Project: Deploying a Scalable AWS Architecture with Infrastructure as Code (100 points)

Objective

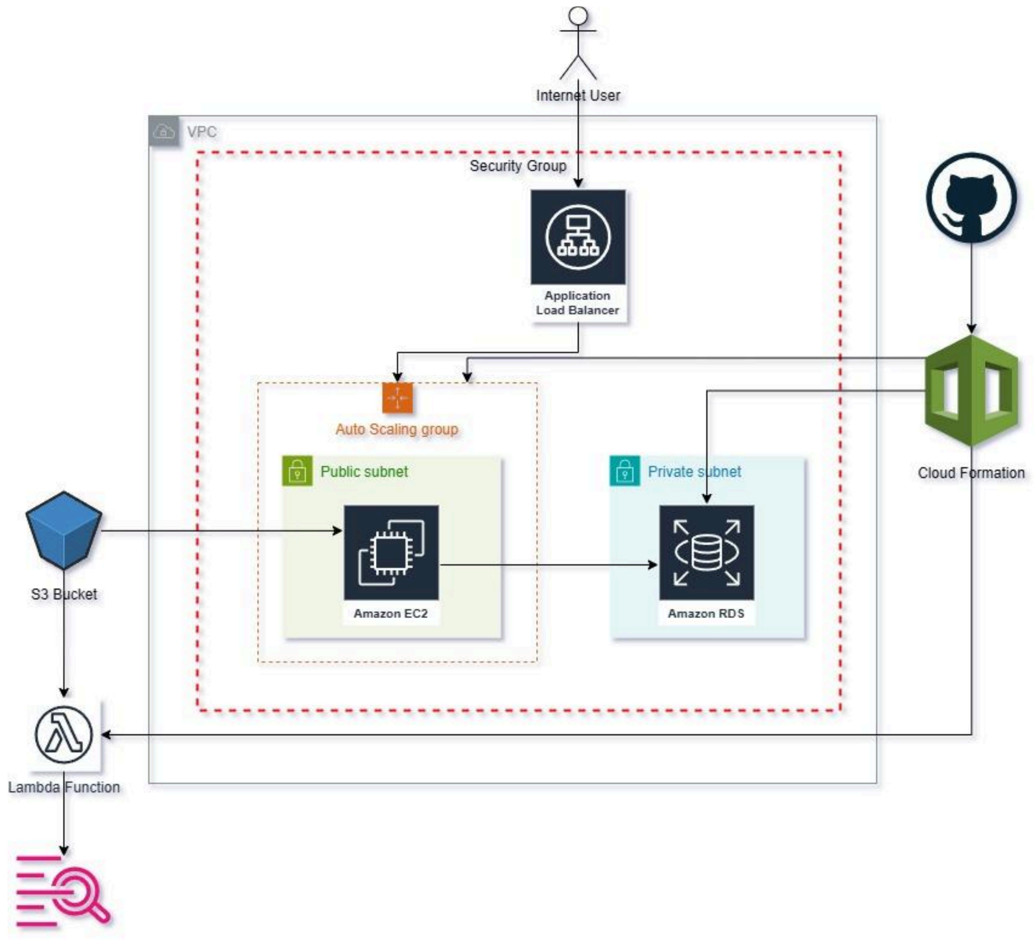
Students will **design, implement, and test** a cloud-based architecture on AWS using EC2, VPC, a database, S3, Lambda, and CloudFormation. They will interact with AWS services through the **AWS console, CLI, and Python Boto3**, use **Terraform** for Infrastructure as Code (IaC), and push their project scripts to **GitHub**.

Project Requirements

1. Design an AWS Architecture

Students must design and **draw an architecture diagram** for their cloud application, which should include:

- **AWS VPC** with at least two subnets (public and private).
- **AWS EC2** instances behind an **Application Load Balancer (ALB)**.
- **Relational database** (RDS or MySQL/PostgreSQL on EC2).
- **S3 bucket** for storing files (e.g., logs, backups, static content).
- **AWS Lambda** to log S3 file uploads to **CloudWatch Logs**.
- **Autoscaling** for the web server layer.
- **AWS CloudFormation** to deploy infrastructure.
- **Security Group** configurations to control network access.
- **GitHub integration** for version control.



The provided architecture diagram depicts a secure and scalable cloud environment on AWS with Infrastructure as Code via AWS CloudFormation, version controlled via GitHub. At the center of this architecture is a Virtual Private Cloud (VPC), which provides a logically isolated network environment. In the VPC, two subnets are configured: a public subnet hosting internet-facing components like EC2 instances and an Application Load Balancer (ALB) and a private subnet hosting the Amazon RDS database securely.

The Application Load Balancer is where incoming traffic from internet users comes in. It is protected by a security group that filters traffic and allows only necessary ports such as HTTP (port 80) or HTTPS (port 443). The ALB directs incoming requests to EC2 instances running in an Auto Scaling Group within the public subnet. This auto-scaling setup offers high availability and handles varying levels of traffic by dynamically scaling the EC2 instances.

These EC2 instances run a web application (e.g., an application showing the weather) and are configured to talk securely to the RDS database in the private subnet. This separation keeps sensitive data in the database from being accessed by the public internet. Beyond this setup, the static files such as backups, logs, or media content are placed in an S3 bucket. A Lambda function is configured to trigger automatically on any file upload to the S3 bucket, logging the event details to CloudWatch Logs for auditing and monitoring.

Overall, the infrastructure is version controlled and automated. CloudFormation templates do provisioning and deployment of AWS resources, while GitHub contains project source code, templates, and documentation. The architecture is secure, operationally efficient, and scalable and aligns with cloud-native best practices.

---

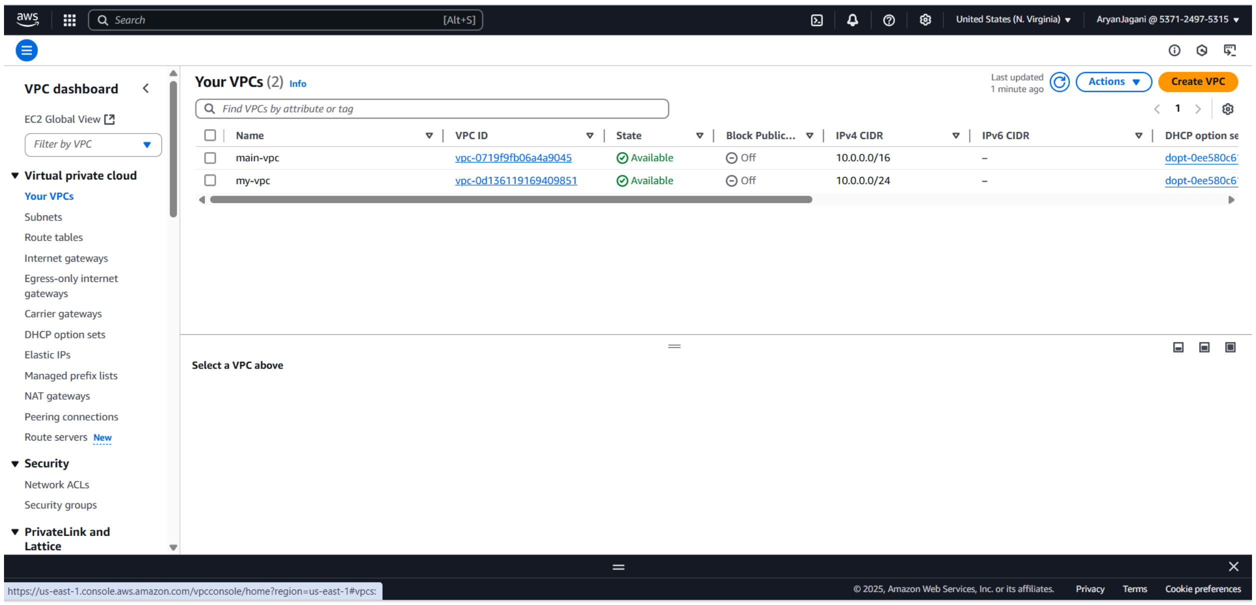
2. Implementation

A. Infrastructure Deployment

main.tf

- **Use Terraform** to create networking components (VPC, subnets, security groups).

Vpc-

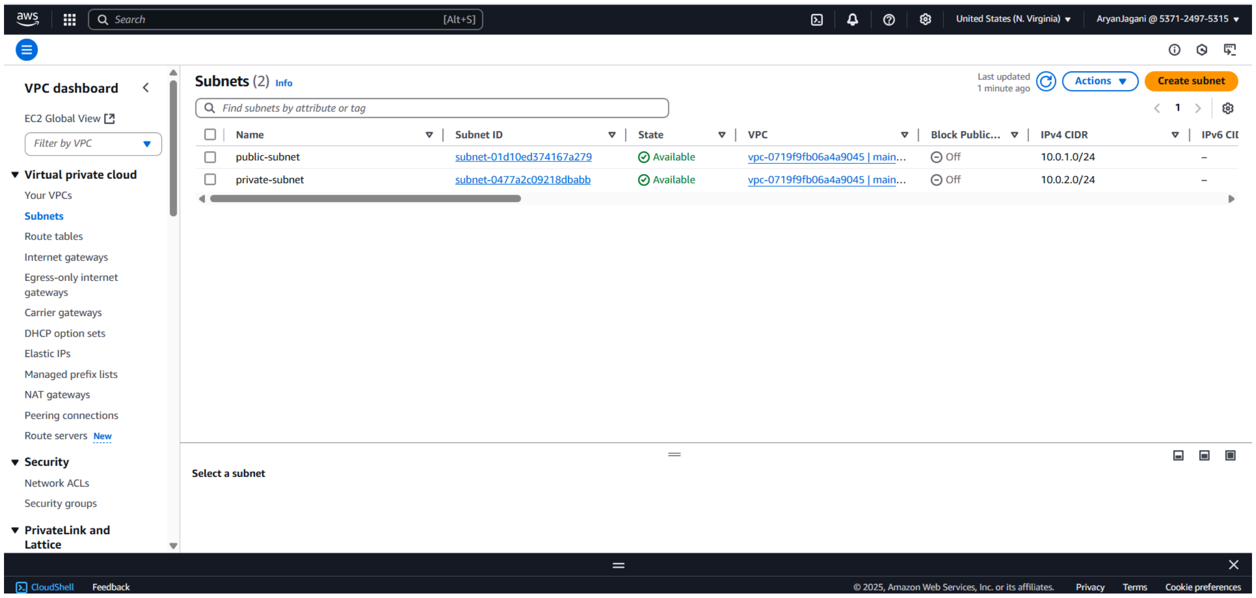


Your VPCs (2) Info

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option se
main-vpc	vpc-0719f9fb06a4a9045	Available	Off	10.0.0.0/16	–	dopt-0ee580c6
my-vpc	vpc-0d136119169409851	Available	Off	10.0.0.0/24	–	dopt-0ee580c6

Select a VPC above

Subnet-



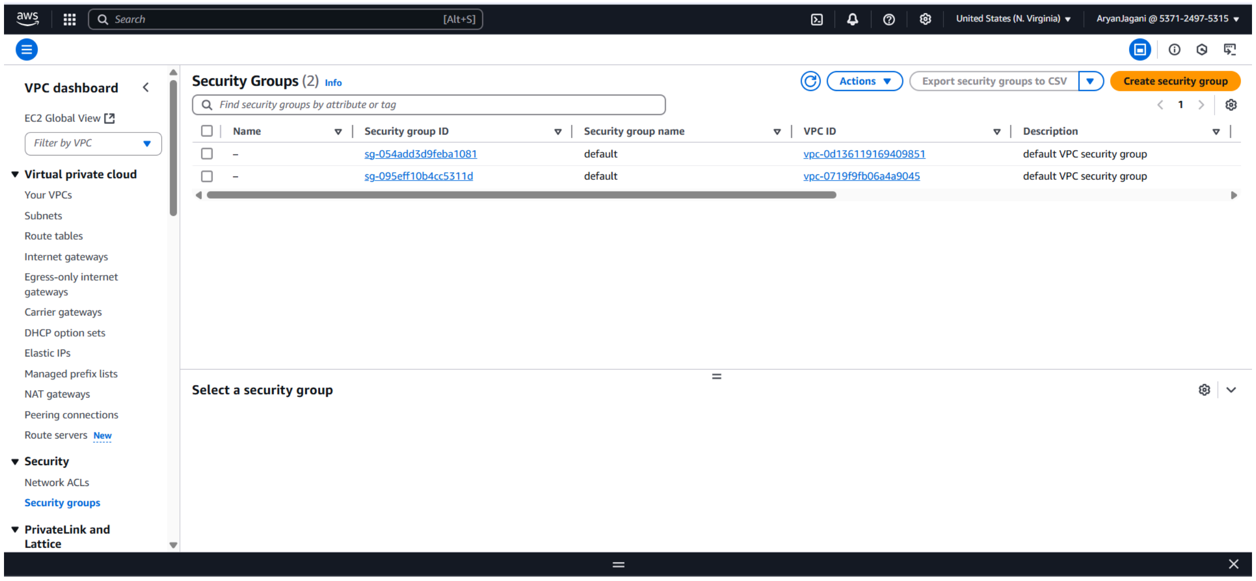
Subnets (2) Info

Last updated 1 minute ago

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIE
public-subnet	subnet-01d10ed374167a279	Available	vpc-0719f9fb06a4a9045 | main...	Off	10.0.1.0/24	–
private-subnet	subnet-0477a2c09218dbabb	Available	vpc-0719f9fb06a4a9045 | main...	Off	10.0.2.0/24	–

Select a subnet

Security group-

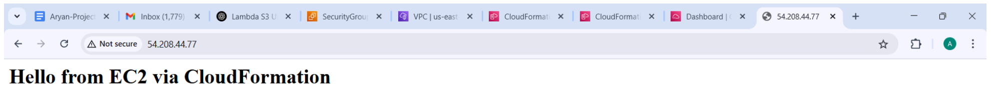


Security Groups (2) Info

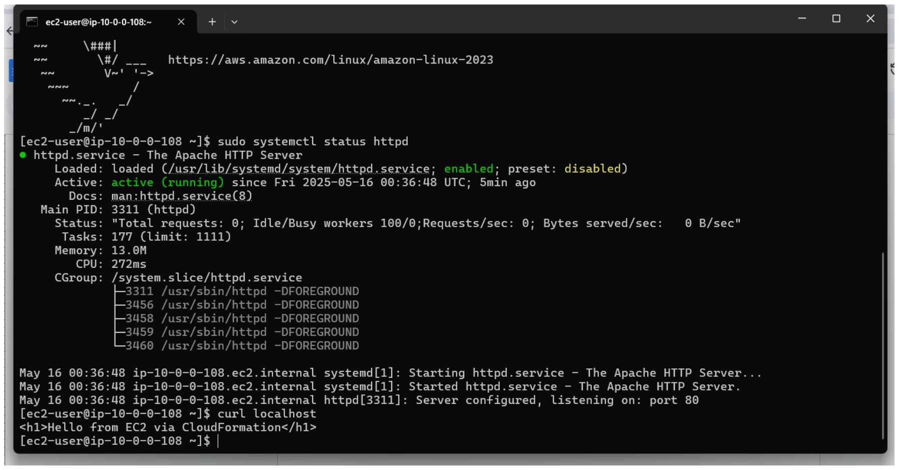
Name	Security group ID	Security group name	VPC ID	Description
–	sg-054add3d9feba1081	default	vpc-0d136119169409851	default VPC security group
–	sg-095eff10b4cc5311d	default	vpc-0719f9fb06a4a9045	default VPC security group

Select a security group

- **Use CloudFormation** to deploy EC2 instances, RDS, and Lambda functions.
    1) EC2-

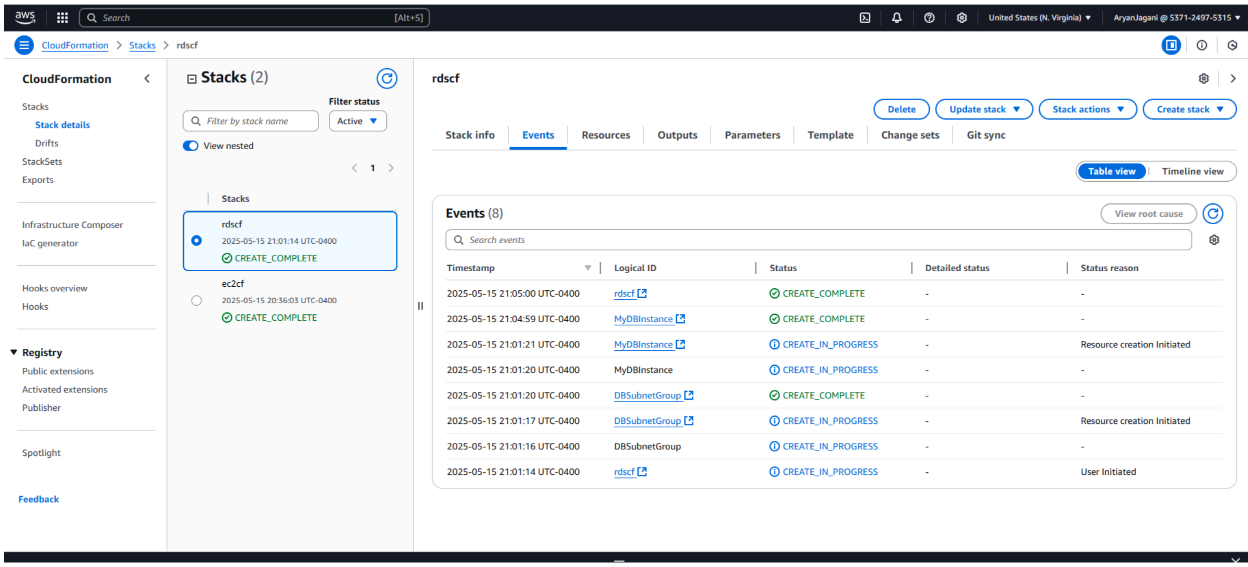


Hello from EC2 via CloudFormation



```
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-0-108 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-05-16 00:36:48 UTC; 5min ago
       Docs: man:httpd.service(8)
   Main PID: 3311 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
      Tasks: 177 (limit: 1111)
     Memory: 13.0M
        CPU: 272ms
     CGroup: /system.slice/httpd.service
             ├─3311 /usr/sbin/httpd -DFOREGROUND
             ├─3456 /usr/sbin/httpd -DFOREGROUND
             ├─3458 /usr/sbin/httpd -DFOREGROUND
             ├─3459 /usr/sbin/httpd -DFOREGROUND
             └─3460 /usr/sbin/httpd -DFOREGROUND

May 16 00:36:48 ip-10-0-0-108.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
May 16 00:36:48 ip-10-0-0-108.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
May 16 00:36:48 ip-10-0-0-108.ec2.internal httpd[3311]: Server configured, listening on: port 80
[ec2-user@ip-10-0-0-108 ~]$ curl localhost
<h1>Hello from EC2 via CloudFormation</h1>
[ec2-user@ip-10-0-0-108 ~]$ |
```

2) RDS



CloudFormation > Stacks > rdscf

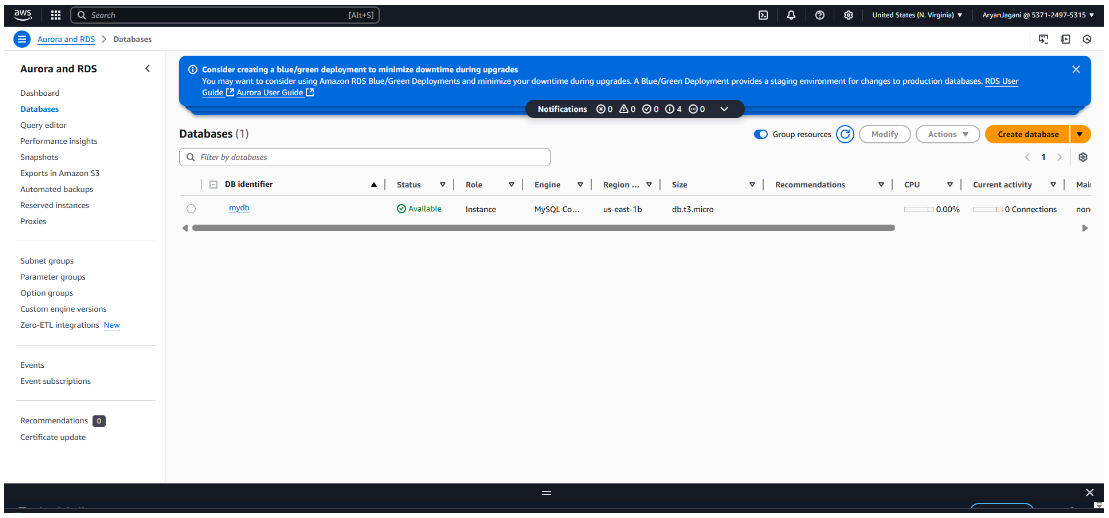
Stacks (2)

Stack	Created	Status
rdscf	2025-05-15 21:01:14 UTC-0400	CREATE_COMPLETE
ec2cf	2025-05-15 20:36:03 UTC-0400	CREATE_COMPLETE

rdscf

Events (8)

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-05-15 21:05:00 UTC-0400	rdscf	CREATE_COMPLETE	-	-
2025-05-15 21:04:59 UTC-0400	MyDBInstance	CREATE_COMPLETE	-	-
2025-05-15 21:01:21 UTC-0400	MyDBInstance	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-05-15 21:01:20 UTC-0400	MyDBInstance	CREATE_IN_PROGRESS	-	-
2025-05-15 21:01:20 UTC-0400	DBSubnetGroup	CREATE_COMPLETE	-	-
2025-05-15 21:01:17 UTC-0400	DBSubnetGroup	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-05-15 21:01:16 UTC-0400	DBSubnetGroup	CREATE_IN_PROGRESS	-	-
2025-05-15 21:01:14 UTC-0400	rdscf	CREATE_IN_PROGRESS	-	User Initiated



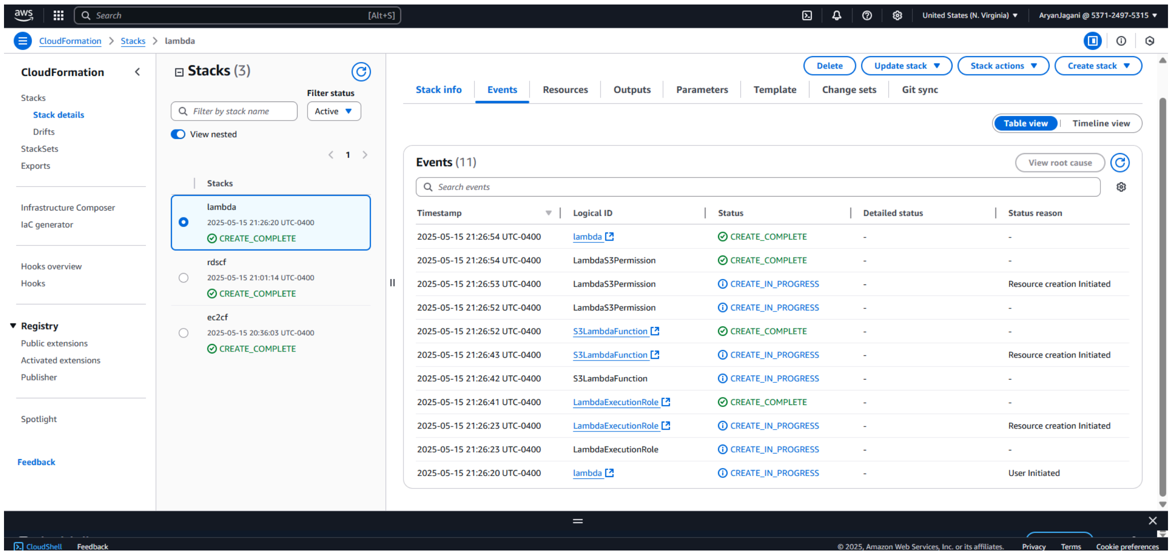
Aurora and RDS > Databases

Consider creating a blue/green deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide

Databases (1)

DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current activity	Mai
mydb	Available	Instance	MySQL Co...	us-east-1b	db.t3.micro		0.00%	0 Connections	non

3) lambda



CloudFormation > Stacks > lambda

Stacks (3)

Stacks		
lambda	2025-05-15 21:26:20 UTC-0400	CREATE_COMPLETE
rdscf	2025-05-15 21:01:14 UTC-0400	CREATE_COMPLETE
ec2cf	2025-05-15 20:36:03 UTC-0400	CREATE_COMPLETE

Events (11)

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-05-15 21:26:54 UTC-0400	lambda	CREATE_COMPLETE	-	-
2025-05-15 21:26:54 UTC-0400	LambdaS3Permission	CREATE_COMPLETE	-	-
2025-05-15 21:26:53 UTC-0400	LambdaS3Permission	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-05-15 21:26:52 UTC-0400	LambdaS3Permission	CREATE_IN_PROGRESS	-	-
2025-05-15 21:26:52 UTC-0400	S3LambdaFunction	CREATE_COMPLETE	-	-
2025-05-15 21:26:43 UTC-0400	S3LambdaFunction	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-05-15 21:26:42 UTC-0400	S3LambdaFunction	CREATE_IN_PROGRESS	-	-
2025-05-15 21:26:41 UTC-0400	LambdaExecutionRole	CREATE_COMPLETE	-	-
2025-05-15 21:26:23 UTC-0400	LambdaExecutionRole	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-05-15 21:26:23 UTC-0400	LambdaExecutionRole	CREATE_IN_PROGRESS	-	-
2025-05-15 21:26:20 UTC-0400	lambda	CREATE_IN_PROGRESS	-	User Initiated

- **Deploy a web application** on EC2 (can be a simple HTML page).

1. EC2 Instance Setup

- **Instance Type:** t2.micro (Free Tier)
- **AMI:** Amazon Linux 2023
- **Public IP:** `44.200.34.241`
- **Key Pair:** `my_key.pem`
- **Subnet:** Public (`project-subnet-public1-us-east-1a`)
- **Security Group:**
  - SSH (port 22) — 0.0.0.0/0
  - HTTP (port 80) — 0.0.0.0/0

2. Apache Web Server Installation

```
sudo yum update -y

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd
```

3. SCP Upload Command (from Git Bash)

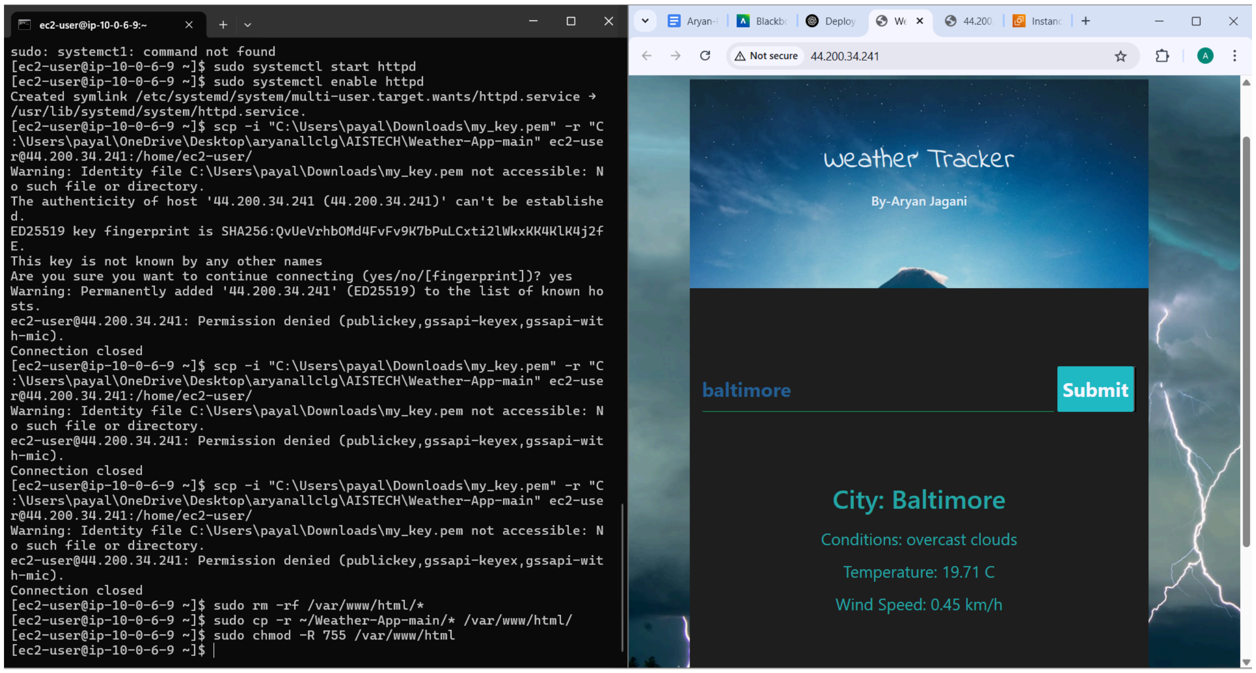
scp -i "/c/Users/payal/Downloads/my_key.pem" -r "/c/Users/payal/OneDrive/Desktop/aryanallclg/AISTECH/Weather-App-main" ec2-user@44.200.34.241:/home/ec2-user/

4. Deploying to Apache Root

sudo rm -rf /var/www/html/*

sudo cp -r ~/Weather-App-main/* /var/www/html/

sudo chmod -R 755 /var/www/html



```
sudo: systemctl: command not found
[ec2-user@ip-10-0-6-9 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-6-9 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service →
/usr/lib/systemd/system/httpd.service.
[ec2-user@ip-10-0-6-9 ~]$ scp -i "C:\Users\payal\Downloads\my_key.pem" -r "C
:\Users\payal\OneDrive\Desktop\aryanallclg\AISTECH\Weather-App-main" ec2-use
r@44.200.34.241:/home/ec2-user/
Warning: Identity file C:\Users\payal\Downloads\my_key.pem not accessible: N
o such file or directory.
The authenticity of host '44.200.34.241 (44.200.34.241)' can't be establishe
d.
ED25519 key fingerprint is SHA256:QvUeVrhbOMd4FvFv9K7bPuLCxti2lWkxKK4KlK4j2f
E.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.200.34.241' (ED25519) to the list of known ho
sts.
ec2-user@44.200.34.241: Permission denied (publickey,gssapi-keyex,gssapi-wit
h-mic).
Connection closed
[ec2-user@ip-10-0-6-9 ~]$ scp -i "C:\Users\payal\Downloads\my_key.pem" -r "C
:\Users\payal\OneDrive\Desktop\aryanallclg\AISTECH\Weather-App-main" ec2-use
r@44.200.34.241:/home/ec2-user/
Warning: Identity file C:\Users\payal\Downloads\my_key.pem not accessible: N
o such file or directory.
ec2-user@44.200.34.241: Permission denied (publickey,gssapi-keyex,gssapi-wit
h-mic).
Connection closed
[ec2-user@ip-10-0-6-9 ~]$ scp -i "C:\Users\payal\Downloads\my_key.pem" -r "C
:\Users\payal\OneDrive\Desktop\aryanallclg\AISTECH\Weather-App-main" ec2-use
r@44.200.34.241:/home/ec2-user/
Warning: Identity file C:\Users\payal\Downloads\my_key.pem not accessible: N
o such file or directory.
ec2-user@44.200.34.241: Permission denied (publickey,gssapi-keyex,gssapi-wit
h-mic).
Connection closed
[ec2-user@ip-10-0-6-9 ~]$ sudo rm -rf /var/www/html/*
[ec2-user@ip-10-0-6-9 ~]$ sudo cp -r ~/Weather-App-main/* /var/www/html/
[ec2-user@ip-10-0-6-9 ~]$ sudo chmod -R 755 /var/www/html
[ec2-user@ip-10-0-6-9 ~]$
```

- **Configure the database** for the web application.

1. MySQL Client Installation (on EC2)

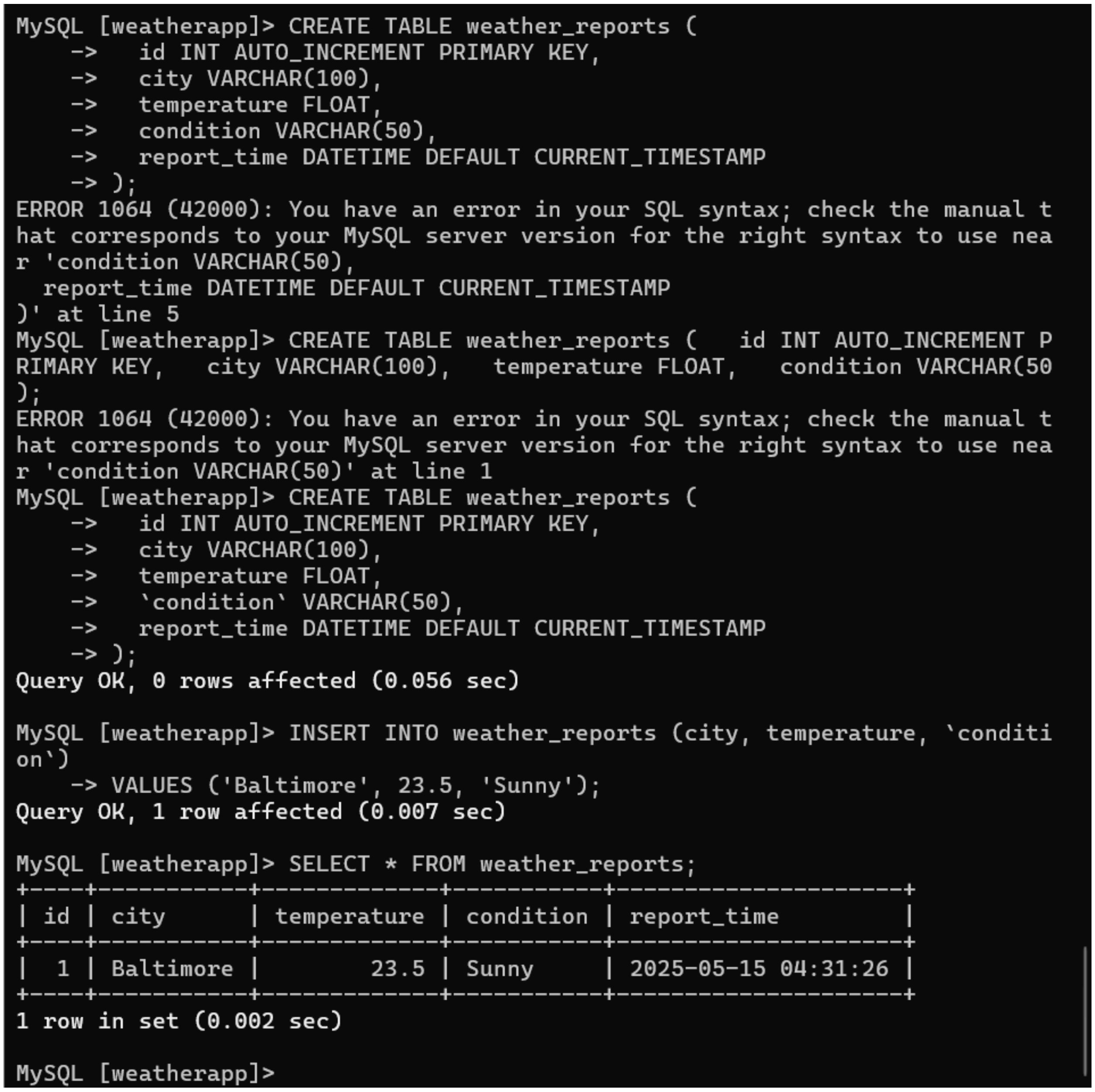
sudo dnf install -y mariadb105

2. RDS Connection from EC2

```
mysql -h weatherapp-db.co3soywkaicy.us-east-1.rds.amazonaws.com -u admin -p
```

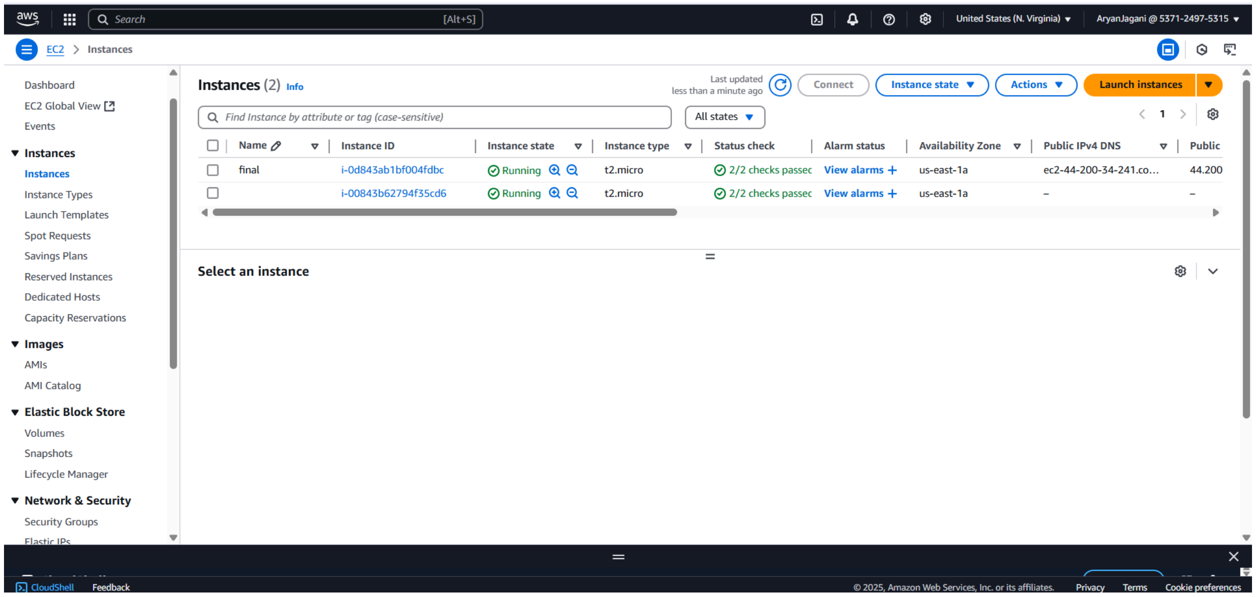
3. Database and Table Creation

```
CREATE DATABASE weatherapp;
USE weatherapp;

CREATE TABLE weather_reports (
  id INT AUTO_INCREMENT PRIMARY KEY,
  city VARCHAR(100),
  temperature FLOAT,
  `condition` VARCHAR(50),
  report_time DATETIME DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO weather_reports (city, temperature, `condition`)
VALUES ('Baltimore', 23.5, 'Sunny');

SELECT * FROM weather_reports;
```



```
MySQL [weatherapp]> CREATE TABLE weather_reports (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   city VARCHAR(100),
    ->   temperature FLOAT,
    ->   condition VARCHAR(50),
    ->   report_time DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual t
hat corresponds to your MySQL server version for the right syntax to use nea
r 'condition VARCHAR(50),
  report_time DATETIME DEFAULT CURRENT_TIMESTAMP
)' at line 5
MySQL [weatherapp]> CREATE TABLE weather_reports (   id INT AUTO_INCREMENT P
RIMARY KEY,   city VARCHAR(100),   temperature FLOAT,   condition VARCHAR(50
);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual t
hat corresponds to your MySQL server version for the right syntax to use nea
r 'condition VARCHAR(50)' at line 1
MySQL [weatherapp]> CREATE TABLE weather_reports (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   city VARCHAR(100),
    ->   temperature FLOAT,
    ->   `condition` VARCHAR(50),
    ->   report_time DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.056 sec)

MySQL [weatherapp]> INSERT INTO weather_reports (city, temperature, `conditi
on`)
    -> VALUES ('Baltimore', 23.5, 'Sunny');
Query OK, 1 row affected (0.007 sec)

MySQL [weatherapp]> SELECT * FROM weather_reports;
+----+-----------+-------------+-----------+---------------------+
| id | city      | temperature | condition | report_time         |
+----+-----------+-------------+-----------+---------------------+
|  1 | Baltimore |        23.5 | Sunny     | 2025-05-15 04:31:26 |
+----+-----------+-------------+-----------+---------------------+
1 row in set (0.002 sec)

MySQL [weatherapp]>
```

- **Implement autoscaling** to manage web server load.

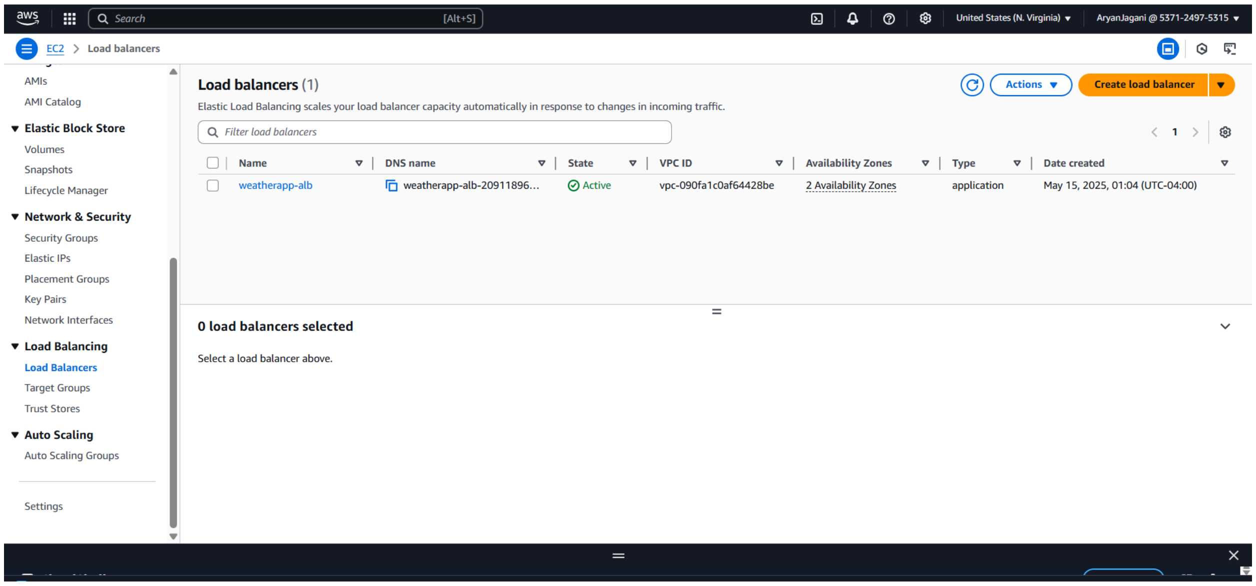


Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
☐	final	i-0d843ab1bf004fdbc	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-200-34-241.co...	44.200
☐		i-00843b62794f35cd6	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	–	–

Select an instance



Load balancers (1)

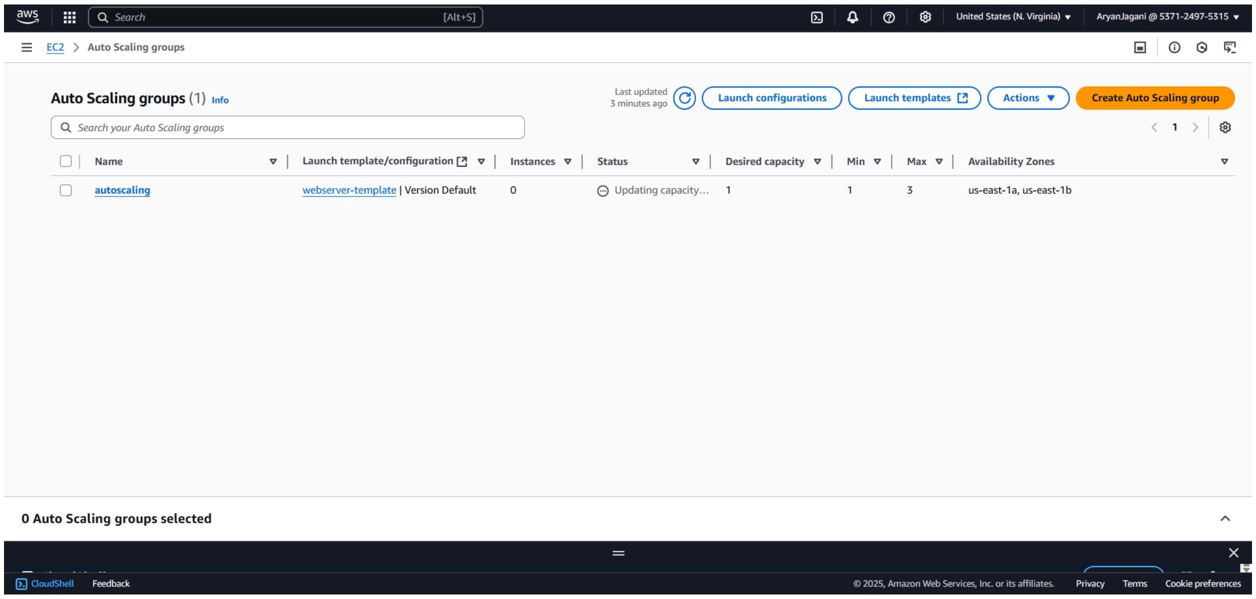
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
☐	weatherapp-alb	weatherapp-alb-20911896...	Active	vpc-090fa1c0af64428be	2 Availability Zones	application	May 15, 2025, 01:04 (UTC-04:00)

0 load balancers selected

Select a load balancer above.



EC2 > Auto Scaling groups

Auto Scaling groups (1) Info

Last updated 3 minutes ago

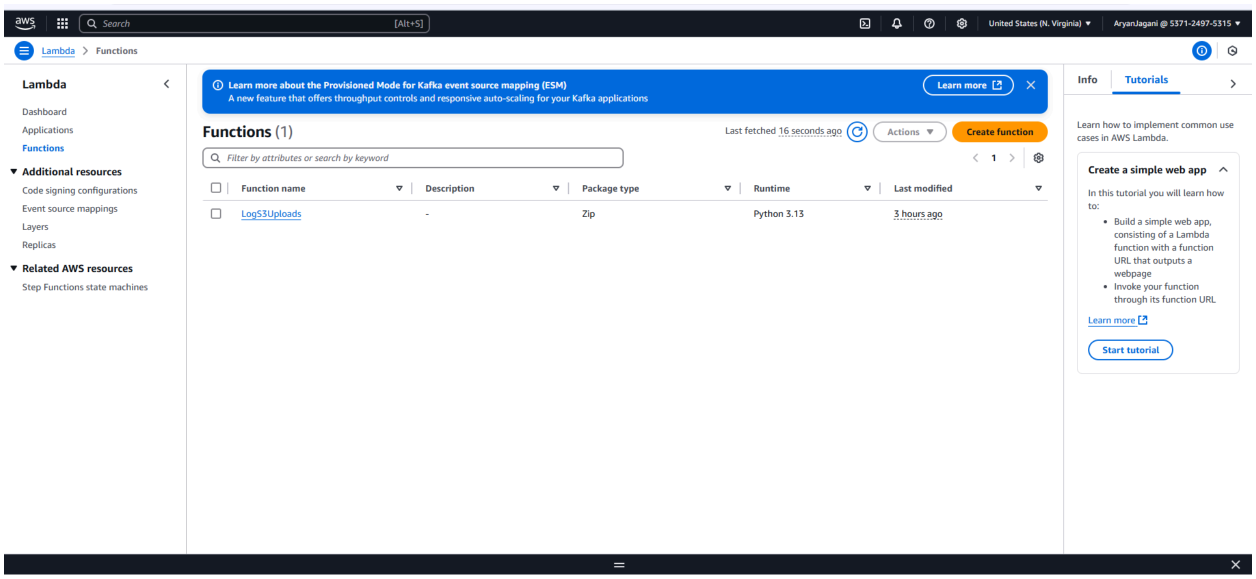
Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
autoscaling	webserver-template | Version Default	0	Updating capacity...	1	1	3	us-east-1a, us-east-1b

0 Auto Scaling groups selected

© 2025, Amazon Web Services, Inc. or its affiliates.

B. AWS Lambda for Logging S3 Uploads

- **Create an AWS Lambda function** that logs S3 file uploads to CloudWatch Logs.
- **Use the following Python script for the Lambda function:**



Lambda > Functions

Learn more about the Provisioned Mode for Kafka event source mapping (ESM)
A new feature that offers throughput controls and responsive auto-scaling for your Kafka applications

Functions (1)

Last fetched 16 seconds ago

Function name	Description	Package type	Runtime	Last modified
LogS3Uploads	-	Zip	Python 3.13	3 hours ago

Learn how to implement common use cases in AWS Lambda.

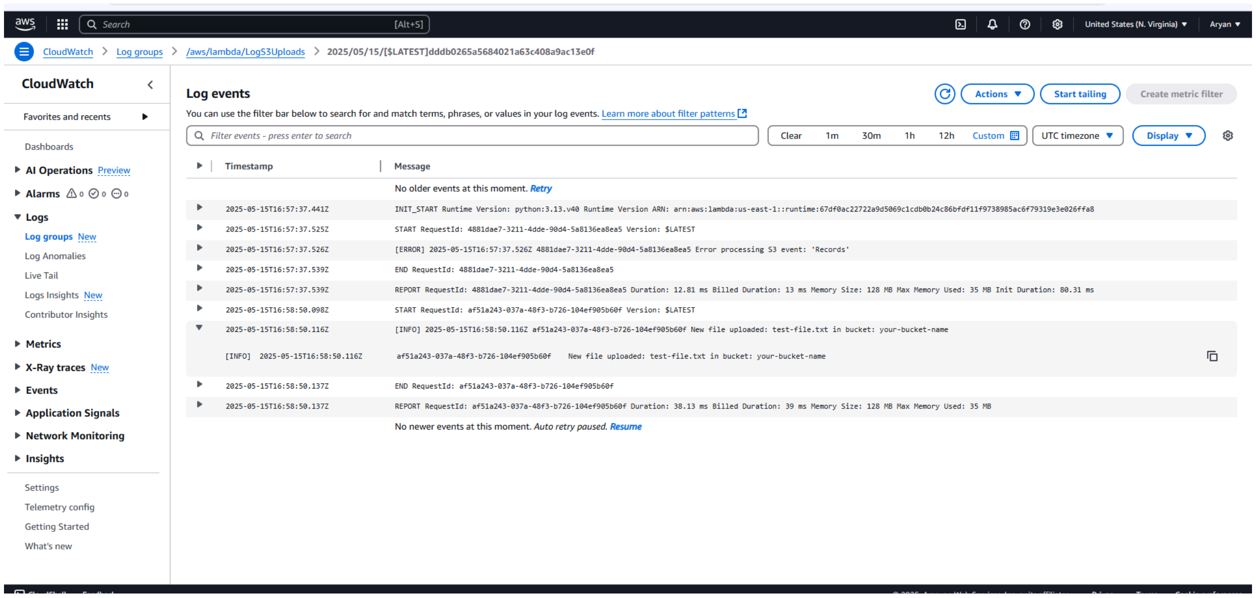
Create a simple web app

In this tutorial you will learn how to:
- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

```
import json
import boto3
import logging
```

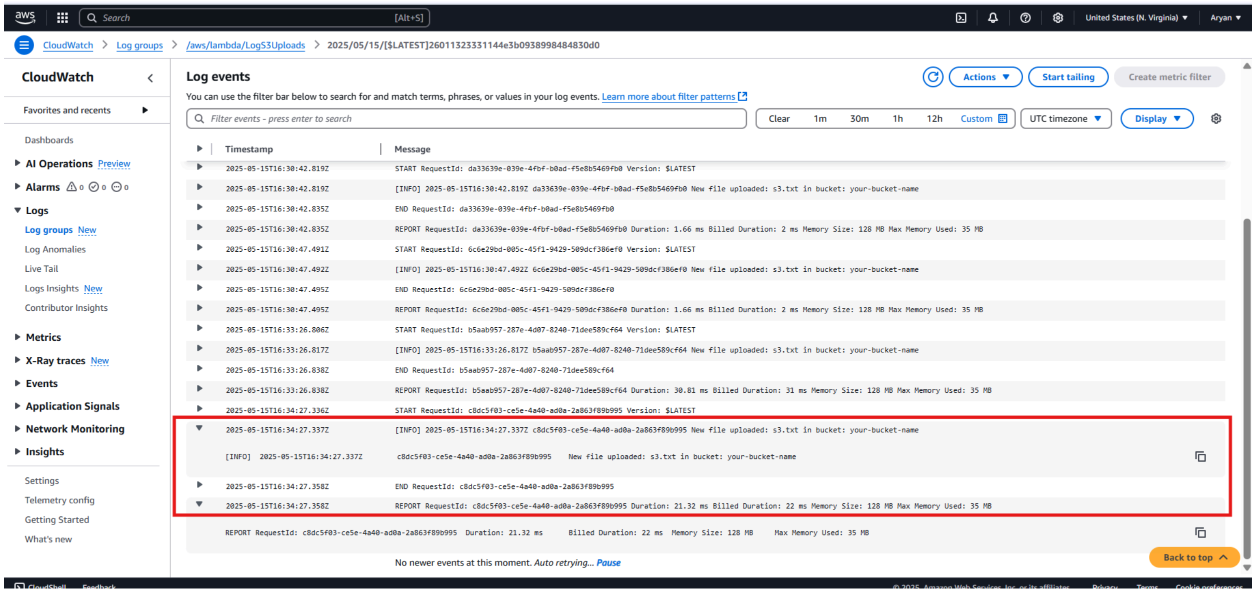
```
# Set up logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        # Extract bucket name and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']

        # Log the upload event
        log_message = f"New file uploaded: {object_key} in bucket:
{bucket_name}"
        logger.info(log_message)

        return {
            "statusCode": 200,
            "body": json.dumps(log_message)
        }

    except Exception as e:
        logger.error(f"Error processing S3 event: {str(e)}")
        return {
            "statusCode": 500,
            "body": json.dumps(f"Error: {str(e)}")
        }
```

- **Set up an S3 event trigger** so the function executes whenever a new file is uploaded.


CloudWatch > Log groups > /aws/lambda/LogS3Uploads > 2025/05/15/[$LATEST]dddb0265a5684021a63c408a9ac13e0f
Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                     Message
                              No older events at this moment. Retry
2025-05-15T16:57:37.441Z      INIT_START Runtime Version: python:3.13.v40 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:67df0ac22722a9d5069c1cdb0b24c86bfdf11f9738985ac6f79319e3e026ffa8
2025-05-15T16:57:37.525Z      START RequestId: 4881dae7-3211-4dde-90d4-5a8136ea8ea5 Version: $LATEST
2025-05-15T16:57:37.526Z      [ERROR] 2025-05-15T16:57:37.526Z 4881dae7-3211-4dde-90d4-5a8136ea8ea5 Error processing S3 event: 'Records'
2025-05-15T16:57:37.539Z      END RequestId: 4881dae7-3211-4dde-90d4-5a8136ea8ea5
2025-05-15T16:57:37.539Z      REPORT RequestId: 4881dae7-3211-4dde-90d4-5a8136ea8ea5 Duration: 12.81 ms Billed Duration: 13 ms Memory Size: 128 MB Max Memory Used: 35 MB Init Duration: 80.31 ms
2025-05-15T16:58:50.098Z      START RequestId: af51a243-037a-48f3-b726-104ef905b60f Version: $LATEST
2025-05-15T16:58:50.116Z      [INFO] 2025-05-15T16:58:50.116Z af51a243-037a-48f3-b726-104ef905b60f New file uploaded: test-file.txt in bucket: your-bucket-name

    [INFO]  2025-05-15T16:58:50.116Z      af51a243-037a-48f3-b726-104ef905b60f    New file uploaded: test-file.txt in bucket: your-bucket-name

2025-05-15T16:58:50.137Z      END RequestId: af51a243-037a-48f3-b726-104ef905b60f
2025-05-15T16:58:50.137Z      REPORT RequestId: af51a243-037a-48f3-b726-104ef905b60f Duration: 38.13 ms Billed Duration: 39 ms Memory Size: 128 MB Max Memory Used: 35 MB
                              No newer events at this moment. Auto retry paused. Resume
```

- **Verify logs in CloudWatch** to ensure it captures the uploaded file's name and bucket details.

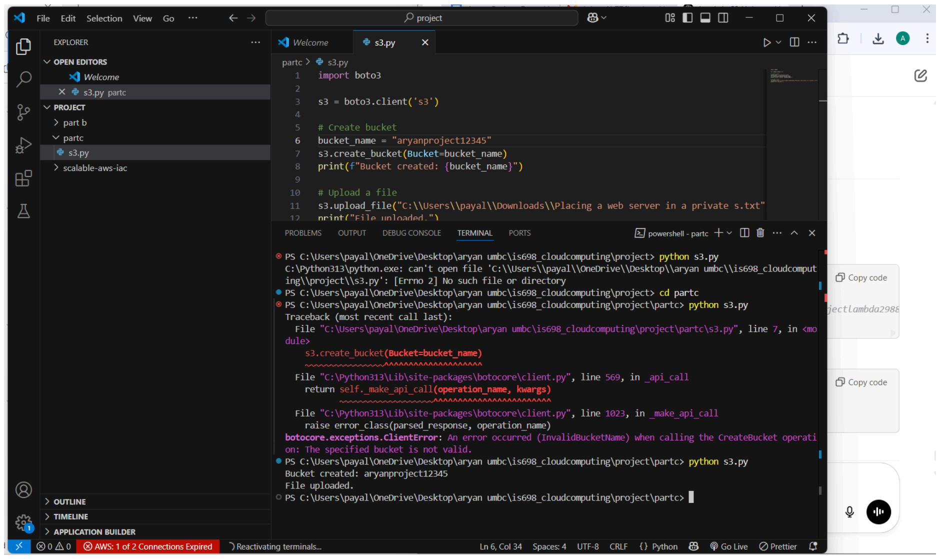

CloudWatch > Log groups > /aws/lambda/LogS3Uploads > 2025/05/15/[$LATEST]26011323331144e3b0938998484830d0
Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                     Message
2025-05-15T16:30:42.819Z      START RequestId: da33639e-039e-4fbf-b0ad-f5e8b5469fb0 Version: $LATEST
2025-05-15T16:30:42.819Z      [INFO] 2025-05-15T16:30:42.819Z da33639e-039e-4fbf-b0ad-f5e8b5469fb0 New file uploaded: s3.txt in bucket: your-bucket-name
2025-05-15T16:30:42.835Z      END RequestId: da33639e-039e-4fbf-b0ad-f5e8b5469fb0
2025-05-15T16:30:42.835Z      REPORT RequestId: da33639e-039e-4fbf-b0ad-f5e8b5469fb0 Duration: 1.66 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 35 MB
2025-05-15T16:30:47.491Z      START RequestId: 6c6e29bd-005c-45f1-9429-509dcf386ef0 Version: $LATEST
2025-05-15T16:30:47.492Z      [INFO] 2025-05-15T16:30:47.492Z 6c6e29bd-005c-45f1-9429-509dcf386ef0 New file uploaded: s3.txt in bucket: your-bucket-name
2025-05-15T16:30:47.495Z      END RequestId: 6c6e29bd-005c-45f1-9429-509dcf386ef0
2025-05-15T16:30:47.495Z      REPORT RequestId: 6c6e29bd-005c-45f1-9429-509dcf386ef0 Duration: 1.66 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 35 MB
2025-05-15T16:33:26.806Z      START RequestId: b5aab957-287e-4d07-8240-71dee589cf64 Version: $LATEST
2025-05-15T16:33:26.817Z      [INFO] 2025-05-15T16:33:26.817Z b5aab957-287e-4d07-8240-71dee589cf64 New file uploaded: s3.txt in bucket: your-bucket-name
2025-05-15T16:33:26.838Z      END RequestId: b5aab957-287e-4d07-8240-71dee589cf64
2025-05-15T16:33:26.838Z      REPORT RequestId: b5aab957-287e-4d07-8240-71dee589cf64 Duration: 30.81 ms Billed Duration: 31 ms Memory Size: 128 MB Max Memory Used: 35 MB
2025-05-15T16:34:27.336Z      START RequestId: c8dc5f03-ce5e-4a40-ad0a-2a863f89b995 Version: $LATEST
2025-05-15T16:34:27.337Z      [INFO] 2025-05-15T16:34:27.337Z c8dc5f03-ce5e-4a40-ad0a-2a863f89b995 New file uploaded: s3.txt in bucket: your-bucket-name

    [INFO]  2025-05-15T16:34:27.337Z      c8dc5f03-ce5e-4a40-ad0a-2a863f89b995    New file uploaded: s3.txt in bucket: your-bucket-name

2025-05-15T16:34:27.358Z      END RequestId: c8dc5f03-ce5e-4a40-ad0a-2a863f89b995
2025-05-15T16:34:27.358Z      REPORT RequestId: c8dc5f03-ce5e-4a40-ad0a-2a863f89b995 Duration: 21.32 ms Billed Duration: 22 ms Memory Size: 128 MB Max Memory Used: 35 MB

    REPORT RequestId: c8dc5f03-ce5e-4a40-ad0a-2a863f89b995  Duration: 21.32 ms      Billed Duration: 22 ms  Memory Size: 128 MB     Max Memory Used: 35 MB

                              No newer events at this moment. Auto retrying... Pause
```

C. Interaction with AWS

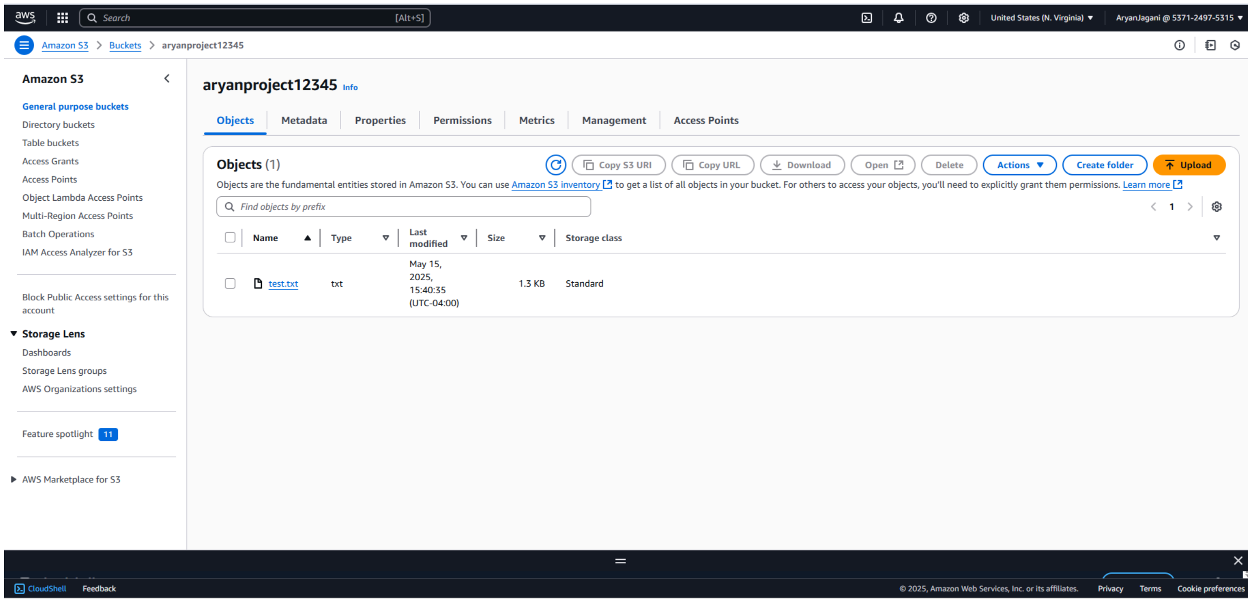
- **Use AWS Console** to verify infrastructure deployment.
- **Use AWS CLI** to interact with EC2, S3, and Lambda.
- **Write Python scripts using Boto3** to:

- Create an S3 bucket and upload a file.



```python
import boto3

s3 = boto3.client('s3')

# Create bucket
bucket_name = "aryanproject12345"
s3.create_bucket(Bucket=bucket_name)
print(f"Bucket created: {bucket_name}")

# Upload a file
s3.upload_file("C:\\Users\\payal\\Downloads\\Placing a web server in a private s.txt"
print("File uploaded.")
```

```
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> python s3.py
C:\Python313\python.exe: can't open file 'C:\\Users\\payal\\OneDrive\\Desktop\\aryan umbc\\is698_cloudcomput
ing\\project\\s3.py': [Errno 2] No such file or directory
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> cd partc
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc> python s3.py
Traceback (most recent call last):
  File "C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc\s3.py", line 7, in <mo
dule>
    s3.create_bucket(Bucket=bucket_name)
    ~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^
  File "C:\Python313\Lib\site-packages\botocore\client.py", line 569, in _api_call
    return self._make_api_call(operation_name, kwargs)
           ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Python313\Lib\site-packages\botocore\client.py", line 1023, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (InvalidBucketName) when calling the CreateBucket operati
on: The specified bucket is not valid.
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc> python s3.py
Bucket created: aryanproject12345
File uploaded.
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc>
```



Amazon S3 > Buckets > aryanproject12345

aryanproject12345 Info

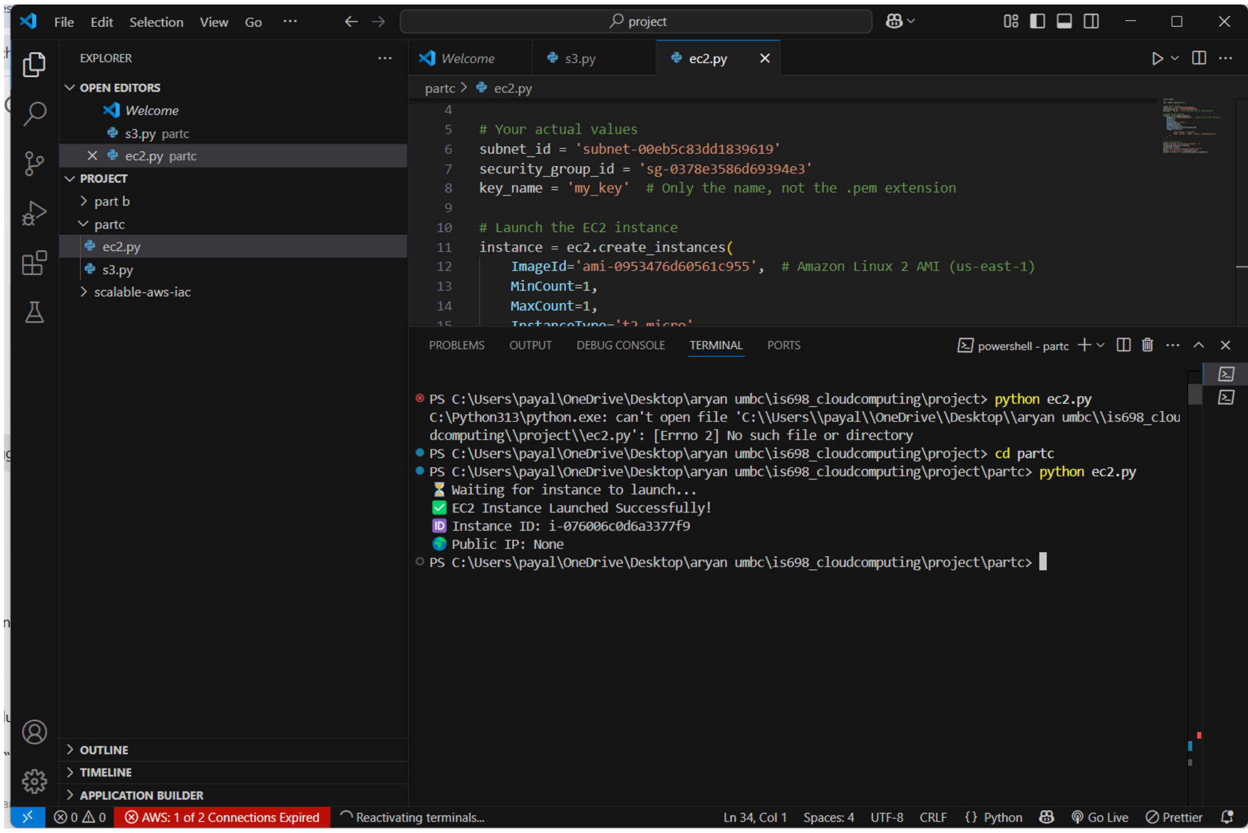
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
test.txt	txt	May 15, 2025, 15:40:35 (UTC-04:00)	1.3 KB	Standard

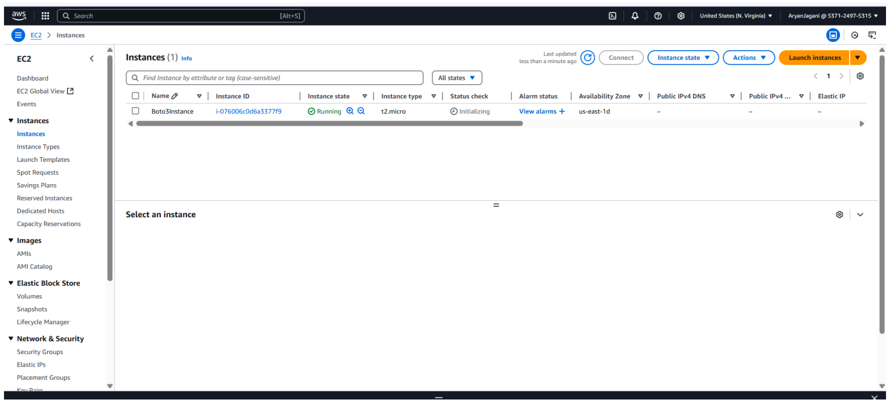
- Retrieve EC2 instance metadata.

Created instance using boto 3



```
# Your actual values
subnet_id = 'subnet-00eb5c83dd1839619'
security_group_id = 'sg-0378e3586d69394e3'
key_name = 'my_key'  # Only the name, not the .pem extension

# Launch the EC2 instance
instance = ec2.create_instances(
    ImageId='ami-0953476d60561c955',  # Amazon Linux 2 AMI (us-east-1)
    MinCount=1,
    MaxCount=1,
```

```
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> python ec2.py
C:\Python313\python.exe: can't open file 'C:\\Users\\payal\\OneDrive\\Desktop\\aryan umbc\\is698_clou
dcomputing\\project\\ec2.py': [Errno 2] No such file or directory
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> cd partc
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc> python ec2.py
⏳ Waiting for instance to launch...
✅ EC2 Instance Launched Successfully!
🆔 Instance ID: i-076006c0d6a3377f9
🌍 Public IP: None
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc>
```

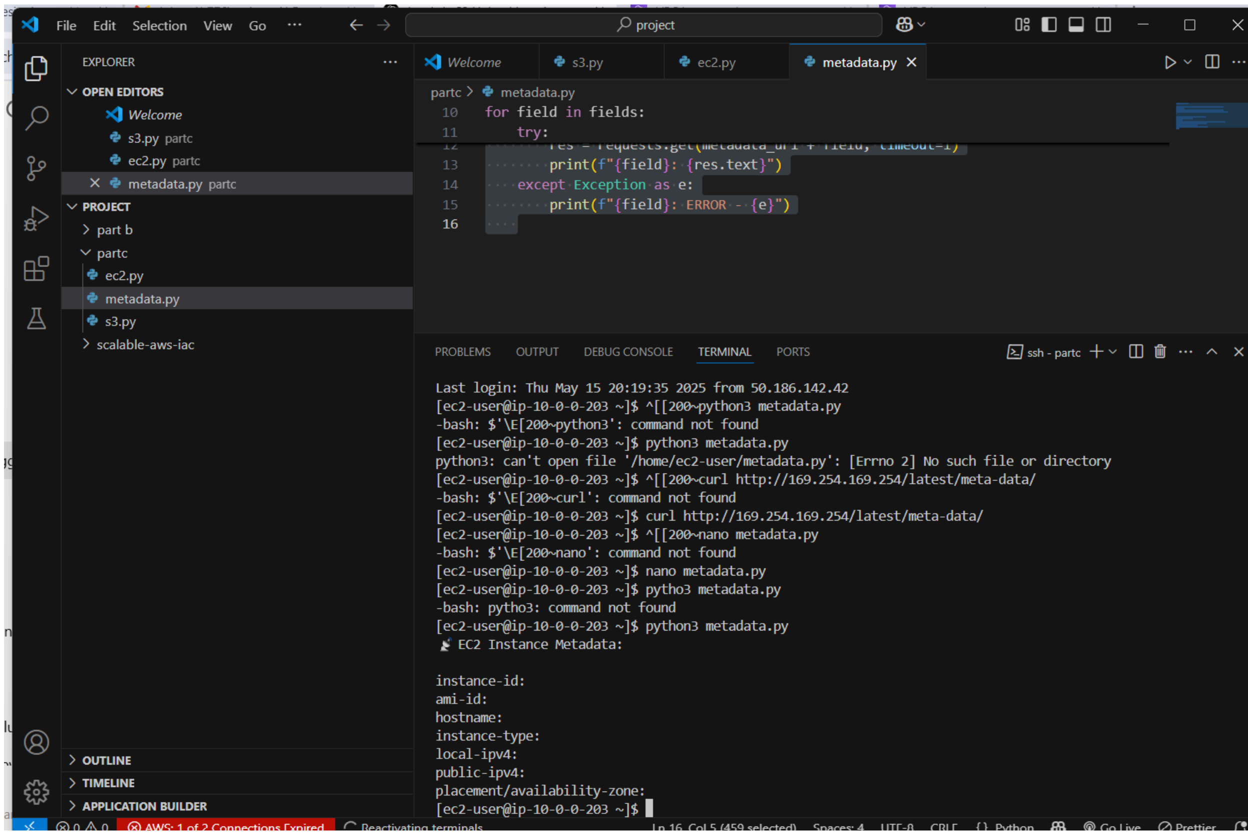


Instances (1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Boto3Instance	i-076006c0d6a3377f9	Running	t2.micro	Initializing	View alarms +	us-east-1d	–	–	–

Retrieved details from instance

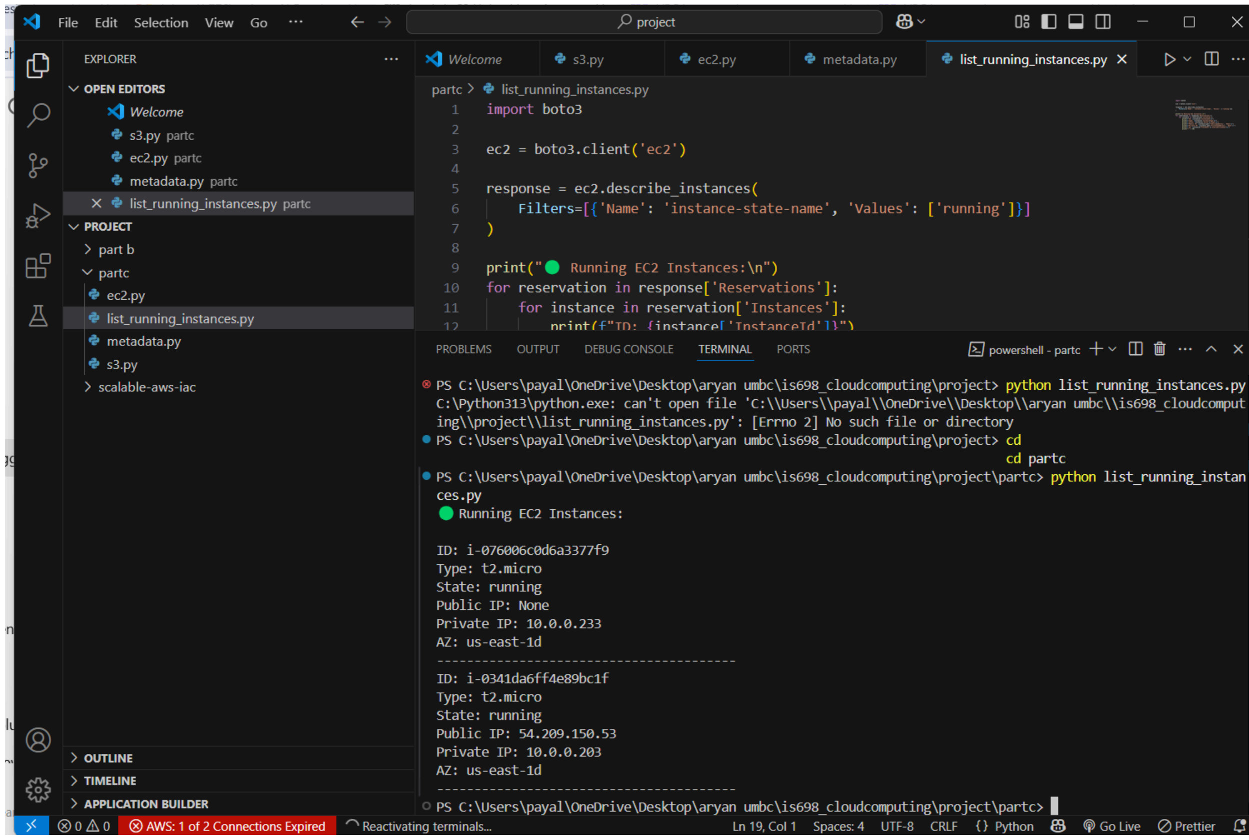
It was an empty instance so wo dont have anything



```python
for field in fields:
    try:
        res = requests.get(metadata_url + field, timeout=1)
        print(f"{field}: {res.text}")
    except Exception as e:
        print(f"{field}: ERROR - {e}")
    
```

```
Last login: Thu May 15 20:19:35 2025 from 50.186.142.42
[ec2-user@ip-10-0-0-203 ~]$ ^[[200~python3 metadata.py
-bash: $'\E[200~python3': command not found
[ec2-user@ip-10-0-0-203 ~]$ python3 metadata.py
python3: can't open file '/home/ec2-user/metadata.py': [Errno 2] No such file or directory
[ec2-user@ip-10-0-0-203 ~]$ ^[[200~curl http://169.254.169.254/latest/meta-data/
-bash: $'\E[200~curl': command not found
[ec2-user@ip-10-0-0-203 ~]$ curl http://169.254.169.254/latest/meta-data/
[ec2-user@ip-10-0-0-203 ~]$ ^[[200~nano metadata.py
-bash: $'\E[200~nano': command not found
[ec2-user@ip-10-0-0-203 ~]$ nano metadata.py
[ec2-user@ip-10-0-0-203 ~]$ pytho3 metadata.py
-bash: pytho3: command not found
[ec2-user@ip-10-0-0-203 ~]$ python3 metadata.py
 EC2 Instance Metadata:

instance-id:
ami-id:
hostname:
instance-type:
local-ipv4:
public-ipv4:
placement/availability-zone:
[ec2-user@ip-10-0-0-203 ~]$
```

- List running EC2 instances.


```python
import boto3

ec2 = boto3.client('ec2')

response = ec2.describe_instances(
    Filters=[{'Name': 'instance-state-name', 'Values': ['running']}]
)

print("🟢 Running EC2 Instances:\n")
for reservation in response['Reservations']:
    for instance in reservation['Instances']:
        print(f"ID: {instance['InstanceId']}")
```

```
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> python list_running_instances.py
C:\Python313\python.exe: can't open file 'C:\\Users\\payal\\OneDrive\\Desktop\\aryan umbc\\is698_cloudcomput
ing\\project\\list_running_instances.py': [Errno 2] No such file or directory
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project> cd
                                                                             cd partc
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc> python list_running_instan
ces.py
🟢 Running EC2 Instances:

ID: i-076006c0d6a3377f9
Type: t2.micro
State: running
Public IP: None
Private IP: 10.0.0.233
AZ: us-east-1d
----------------------------------------
ID: i-0341da6ff4e89bc1f
Type: t2.micro
State: running
Public IP: 54.209.150.53
Private IP: 10.0.0.203
AZ: us-east-1d
----------------------------------------
PS C:\Users\payal\OneDrive\Desktop\aryan umbc\is698_cloudcomputing\project\partc>
```

- Invoke the AWS Lambda function manually.

  → Lambda function was previously deployed and tested for S3 event logging. It has since been deleted after verification and is not available for manual invocation at this time.

---

3. GitHub Integration

- **Create a GitHub repository** for the project.
- The repository should include:
  - **Terraform scripts**
  - **CloudFormation templates**
  - **Python Boto3 scripts**
  - **README file** with project setup instructions
  - **Architecture Diagram**
- **Regular commits** must be made to track progress.
- **Students must submit their GitHub repository link** as part of the project submission.

Challenges Faced and Solutions Implemented

Throughout the project, several technical challenges were encountered and addressed effectively. One major challenge was managing dependencies during VPC deletion and redeployment. To resolve this, the resources were carefully decommissioned in a specific order using Terraform and CloudFormation, ensuring dependencies were respected. Another issue involved EC2 instances failing to connect to the RDS database. This was resolved by properly configuring the security groups to allow inbound MySQL traffic on port 3306 from the EC2 security group, and confirming both resources resided within the same VPC.

There were also permission issues with the Lambda function, which initially could not log S3 uploads to CloudWatch. This was fixed by attaching a custom IAM execution role to the function with permissions for `logs:PutLogEvents` and `s3:GetObject`. File upload problems using SCP were also encountered, typically due to path errors or incorrect permissions. These were resolved by using the full path in quotes and verifying the correct key and EC2 IP address.

CloudFormation template validation errors, including null values and syntax mistakes, were resolved using the AWS CLI validator and code editor extensions. Terraform occasionally caused resource duplication or state conflicts, which were handled by inspecting and correcting the state file using commands like `terraform state list` and `terraform taint`. Finally, some Boto3 scripts failed due to incorrect region or profile settings, which were fixed by ensuring proper AWS CLI configuration and explicitly setting the required parameters in the script.

Future Improvements for Better Scalability and Automation

To enhance scalability and automation, several improvements can be made. First, a CI/CD pipeline using GitHub Actions and AWS CodeDeploy can automate application builds, tests, and deployments to EC2 instances, streamlining updates. Introducing API Gateway to expose Lambda functions over HTTP would allow external access in a secure and managed way. Additionally, integrating AWS Step Functions can help automate complex workflows like validating uploaded files before logging them.

Monitoring and reliability can be improved by enabling automated RDS backups and setting up CloudWatch alarms for EC2 performance metrics. Adding Amazon CloudFront as a CDN in front of the S3 bucket would ensure faster and more reliable delivery of static assets across regions. To make the infrastructure code more reusable, the Terraform and CloudFormation templates can be refactored to use parameterized variables for different environments such as development, testing, and production.

Furthermore, implementing S3 lifecycle policies can automate the archival or deletion of old files, helping control storage costs. Lastly, performance enhancements on the database side—such as enabling query caching and connection pooling—can further optimize backend performance as the application scales.

- **GitHub repository link-**
- **https://github.com/AryanJ09/IS698**
-
