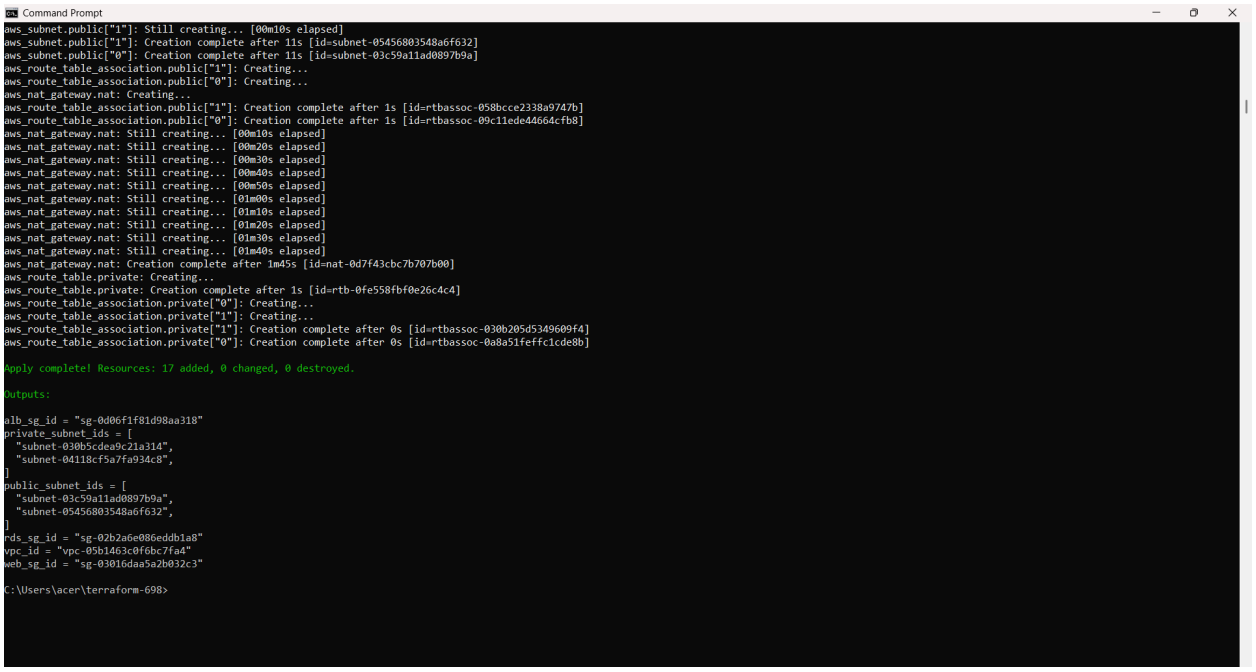
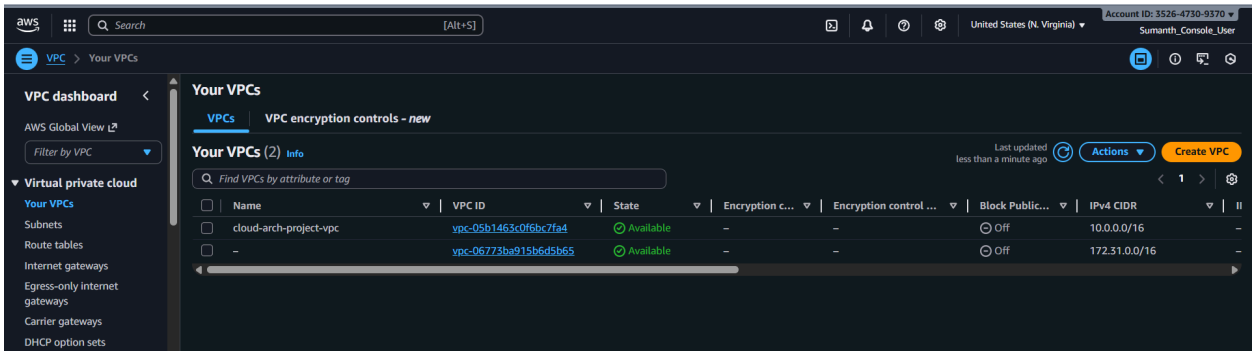
IS698 - DC93457

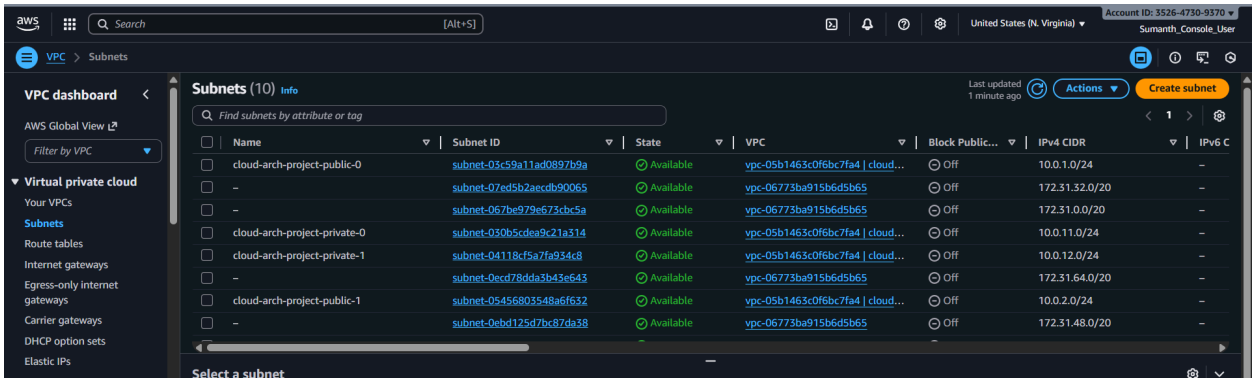
Project: Deploying a Scalable AWS Architecture with Infrastructure as Code

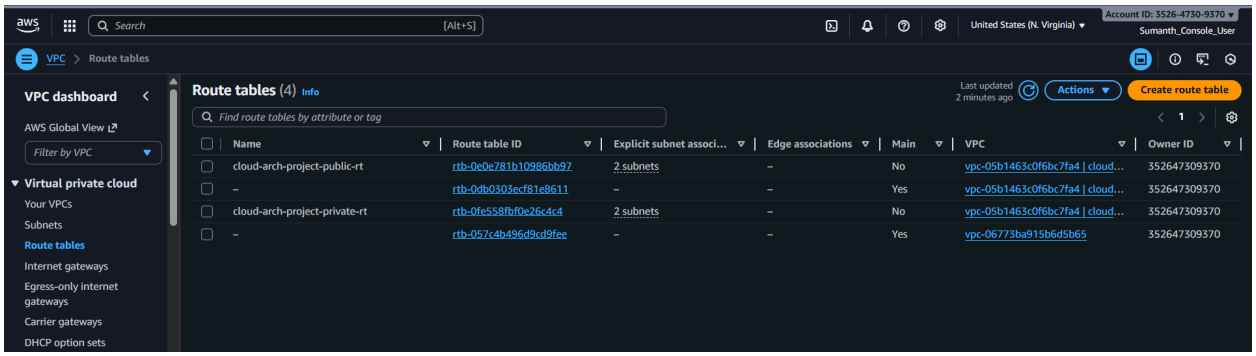
A. Infrastructure Deployment

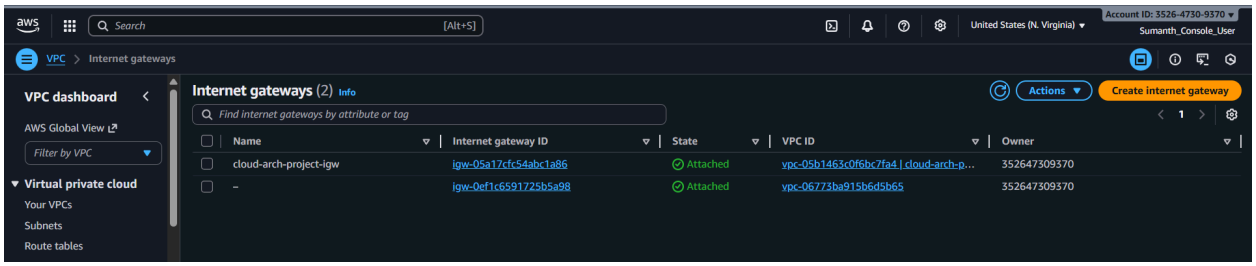
Terraform

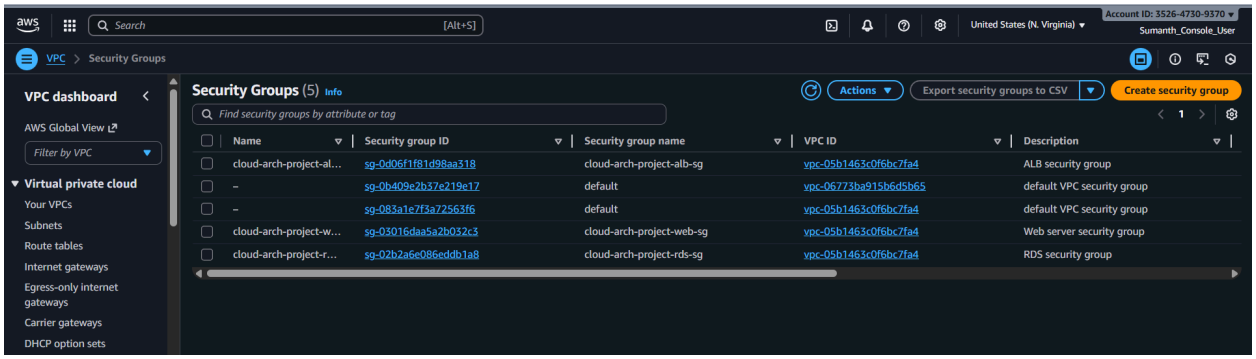






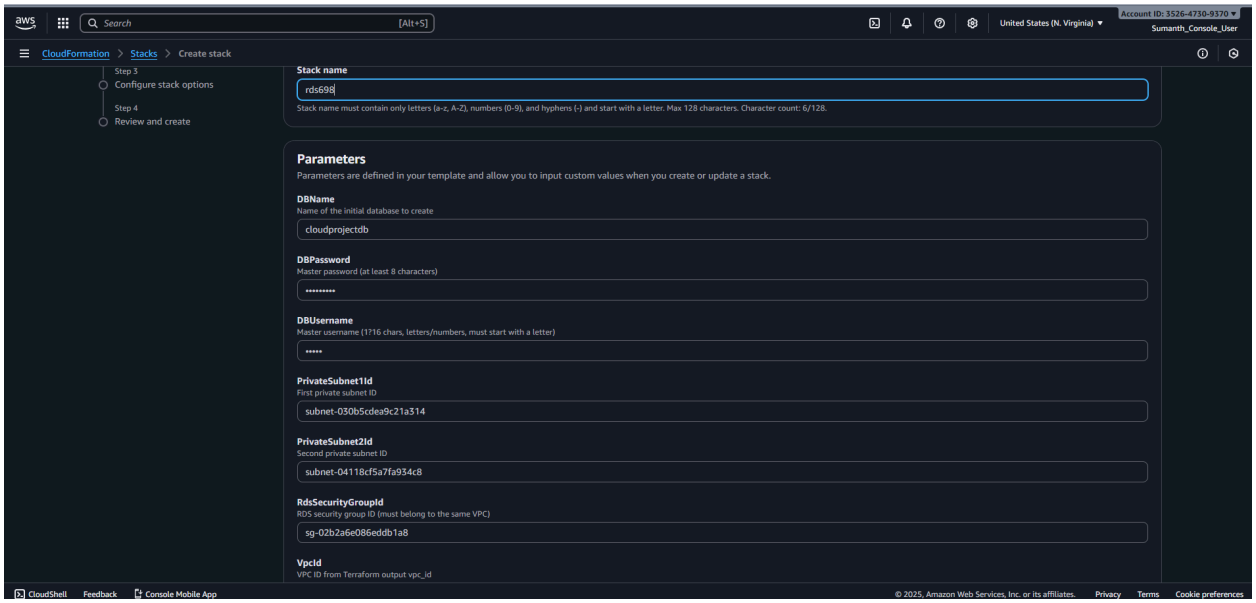


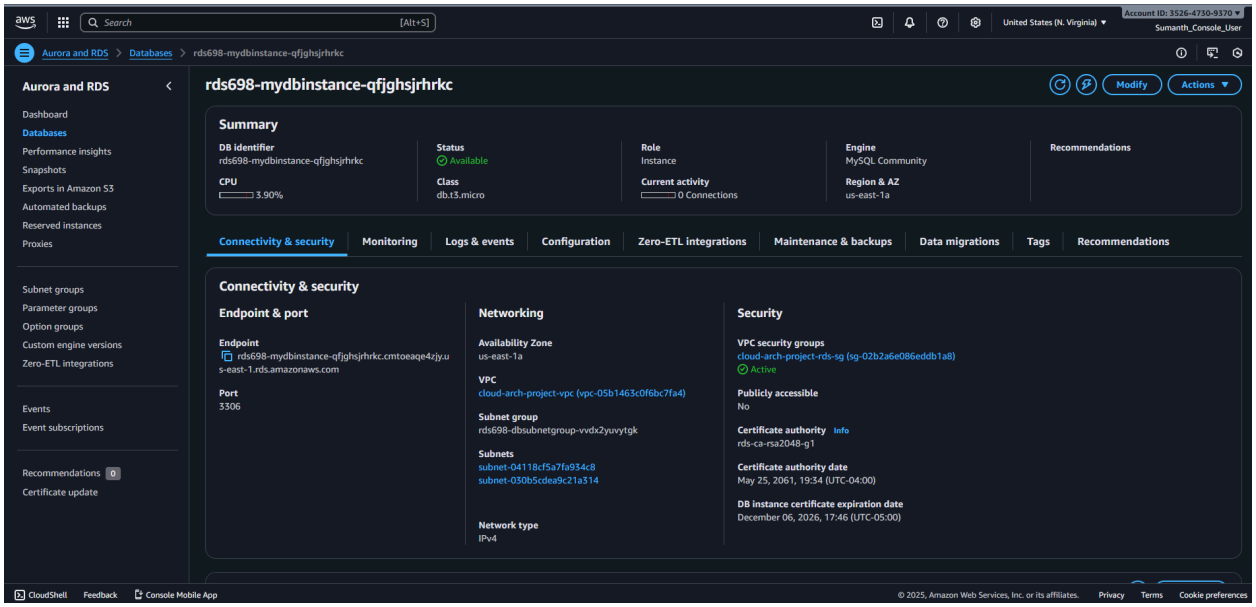


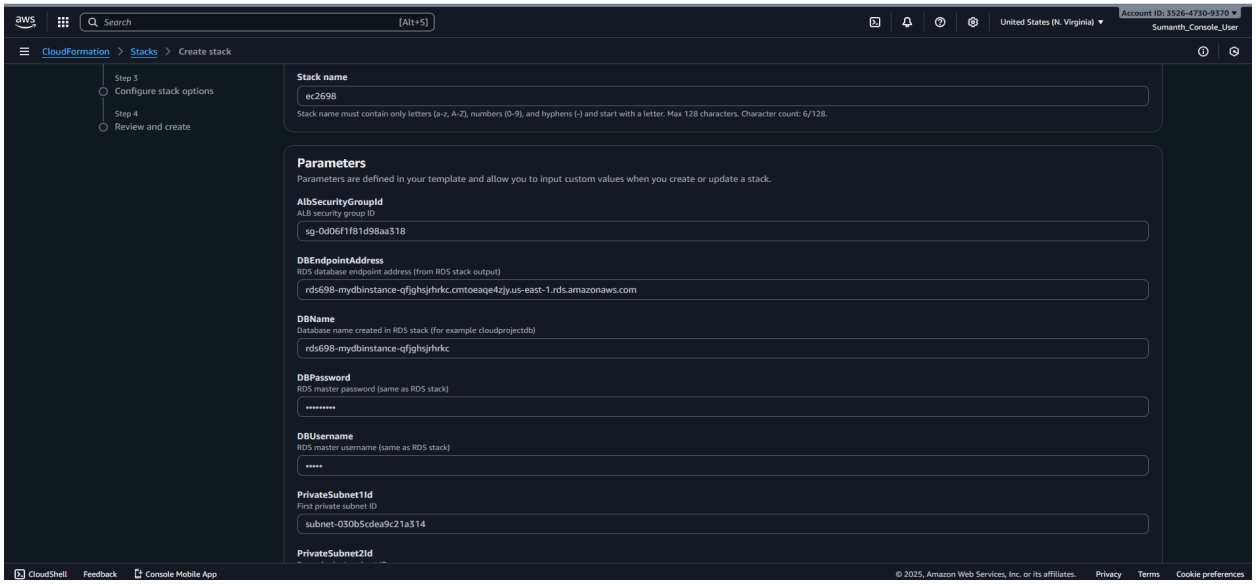


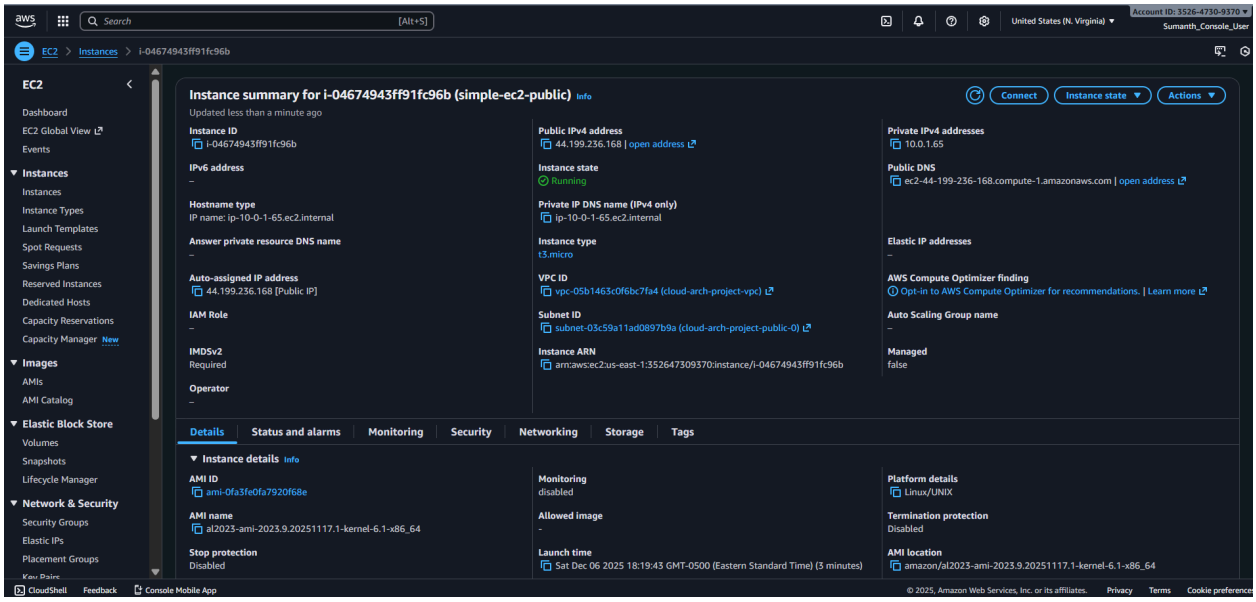
In this section, I used Terraform to build the core networking layer for the project. Terraform automatically created my VPC, subnets, route tables and security groups exactly the way I defined them in the code.Instead of manually creating networking resources,Terraform handled all of it in one shot.

Cloudformation







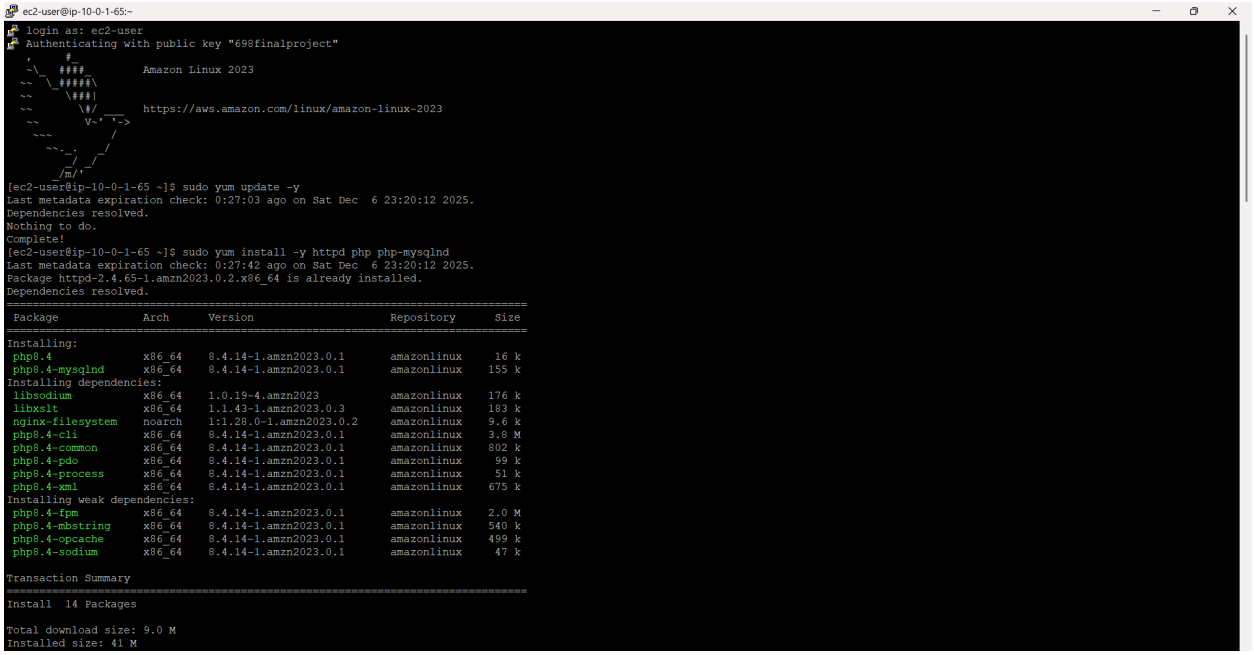


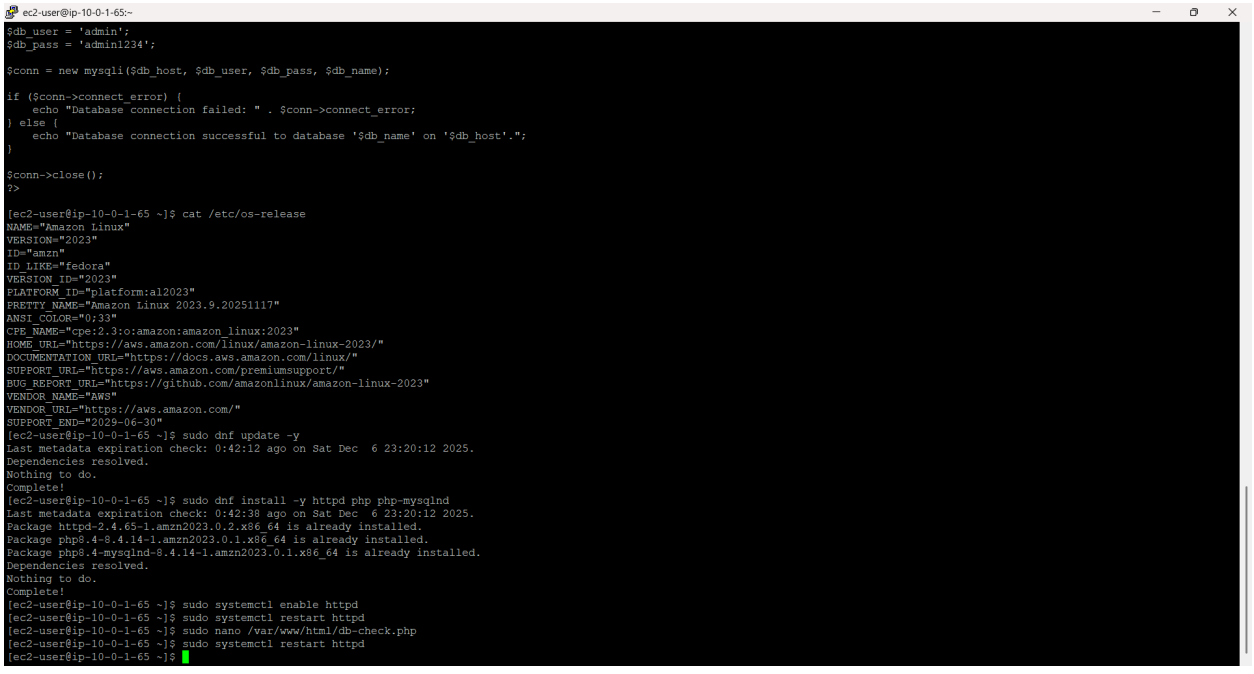
Hello from a simple EC2 instance!

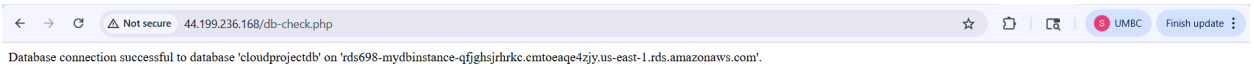
This instance was created using a basic CloudFormation template in a public subnet.

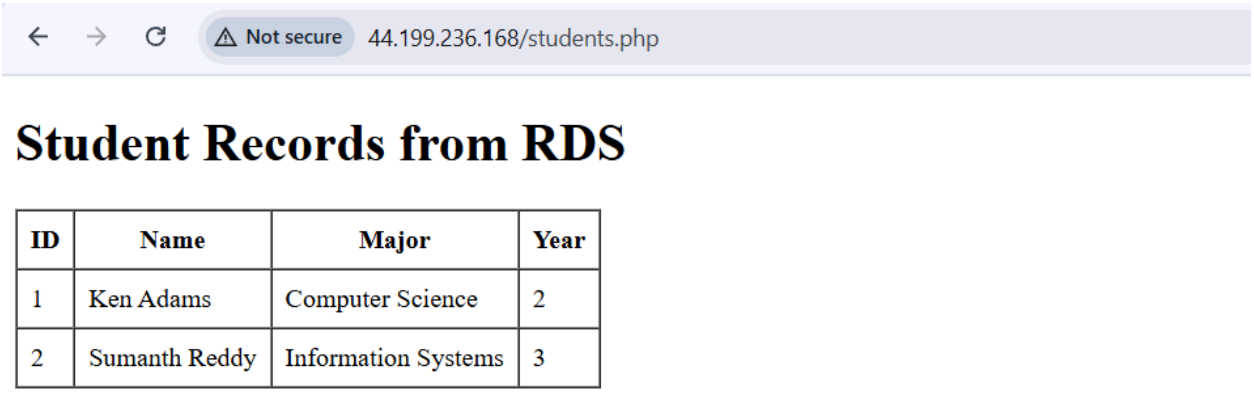
This part covers the CloudFormation stacks I deployed for EC2 and RDS. Each YAML file defined a specific resource. CloudFormation read the template and created the servers for me. Also it created a static web page for the EC2 instance.

Configure database with application





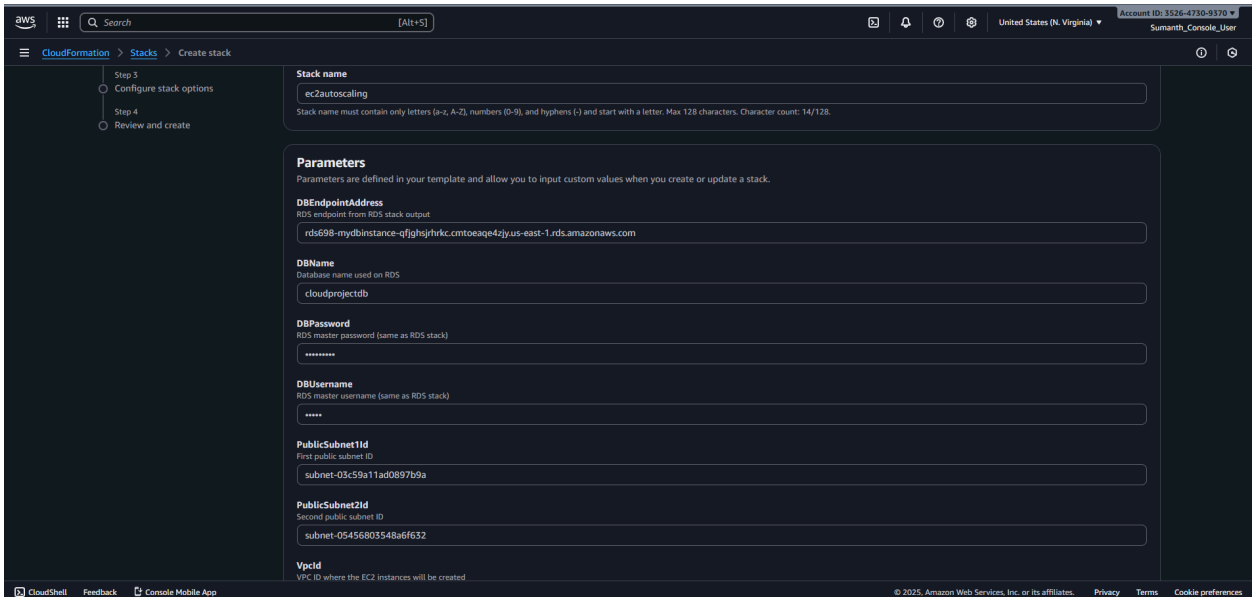


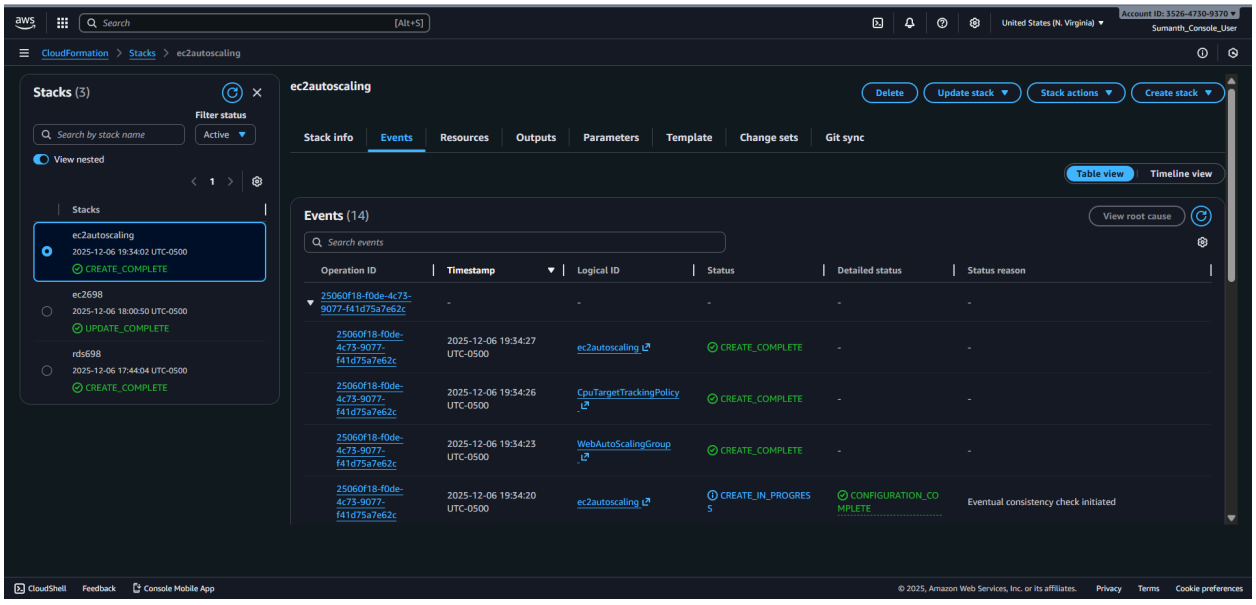


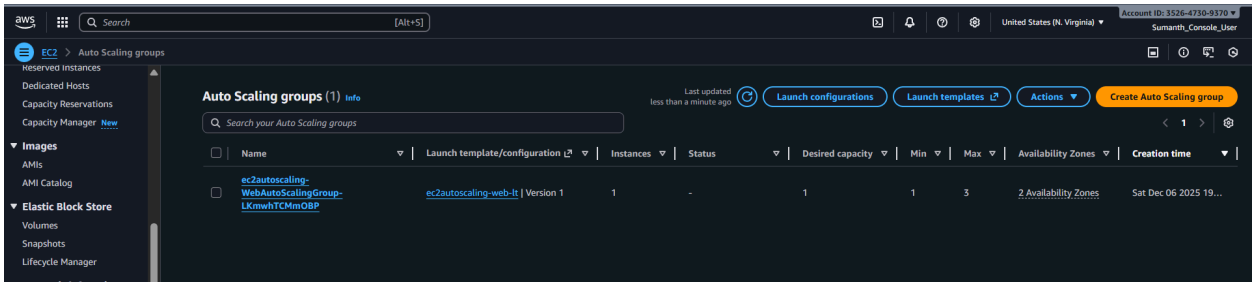
In this step I connected my EC2 web server to the RDS database. I wrote a small simple PHP script (db-check.php) using SSH and uploaded it to /var/www/html/. The script tries to connect to RDS using the endpoint, username, and password. Testing this in the browser showed “Database connection successful” which confirms that EC2 and RDS databases are connected.

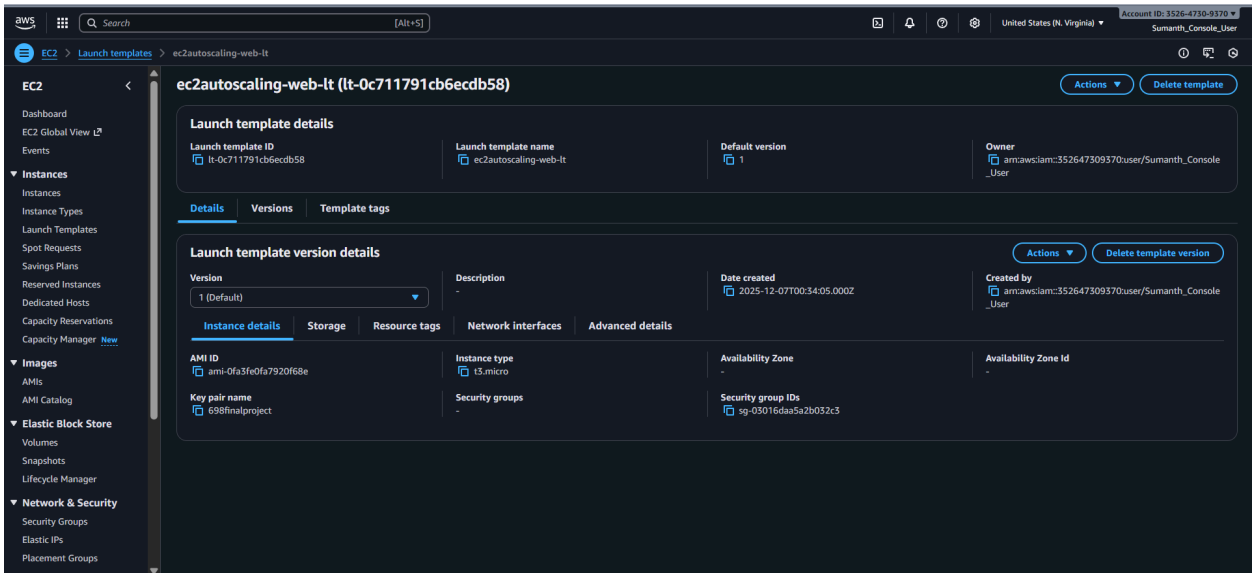
After the successful connection using MySql in SSH, I created a table ‘Students’ in database and inserted student values. I created a new php script called ‘students.php’ where it will connect the database to the dynamic web page. When opened it with EC2 instance then, I see 2 student records in the web page.

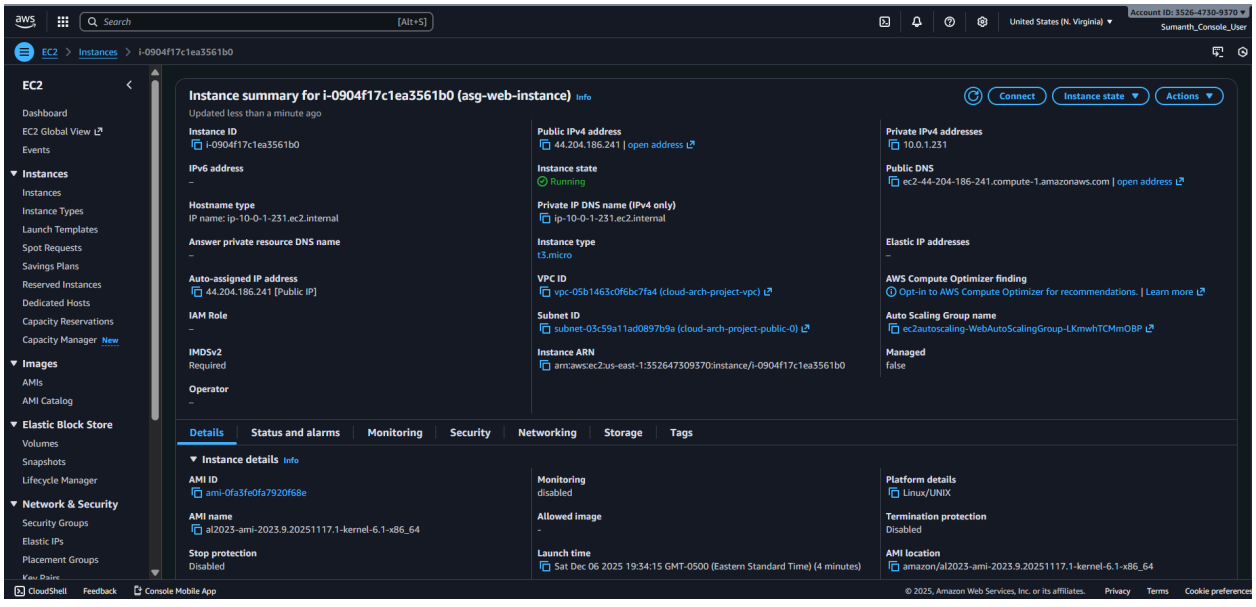
Autoscaling






Auto Scaling groups (1) Info
Launch configurations
Launch templates
Actions
Create Auto Scaling group
ec2autoscaling-WebAutoScalingGroup-LKmwhTCMmOBP
ec2autoscaling-web-lt | Version 1
2 Availability Zones
Sat Dec 06 2025 19...


ec2autoscaling-web-lt (lt-0c711791cb6ecdb58)
Actions
Delete template
Launch template details
Launch template ID
lt-0c711791cb6ecdb58
Launch template name
ec2autoscaling-web-lt
Default version
1
Owner
arn:aws:iam::352647309370:user/Sumanth_Console_User
Details
Versions
Template tags
Launch template version details
Actions
Delete template version
Version
1 (Default)
Description
Date created
2025-12-07T00:34:05.000Z
Created by
arn:aws:iam::352647309370:user/Sumanth_Console_User
Instance details
Storage
Resource tags
Network interfaces
Advanced details
AMI ID
ami-0fa3fe0fa7920f68e
Instance type
t3.micro
Availability Zone
Availability Zone Id
Key pair name
698finalproject
Security groups
Security group IDs
sg-03016daa5a2b032c3


Instance summary for i-0904f17c1ea3561b0 (asg-web-instance) Info
Updated less than a minute ago
Connect
Instance state
Actions
Instance ID
i-0904f17c1ea3561b0
Public IPv4 address
44.204.186.241 | open address
Private IPv4 addresses
10.0.1.231
IPv6 address
Instance state
Running
Public DNS
ec2-44-204-186-241.compute-1.amazonaws.com | open address
Hostname type
IP name: ip-10-0-1-231.ec2.internal
Private IP DNS name (IPv4 only)
ip-10-0-1-231.ec2.internal
Answer private resource DNS name
Instance type
t3.micro
Elastic IP addresses
Auto-assigned IP address
44.204.186.241 [Public IP]
VPC ID
vpc-05b1463c0f6bc7fa4 (cloud-arch-project-vpc)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role
Subnet ID
subnet-03c59a11ad0897b9a (cloud-arch-project-public-0)
Auto Scaling Group name
ec2autoscaling-WebAutoScalingGroup-LKmwhTCMmOBP
IMDSv2
Required
Instance ARN
arn:aws:ec2:us-east-1:352647309370:instance/i-0904f17c1ea3561b0
Managed
false
Operator
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance details
AMI ID
ami-0fa3fe0fa7920f68e
Monitoring
disabled
Platform details
Linux/UNIX
AMI name
al2023-ami-2023.9.20251117.1-kernel-6.1-x86_64
Allowed image
Termination protection
Disabled
Stop protection
Disabled
Launch time
Sat Dec 06 2025 19:34:15 GMT-0500 (Eastern Standard Time) (4 minutes)
AMI location
amazon/al2023-ami-2023.9.20251117.1-kernel-6.1-x86_64

Not secure 44.204.186.241

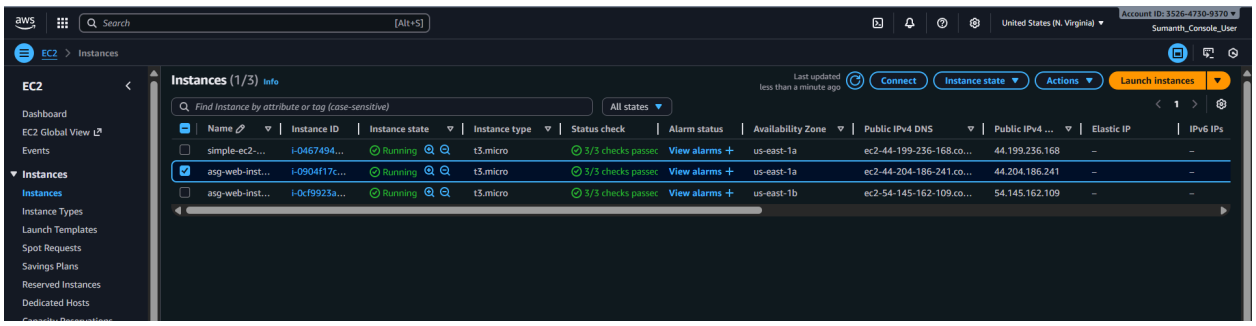
Hello from an Auto Scaling EC2 instance!

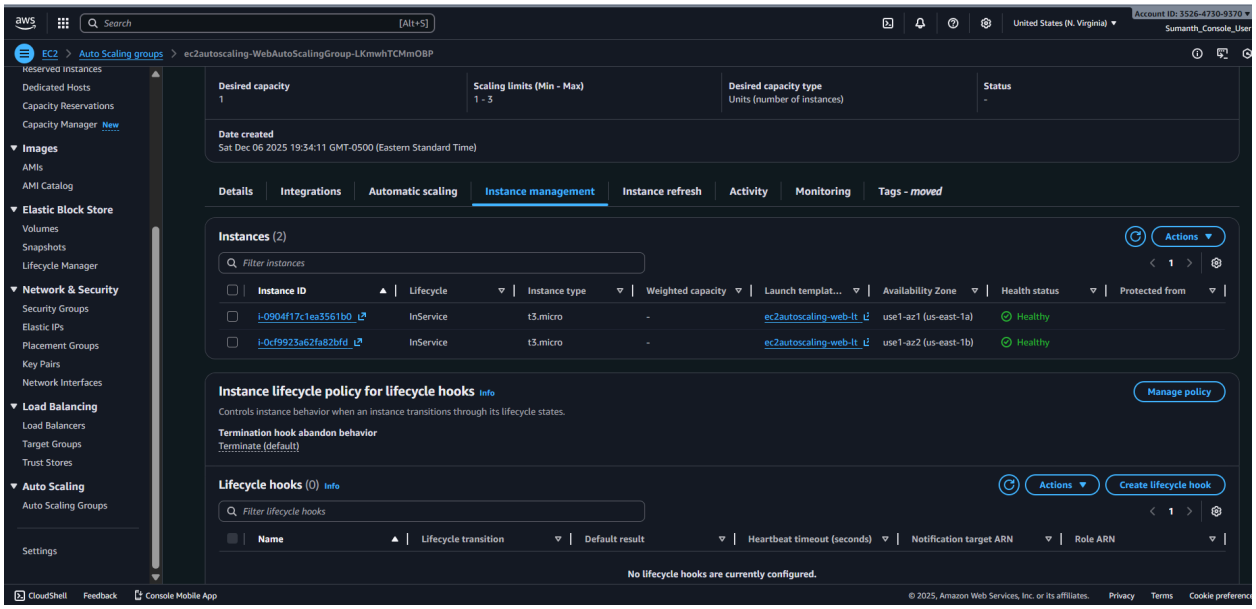
This instance was launched by an Auto Scaling Group.

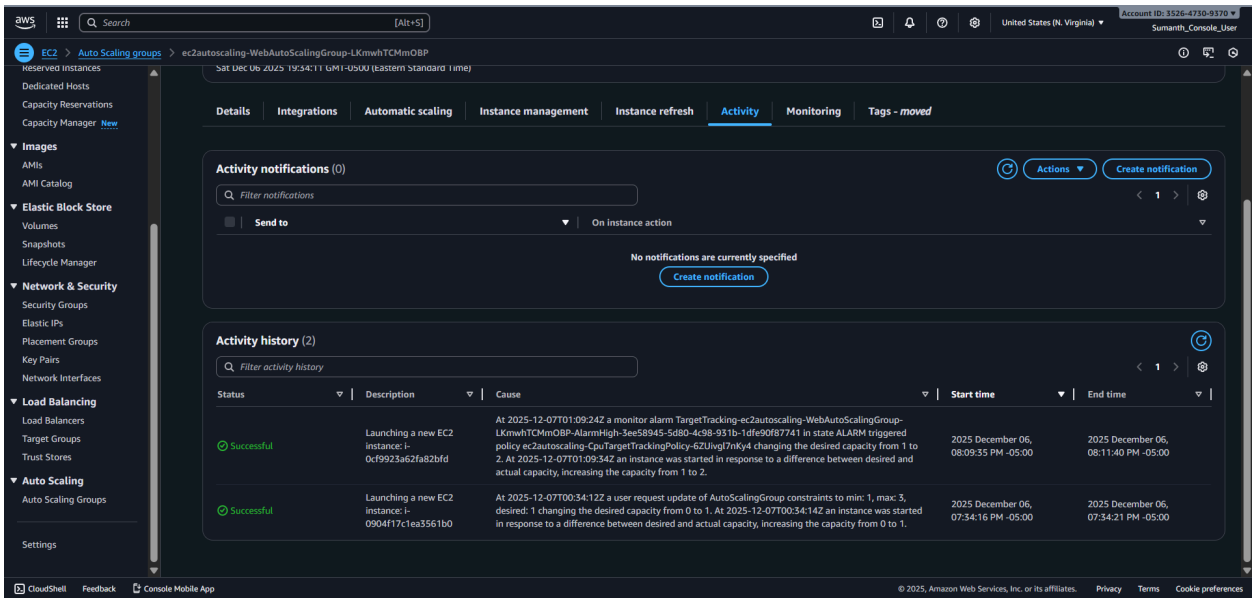
To test the database connection, open `/db-check.php`.

ec2-user@ip-10-0-1-231:~

```
login as: ec2-user
Authenticating with public key "698finalproject"
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-231 ~]$ sudo yum install -y stress
Last metadata expiration check: 0:28:31 ago on Sun Dec  7 00:34:42 2025.
Dependencies resolved.
================================================================================
 Package      Architecture Version                    Repository         Size
================================================================================
Installing:
 stress       x86_64       1.0.7-2.amzn2023.0.1      amazonlinux        34 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm          1.1 MB/s |  34 kB     00:00
--------------------------------------------------------------------------------
Total                                           580 kB/s |  34 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : stress-1.0.7-2.amzn2023.0.1.x86_64                     1/1
  Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64                     1/1
  Verifying        : stress-1.0.7-2.amzn2023.0.1.x86_64                     1/1

Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-10-0-1-231 ~]$ stress --cpu 4 --timeout 120
stress: info: [26995] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

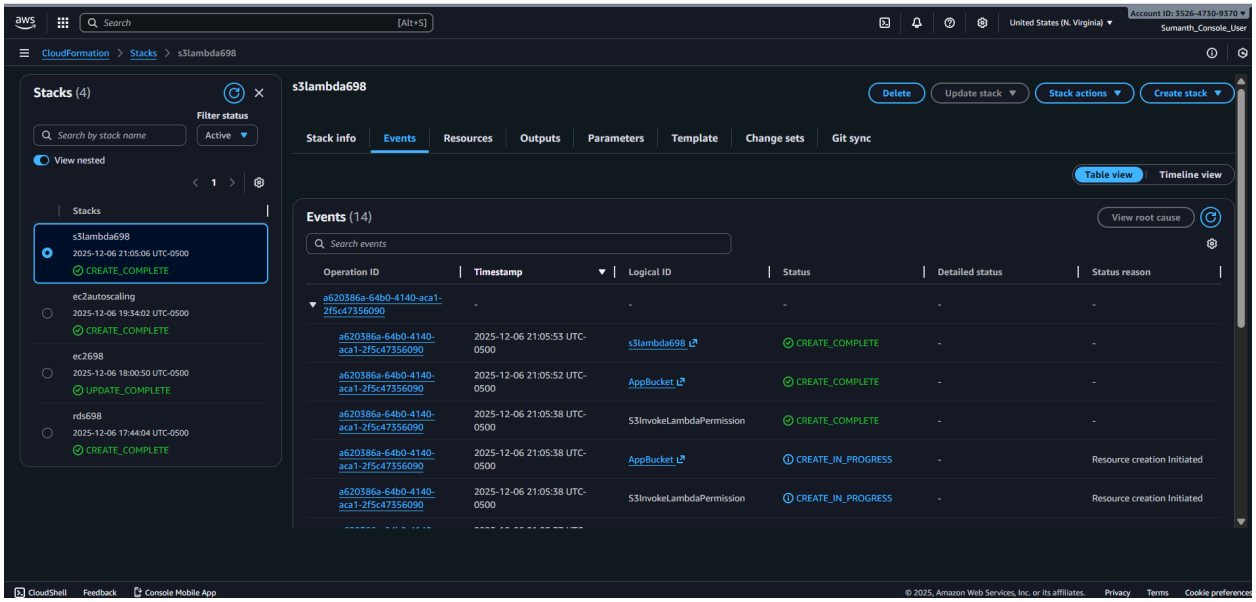


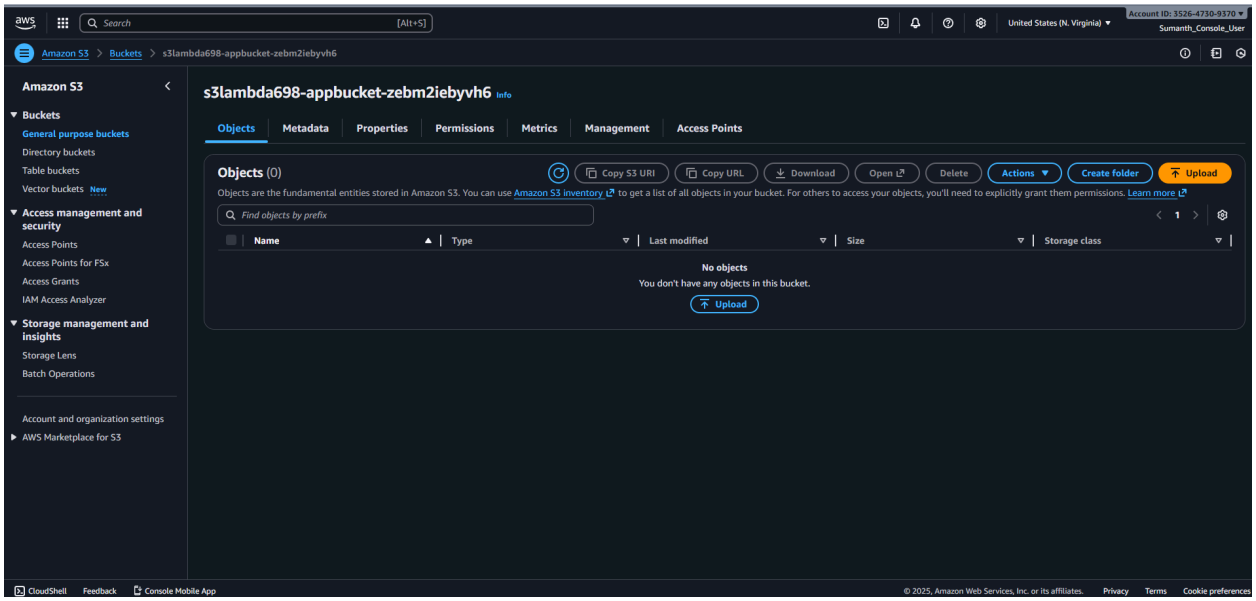


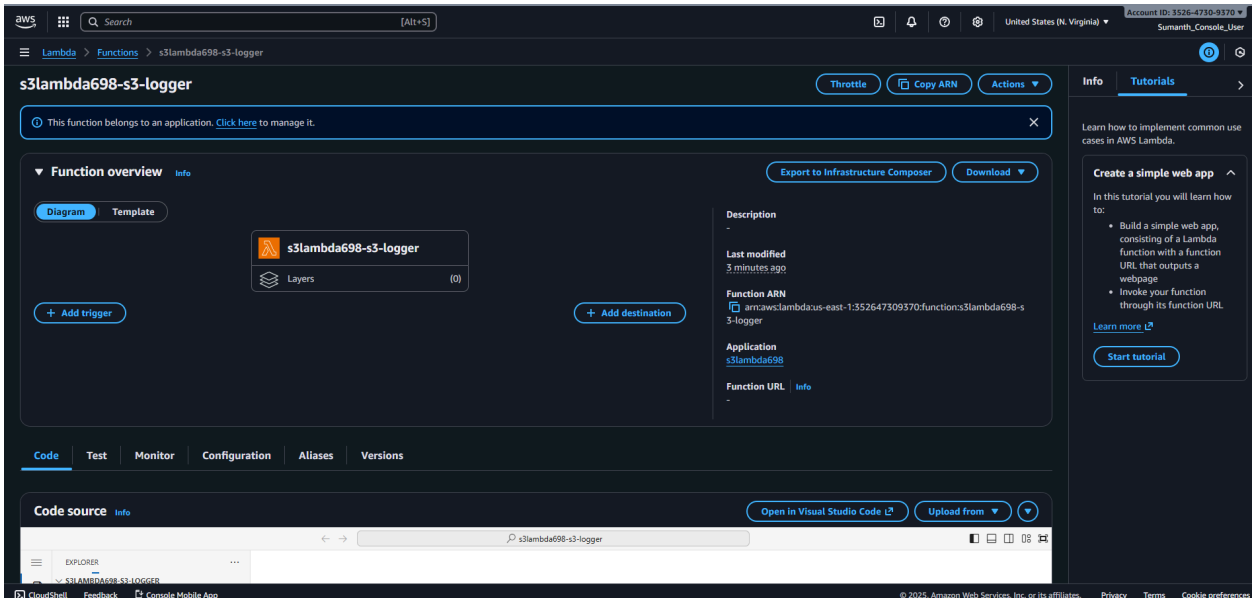


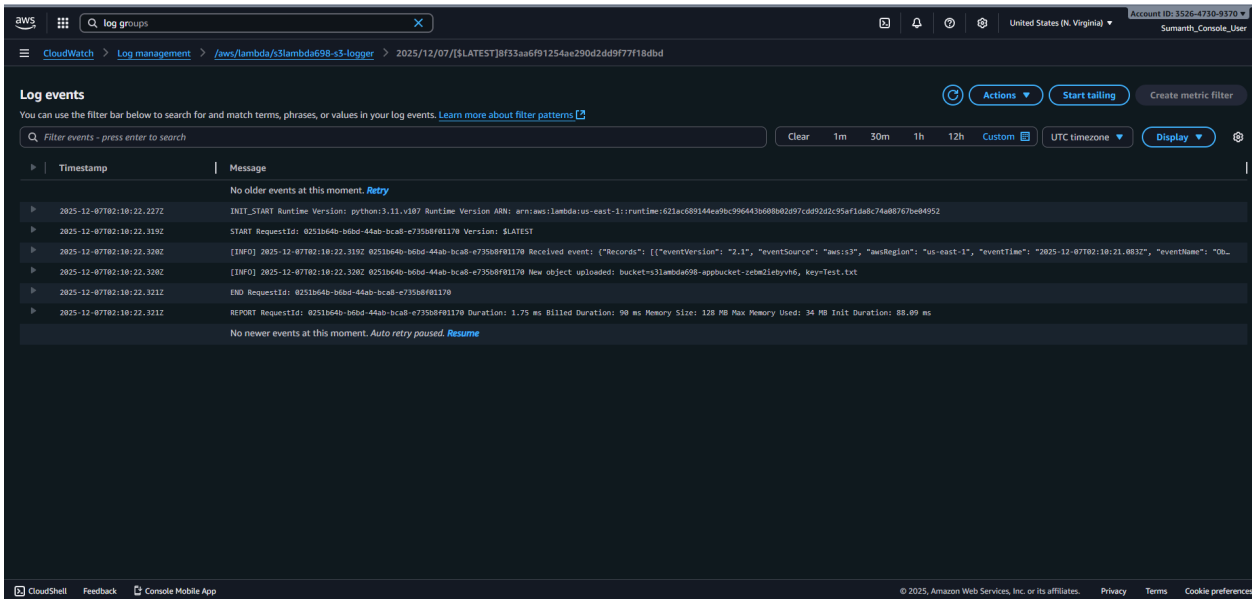
Here, I created a launch template and linked it to an Auto Scaling Group (ASG). The ASG automatically launched new EC2 instances in the public subnets whenever the load increased. I tested autoscaling by SSH-ing into the instance and stressing the CPU. When CloudWatch detected high CPU usage, a new EC2 instance was created.

B. AWS Lambda for Logging S3 Uploads









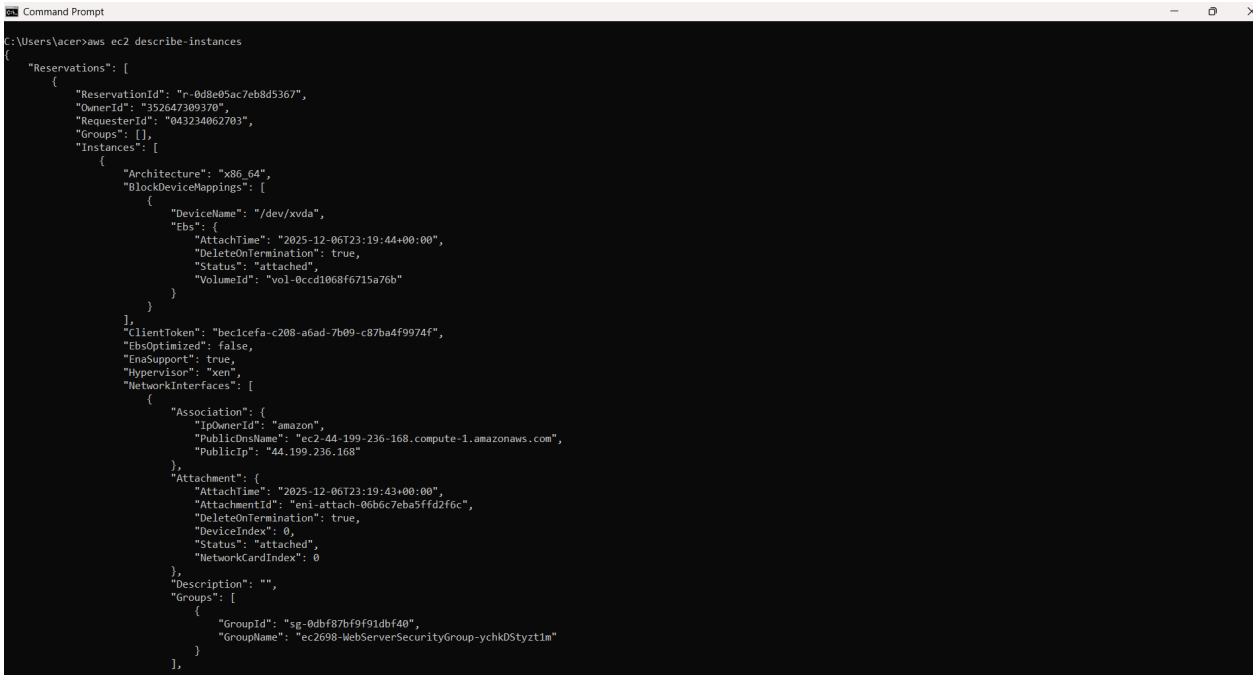
In this step I created a Lambda function in Python that writes logs to CloudWatch whenever someone uploads a file to a specific S3 bucket. After attaching the S3 event trigger, every upload will trigger the Lambda function. When I checked CloudWatch Logs, I could see the file name and metadata.

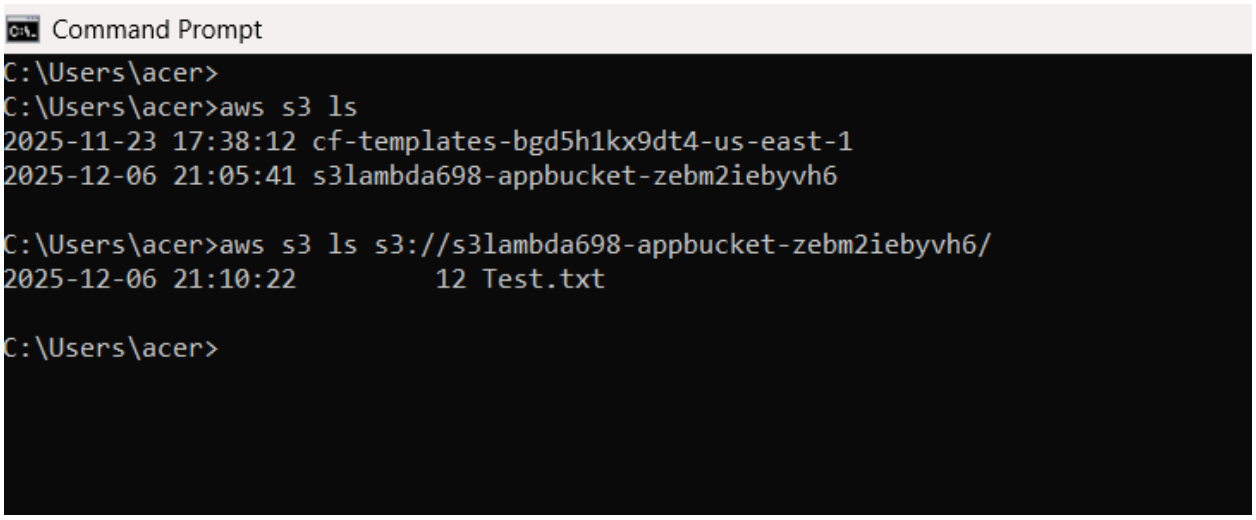
C. AWS Interaction

AWS CLI: Manage EC2, S3, and Lambda resources.

Using AWS CLI commands, I listed the running instances in the EC2, listed the Buckets and files in the S3 bucket. Also I invoked the lambda function.

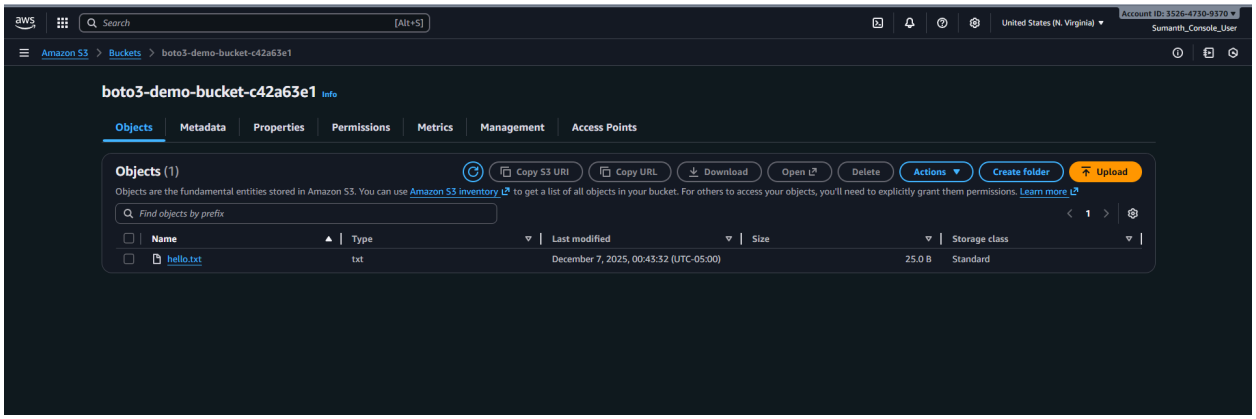
Python Boto3 scripts:

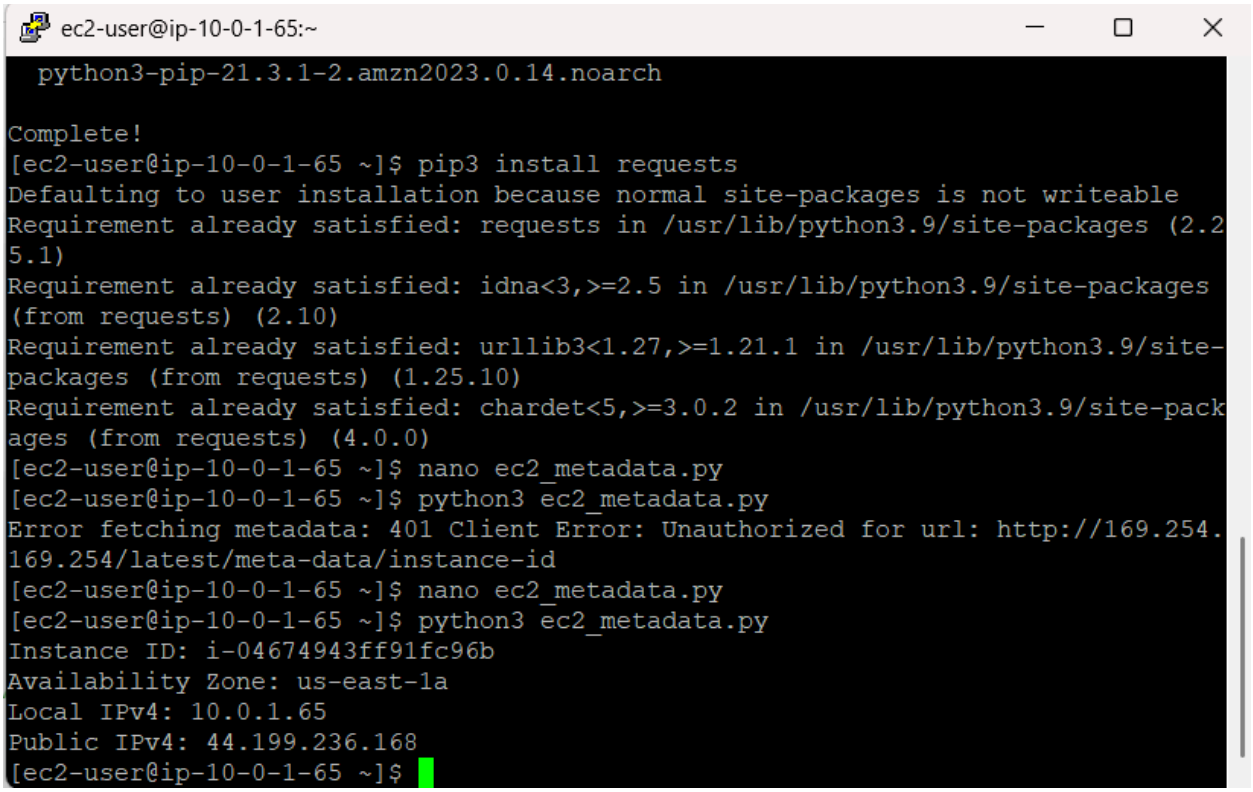


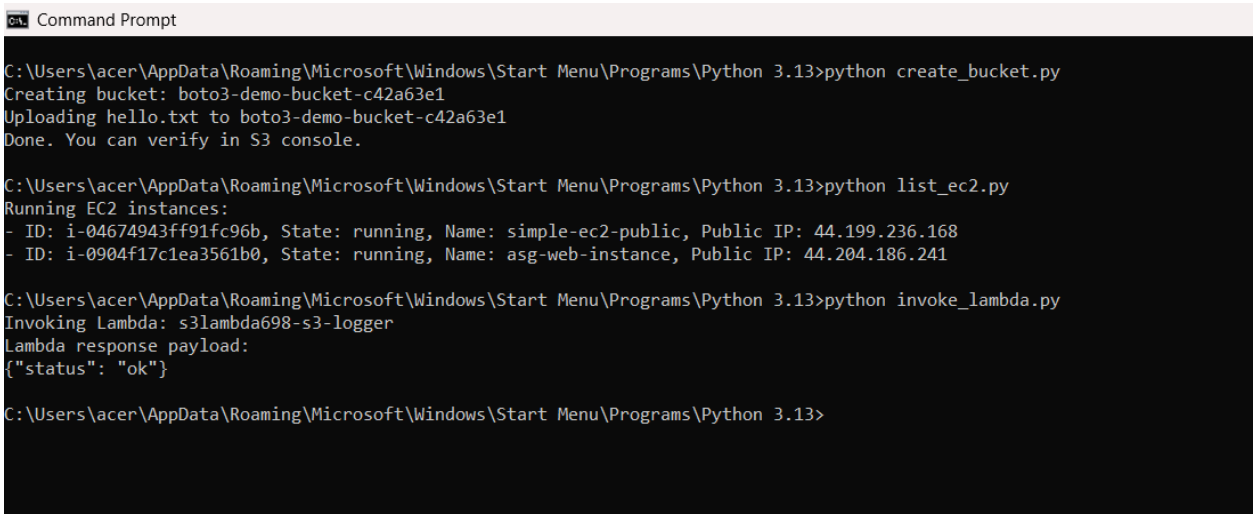


Command Prompt

```
C:\Users\acer>aws s3 ls
2025-11-23 17:38:12 cf-templates-bgd5h1kx9dt4-us-east-1
2025-12-06 21:05:41 s3lambda698-appbucket-zebm2iebyvh6

C:\Users\acer>aws s3 ls s3://s3lambda698-appbucket-zebm2iebyvh6/
2025-12-06 21:10:22         12 Test.txt

C:\Users\acer>aws lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "s3lambda698-s3-logger",
            "FunctionArn": "arn:aws:lambda:us-east-1:352647309370:function:s3lambda698-s3-logger",
            "Runtime": "python3.11",
            "Role": "arn:aws:iam::352647309370:role/s3lambda698-lambda-role",
            "Handler": "index.lambda_handler",
            "CodeSize": 386,
            "Description": "",
            "Timeout": 10,
            "MemorySize": 128,
            "LastModified": "2025-12-07T02:05:30.909+0000",
            "CodeSha256": "k+a+cYMfo9fwm5jYE8hf/bBJO3N6dDrfQBoKQ5bCPhY=",
            "Version": "$LATEST",
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "af49d294-1475-44c3-8939-7772abedb755",
            "PackageType": "Zip",
            "Architectures": [
                "x86_64"
            ],
            "EphemeralStorage": {
                "Size": 512
            },
            "SnapStart": {
                "ApplyOn": "None",
                "OptimizationStatus": "Off"
            },
            "LoggingConfig": {
                "LogFormat": "Text",
                "LogGroup": "/aws/lambda/s3lambda698-s3-logger"
            }
        }
    ]
}

C:\Users\acer>
```





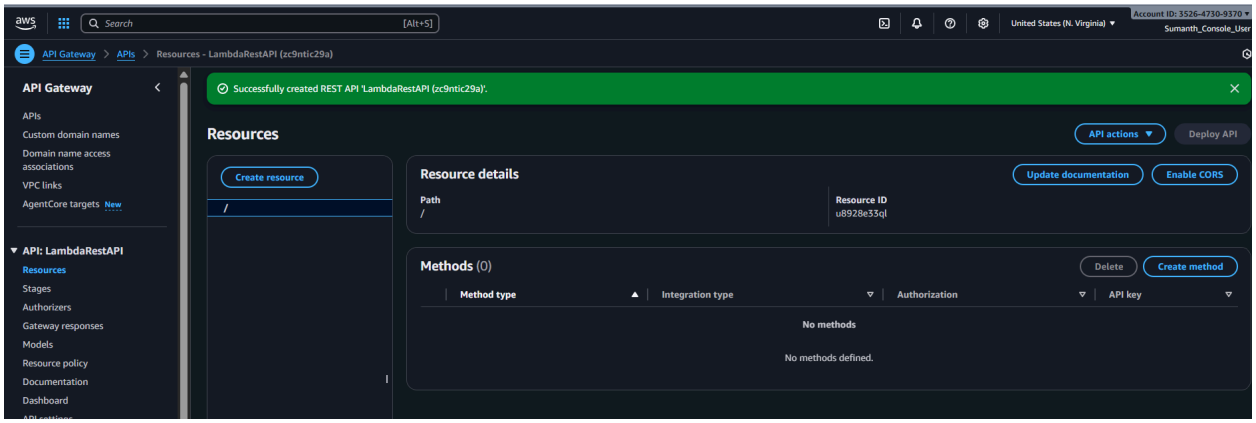


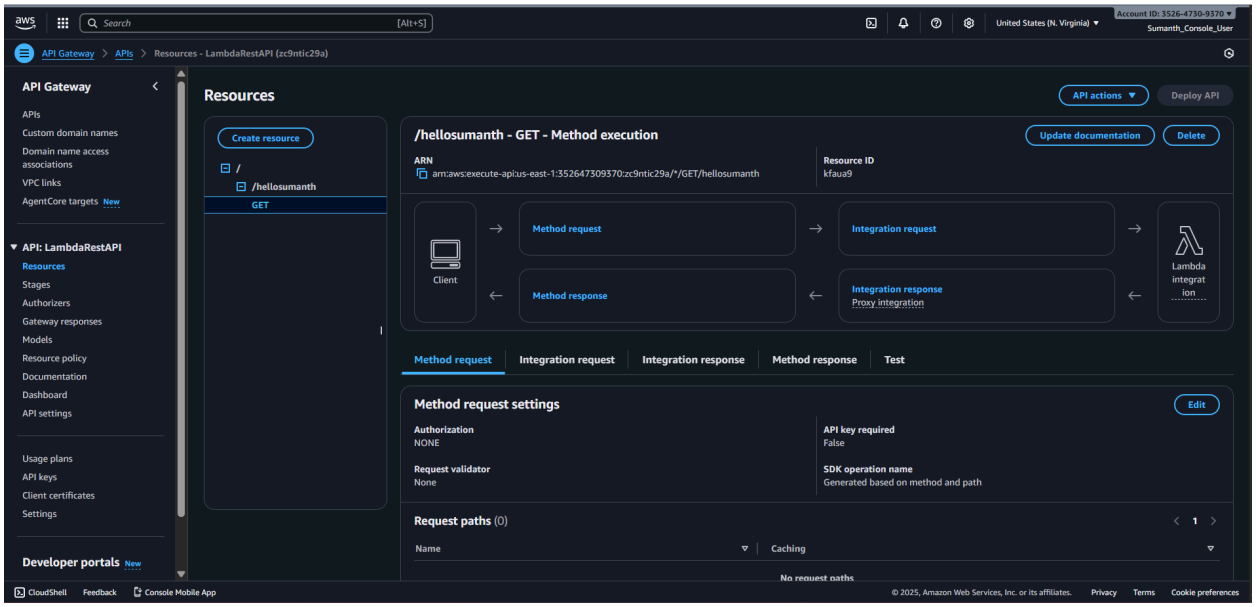
I wrote Python scripts to:

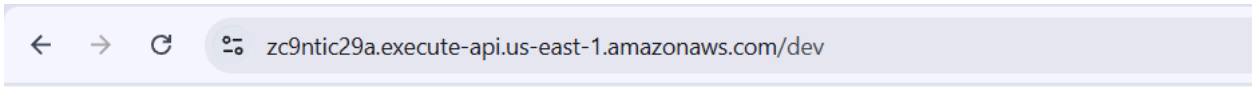
- create an S3 bucket
- upload a file
- retrieve EC2 metadata
- list running EC2 instances
- invoke Lambda manually

Bonus Challenge

Deploy **API Gateway** to invoke Lambda via HTTP requests.







Pretty-print ☑


```
{
  "message": "S3 upload logger Lambda executed successfully.",
  "source": "apigateway",
  "uploaded_files": []
}
```