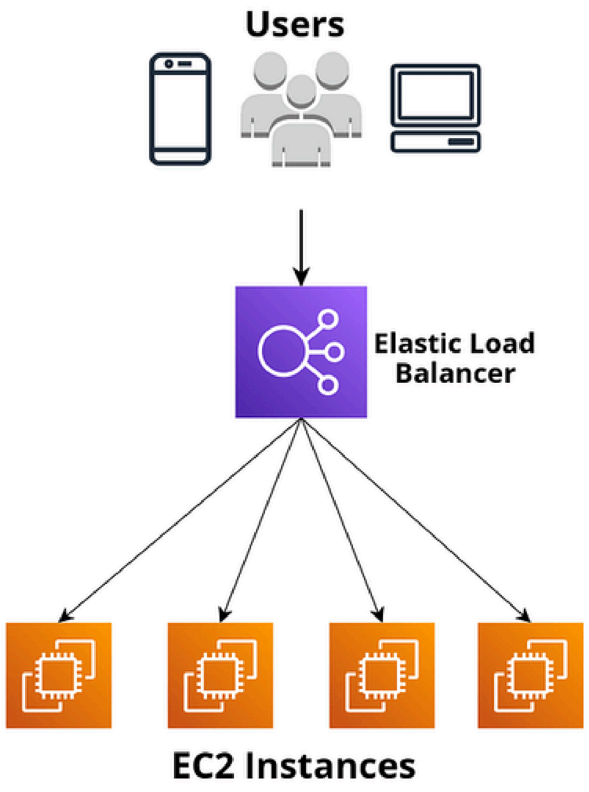
AWS ELB

Elastic Load Balancer & Target Group (TG)

Concept Overview:

About ELB:

Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.



Types of ELB:

There are Four types of Load Balancer.

Application Load Balancer
Routes and load balances at the application layer (HTTP/HTTPS), and supports path-based routing. An Application Load Balancer can route requests to ports on one or more registered targets, such as EC2 instances, in your virtual private cloud (VPC).

Network Load Balancer
Routes and load balances at the transport layer (TCP/UDP Layer-4), based on address information extracted from the Layer-4 header. Network Load Balancers can handle traffic bursts, retain the source IP of the client, and use a fixed IP for the life of the load balancer.

Gateway Load Balancer
Distributes traffic to a fleet of appliance instances. Provides scale, availability, and simplicity for third-party virtual appliances, such as firewalls, intrusion detection and prevention systems, and other appliances. Gateway Load Balancers work with virtual appliances that support the GENEVE protocol. Additional technical integration is required, so make sure to consult the user guide before choosing a Gateway Load Balancer.

Classic Load Balancer
Routes and load balances either at the transport layer (TCP/SSL), or at the application layer (HTTP/HTTPS).

Why & When use ELB:

Why ELB:
A load balancer distributes workloads across multiple compute resources, such as virtual servers. Using a load balancer increases the availability and fault tolerance of your applications.

You can add and remove compute resources from your load balancer as your needs change, without disrupting the overall flow of requests to your applications.

You can configure health checks, which monitor the health of the compute resources, so that the load balancer sends requests only to the healthy ones. You can also offload the work of encryption and decryption to your load balancer so that your compute resources can focus on their main work.

Why & When use ELB:

When to use ELB:

- **Distributing traffic across multiple instances:** Use ELB to automatically balance incoming requests among multiple EC2 instances for better performance.
- **High availability and fault tolerance**: Ideal when you need your application to stay online even if one or more instances fail.
- **Auto Scaling integration:** When using Auto Scaling groups, ELB ensures traffic is routed to newly launched healthy instances.
- **SSL/TLS termination:** Use ELB to manage and offload HTTPS encryption, simplifying certificate management.
- **Multi-AZ deployments:** Best when you want to route traffic to healthy instances across multiple Availability Zones for disaster recovery.

Target Group(TG):

Target Group (TG) is a component of Elastic Load Balancing that routes traffic to one or more registered targets, such as EC2 instances, IP addresses, or Lambda functions, based on specified rules.

Each target group is associated with a protocol and port, and can be linked to an Application Load Balancer (ALB), Network Load Balancer (NLB), or Gateway Load Balancer.

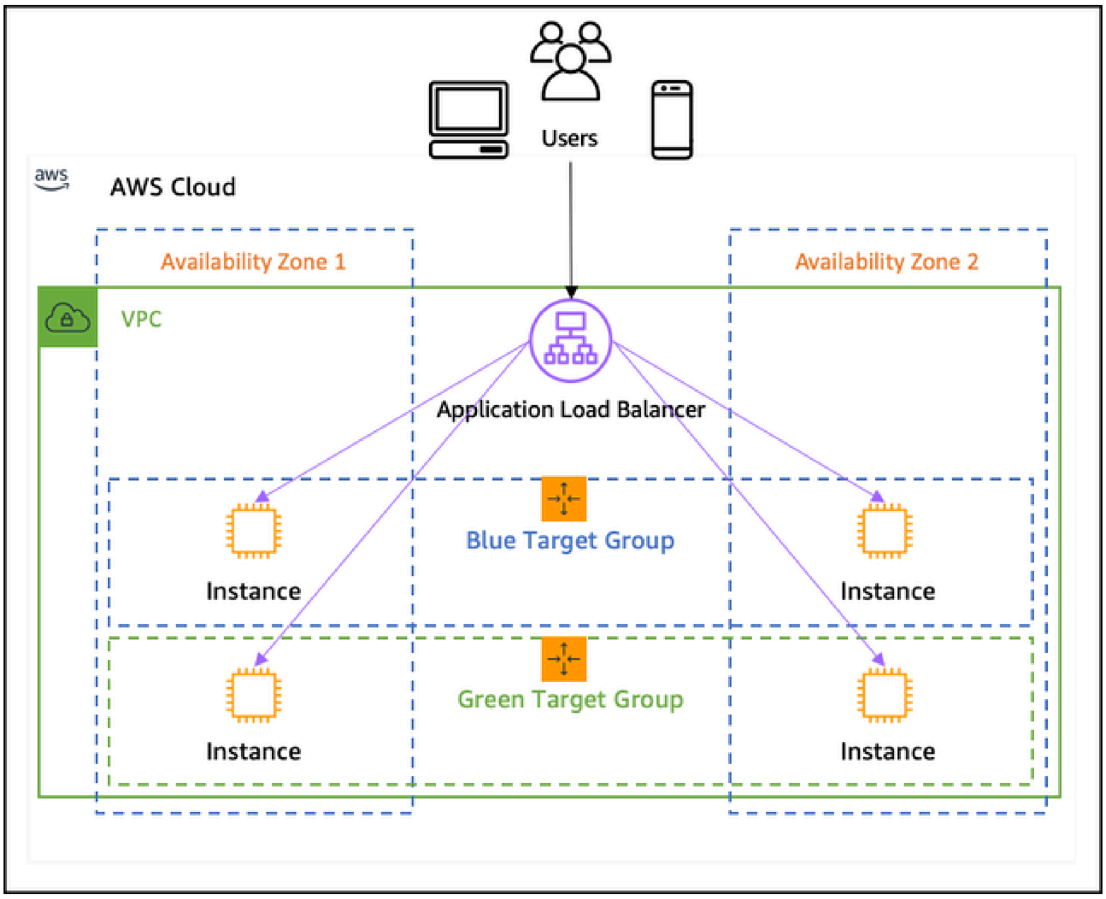
Target groups enable health checks, ensuring traffic only reaches healthy resources, and allow fine-grained traffic routing for microservices or specific application paths.

We choose a target group for ELB because it is required when practicing hands-on. A target group is linked with the Application Load Balancer, which we also use.

Application Load Balancer(ALB):

An Application Load Balancer (ALB) works at Layer 7 of the OSI model, routing HTTP/HTTPS traffic based on rules you define. It evaluates listener rules in order, then sends requests to the right target group. You can route traffic by URL, hostname, or other content, and even use different routing algorithms like round robin or least outstanding requests.

Targets can be added or removed without downtime, and ALB automatically scales with traffic. Health checks ensure requests go only to healthy targets, keeping your application available and responsive.



Hands-on practice:

Before diving into the hands-on practice, you need some prerequisites. You’ll need at least two EC2 instances. For creating them, you can either create a new security group or use an existing one. For better results, configure different servers on each EC2 instance, for example, run Nginx on one and Apache on the other. This will help you clearly see which server is handling the requests.

Here the user-data script for configure server:

Apache:

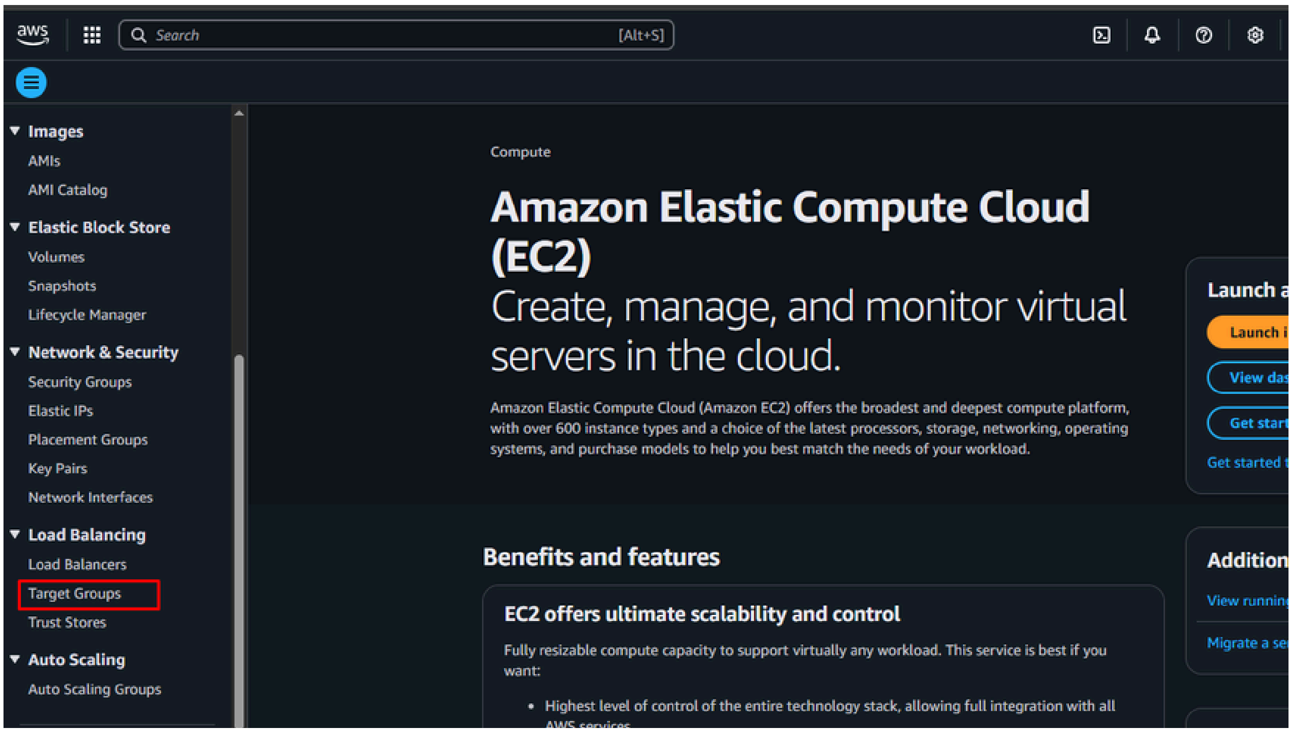
```
#!bin/bash
apt update -y
apt install apache2 -y
systemctl start apache2
systemctl enable apache2
echo "<h1>Apache Server. Hostname is $(hostname -f)
</h1>" > /var/www/html/index.html
```

Nginx:

```
#!/bin/bash
apt update -y
apt install -y nginx
systemctl enable nginx
systemctl start nginx
echo "<h1>Nginx Server. Hostname is $(hostname -f)</h1>"
> /var/www/html/index.html
```

Hands-on practice:

To configure a Load Balancer, we first need to create a Target Group (TG). After configuring it successfully, we can proceed to create the Load Balancer using this TG. To create a TG, you can search for ‘Target Group’ or access it directly from your EC2 dashboard. Here are the instructions:



Amazon Elastic Compute Cloud (EC2)

Create, manage, and monitor virtual servers in the cloud.

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

Benefits and features

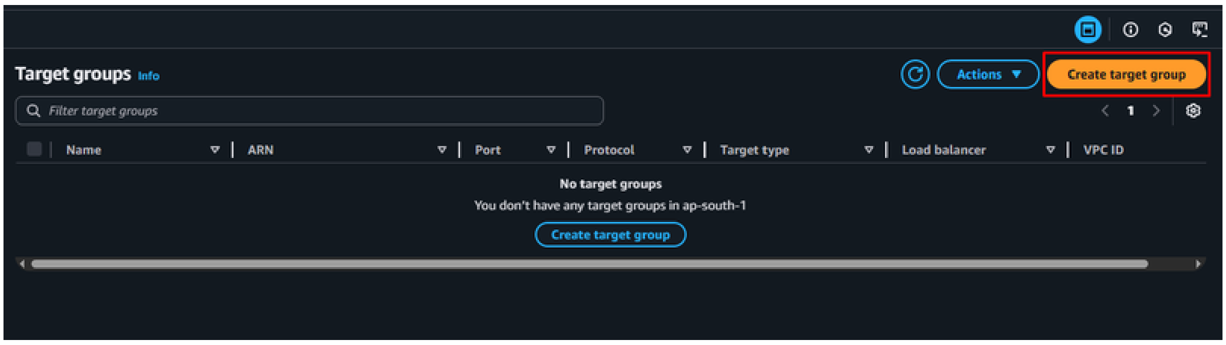
EC2 offers ultimate scalability and control

Fully resizable compute capacity to support virtually any workload. This service is best if you want:

- Highest level of control of the entire technology stack, allowing full integration with all AWS services

Hands-on practice:

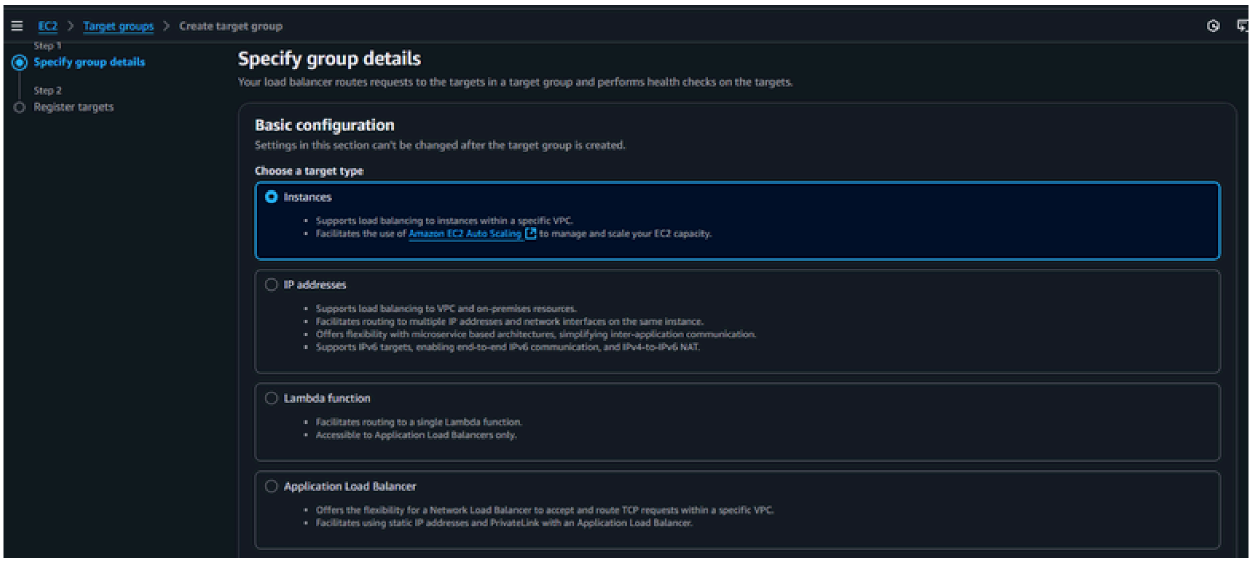
Now, click on Target Groups, get this page. I didn't create any TG before that's why it shows empty.



No target groups

You don't have any target groups in ap-south-1

Click on Create Target Group button and get this configure page.



Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

[✓ Instances]
- Supports load balancing to instances within a specific VPC.
- Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.

IP addresses
- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

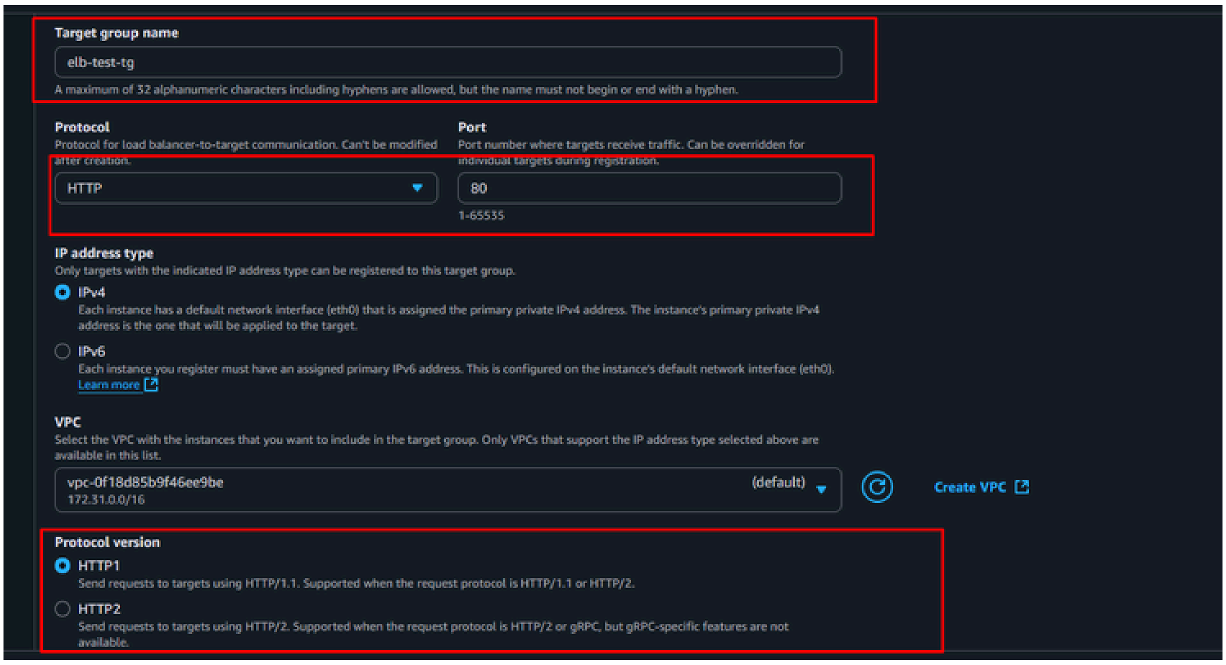
Lambda function
- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer
- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

In my case, i choose an instance because i make TG with EC2 instace.

Hands-on practice:

Now fillup all the configuration based on your needs.



Target group name

elb-test-tg

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

Protocol

Protocol for load balancer-to-target communication. Can't be modified after creation.

HTTP

Port

Port number where targets receive traffic. Can be overridden for individual targets during registration.

80

1-65535

IP address type

Only targets with the indicated IP address type can be registered to this target group.

[✓ IPv4]

Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6

Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

vpc-0f18d85b9f46ee9be (default)
172.31.0.0/16

Create VPC

Protocol version

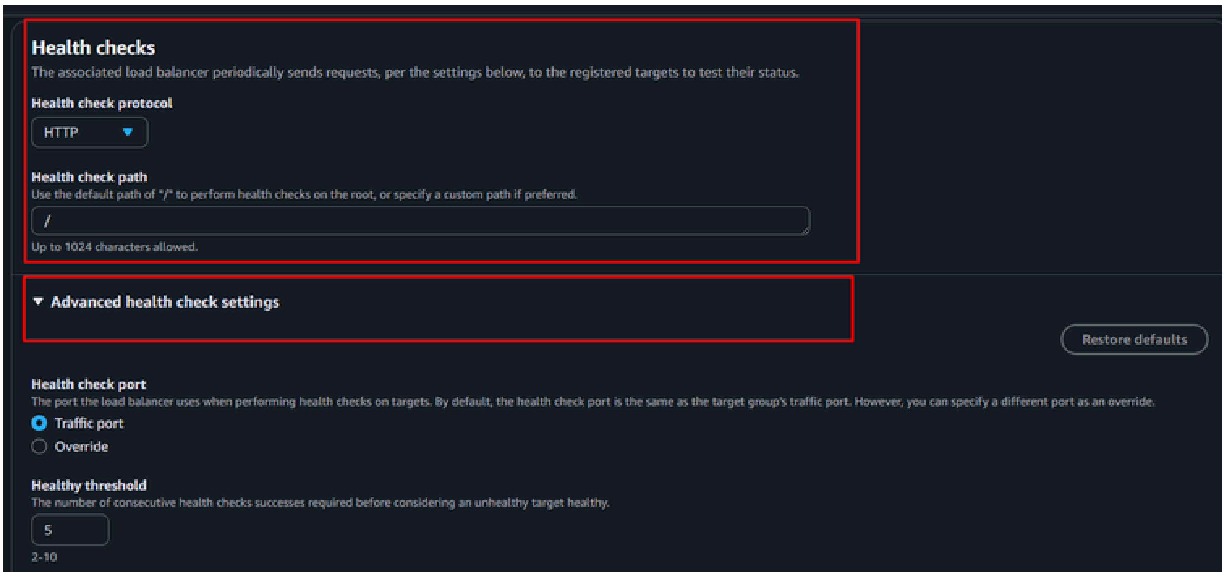
[✓ HTTP1]

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2

Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

If you need advanced configuration for health checking, you can use. I kept all the things as it is.



Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

HTTP

Health check path

Use the default path of "/" to perform health checks on the root, or specify a custom path if preferred.

/

Up to 1024 characters allowed.

Advanced health check settings

Restore defaults

Health check port

The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.

[✓ Traffic port]

Override

Healthy threshold

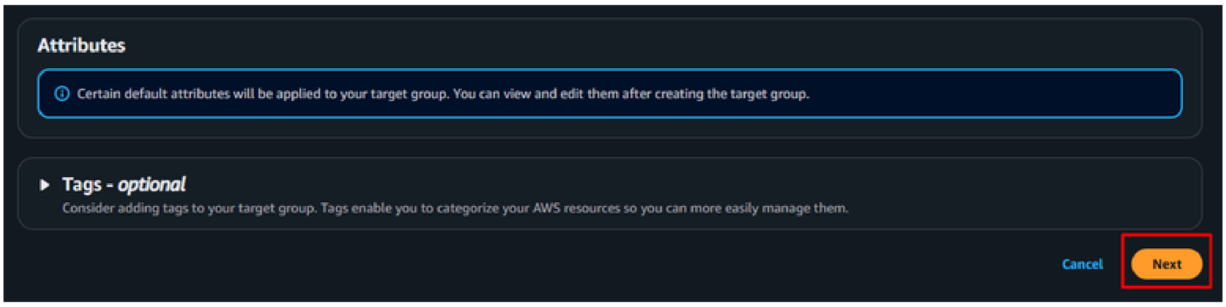
The number of consecutive health checks successes required before considering an unhealthy target healthy.

5

2-10

Hands-on practice:

After complete the first step click on next button get the second step.



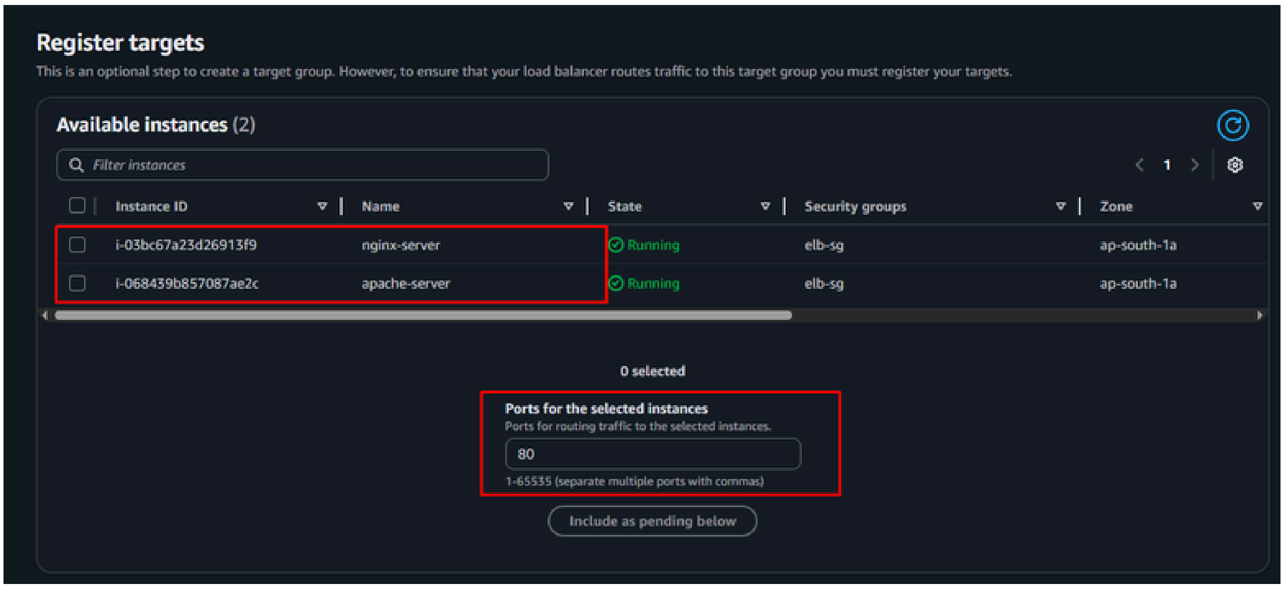
Attributes

Certain default attributes will be applied to your target group. You can view and edit them after creating the target group.

Tags - *optional*

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

Now, register TG with select instances and set the port. And scroll down and click on create target group.



Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (2)

	Instance ID	Name	State	Security groups	Zone
☐	i-03bc67a23d26913f9	nginx-server	Running	elb-sg	ap-south-1a
☐	i-068439b857087ae2c	apache-server	Running	elb-sg	ap-south-1a

0 selected

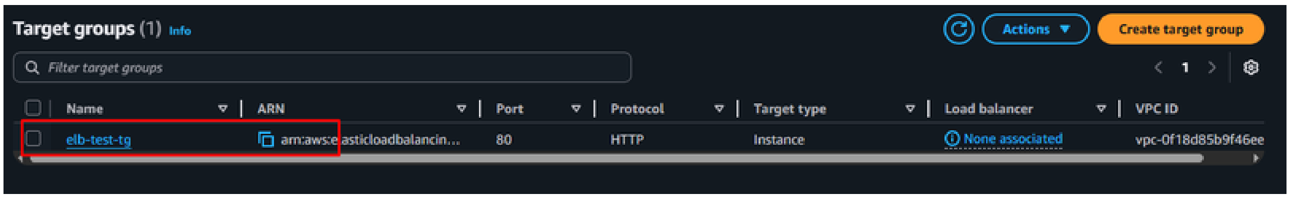
Ports for the selected instances

Ports for routing traffic to the selected instances.

80

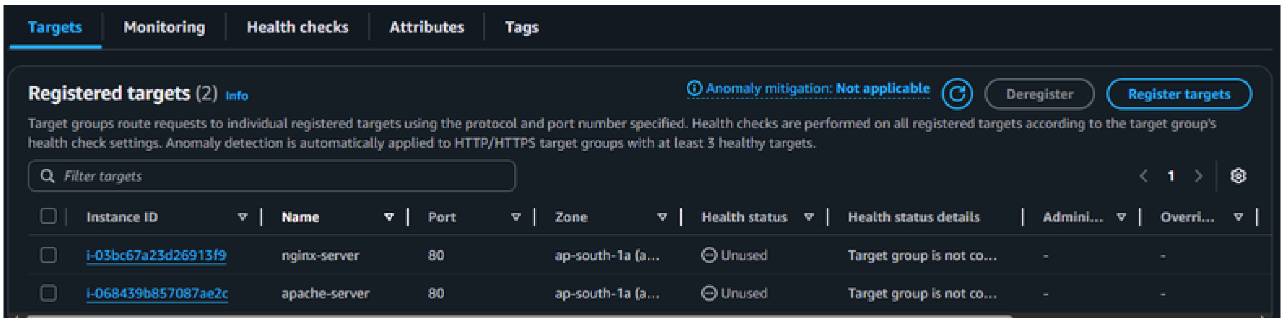
1-65535 (separate multiple ports with commas)

Successfully created TG, here it is.



Target groups (1) Info

	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
☐	elb-test-tg	arn:aws:elasticloadbalancin...	80	HTTP	Instance	None associated	vpc-0f18d85b9f46ee



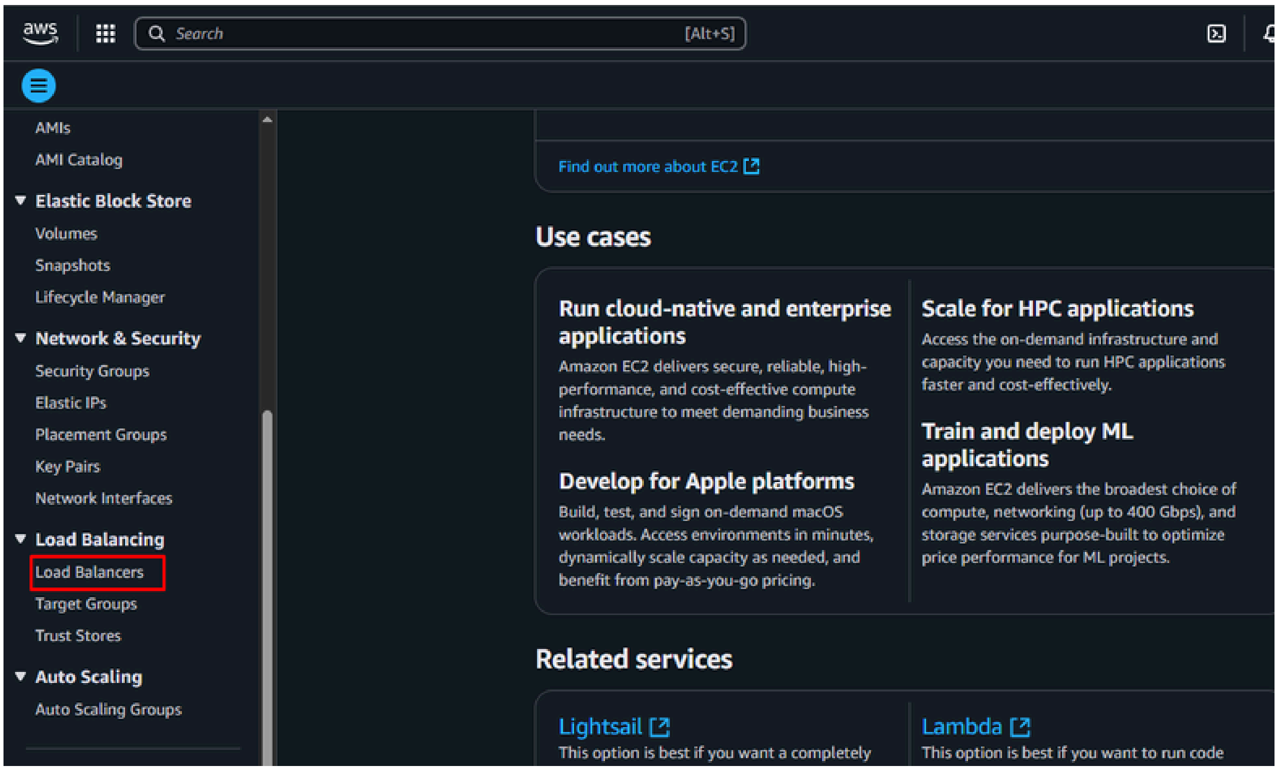
Registered targets (2) Info

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.

	Instance ID	Name	Port	Zone	Health status	Health status details	Admini...	Overri...
☐	i-03bc67a23d26913f9	nginx-server	80	ap-south-1a (a...	Unused	Target group is not co...	-	-
☐	i-068439b857087ae2c	apache-server	80	ap-south-1a (a...	Unused	Target group is not co...	-	-

Hands-on practice:

Now it’s time to create Load balancer, for creating LB click on load balancers.



Use cases

Run cloud-native and enterprise applications
Amazon EC2 delivers secure, reliable, high-performance, and cost-effective compute infrastructure to meet demanding business needs.

Develop for Apple platforms
Build, test, and sign on-demand macOS workloads. Access environments in minutes, dynamically scale capacity as needed, and benefit from pay-as-you-go pricing.

Scale for HPC applications
Access the on-demand infrastructure and capacity you need to run HPC applications faster and cost-effectively.

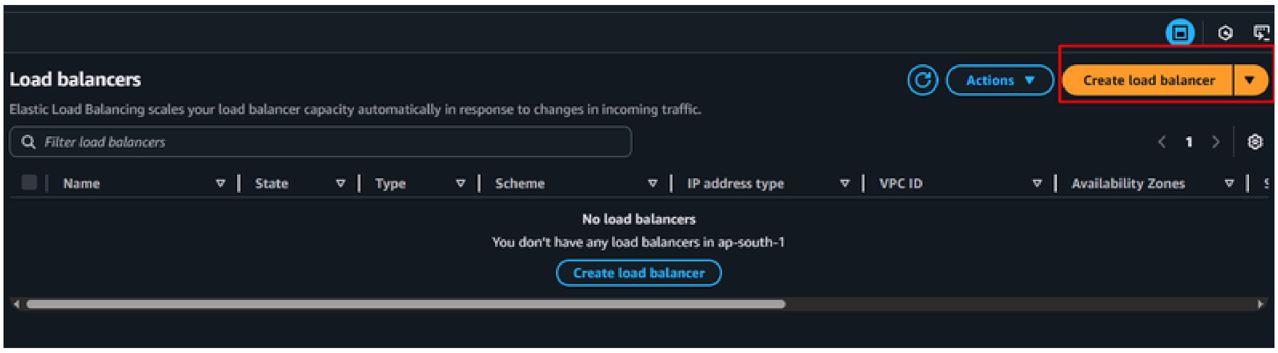
Train and deploy ML applications
Amazon EC2 delivers the broadest choice of compute, networking (up to 400 Gbps), and storage services purpose-built to optimize price performance for ML projects.

Related services

Lightsail
This option is best if you want a completely

Lambda
This option is best if you want to run code

After clicking on Load balancer, you get this page. Now click on Create Load Balancer button.



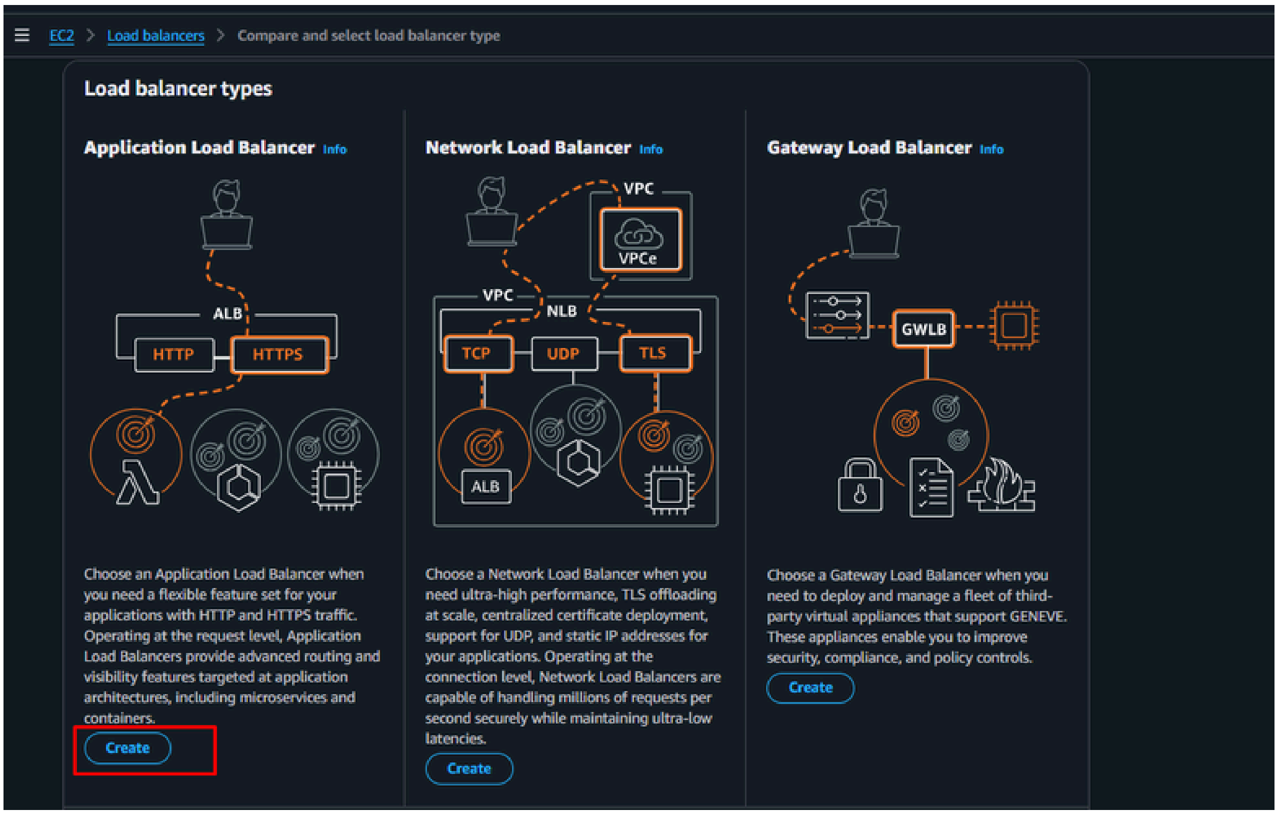
Load balancers
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones

No load balancers
You don't have any load balancers in ap-south-1

Hands-on practice:

Choose the LB, I will use ALB that I discussed before.


☰ EC2 > Load balancers > Compare and select load balancer type
ALB
HTTP
HTTPS
VPC
VPCe
VPC
NLB
TCP
UDP
TLS
ALB
GWLB
Create
Create
Create

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.

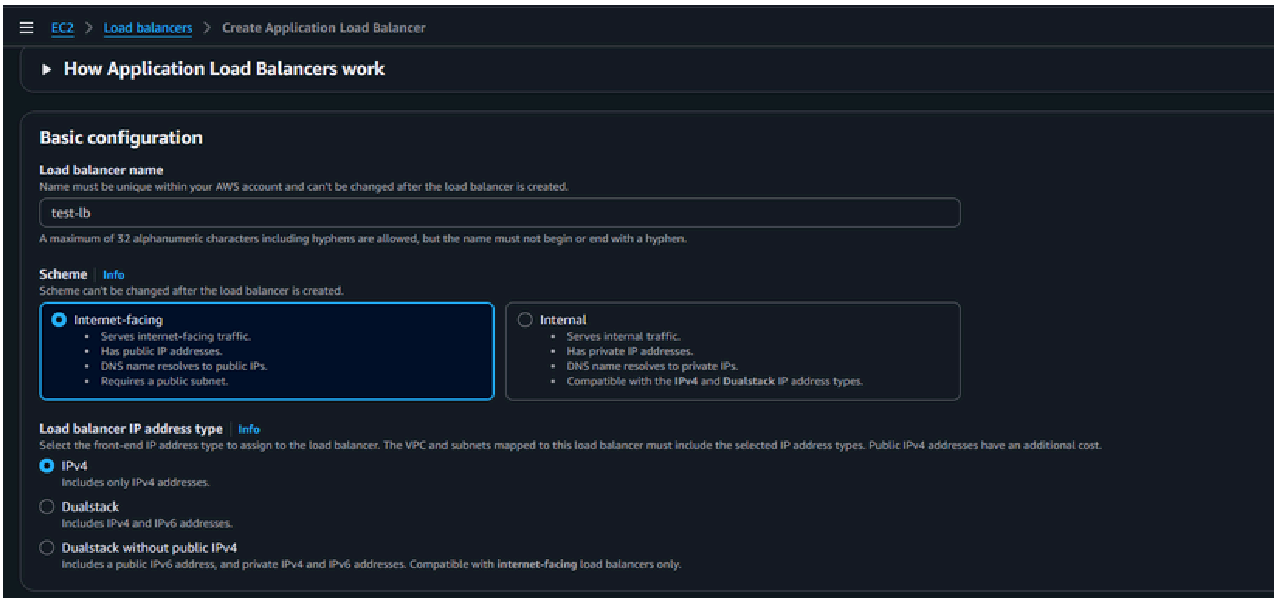
Network Load Balancer Info

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.

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.

Now configure it based on you needs.


☰ EC2 > Load balancers > Create Application Load Balancer
▸ How Application Load Balancers work

Basic configuration

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

test-lb

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 resolves to public IPs.
- Requires a public subnet.

Internal
- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.
- 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.

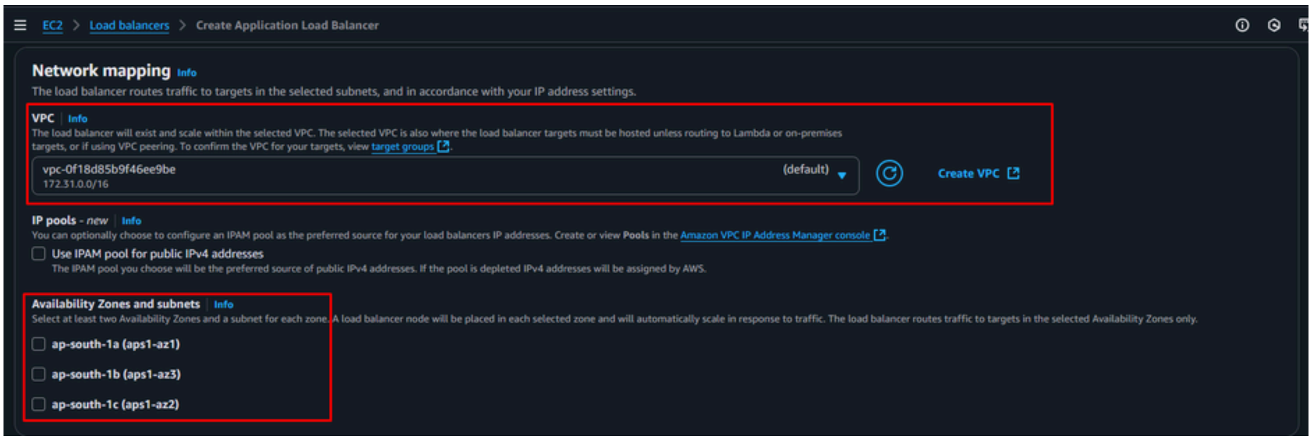
Dualstack
Includes IPv4 and IPv6 addresses.

Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with **internet-facing** load balancers only.

/in/alamgirweb11

Hands-on practice:

In network mapping section, you can specify your custom VPC if you have any, then select the AZ.



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.

vpc-0f18d85b9f46ee9be (default)
172.31.0.0/16

Create VPC

IP pools - *new* | Info

You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view **Pools** in the Amazon VPC IP Address Manager console.

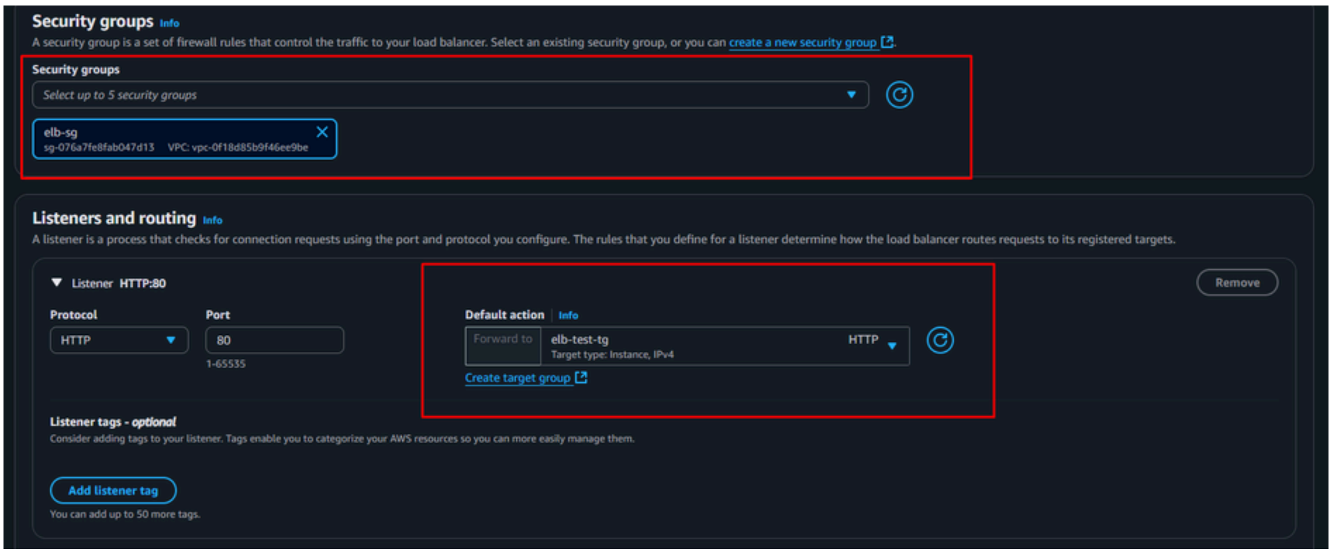
- [ ] Use IPAM pool for public IPv4 addresses
  The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

Availability Zones and subnets | Info

Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

- [ ] ap-south-1a (aps1-az1)
- [ ] ap-south-1b (aps1-az3)
- [ ] ap-south-1c (aps1-az2)

In the Security Group section, you can either add a custom SG or use the default one. In the Listeners and Routing section, select the TG you created earlier.



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

elb-sg
sg-076a7fe8fab047d13 VPC: vpc-0f18d85b9f46ee9be

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**

Remove

Protocol: HTTP

Port: 80
1-65535

Default action | Info

Forward to elb-test-tg HTTP
Target type: Instance, IPv4

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.

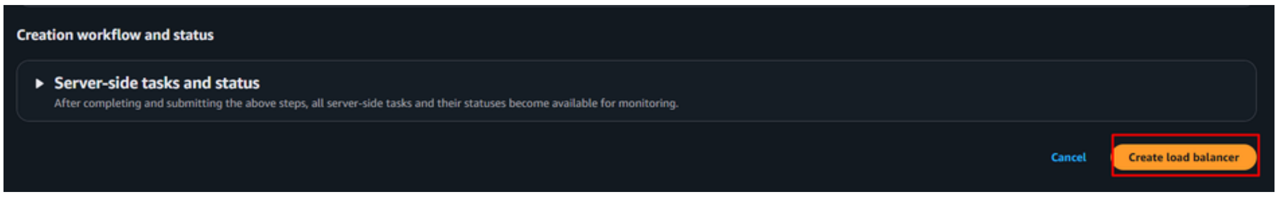
Add listener tag

You can add up to 50 more tags.

/in/alamgirweb11

Hands-on practice:

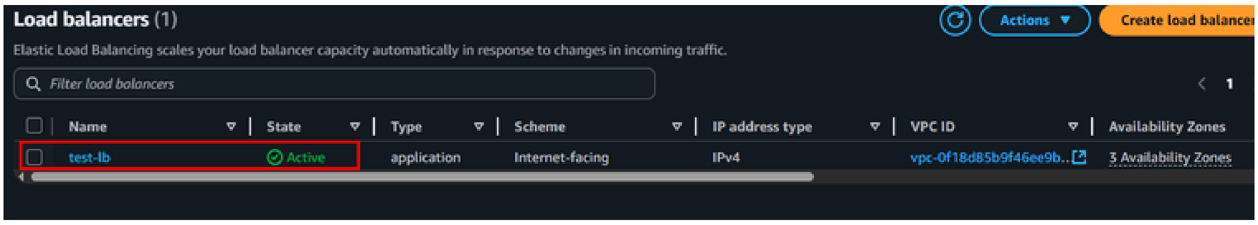
If you need other configurations, you can configure them otherwise, scroll down and click on create load balancer button for create LB.


Creation workflow and status

▸ Server-side tasks and status

After completing and submitting the above steps, all server-side tasks and their statuses become available for monitoring.

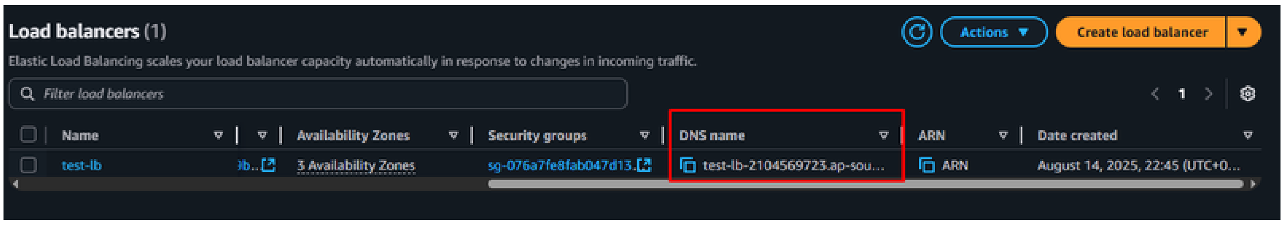
It takes some time to get ready, so please be patient and wait.
Now it's ready to serve.


Load balancers (1)

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

Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones
test-lb	Active	application	Internet-facing	IPv4	vpc-0f18d85b9f46ee9b..	3 Availability Zones

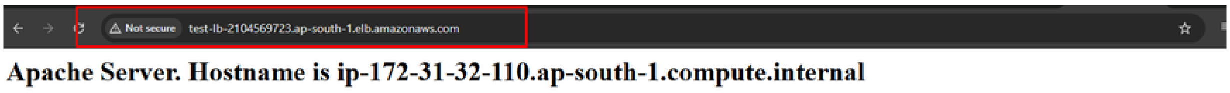
Here the DNS name just copy it and browse from your browser.


Load balancers (1)

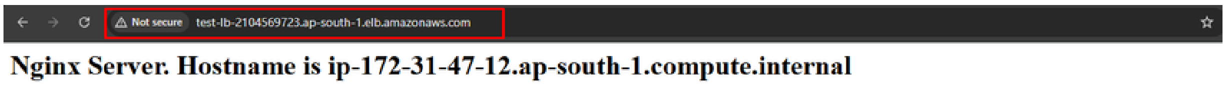
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name		Availability Zones	Security groups	DNS name	ARN	Date created
test-lb	ɘb..	3 Availability Zones	sg-076a7fe8fab047d13.	test-lb-2104569723.ap-sou...	ARN	August 14, 2025, 22:45 (UTC+0...

The Load Balancer has been successfully created. Here are the results:


Not secure test-lb-2104569723.ap-south-1.elb.amazonaws.com

Apache Server. Hostname is ip-172-31-32-110.ap-south-1.compute.internal


Not secure test-lb-2104569723.ap-south-1.elb.amazonaws.com

Nginx Server. Hostname is ip-172-31-47-12.ap-south-1.compute.internal

/in/alamgirweb11

Thank You

Stay Connect:

/in/alamgirweb11

/alamgirweb11

AWS Secret Manager

AWS Secrets Manager helps you manage, retrieve, and rotate database credentials, application credentials, OAuth tokens, API keys, and other secrets throughout their lifecycles. Many AWS services store and use secrets in Secrets Manager.

Secrets Manager helps you improve your security posture, because you no longer need hard-coded credentials in application source code. Storing the credentials in Secrets Manager helps avoid possible compromise by anyone who can inspect your application or the components. You replace hard-coded credentials with a runtime call to the Secrets Manager service to retrieve credentials dynamically when you need them.

With Secrets Manager, you can configure an automatic rotation schedule for your secrets. This enables you to replace long-term secrets with short-term ones, significantly reducing the risk of compromise. Since the credentials are no longer stored with the application, rotating credentials no longer requires updating your applications and deploying changes to application clients.

Key Features

1. **Secure Secret Storage**
    - Secrets are encrypted at rest using AWS KMS keys.
    - By default, Secrets Manager uses an AWS-managed KMS key but you can use customer-managed keys.
2. **Retrieving Secrets**

    - Applications can retrieve secrets securely through the AWS SDK, CLI, or Secrets Manager API.
    - IAM policies and resource-based policies restrict who/what can access secrets.

3. **Automatic Secret Rotation**
    - You can define a rotation schedule (e.g., every 30 days).
    - AWS Secrets Manager integrates with **Lambda functions** to rotate credentials automatically.
    - For supported services (like **RDS, Aurora, Redshift**), AWS provides pre-built Lambda rotation templates.
4. **Audit & Monitoring**
    - Every secret access is logged in **CloudTrail**.
    - Metrics like "number of secrets rotated" can be tracked with **CloudWatch**.

Steps to create a secret (console)

1. Open the Secrets Manager console.
2. Choose **Store a new secret**.
3. On the **Choose secret type** page, do the following:
    a. For **Secret type**, do one of the following:
        - To store database credentials, choose the type of database credentials to store. Then choose the **Database** and then enter the **Credentials**.
        - To store API keys, access tokens, credentials that aren't for databases, choose **Other type of secret**.

In **Key/value pairs**, either enter your secret in JSON **Key/value** pairs, or choose the **Plaintext** tab and enter the secret in any format. You can store up to 65536 bytes in the secret. Some examples:

- API key
- OAuth token
- Digital certificate
- Private key

Enter as key/value pairs:

ClientID : *my_client_id*

ClientSecret : *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

b. For **Encryption key**, choose the AWS KMS key that Secrets Manager uses to encrypt the secret value.

- For most cases, choose **aws/secretsmanager** to use the AWS managed key for Secrets Manager. There is no cost for using this key.
- If you need to access the secret from another AWS account, or if you want to use your own KMS key so that you can rotate it or apply a key policy to it, choose a customer managed key from the list or choose **Add new key** to create one.

You must have Permissions for the KMS key.

c. Choose **Next**.

4. On the **Configure secret** page, do the following:

   a. Enter a descriptive **Secret name** and **Description**. Secret names can contain 1-512 alphanumeric and /_+=.@- characters.

   b. (Optional) In the **Tags** section, add tags to your secret. Don't store sensitive information in tags because they aren't encrypted.

   c. (Optional) In **Resource permissions**, to add a resource policy to your secret, choose **Edit permissions**.

   d. (Optional) In **Replicate secret**, to replicate your secret to another AWS Region, choose **Replicate secret**. You can replicate your secret now or come back and replicate it later.

   e. Choose **Next**.

5. (Optional) On the **Configure rotation** page, you can turn on automatic rotation. You can also keep rotation off for now and then turn it on later. Choose **Next**.

6. On the **Review** page, review your secret details, and then choose **Store**.

Secrets Manager returns to the list of secrets. If your new secret doesn't appear, choose the refresh button.

Steps to update a secret you manage (console)

1. Open the Secrets Manager console.

2. From the list of secrets, choose your secret.

3. On the secret details page, do any of the following:

Note that you can't change the name or ARN of a secret.

- To update the description, in the **Secrets details** section, choose **Actions**, and then choose **Edit description**.
- To update the encryption key.
- To update tags, on the **Tags** tab, choose **Edit tags**.
- To update the secret value.
- To update permissions for your secret, on the **Overview** tab, choose **Edit permissions**.
- To update rotation for your secret, on the **Rotation** tab, choose **Edit rotation**.
- To replicate your secret to other Regions.
- If your secret has replicas, you can change the encryption key for a replica. On the **Replication** tab, select the radio button for the replica, and then on the **Actions** menu, choose **Edit encryption key**.
- To change a secret so that it is managed by another service, you need to recreate the secret in that service.

Steps to delete a secret (console)

1. Open the Secrets Manager console.
2. In the list of secrets, choose the secret you want to delete.
3. In the **Secret details** section, choose **Actions**, and then choose **Delete secret**.
4. In the **Disable secret and schedule deletion** dialog box, in **Waiting period**, enter the number of days to wait before the deletion becomes permanent. Secrets Manager attaches a field

called DeletionDate and sets the field to the current date and time, plus the number of days specified for the recovery window.

5. Choose **Schedule deletion**.

To view deleted secrets

1. Open the Secrets Manager console.
2. On the **Secrets** page, choose **Preferences** (⚙).
3. In the Preferences dialog box, select **Show secrets scheduled for deletion**, and then choose **Save**.

To delete a replica secret

1. Open the Secrets Manager console.
2. Choose the primary secret.
3. In the **Replicate Secret** section, choose the replica secret.
4. From the **Actions** menu, choose **Delete Replica**.

Real-World Use Case

- **E-commerce Website**
  Your web app needs to connect to an **RDS MySQL** database.
  - Without Secrets Manager: you hardcode the username/password in your app → risky, requires manual updates.
  - With Secrets Manager:
    - You store DB credentials in Secrets Manager.
    - Your app retrieves them at runtime.

- Secrets Manager automatically rotates DB passwords every 30 days → updates DB and secret store in sync.
  ✅ **Result**: Zero hardcoded credentials, continuous security, compliance-friendly.

Reference docs: - [What is AWS Secrets Manager? - AWS Secrets Manager]

AWS System Manager

Amazon Systems Manager helps you centrally view, manage, and operate nodes at scale in Amazon, on-premises, and multicloud environments. With the launch of a unified console experience, Systems Manager consolidates various tools to help you complete common node tasks across Amazon Web Services accounts and Amazon Web Services Regions.

Within it, the **SSM Parameter Store** overlaps with Secrets Manager in secret storage.

Key Features

1. **Parameter Store**
   - Store **configuration data** and **secrets** as parameters.
   - Types:
     - **String**: plaintext values.
     - **SecureString**: encrypted using KMS.
2. **Configuration Management**
   - Store app configs like api_endpoint=https://api.example.com.
   - Store secrets like DB password as SecureString.
3. **Integration**
   - Retrieve parameters via SDK, CLI, or environment variables.
   - Works with EC2, ECS, Lambda, CodePipeline, CloudFormation.

4. **Change Management**
   - Versioning → every parameter update creates a new version.
   - Policies → enforce parameter expiration or no overwrite.
5. **Automation & Ops**
   - Run Command → execute scripts across EC2 fleets.
   - Patch Manager → automate patching.
   - Session Manager → secure shell access to EC2 without SSH keys.

How SSM Works

Permissions

- Instances need an **IAM Role** with AmazonSSMManagedInstanceCore policy.
- Users need IAM permissions to run commands, open sessions, etc.

Real-World Use Case

- **Microservices Application**
  Suppose you have multiple microservices running on ECS. Each service needs different configuration values:
  - API keys
  - Endpoints
  - Feature flags
  - Some secrets (DB credentials, tokens)
    You can store everything in **Parameter Store**, and services retrieve only what they need.

Result: Unified config store, secrets encrypted, services stay decoupled.

Reference docs: - What is AWS Systems Manager? - AWS Systems Manager

When to Use Which?

Use AWS Secrets Manager if:

- You need **automatic rotation** of credentials (e.g., DB, API keys).
- Secrets must be managed with high security & compliance (HIPAA, PCI DSS).
- You want built-in integrations (e.g., RDS password rotation).
- Budget allows (since Secrets Manager is more expensive).

Example: Rotating production database credentials automatically every 30 days for a fintech application.

Use AWS Systems Manager Parameter Store if:

- You need a **centralized config + secrets** solution.
- You don't need automatic rotation.
- You want a **cheaper/free** solution.
- You need to store a mix of **parameters, flags, and secrets**.

Example: Storing API endpoints, feature flags, and a few secrets for a microservice app. Developers fetch them securely at runtime.

Best Practice (Hybrid Approach)

In many production environments, companies **use both**:

- Use **Secrets Manager** for **sensitive credentials** (DB, API keys) that require rotation.
- Use **Parameter Store** for **general configs** (URLs, feature flags, non-sensitive values).

Example:
A SaaS company running multi-tenant architecture:

- Stores database credentials per tenant in **Secrets Manager** (with rotation).
- Stores tenant-specific configs (UI theme, API endpoint) in **Parameter Store**.
  This balances **security, cost, and manageability**.