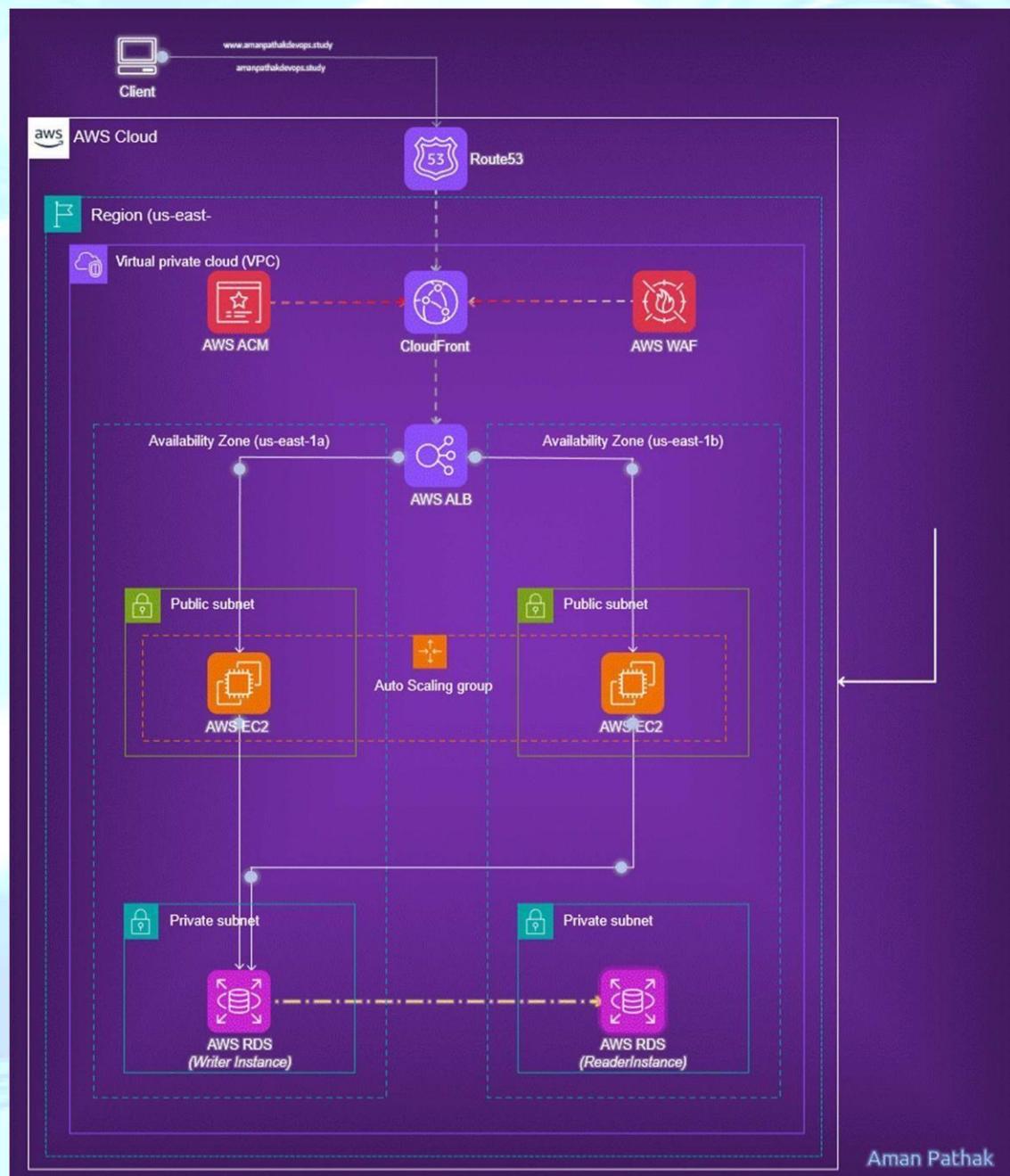


**NAME: S. khalindar , BATCH NO: 140**

## 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).

- Cerate VPC
- VPC name is **project-02-vpc**
- IPV4 address 120.0.0.0/16
- VPC created

The screenshot shows the AWS VPC console interface. At the top, there's a search bar and navigation links for EC2, S3, Aurora and RDS, IAM, Lambda, and VPC. The VPC link is highlighted. Below the navigation is a breadcrumb trail: VPC > Your VPCs. On the left, a sidebar titled 'VPC dashboard' lists 'Virtual private cloud' options like 'Your VPCs', 'Subnets', 'Route tables', 'Internet gateways', 'Egress-only Internet gateways', 'DHCP option sets', and 'Elastic IPs'. The main area is titled 'Your VPCs (2)' and contains a table with two rows. The columns are 'Name', 'VPC ID', 'State', 'Block Public...', and 'IPv4 CIDR'. The first row has a Name of '-' and a VPC ID of 'vpc-0940723d4a8ad3f0d', both marked as 'Available' with 'Off' for Block Public... and an IPv4 CIDR of '172.31.0.0/16'. The second row has a Name of 'project-02-vpc' and a VPC ID of 'vpc-0550d6e8f86da1900', also marked as 'Available' with 'Off' for Block Public... and an IPv4 CIDR of '10.0.0.0/16'. A message 'Select a VPC above' is displayed below the table.

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

### 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

The screenshot shows the AWS VPC Subnets page. On the left, there's a sidebar for 'Virtual private cloud' with options like 'Your VPCs', 'Subnets', 'Route tables', etc. The main area displays a table titled 'Subnets (7)'. The columns are 'Name', 'Subnet ID', 'State', and 'VPC'. The subnets listed are: private-1-route53, public-2-route53, private-2-route53, and public-1-route53, all in an 'Available' state. A search bar at the top says 'Find subnets by attribute or tag'. An orange 'Create subnet' button is at the top right.

Name	Subnet ID	State	VPC
private-1-route53	subnet-07c27b3c3929da582	Available	vpc-0550d6e8f86da1900   proj...
public-2-route53	subnet-08fefafa02733cf62f0	Available	vpc-0550d6e8f86da1900   proj...
private-2-route53	subnet-081111a5aed038116	Available	vpc-0550d6e8f86da1900   proj...
public-1-route53	subnet-0bf3ad826ebc1f619	Available	vpc-0550d6e8f86da1900   proj...

### 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)

The screenshot shows the AWS Internet Gateways page. The sidebar includes 'Your VPCs', 'Subnets', 'Route tables', and 'Internet gateways'. The main area shows a table titled 'Internet gateways (2)'. The columns are 'Name', 'Internet gateway ID', 'State', and 'VPC ID'. The gateways listed are: project-02-igw and - (with a hyphen), both in an 'Attached' state. A search bar at the top says 'Find internet gateways by attribute or tag'. An orange 'Create internet gateway' button is at the top right.

Name	Internet gateway ID	State	VPC ID
project-02-igw	igw-01134f2acaf4316cd	Attached	vpc-0550d6e8f86da1900   proj...
-	igw-08ab6e3c6e2e911b3	Attached	vpc-0940723d4a8ad3f0d

## 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 VPC Route tables page. At the top, there's a navigation bar with icons for EC2, S3, Aurora and RDS, IAM, Lambda, and VPC. The VPC icon is highlighted. Below the navigation bar, the page title is "Route tables (4) Info". A search bar says "Find route tables by attribute or tag". To the right, there are buttons for "Actions" and "Create route table". On the left, there's a sidebar titled "Virtual private cloud" with sections for "Your VPCs", "Subnets", "Route tables" (which is selected), "Internet gateways", "Egress-only Internet gateways", "DHCP option sets", and "Elastic IPs". A "Managed prefix lists" section is also present. The main content area displays a table of route tables:

Name	Route table ID	Explicit subnet associations	Edge associations	Main
-	<a href="#">rtb-0607a4a028719684f</a>	-	-	Yes
project-2-private-rt	<a href="#">rtb-0d4f0e5edb1480fb3</a>	2 subnets	-	No
project02-pub-rt	<a href="#">rtb-0742c9a7333349de6</a>	2 subnets	-	No
-	<a href="#">rtb-0d21c8f252bd64eac</a>	-	-	Yes

## 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

The screenshot shows the AWS Management Console with the EC2 service selected. In the left navigation pane, under 'Network & Security', 'Security Groups' is selected. The main content area displays a table titled 'Security Groups (7)'. The columns are 'Name', 'Security group ID', 'Security group name', and 'VPC ID'. The table lists seven security groups: 'project2-sg' (sg-0526291afa61360aa), 'default' (sg-0e169b522411fb4ef), 'launch-wizard-2' (sg-0d22a57b383a15d7c), and 'launch-wizard-4' (sg-03f093aa9533d0947). Each row has a link to its details page.

## 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 (sgroute53) 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.

The screenshot shows the AWS Management Console with the EC2 service selected. In the left navigation pane, under 'Instances', 'Instances' is selected. The main content area displays a table titled 'Instances (4)'. The columns are 'Name', 'Instance ID', 'Instance state', 'Instance type', 'Status check', and 'Alarm status'. The table lists four instances: 'webserver2' (i-0f1513a13de313bea), 'new1' (i-0dd159e0233df05f), 'new2' (i-0e036c485a1ee09e7), and 'webserver01' (i-0322d7186b2544dc2). All instances are shown as 'Running' with green checkmarks. Each row has a link to its details page.

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

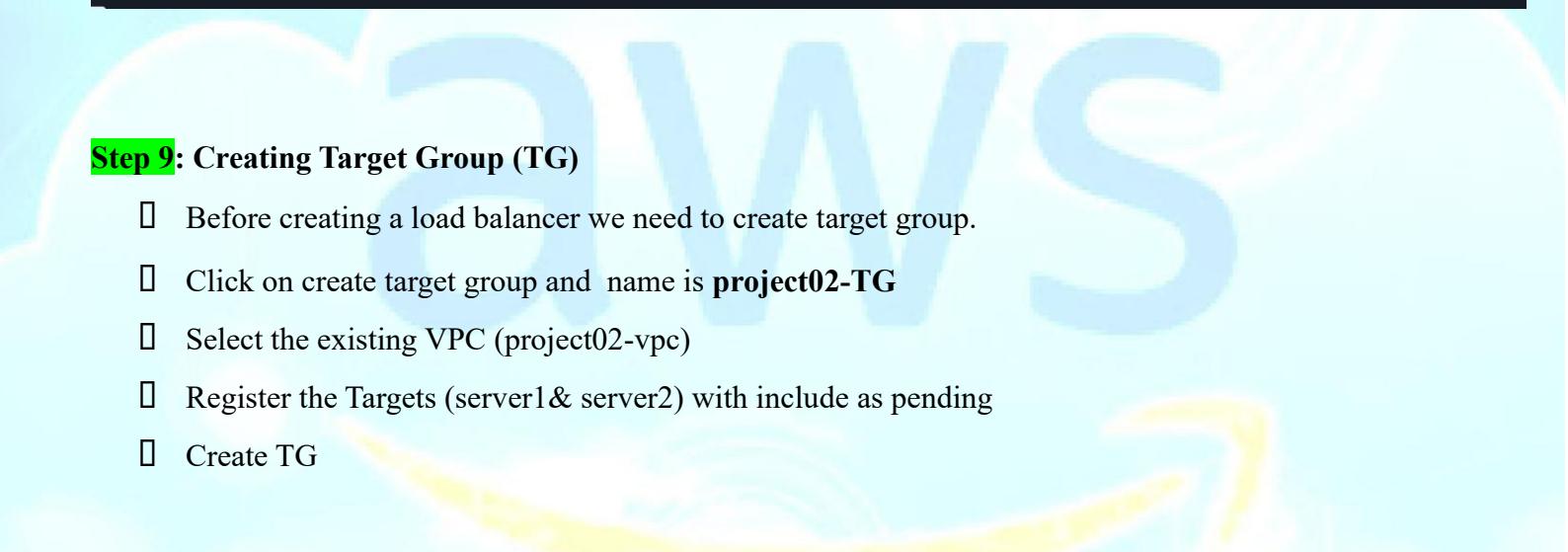
- We need to setup auto scaling on that create AMI and launch template.
- First create a AMI and name is **server1-ami**
- To create an AMI first select the instance and in actions click on image and template option then click on image. □ Provide the details for AMI
- Create AMI.

The screenshot shows the AWS Management Console with the EC2 service selected. In the left sidebar, under 'Images', 'AMIs' is selected. The main pane displays a table titled 'Amazon Machine Images (AMIs) (1)'. The table has columns for 'Name', 'AMI ID', 'Source', and 'Owner'. One row is listed with the name 'server1-ami', AMI ID 'ami-09cdc74ed397e6279', Source '864285225801/server1-ami', and Owner '864285225801'. Below the table, a button labeled 'Select an AMI' is visible.

Name	AMI ID	Source	Owner
server1-ami	ami-09cdc74ed397e6279	864285225801/server1-ami	864285225801

## 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.



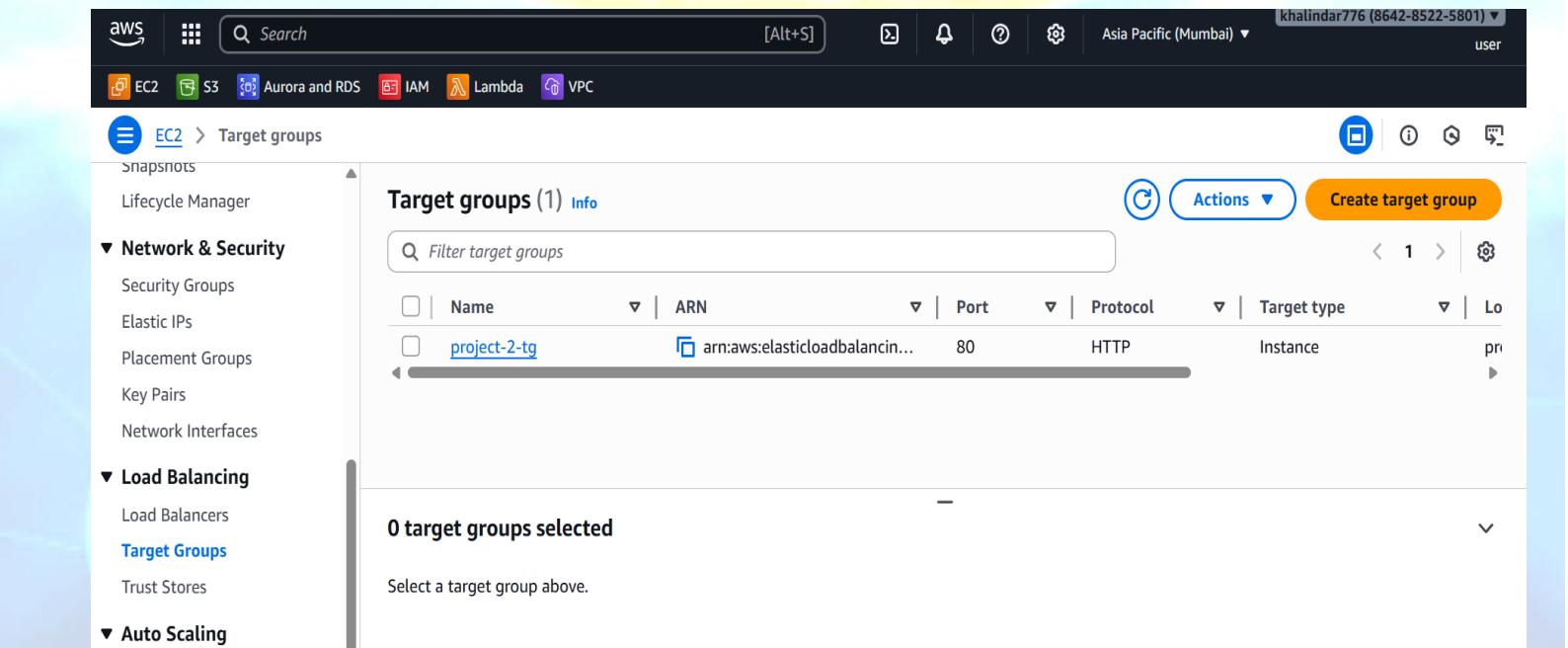
Screenshot of the AWS EC2 Launch Templates page. The left sidebar shows 'Launch Templates' is selected under 'Instances'. The main table lists one launch template:

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time
lt-07427ad01e4a04b14	project-2-template	1	1	2025-10-18T10:25:00Z

The 'Actions' button is highlighted in blue.

## 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



Screenshot of the AWS EC2 Target groups page. The left sidebar shows 'Target Groups' is selected under 'Load Balancing'. The main table lists one target group:

Name	ARN	Port	Protocol	Target type
project-2-tg	arn:aws:elasticloadbalancing:... 	80	HTTP	Instance

The 'Actions' button is highlighted in blue.

## 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

The screenshot shows the AWS EC2 Load Balancers page. On the left, there's a navigation sidebar with options like Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing (which is expanded to show Load Balancers, Target Groups, and Trust Stores), and Auto Scaling (which is also expanded to show Auto Scaling Groups). The main content area is titled 'Load balancers (1)' and contains a table with one row for 'project2-lb'. The table columns are Name, State, Type, Scheme, IP address type, and VPC ID. The 'Name' column shows 'project2-lb', 'State' shows 'Active' with a green checkmark, 'Type' shows 'application', 'Scheme' shows 'Internet-facing', 'IP address type' shows 'IPv4', and 'VPC ID' shows 'vpc-0550d6e...'. Below the table, it says '0 load balancers selected' and 'Select a load balancer above.'

## 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).



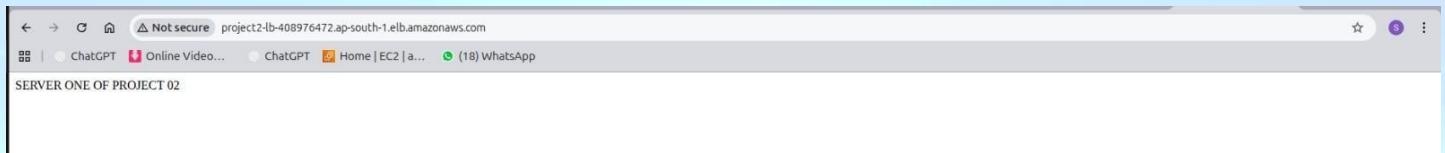
Screenshot of the AWS Management Console showing the Auto Scaling Groups page. The navigation bar includes links for EC2, S3, Aurora and RDS, IAM, Lambda, and VPC. The left sidebar shows categories like Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Auto Scaling, and Auto Scaling Groups. The main content area displays an Auto Scaling group named "project2-asg" with a launch template "project-2-template" and 2 instances. A search bar at the top allows searching for Auto Scaling groups.

### 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 : **54.209.155.54:80 /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- **SERER 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
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**.

The screenshot shows the AWS Route 53 service interface. In the top navigation bar, the user is signed in as 'khalindar776 (8642-8522-5801)'. The main menu on the left includes 'Route 53', 'Dashboard', 'Hosted zones' (which is selected), 'Health checks', 'Profiles New', 'IP-based routing', 'Traffic flow', and 'Domains'. The 'Hosted zones' section displays a table with one entry: 'mnop.shop' (Hosted zone name), 'Public' (Type), 'Route 53' (Created by), '4' (Record count), '-' (Description), and 'Z0951557...' (Hosted zone ID). There are buttons for 'View details', 'Edit', 'Delete', and 'Create hosted zone'.

## 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

This screenshot shows the 'Records' page for the 'mnop.shop' hosted zone. The left sidebar is identical to the previous screenshot. The main content area shows a table of records with the following data:

Record ...	Type	Routing p...	Differ...	Alias	Value/Route traffic to	TTL (s...)
mnop.shop	A	Simple	-	Yes	dro806kpf7cz3.cloudfront.net.	-
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
mnop.shop	SOA	Simple	-	No	ns-596.awsdns-10.net. awsd...	900
_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)

The screenshot shows a web browser window for GoDaddy's domain management interface. The URL is [dcc.godaddy.com/control/portfolio/mnop.shop/settings?ventureId=d407430b-aa58-4384-b88e-572146ae506f&ua\\_placement=shared\\_header&subta...](https://dcc.godaddy.com/control/portfolio/mnop.shop/settings?ventureId=d407430b-aa58-4384-b88e-572146ae506f&ua_placement=shared_header&subta...). The main content area is titled "Nameservers" with the sub-instruction "Using custom nameservers". A "Change Nameservers" button is visible. On the left, a sidebar lists "mnop.shop" under "Domain" and other services like "Dashboard", "Website", "Email", "Store", and "Appointments". Below the sidebar, three nameservers are listed: "ns-900.awsdns-48.net", "ns-1255.awsdns-28.org", and "ns-1976.awsdns-55.co.uk".

**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).

AWS Certificate Manager (ACM)

811fac3e-34a5-4ac9-ad4b-1271b852cd88

Certificate status

Identifier: 811fac3e-34a5-4ac9-ad4b-1271b852cd88

Status: Issued

ARN: arn:aws:acm:ap-south-1:864285225801:certificate/811fac3e-34a5-4ac9-ad4b-1271b852cd88

Type: Amazon Issued

Domains (1): mnop.shop

Create records in Route 53 | Export to CSV

## 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

CloudFront > Distributions

Distributions (1)

ID	Status	Type	Domain	Alternate	Origins
E1JV71V11XGDB	Enabled	Standard	dro80...	mnop.shop	project2-lb-213

Filter type: All distributions

Search all distributions

Enable | Disable | Delete | Create distribution

Distributions

Policies

Functions

Static IPs

VPC origins

What's new

SaaS

Multi-tenant distributions

Distribution tenants

Telemetry

Monitoring

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

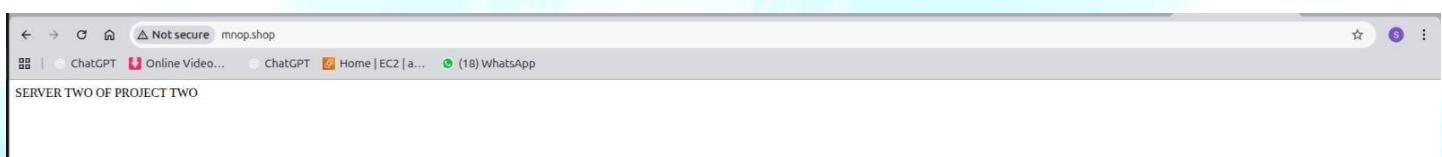
### Web ACL:

- 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.

The screenshot shows the AWS WAF & Shield console. On the left, there's a navigation sidebar with sections for AWS WAF (Getting started, Web ACLs, Bot control dashboard, Application integration, IP sets, Regex pattern sets, Rule groups, Add-on protections, Switch to AWS WAF Classic), AWS Shield (Getting started, Overview, Protected resources, Events, Global threat dashboard), and AWS Shield network security director. The main area is titled 'Web ACLs info' and shows 'Web ACLs (1/1)'. A table lists one item: 'proj2-WAF' with an ARN: arn:aws:wafv2:ap-south-1:683465991554:regional/webacl/proj2-WAF/a28c-0e281d557cc5'. There are buttons for 'Create web ACL' and 'Delete'.

## 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.

The screenshot shows the AWS CloudFormation console with a stack named 'HelloWorld'. The 'Outputs' tab is selected, displaying the output 'HelloWorldFunctionArn' with the value 'arn:aws:lambda:ap-south-1:683465991534:function:HelloWorld'. Below this, there is a table with columns 'Name', 'Type', and 'Value'.

Name	Type	Value
HelloWorldFunctionArn	AWS::Lambda::Function	arn:aws:lambda:ap-south-1:683465991534:function:HelloWorld

The screenshot shows the AWS CloudFormation console with a stack named 'HelloWorld'. The 'Outputs' tab is selected, displaying the output 'HelloWorldFunctionArn' with the value 'arn:aws:lambda:ap-south-1:683465991534:function:HelloWorld'. Below this, there is a table with columns 'Name', 'Type', and 'Value'.

Name	Type	Value
HelloWorldFunctionArn	AWS::Lambda::Function	arn:aws:lambda:ap-south-1:683465991534:function:HelloWorld

## 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.

The screenshot shows a web browser window with the URL `mnop.shop`. The page content is 'SERVER ONE OF PROJECT 02'.

## 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.

# THANK YOU