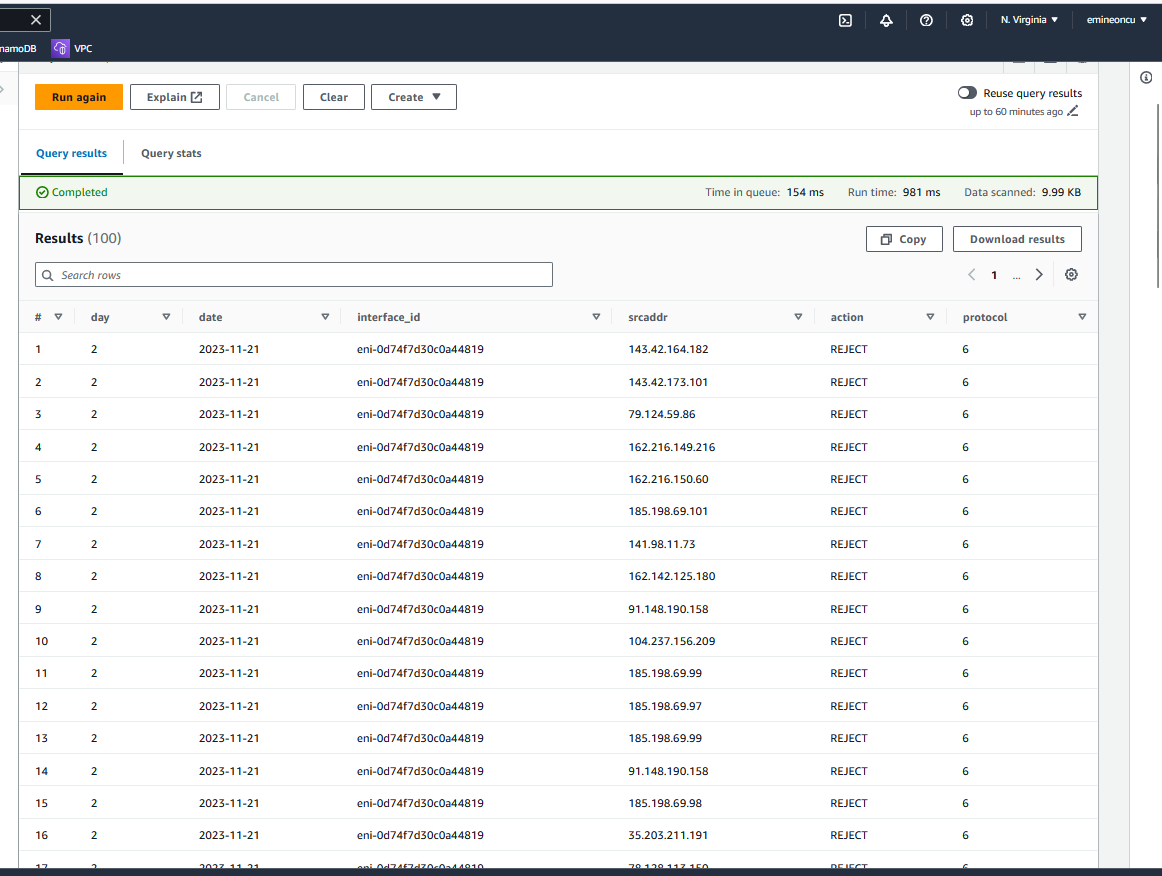


* + **Created a partition to capture flow log data for a specific date. This can be automated using AWS Glue, so a Glue ETL job can be scheduled to create these partitions on a daily basis.**

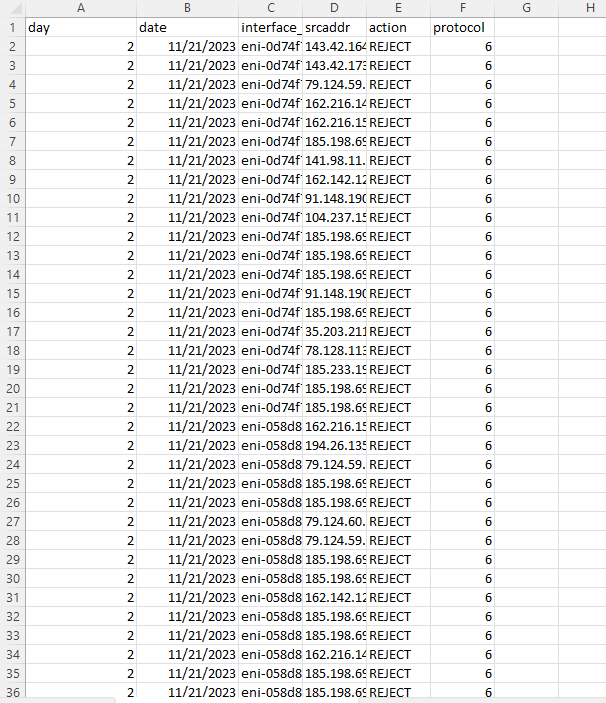


* + **We can now query this table to look for ‘Rejects’ as an example:**

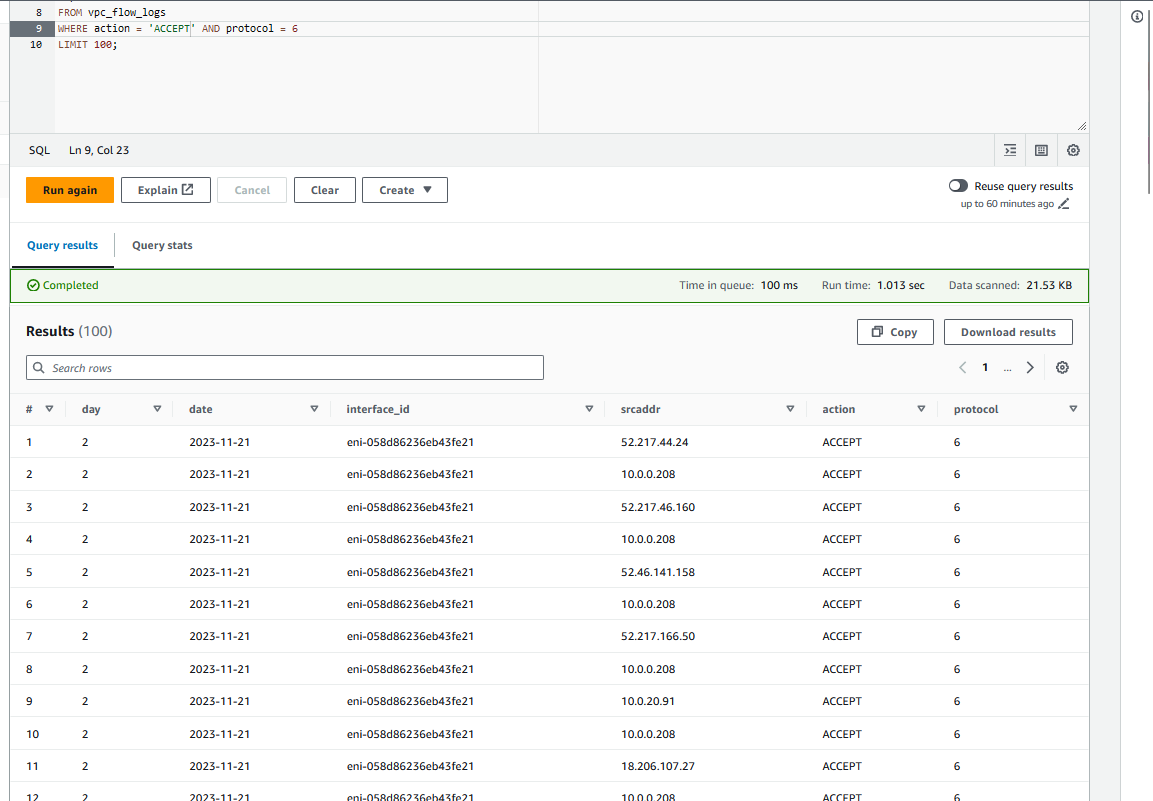




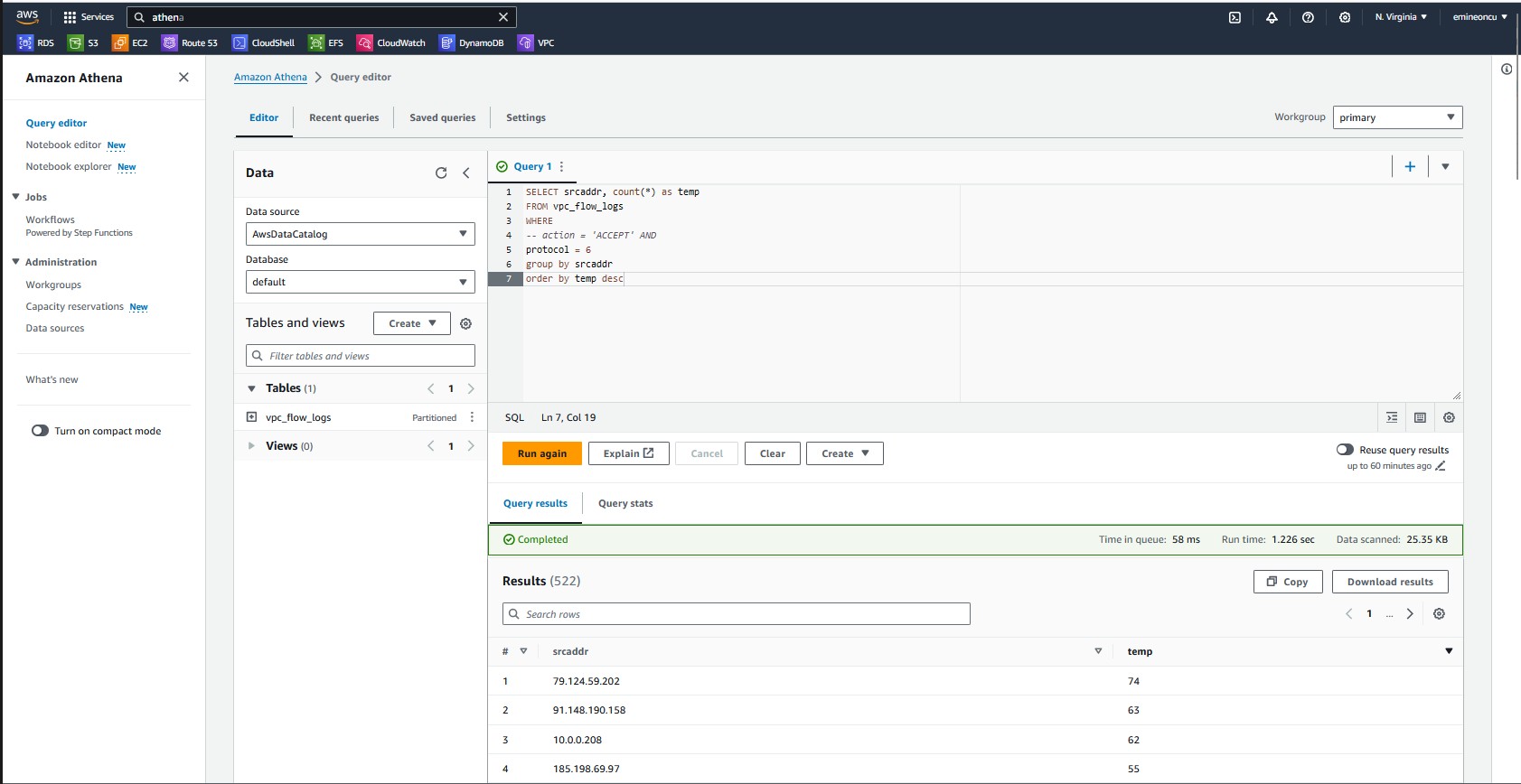
# Results downloaded (excel file)



* + **We can now query this table to look for ‘ACCEPT’ as an example:**



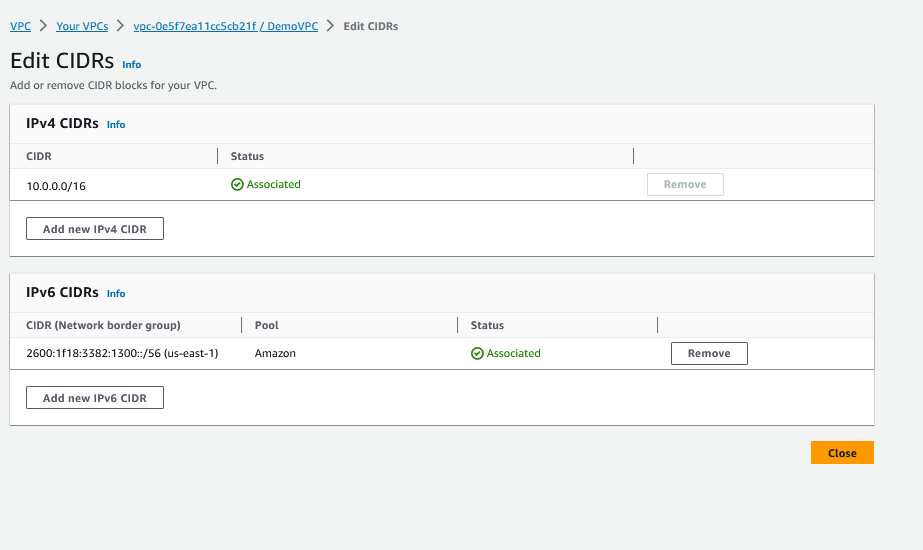
* + **This query will give a count (in descending order) of the src IP addresses invoking S3.**



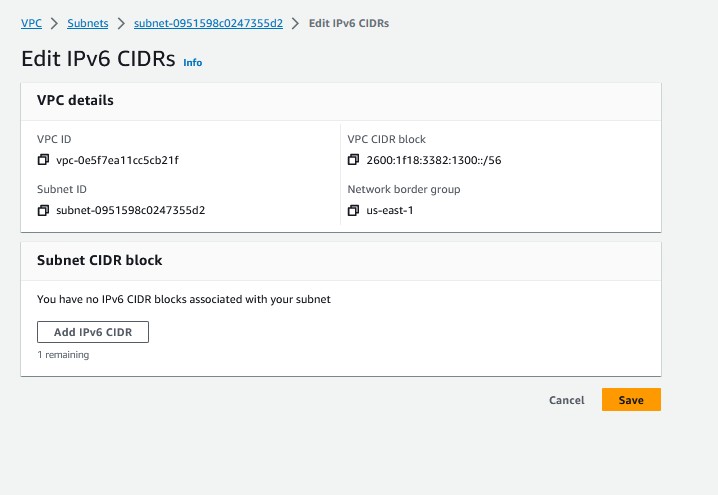
IPV6 SECTION

* + **Create a new IPV6 CIDR range within the DemoVPC.**

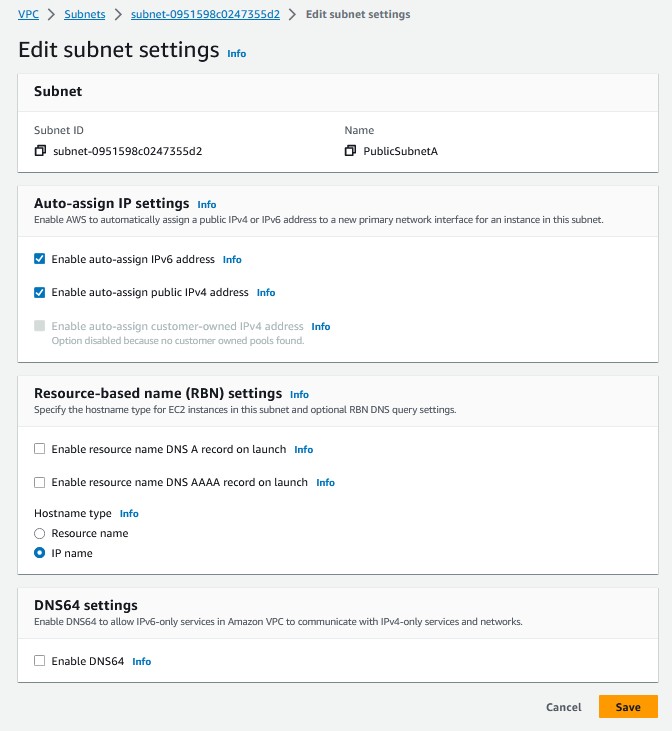


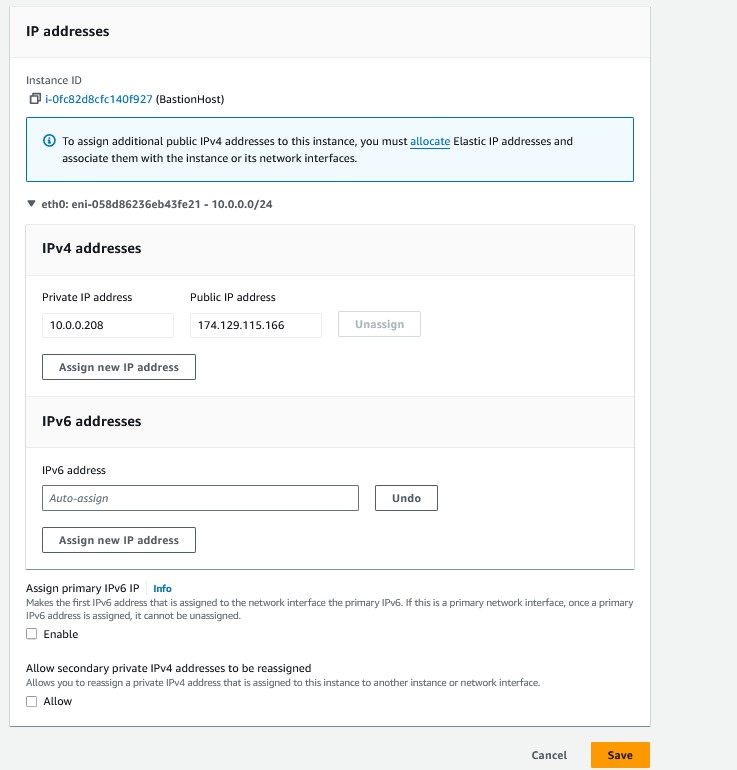


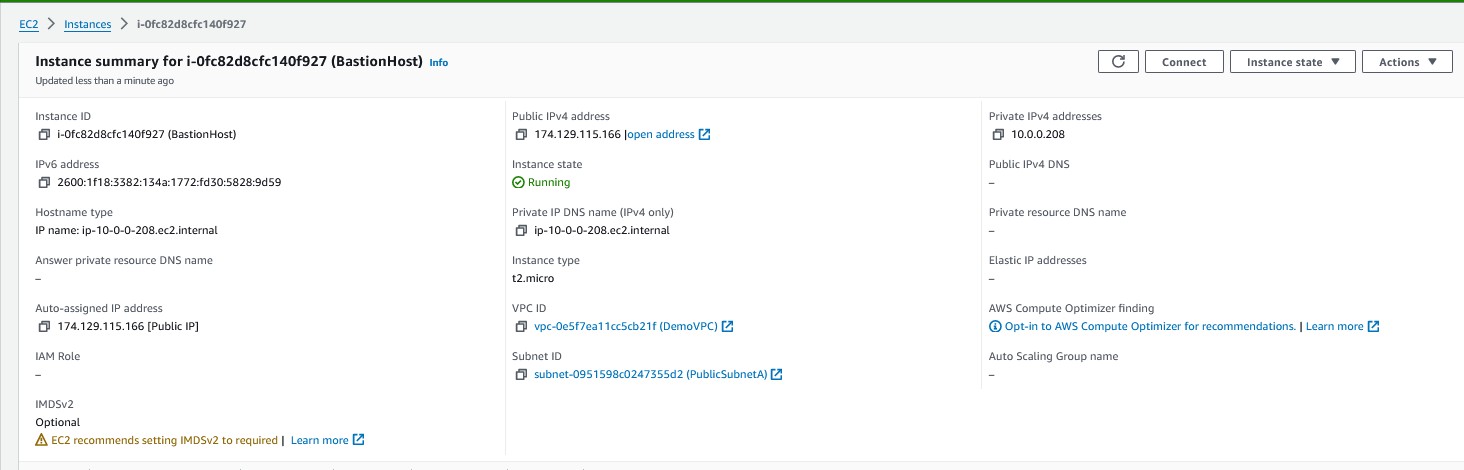
* + **Enabled Auto Assign IPV6 at the Subnet level.**

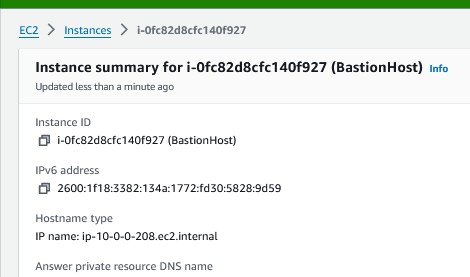


* + **Auto assigning IPV6 address to Bastion Host**

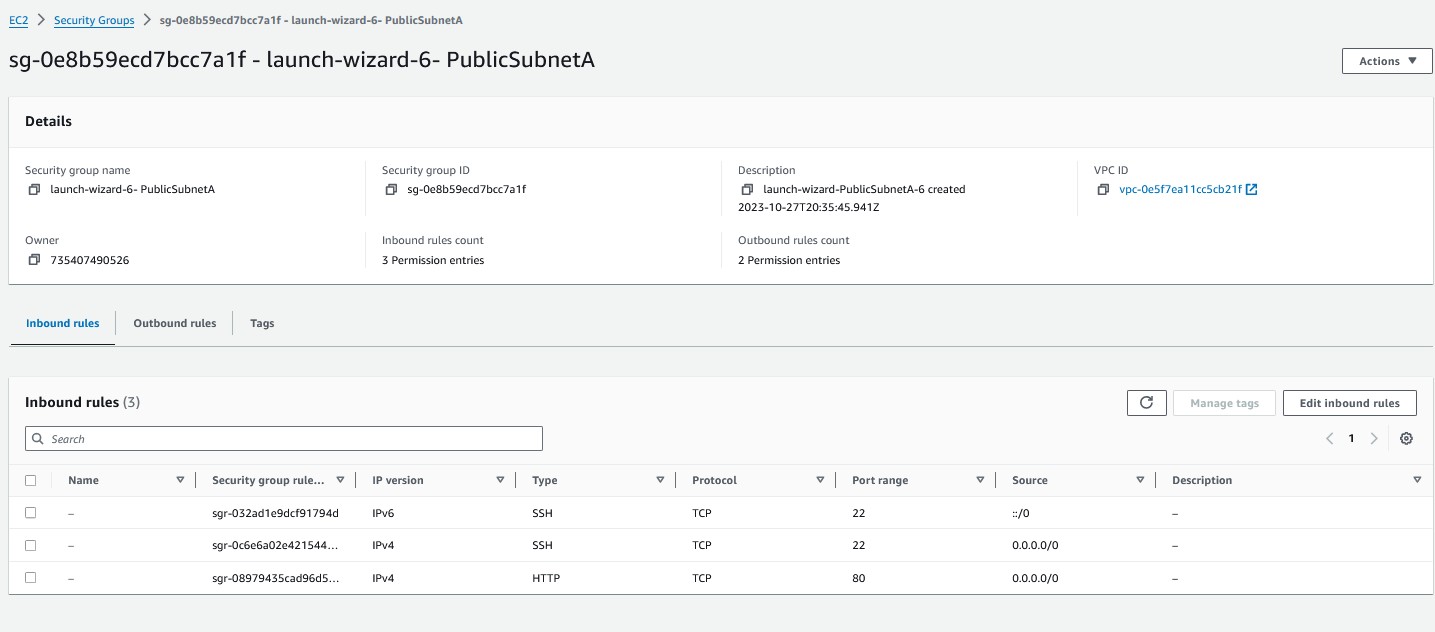


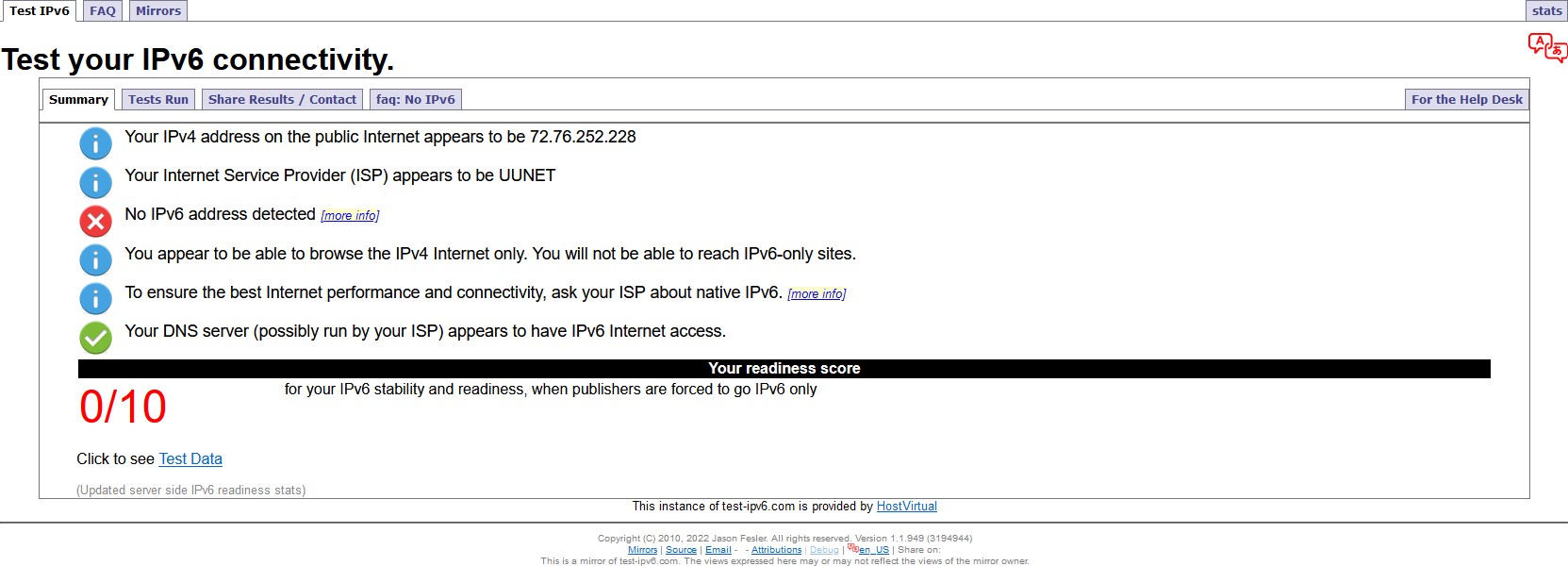






* + ***Added an IPV6 Inbound rule for the Bastion Host Security Group***





* + **Route tables also have a auto assigned IPV6 address, which means all connectivity within Demo VPC for our Subnets can now use IPV6**

