Assignment 2: AWS Project

COMP5349: Cloud Computing

SID: 540967223

Name: Oviya Balamurugan

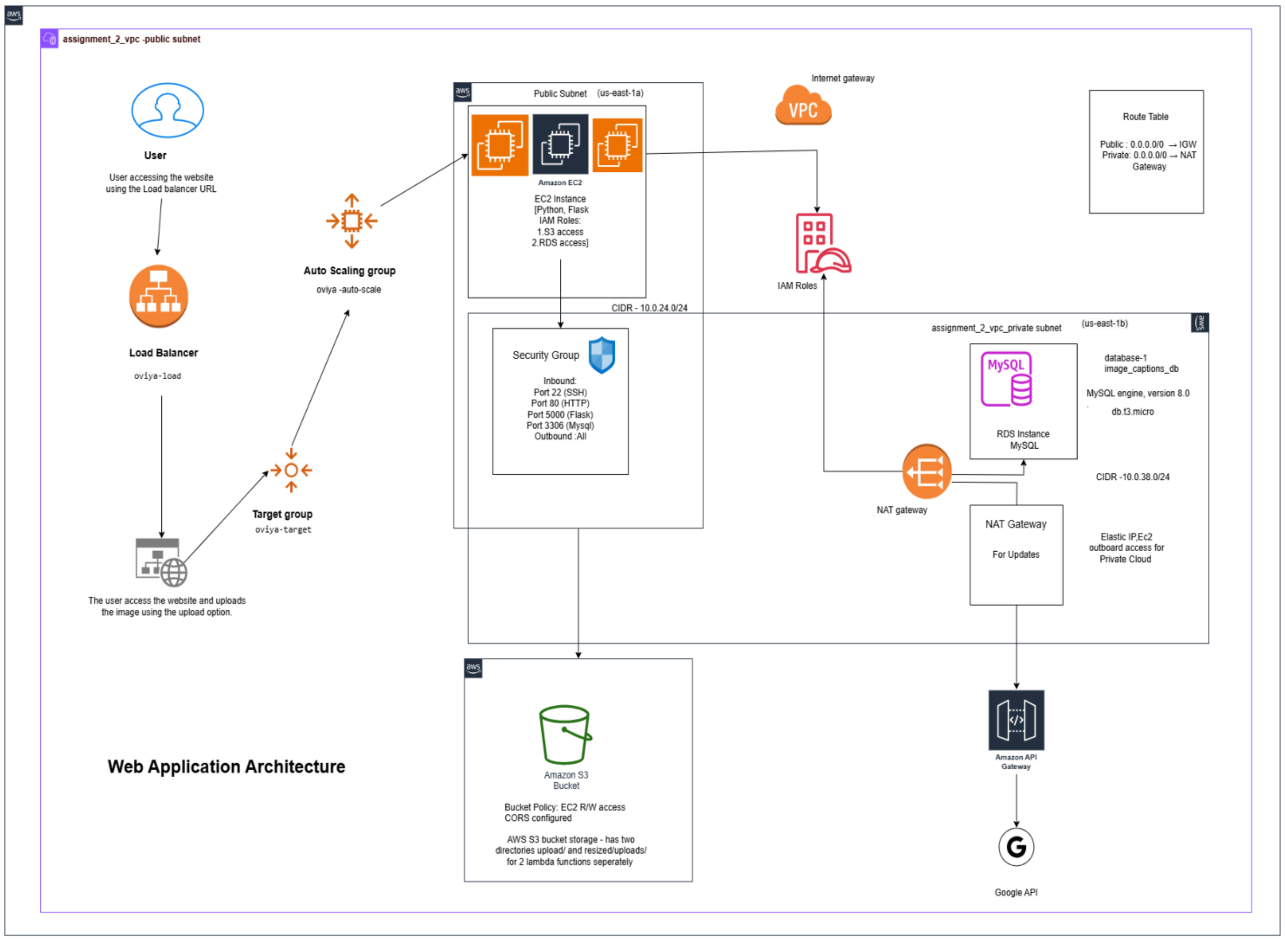
Introduction

In this assignment I have demonstrated an end-to-end deployment which is featured with main functionality such as scalability, uploading the image to the AWS platform, integration of the web-based system and the modern serverless architecture which is achieved by EC2 web-based hosting in turn delivering a robust system and having a cloud native solution. One of the important goals was to upload the images from the user end through the web page and have a real time backend processing with AI generated captions and thumbnail creation. This task is accomplished by 2 lambda functions which are triggered automatically. As I have integrated Auto scaling groups, load balancer, S3, RDS and lambda I was able to attain the availability ensuring the automation and having the best security practices as well.

The website is deployment on the Ec2 instance by using the Auto Scaling groups and which is distributed by load balancer which ensures seamless scaling and traffic balance. The image storage is handled by S3 bucket along with the metadata and captions which is stored in private RDS (MySQL) instance. The deployment process comes under the secure VPC environment using both the public and private subnets so there is network ispolationa nd controlled access of the lambda functions (obal0980-lambda-captions and ovi-lambda-thumbnail) which is triggered by the S3 bucket using the Event Bridge. The layers are added in the particular lambda functions. Access control is maintained by the IAM roles, security groups, subnetting and connecting Lambda and RDS in a secure way.

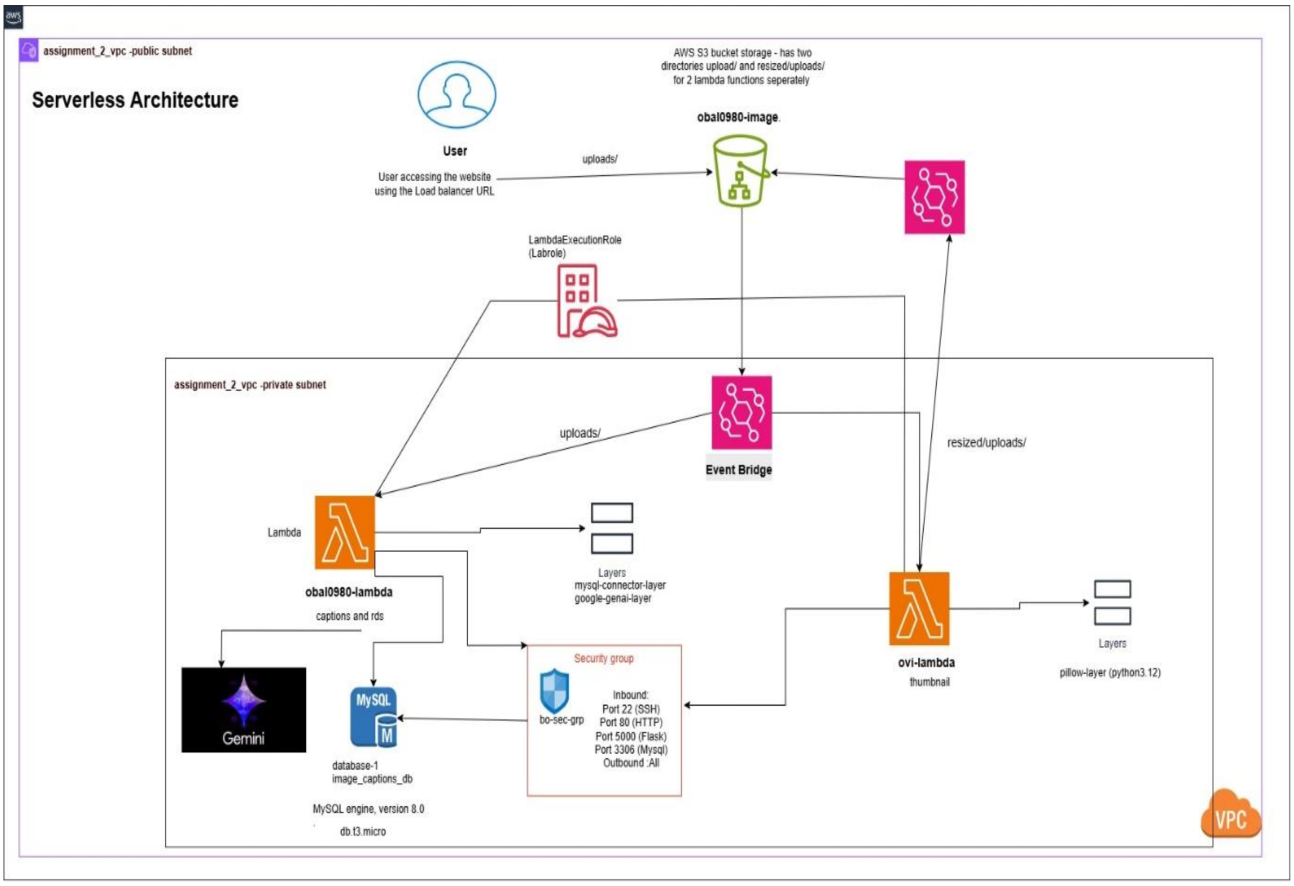
Web Application Architecture:

The diagram effortlessly distinguishes the VPC with public and private subnets, clearly showing secure network partitioning between the web app (ALB) and the RDS database. It highlights the S3 bucket as an integration point of integration, used for uploads by users as well as an initiating point for backend Lambda processing.


assignment_2_vpc -public subnet
User
User accessing the website using the Load balancer URL
Load Balancer
oviya-load
The user access the website and uploads the image using the upload option.
Target group
oviya-target
Auto Scaling group
oviya -auto-scale
Public Subnet (us-east-1a)
Amazon EC2
EC2 Instance [Python, Flask IAM Roles: 1.S3 access 2.RDS access]
CIDR - 10.0.24.0/24
Security Group
Inbound: Port 22 (SSH) Port 80 (HTTP) Port 5000 (Flask) Port 3306 (Mysql) Outbound :All
Internet gateway
VPC
IAM Roles
Route Table
Public : 0.0.0.0/0 → IGW Private: 0.0.0.0/0 → NAT Gateway
assignment_2_vpc_private subnet
(us-east-1b)
MySQL
RDS Instance MySQL
database-1 image_captions_db
MySQL engine, version 8.0
db.t3.micro
CIDR -10.0.38.0/24
NAT gateway
NAT Gateway For Updates
Elastic IP,Ec2 outboard access for Private Cloud
Web Application Architecture
Amazon S3 Bucket
Bucket Policy: EC2 R/W access CORS configured
AWS S3 bucket storage - has two directories upload/ and resized/uploads/ for 2 lambda functions seperately
Amazon API Gateway
Google API

Serverless Architecture:

The diagram itself is a wonderful representation of the serverless event-driven process, showcasing how the S3 ObjectCreated events invoke the Lambda functions so gracefully through EventBridge orchestration within the secure private subnetted environment. It significantly illustrates separation of the processing tasks, with the caption-generating functionality leveraging external Gemini API integration and RDS connectivity and the thumbnail functionality being standalone with S3-to-S3 processing, both leveraging Lambda layers for optimal dependency management as well as cross-security group configuration.



Web Application Deployment:

Compute Environment:

EC2 with Auto Scaling Group (ASG):

The web application is deployed on the EC2 instance and is managed by an Auto Scaling group (ASG) in my case I have named it as oviya-auto-scale. The ASG adjusts the number of instances based on the real time demand which ensures the high availability and cost effectiveness.

Web Application Deployment:

Compute Environment:

Configuration of Auto Scaling group:

Parameter	Value / Description
Auto Scaling Group Name	oviya-auto-scale
Min / Max Capacity	2 / 5
Scaling Policy	Scale out >60% CPU, scale in <30% CPU
Subnets	Public subnets in us-east-1a, us-east-1b

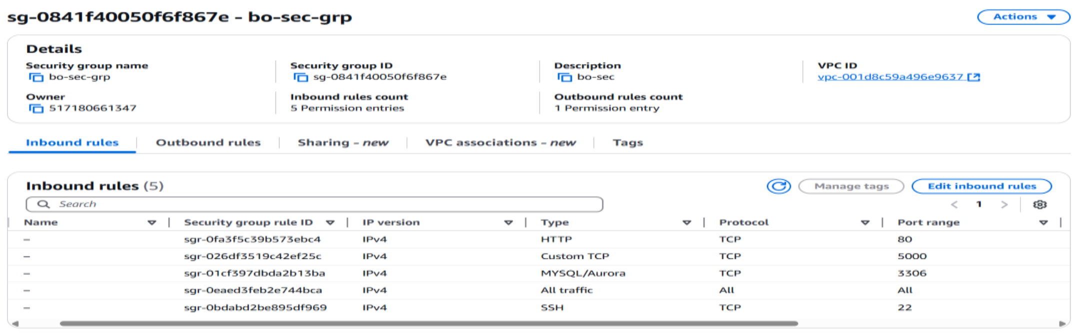
Launch Template: I have used this feature to setup my key features of EC2 which is detailed in the table below.

Configuration	values
AMI ID	ami-0953476d60561c955
Instance type	t2.micro
Key pair name	vockey
Security groups	bo-sec-grp
Security group IDs	sg-0841f40050f6f867e

VPC and Subnets:

- The services are present in the assignment_2_vpc which is in the subnets public and private.
- **Public Subnets**: EC2, ALB.
- **Private Subnets**: RDS and Lambda. (Since the there is a connection between the rds and lambda)

Security groups:



sg-0841f40050f6f867e - bo-sec-grp

Details

Security group name	Security group ID	Description	VPC ID
bo-sec-grp	sg-0841f40050f6f867e	bo-sec	vpc-001d8c59a496e9637

Owner	Inbound rules count	Outbound rules count
517180661347	5 Permission entries	1 Permission entry

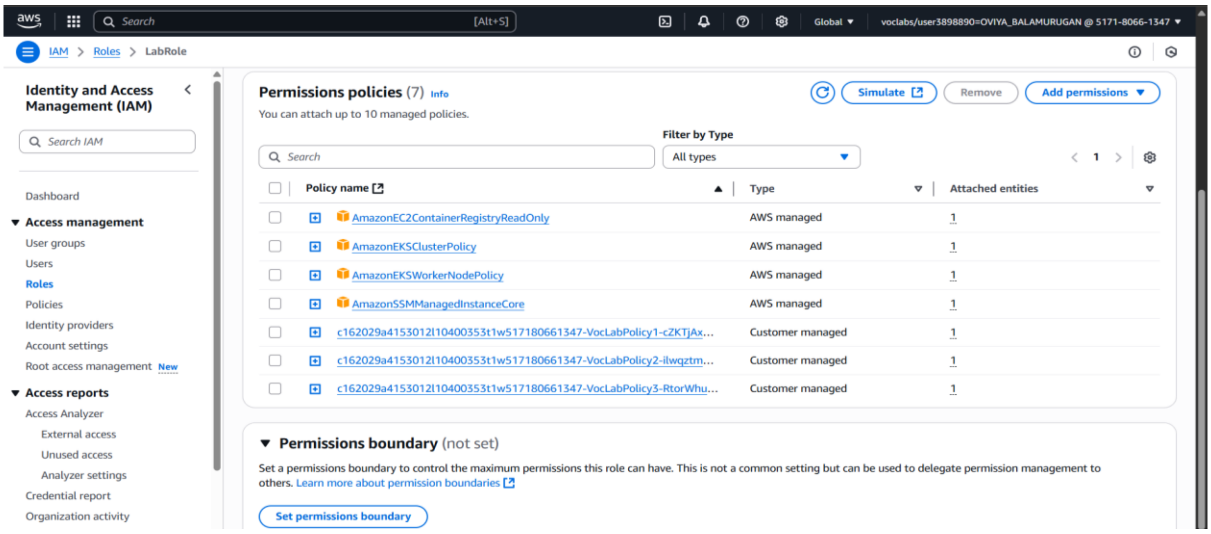
Inbound rules (5)

Name	Security group rule ID	IP version	Type	Protocol	Port range
–	sgr-0fa3f5c39b573ebc4	IPv4	HTTP	TCP	80
–	sgr-026df3519c42ef25c	IPv4	Custom TCP	TCP	5000
–	sgr-01cf397dbda2b13ba	IPv4	MYSQL/Aurora	TCP	3306
–	sgr-0eaed3feb2e744bca	IPv4	All traffic	All	All
–	sgr-0bdabd2be895df969	IPv4	SSH	TCP	22

Security Group	Purpose	Key Rules
bo-sec-grp	Web server access	Allow HTTP (80), HTTPS (443)
RDS SG	DB protection	Allow MySQL (3306) from EC2/Lambda only
Bastion SG	Admin access	Allow SSH (22) from my personal IP

IAM Roles:

- **EC2 Role:** Grants access to S3 (obal0980-image) and RDS (database-1) via Secrets Manager.
- **Lambda Role:** Allows S3 read/write, RDS access, and VPC attachment for private subnet connectivity.
- IAM roles provide temporary, fine-grained permissions that enable services to interact securely without using long-term credentials.



Permissions policies (7) Info

You can attach up to 10 managed policies.

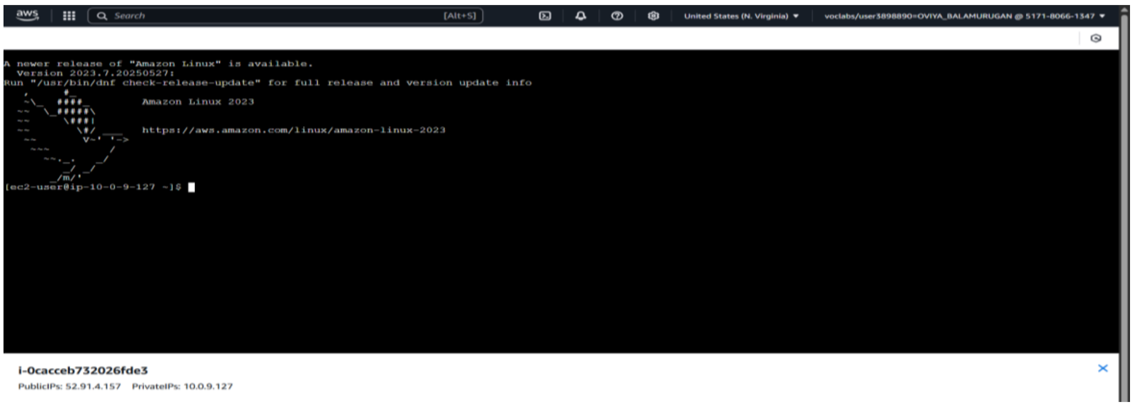
Policy name	Type	Attached entities
AmazonEC2ContainerRegistryReadOnly	AWS managed	1
AmazonEKSClusterPolicy	AWS managed	1
AmazonEKSWorkerNodePolicy	AWS managed	1
AmazonSSMManagedInstanceCore	AWS managed	1
c162029a4153012l10400353t1w517180661347-VocLabPolicy1-cZKTjAx...	Customer managed	1
c162029a4153012l10400353t1w517180661347-VocLabPolicy2-ilwqztm...	Customer managed	1
c162029a4153012l10400353t1w517180661347-VocLabPolicy3-RtorWhu...	Customer managed	1

Permissions boundary (not set)

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others. Learn more about permission boundaries

Admin Access:

The public subnet provides secure SSH access to EC2 instances and the private subnets for administration and troubleshooting.


United States (N. Virginia) voclabs/user3898890=OVIYA_BALAMURUGAN @ 5171-8066-1347

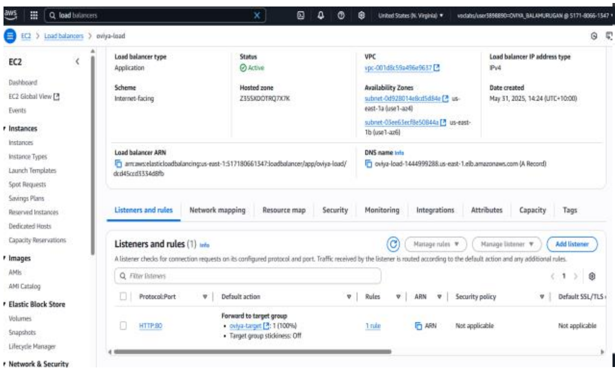
```
A newer release of "Amazon Linux" is available.
  Version 2023.7.20250527:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-9-127 ~]$ 
```

i-0cacceb732026fde3
PublicIPs: 52.91.4.157 PrivateIPs: 10.0.9.127

Component	**Details**
Bastion Host	EC2 in public subnet
Key Pair	Same as web app EC2
IP Restriction	59.102.83.144/32

Load Balancer Configuration

- The Application Load Balancer (ALB) listens on port 80 for HTTP traffic and directs it to a target group of EC2 instances which are spread across multiple Availability Zones, so there is no single server that gets bogged down.
- Health checks are configured so that only healthy EC2 instances get traffic, providing high availability and seamless failover for your app

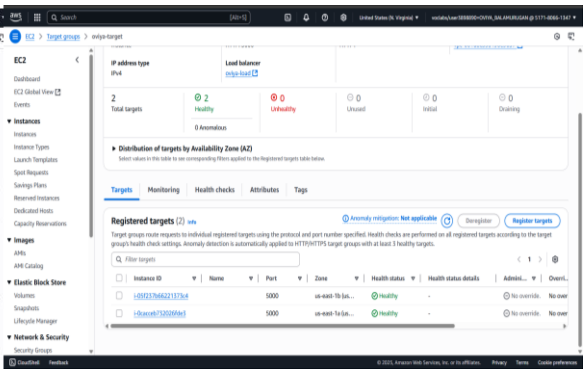

EC2 > Load balancers > oviya-load

Load balancer type: Application
Status: Active
VPC: vpc-001d8c59a496e9637
Load balancer IP address type: IPv4
Scheme: Internet-facing
Hosted zone: Z35SXDOTRQ7X7K
Availability Zones: subnet-0d928014e8cd5d84e us-east-1a (use1-az4); subnet-03ee63ecf8e50844a us-east-1b (use1-az6)
Date created: May 31, 2025, 14:24 (UTC+10:00)
Load balancer ARN: arn:aws:elasticloadbalancing:us-east-1:517180661347:loadbalancer/app/oviya-load/dcd45ccd3334d8fb
DNS name: oviya-load-1444999288.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules (1)

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS
HTTP:80	Forward to target group • oviya-target: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable


EC2 > Target groups > oviya-target

IP address type: IPv4
Load balancer: oviya-load

2 Total targets | 2 Healthy | 0 Unhealthy | 0 Unused | 0 Initial | 0 Draining | 0 Anomalous

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

Registered targets (2)

Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...
i-05f237b66221373c4		5000	us-east-1b (us...	Healthy	-	No override.	No over
i-0cacceb732026fde3		5000	us-east-1a (us...	Healthy	-	No override.	No over

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

ALB Element	**Configuration Details**
Listener	Port 80 (HTTP) → Target Group (`oviya-target`)
Target Group	Registered with ASG, health checks on the endpoint
Subnets	Public subnets.

Database:

- MySQL database is provisioned through Amazon RDS, running in my own VPC's private subnets for security.
- I chose instance type (db.t3.micro), specified storage, and securely handled credentials with AWS Secrets Manager.
- Security groups permit EC2 instances and Lambda functions in the VPC to connect (3306), and public access is disabled.
- Application and Lambdas exchange information through the RDS endpoint and environment variable or Secrets Manager credentials, so that all information is contained within AWS's trusted network.

Database elements	**Values**
DB Engine	MySQL
Instance Class	db.t3.micro
Subnet Group	Private subnets
Security Group	Allows MySQL from Lambda
Authentication	IAM + Secrets Manager

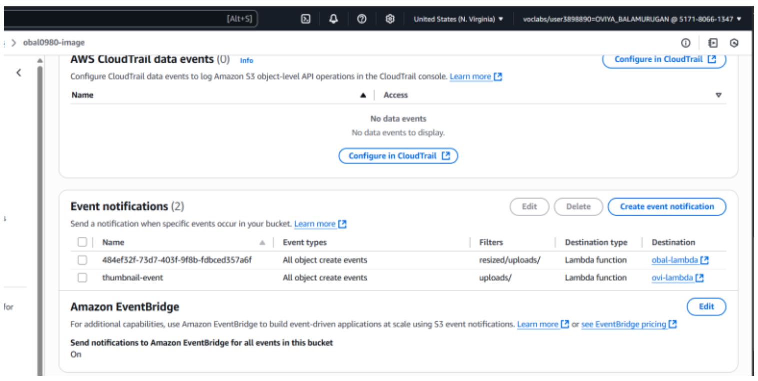
Storage:

The S3 bucket (obal0980-image) is used for storing user-uploaded images and Lambda-generated thumbnails.

S3 Folder	Purpose
uploads/	Raw user image uploads
resized/uploads/	Thumbnails from Lambda

- Event Triggers: S3 ObjectCreated events on uploads/ trigger two Lambda functions:
    - obal0980-lambda-captions (caption generation via Gemini API)
    - ovi-lambda-thumbnail (thumbnail creation)

Event Bridge and Event notification


[Alt+S]
United States (N. Virginia)
voclabs/user3898890=OVIYA_BALAMURUGAN @ 5171-8066-1347
obal0980-image

AWS CloudTrail data events (0) Info

Configure CloudTrail data events to log Amazon S3 object-level API operations in the CloudTrail console. Learn more

Name	Access

No data events

No data events to display.

Configure in CloudTrail

Event notifications (2)

Edit Delete Create event notification

Send a notification when specific events occur in your bucket. Learn more

Name	Event types	Filters	Destination type	Destination
484ef32f-73d7-403f-9f8b-fdbced357a6f	All object create events	resized/uploads/	Lambda function	obal-lambda
thumbnail-event	All object create events	uploads/	Lambda function	ovi-lambda

Amazon EventBridge

Edit

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

Send notifications to Amazon EventBridge for all events in this bucket

On

Serverless Component Deployment:

Event-Driven Architecture:

The serverless components of my application are designed using an **event-driven architecture**. Specifically, the backend automation is triggered by **S3 ObjectCreated events**, which are routed via **Amazon EventBridge** to initiate two Lambda functions in response to image uploads.

Working of the Trigger**:**

- Whenever a user uploads an image to the S3 bucket obal0980-image, specifically under the uploads/ prefix, an **ObjectCreated event** is generated by S3.
- This event is picked up by **Amazon EventBridge**, which has two rules configured:
- **Rule 1** targets the Lambda function obal0980-lambda-captions, which handles image annotation.
- **Rule 2** targets the Lambda function ovi-lambda-thumbnail, which generates image thumbnails.

Resources Interacted With:

Lambda Function	Triggered By	Resources Accessed
obal0980-lambda-captions	S3 ObjectCreated → EventBridge	- S3 bucket (download image) - Gemini API (generate captions), RDS MySQL (database-1), (store metadata)
ovi-lambda-thumbnail	S3 ObjectCreated → EventBridge	- S3 bucket (read original, write resized to resized/uploads/)

- Both Lambda functions are configured with execution roles granting them access to the necessary AWS services securely via IAM, and obal0980-lambda-captions is additionally placed inside a private subnet within the VPC to connect to the private RDS database.

Annotation Function:

(obal0980-lambda-captions)

- The function code is simply a .zip file with only the captions.py script, while all of the dependencies (like Pillow) are included in a separate Lambda custom layer attached to the function.
- Deployment was through the AWS Management Console (or CloudShell), and the function has proper IAM permissions to access S3 (to read images), RDS (to store captions), and the Gemini API (to make to AI-generated captions) with sensitive attributes and secret managed as environment variables and in Secrets Manager.
- The Lambda is configured to run inside the VPC for secure access to the RDS, and it is then automatically triggered on any S3 ObjectCreated event sent through EventBridge, so it creates a back-end processing system and works smoothly in an event-driven fashion.

Attribute	Value
Runtime	Python 3.12
Packaging Format	.zip archive with captions.py script
Deployment Method	AWS Console via CloudShell
Layers Used	pillow-layer for image handling
Event Trigger	S3 ObjectCreated to EventBridge Rule
IAM Role	Allows access to S3, RDS, and HTTP calls to Gemini
VPC Connection	Yes - private RDS access via subnet
Environment Variables	DB credentials (via Secrets Manager), Gemini API Key

Thumbnail Generator:

- The code for this Lambda is also zipped with only the thumbnail.py, while all dependencies are in an attached Lambda layer to keep the deployment package lean and maintainable.
- The function is deployed via the AWS Console with an IAM role that grants permissions to read & write to the S3 bucket so it can process uploaded images and generate thumbnails.

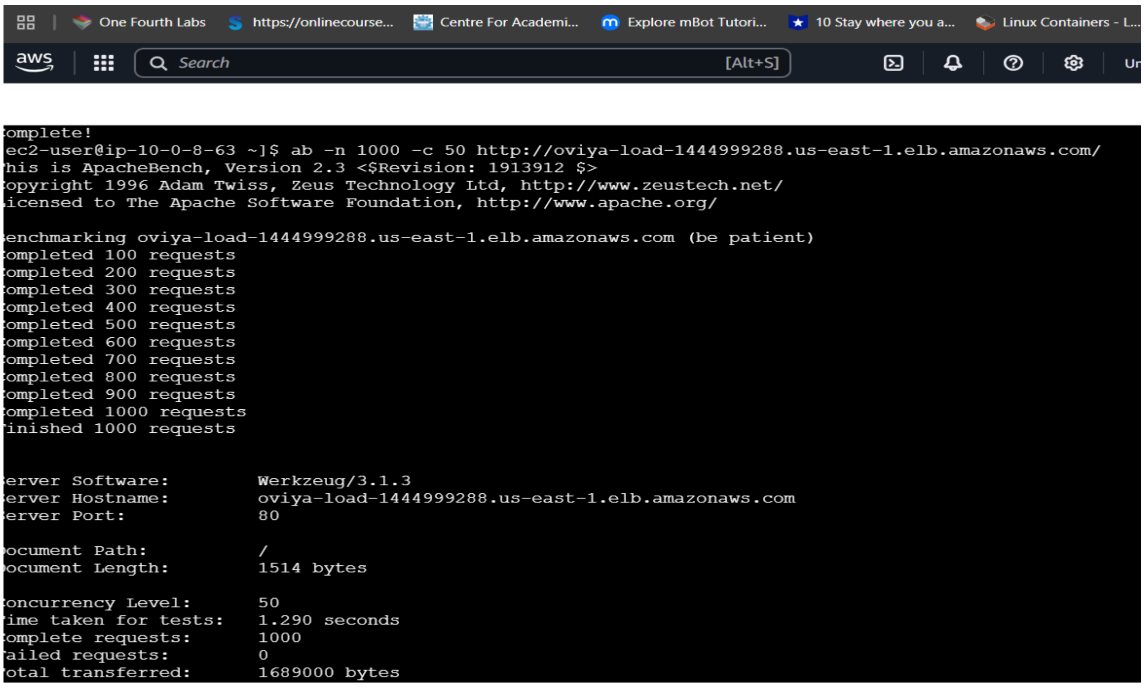
- The code is triggered by S3 ObjectCreated events (by using EventBridge), it processes the image and writes the thumbnail to a specified S3 folder, and there is no need for VPC access since it only communicates with S3.

Attribute	Value
Runtime	Python 3.12
Packaging Format	.zip archive containing thumbnail.py script
Deployment Method	AWS Console (uploaded ZIP from CloudShell terminal)
Layers Used	pillow-layer for image processing
Event Trigger	S3 ObjectCreated event via Amazon EventBridge
IAM Role	Lambda execution role with permission to read from and write to S3
VPC Connection	Not required (no RDS access needed)
Environment Variables	S3 bucket name, thumbnail output path (e.g., resized/)

Auto Scaling:

Configuration	**Value**
Auto Scaling Group Name	oviya-auto-scale
Launch Template	AMI with Flask app, port 5000 exposed
Load Balancer	oviya-load (Application Load Balancer)
Target Group	oviya-target (Port 5000 health check)
Scaling Policy	Target tracking (CPU > 60%)
Initial Capacity	2 instance
Max Capacity	4 instances

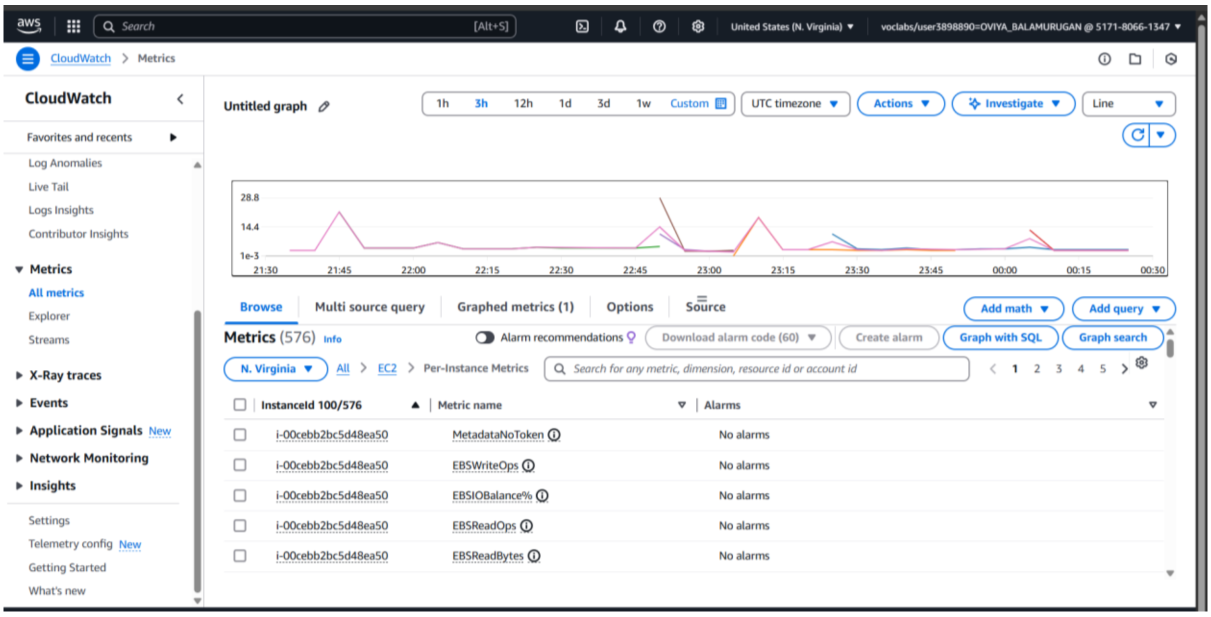
Test Tool	ApacheBench (ab)
Test Command	ab -n 1000 -c 50 http://oviya-load-1444999288.us-east-1.elb.amazonaws.com/



```
omplete!
ec2-user@ip-10-0-8-63 ~]$ ab -n 1000 -c 50 http://oviya-load-1444999288.us-east-1.elb.amazonaws.com/
his is ApacheBench, Version 2.3 <$Revision: 1913912 $>
opyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
icensed to The Apache Software Foundation, http://www.apache.org/

enchmarking oviya-load-1444999288.us-east-1.elb.amazonaws.com (be patient)
ompleted 100 requests
ompleted 200 requests
ompleted 300 requests
ompleted 400 requests
ompleted 500 requests
ompleted 600 requests
ompleted 700 requests
ompleted 800 requests
ompleted 900 requests
ompleted 1000 requests
inished 1000 requests


erver Software:        Werkzeug/3.1.3
erver Hostname:        oviya-load-1444999288.us-east-1.elb.amazonaws.com
erver Port:            80

ocument Path:          /
ocument Length:        1514 bytes

oncurrency Level:      50
ime taken for tests:   1.290 seconds
omplete requests:      1000
ailed requests:        0
otal transferred:      1689000 bytes
```

i-03098d4a5d76c4d34

PublicIPs: 18.209.47.184   PrivateIPs: 10.0.8.63



InstanceId 100/576	Metric name	Alarms
i-00cebb2bc5d48ea50	MetadataNoToken	No alarms
i-00cebb2bc5d48ea50	EBSWriteOps	No alarms
i-00cebb2bc5d48ea50	EBSIOBalance%	No alarms
i-00cebb2bc5d48ea50	EBSReadOps	No alarms
i-00cebb2bc5d48ea50	EBSReadBytes	No alarms

Spikes were detected around 21:45–22:15, likely due to simulated load or application interactions, followed by stabilization.

The auto scaling feature worked as expected. During high traffic situations, when our application demand was particularly high, a new EC2 instance was launched automatically, and the ALB was load balancing traffic to both servers. Once demand calmed down again, the extra instance was terminated, and we confirmed that we are deploying and consuming resources efficiently and dynamically.

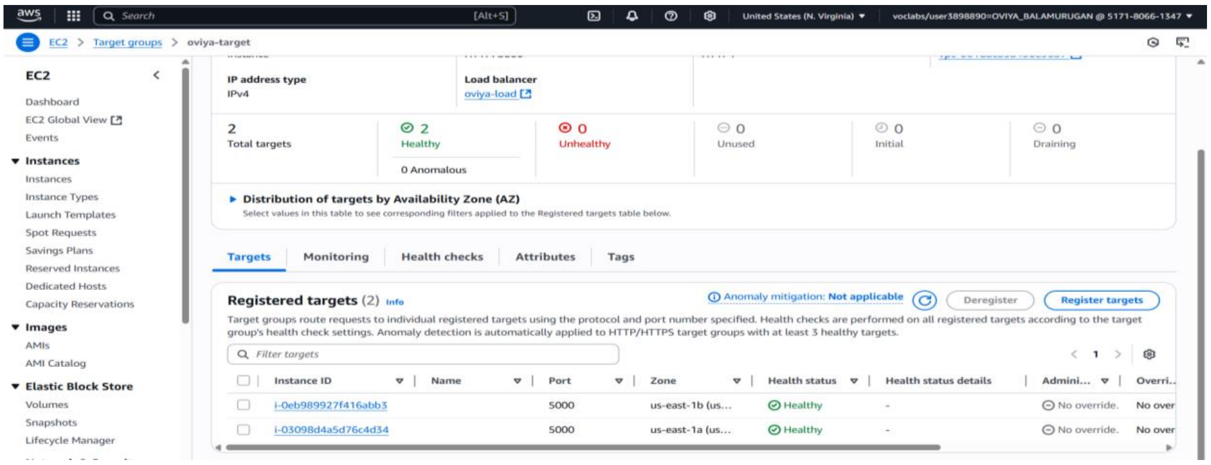
Test Steps

- Accessed web application via ALB endpoint.
- Ran ApacheBench: 1000 requests with concurrency of 50.
- Monitored EC2 scaling, ALB target health, and CPU usage in the AWS Console.

Outcomes & Evidence

- ASG scaled EC2 instances from 1 to 2 due to CPU threshold being exceeded.
- ALB distributed traffic equally between both healthy instances.
- ApacheBench: 1000 requests completed with very high throughput.

Target group health check:



IP address type: IPv4
Load balancer: oviya-load

2	2	0	0	0	0
Total targets	Healthy	Unhealthy	Unused	Initial	Draining
	0 Anomalous				

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

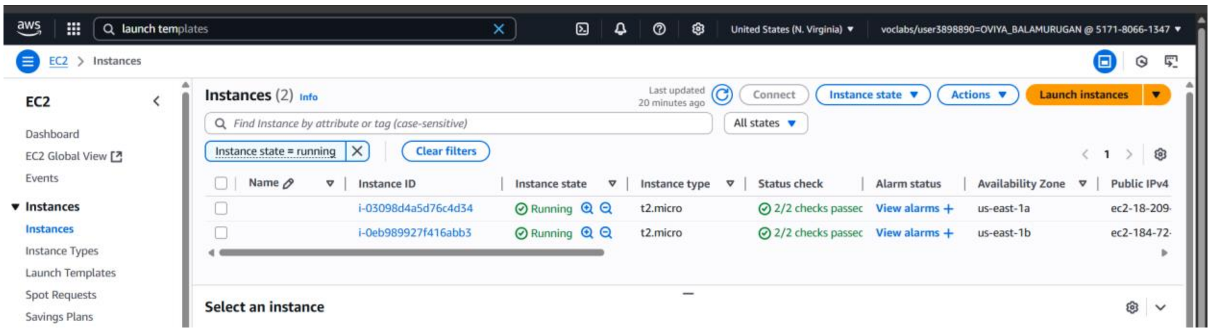
Registered targets (2) Info
Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...
i-0eb989927f416abb3		5000	us-east-1b (us...	Healthy	-	No override.	No over
i-03098d4a5d76c4d34		5000	us-east-1a (us...	Healthy	-	No override.	No over

Ec2 instance creation:



Instances (2) Info
Last updated 20 minutes ago

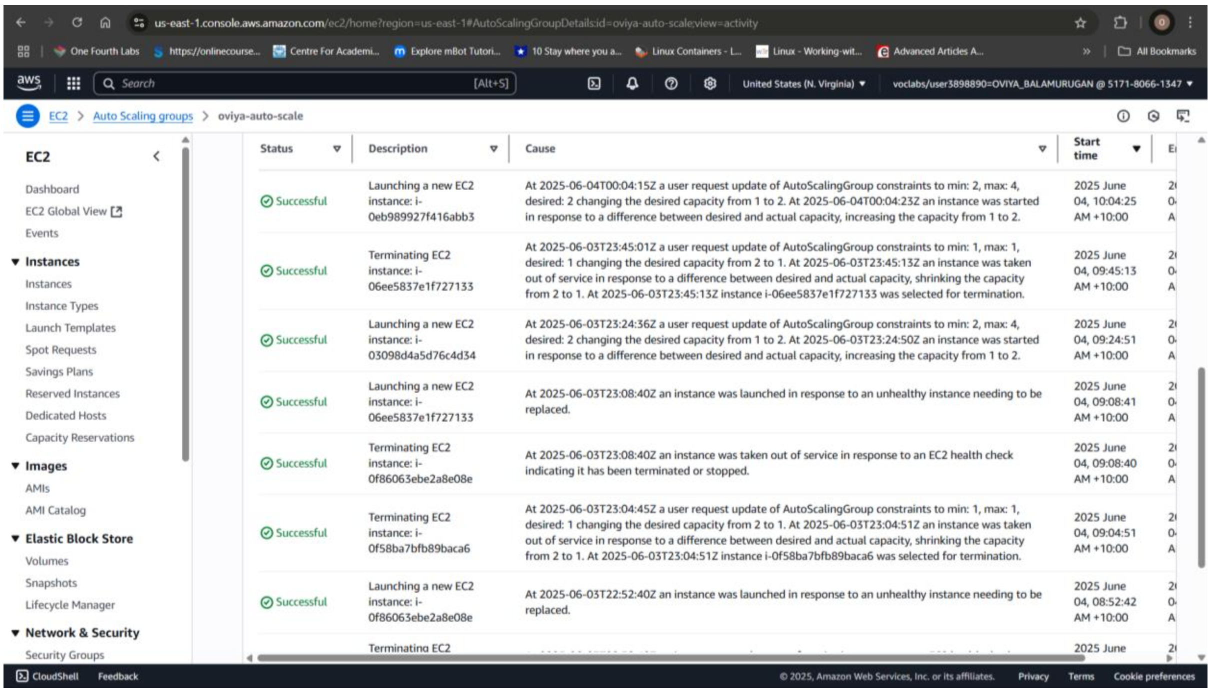
Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
	i-03098d4a5d76c4d34	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-18-209-
	i-0eb989927f416abb3	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-184-72-

Select an instance

The scale in and scale out is shown as launch and terminated in this screenshot.



Status	Description	Cause	Start time
Successful	Launching a new EC2 instance: i-0eb989927f416abb3	At 2025-06-04T00:04:15Z a user request update of AutoScalingGroup constraints to min: 2, max: 4, desired: 2 changing the desired capacity from 1 to 2. At 2025-06-04T00:04:23Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 June 04, 10:04:25 AM +10:00
Successful	Terminating EC2 instance: i-06ee5837e1f727133	At 2025-06-03T23:45:01Z a user request update of AutoScalingGroup constraints to min: 1, max: 1, desired: 1 changing the desired capacity from 2 to 1. At 2025-06-03T23:45:13Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-06-03T23:45:13Z instance i-06ee5837e1f727133 was selected for termination.	2025 June 04, 09:45:13 AM +10:00
Successful	Launching a new EC2 instance: i-03098d4a5d76c4d34	At 2025-06-03T23:24:36Z a user request update of AutoScalingGroup constraints to min: 2, max: 4, desired: 2 changing the desired capacity from 1 to 2. At 2025-06-03T23:24:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 June 04, 09:24:51 AM +10:00
Successful	Launching a new EC2 instance: i-06ee5837e1f727133	At 2025-06-03T23:08:40Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 June 04, 09:08:41 AM +10:00
Successful	Terminating EC2 instance: i-0f86063ebe2a8e08e	At 2025-06-03T23:08:40Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2025 June 04, 09:08:40 AM +10:00
Successful	Terminating EC2 instance: i-0f58ba7bfb89baca6	At 2025-06-03T23:04:45Z a user request update of AutoScalingGroup constraints to min: 1, max: 1, desired: 1 changing the desired capacity from 2 to 1. At 2025-06-03T23:04:51Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-06-03T23:04:51Z instance i-0f58ba7bfb89baca6 was selected for termination.	2025 June 04, 09:04:51 AM +10:00
Successful	Launching a new EC2 instance: i-0f86063ebe2a8e08e	At 2025-06-03T22:52:40Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 June 04, 08:52:42 AM +10:00
	Terminating EC2		2025 June

Conclusion, challenges and lessons learned:

- During this assignment, I was able to develop and deploy a scalable, event-driven image annotation platform on AWS. I did load testing using ApacheBench (ab) to see if the platform would scale and stayed observant of the system with CloudWatch. It was really cool to see the platform scale bi-directionally into/out of running services, which was almost synonymous with high availability, as well as have services utilized effectively.
- While doing this project, I did run into some hiccups along the way, like the package upload limits of Lambda, and I felt somewhat lost with EventBridge rules at the beginning. However, learning how to use Lambda layers eliminated my package size issue, and therefore deployment went a lot more smoothly. Once I learned how to use EventBridge, linking S3 events to my Lambda functions was much easier and intuitive.
- All in all, this project helped increase my confidence with AWS. I learned a lot about networking, permissions, and automating cloud workflows. More importantly, I now feel a lot more confident designing secure, cloud-native solutions that take performance, cost, and practicality into account.