Scalable Web Application with Load Balancing

Services used:
EC2, Application Load Balancer (ELB), Auto Scaling, RDS, S3, Route 53
Skills covered:
VPC, Subnets, Security Groups, EC2 Launch Template, User Data, Auto Scaling

PROJECT GOAL

Deploy a **highly available and scalable web application** that:

- Automatically **scales EC2 instances** based on traffic
- Uses **Load Balancer** to distribute traffic
- Stores data in **RDS**
- Serves static content from **S3**
- Uses **Route 53** for custom domain routing

Project Overview: Scalable Web Application with Load Balancing

This project focuses on deploying a **highly available and scalable web application** using AWS cloud services. The application is designed to automatically handle varying user traffic by distributing requests across multiple servers and scaling resources based on demand.

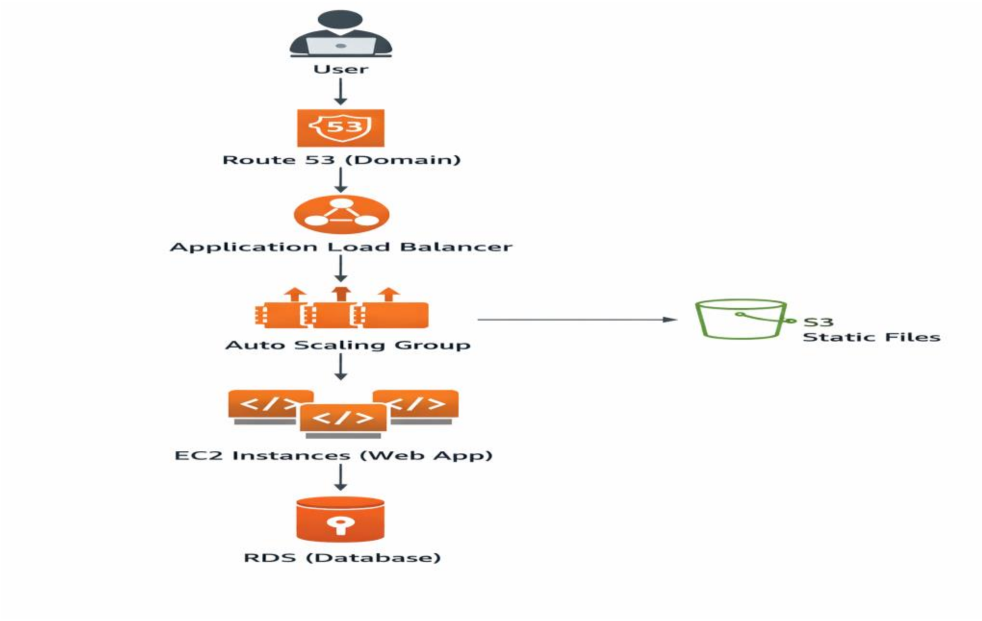
The solution uses **Amazon Route 53** to route user requests to an **Application Load Balancer**, which evenly distributes traffic to EC2 instances managed by an **Auto Scaling Group**. The EC2 instances host the web application and automatically launch or terminate based on CPU utilization. Application data is securely stored in an **Amazon RDS** database, while static content such as images, CSS, and JavaScript files are served from **Amazon S3**.

The entire setup is deployed inside a custom **VPC** with properly configured subnets, route tables, and security groups to ensure network isolation and security. This architecture provides fault tolerance, improved performance, cost efficiency, and scalability, making it suitable for real-world production workloads and enterprise-level applications.

ARCHITECTURE OVERVIEW

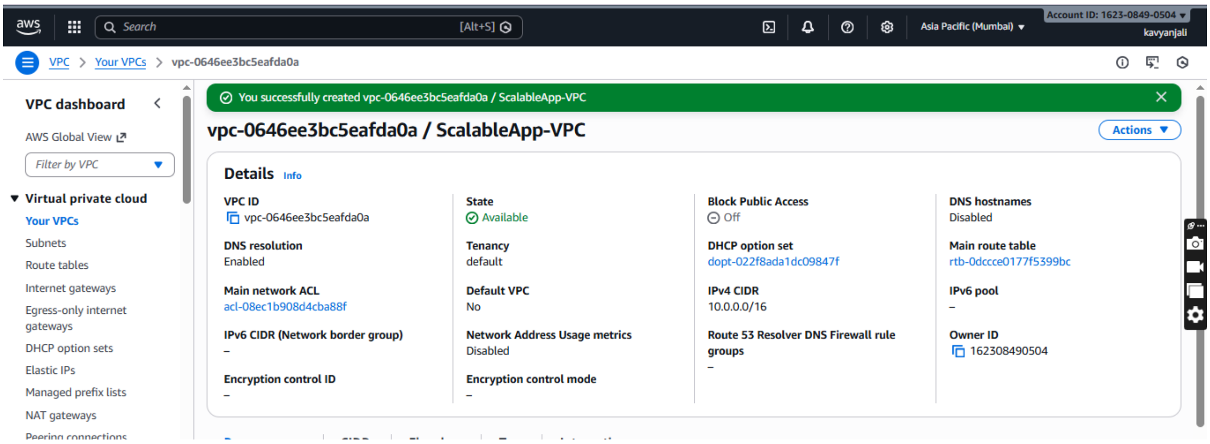
User
↓
Route 53 (Domain)
↓
Application Load Balancer
↓
Auto Scaling Group
↓
EC2 Instances (Web App)
↓
RDS (Database)

Static files → S3



STEP 1: Create VPC

1. Open **AWS Console**
2. Go to **VPC → Your VPCs**
3. Click **Create VPC**
4. Choose:
   - Name: ScalableApp-VPC
   - IPv4 CIDR: 10.0.0.0/16
5. Click **Create VPC**



You successfully created vpc-0646ee3bc5eafda0a / ScalableApp-VPC

vpc-0646ee3bc5eafda0a / ScalableApp-VPC

Details

VPC ID
vpc-0646ee3bc5eafda0a

State
Available

Block Public Access
Off

DNS hostnames
Disabled

DNS resolution
Enabled

Tenancy
default

DHCP option set
dopt-022f8ada1dc09847f

Main route table
rtb-0dccce0177f5399bc

Main network ACL
acl-08ec1b908d4cba88f

Default VPC
No

IPv4 CIDR
10.0.0.0/16

IPv6 pool
–

IPv6 CIDR (Network border group)
–

Network Address Usage metrics
Disabled

Route 53 Resolver DNS Firewall rule groups
–

Owner ID
162308490504

Encryption control ID
–

Encryption control mode
–

STEP 2: Create Subnets

Create 2 Public Subnets (for ALB + EC2)

Go to **VPC → Subnets → Create subnet**

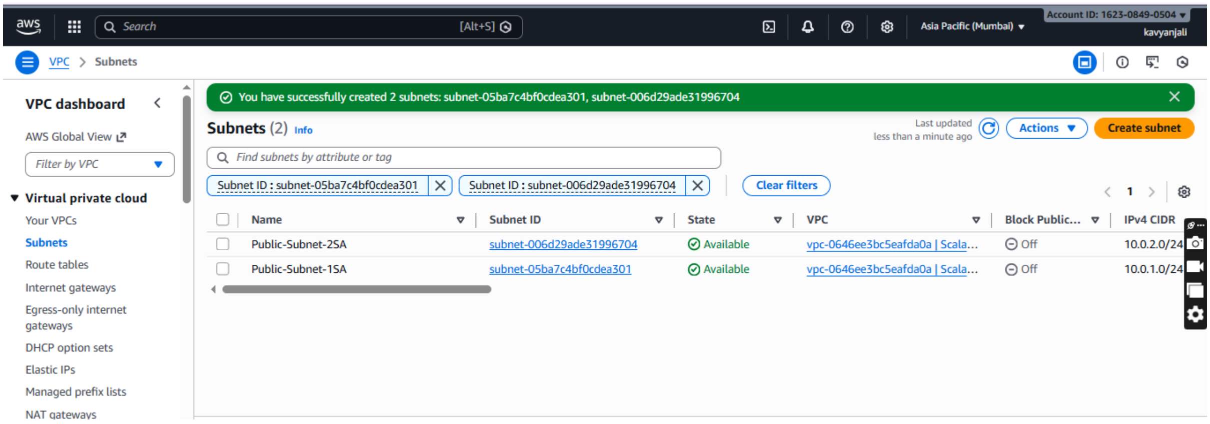
Subnet 1

- Name: Public-Subnet-1
- AZ: ap-south-1a
- CIDR: 10.0.1.0/24

Subnet 2

- Name: Public-Subnet-2
- AZ: ap-south-1b
- CIDR: 10.0.2.0/24

Click **Create subnet**



You have successfully created 2 subnets: subnet-05ba7c4bf0cdea301, subnet-006d29ade31996704

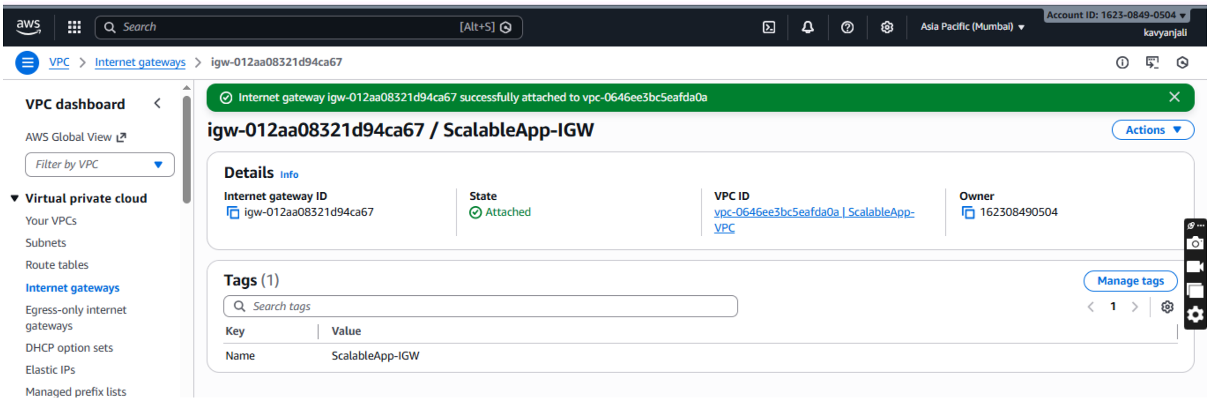
Subnets (2) Info

Subnet ID : subnet-05ba7c4bf0cdea301 ✕ Subnet ID : subnet-006d29ade31996704 ✕

□	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
□	Public-Subnet-2SA	subnet-006d29ade31996704	✓ Available	vpc-0646ee3bc5eafda0a | Scala...	⊖ Off	10.0.2.0/24
□	Public-Subnet-1SA	subnet-05ba7c4bf0cdea301	✓ Available	vpc-0646ee3bc5eafda0a | Scala...	⊖ Off	10.0.1.0/24

STEP 3: Internet Gateway

1. Go to **VPC → Internet Gateways**
2. Click **Create internet gateway**
   - Name: ScalableApp-IGW
3. Attach it to ScalableApp-VPC



Internet gateway igw-012aa08321d94ca67 successfully attached to vpc-0646ee3bc5eafda0a

igw-012aa08321d94ca67 / ScalableApp-IGW

Details Info

Internet gateway ID: igw-012aa08321d94ca67
State: Attached
VPC ID: vpc-0646ee3bc5eafda0a | ScalableApp-VPC
Owner: 162308490504

Tags (1)

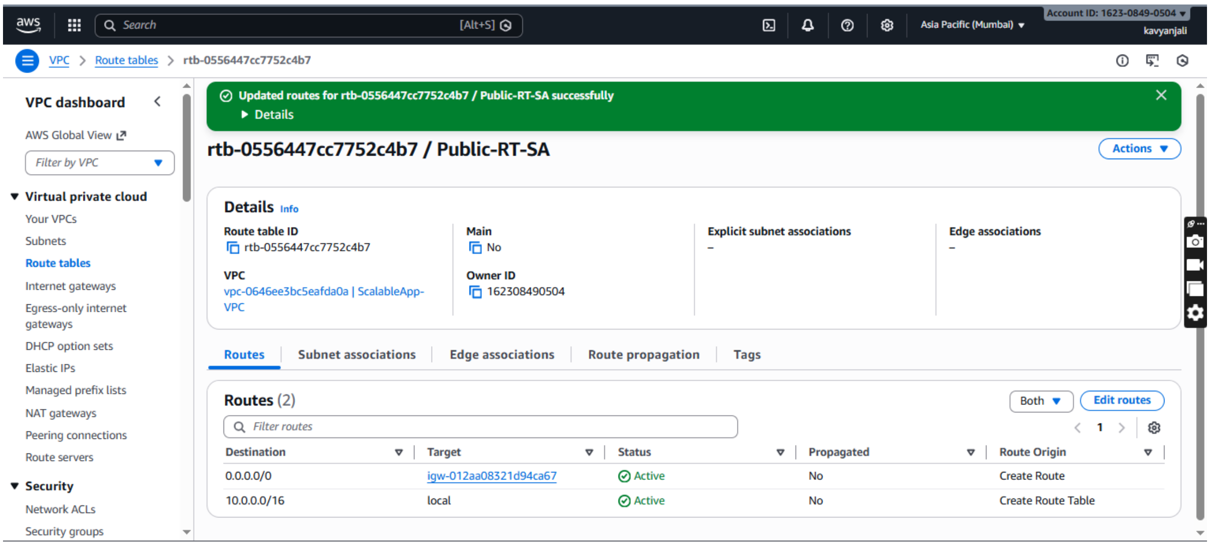
Key	Value
Name	ScalableApp-IGW

STEP 4: Route Table

1. Go to **VPC → Route Tables**
2. Create route table:

- Name: Public-RT
- VPC: ScalableApp-VPC

3. Edit routes:
   - Destination: 0.0.0.0/0
   - Target: Internet Gateway
4. Associate both public subnets



Updated routes for rtb-0556447cc7752c4b7 / Public-RT-SA successfully

Details

rtb-0556447cc7752c4b7 / Public-RT-SA

Details Info

Route table ID: rtb-0556447cc7752c4b7

Main: No

Explicit subnet associations: –

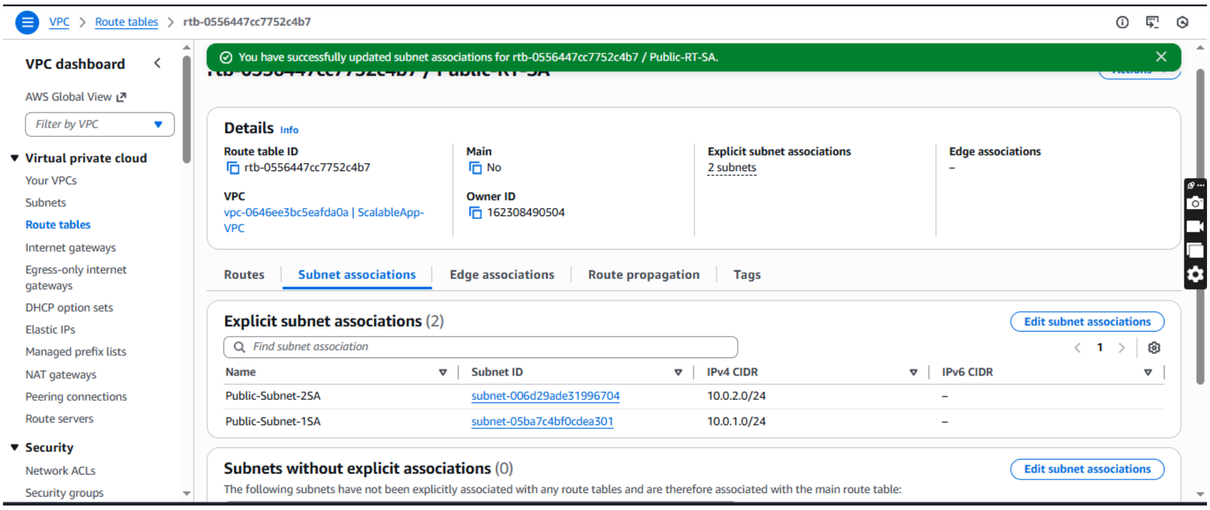
Edge associations: –

VPC: vpc-0646ee3bc5eafda0a | ScalableApp-VPC

Owner ID: 162308490504

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-012aa08321d94ca67	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table



You have successfully updated subnet associations for rtb-0556447cc7752c4b7 / Public-RT-SA.

Details Info

Route table ID: rtb-0556447cc7752c4b7

Main: No

Explicit subnet associations: 2 subnets

Edge associations: –

VPC: vpc-0646ee3bc5eafda0a | ScalableApp-VPC

Owner ID: 162308490504

Explicit subnet associations (2)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Public-Subnet-2SA	subnet-006d29ade31996704	10.0.2.0/24	–
Public-Subnet-1SA	subnet-05ba7c4bf0cdea301	10.0.1.0/24	–

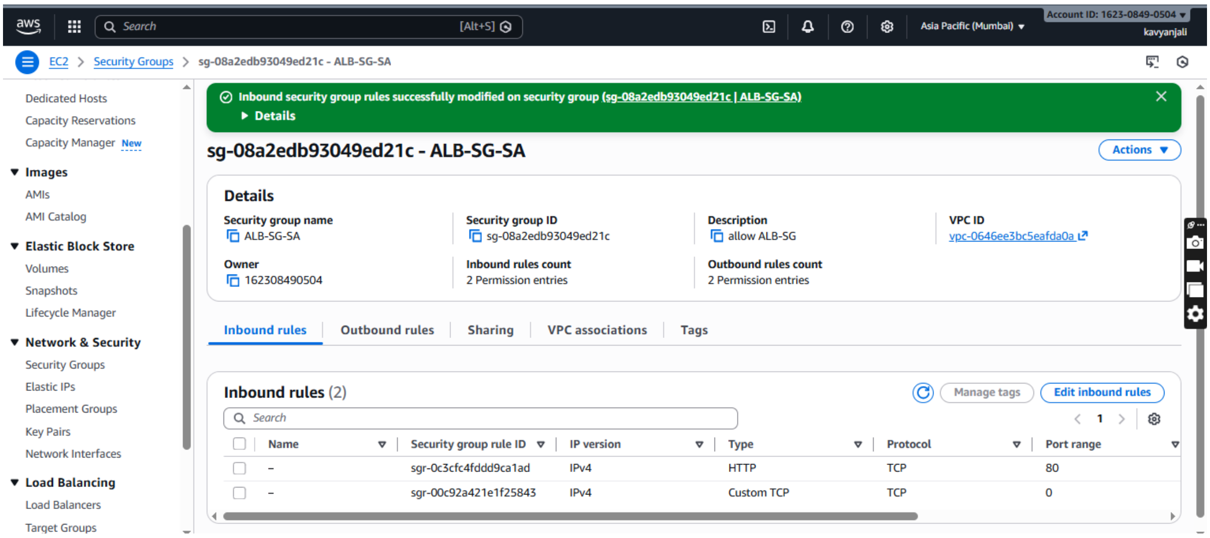
Subnets without explicit associations (0)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

STEP 5: Security Groups

1.ALB Security Group

- Name: ALB-SG
- Inbound:
  - HTTP (80) → 0.0.0.0/0



Inbound security group rules successfully modified on security group (sg-08a2edb93049ed21c | ALB-SG-SA)

▶ Details

sg-08a2edb93049ed21c - ALB-SG-SA

Details

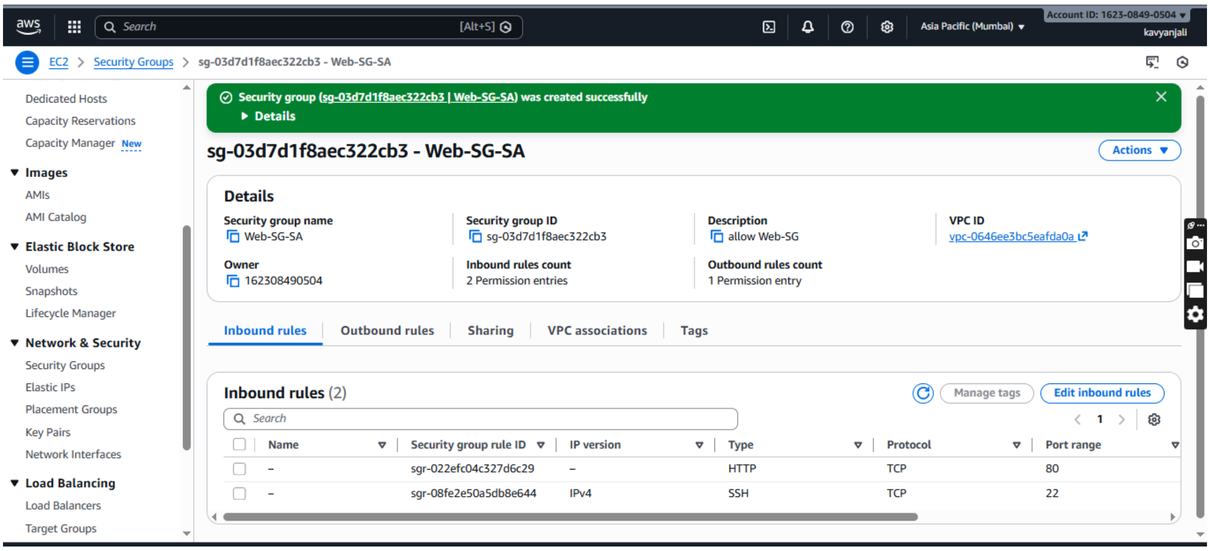
Security group name ALB-SG-SA	Security group ID sg-08a2edb93049ed21c	Description allow ALB-SG	VPC ID vpc-0646ee3bc5eafda0a
Owner 162308490504	Inbound rules count 2 Permission entries	Outbound rules count 2 Permission entries	

Inbound rules (2)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range
□	–	sgr-0c3cfc4fddd9ca1ad	IPv4	HTTP	TCP	80
□	–	sgr-00c92a421e1f25843	IPv4	Custom TCP	TCP	0

2. EC2 Security Group

- Name: Web-SG
- Inbound:
  - HTTP (80) → ALB-SG
  - SSH (22) → Your IP



Security group (sg-03d7d1f8aec322cb3 | Web-SG-SA) was created successfully

▶ Details

sg-03d7d1f8aec322cb3 - Web-SG-SA

Details

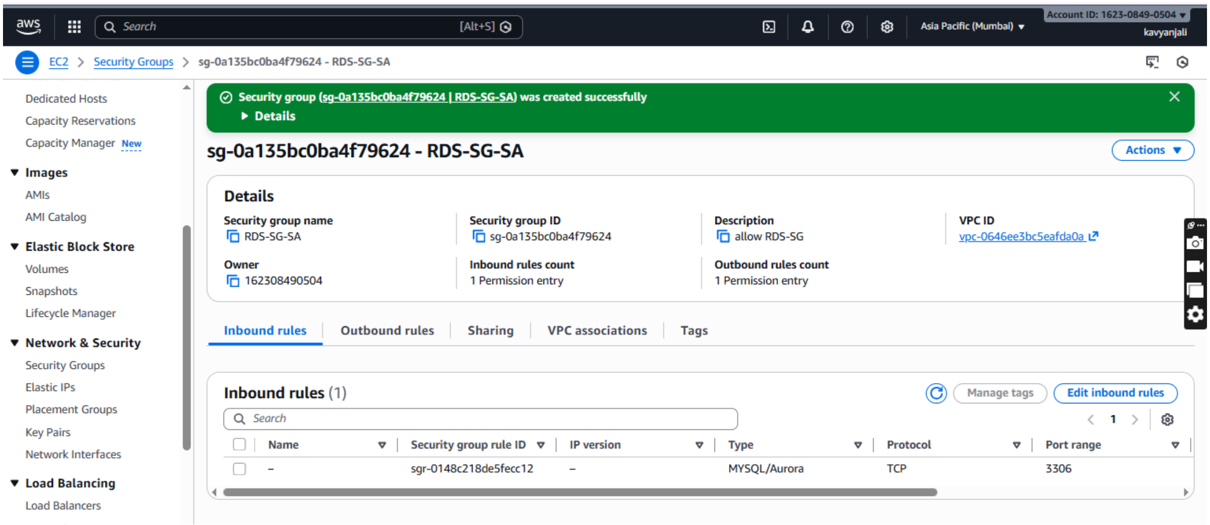
Security group name Web-SG-SA	Security group ID sg-03d7d1f8aec322cb3	Description allow Web-SG	VPC ID vpc-0646ee3bc5eafda0a
Owner 162308490504	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (2)

□	Name	Security group rule ID	IP version	Type	Protocol	Port range
□	–	sgr-022efc04c327d6c29	–	HTTP	TCP	80
□	–	sgr-08fe2e50a5db8e644	IPv4	SSH	TCP	22

3. RDS Security Group

- Name: RDS-SG
- Inbound:
  - MySQL (3306) → Web-SG



Security group (sg-0a135bc0ba4f79624 | RDS-SG-SA) was created successfully

Details

sg-0a135bc0ba4f79624 - RDS-SG-SA

Details

Security group name: RDS-SG-SA

Security group ID: sg-0a135bc0ba4f79624

Description: allow RDS-SG

VPC ID: vpc-0646ee3bc5eafda0a

Owner: 162308490504

Inbound rules count: 1 Permission entry

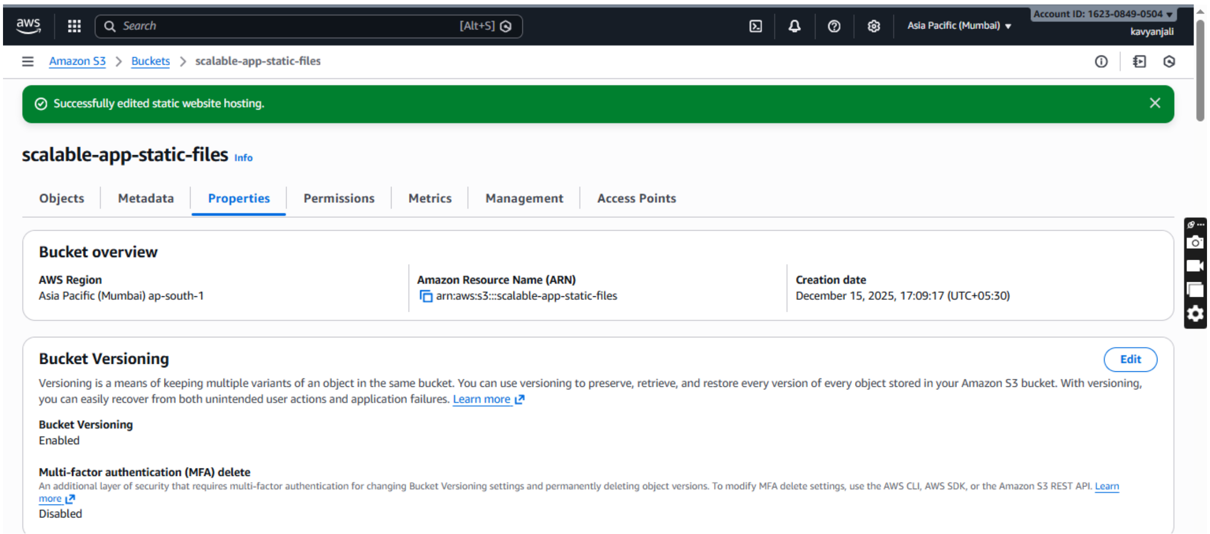
Outbound rules count: 1 Permission entry

Inbound rules (1)

Name	Security group rule ID	IP version	Type	Protocol	Port range
–	sgr-0148c218de5fecc12	–	MYSQL/Aurora	TCP	3306

STEP 6: Create S3 Bucket (Static Files)

1. Go to **S3 → Create bucket**
2. Name: scalable-app-static-files
3. Disable **Block Public Access**
4. Enable **Static Website Hosting**
5. Upload:
   - CSS
   - Images
   - JS files



Successfully edited static website hosting.

scalable-app-static-files

Bucket overview

AWS Region: Asia Pacific (Mumbai) ap-south-1

Amazon Resource Name (ARN): arn:aws:s3:::scalable-app-static-files

Creation date: December 15, 2025, 17:09:17 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Enabled

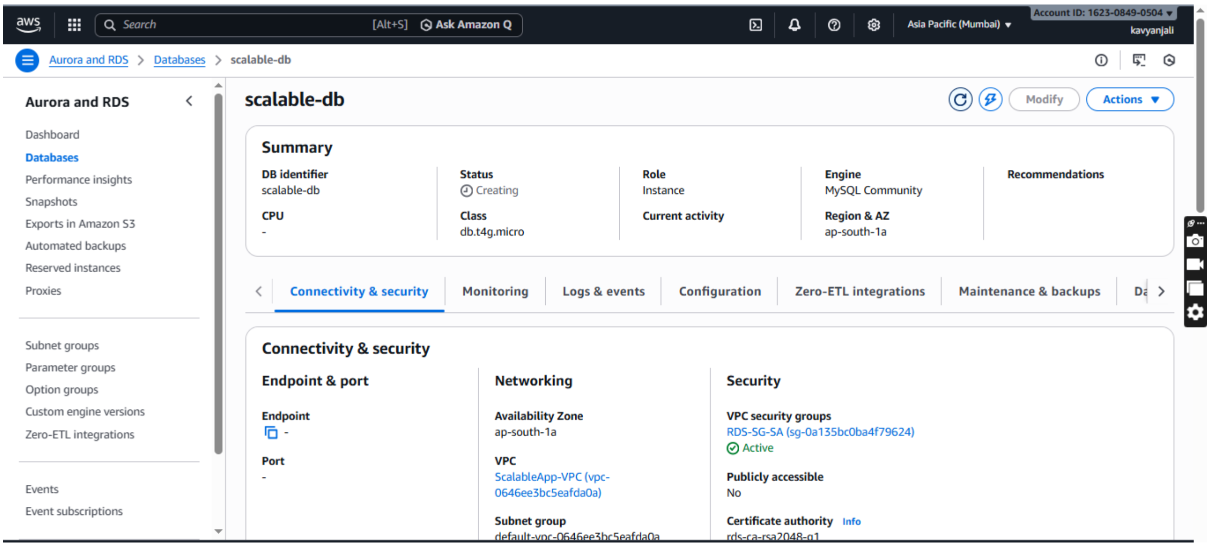
Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

STEP 7: Create RDS Database

1. Go to **RDS → Create database**
2. Choose:

    - Engine: **MySQL**
    - Template: **Free tier**

3. Settings:
    - DB name: scalable-db
    - Username: admin
    - Password: StrongPassword123
4. Connectivity:
    - VPC: ScalableApp-VPC
    - Security Group: RDS-SG
    - Public access: No
5. Click **Create database**



scalable-db

Summary

DB identifier: scalable-db
CPU: -
Status: Creating
Class: db.t4g.micro
Role: Instance
Current activity
Engine: MySQL Community
Region & AZ: ap-south-1a
Recommendations

Connectivity & security

Endpoint & port
Endpoint: -
Port: -

Networking
Availability Zone: ap-south-1a
VPC: ScalableApp-VPC (vpc-0646ee3bc5eafda0a)
Subnet group: default-vpc-0646ee3bc5eafda0a

Security
VPC security groups: RDS-SG-SA (sg-0a135bc0ba4f79624) Active
Publicly accessible: No
Certificate authority Info: rds-ca-rsa2048-g1

STEP 8: Create EC2 Launch Template

1. Go to **EC2 → Launch Templates**
2. Click **Create launch template**
3. Name: WebApp-LT

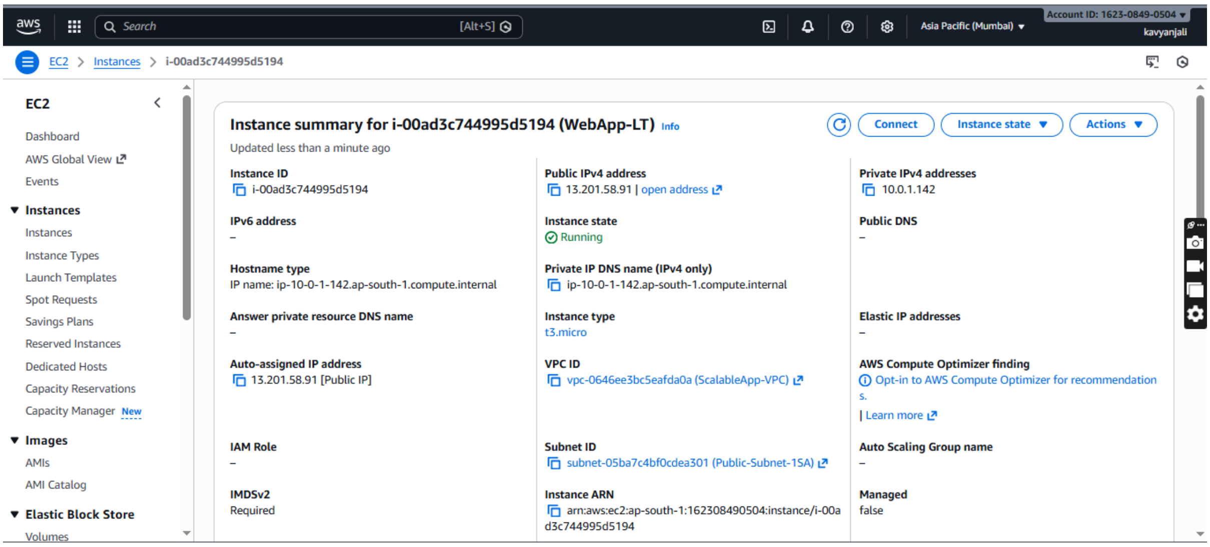
Configuration:

- AMI: Amazon Linux 2
- Instance type: t2.micro
- Key pair: Select your key
- Security Group: Web-SG

User Data Script:

```
#!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "<h1>Scalable Web App Running on EC2</h1>" > /var/www/html/index.html
```

Click **Create launch template**



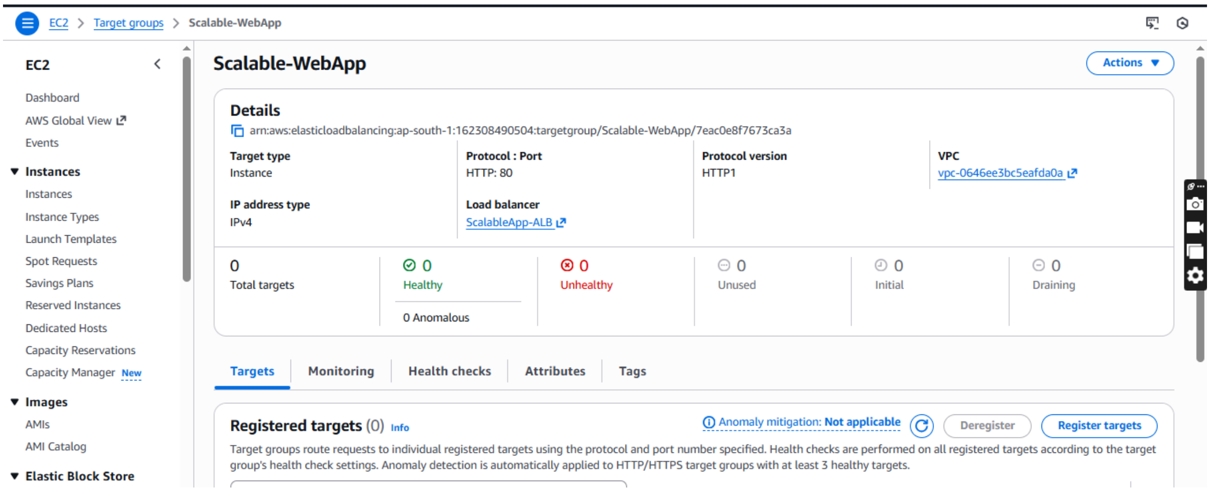
Instance summary for i-00ad3c744995d5194 (WebApp-LT) Info

Updated less than a minute ago

Instance ID i-00ad3c744995d5194	**Public IPv4 address** 13.201.58.91 | open address	**Private IPv4 addresses** 10.0.1.142
IPv6 address –	**Instance state** Running	**Public DNS** –
Hostname type IP name: ip-10-0-1-142.ap-south-1.compute.internal	**Private IP DNS name (IPv4 only)** ip-10-0-1-142.ap-south-1.compute.internal	
Answer private resource DNS name –	**Instance type** t3.micro	**Elastic IP addresses** –
Auto-assigned IP address 13.201.58.91 [Public IP]	**VPC ID** vpc-0646ee3bc5eafda0a (ScalableApp-VPC)	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	**Subnet ID** subnet-05ba7c4bf0cdea301 (Public-Subnet-1SA)	**Auto Scaling Group name** –
IMDSv2 Required	**Instance ARN** arn:aws:ec2:ap-south-1:162308490504:instance/i-00ad3c744995d5194	**Managed** false

STEP 9: Create Target Group

1. Go to **EC2 → Target Groups**
2. Click **Create target group**
3. Choose:
   - Type: Instances
   - Protocol: HTTP
   - Port: 80
   - VPC: ScalableApp-VPC
4. Health check path: /
5. Create target group



Scalable-WebApp

Details

arn:aws:elasticloadbalancing:ap-south-1:162308490504:targetgroup/Scalable-WebApp/7eac0e8f7673ca3a

Target type Instance	**Protocol : Port** HTTP: 80	**Protocol version** HTTP1	**VPC** vpc-0646ee3bc5eafda0a
IP address type IPv4	**Load balancer** ScalableApp-ALB		

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

Registered targets (0) Info

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.

STEP 10: Create Application Load Balancer

1. Go to **EC2 → Load Balancers**
2. Click **Create Load Balancer**
3. Choose **Application Load Balancer**

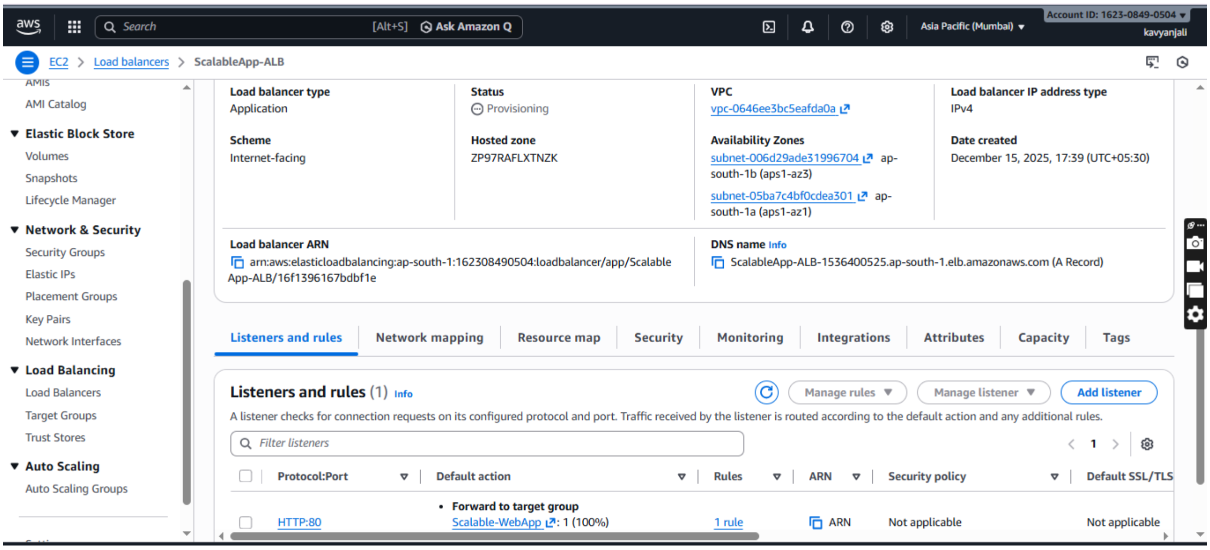
Settings:

- Name: ScalableApp-ALB
- Scheme: Internet-facing
- Subnets: Public Subnet 1 & 2
- Security Group: ALB-SG

Listener:

- HTTP → Forward to Target Group

Click **Create**



Load balancer type
Application

Status
Provisioning

VPC
vpc-0646ee3bc5eafda0a

Load balancer IP address type
IPv4

Scheme
Internet-facing

Hosted zone
ZP97RAFLXTNZK

Availability Zones
subnet-006d29ade31996704 ap-south-1b (aps1-az3)
subnet-05ba7c4bf0cdea301 ap-south-1a (aps1-az1)

Date created
December 15, 2025, 17:39 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:162308490504:loadbalancer/app/ScalableApp-ALB/16f1396167bdbf1e

DNS name Info
ScalableApp-ALB-1536400525.ap-south-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

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** Scalable-WebApp: 1 (100%)	1 rule	ARN	Not applicable	Not applicable

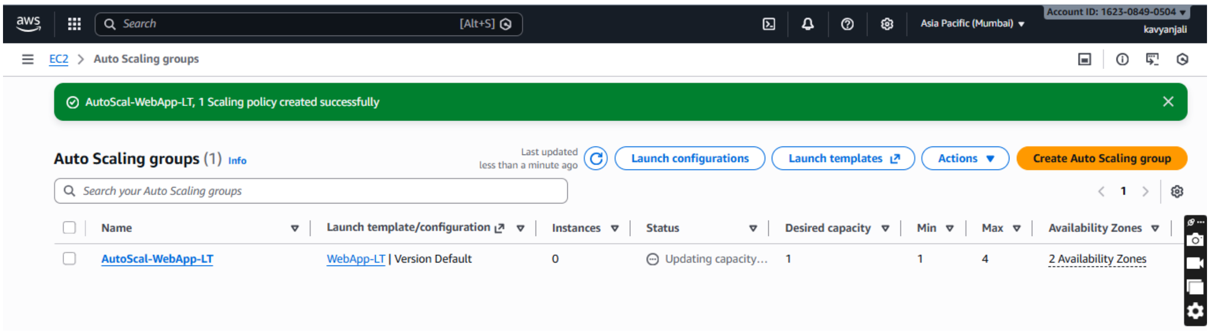
STEP 11: Create Auto Scaling Group

1. Go to **EC2 → Auto Scaling Groups**
2. Click **Create Auto Scaling group**
3. Choose launch template: WebApp-LT
4. VPC: ScalableApp-VPC
5. Subnets: Both public subnets
6. Attach to:
   - Existing Load Balancer
   - Select Target Group

Scaling Policy:

- Min: 1
- Desired: 2
- Max: 4
- Target Tracking:
  - CPU Utilization: 50%

Create Auto Scaling Group


aws
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 1623-0849-0504
kavyanjali
EC2 > Auto Scaling groups
Launch configurations
Launch templates
Actions
Create Auto Scaling group
Search your Auto Scaling groups
1

AutoScal-WebApp-LT, 1 Scaling policy created successfully

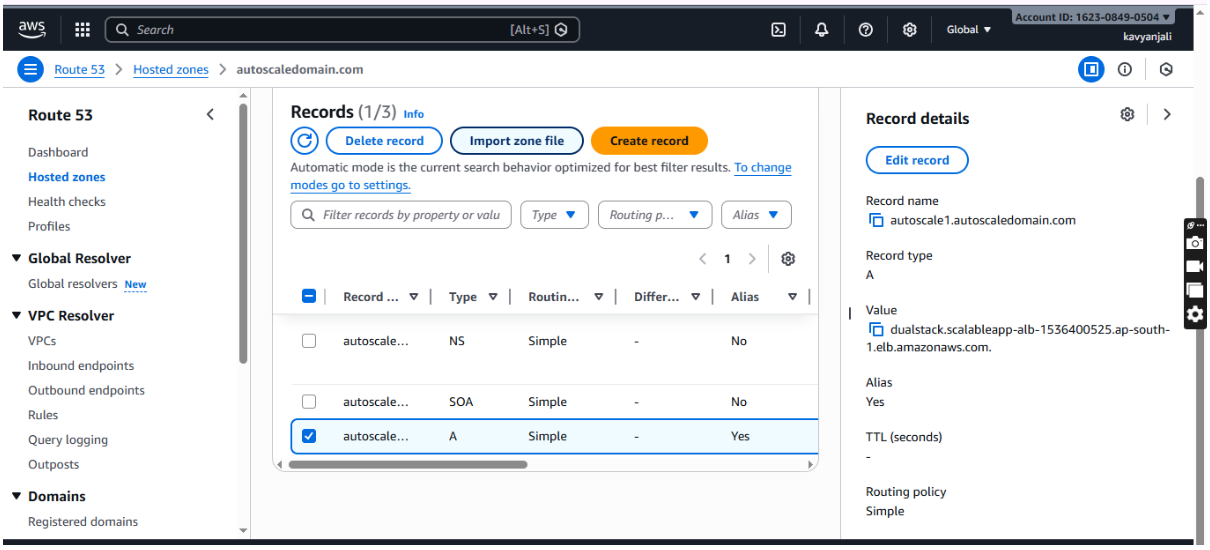
Auto Scaling groups (1) Info

Last updated less than a minute ago

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
□	AutoScal-WebApp-LT	WebApp-LT | Version Default	0	Updating capacity...	1	1	4	2 Availability Zones

STEP 12: Configure Route 53

1. Go to **Route 53 → Hosted Zones**
2. Create hosted zone:
   - Domain: autoscaledomain.com
3. Create record:
   - Type: A (Alias)
   - Alias target: Application Load Balancer
   - Routing: Simple



Records (1/3) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

	Record ...	Type	Routin...	Differ...	Alias
□	autoscale...	NS	Simple	-	No
□	autoscale...	SOA	Simple	-	No
☑	autoscale...	A	Simple	-	Yes

Record details

Record name
autoscale1.autoscaledomain.com

Record type
A

Value
dualstack.scalableapp-alb-1536400525.ap-south-1.elb.amazonaws.com.

Alias
Yes

TTL (seconds)
-

Routing policy
Simple

STEP 13: Testing & Verification

□ 13.1 Verify Auto Scaling Group Instances

1. Go to **EC2 Console**
2. Click **Auto Scaling Groups**
3. Select your ASG (e.g., WebApp-ASG)
4. Open **Instance management**
5. Confirm:
   - Desired number of instances are **Running**
   - Instances are **InService**

□ 13.2 Verify Target Group Health

1. Go to **EC2 → Target Groups**
2. Select your target group
3. Open **Targets** tab
4. Confirm all instances show **Healthy**

□ 13.3 Verify Load Balancer

1. Go to **EC2 → Load Balancers**
2. Select your **Application Load Balancer**
3. Copy **DNS name: https://scalableapp-alb-1536400525.ap-south-1.elb.amazonaws.com/**
4. Paste DNS name in a web browser
5. Confirm web page loads successfully

□ 13.4 Verify Auto Scaling (Optional Test)

1. Go to **EC2 → Auto Scaling Groups**
2. Select your ASG
3. Click **Edit**
4. Increase **Desired capacity** (e.g., from 2 → 3)
5. Save
6. Check **Instance management**
7. Confirm a **new EC2 instance launches**

□ 13.5 Verify S3 Static Content (If Used)

1. Go to **S3**
2. Open your bucket
3. Click **Properties**
4. Copy **Static website endpoint**
5. Open endpoint in browser
6. Confirm static files load

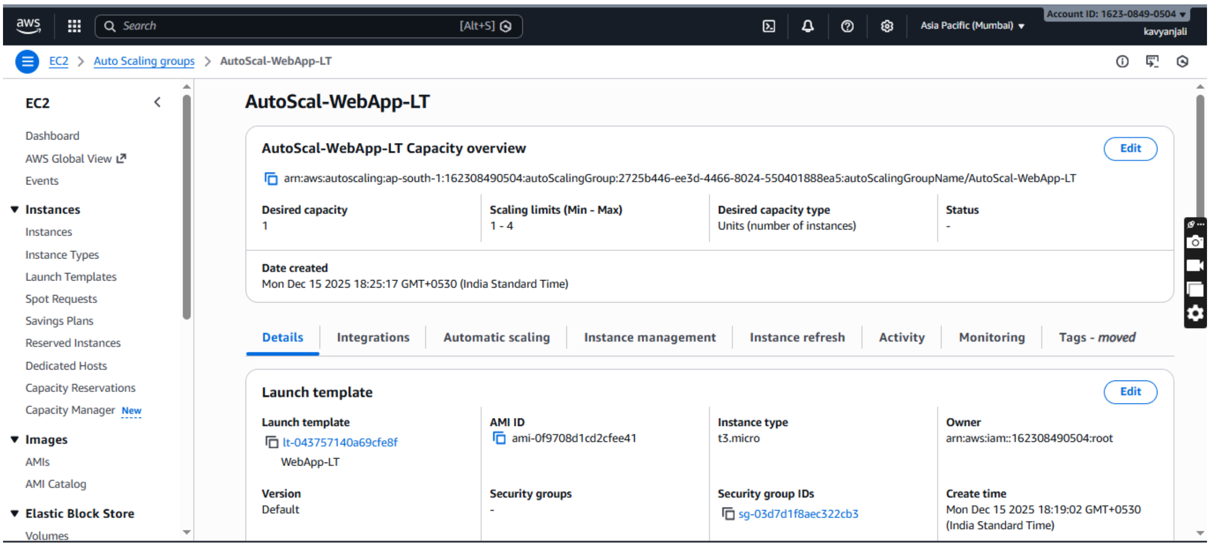
□ 13.6 Verify RDS Connectivity

1. Go to **RDS**
2. Select your database
3. Confirm **Status = Available**
4. Note **Endpoint** for application use

STEP 13 COMPLETE WHEN:

- ALB DNS opens website
- Target group shows **Healthy And Unhealthy**

- ASG launches/terminates instances correctly
- RDS is available
- S3 static content accessible



AutoScal-WebApp-LT

AutoScal-WebApp-LT Capacity overview

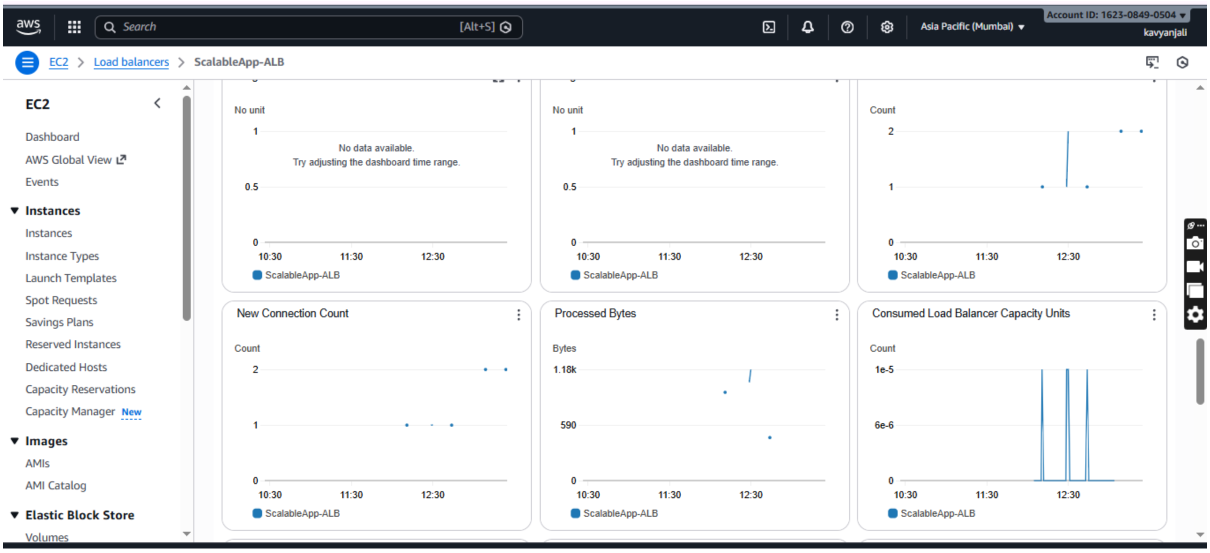
arn:aws:autoscaling:ap-south-1:162308490504:autoScalingGroup:2725b446-ee3d-4466-8024-550401888ea5:autoScalingGroupName/AutoScal-WebApp-LT

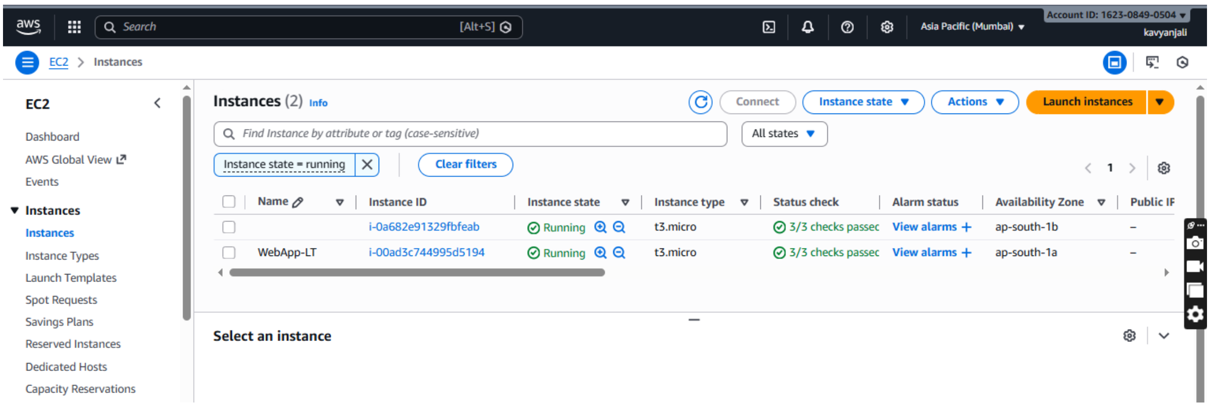
Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 4	Units (number of instances)	-

Date created
Mon Dec 15 2025 18:25:17 GMT+0530 (India Standard Time)

Launch template

Launch template	AMI ID	Instance type	Owner
lt-043757140a69cfe8f WebApp-LT	ami-0f9708d1cd2cfee41	t3.micro	arn:aws:iam::162308490504:root
Version	**Security groups**	**Security group IDs**	**Create time**
Default	-	sg-03d7d1f8aec322cb3	Mon Dec 15 2025 18:19:02 GMT+0530 (India Standard Time)

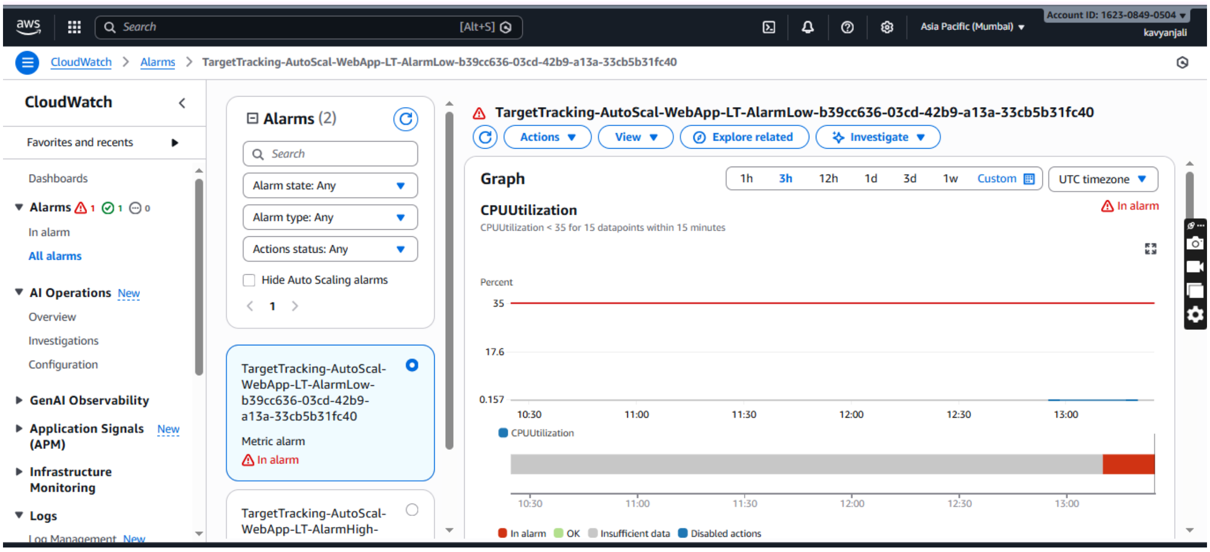




Instances (2)

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
□		i-0a682e91329fbfeab	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1b	–
□	WebApp-LT	i-00ad3c744995d5194	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1a	–

Select an instance



CloudWatch > Alarms > TargetTracking-AutoScal-WebApp-LT-AlarmLow-b39cc636-03cd-42b9-a13a-33cb5b31fc40

TargetTracking-AutoScal-WebApp-LT-AlarmLow-b39cc636-03cd-42b9-a13a-33cb5b31fc40

CPUUtilization

CPUUtilization < 35 for 15 datapoints within 15 minutes

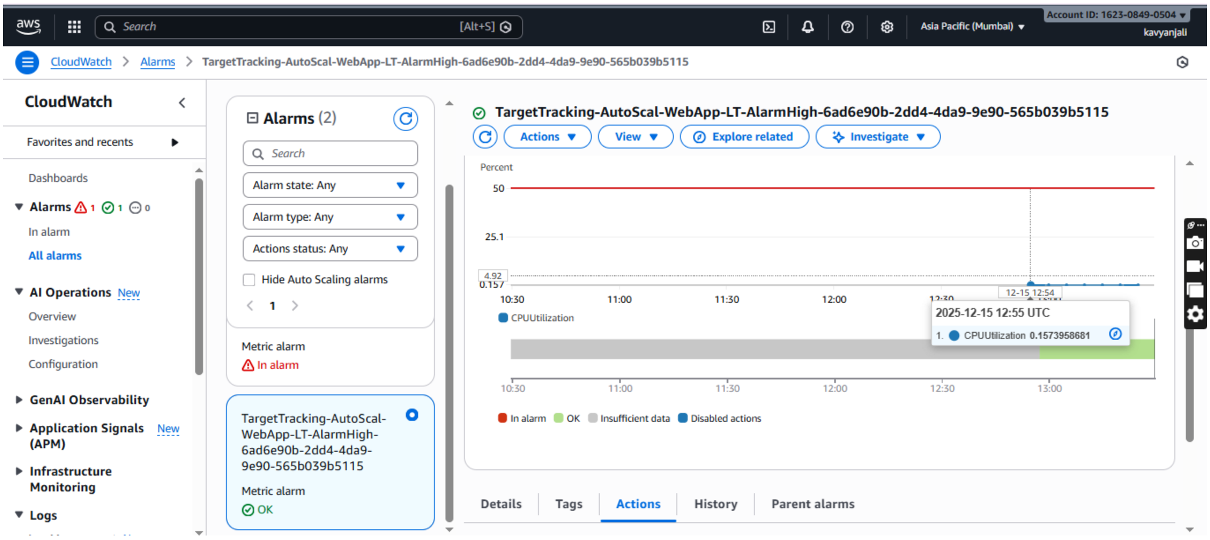
In alarm

Alarm state: Any

Alarm type: Any

Actions status: Any

TargetTracking-AutoScal-WebApp-LT-AlarmLow-b39cc636-03cd-42b9-a13a-33cb5b31fc40 — Metric alarm — In alarm



CloudWatch > Alarms > TargetTracking-AutoScal-WebApp-LT-AlarmHigh-6ad6e90b-2dd4-4da9-9e90-565b039b5115

TargetTracking-AutoScal-WebApp-LT-AlarmHigh-6ad6e90b-2dd4-4da9-9e90-565b039b5115

2025-12-15 12:55 UTC

1. CPUUtilization 0.1573958681

TargetTracking-AutoScal-WebApp-LT-AlarmHigh-6ad6e90b-2dd4-4da9-9e90-565b039b5115 — Metric alarm — OK

Summary

- A Virtual Private Cloud (VPC) was created to isolate and secure the application network.
- Public subnets were configured across multiple Availability Zones to ensure high availability.
- An Internet Gateway and route table enabled internet access.
- Security Groups were configured to control traffic between the Load Balancer, EC2 instances, and RDS database.
- An S3 bucket was created to host static website assets.
- An RDS MySQL database was deployed within the VPC for persistent data storage.
- An EC2 Launch Template with a user data script was used to automatically configure web servers.
- An Application Load Balancer was configured to distribute traffic across EC2 instances.
- An Auto Scaling Group was set up to automatically scale EC2 instances based on CPU utilization.
- Route 53 was configured to route domain traffic to the Load Balancer.

Conclusion

This setup successfully deploys a **scalable, highly available, and fault-tolerant web application** using AWS managed services. The architecture ensures automatic traffic distribution, dynamic scaling based on demand, secure database access, and reliable static content delivery. This solution improves performance, availability, and operational efficiency while maintaining cost optimization through Auto Scaling.