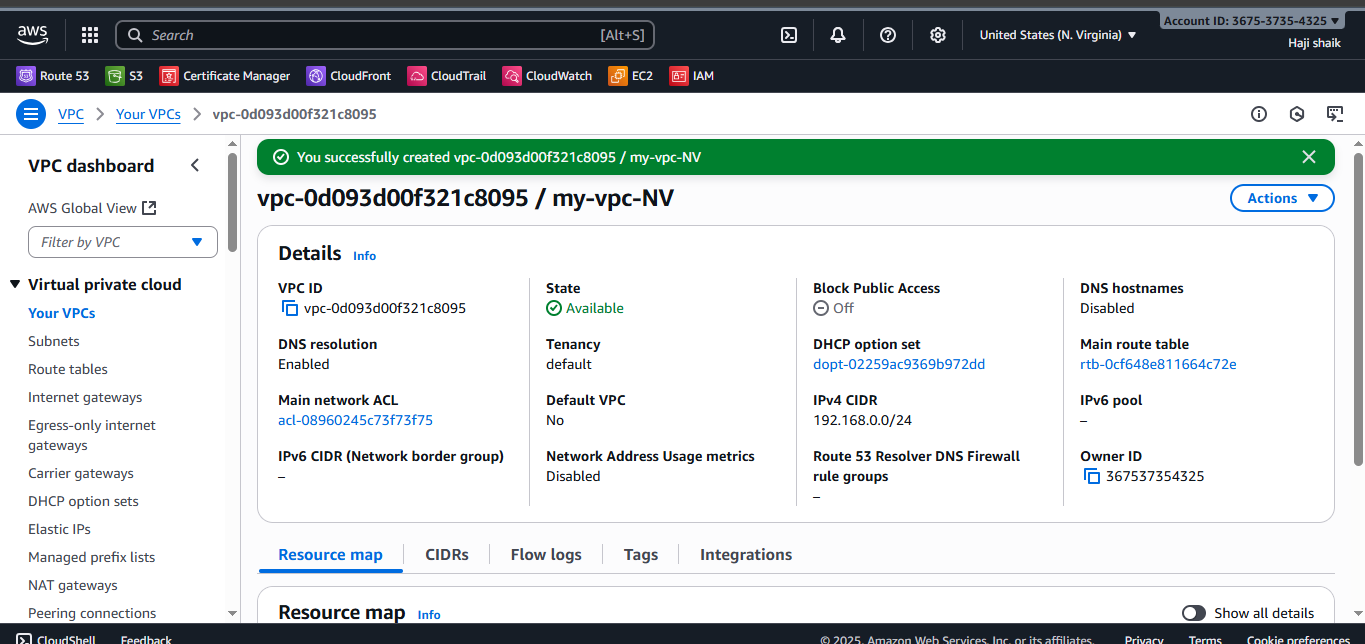
# Auto scalling tasks.

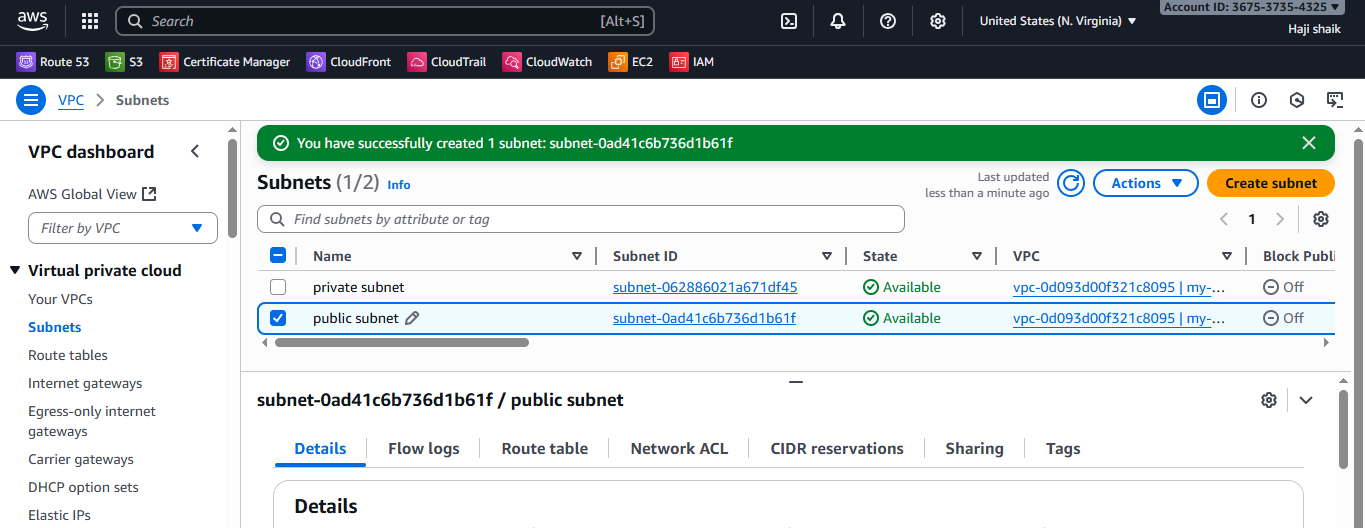
1. **Create one VPC in N. Virginia region.**

* Go to aws console.
* Choose a region N. Verginia (us-east-1)
* Then open a search bar and create vpc in the region.



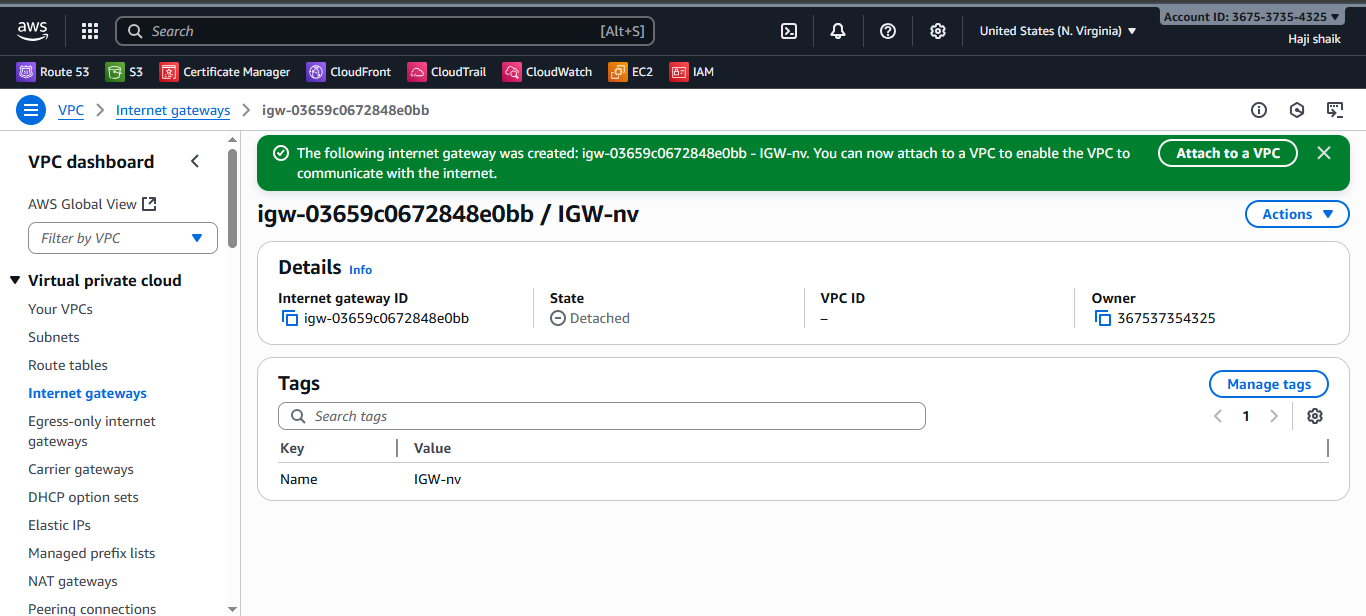
1. **Create two subnets: one public subnet and one private subnet.**

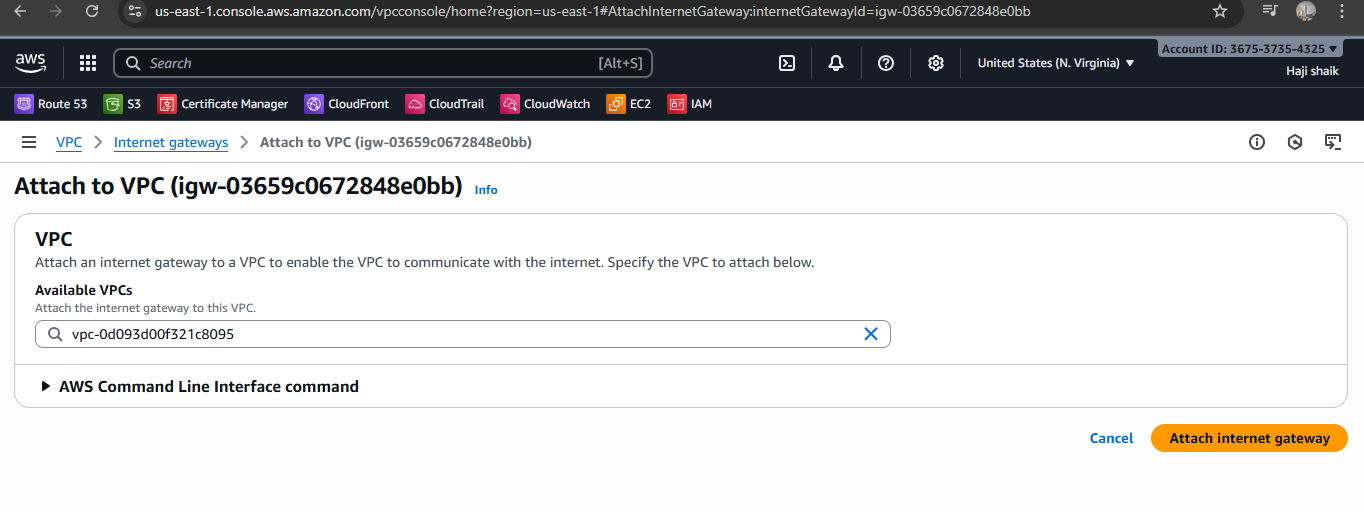
* Go to aws console.
* Open a search bar nd go to vpc.
* Then go to create subnets.

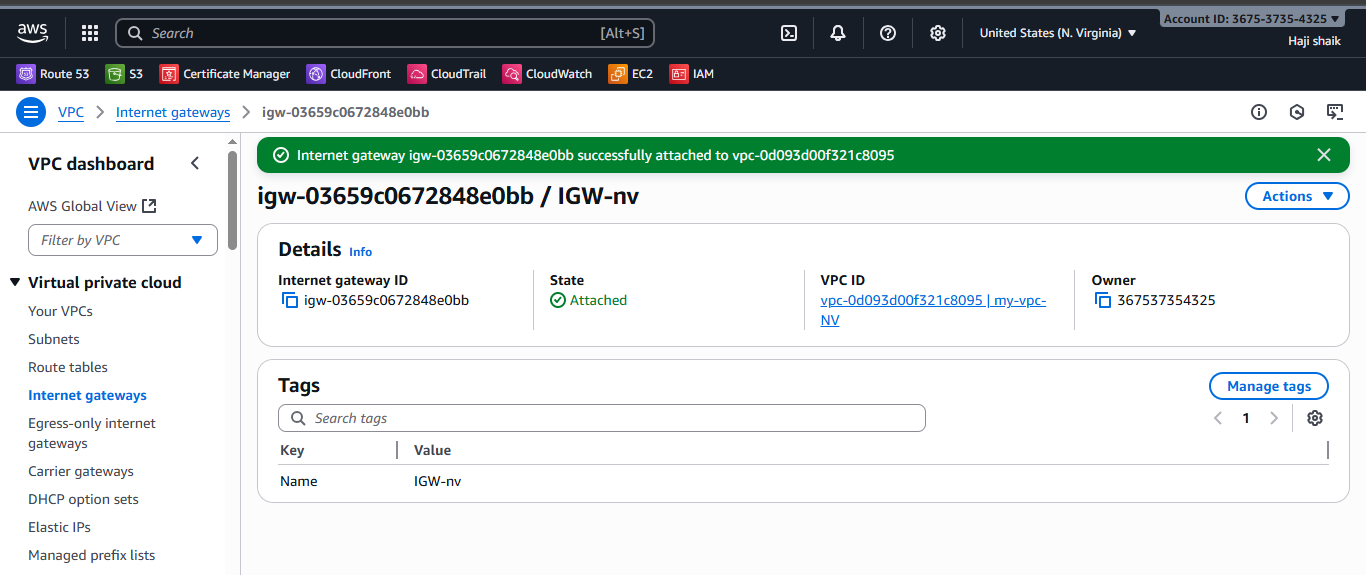


1. **Attach an IGW to the VPC.**

* **Go** to aws console.
* Then go ti IGW
* Then create IGW.
* After a creating IGW go to actions and attach vpc there.

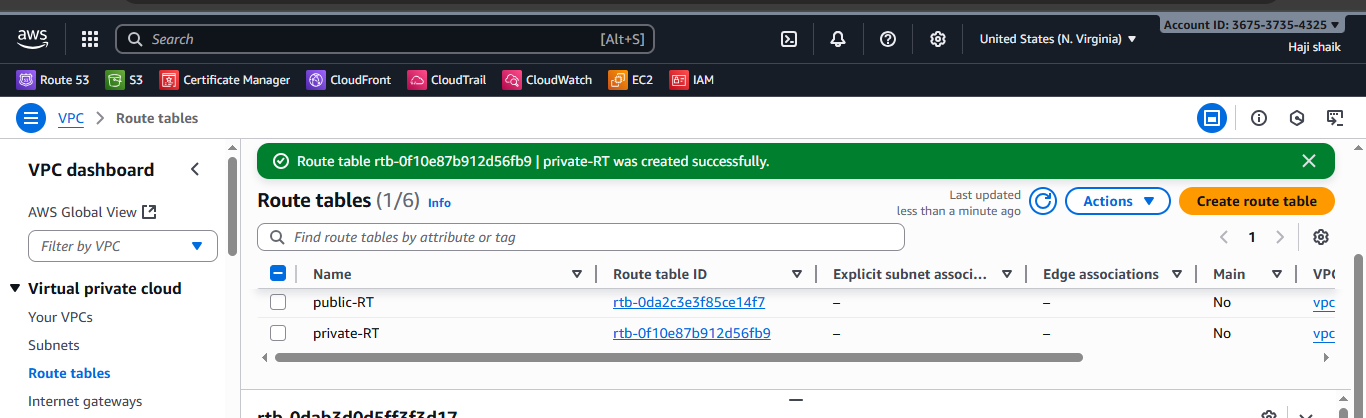




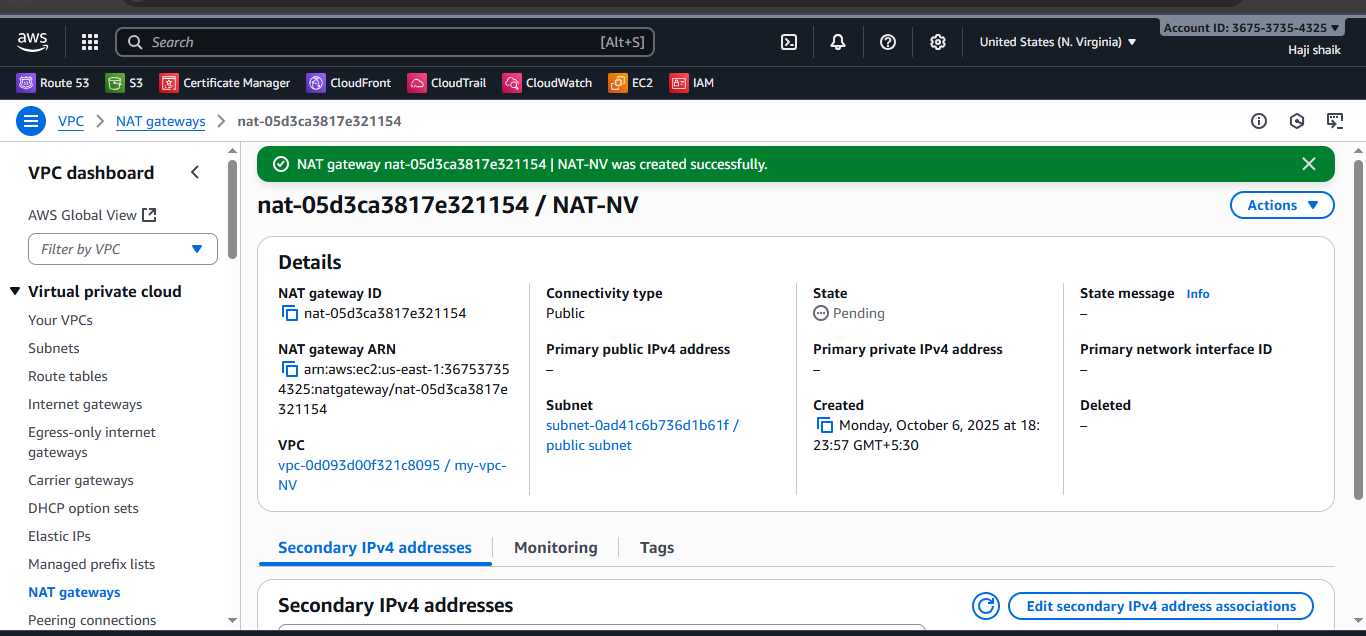


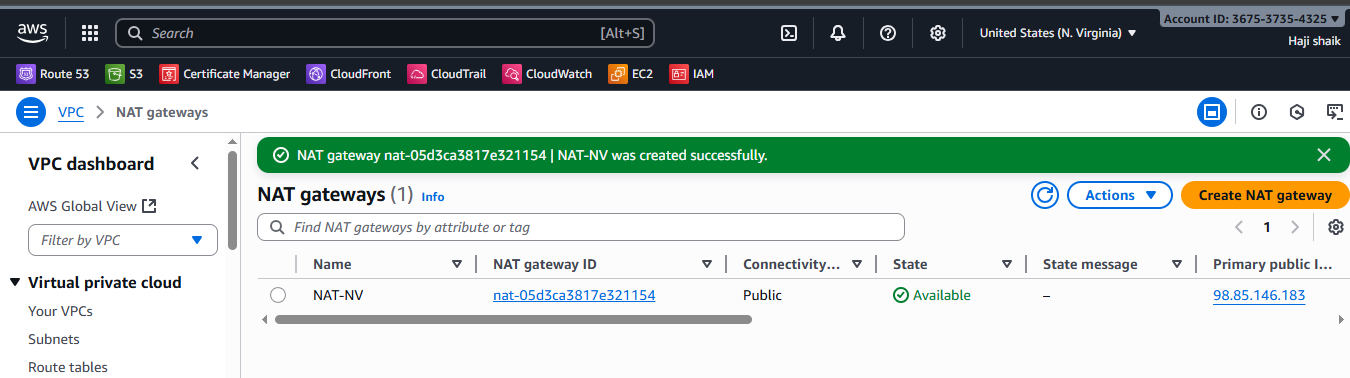
1. **Create one public route table (RT) and one private route table.**

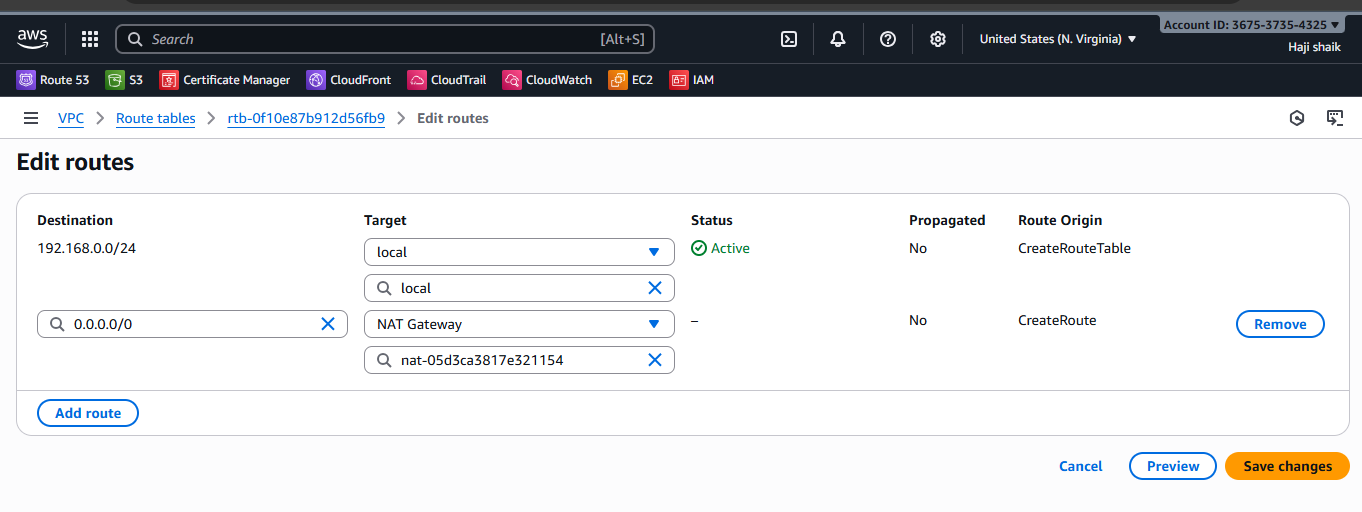
* **Go** to aws console.
* Then go to the sarch bar nd enter vpc.
* Then go to the route table one for public and one for private.
* Here the results are:

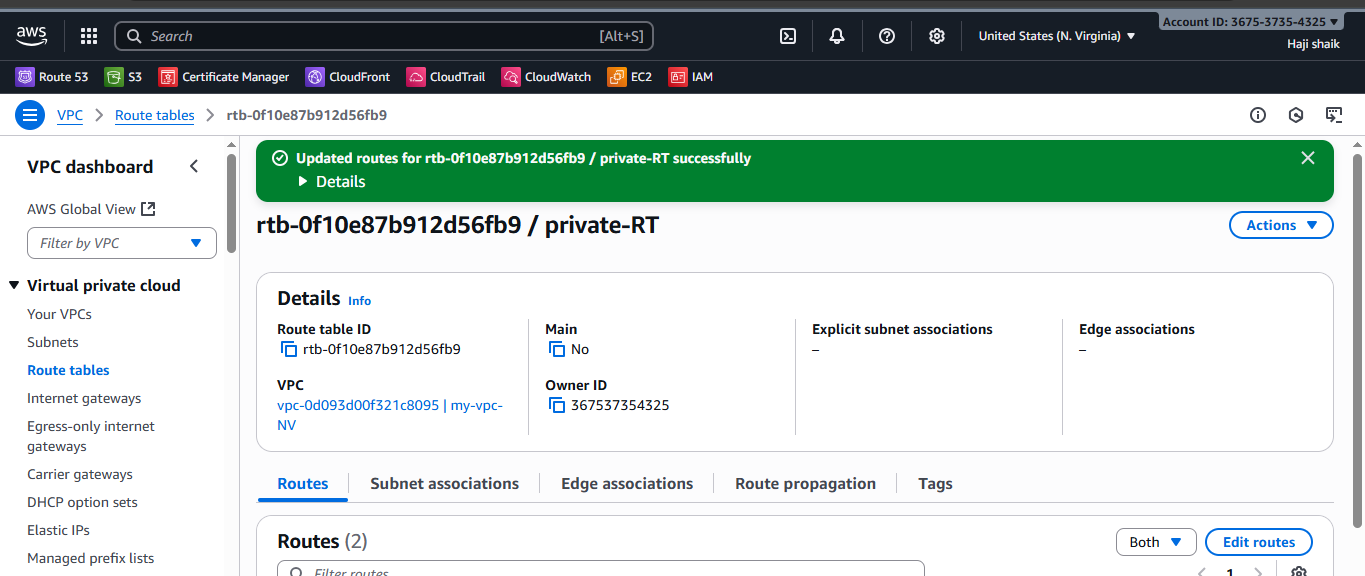


1. **Deploy a NAT gateway in the public subnet and attach the NAT gateway to the private subnet.**





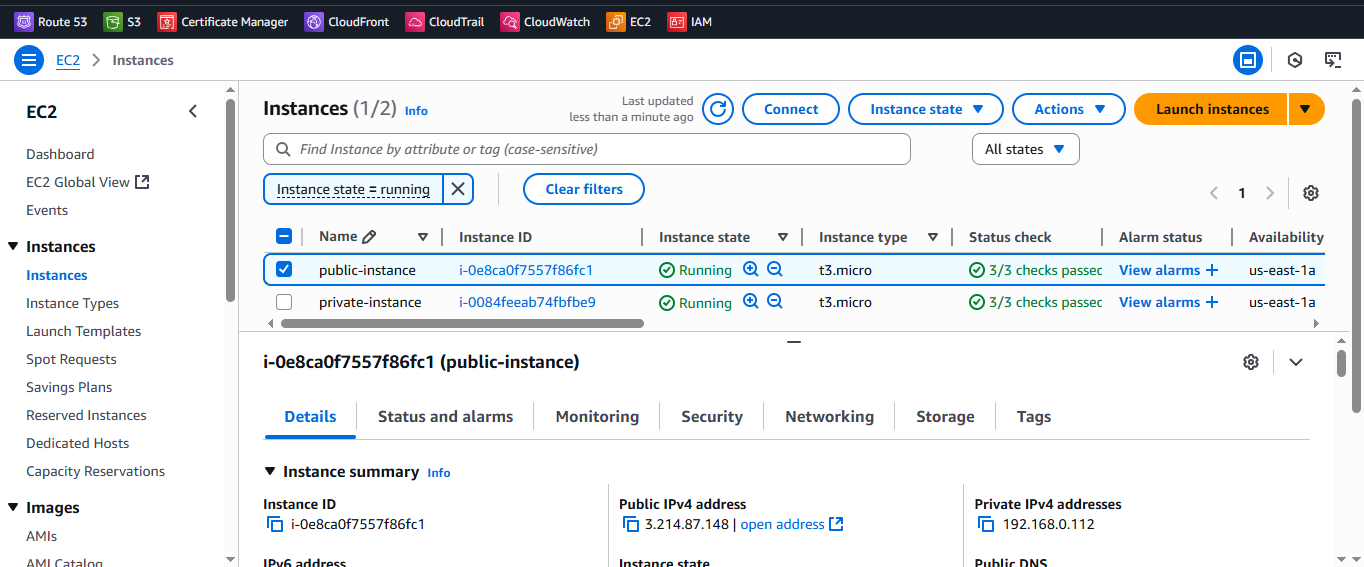




1. **Create two instances, one in the public subnet and one in the private subnet.**

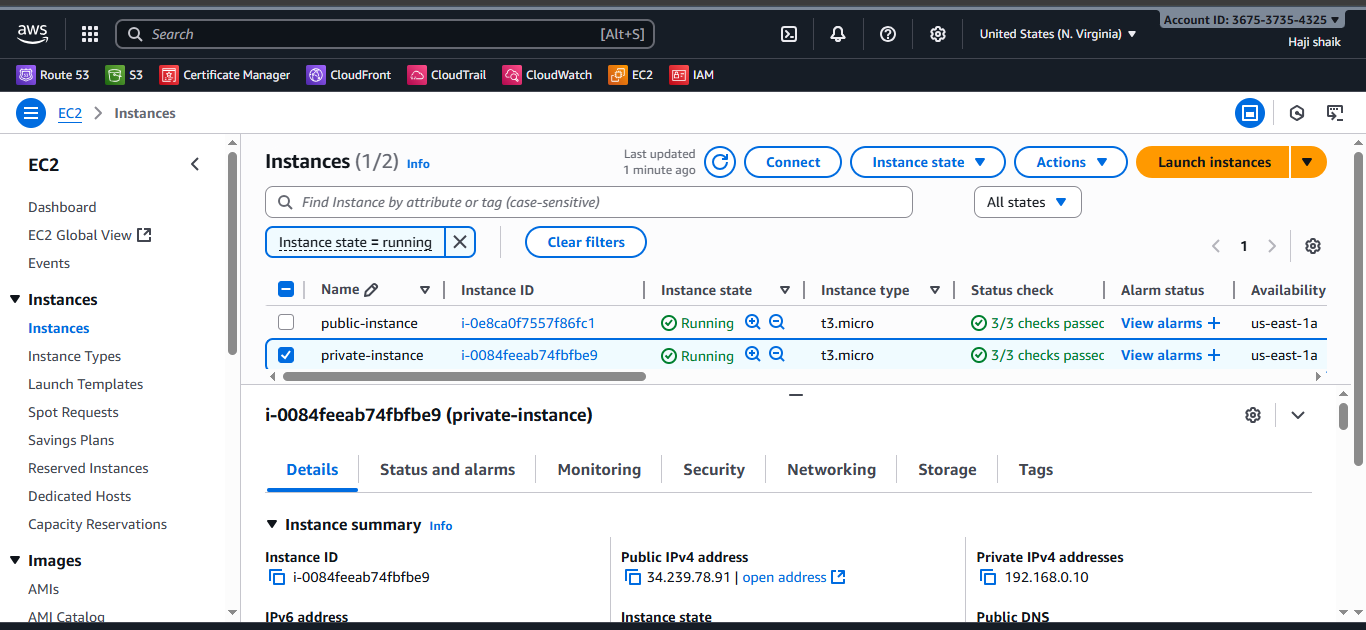
**STEP -1**

* **Go** to the ec2 instance.
* Then launch instance name with public-instance.
* Select the image amazon linux.
* Then create t3.micro (free tier eligible)
* Then select vpc nd in subnet level gave public subnet.
* And enable ip (imp\*)
* Then launch a unstance.

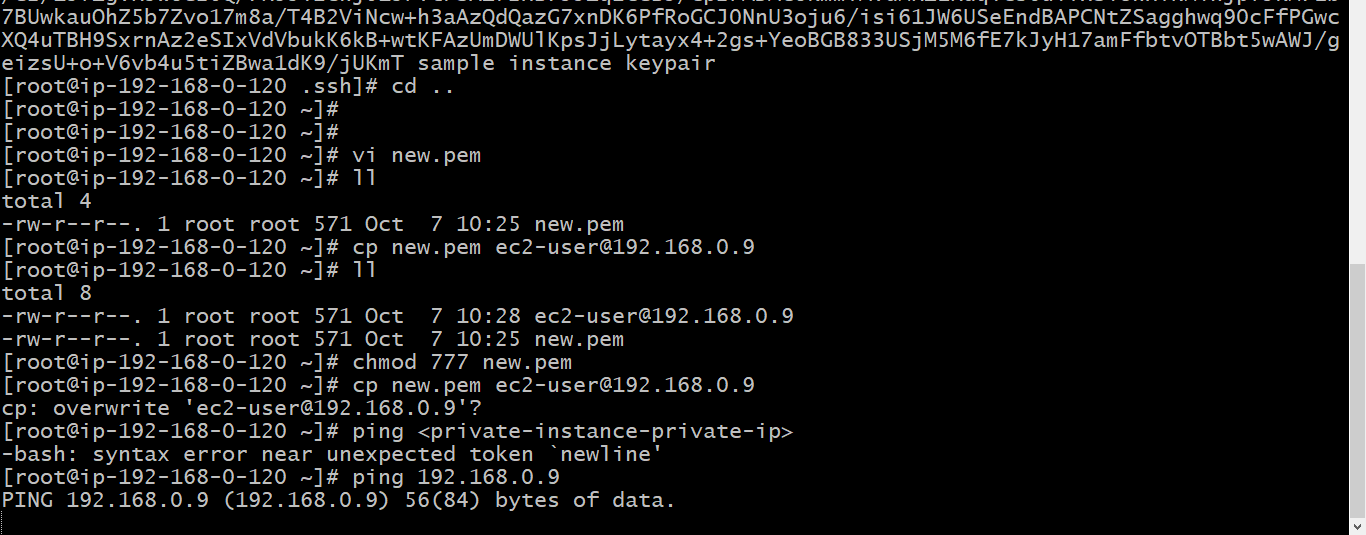


* **Go** to the ec2 instance.
* Then launch instance name with public-instance.
* Select the image amazon linux.
* Then create t3.micro (free tier eligible)
* Then select vpc nd in subnet level gave private subnet.
* And disable ip (imp\*)
* Then launch a unstance.

**STEP -2**

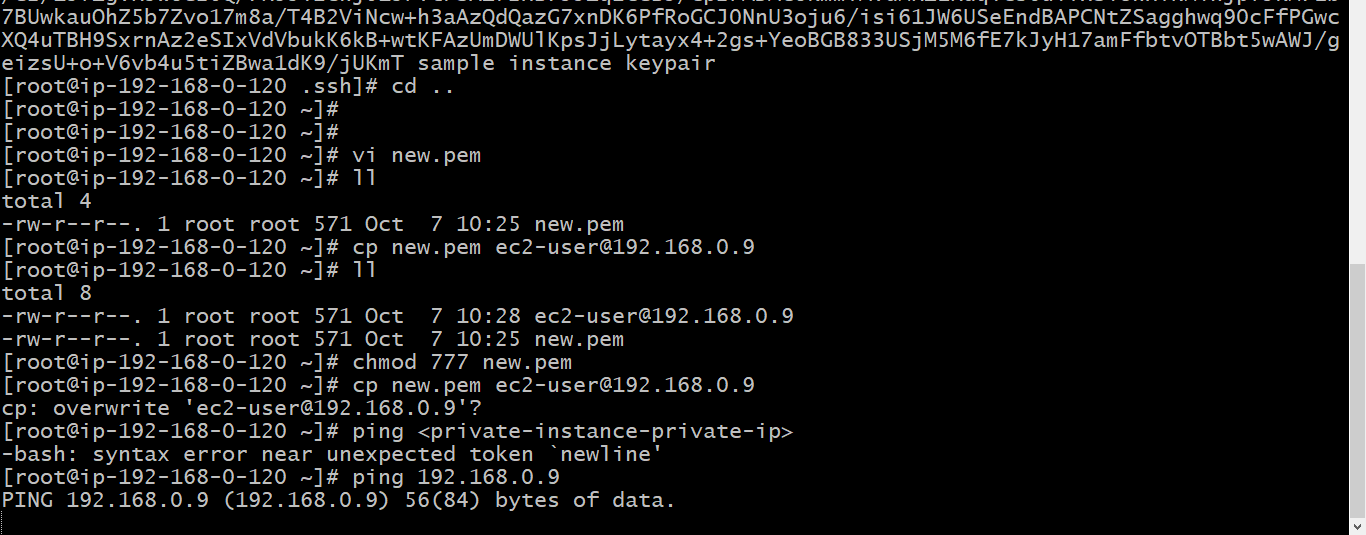


* Then go to git bash nd connect to the server. With public instance.
* Then ls, .ssh and Authorized keys then cat pem.
* Then vi new.pem then gave a permissons to file.
* Then use a command of **ssh new.pem ec2-user@private instance private ip.**
* **Then ping private instance private ip.**

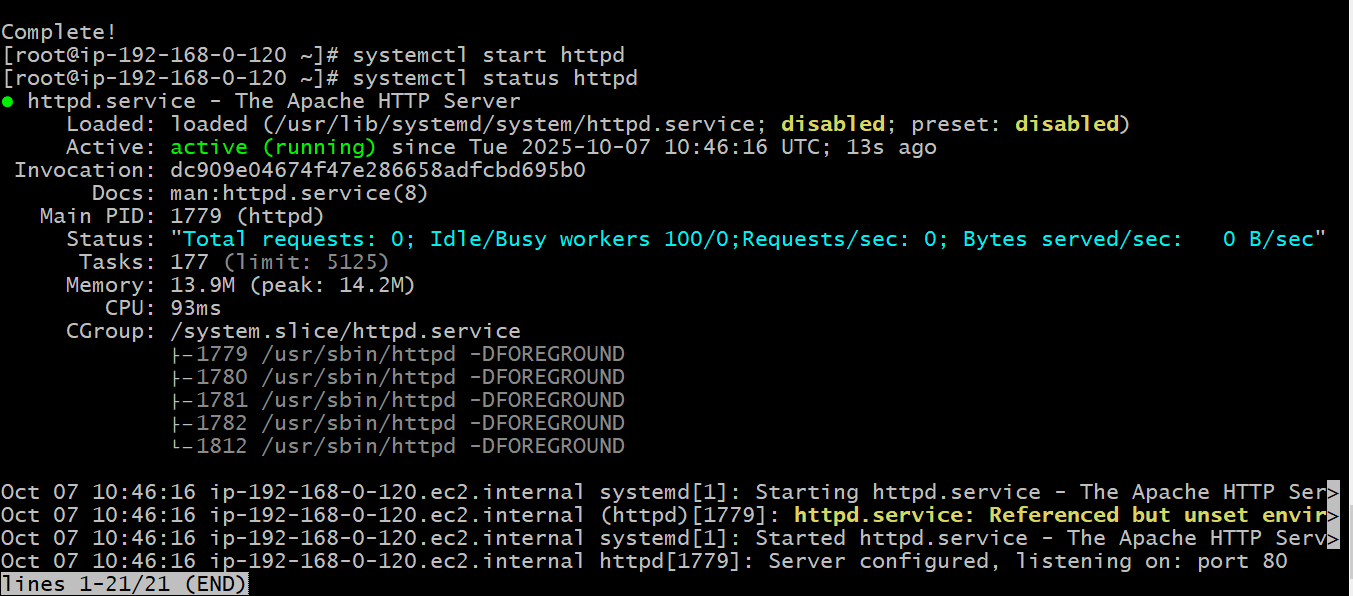


1. **Deploy Apache server on both EC2 instances with a sample index.html file.**

* **Create 2 instances.**
* **One in private one in public instances.**
* Then go to git bash nd connect to the server. With public instance.
* Then ls, .ssh and Authorized keys then cat pem.
* Then vi new.pem then gave a permissons to file.
* Then use a command of **ssh new.pem ec2-user@private instance private ip.**
* **Then ping private instance private ip.**

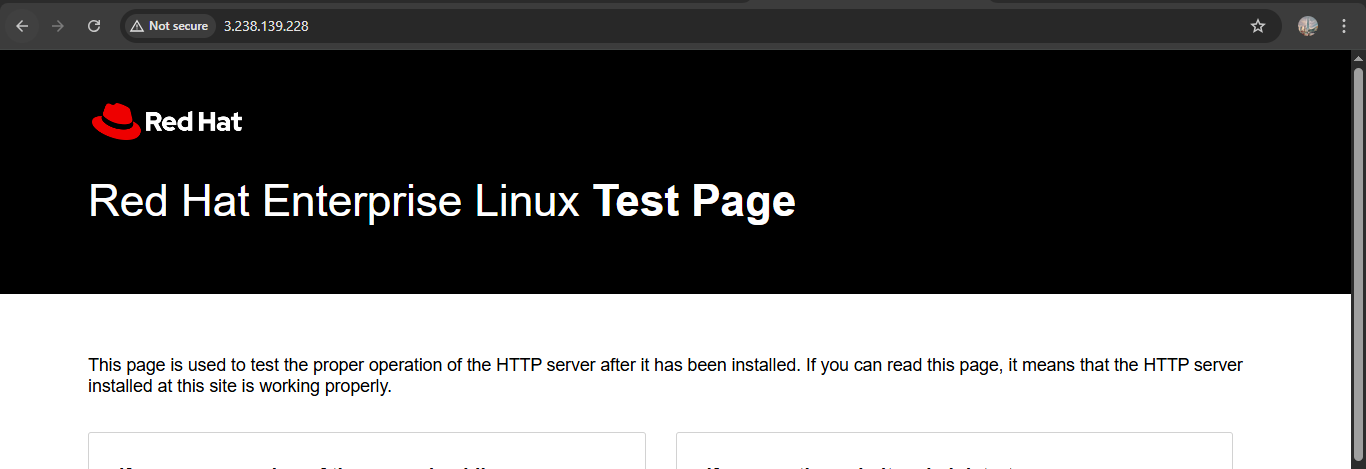


* After install httpd check status.
* curl localhost:80
* This For private instance.



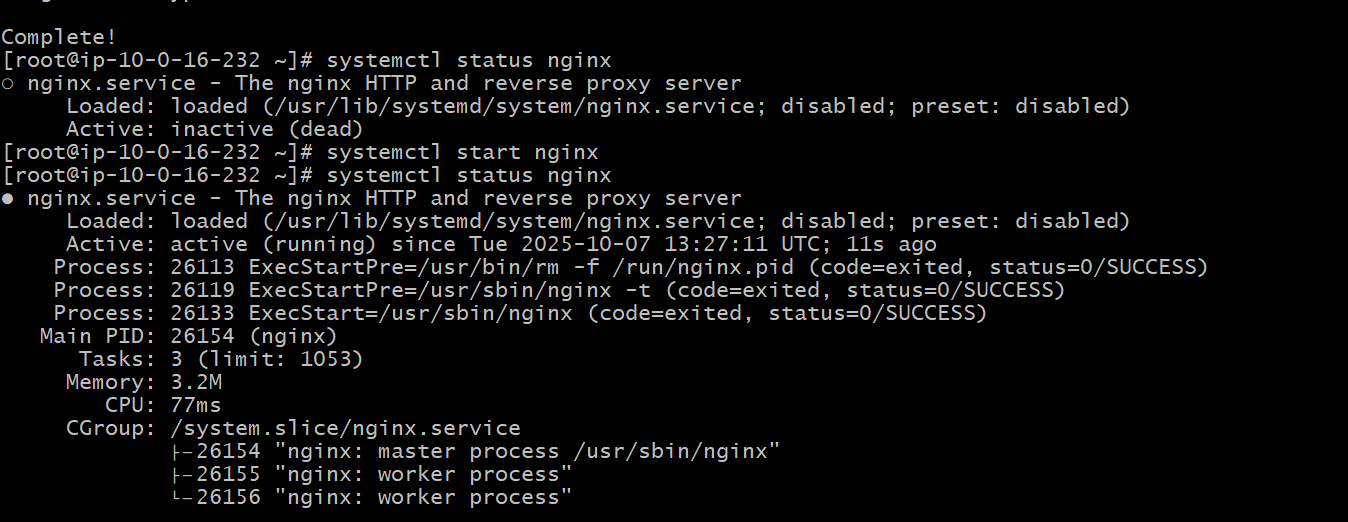


* For public instance.
* Use public ip nd :80 in browser here the results are.

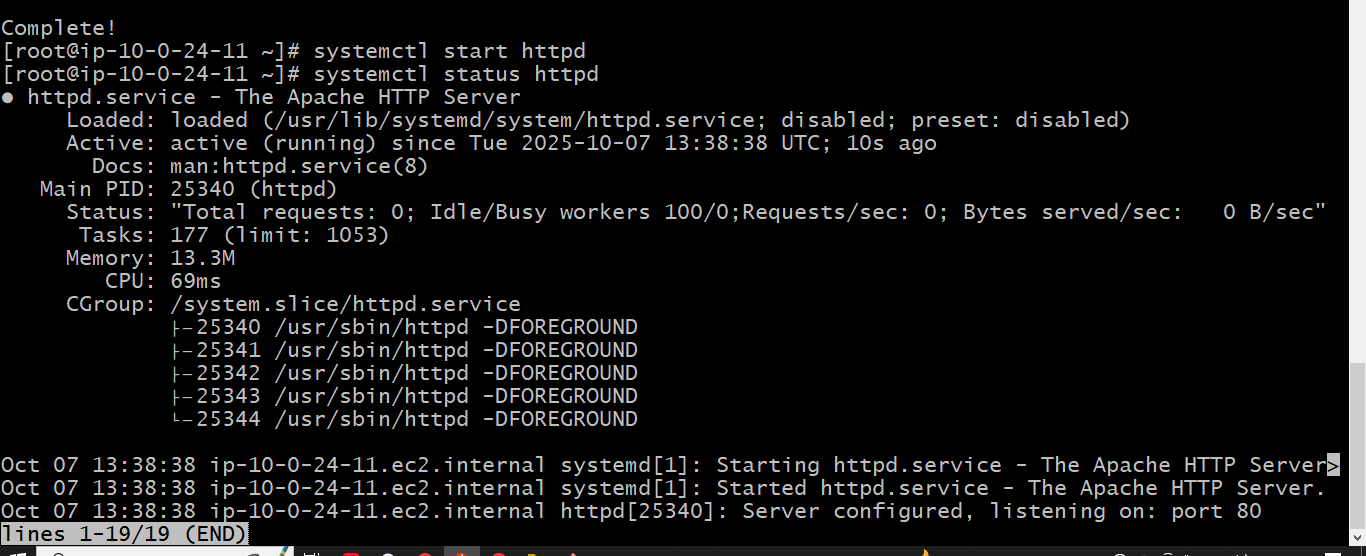


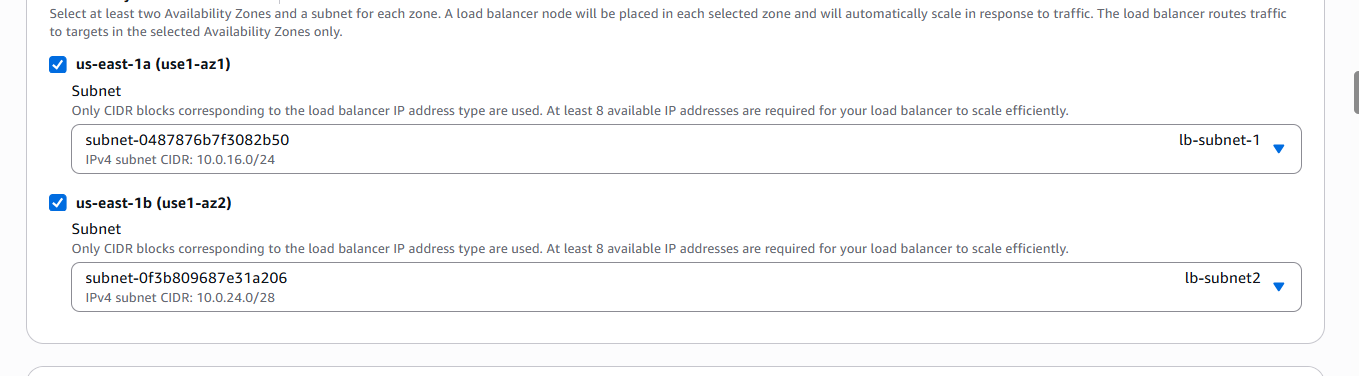
1. **Create one application load balancer and attach it to both EC2 instances.**

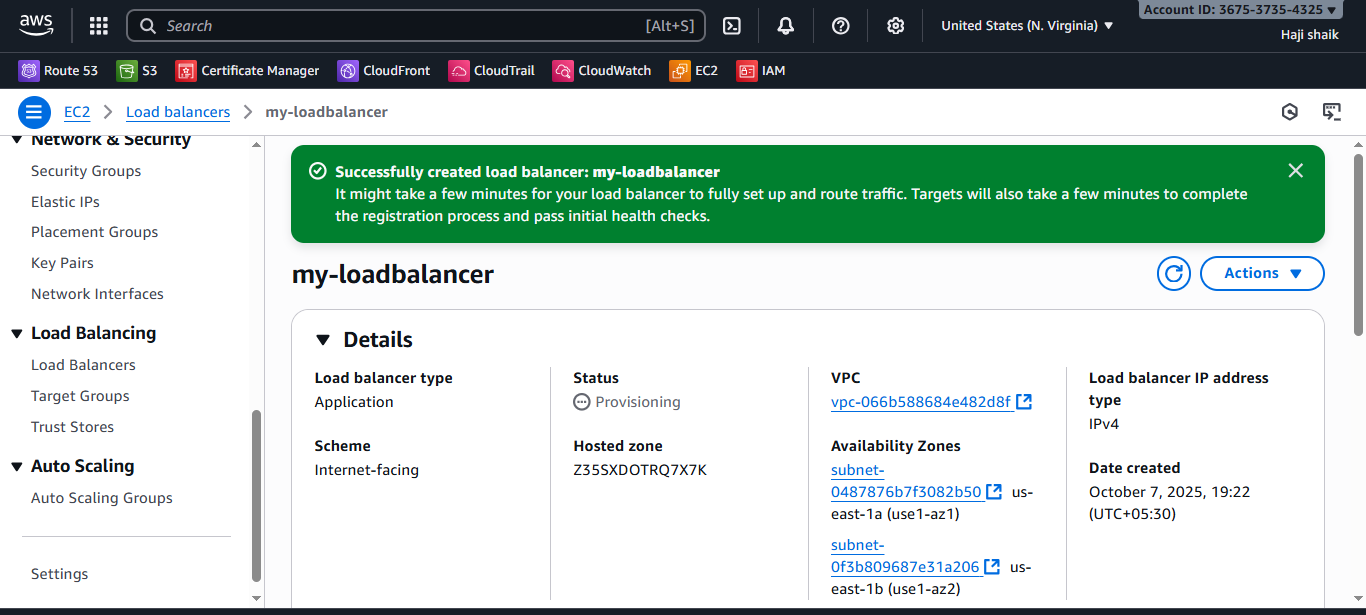
* **Instance for us-east-1a**

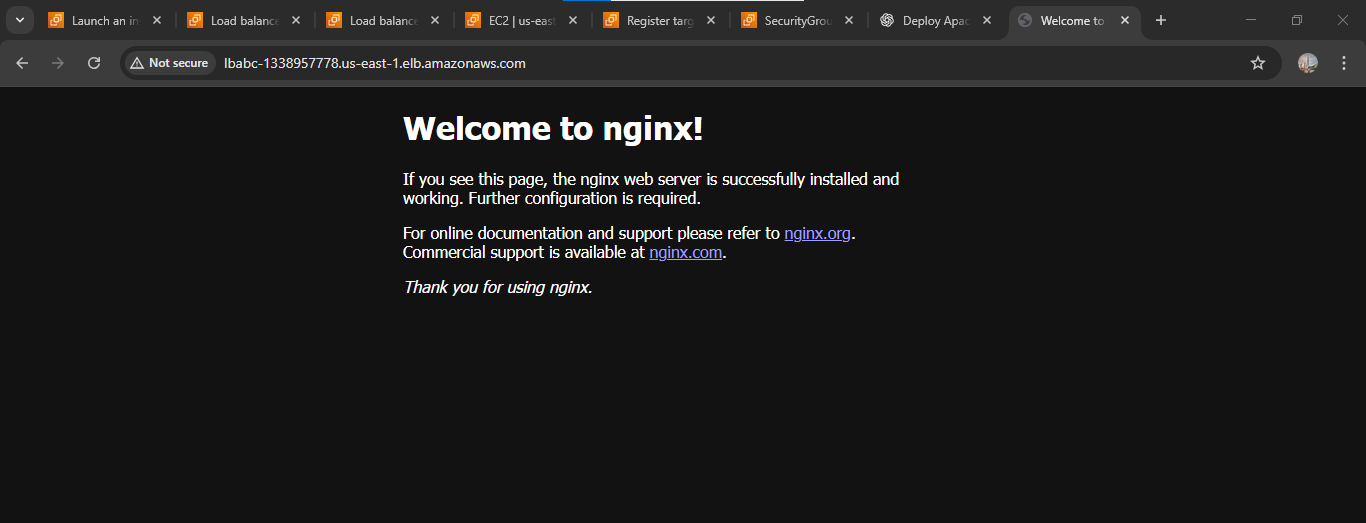


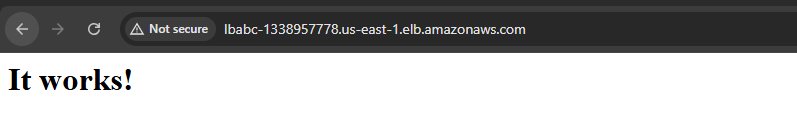
* **Instance for us-east-1b**





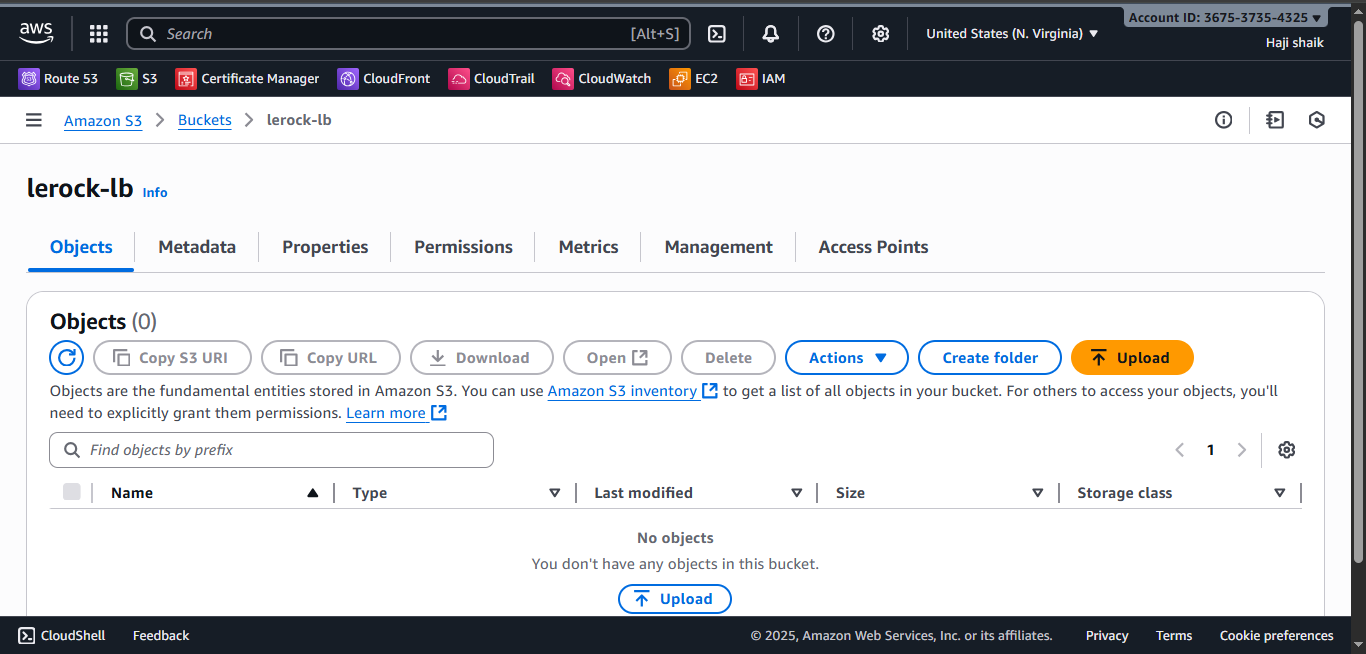




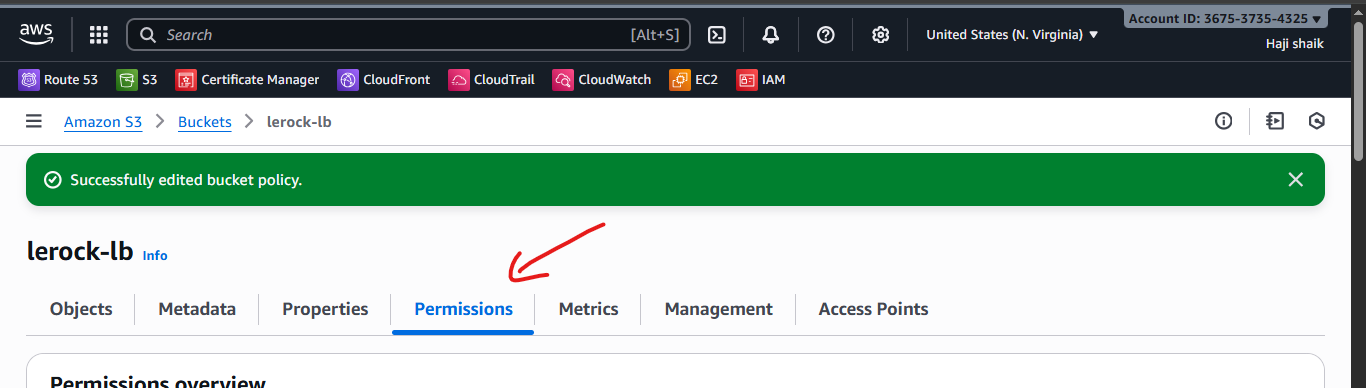


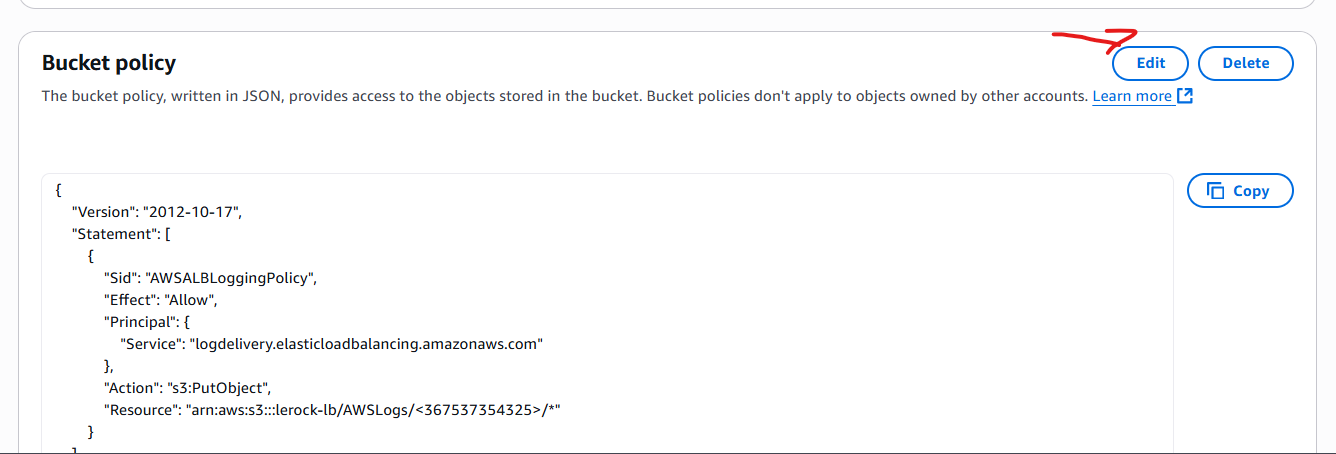
1. **Store application load balancer logs in S3.**

* **Go**  to the aws console.
* Then open search bar and go to the s3 bucket.
* Nd create a s3 bucket with a unique name.

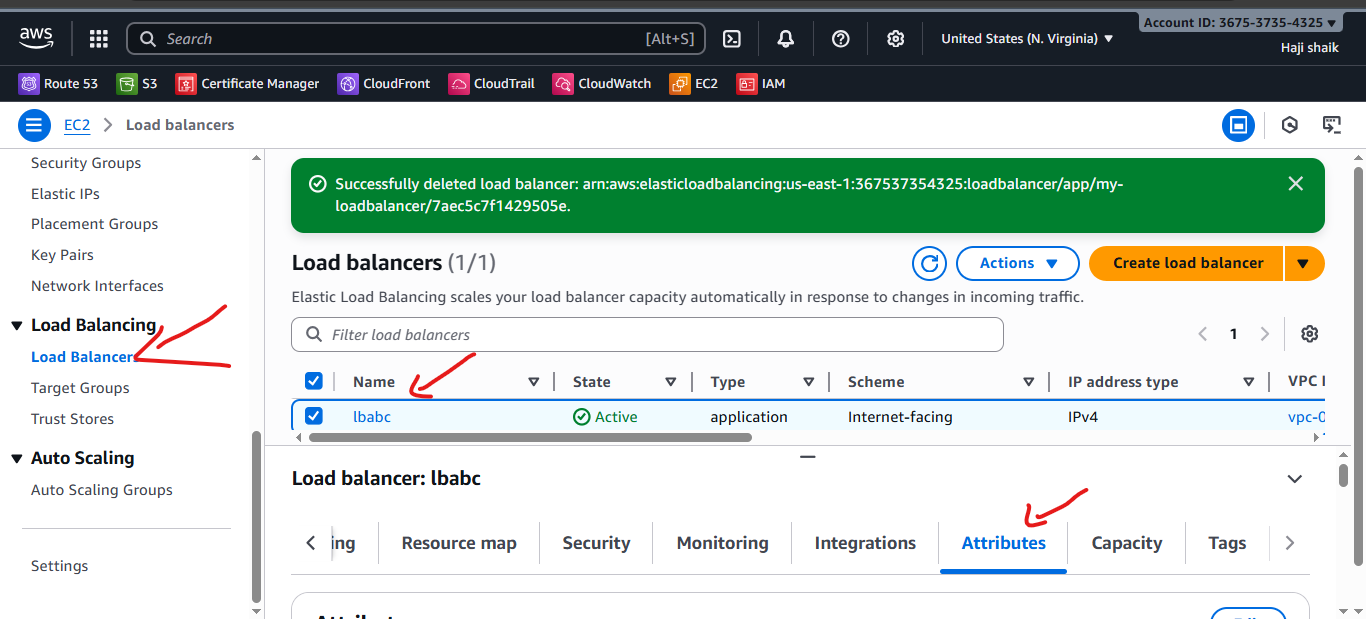


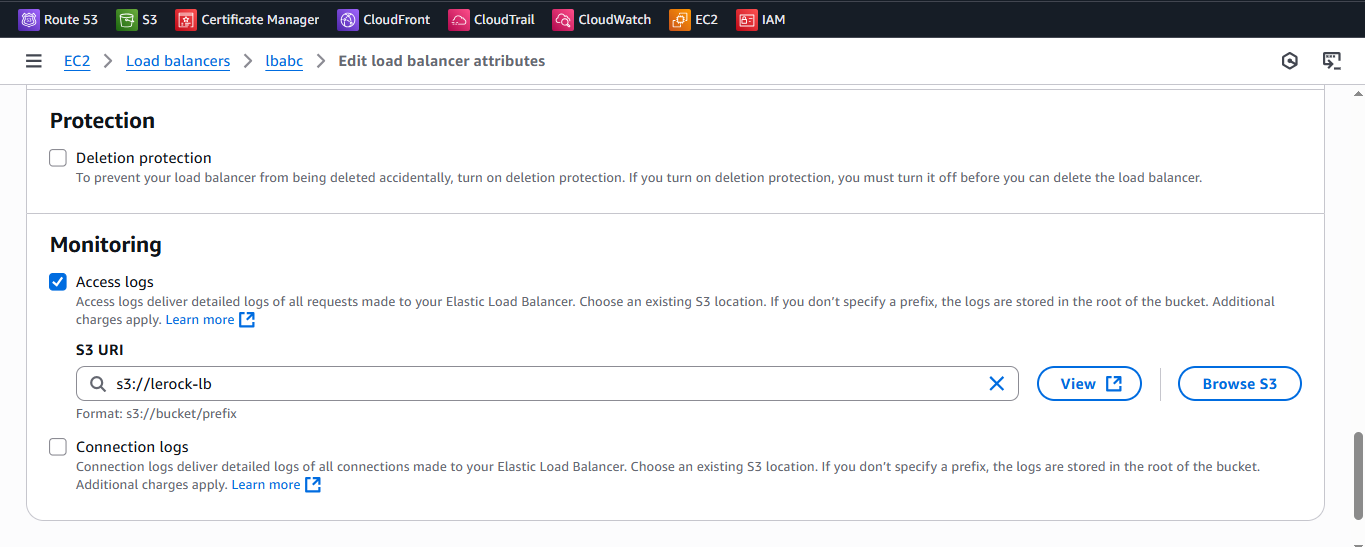
* **Go**  to the permissons in s3 bucket.
* Then edit a policy and use refernce from google.
* Here the script are.
* {  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Sid": "AWSALBLoggingPermissions",  
        "Effect": "Allow",  
        "Principal": {  
          "Service": "[logdelivery.elasticloadbalancing.amazonaws.com](http://logdelivery.elasticloadbalancing.amazonaws.com/" \t "https://app.slack.com/client/T03TRQ064Q0/_blank)"  
        },  
        "Action": "s3:PutObject",  
        "Resource": "arn:aws:s3:::<BUCKET\_NAME>/AWSLogs/<ACCOUNT\_ID>/\*"  
      }  
    ]  
  }



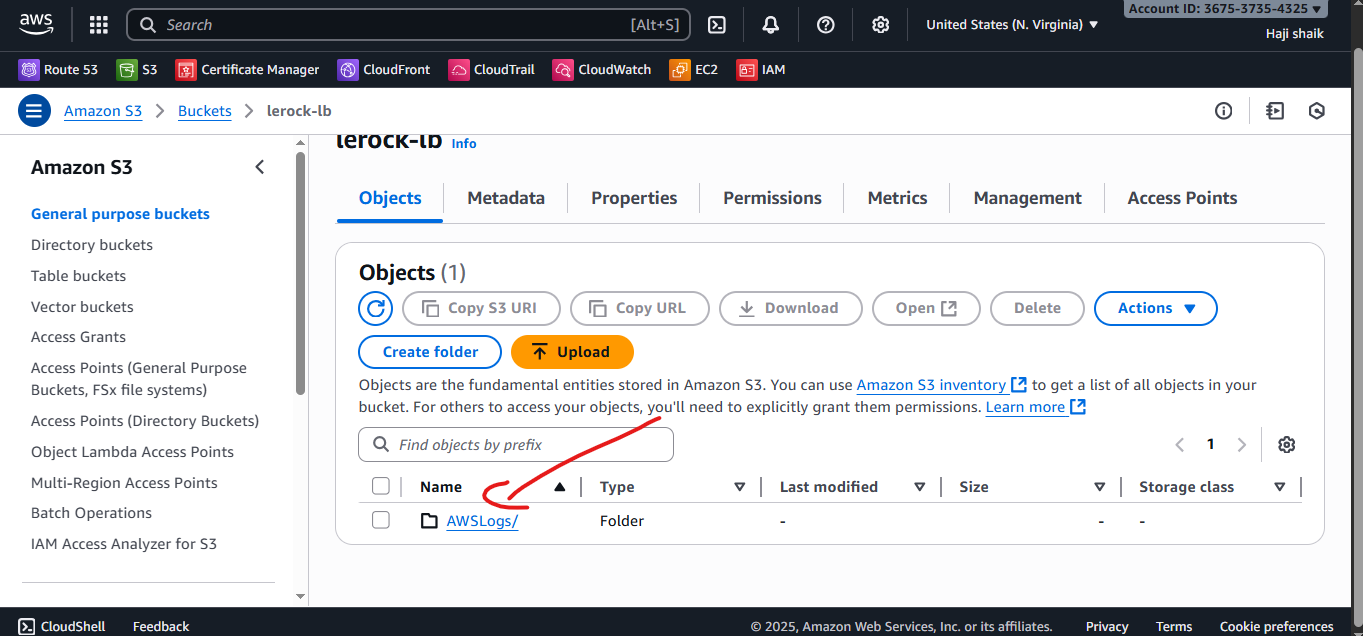


* **Go**  to the load balncer again.
* Go the **Attributes**.
* Then in bottom there is Monitoring Option.
* Enable access log option there choose your **S3 bucket.**





* Then **go back to the S3 bucket refresh the page.**
* **Open a object bar here the results are:**



1. **Store the VPC flow logs in a CloudWatch log group.**

**STEP-1**

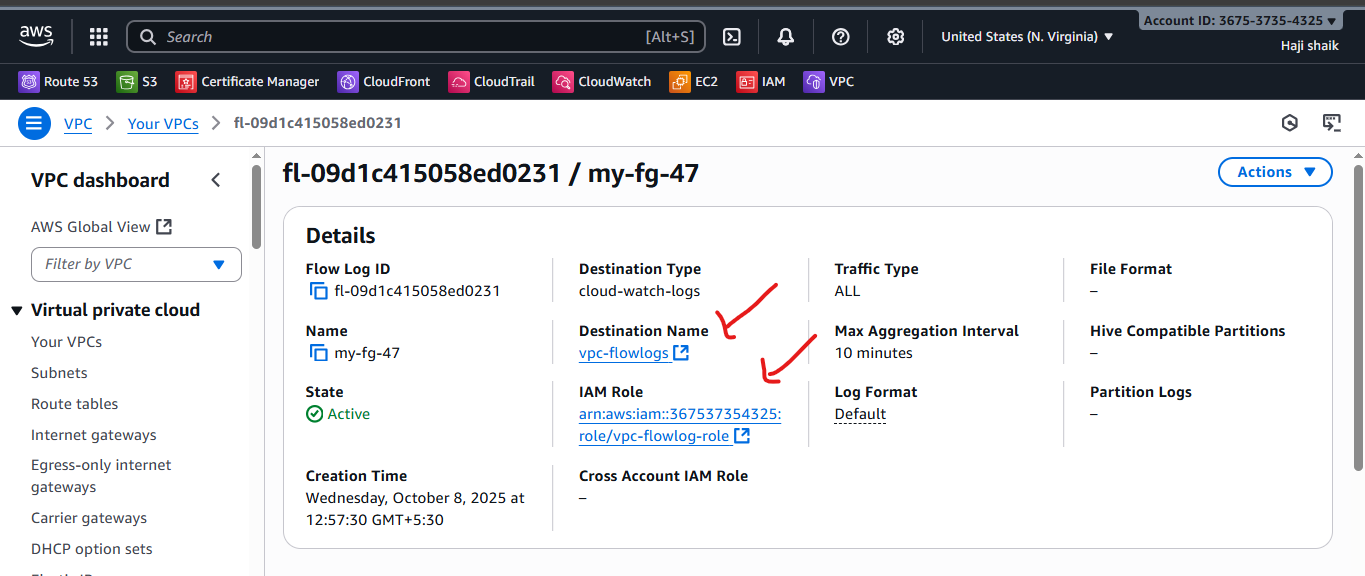
* Go to **AWS Management Console → VPC**
* In the left menu, select **Your VPCs**
* Choose the **VPC** you want to enable flow logs for
* Click on the **“Flow Logs”** tab
* Click **“Create flow log”**
* Click **“Set up permissions” → “Create new role”**
* It will automatically attach the policy:(AWSFlowLogsDeliveryPolicy)
* Save the role (e.g., name it vpc-flowlogs-role

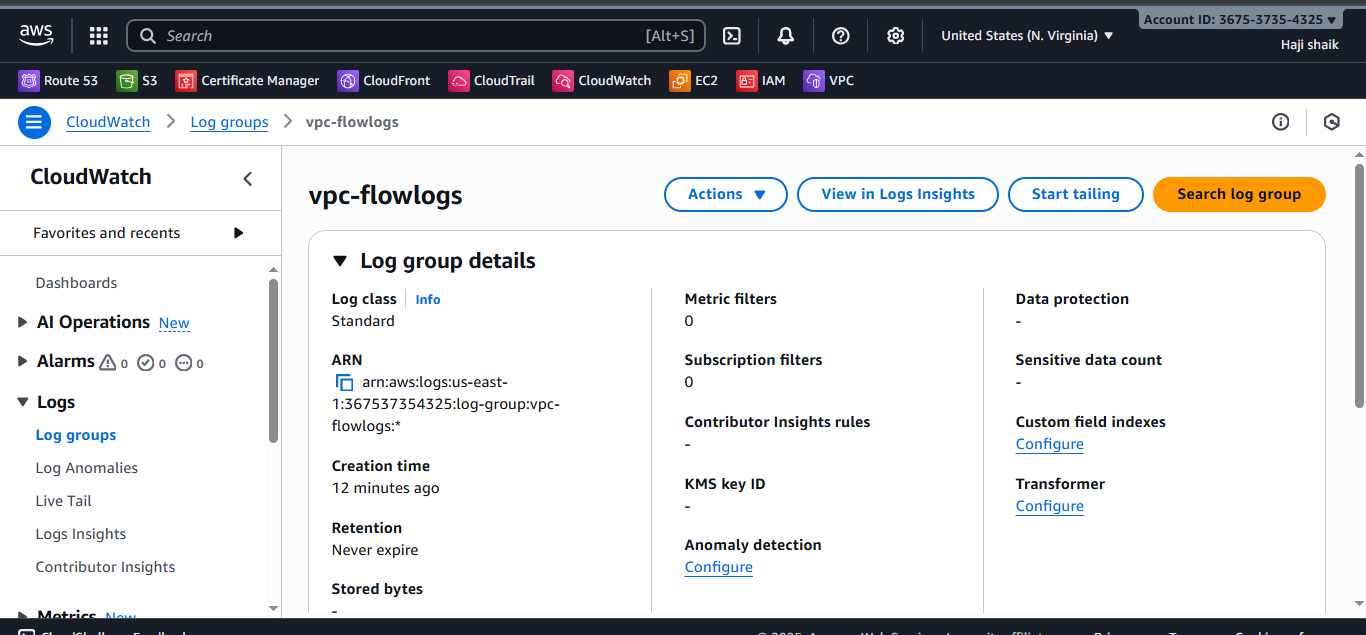
**STEP-2**

* Go to **CloudWatch → Logs → Log groups**
* Click **Create log group**
* Name it /vpc/flowlogs

**STEP-3**

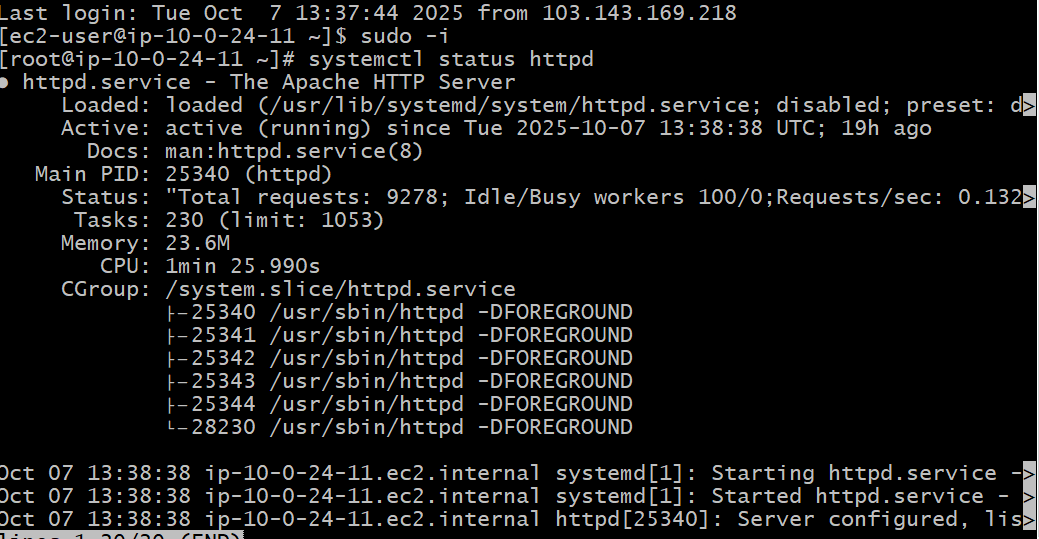
* Go back to the VPC console
* Select: **Destination** → CloudWatch Logs , **Log group name** → /vpc/flowlogs
* **IAM role** → vpc-flowlogs-role
* Go to **CloudWatch → Logs → Log groups → /vpc/flowlogs**



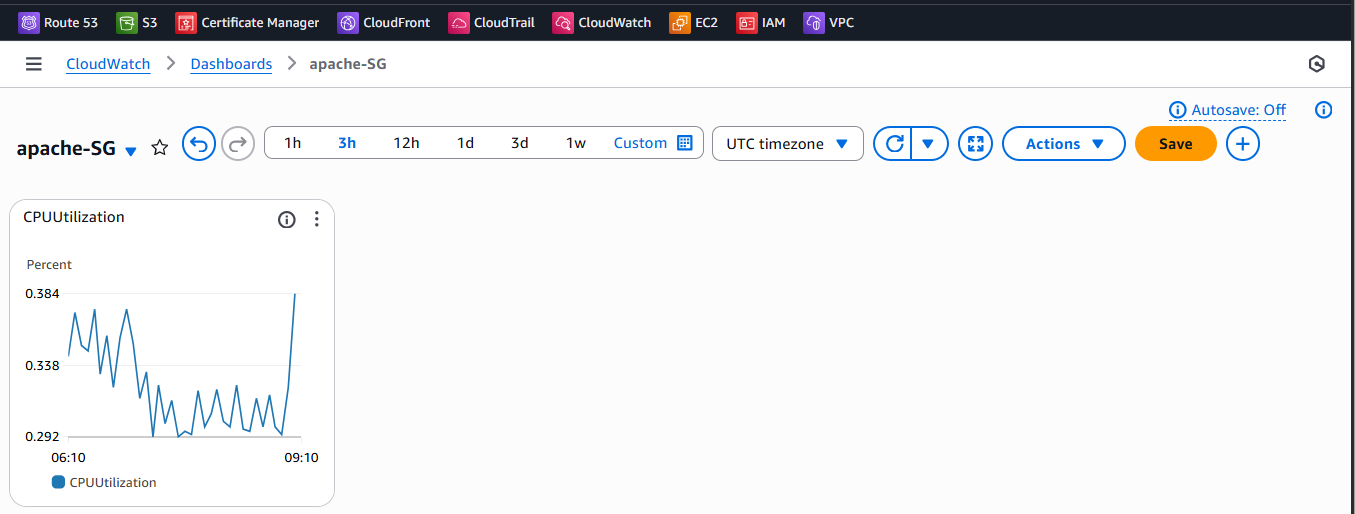


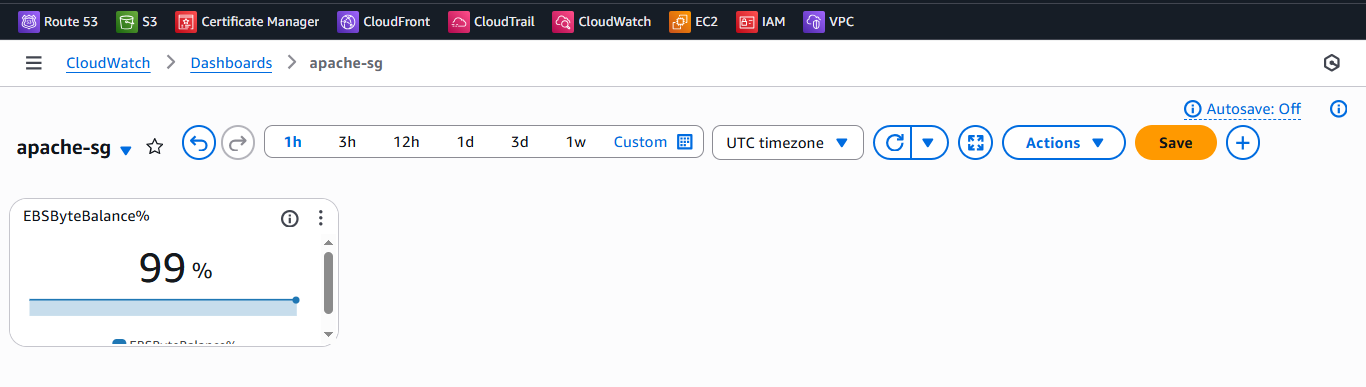
1. **Create monitoring dashboards to monitor CPU utilization and to monitor the Apache service.**

* **Go** to AWS console.
* Then create instance nd install **HTTPD.**



* **Go** to cloud watch nd create a dashboard.
* With name of apache-sg.
* Then use line & Number to diffrent widget nd take in to ur instances.
* Then copy of instance id.
* Select the instance.
* Then here the results are.





1. **If CPU utilization is more than 70%, then it should trigger auto scaling and launch new instance**.

