## Darshan Nikam Date: 04/03/2024

### **Load Balancing**

The Load Balancer is a special service that sits between your users and your servers. When someone tries to access your website or application, their request goes to the load balancer first. The load balancer then decides which server or instance should handle the request. It does this by distributing the incoming requests evenly among the available servers.

AWS load balancing helps you manage **incoming traffic** to your website or application by **distributing it across multiple servers**, ensuring better **performance, scalability, and reliability.**

**Types:**

|  |  |
| --- | --- |
| 1) Classic Load Balancer (CLB):   * + Traditional load balancer is an old service.   + Operates at request and connection levels.   + Balances traffic across EC2 instances in multiple AZs.   + Supports both internet-facing and internal load balancing.   + Basic load balancing without advanced features. | * 3) Network Load Balancer (NLB):   + Operates at Layer 4 (transport layer).   + Handles millions of requests with low latency.   + Supports TCP/UDP traffic.   + Ideal for high-performance and static IP needs.   + Automatically scales to handle high traffic. |
| * 2) Application Load Balancer (ALB):   + Operates at Layer 7 (application layer).   + Routes HTTP/HTTPS traffic based on content.   + Supports host-based and path-based routing.   + Ideal for modern web apps and microservices.   + Integrates with container services like ECS and EKS. | * 4) Gateway Load Balancer (GWLB):   + Works with VPC traffic.   + Distributes ingress traffic across multiple targets.   + Supports TLS termination and centralized security.   + Acts as a single entry point for VPC services.   + Integrates with AWS Network Firewall for enhanced security. |

**Target Groups**

Target groups are an essential component of the Application Load Balancer (ALB) and Network Load Balancer (NLB) in AWS. They serve as a logical grouping of targets, such as EC2 instances, containers, IP addresses, or Lambda functions, that receive incoming traffic from the load balancer.

Functionality:

* + When a load balancer receives a request, it forwards the request to one or more target groups based on routing rules.
  + Each target group maintains health checks to ensure that only healthy targets receive traffic.
  + Target groups support both instance-level and IP-level health checks.
  + They can be associated with multiple listeners on the same load balancer.

Features:

* + Health checks: Target groups periodically check the health of targets to ensure they are capable of handling traffic.
  + Sticky sessions: ALB supports sticky sessions, allowing the load balancer to direct requests from the same client to the same target.
  + Weighted routing: NLB supports weighted routing, enabling traffic distribution based on weights assigned to targets within a target group.
  + Path-based routing: ALB supports path-based routing, allowing different paths of a URL to be routed to different target groups.

Use Cases:

* + Target groups are used to distribute incoming traffic to multiple targets, improving availability and fault tolerance.
  + They are essential for scaling applications horizontally by adding or removing targets dynamically.
  + Target groups are used in conjunction with auto-scaling groups to automatically register and deregister instances based on demand.

Now we Perform Application load balancer Practical.

1. We launch a total of 9 Instances with a pre-configured HTTPD package using the shell script, write into user data option.
2. The 4 instances for a Home page, 3 instances for Product list pages, and 2 instances for a Price list page.
3. We will give the default path [/var/www/html/index.html] to the Home page.
4. For the product list page we will create a new directory in [/var/www/html/product], and the same as for the price list. These directories are our HTML index pages.
5. We need to add text to these pages to identify it, with text we add the variable $HOSTNAME to verify load balancer is working or not.
6. Then we create an Application load balancer. With the same VPC in and mapping availability zone where our Instances.
7. Select the security group that has HTTP traffic allowed
8. Then create target groups 1st is home in the default path, 2nd is Product in the product directory path, and 3rd is the price in the price directory.
9. Then attach these target groups to the load balancer in the add Listener option.

After all steps are done we copy the load balancer DNS link, paste it into the new browser, and check it by refreshing,

And giving the path of pages. If the IP address changes after refreshing the page that means our load balancer is successfully working.

Now we start

Note: Must allow HTTP traffic for all instances. And are in the same VPC.

Step 1) Launch 4 Instances for the Home page by writing the below shell script in the user data option.

#!/bin/bash

sudo -i

yum install httpd -y

systemctl start httpd

systemctl enable httpd

echo “Hello everyone, this Home Page of the website has a $HOSTNAME” > /var/www/html/index.html

Step 2) Launch 3 Instances for the Product list page by writing the below shell script in the user data option.

#!/bin/bash

sudo -i

yum install httpd -y

systemctl start httpd

systemctl enable httpd

mkdir /var/www/html/product

echo “Hello everyone, this Product List Page of the website has a $HOSTNAME” > /var/www/html/product/index.html

Step 3) ) Launch 2 Instances for the Price list page by writing the below shell script in the user data option.

#!/bin/bash

sudo -i

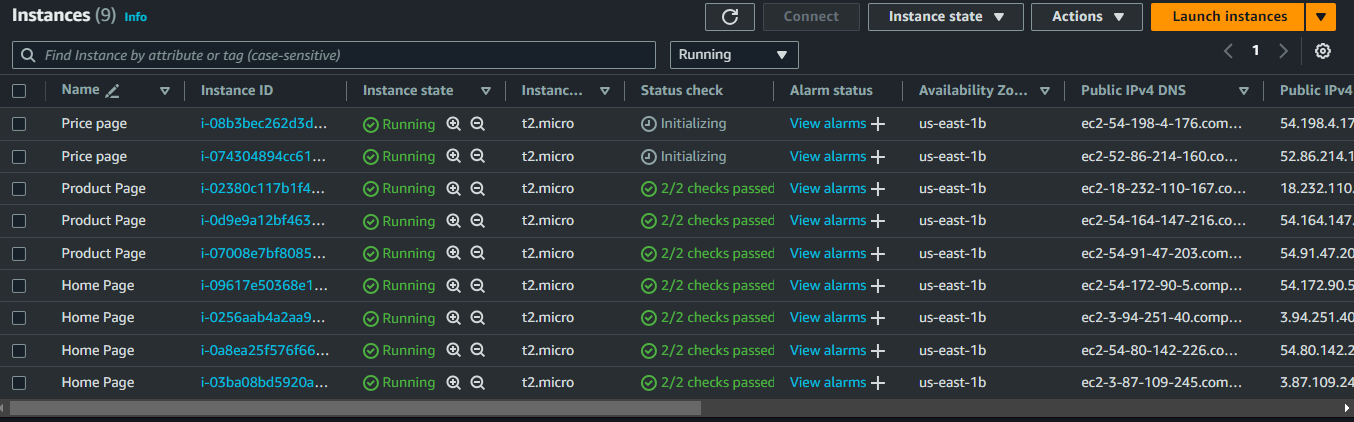
yum install httpd -y

systemctl start httpd

systemctl enable httpd

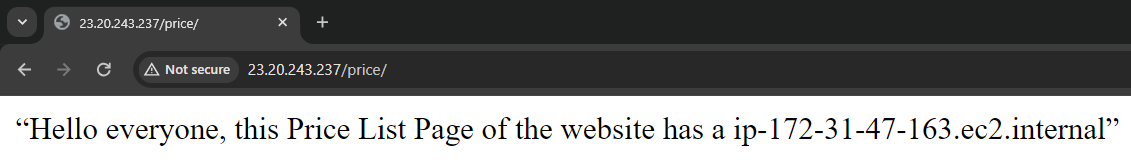
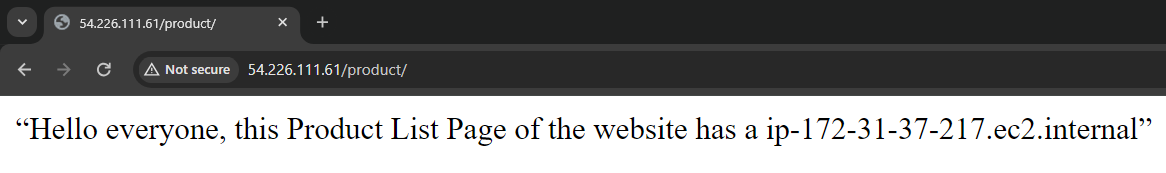
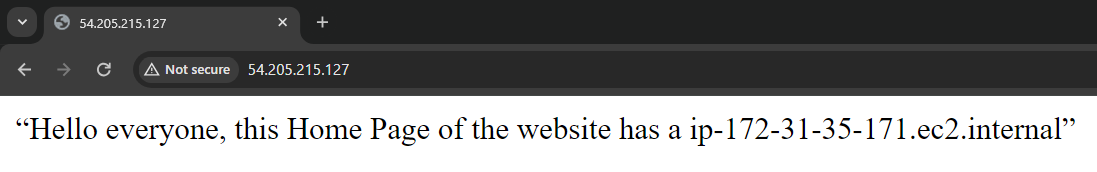
mkdir /var/www/html/price

echo “Hello everyone, this Price List Page of the website has a $HOSTNAME” > /var/www/html/price/index.html



Step 4) After Launching instances, copy their Public IP address, paste it into the browser, and check web page is running or not. If it's running everything is correct or not then check the shell scripting or HTTP traffic.

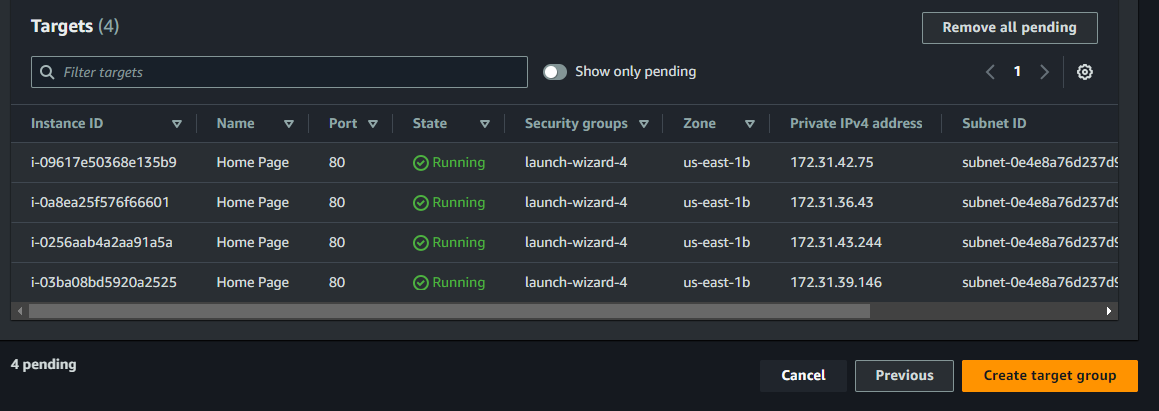
Now we will see our web pages are running.



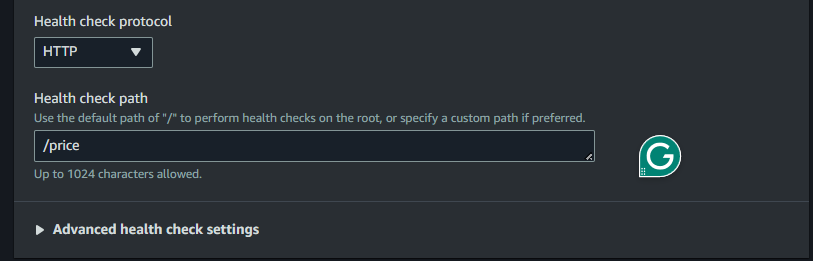
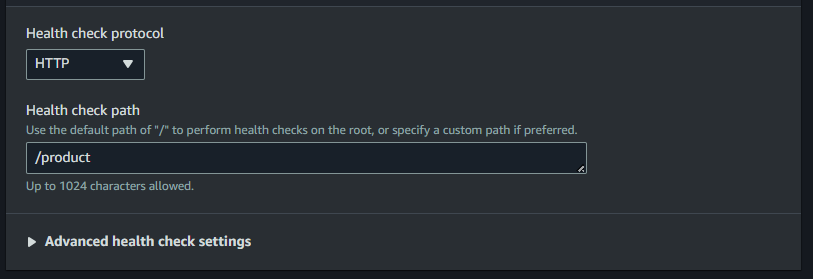
Now we create Target groups

Step 5) Go to Load Balancing Menu in Ec2 dashboard, Select the Target Groups option and click on Create Target Group button.

Step 6) Our target type is Ec2 Instance, type target group name rest all options as by default, click on next button, and select your instance that you created for home page then click Includes as pending below button, and click on Create Target Group button.

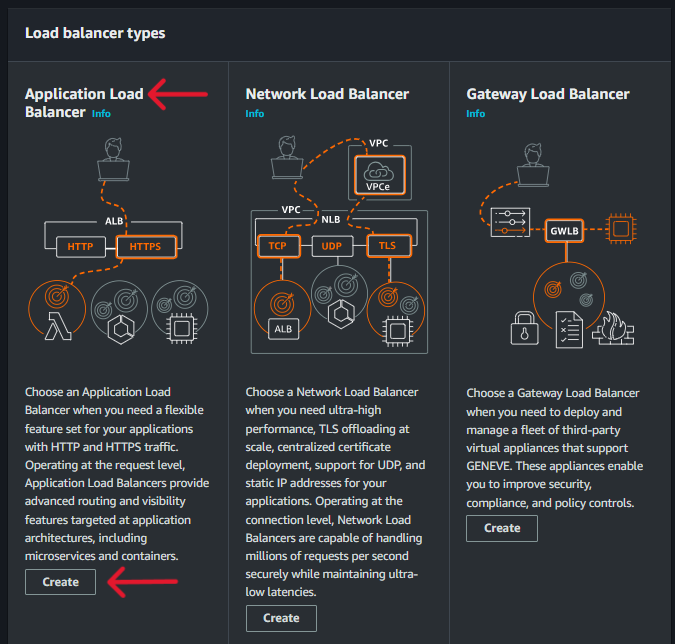


As same creates target groups for Product and Price list Instances but in the above steps we gave the default path for the Home page, now we need to give that path where we create our Product and Price List.

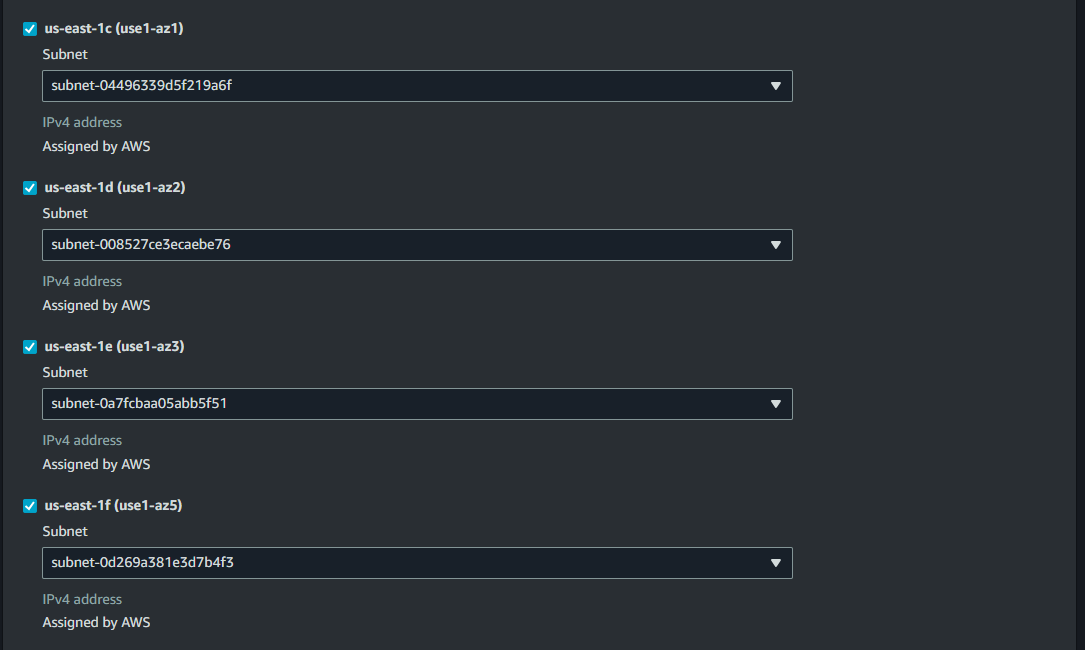
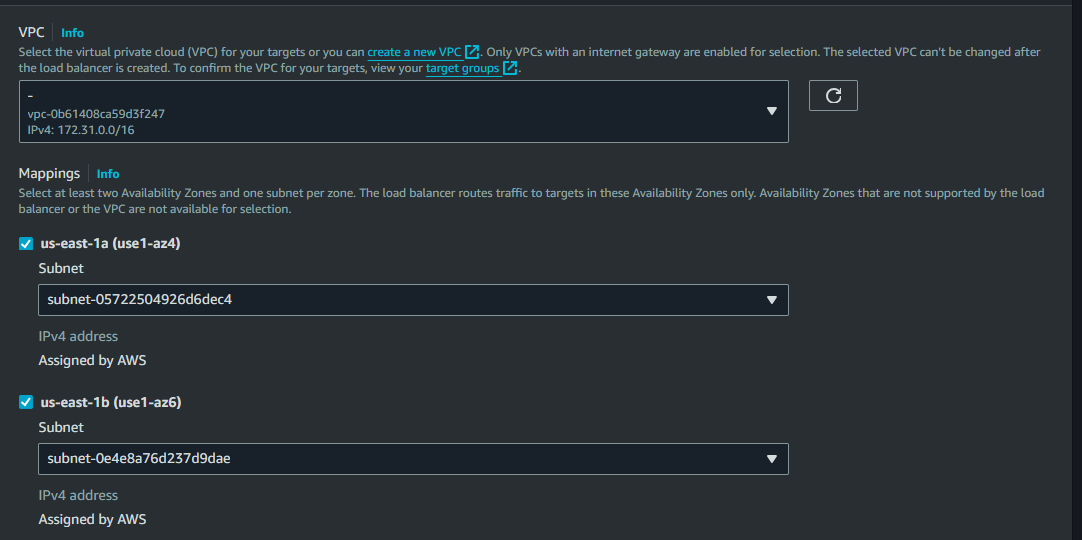


Our Target Groups are created now we create an Application load Balancer and attach these target groups to it.

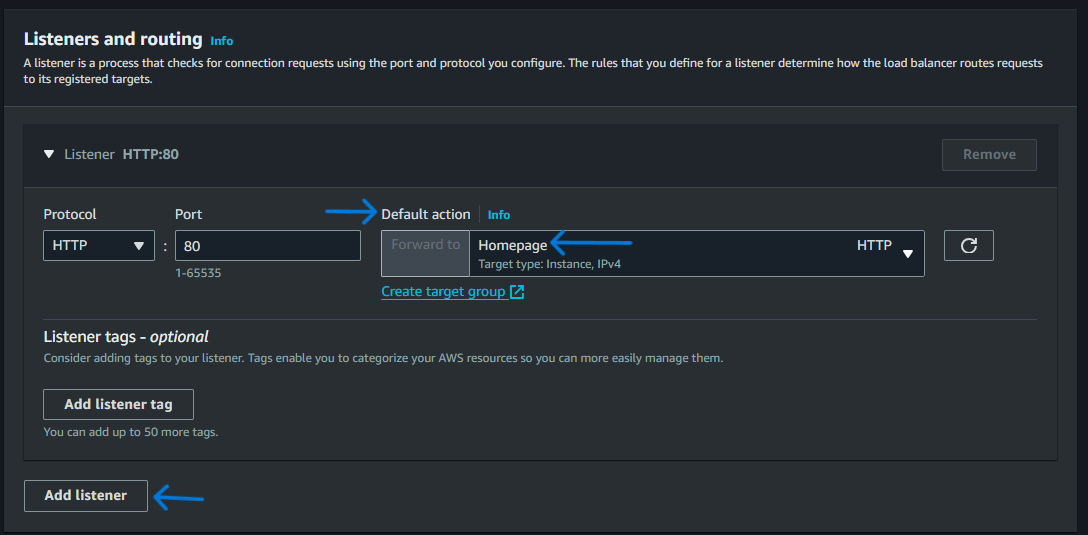
Step 7) Go to the Create load balancer option and Click on the create button of Application Load Balancer



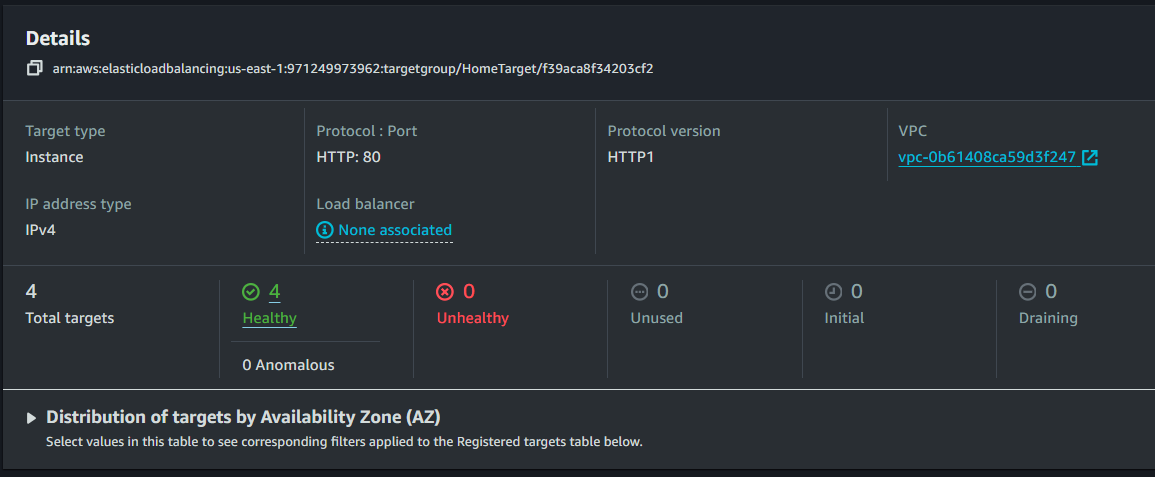
Type your ALB name, and Select VPC, and select the Availability zone where Instances are launched.



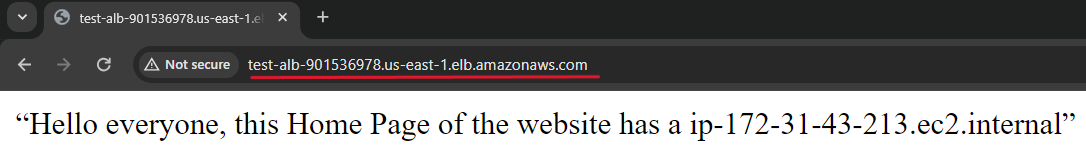
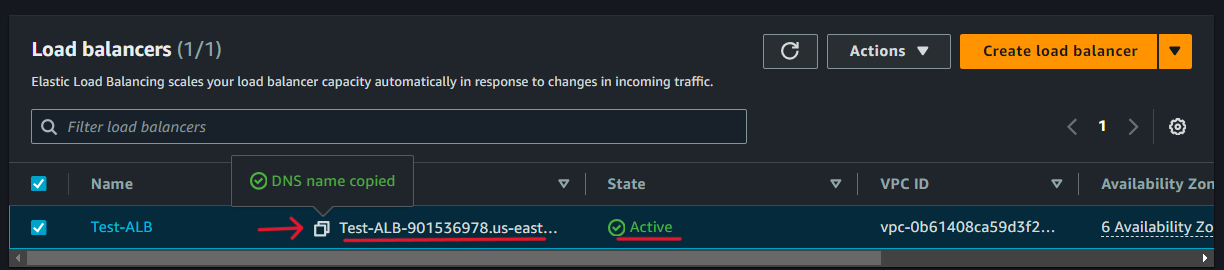
Step 8) In the Listener and routing option, the Default option Select our Home page target group.



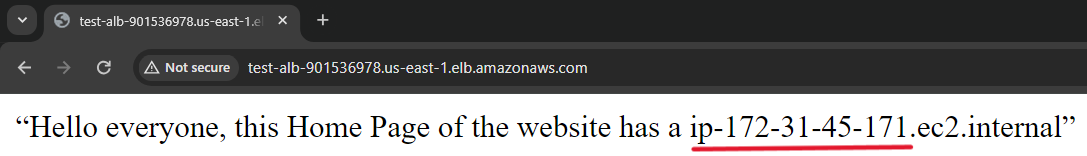
Step 9) Click on the Create Load Balancer Button, It will take some time to provision, once in Active status then go to the target groups and check the targets health Status.



Step 10) Copy DNS of load balancer and paste it into another browser.



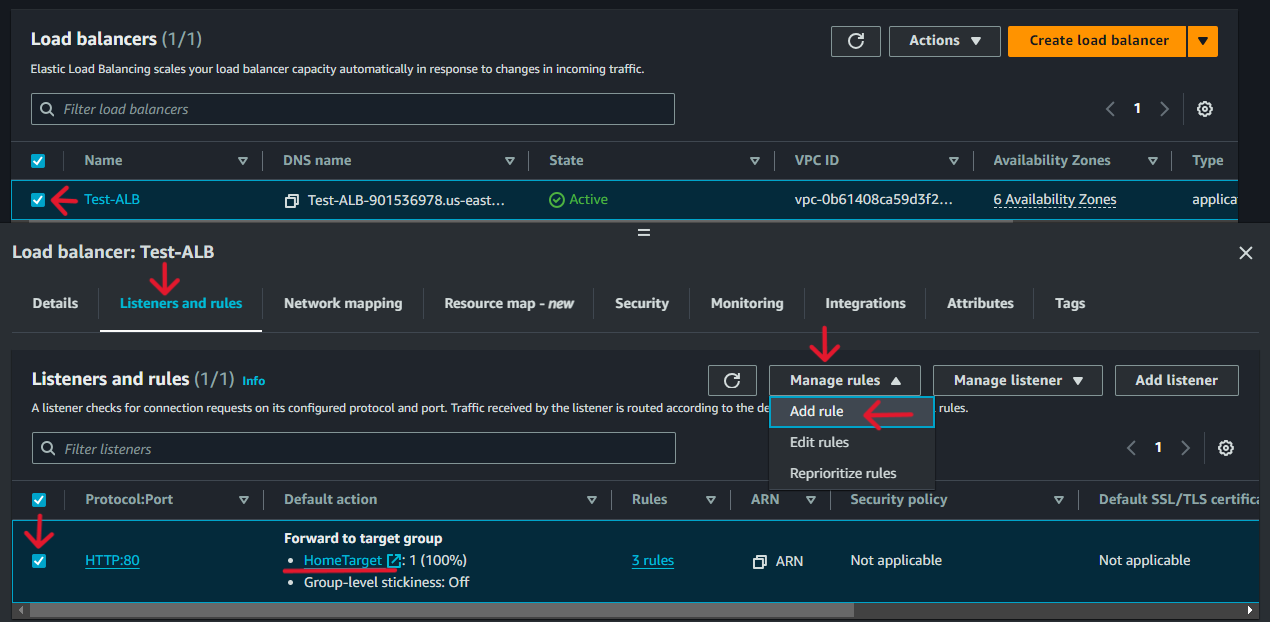
Our load balancer DNS is working, now we refresh it, if the IP changes, that means the load balancer is distributing the traffic on other instances.



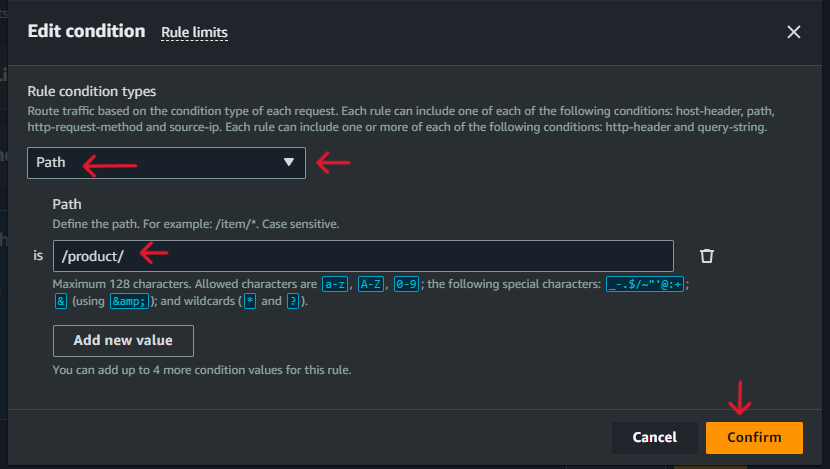
Traffic Distribution is properly working

Step 10) We have only Added the Home Target group in Default Action, now we add the rules for Product Target and List Target Groups in the Home Target listener’s add rule option.

* Select our Load Balancer
* Click on the Listener and Rules option
* Select our default listener, click the manage rule, and click on the add rule button.



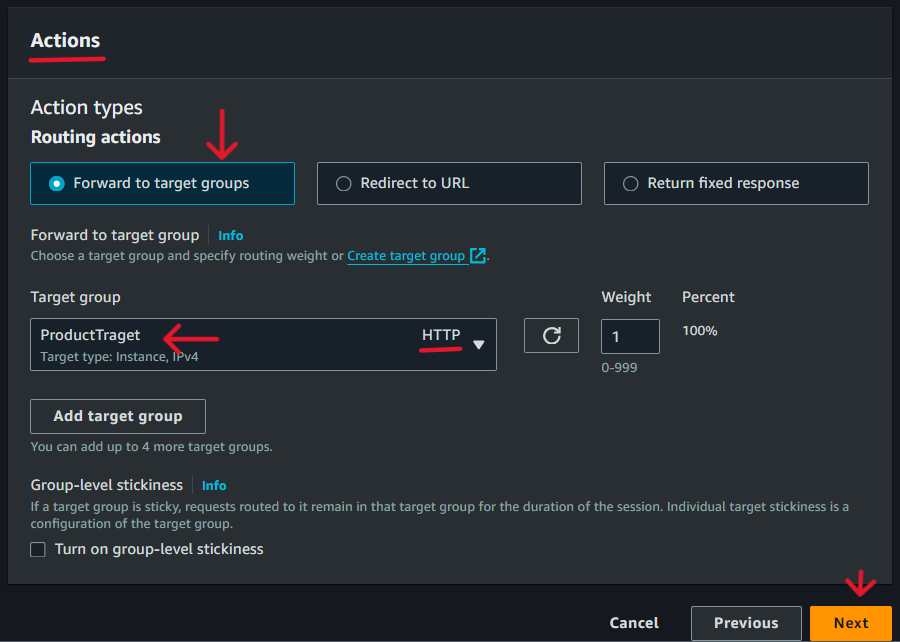
* Enter the rule name, click on the next button
* Click on the add condition button, and Select the condition type as a path. Then enter the path, click confirm, and next.



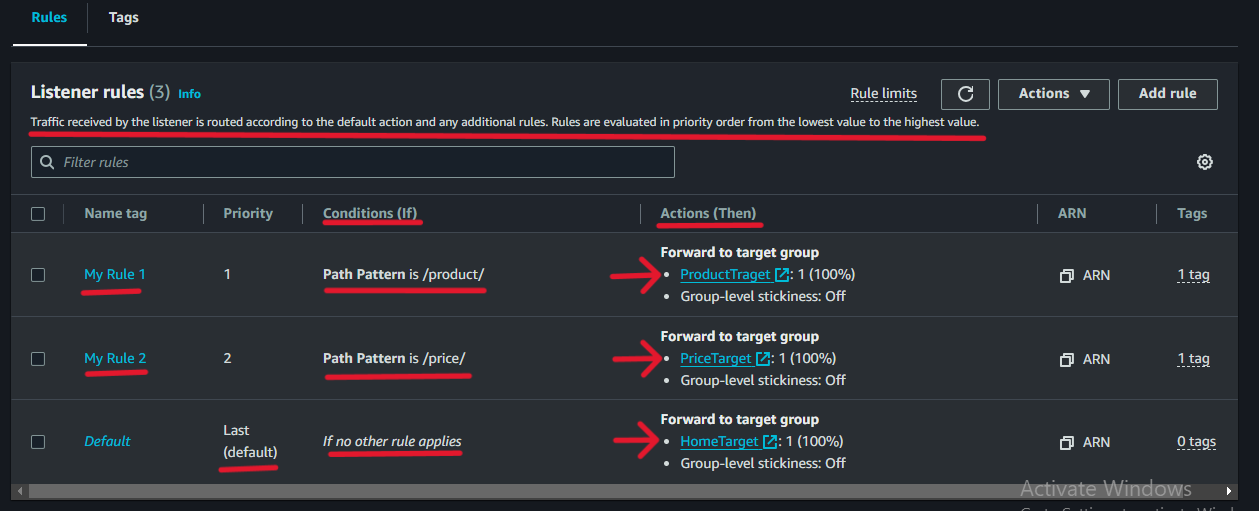
We have given different paths for our Product and Price pages and all have the same protocol port HTTP 80. So here we doing Path Base routing. It allows to direct incoming requests to different target groups based on the URL path. In our scenario for the Home server Product, and Price are two different backend Servers hosted on the same ALB but route requests to different backend resources depending on the requested path.

This setup enables to host multiple applications or services behind one ALB, simplifying your infrastructure while ensuring each application handles requests directed to its specific path.

* Select the Forward to target groups option, select target group, and click on next button.



* Enter priority and click on the next button, then review and click on the create button.
* Now do the same for the price target group.



Step 11) Now go target group and check the health status of all target group's Instances. These health checks verify that the targets are responsive and able to handle incoming requests.

Step 12) Now copy the DNS URL of ALB and paste it into the new Browser and add the path of our Product and Price Pages.

Step 13) Refresh the web, the IP address will change. It means the load balancer is Distributing traffic among the backend servers.



