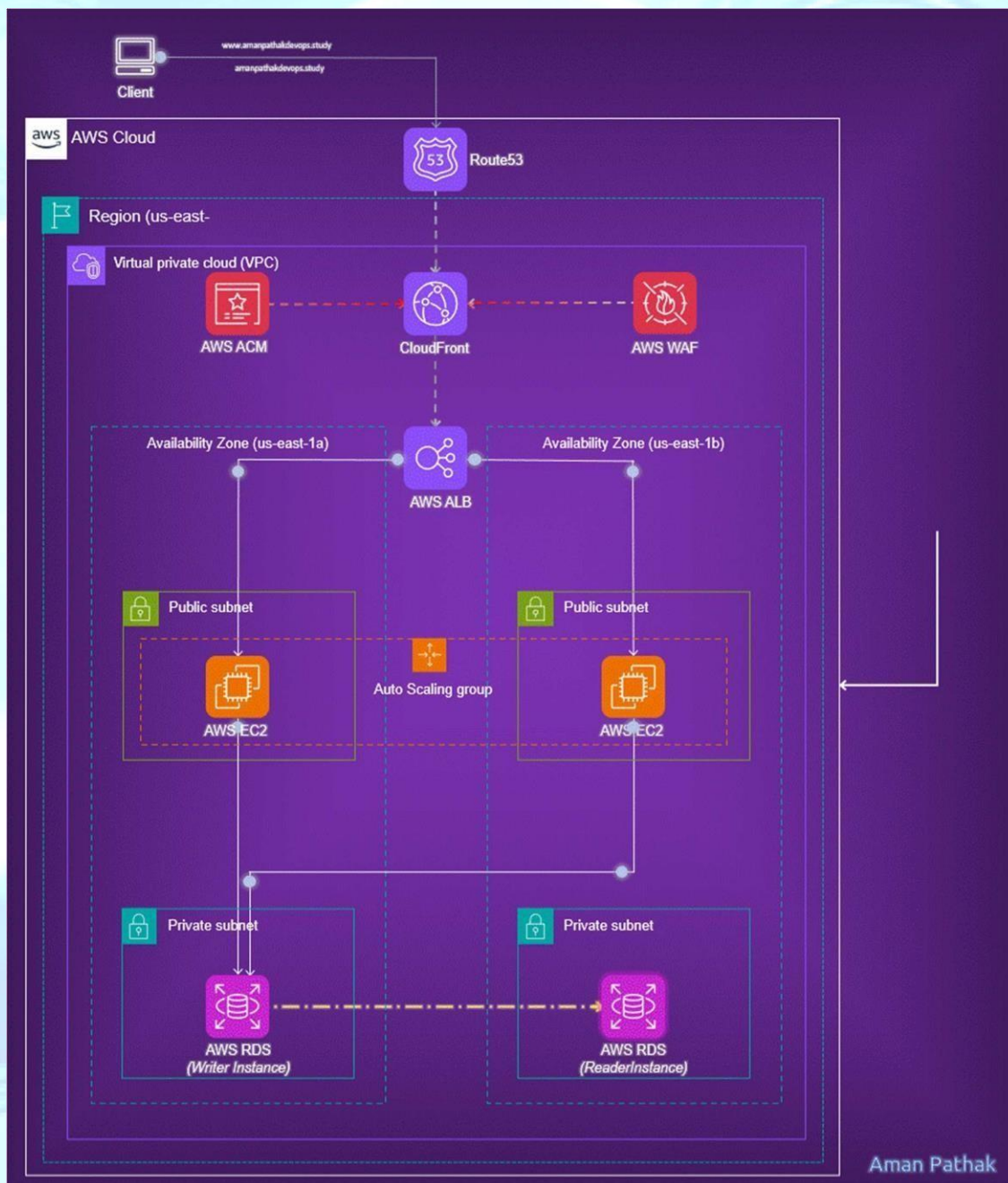


NAME: S. khalindar , **BATCH NO:** 140

Implementation of 2-Tier Architecture in AWS Using Route 53, EC2, Load Balancer, Auto Scaling, CloudFront, and WAF



AWS Architecture Overview:

This architecture represents a **highly available, scalable, and secure web application deployment** on AWS. It leverages multiple AWS services across two Availability Zones to ensure fault tolerance, performance optimization, and robust security.

Key Strengths of this Architecture:

- ❑ **High Availability**: Two Availability Zones (us-east-1a and us-east-1b) ensure redundancy. If one zone goes down, the other keeps your app running.
- ❑ **Scalability**: Auto Scaling Groups dynamically adjust the number of EC2 instances based on traffic, keeping performance smooth and cost optimized.
- ❑ **Security Layers**:
 - ✚ AWS WAF protects against common web exploits.
 - ✚ AWS ACM handles SSL/TLS certificates for secure communication.
 - ✚ Private subnets isolate sensitive components like databases from public access.
- ❑ **Global Performance**: CloudFront CDN caches content closer to users, reducing latency.
- ❑ **Traffic Management**: Route 53 handles DNS routing, while the ALB smartly distributes incoming traffic across healthy EC2 instances.

Uses cases for business perspective:

- ❑ Hosting highly available and scalable web applications or websites.
- ❑ Deploying multi-tier applications where separation of presentation, application, and data layers is crucial for security and manageability.
- ❑ Applications requiring low-latency content delivery and protection against web exploits.
- ❑ Businesses needing a flexible and cost-effective infrastructure that scales with demand.

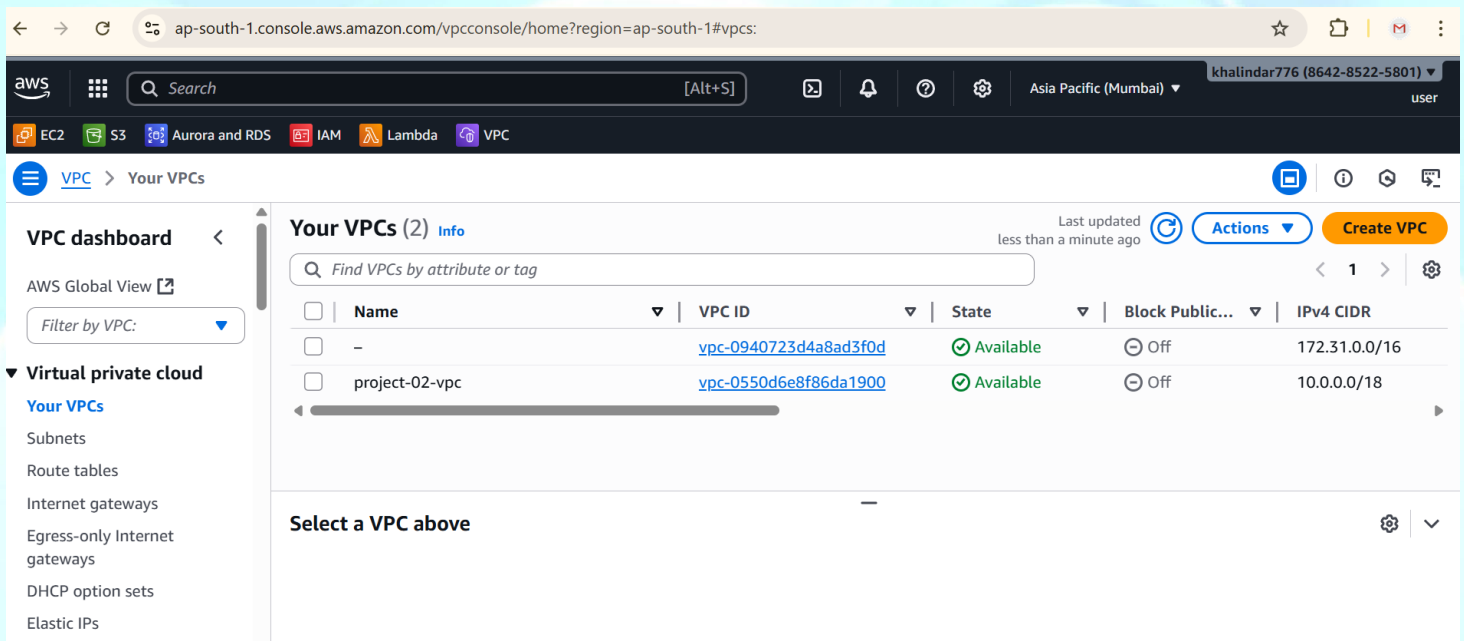
Steps to follow:

- ✚ VPC (Subnets, Route Tables, and IGW)
- ✚ EC2 (Security Group, AMI, Template, Target Groups, Load Balancer, and Auto Scaling Group) ✚ Route 53 (Hosted Zone, and health checks)
- ✚ Go Daddy (DNS)
- ✚ ACM (Public TLS/SSL Certificate)
- ✚ Cloud Front
- ✚ WAF (For security)

PROCESS :

Step 1: Create VPC (Virtual Private Cloud).

- Create VPC
- VPC name is **project-02-vpc**
- IPv4 address 120.0.0.0/16
- VPC created



The screenshot shows the AWS Management Console for the 'ap-south-1' region. The 'VPC' service is selected in the left-hand navigation pane. The main content area displays 'Your VPCs (2)' with a table listing the VPCs. The table has columns for Name, VPC ID, State, Block Public Access, and IPv4 CIDR. Two VPCs are listed: one with ID 'vpc-0940723d4a8ad3f0d' and another named 'project-02-vpc' with ID 'vpc-0550d6e8f86da1900'. Both are in an 'Available' state. Below the table, there is a section titled 'Select a VPC above'.

Name	VPC ID	State	Block Public...	IPv4 CIDR
-	vpc-0940723d4a8ad3f0d	Available	Off	172.31.0.0/16
project-02-vpc	vpc-0550d6e8f86da1900	Available	Off	10.0.0.0/18

Step 2: Create Subnets (4).

- Create 2 public subnets & 2 private subnets
- Given names for 4 Subnets (**pub-1-route53, pub-2-route53, private-1-route53, and private2route53**).
- Select the given VPC id : project-vpc
- Select Availability zones for all subnets
- Customize the IP address with the proper ranges for all subnets □ Create the Subnet

aws [Search] [Alt+S] Asia Pacific (Mumbai) khalindar776 (8642-8522-5801) user

EC2 S3 Aurora and RDS IAM Lambda VPC

VPC > Subnets

VPC dashboard < AWS Global View [?] Filter by VPC: [v]

▼ **Virtual private cloud**

- Your VPCs
- Subnets**
- Route tables
- Internet gateways
- Egress-only Internet gateways
- DHCP option sets
- Elastic IPs

Subnets (7) Info Last updated 1 minute ago [Actions] [Create subnet]

Find subnets by attribute or tag

<input type="checkbox"/>	Name	Subnet ID	State	VPC
<input type="checkbox"/>	private-1-route53	subnet-07c27b3c3929da582	Available	vpc-0550d6e8f86da1900 proj...
<input type="checkbox"/>	public-2-route53	subnet-08fefa02733cf62f0	Available	vpc-0550d6e8f86da1900 proj...
<input type="checkbox"/>	private-2-route53	subnet-081111a5aed038116	Available	vpc-0550d6e8f86da1900 proj...
<input type="checkbox"/>	public-1-route53	subnet-0bf3ad826ebc1f619	Available	vpc-0550d6e8f86da1900 proj...

Select a subnet

Step 3: Create an Internet Gateway (IGW).

- Create Internet Gateway for public subnets access the internet connection □ Name : **project-02-ijw** □ Create IG
- In Actions select and Attach VPC (in Notifications)
- Attach given VPC (Project-02-vpc)

aws [Search] [Alt+S] Asia Pacific (Mumbai) khalindar776 (8642-8522-5801) user

EC2 S3 Aurora and RDS IAM Lambda VPC

VPC > Internet gateways

VPC dashboard < AWS Global View [?] Filter by VPC: [v]

▼ **Virtual private cloud**

- Your VPCs
- Subnets
- Route tables
- Internet gateways**
- Egress-only Internet gateways
- DHCP option sets
- Elastic IPs

Internet gateways (2) Info [Actions] [Create internet gateway]

Find internet gateways by attribute or tag

<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID
<input type="checkbox"/>	project-02-igw	igw-01134f2acaf4316cd	Attached	vpc-0550d6e8f86da1900 proj...
<input type="checkbox"/>	-	igw-08ab6e3c6e2e911b3	Attached	vpc-0940723d4a8ad3f0d

Select an internet gateway above

Step 4: Create Route tables (2)

- ❑ Create 1 public Route table & 1 private Route table
- ❑ Name : **project02-pub-rt**
- ❑ Select the existing VPC (project02-vpc)
- ❑ Created Root table

The screenshot shows the AWS Management Console interface for Route tables. The top navigation bar includes the AWS logo, a search bar, and various service icons (EC2, S3, Aurora and RDS, IAM, Lambda, VPC). The user is logged in as 'khalindar776 (8642-8522-5801)' in the 'Asia Pacific (Mumbai)' region.

The main content area is titled 'Route tables (4)' and shows a list of route tables. The table has columns for Name, Route table ID, Explicit subnet associations, Edge associations, and Main. The 'project02-pub-rt' is highlighted, showing it is associated with 2 subnets.

Name	Route table ID	Explicit subnet associations	Edge associations	Main
-	rtb-0607a4a028719684f	-	-	Yes
project-2-private-rt	rtb-0d4f0e5edb1480fb3	2 subnets	-	No
project02-pub-rt	rtb-0742c9a7333349de6	2 subnets	-	No
-	rtb-0d21c8f252bd64eac	-	-	Yes

The left sidebar shows the 'VPC dashboard' with a 'Virtual private cloud' section. The 'Route tables' link is selected, and the 'project02-pub-rt' is highlighted in the list.

Step5: Create Security Group

- ❑ It maintains inbound and outbound traffic rules
- ❑ Name : **project02-sg**
- ❑ Add inbound rules for SSH (22), HTTP (80), and HTTPS (443) to allow secure access and web traffic. These rules permit incoming connections from all IP addresses, enabling public access to the EC2 instances.
- ❑ Create SG

Security Groups (7) Info

Find security groups by attribute or tag

Name	Security group ID	Security group name	VPC ID
-	sg-0526291afa61360aa	project2-sg	vpc-0940723d4a8ad3f0d
-	sg-0e169b522411fb4ef	default	vpc-0940723d4a8ad3f0d
-	sg-0d22a57b383a15d7c	launch-wizard-2	vpc-0940723d4a8ad3f0d
-	sg-03f093aa9533d0947	launch-wizard-4	vpc-0940723d4a8ad3f0d

Select a security group

Step 6: Create and launch EC2 instances (2).

- Create two EC2 instances and names (**server1**) & (**server2**).
- Select the existing VPC (project02-vpc).
- Select the Security Group (sgrouete53) for 2 instances □ Launch instance with ubuntu OS, Instance type (t2micro) □ Select the keypair which we created (Mumbai.pem).
- Configure the Network settings (project02-vpc, Subnets (1a, 1b), SG, And Public IP enables). □ Click on launch Instance.

Instances (4) Info

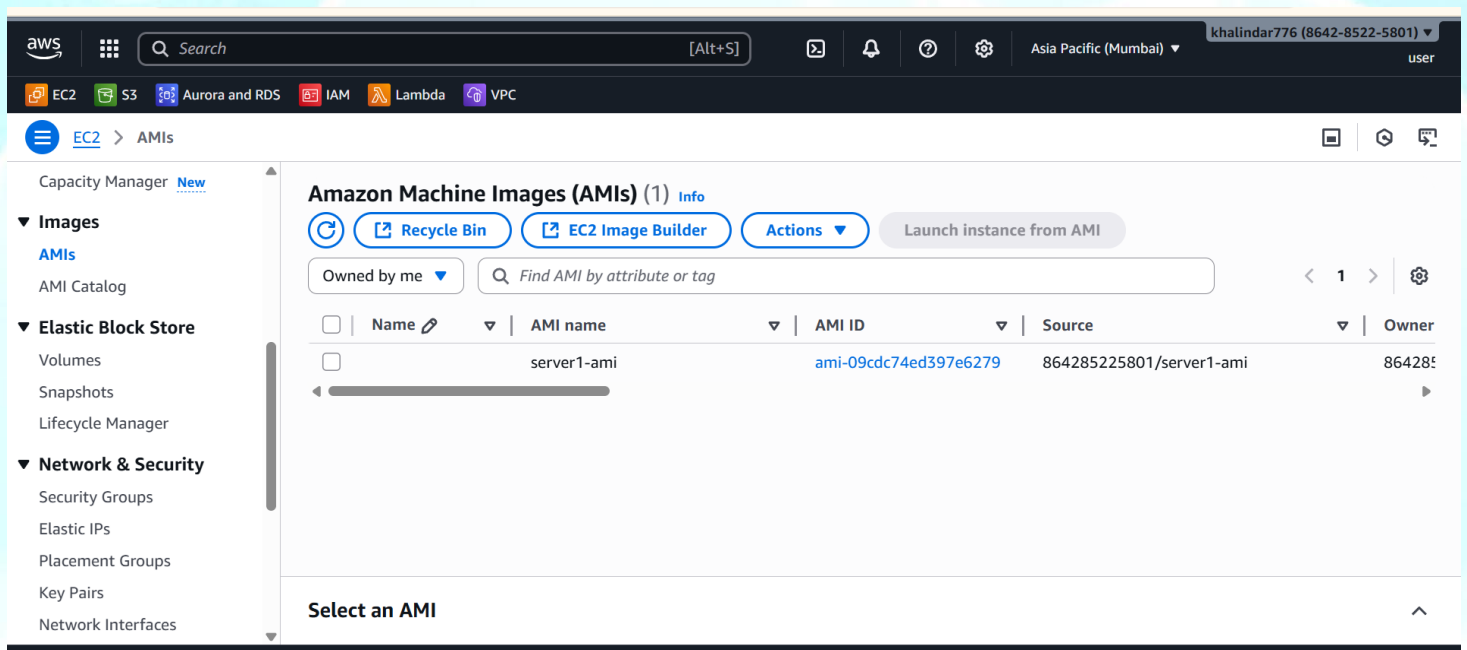
Find Instance by attribute or tag (case-sensitive)

Instance state = running Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	webserver2	i-0f1513a13de313bea	Running	t3.micro	3/3 checks passed	View alarms
<input type="checkbox"/>	new1	i-0dd159e0233dfd05f	Running	t3.micro	3/3 checks passed	View alarms
<input type="checkbox"/>	new2	i-0e036c485a1ee09e7	Running	t3.micro	3/3 checks passed	View alarms
<input type="checkbox"/>	webserver01	i-0322d7186b2544dc2	Running	t3.micro	3/3 checks passed	View alarms

Step 7: Creating AMI's (image)with Public Server

- We need to setup auto scaling on that create AMI and launch template.
- First create a AMI and name is **server1-ami**
- To create an AMI first select the instance and in actions click on image and template option then click on image. □ Provide the details for AMI
- Create AMI.



Step 8 : Creating Launch Template

- Click on “Launch Templates” in the left sidebar, then “Create launch template”. □ Configure Template Details.
- Name and description (**project02-template**)
- AMI ID (select existing AMI)
- Instance type (t2.micro)
- Key pair (select existing key pair)
- Security groups (select existing SG)
- Select the security group and enable the auto assign public-ip in advanced network configuration.
- Launch Template.

Launch Templates (1) Info

Search

<input type="checkbox"/>	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time
<input type="checkbox"/>	lt-07427ad01e4a04b14	project-2-template	1	1	2025-10-18T11:00:00Z

Select a launch template

Step 9: Creating Target Group (TG)

- Before creating a load balancer we need to create target group.
- Click on create target group and name is **project02-TG**
- Select the existing VPC (project02-vpc)
- Register the Targets (server1& server2) with include as pending
- Create TG

Target groups (1) Info

Filter target groups

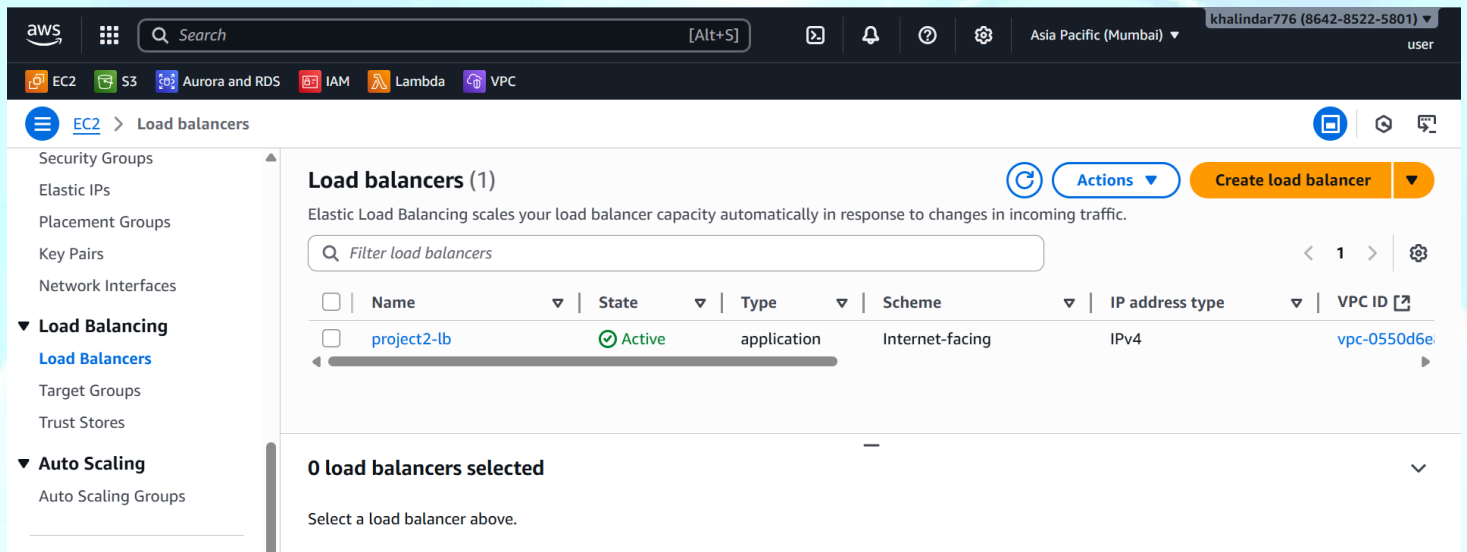
<input type="checkbox"/>	Name	ARN	Port	Protocol	Target type	Lo
<input type="checkbox"/>	project-2-tg	arn:aws:elasticloadbalancing:ap-south-1:123456789012:targetgroup/project-2-tg/123456789012	80	HTTP	Instance	pr

0 target groups selected

Select a target group above.

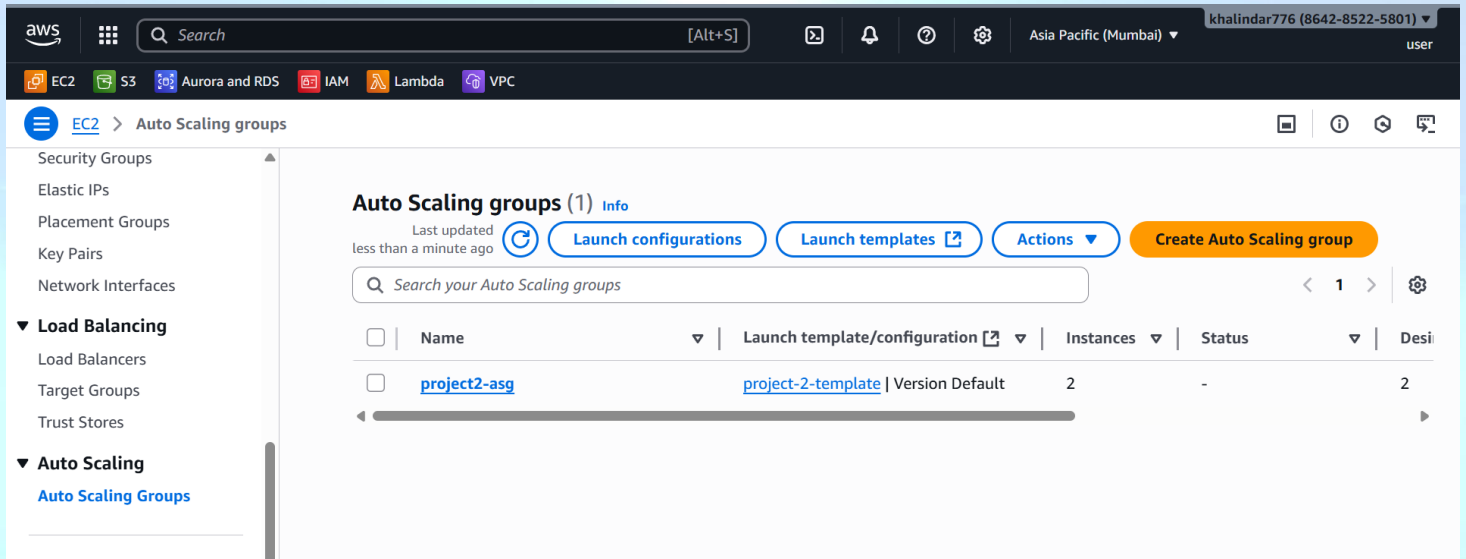
Step 10: Creating Load Balancer(LB)

- Now Go to load balancer page and click on create. Select Application Load Balancer.
- Then provide the LB name is **project2-lb**
- Select the VPC and select the public Availability Zones(2) □ Select the existing target group (TGroute53) in default action.
- Create LB



Step 11 : Creating Auto Scaling

- As per the requirement we need to create auto scaling group □ Create an Auto scaling Group and name is **project2-ASG**.
- select the existing template (template-route53) and image (image-Route53).
- And select the existing VPC (vpc-Route53) and availability zones (Pub-1-route53, Pub-2-route53).
- Then select on attach to an existing load balancer(LB-route53) and target group (TG-route53)
- Now provide the details for **desired capacity (2)** in group size.And provide details for **min desired (2)** and **max desired capacity (4)** in scaling section.
- Select the target tracking policies. In Desired policies and **target value (ex:60%)**, and **instance warmup (ex:100 secs)**.
- Create Auto scaling Group.
- Once created the ASG and then automatically created the desired capacity servers (2).

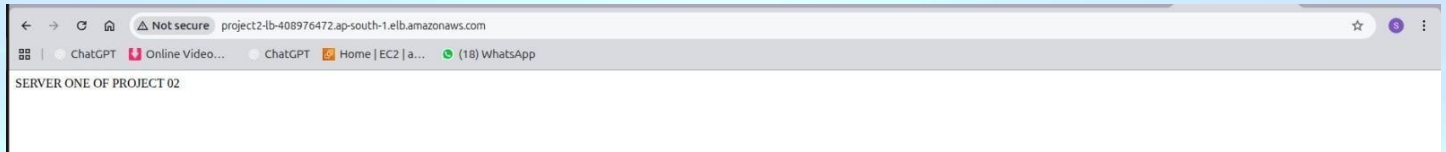


Step 12 : Connect Server-1 to Terminal with SSH String (Using below commands)

- ❑ Go to server-1 and connect the terminal with SSH string
- ❑ Change the ubuntu to root user : **sudo -i**
- ❑ Update the server : **sudo apt update -y**
- ❑ Install the apache2 software in server : **sudo apt install nginx**
- ❑ Then check the index.html file : **cd /usr/share/nginx/html**
- ❑ Remove the existing index.html file : **rm index.html** ❑ Create the file : **vi index.html**
Insert the data : Ex- **SERVER ONE OF PROJECT 02**
- ❑ Restart the server : **systemctl restart nginx**
- ❑ Check the output with Public ip/LB-DNS name : **54.209155.54:80 /LB-DNS name in web browser**
- ❑ Even you can check in terminal with private ip : **curl 120.0.0.178:80**

Output:1

- ❑ The output for server-1 below, but it's not secure.



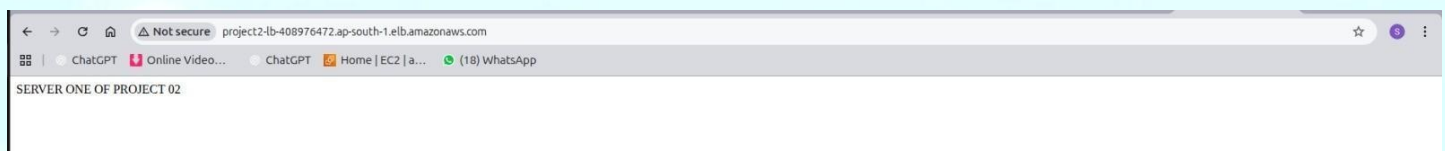
Step 13 : Connect Server-2 to Terminal with SSH String (Using below commands)

- ❑ Go to server-2 and connect the terminal with SSH string
- ❑ Change the ubuntu to root user : **sudo -i**
- ❑ Update the server : **sudo apt update -y**
- ❑ Install the apache2 software in server : **sudo apt install nginx**
- ❑ Then check the index.html file : **cd /usr/share/nginx/html**
- ❑ Remove the existing index.html file : **rm index.html**
- ❑ Create the file : **vi index.html**
- ❑ Insert the data : Ex- **SERVER ONE OF PROJECT 02**
- ❑ Restart the server : **systemctl restart nginx**
- ❑ Check the output with Public ip/LB-DNS name : **3.85.243.100:80 /LB-DNS name in web browser**
- ❑ Even you can check in terminal with private ip : **curl 120.0.10.247:80**

```
ubuntu@ip-120-0-0-6: /usr/share/nginx/html
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-120-0-0-6:~$ sudo systemctl start nginx
ubuntu@ip-120-0-0-6:~$ cd /usr/share/nginx/html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ ls
index.html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ sudo rm index.html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ ls
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ sudo vim /usr/share/nginx/html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ sudo vim /usr/share/nginx/html/index.html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ ls
index.html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ sudo systemctl start nginx
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ ls -l /usr/share/nginx/html/
total 4
-rw-r--r-- 1 root root 25 Oct 10 09:20 index.html
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ cat /etc/nginx/sites-available/default | grep root
root /var/www/html;
# deny access to .htaccess files, if Apache's document root
root /var/www/example.com;
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ sudo mv /usr/share/nginx/html/index.html /var/www/html/
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ cat /etc/nginx/sites-available/default | grep root
root /var/www/html;
# deny access to .htaccess files, if Apache's document root
root /var/www/example.com;
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$ ls -l /usr/share/nginx/html/
cat /etc/nginx/sites-available/default | grep root
sudo systemctl status nginx | grep Active
total 0
root /var/www/html;
# deny access to .htaccess files, if Apache's document root
root /var/www/example.com;
Active: active (running) since Fri 2025-10-10 09:16:05 UTC; 12min ago
ubuntu@ip-120-0-0-6:/usr/share/nginx/html$
```

Output: 2

- The output for server-2 below, but it's not secure.



Note : For Secure the servers , we need to add the HTTPS (443) in LB (Add Listener)

Step 14: Configure DNS with Route-53

- Create the Hosted Zone in Route 53 for the registered domain.
- Domain name is **mnop.shop**.

Route 53

Dashboard

Hosted zones

Health checks

Profiles [New](#)

▼ **IP-based routing**

CIDR collections

▼ **Traffic flow**

Traffic policies

Policy records

▼ **Domains**

Registered domains

Hosted zones (1)

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

Filter records by property or value

Hosted zone name	Type	Created by	Record co...	Description	Hosted ...
mnop.shop	Public	Route 53	4	-	Z0951557...

Hosted zone :

- While we create the Hosted zone automatically generate the default records (i.e NS and SOA)
- Name servers and SOA created under the DNS name mnop.shop

Route 53

Dashboard

Hosted zones

Health checks

Profiles [New](#)

▼ **IP-based routing**

CIDR collections

▼ **Traffic flow**

Traffic policies

Policy records

▼ **Domains**

Registered domains

Records (4) [Info](#)

Automatic mode is the current search behavior optimized for best filter results. [To change modes go to settings.](#)

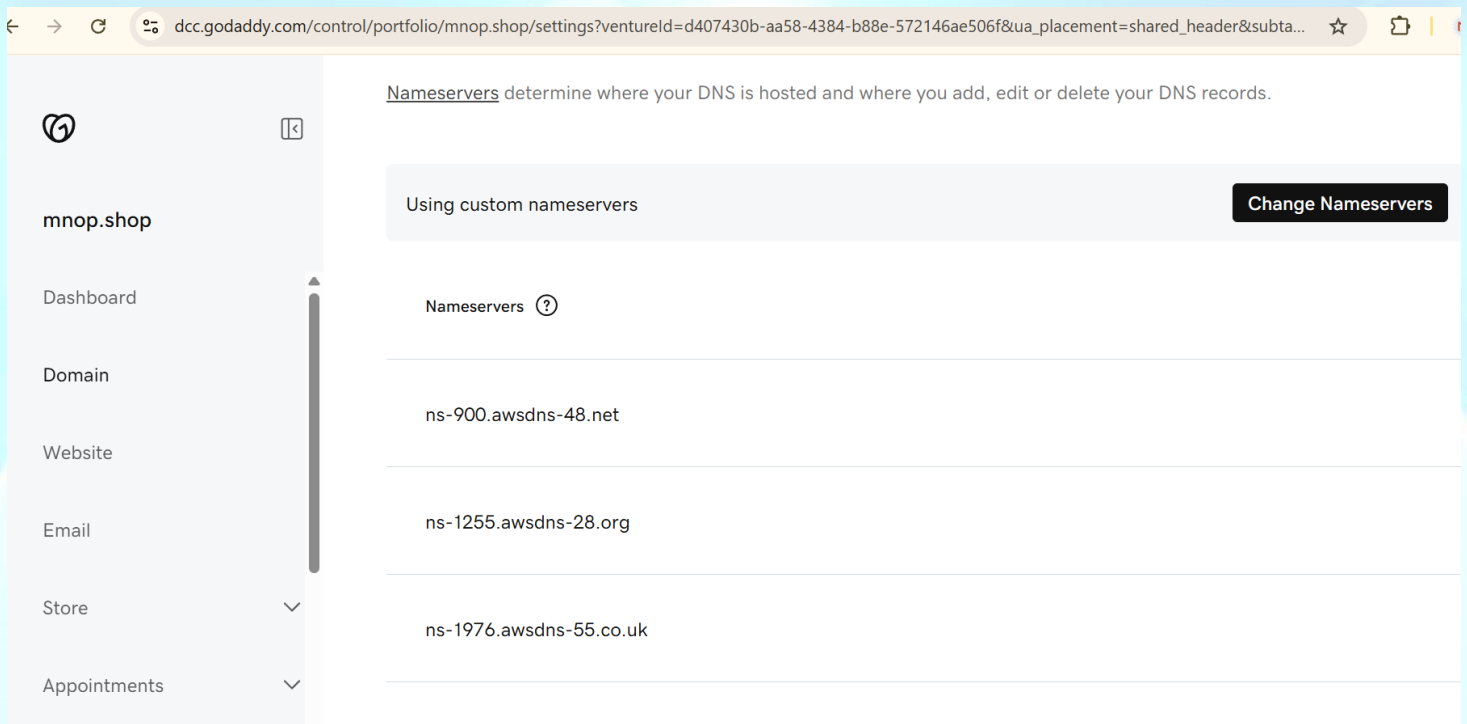
Filter records by property or value

Type Routing p... Alias

Record ...	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...
<input type="checkbox"/> mnop.shop	A	Simple	-	Yes	dro8o6kpf7cz3.cloudfront.net.	-
<input type="checkbox"/> mnop.shop	NS	Simple	-	No	ns-596.awsdns-10.net. ns-1977.awsdns-55.co.uk. ns-314.awsdns-39.com. ns-1112.awsdns-11.org.	172800
<input type="checkbox"/> mnop.shop	SOA	Simple	-	No	ns-596.awsdns-10.net. awsd...	900
<input type="checkbox"/> _25e3cc2...	CNAME	Simple	-	No	_a980fac1a103836ff208b7c...	300

Go Daddy : Registered a custom domain name using a third-party (e.g., GoDaddy)

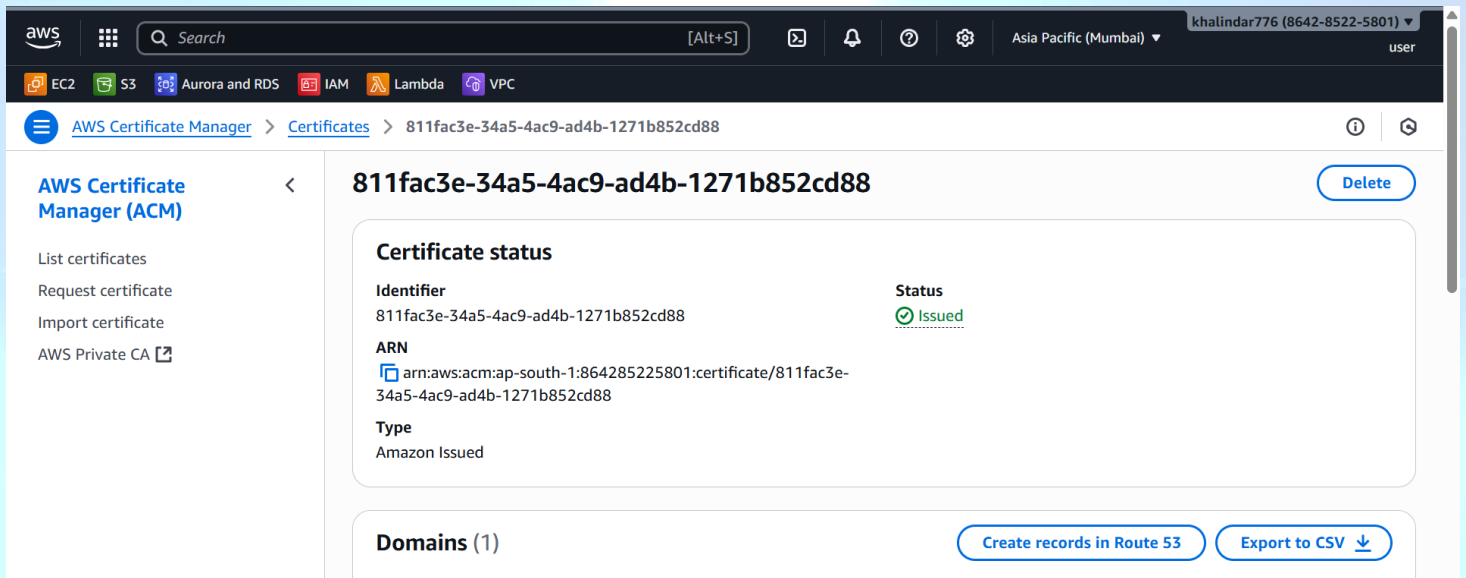
- Go to Go Daddy and login by using the existing credentials.
- Then check with domain and Name servers page with default servers are available.
- whatever the name servers generated in Hosted zone .Add/modify those server details to Name servers in DNS(Go Daddy-reference below attachment)




Note:As of now created the Hosted zone and Records. But when we created the Amazon Certificate Manager(ACM) the same time created the CNAME in Hosted zone (reference below attachment-Step 15)

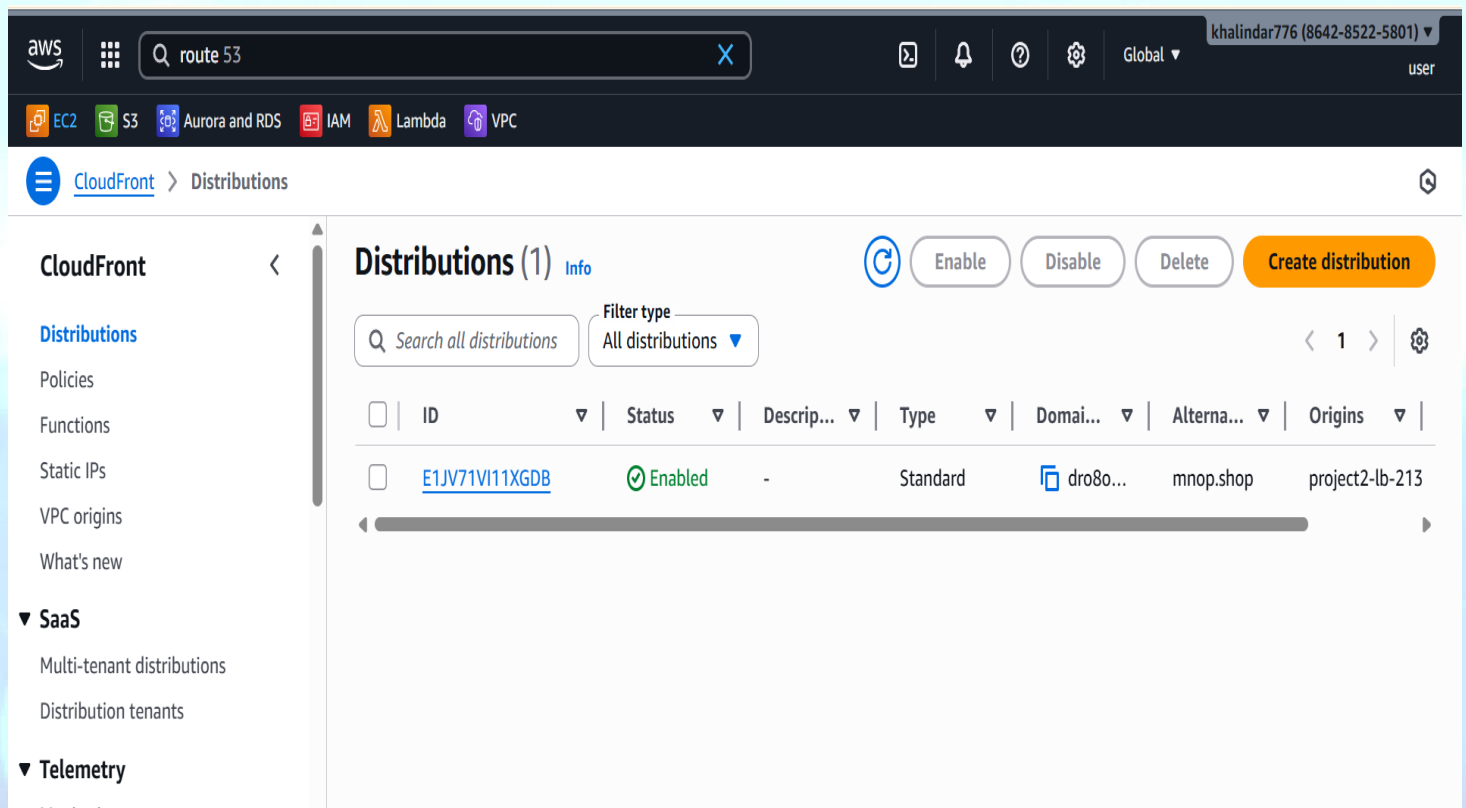
Step 15: ACM (Amazon Certificate Manager)

- Go to **ACM** and Request public certificate
- Domain: mnop.shop
- Validate via DNS
- Ensured secure routing by integrating with ACM (AWS Certificate Manager) for SSL/TLS □ Actually it takes time to issue the certificate status (have to wait some time).
- Once issued the certificate, and then automatically create the CNAME in Hosted zone (Route-53).



Step 16: Cloud Front (Content Delivery Network) with securely

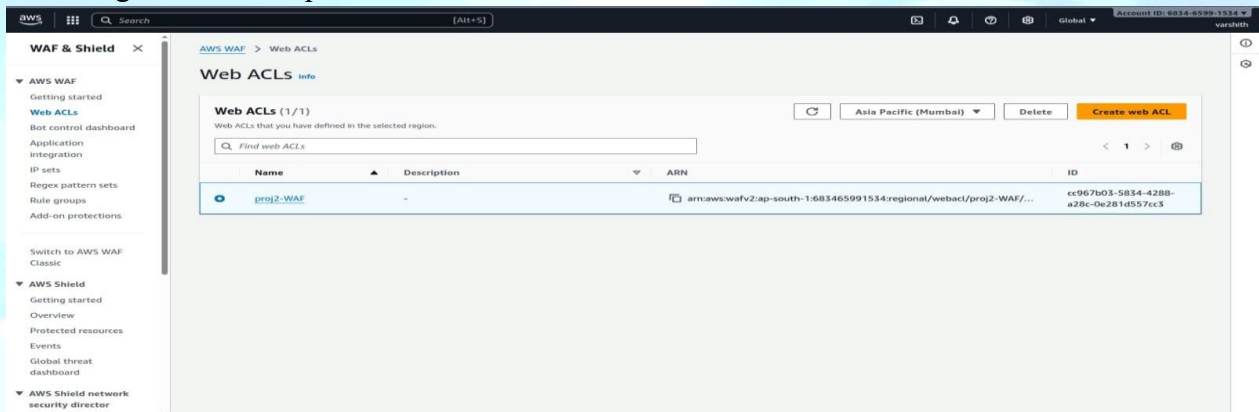
- Create Cloudfront Distribution and the name is **cloudfront-route53**.
- Created a **CloudFront Distribution** to serve static and dynamic content globally  Linked the distribution to the ALB/ S3 bucket for origin content.
- Configured **Caching Behavior** to optimize performance and reduce latency



Step 17: WAF (Web Application Firewall) and shield {Secure Your Application from Threats}

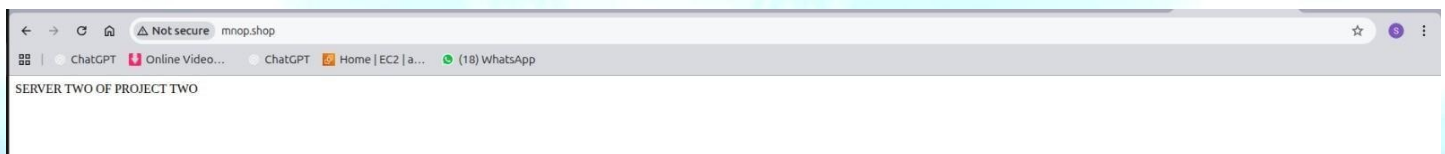
Web ACL:

- Created a **Web ACL** to define security rules for the application
- Attached the Web ACL to the Application Load Balancer (ALB)
- Monitored traffic using **WAF Logs** via CloudWatch for insights Validated rule effectiveness by simulating malicious request.



Step 18: Output

- Here we have to check the servers with DNS name (mnop.shop) still it is not secure. So, we need to add the Listeners for security purpose.



Step 19 : Add Listener for secure the connection – HTTPS (443)

- Go to Load Balancer and add Listener.
- Select the protocol (HTTPS - 443).
- Select existing Target Group (TG-route53) and ACM (DNS Name) □ Finally added the Listener ,it shows HTTP-80 and HTTPS-443.

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:ap-south-1:683465991534:loadbalancer/app/project2-LB/a5f6b0124bdc66c9

ChatGPT Online Video... ChatGPT Home | EC2 | a... (18) WhatsApp

aws waf & Shield

Account ID: 6834-6599-1534 varshith

EC2 > Load balancers > project2-LB > Add listener

Add listener Info

Add a listener to your Application Load Balancer (ALB) to define how client requests and network traffic are routed within your application. Every listener is made up of a default action that's required and can only be edited. Additional rules can be added, edited and deleted from the listener.

▶ Load balancer details: project2-LB

Listener: HTTPS:443
A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol
Used for connections from clients to the load balancer.
HTTPS

Port
The port on which the load balancer is listening for connections.
443
1-65535

Default action Info
The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Authentication action - optional Info
Authentication requires IPv4 connectivity to authentication endpoints. [Learn more](#) Info

☐ **Authenticate users**
Configure user authentication through either OpenID Connect (OIDC) or Amazon Cognito.

Routing action

☒ Forward to target groups ☐ Redirect to URL ☐ Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or [create target group](#) Info

Target group
project2-TG
Target type: Instance, IPv4 | Target stickiness: Off HTTP

Weight
1
0-999

Percent
100%

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:ap-south-1:683465991534:loadbalancer/app/project2-LB/a5f6b0124bdc66c9;tab=listeners

ChatGPT Online Video... ChatGPT Home | EC2 | a... (18) WhatsApp

aws waf & Shield

Account ID: 6834-6599-1534 varshith

EC2 > Load balancers > project2-LB

Capacity Reservations

▼ **Images**
AMIs
AMI Catalog

▼ **Elastic Block Store**
Volumes
Snapshots
Lifecycle Manager

▼ **Network & Security**
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

▼ **Load Balancing**
Load Balancers
Target Groups
Trust Stores

▼ **Auto Scaling**
Auto Scaling Groups

Settings

Application

Scheme
Internet-facing

Hosted zone
ZP97RAFLXTNJK

Availability Zones
subnet-01c15b4e3256f211a ap-south-1a (aps1-az1)
subnet-0b87dc9a15f6c6baf ap-south-1b (aps1-az3)

Date created
October 10, 2025, 15:29 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:683465991534:loadbalancer/app/project2-LB/a5f6b0124bdc66c9

DNS name Info
project2-LB-408976472.ap-south-1.elb.amazonaws.com (A Record)

Listeners and rules (2) Info

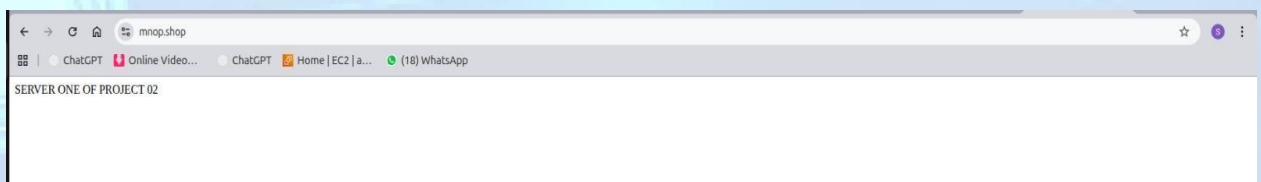
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

<input type="checkbox"/>	Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
<input type="checkbox"/>	HTTP:80	• Forward to target group project2-TG 1 (100%) Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable
<input type="checkbox"/>	HTTPS:443 ⚠ Not reachable	• Forward to target group project2-TG 1 (100%) Target group stickiness: Off	1 rule	ARN	ELBSecurityPolicy-TLS13-1-2-...	mnop.shop (Certificate ID: efb...	Off

Step 20: Final output

- Check with DNS like mnop.shop in web server then the output shows the connection is Secure for both server-1 and server -2.



Conclusion

The depicted AWS Cloud architecture offers a robust and well-structured foundation for deploying modern web applications. By strategically integrating services such as Route 53, CloudFront, WAF, Application Load Balancer (ALB), EC2 with Auto Scaling, and RDS within a Virtual Private Cloud (VPC), it ensures a solution that is inherently scalable, highly available, secure, and performance-optimized. This allows businesses to concentrate on innovation and application development, while AWS handles the complexities of infrastructure management.

The background features a large, faint, light blue AWS logo. Below the logo is a yellow curved line, resembling a smile. At the bottom, there is a network diagram with two circular nodes connected by lines. The overall background is a light blue gradient with some abstract cloud-like shapes.

THANK YOU