CLOUD INFRASTRUCTURE & SECURITY TEJAS

Cloud Security with AWS – Assignment Report

Hosting a Website on EC2 with a Load Balancer

Submitted by: TEJAS K MAHALE

Date: 22/02/25

Course: Cloud Security with AWS

Introduction

This report outlines the process of deploying a website on an **Amazon EC2 instance** and configuring an **Application Load Balancer (ALB)** to distribute traffic efficiently. The goal is to implement a scalable and highly available infrastructure using AWS services.

Objectives

- Deploy and configure an **EC2 instance** as a web server
- Install and set up **Apache Web Server**
- Set up an **Application Load Balancer (ALB)** for load distribution
- Verify the accessibility of the website through the **Load Balancer's DNS**
- Understand the **benefits of load balancing** in cloud security

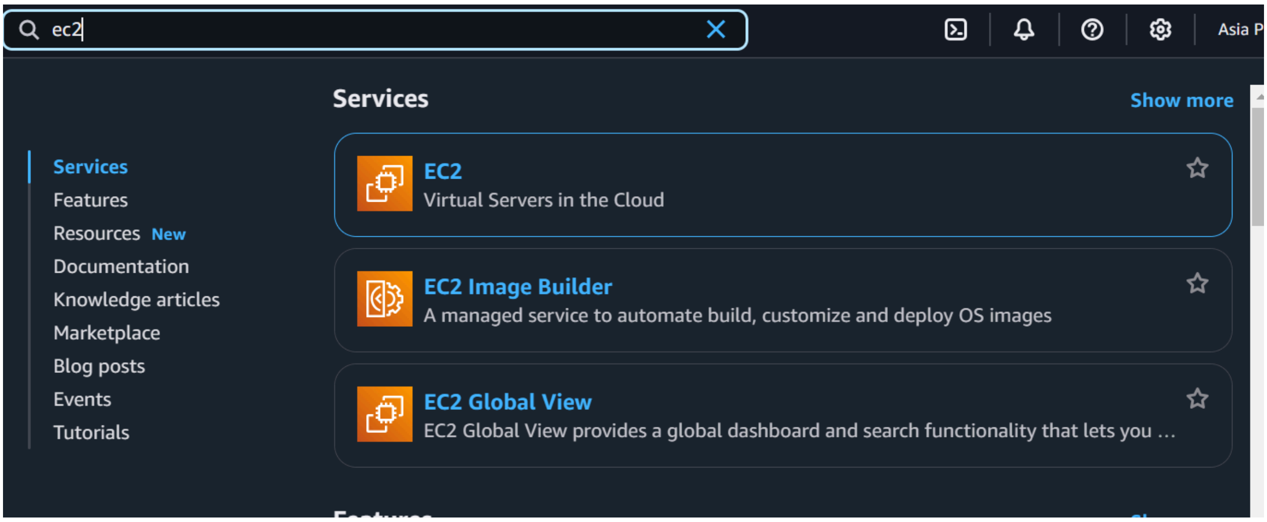
---

Steps to Deploy a Website on EC2

1. Launching an EC2 Instance

1. Sign in to the **AWS Management Console** and open the **EC2 Dashboard**
2. Click **Launch Instance** and provide a name (e.g., "WebServer1")
3. Select **Amazon Linux 2 AMI** as the operating system

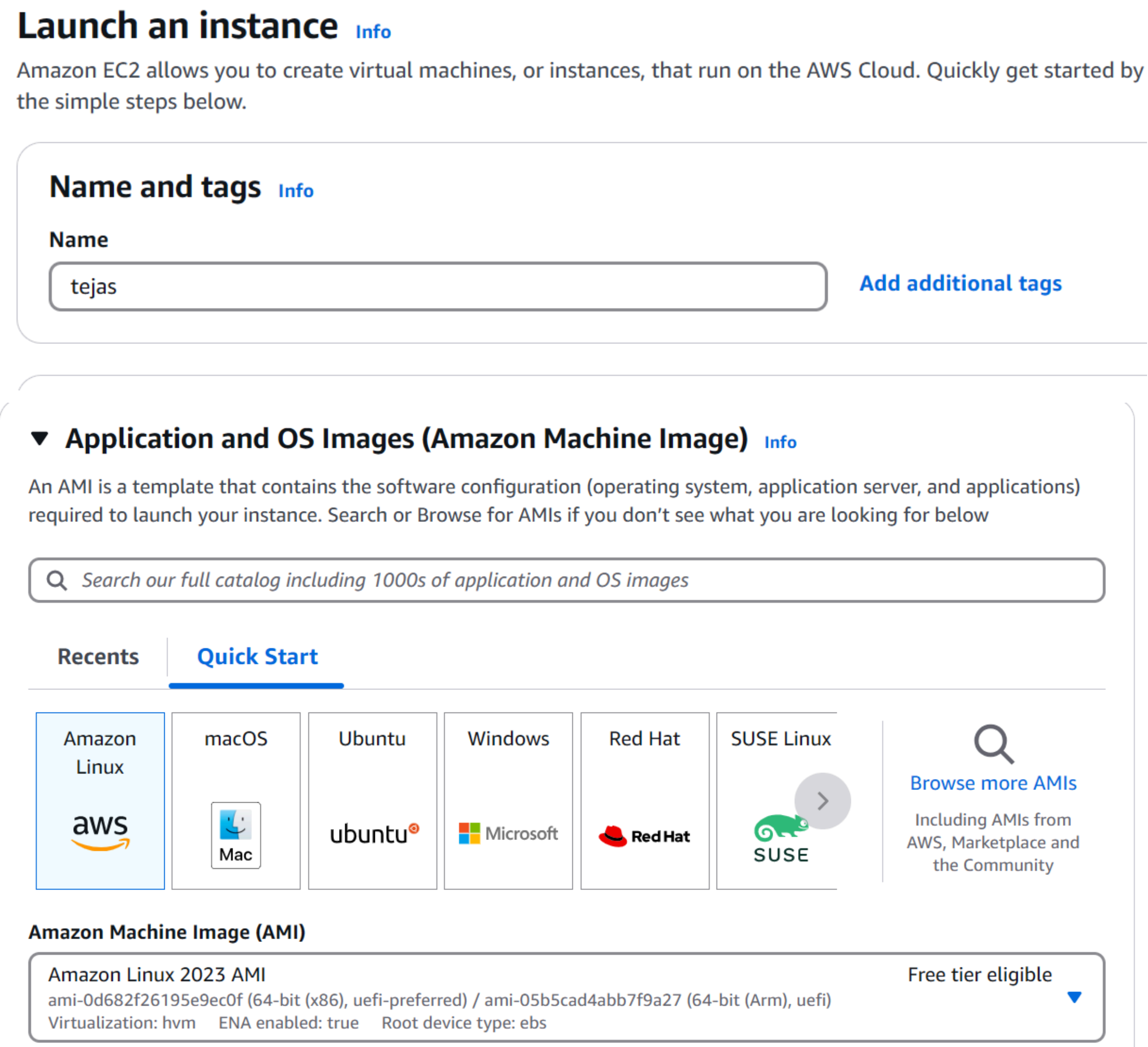
4. Choose or create a **key pair** for SSH access
5. Configure **Security Group** settings:
   - Allow **SSH (22)** for administrative access
   - Allow **HTTP (80)** for web traffic
   - Allow **HTTPS (443)** for secure web access
6. Click **Launch** and wait for the instance to start



EC2
Virtual Servers in the Cloud

EC2 Image Builder
A managed service to automate build, customize and deploy OS images

EC2 Global View
EC2 Global View provides a global dashboard and search functionality that lets you …



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by the simple steps below.

Name and tags Info

Name: tejas

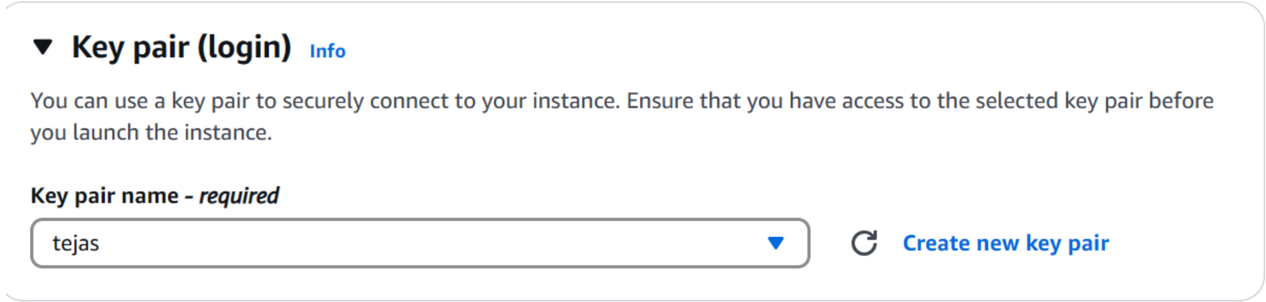
▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
aws
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
SUSE Linux
SUSE
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-0d682f26195e9ec0f (64-bit (x86), uefi-preferred) / ami-05b5cad4abb7f9a27 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs



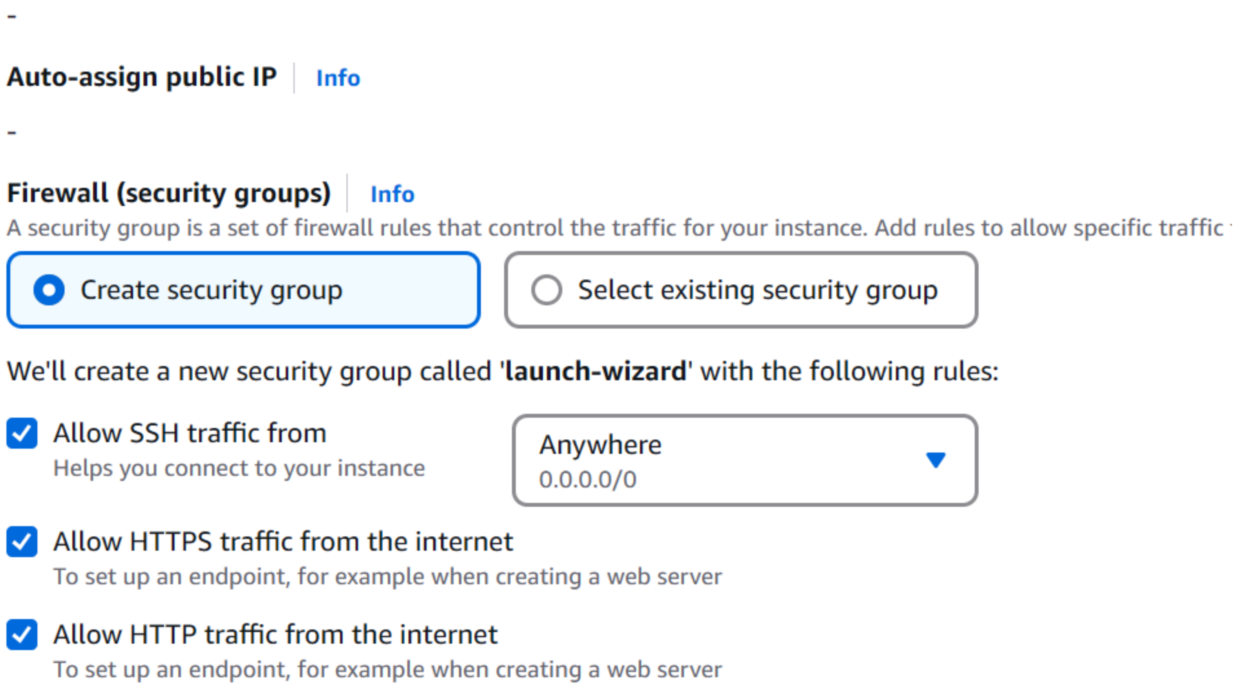
▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

tejas ▼

Create new key pair



-

Auto-assign public IP | Info

-

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic

● Create security group ○ Select existing security group

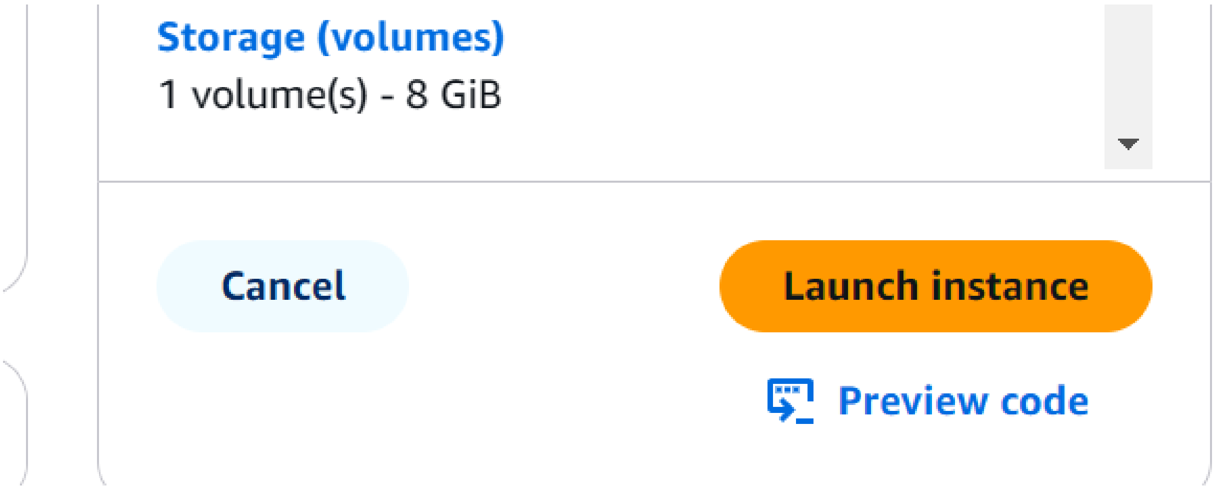
We'll create a new security group called '**launch-wizard**' with the following rules:

☑ Allow SSH traffic from
Helps you connect to your instance

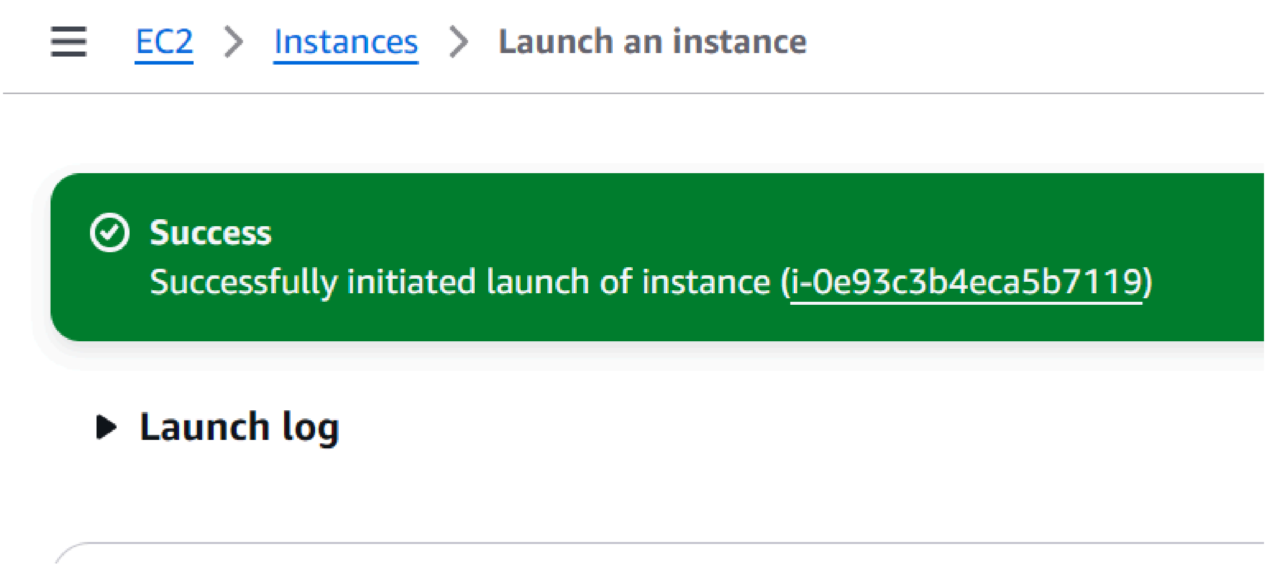
Anywhere
0.0.0.0/0 ▼

☑ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server



Storage (volumes)
1 volume(s) - 8 GiB



Success
Successfully initiated launch of instance (i-0e93c3b4eca5b7119)

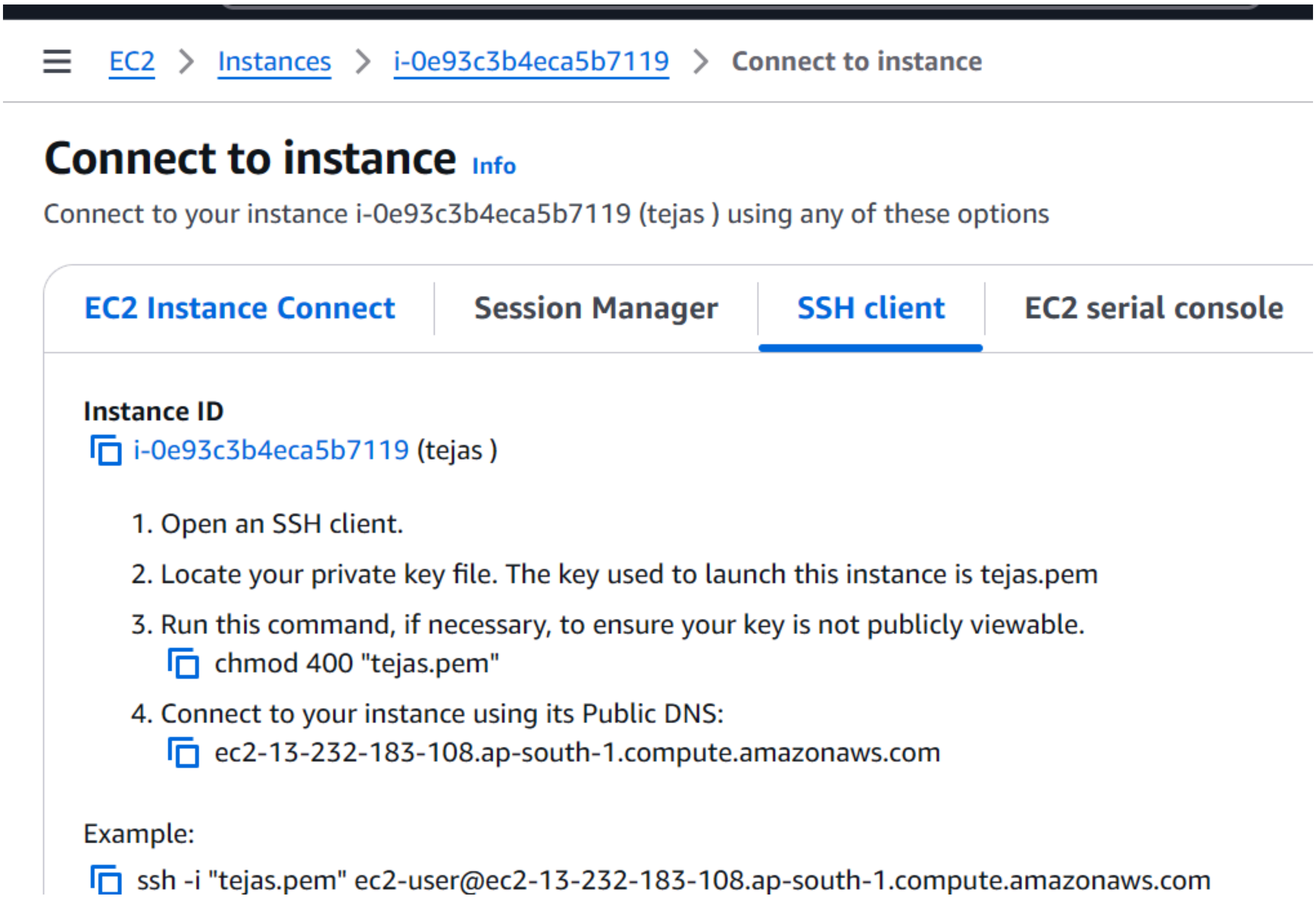
2. Connecting to EC2 via SSH

1. Open a terminal or command prompt and connect to the instance using the key pair:

```
ssh -i your-key.pem ec2-user@your-ec2-public-ip
```

2. Update system packages and install Apache:

```
sudo yum update -y
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
```



Connect to instance Info

Connect to your instance i-0e93c3b4eca5b7119 (tejas) using any of these options

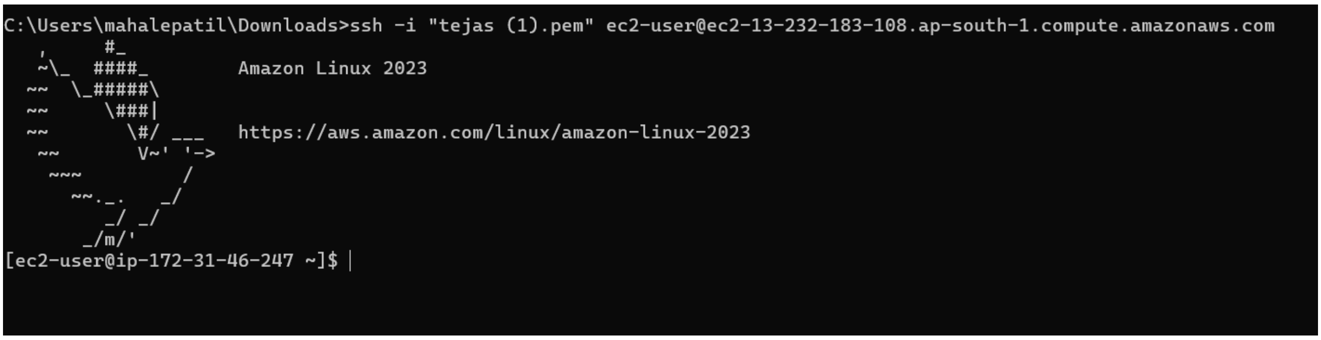
Instance ID

i-0e93c3b4eca5b7119 (tejas)

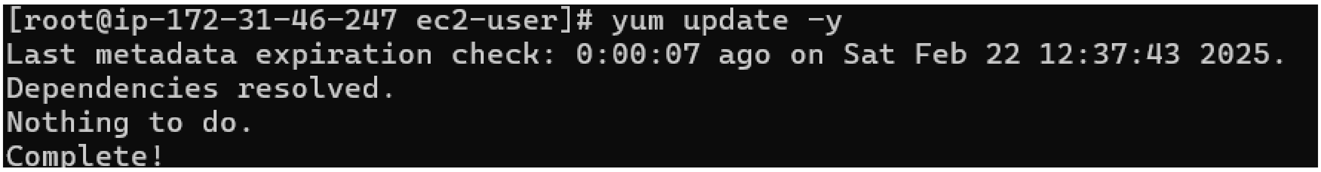
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is tejas.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "tejas.pem"
4. Connect to your instance using its Public DNS:
   ec2-13-232-183-108.ap-south-1.compute.amazonaws.com

Example:

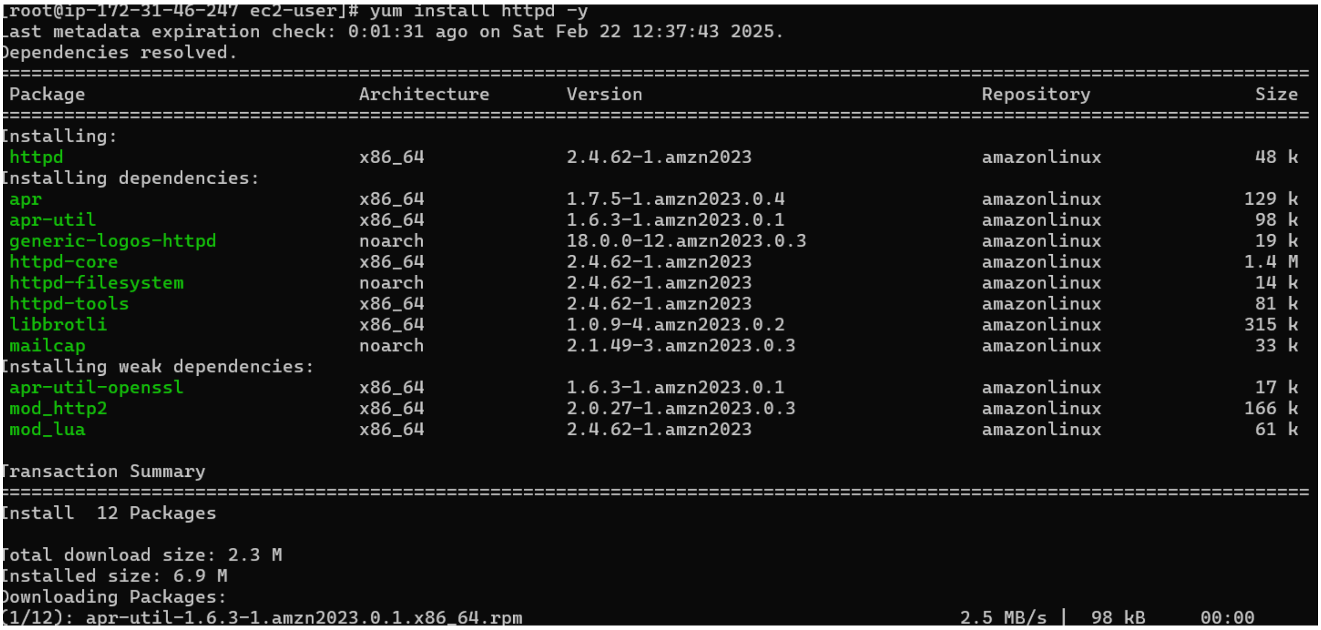
```
ssh -i "tejas.pem" ec2-user@ec2-13-232-183-108.ap-south-1.compute.amazonaws.com
```



```
C:\Users\mahalepatil\Downloads>ssh -i "tejas (1).pem" ec2-user@ec2-13-232-183-108.ap-south-1.compute.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-46-247 ~]$
```



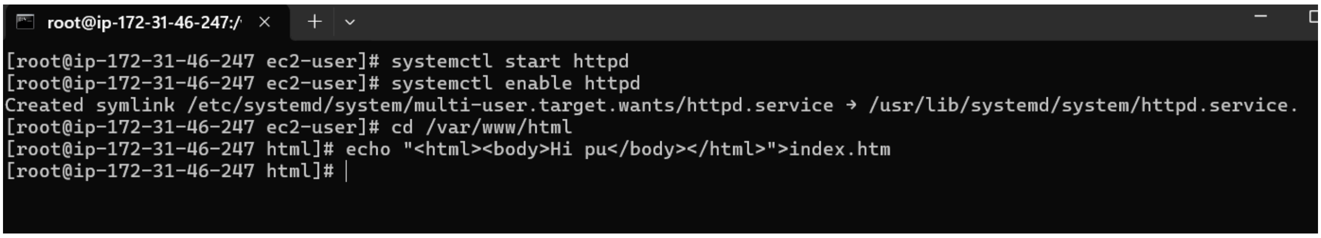
```
[root@ip-172-31-46-247 ec2-user]# yum update -y
Last metadata expiration check: 0:00:07 ago on Sat Feb 22 12:37:43 2025.
Dependencies resolved.
Nothing to do.
Complete!
```



```
[root@ip-172-31-46-247 ec2-user]# yum install httpd -y
Last metadata expiration check: 0:01:31 ago on Sat Feb 22 12:37:43 2025.
Dependencies resolved.
==========================================================================================================
 Package                   Architecture   Version                        Repository                  Size
==========================================================================================================
Installing:
 httpd                     x86_64         2.4.62-1.amzn2023              amazonlinux                 48 k
Installing dependencies:
 apr                       x86_64         1.7.5-1.amzn2023.0.4           amazonlinux                129 k
 apr-util                  x86_64         1.6.3-1.amzn2023.0.1           amazonlinux                 98 k
 generic-logos-httpd       noarch         18.0.0-12.amzn2023.0.3         amazonlinux                 19 k
 httpd-core                x86_64         2.4.62-1.amzn2023              amazonlinux                1.4 M
 httpd-filesystem          noarch         2.4.62-1.amzn2023              amazonlinux                 14 k
 httpd-tools               x86_64         2.4.62-1.amzn2023              amazonlinux                 81 k
 libbrotli                 x86_64         1.0.9-4.amzn2023.0.2           amazonlinux                315 k
 mailcap                   noarch         2.1.49-3.amzn2023.0.3          amazonlinux                 33 k
Installing weak dependencies:
 apr-util-openssl          x86_64         1.6.3-1.amzn2023.0.1           amazonlinux                 17 k
 mod_http2                 x86_64         2.0.27-1.amzn2023.0.3          amazonlinux                166 k
 mod_lua                   x86_64         2.4.62-1.amzn2023              amazonlinux                 61 k

Transaction Summary
==========================================================================================================
Install  12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:
(1/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm                        2.5 MB/s |  98 kB     00:00
```



```
[root@ip-172-31-46-247 ec2-user]# systemctl start httpd
[root@ip-172-31-46-247 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-46-247 ec2-user]# cd /var/www/html
[root@ip-172-31-46-247 html]# echo "<html><body>Hi pu</body></html>">index.htm
[root@ip-172-31-46-247 html]# |
```

3. Creating a Web Page

1. Navigate to the web root directory:

```
cd /var/www/html/
```

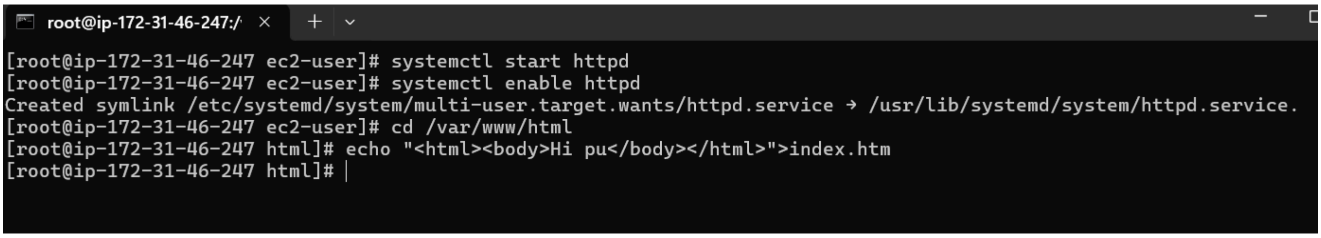
2. Create an `index.html` file with a simple message:

```
echo "<h1>Welcome to My AWS Website</h1>" | sudo tee index.html
```

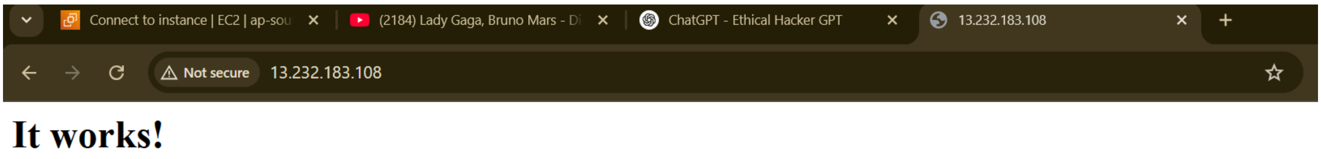
3. Open a web browser and enter the **EC2 Public IP** to verify the webpage:

```
http://your-ec2-public-ip
```

If configured correctly, the webpage should display "Welcome to My AWS Website."



```
[root@ip-172-31-46-247 ec2-user]# systemctl start httpd
[root@ip-172-31-46-247 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-46-247 ec2-user]# cd /var/www/html
[root@ip-172-31-46-247 html]# echo "<html><body>Hi pu</body></html>">index.htm
[root@ip-172-31-46-247 html]# |
```



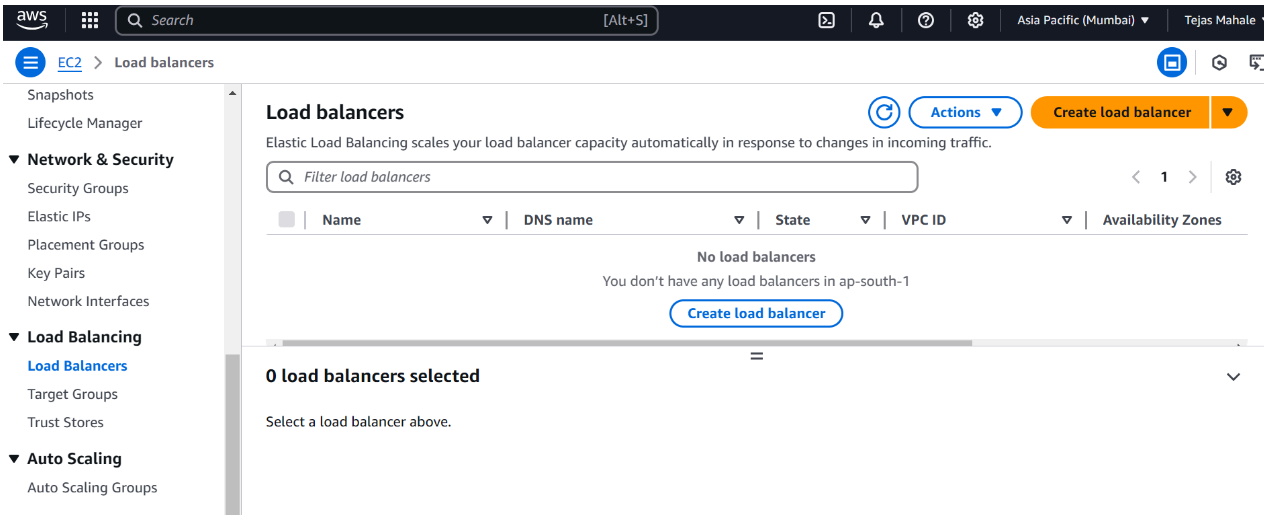
Connect to instance | EC2 | ap-sou
(2184) Lady Gaga, Bruno Mars - Di
ChatGPT - Ethical Hacker GPT
13.232.183.108
Not secure
13.232.183.108

It works!

Setting Up an Application Load Balancer (ALB)

4. Creating an Application Load Balancer

1. Navigate to the **EC2 Dashboard** and go to **Load Balancers**
2. Click **Create Load Balancer** and choose **Application Load Balancer (ALB)**
3. Configure the ALB settings:
   - Name: "MyALB"
   - Scheme: Internet-facing
   - VPC: Select the same VPC as the EC2 instance
   - Availability Zones: Select at least two zones
   - Security Group: Ensure HTTP (80) is allowed
4. Create a **Target Group** and register the EC2 instance
5. Click **Create Load Balancer**



Load balancers

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

Name	DNS name	State	VPC ID	Availability Zones

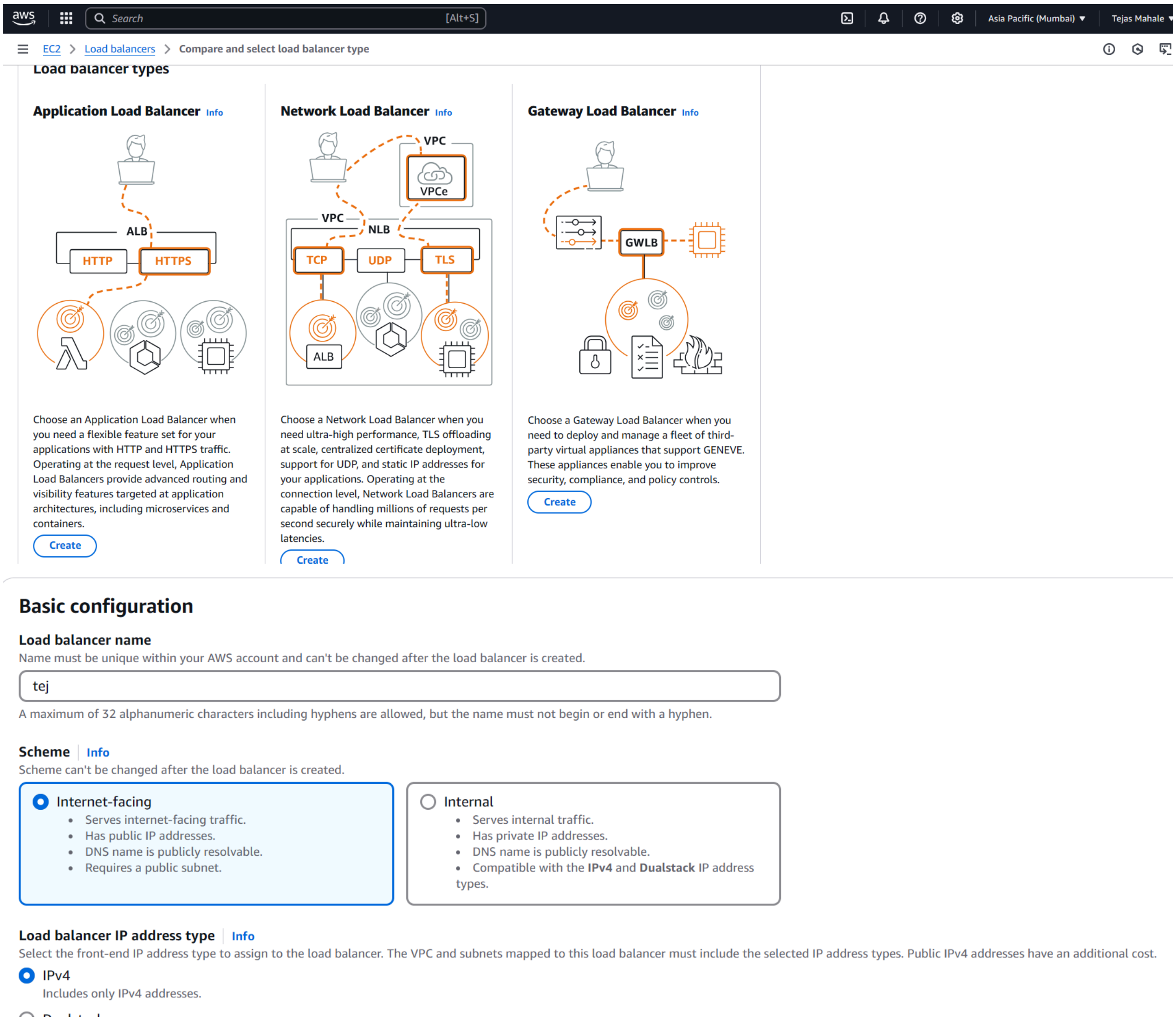
No load balancers

You don't have any load balancers in ap-south-1

Create load balancer

0 load balancers selected

Select a load balancer above.



EC2 > Load balancers > Compare and select load balancer type

Load balancer types

Application Load Balancer Info

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

Create

Network Load Balancer Info

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

Create

Gateway Load Balancer Info

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

Create

Basic configuration

Load balancer name

Name must be unique within your AWS account and can't be changed after the load balancer is created.

tej

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info

Scheme can't be changed after the load balancer is created.

● Internet-facing
- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name is publicly resolvable.
- Requires a public subnet.

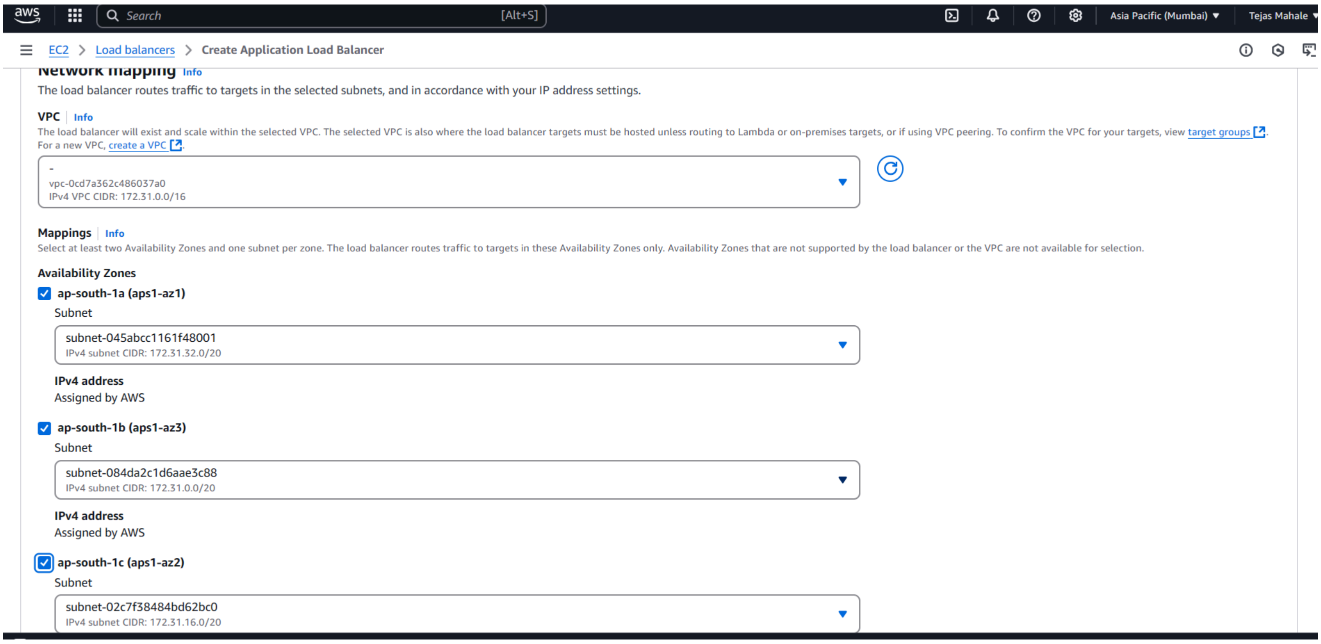
○ Internal
- Serves internal traffic.
- Has private IP addresses.
- DNS name is publicly resolvable.
- Compatible with the **IPv4** and **Dualstack** IP address types.

Load balancer IP address type Info

Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.

● IPv4

Includes only IPv4 addresses.


EC2 > Load balancers > Create Application Load Balancer

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC | Info

The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view target groups. For a new VPC, create a VPC.

-
vpc-0cd7a362c486037a0
IPv4 VPC CIDR: 172.31.0.0/16

Mappings | Info

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

Availability Zones

☑ ap-south-1a (aps1-az1)

Subnet: subnet-045abcc1161f48001
IPv4 subnet CIDR: 172.31.32.0/20

IPv4 address: Assigned by AWS

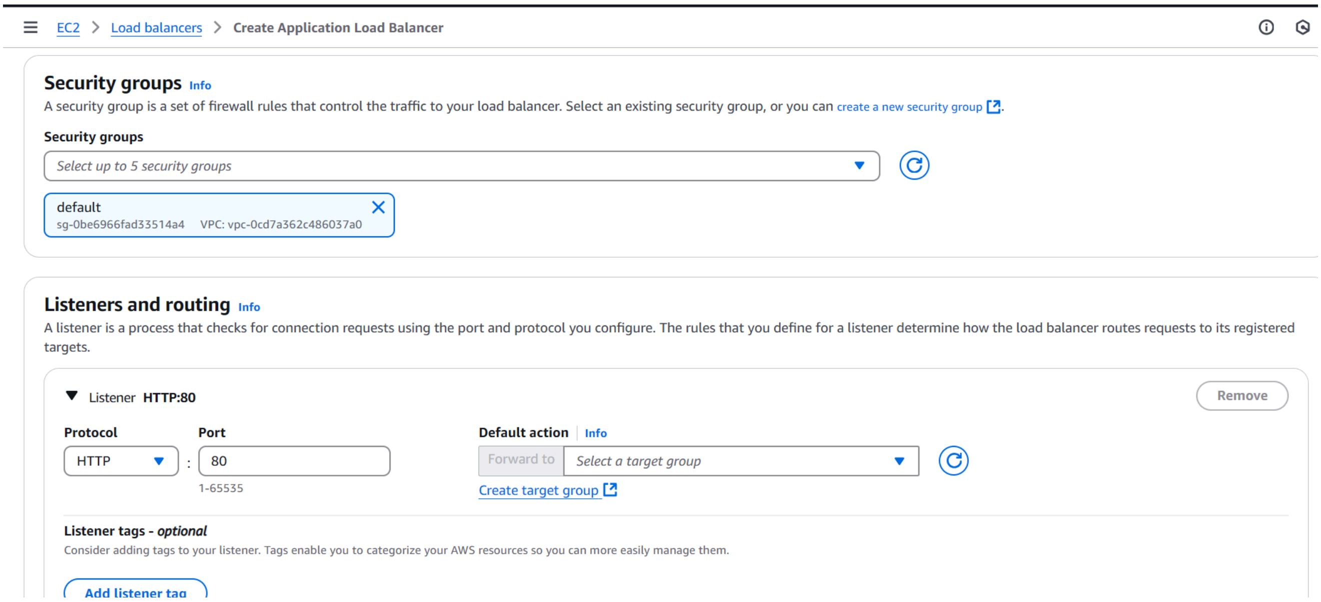
☑ ap-south-1b (aps1-az3)

Subnet: subnet-084da2c1d6aae3c88
IPv4 subnet CIDR: 172.31.0.0/20

IPv4 address: Assigned by AWS

☑ ap-south-1c (aps1-az2)

Subnet: subnet-02c7f38484bd62bc0
IPv4 subnet CIDR: 172.31.16.0/20


EC2 > Load balancers > Create Application Load Balancer

Security groups Info

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can create a new security group.

Security groups: Select up to 5 security groups

default
sg-0be6966fad33514a4 VPC: vpc-0cd7a362c486037a0

Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Listener HTTP:80

Protocol: HTTP
Port: 80
1-65535

Default action | Info
Forward to: Select a target group
Create target group

Listener tags - *optional*

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

5. Verifying Website Accessibility via Load Balancer

1. Copy the **DNS Name** of the ALB from the AWS console
2. Open a browser and enter the **Load Balancer's DNS**:

```
http://your-load-balancer-dns
```

3. The webpage should load, confirming that traffic is being routed through the **Load Balancer**

Understanding Load Balancing in AWS

An **Application Load Balancer (ALB)** is designed to distribute traffic across multiple EC2 instances, improving performance, reliability, and scalability.

Key Benefits of ALB

- Ensures **high availability** by distributing traffic evenly
- Supports **scalability** to handle increased traffic loads
- Improves **fault tolerance** by redirecting traffic if an instance fails
- Provides **layer 7 routing** for enhanced request management

---

Conclusion

This assignment successfully demonstrated the deployment of a website on **Amazon EC2**, the installation of a **web server**, and the configuration of an **Application Load Balancer** to ensure scalability and reliability. By implementing **load balancing**, traffic is efficiently distributed, ensuring website availability even in case of failures.

Steps to Configure AWS WAF

Introduction

This report provides a step-by-step guide to configuring **AWS Web Application Firewall (WAF)** to enhance the security of a web application by restricting access from a specific IP range. The **WAF is attached to an Application Load Balancer (ALB)** to filter and manage incoming traffic, ensuring that unauthorized requests are blocked before reaching the application.

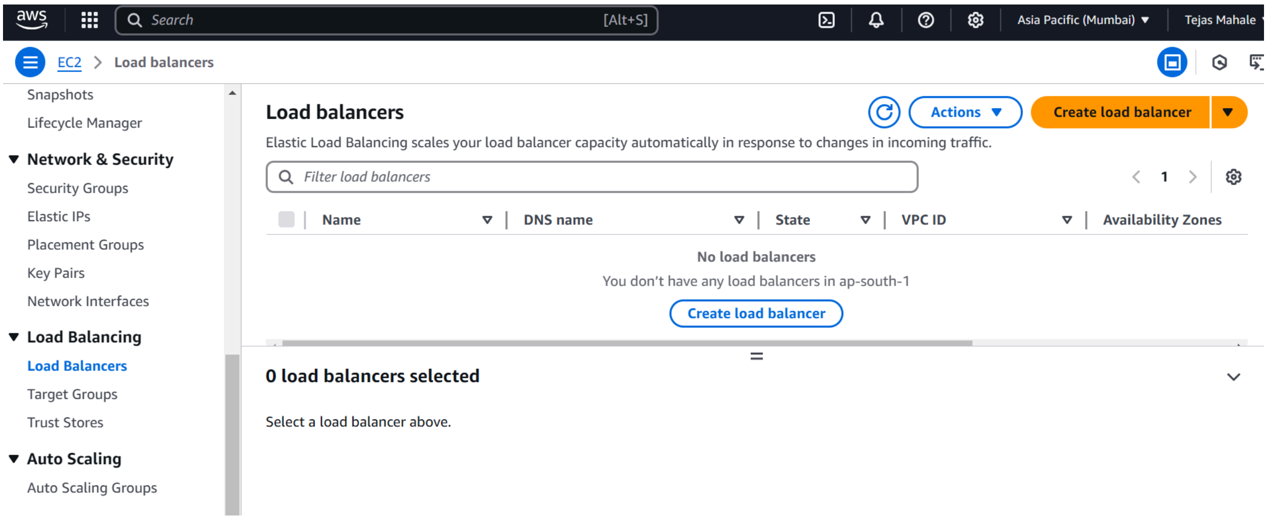
Objectives

- Deploy and configure **AWS WAF** for web application security.
- Set up **rules to block specific IP addresses**.
- Attach **AWS WAF to an Application Load Balancer (ALB)**.
- Test and verify that traffic from blocked IPs is rejected.

Steps to Configure AWS WAF

1. Selecting the Application Load Balancer (ALB)

1. Log in to the **AWS Management Console**.
2. Navigate to **EC2 Dashboard** → **Load Balancers**.
3. Select an existing **Application Load Balancer (ALB)** or create a new one.
4. Ensure that the ALB is configured to handle web traffic (HTTP/HTTPS).



Load balancers

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

□	Name	DNS name	State	VPC ID	Availability Zones

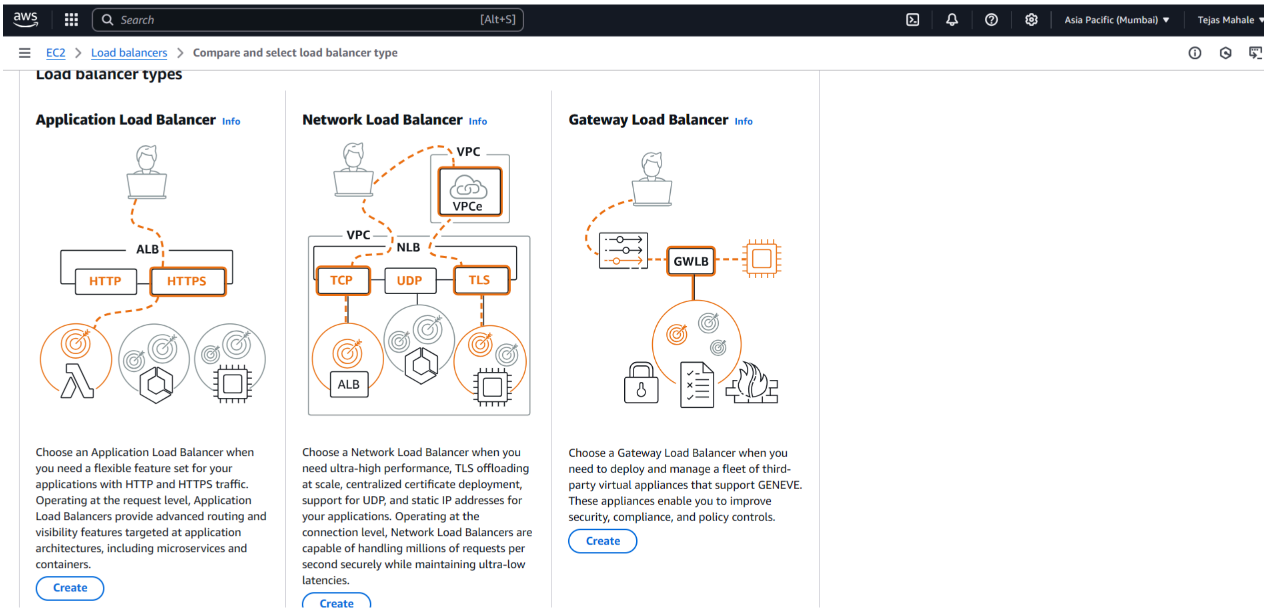
No load balancers

You don't have any load balancers in ap-south-1

Create load balancer

0 load balancers selected

Select a load balancer above.



Application Load Balancer Info

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

Create

Network Load Balancer Info

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

Create

Gateway Load Balancer Info

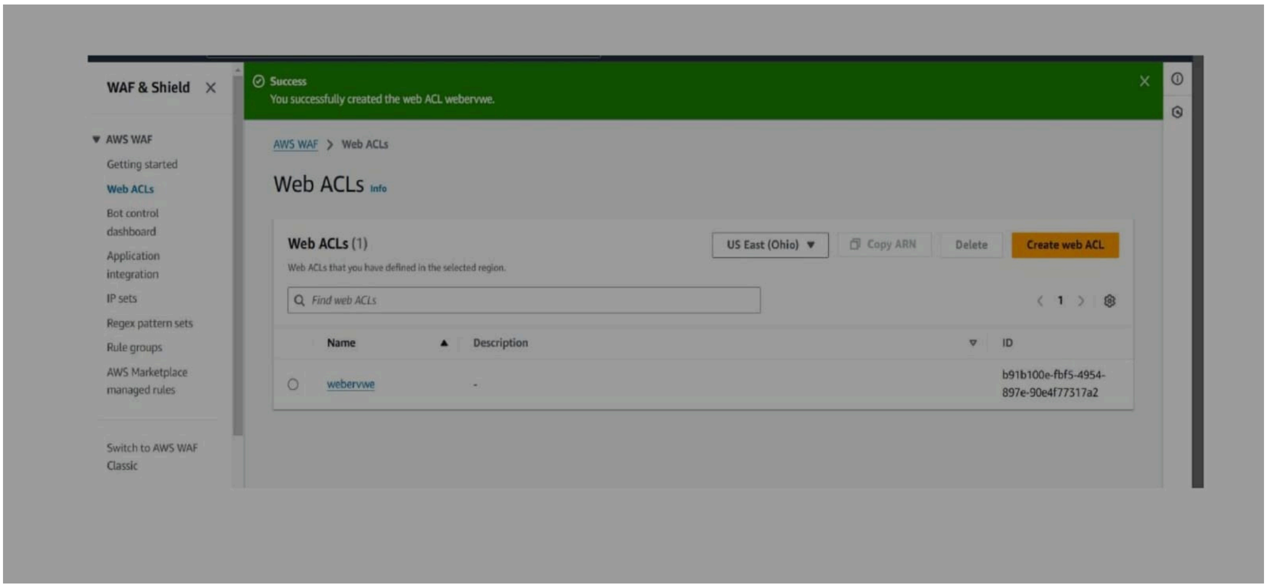
Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

Create

2. Setting Up AWS WAF

1. Open **AWS WAF & Shield** from the AWS console.
2. Click **Create Web ACL**.
3. Choose the same **AWS region** as the ALB.
4. Select **Resource Type: Application Load Balancer**.

5. Click **Next** to proceed with the setup.



Success
You successfully created the web ACL webervwe.

AWS WAF > Web ACLs

Web ACLs Info

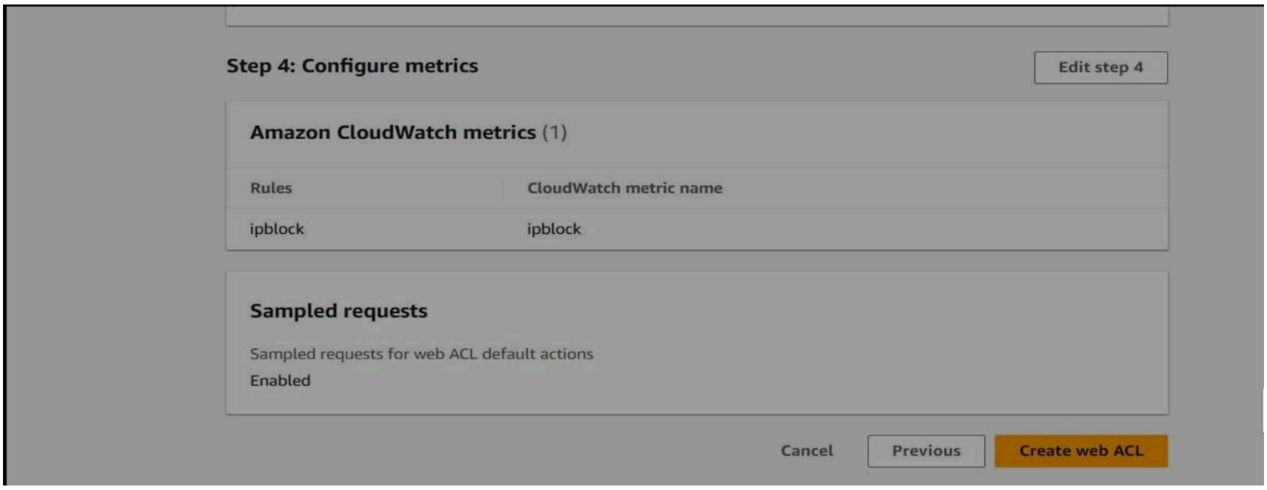
Web ACLs (1)

Web ACLs that you have defined in the selected region.

Name	Description	ID
webervwe	-	b91b100e-fbf5-4954-897e-90e4f77317a2

3. Creating a Rule to Block Specific IP Addresses

1. In the **Rules** section, click **Add rules** → **Create rule**.
2. Provide a **Rule Name** (e.g., "BlockRestrictedIPs").
3. Choose **Rule Type: IP Set**.
4. Click **Create IP Set** and configure:
   - **Name**: RestrictedIPs
   - **Region**: Same as the Web ACL
   - **IP Addresses**: Enter the **IP range to block** (e.g., `203.0.113.0/24`).
5. Save the IP Set and return to the **rule configuration page**.
6. Under **Action**, select **Block**.
7. Click **Save Rule** to apply changes.



Step 4: Configure metrics

Amazon CloudWatch metrics (1)

Rules	CloudWatch metric name
ipblock	ipblock

Sampled requests

Sampled requests for web ACL default actions
Enabled

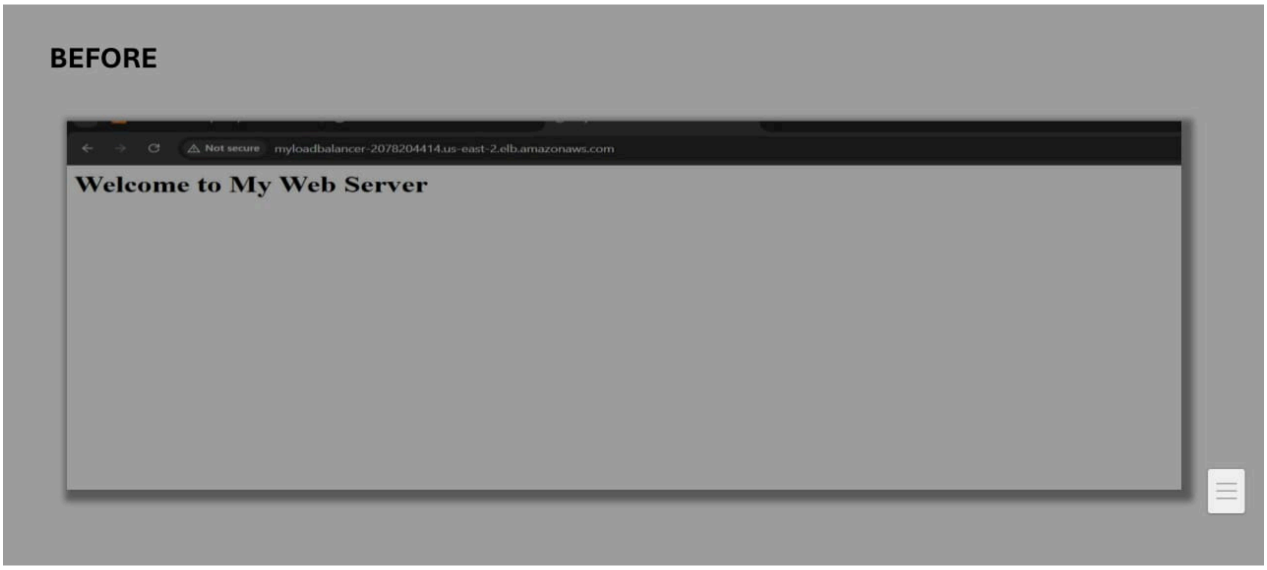
4. Associating WAF with the Load Balancer

1. Navigate back to the **Web ACL settings**.
2. Under **Associated AWS Resources**, select the **Application Load Balancer**.

3. Confirm the selection and click **Save** to apply WAF protection to the ALB.
   ]]

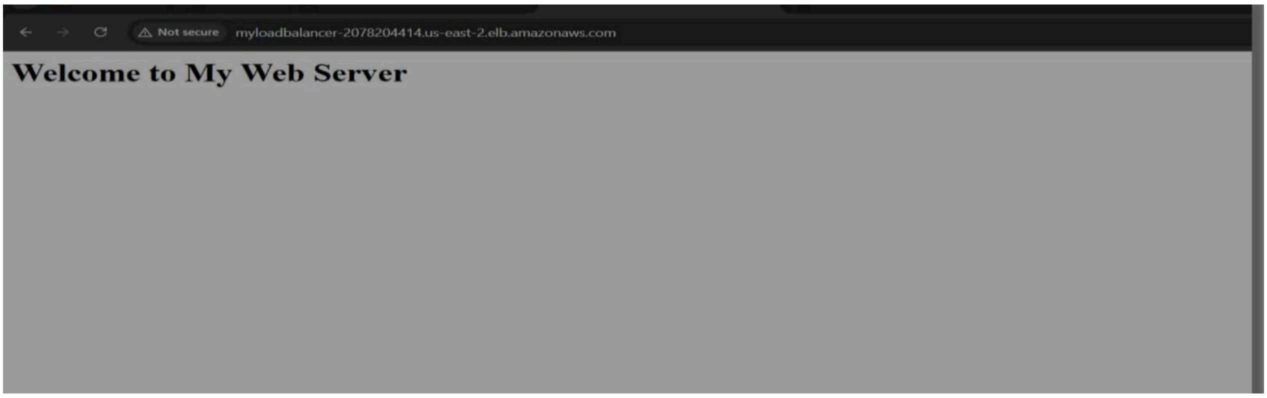
5. Testing the WAF Configuration

1. From a system within the **blocked IP range**, try to access the website via the ALB’s DNS name.
2. The request should fail with an **HTTP 403 Forbidden** error.
3. Test access from an **unblocked IP** to ensure that allowed traffic is processed normally.



Not secure myloadbalancer-2078204414.us-east-2.elb.amazonaws.com

Welcome to My Web Server



Not secure myloadbalancer-2078204414.us-east-2.elb.amazonaws.com

Welcome to My Web Server

Security Benefits of AWS WAF

AWS WAF helps in securing applications by:

- **Blocking malicious traffic** based on custom rules.
- **Preventing unauthorized access** from specific IPs.
- **Protecting against OWASP Top 10 threats**, including SQL injection and cross-site scripting (XSS).
- **Integrating seamlessly with ALB and CloudFront** for scalable protection.

Conclusion

This report demonstrated the implementation of **AWS WAF** to restrict access from a specific IP range and enhance application security. By integrating WAF with **Application Load Balancer (ALB)**, we ensured **traffic filtering, security enforcement, and controlled access**, strengthening the overall security posture of the web application.

Theoretical Questions – AWS Security

1. Role of Security Groups and Firewalls in AWS

What are Security Groups in AWS?

Security Groups act as virtual firewalls that control inbound and outbound traffic for AWS resources, primarily EC2 instances. They define which IP addresses and ports are allowed to access or leave an instance. Security Groups are **stateful**, meaning that if an inbound request is allowed, the corresponding outbound response is automatically permitted.

What are Firewalls in AWS?

AWS does not have a traditional firewall like hardware-based firewalls in on-premises data centers. Instead, it provides a **layered security model** through Security Groups, Network Access Control Lists (NACLs), AWS Web Application Firewall (WAF), and AWS Shield. These work together to filter malicious traffic and protect AWS workloads from unauthorized access and attacks.

Role of Security Groups and Firewalls in AWS Security

- **Security Groups** restrict access to EC2 instances by defining rules for allowed IPs, protocols, and ports.
- **AWS WAF** protects applications from common web-based attacks like SQL injection and cross-site scripting (XSS).
- **AWS Shield** helps mitigate Distributed Denial of Service (DDoS) attacks.
- **Network ACLs (NACLs)** add another layer of network security by controlling traffic at the subnet level.

---

2. How Security Groups and Network ACLs Work in AWS

Security Groups

- Security Groups operate at the instance level and define rules for both **inbound and outbound** traffic.
- They are **stateful**, meaning if an inbound request is allowed, the corresponding outbound response is automatically permitted.
- By default, all inbound traffic is **denied**, and outbound traffic is **allowed**.
- Security Groups allow only **allow rules**; they do not support explicit deny rules.

Network ACLs (NACLs)

- NACLs operate at the **subnet level** in AWS VPC.
- They provide **stateless filtering**, meaning both inbound and outbound rules must be explicitly defined.
- Unlike Security Groups, NACLs support **allow and deny rules**.
- By default, NACLs allow all traffic, but they can be configured to block unwanted access.

Key Differences Between Security Groups and NACLs

Feature	Security Groups	Network ACLs
Level of Control	Instance Level	Subnet Level
Stateful or Stateless	Stateful	Stateless
Supports Deny Rules?	No	Yes
Default Behavior	Denies all inbound, allows all outbound	Allows all traffic unless restricted
Traffic Evaluation	Checks rules when traffic reaches an instance	Evaluates rules before traffic enters/exits a subnet

Comparison with Traditional Firewalls

- **Traditional firewalls** are hardware or software-based security solutions deployed at network perimeters, whereas **AWS security groups and NACLs** provide security within the cloud infrastructure.
- Unlike traditional firewalls that filter traffic for an entire network, **Security Groups focus on individual instances** while **NACLs protect entire subnets**.

- Traditional firewalls often require **manual configuration**, while AWS offers **automated security rule enforcement** through IAM policies and AWS Firewall Manager.

---

3. Incident Response in AWS

What is Incident Response?

Incident response refers to the process of identifying, mitigating, and recovering from security breaches or cyberattacks. In AWS, incident response involves **monitoring, detecting, investigating, and responding** to security threats to minimize damage and ensure compliance.

Key Steps in Incident Response in AWS

1. **Preparation**
    - Define security policies and incident response plans.
    - Set up logging and monitoring using AWS CloudTrail, AWS Config, and Amazon GuardDuty.
2. **Detection & Identification**
    - Use **AWS CloudTrail** to monitor API activity.
    - Enable **Amazon GuardDuty** for threat detection.
    - Implement **AWS Security Hub** to centralize security alerts.
3. **Containment**
    - Isolate affected instances using **Security Groups** and **NACLs**.
    - Revoke compromised credentials using **AWS Identity and Access Management (IAM)**.
    - Block malicious traffic with **AWS WAF**.
4. **Eradication**
    - Identify and remove malicious files, unauthorized changes, or misconfigurations.
    - Perform forensic analysis using **AWS Detective**.
5. **Recovery**
    - Restore affected systems from backups stored in **Amazon S3** or **AWS Backup**.
    - Verify system integrity before bringing services back online.
6. **Post-Incident Review & Improvement**
    - Analyze the root cause using **AWS Security Hub** and **AWS Config**.
    - Update security policies and configurations to prevent future incidents.

AWS Services for Incident Response

AWS Service	Role in Incident Response
AWS CloudTrail	Tracks API activity and detects suspicious operations.
Amazon GuardDuty	Uses machine learning to detect threats in AWS environments.
AWS Security Hub	Aggregates security alerts and findings from multiple sources.
AWS IAM	Manages access permissions and protects against unauthorized access.
AWS WAF	Blocks malicious web traffic.
AWS Shield	Protects against DDoS attacks.
AWS Detective	Assists in security investigations with detailed analysis.

Conclusion

Security in AWS is enforced through **Security Groups, NACLs, and various firewall solutions** that regulate traffic at different levels. These security measures provide **scalability, automation, and flexibility**, making cloud security more dynamic than traditional firewalls. Incident response in AWS is a structured process that involves **continuous monitoring, rapid detection, containment, and mitigation** of security threats using AWS services. By leveraging AWS security tools, organizations can **effectively manage risks and protect cloud workloads** from cyber threats.