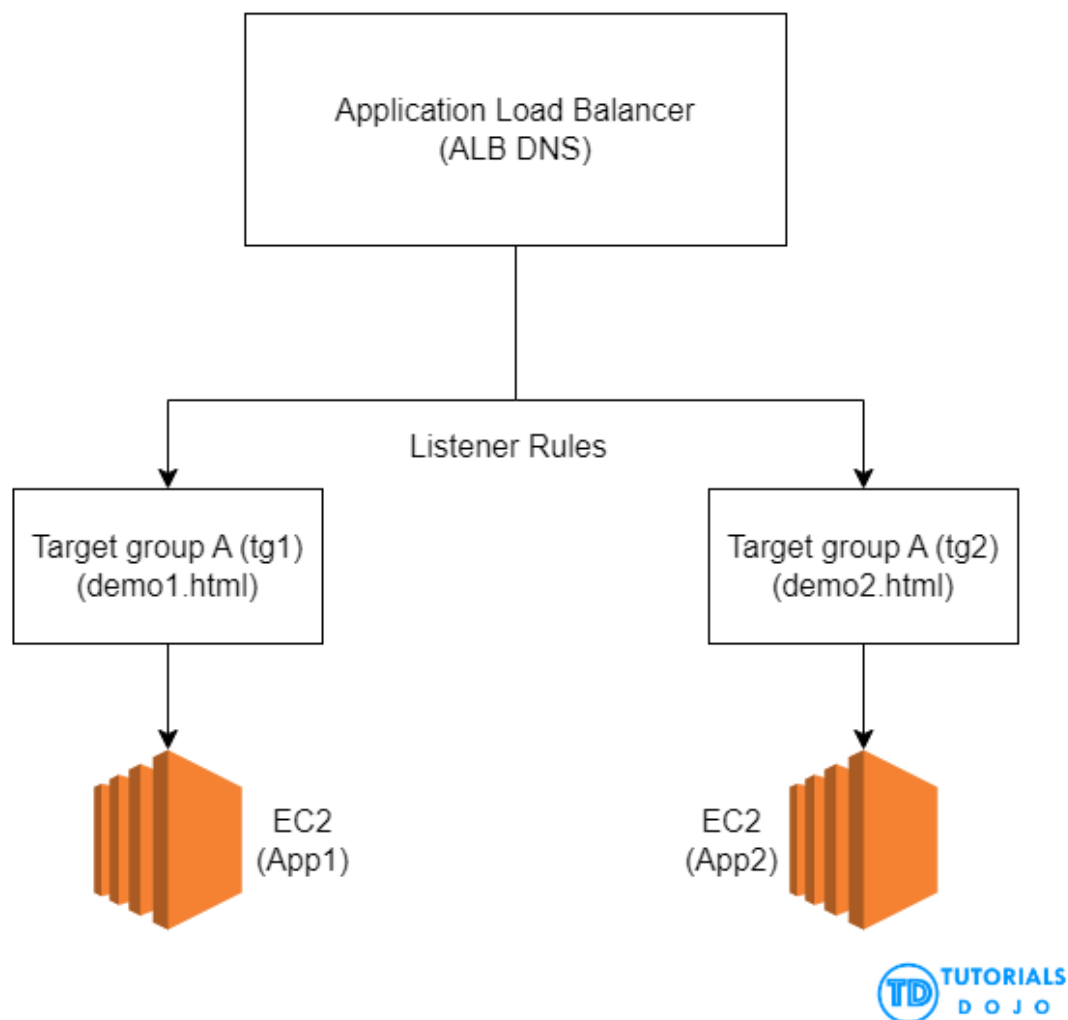


Guided Lab: Path-based routing with Application Load Balancer (AWS ALB)

Description

In this hands-on lab, you'll learn how to configure an Application Load Balancer (ALB) to route incoming requests based on different paths. Path-based routing allows you to host multiple microservices behind a single ALB, directing traffic to the appropriate service based on the requested path.

An Application Load Balancer enables you to set up a listener with rules that direct incoming requests to target groups based on the URL. This capability is unique to Application Load Balancers and is not offered by other load balancer types like Classic Load Balancer, Network Load Balancer, and Gateway Load Balancer. The path pattern rules only apply to the path of the URL and do not consider the URL's query parameters.



Objectives

In this lab, you will learn how to:

- Understand the concept of path-based routing.
- Configure an ALB to route requests based on distinct paths.
- Test the setup using a sample HTML file.

Prerequisites

This lab assumes you have basic knowledge of AWS services (EC2, VPC, ALB).

If you find any gaps in your knowledge, consider taking the following lab:

- Creating an Amazon EC2 instance (Linux)
- Setting up a Web server on an EC2 instance
- Creating Your First Application Load Balancer

Lab Steps

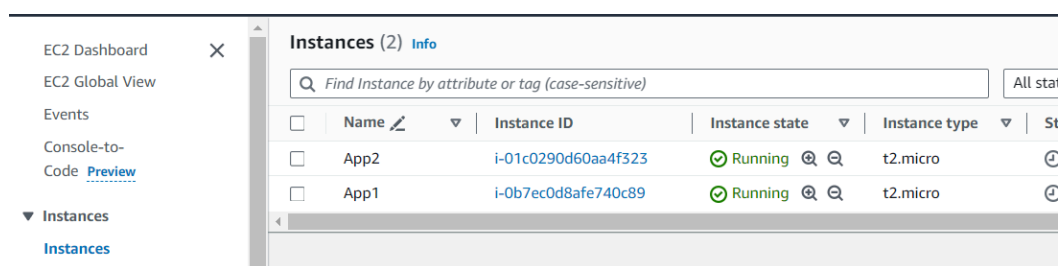
Launching two EC2 Instances:

1. Launch the first EC2 Instances using the following configurations:

- Name: App1
- AMI: Ubuntu Server 24.04
- Key pair: **(Create a new one.)**
 - Key pair name: **path-based-routing**
 - Key pair type: **RSA**
 - Private key file format: **.pem**
- Network settings
 - Allow SSH traffic from My IP
 - Allow HTTP traffic from the internet
- Create the instance.

2. Launch the second EC2 Instance using the following configurations:

- Name: App2
- AMI: Ubuntu Server 24.04
- Key pair: Use the key pair that was created on the first instance.
- Network settings
 - Allow SSH traffic from My IP
 - Allow HTTP traffic from the internet



Installing the NGINX web server

Note: Do these steps on both App1 and App2 instances.

1. Connect to the server via ssh.
2. Install the NGINX web server.

```
sudo apt update
```

```
sudo apt install nginx
```

3. Once installed, open the Public IPv4 address of each instance in the browser using HTTP to verify if the NGINX has been successfully installed.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Creating a simple UI

Note: Do these steps on both App1 and App2 instances.

1. After installing the NGINX web server, create a simple HTML.
2. Connect again to the instance via SSH.
3. Run the following commands:

- For App1 instance:

```
sudo vi /var/www/html/demo1.html
```

- Copy and paste the HTML block below.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Welcome to App1!</title>
```

```
</head>
```

```
<body>
```

```
<h1>Welcome to App1!</h1>
```

```
</body>
```

```
</html>
```

- Save the file using `:wq!` command.

- For App2 instance:

```
sudo vi /var/www/html/demo2.html
```

- Copy and paste the HTML block below.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Welcome to App2!</title>
```

```
</head>
```

```
<body>
```

```
<h1>Welcome to App2!</h1>
```

```
</body>
```

```
</html>
```

- Save the file using `:wq!` command.

Creating Target Groups:

1. Target Type: Instances
2. Set up two target groups (e.g., `tg1` and `tg2`) with Protocol as HTTP and Port as 80.
3. Select the existing VPC
4. Register EC2 instances running App1 with `tg1` (set HealthCheckPath as `/demo1.html`).
5. Register EC2 instances running App2 with `tg2` (set HealthCheckPath as `/demo2.html`).

EC2 > Target groups > Create target group

Step 1
[Specify group details](#)

Step 2
Register targets

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 (1/2)

Filter instances

Instance ID	Name	State	Security groups	Zone
<input type="checkbox"/> i-01c0290d60aa4f323	App2	Running	launch-wizard-2	us-east-1b
<input checked="" type="checkbox"/> i-0b7ec0d8afe740c89	App1	Running	launch-wizard-1	us-east-1b

1 selected

Ports for the selected instances
Ports for routing traffic to the selected instances.

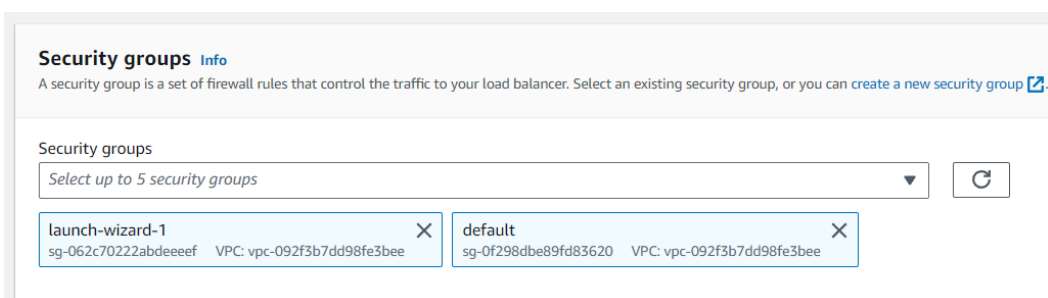
80

1-65535 (separate multiple ports with commas)

Include as pending below

Creating an Application Load Balancer

- **Load Balancer name:** demo
- **Scheme:** Internet-facing
- **Load Balancer IP address type:** IPV4
- **Network mapping**
 - Select the existing VPC
 - Mappings: Select all AZs
- **Security groups**
 - Select the default and the launch-wizard-1



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

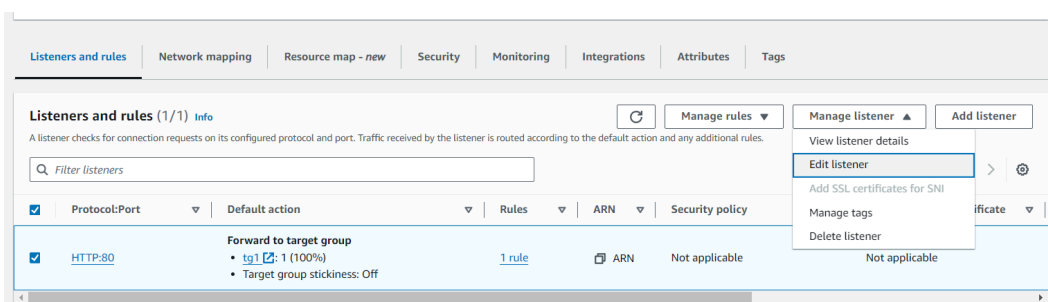
Select up to 5 security groups

launch-wizard-1
sg-062c70222abdeeeef VPC: vpc-092f3b7dd98fe3bee

default
sg-0f298dbe89fd83620 VPC: vpc-092f3b7dd98fe3bee

- **Listeners and routing**
 - Protocol: HTTP
 - Port: 80
 - In the meantime, forward it to tg1 because there is no path condition upon creating the ALB.
 - Click **Create Load Balancer**.

Configure Listener and rules:



Listeners and rules (1/1) [Info](#)

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

Filter listeners

<input checked="" type="checkbox"/>	Protocol:Port	Default action	Rules	ARN	Security policy
<input checked="" type="checkbox"/>	HTTP:80	Forward to target group <ul style="list-style-type: none">tg1 (100%)Target group stickiness: Off	1 rule	ARN	Not applicable

Manage listener

- View listener details
- Edit listener
- Add SSL certificates for SNI
- Manage tags
- Delete listener

- Go to the **Listener details** pane.
 - Change the **Routing actions** to **Return fixed response**
 - Click **Save Changes**.

Listener details

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.

Listener ARN

 arn:aws:elasticloadbalancing:us-east-1:975050133651:listener/app/demo/6872bb49c91361da/466e8fe28f9aa375

Listener configuration

The listener will be identified by the protocol and port.

Protocol

Used for connections from clients to the load balancer.

HTTP

Port

The port on which the load balancer is listening for connections.

80

1-65535

Default actions [Info](#)

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

Routing actions

☐ Forward to target groups

☐ Redirect to URL

☒ Return fixed response

Return fixed response [Info](#)

Use fixed-response actions to drop client requests and return a custom HTTP response. When a fixed-response action is taken, the action and the URL of the redirect target are recorded in the access logs.

Response code

The type of message you want to send.

503

2xx, 4xx, 5xx

Content type - *optional*

The format of your message.

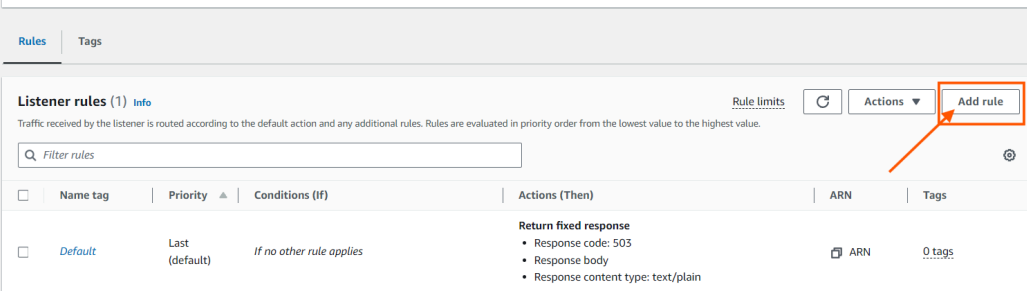
text/plain

Response body - *optional*

Enter your response message.


- Define listener rules based on path patterns:

- Click **Add rule** → **Next**



Listener rules (1) [Info](#) [Rule limits](#) [Actions](#) [Add rule](#)

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

<input type="checkbox"/>	Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
<input type="checkbox"/>	Default	Last (default)	If no other rule applies	Return fixed response <ul style="list-style-type: none">Response code: 503Response bodyResponse content type: text/plain	 ARN	0 tags

- Add condition

- Rule condition type: Path
- For requests to `/demo1.html`, forward to `tg1`.
- For requests to `/demo2.html`, forward to `tg2`.
- Click Next.

EC2 > Load balancers > demo > HTTP:80 listener > Add rule

Step 1
[Add rule](#)

Step 2
[Define rule conditions](#)

Step 3
Define rule actions

Step 4
[Set rule priority](#)

Step 5
[Review and create](#)

Define rule actions [Info](#)

These actions will be applied to requests matching the rule conditions.

► **Listener details:** HTTP:80

Actions

Action types

Routing actions

☒ 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](#)

Target group	HTTP	Weight	Percent
tg1 Target type: Instance, IPv4	HTTP	1	100%

0-999

[Add target group](#)

You can add up to 4 more target groups.

Target group stickiness [Info](#)
Stickiness enables the load balancer to bind a user's session to a specific target group. To use stickiness the client must support cookies.

☐ Turn on target group stickiness

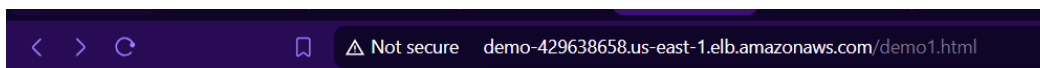
Cancel Previous **Next**

- Rule Priority
 - First rule (tg1): 1
 - Second rule (tg2): 2

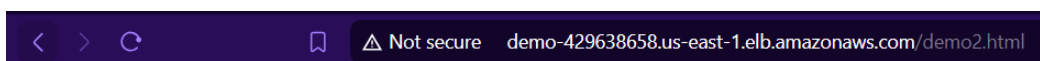
Test Path-Based Routing:

Access your ALB using different paths (e.g., `http://your-alb-dns-name/demo1.html` and `http://your-alb-dns-name/demo2.html`).

- Verify that requests are routed correctly to the respective pages.



Welcome to App1!



Welcome to App2!

Congratulations! You've successfully set up path-based routing with an Application Load Balancer (ALB) on AWS. By configuring listener rules based on URL paths, you've efficiently directed requests to

different instances behind a single ALB. Whether it's

`/demo1.html/` or `/demo2.html/`, your ALB knows where to route the traffic.
