Static website hosting => userdata
Loadbalancer, Monolith, Microservices

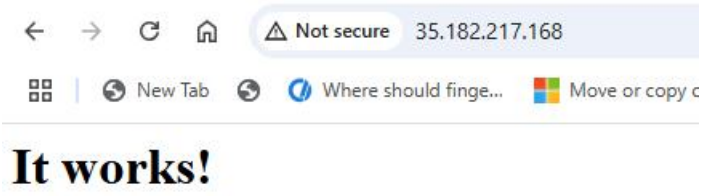
Website: Collection of web pages (html pages)
Static website (gives same response to every user) and Dynamic website (gives response based on user activity)

Webserver -> used to host/run our website

For Static websites we have httpd, apache2
For Dynamic websites: tomcat, IIS

Hosting website using httpd:

[ec2-user@ip-172-31-13-80 ~]$ sudo systemctl start httpd



[ec2-user@ip-172-31-13-80 ~]$ cd /var/www/html

[ec2-user@ip-172-31-13-80 html]$ sudo vi index.html

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

Note: Enable HTTP: 80 in security group inbound rules

Access our website using EC2 VM public IP

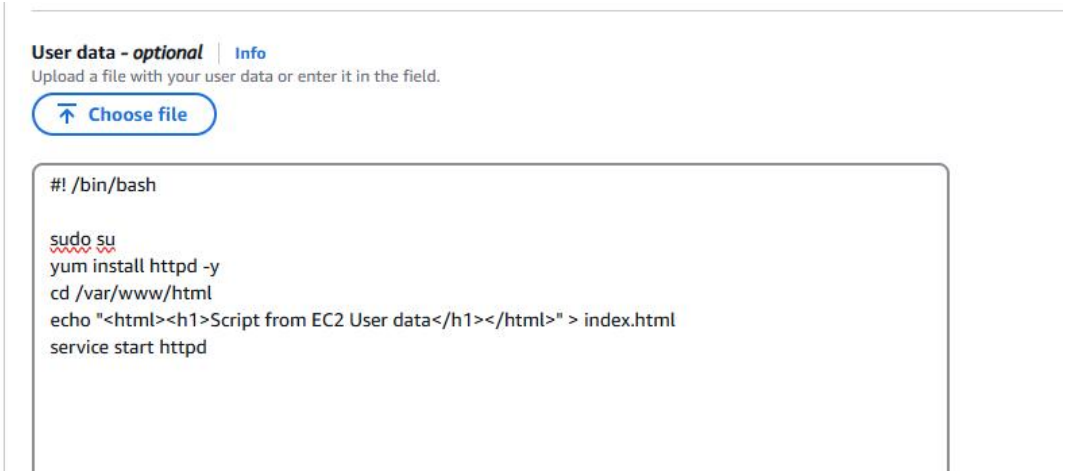
To modify the content we can navigate:
cd /var/www/html
sudo vi index.html

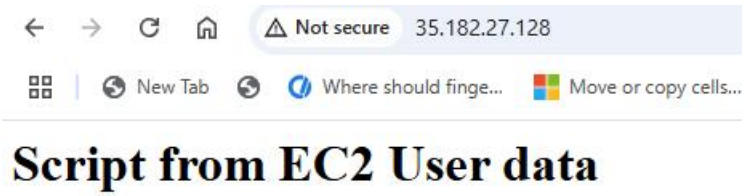
Access again our website using EC2 instance public IP
User-data in EC2 VM:
--> used to execute script while launching machine
--> user data will execute only once

Create a new EC2 VM with below user data

#! /bin/bash

sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Script from EC2 User data</h1></html>" > index.html
systemctl start httpd





Disadvantages of having one server for our application:
One server must handle all the incoming requests
High burden on server, which might result in delay in responses
Can lead to server crash (single point of failure)
All these problems will lead to business loss

Deploying same application into multiple servers
App is now deployed in say 3 servers
How can we divert the traffic to multiple servers?
Load balancer: All requests to application will be diverted to multiple servers (Round robin)
Application will run on multiple servers
-> load will be distributed
-> Fast performance
-> High availability

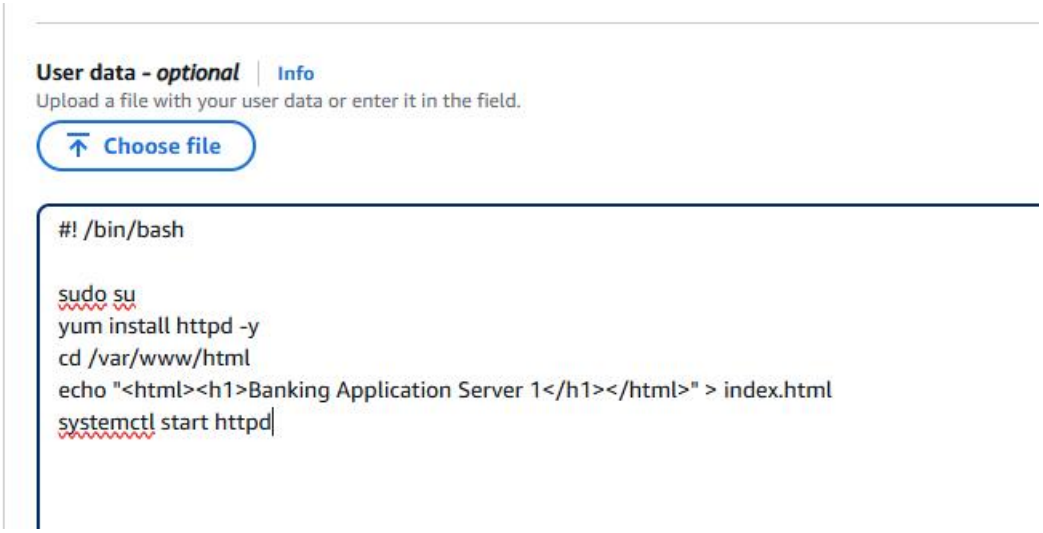
Load balancer: it is used to distribute incoming load / requests to multiple servers in round robin technique

There are different types of Load balancers in AWS:
1. Application load balancer (http & https)
2. Network load balancer

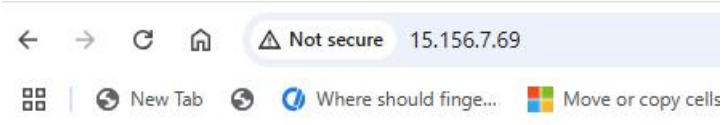
3. Gateway load balancer

Classic Load Balancer (outdated / old gen)

Practical task on Load balancer:



Banking Application Server 1



BankingApplication Server 2

Target group: List of servers running our application
These two servers together is called as a Target group

Create two VMs and deploy Baning Application

Add these instances to one target group -> List of servers running our application

Target groups Info

Filter target groups

Name	ARN	Port	Protocol	Target type

No target groups

You don't have any target groups in ca-central-1

Create target group

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.

Target group name

BankingApplication

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

Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection created. This choice cannot be changed after creation

HTTP | 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

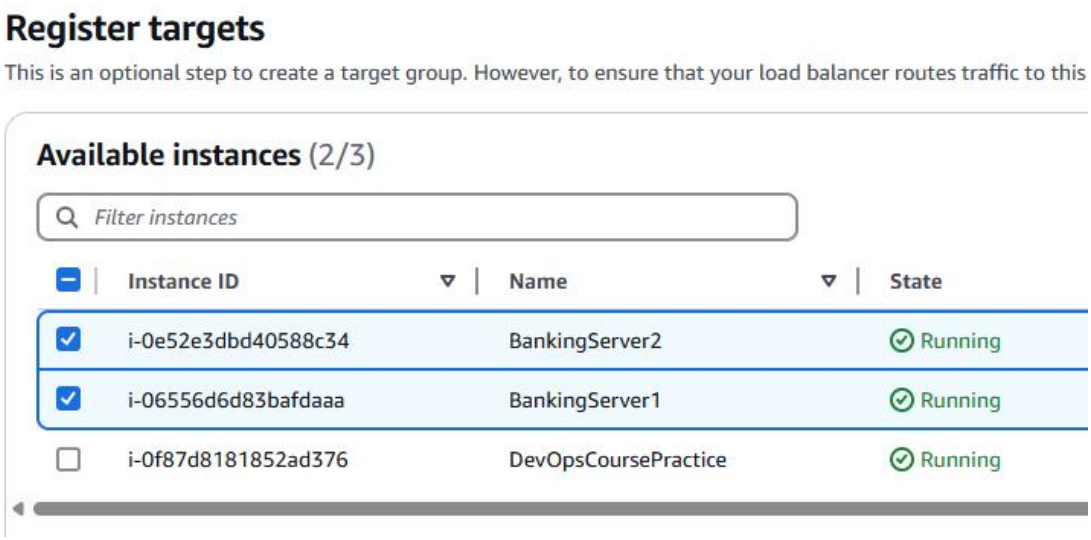
VPC

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

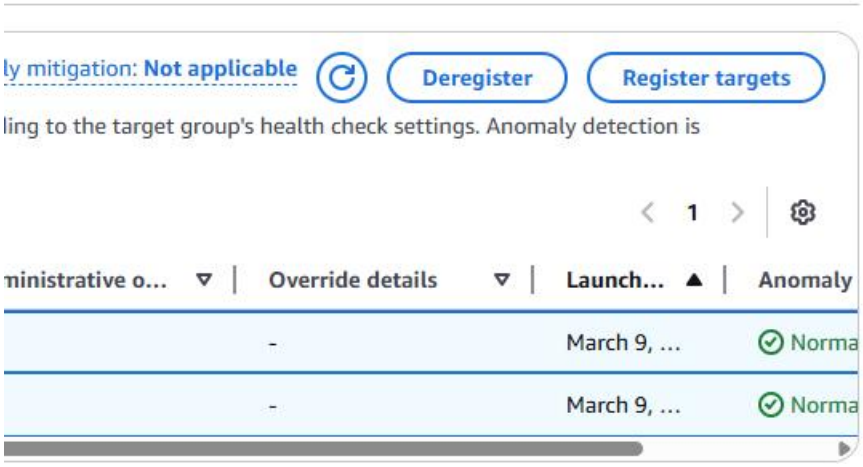
-
vpc-0a752647f0a021f2e
IPv4 VPC CIDR: 172.31.0.0/16

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.

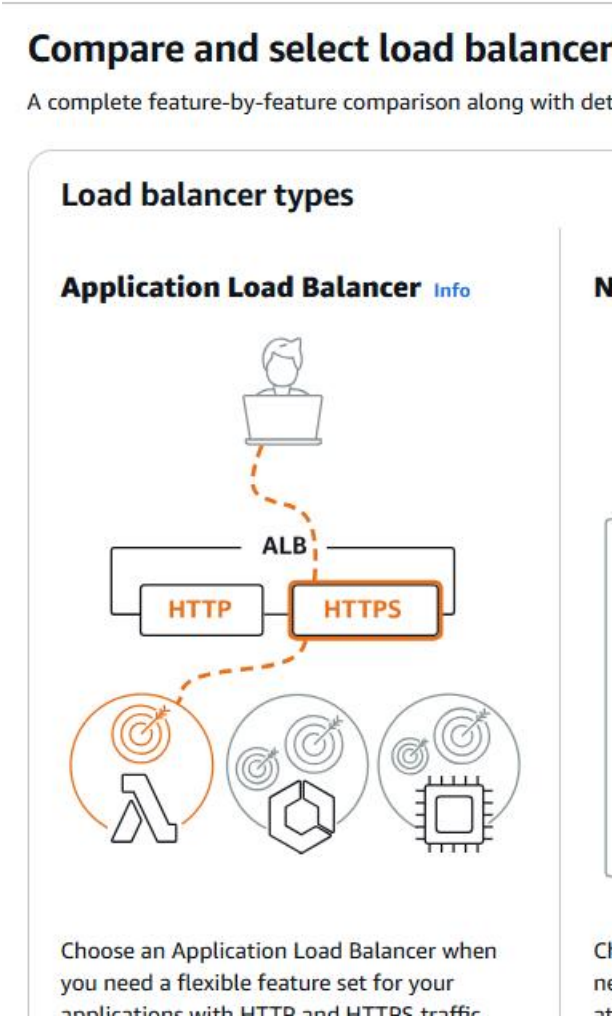


Click ->Include as pending below -> Create Target Group

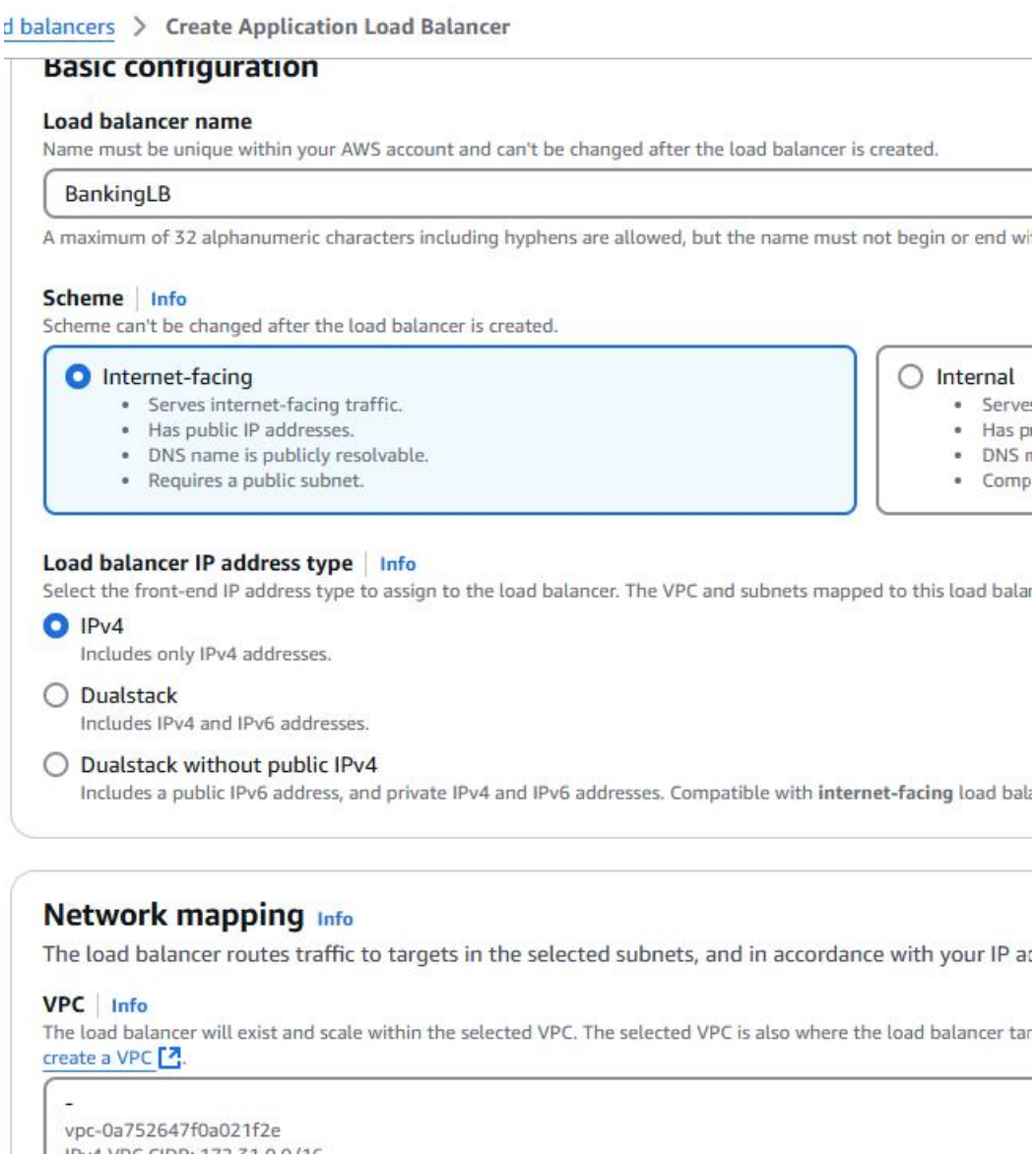


Register Targets

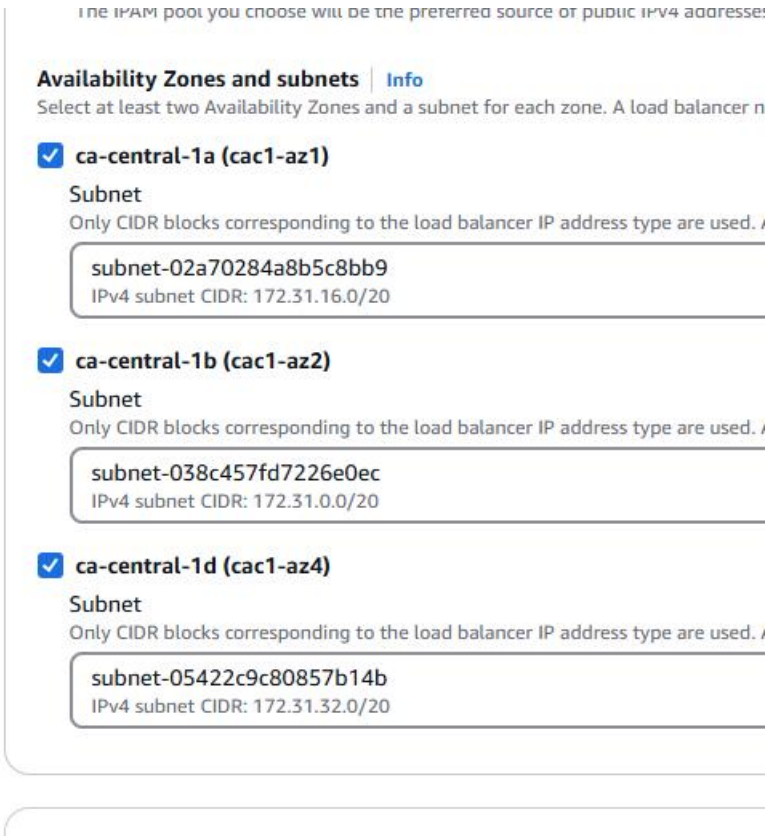
Go to Load Balancers



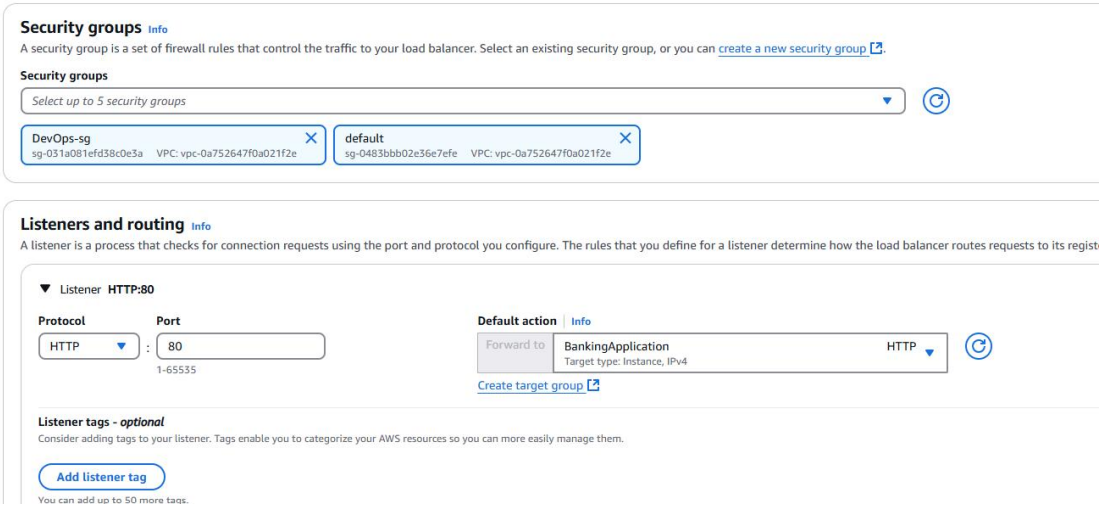
Create Application Load Balancer
Internet facing and leave other options as it is

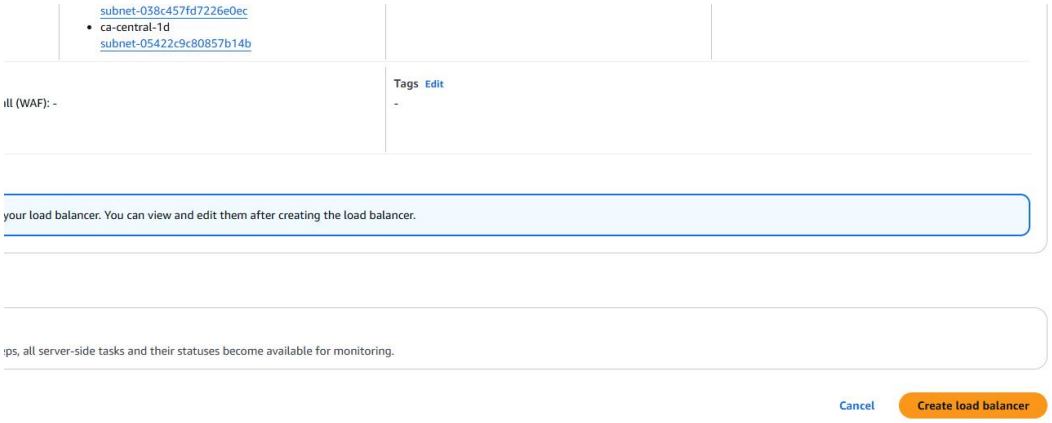


Select Availability Zones



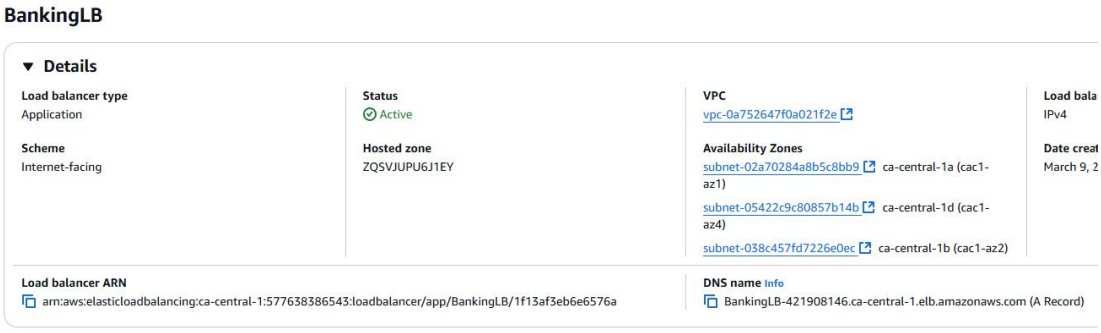
Select Security Groups, Target Group



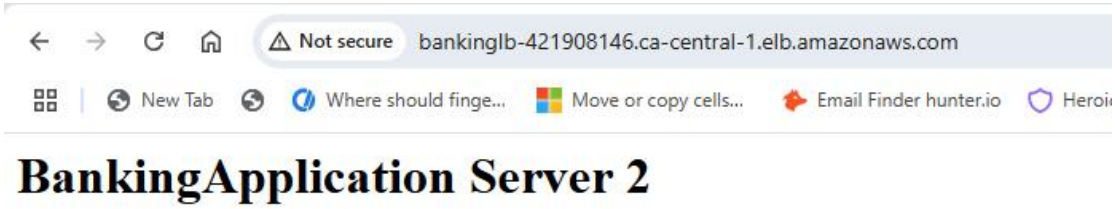


Scroll down then Create Load Balancer

It gives public DNS



DNS



BankingApplication Server 2

When we refresh the page:

Banking Application Server 1

If you decide to add a new Server say Server 3, we add to the Target group that's it Load balancer will be updated as well

Architectural design patterns:
Monolithic Vs Microservices

Monolithic -> developing all functionalities in single application

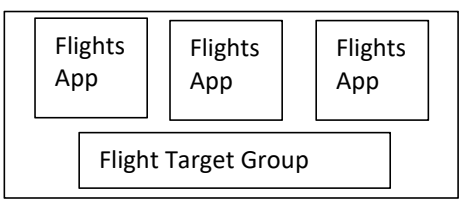
Deploying one application into multiple servers, it is Monolithic

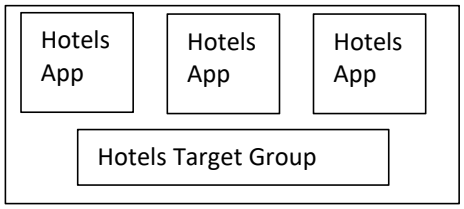
For Monolithic app usually we need one target group
For Microservices: dividing functionalities into multiple APIs . One application is divided into micro applications.
BookingApplication -> features like hotels, flights, car rentals everything in one application. This is monolithic

(Monolithic) App -> UI <- end user -> UI -> Hotels App, Car rentals App, Flights App (Microservices)

So many requests are coming to Application -> we have a Load balancer in between. In Target group, we have 3 applications -> this is Monolithic

In Microservices, I have 3 car rental applications deployed on multiple servers, 3 hotel applications deployed on multiple servers, 3 flight applications deployed on multiple servers





So it is challenging for Load balancer. that's why we need to configure "Routing"

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Service 1</h1></html>" > index.html
systemctl start httpd
```

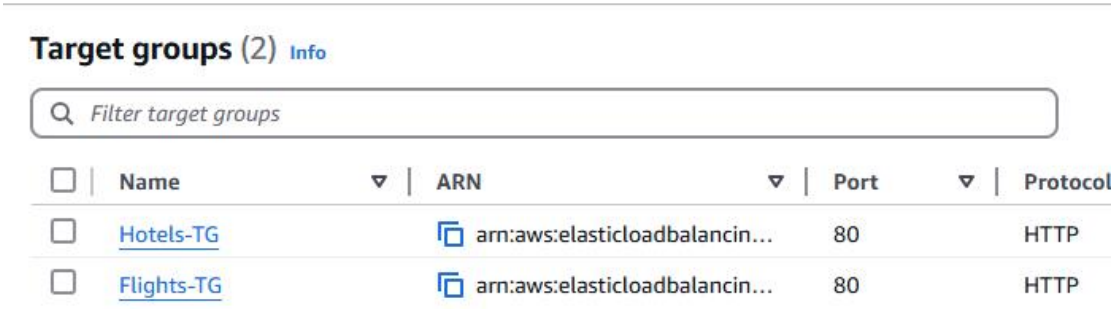
```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Service 2</h1></html>" > index.html
systemctl start httpd
```

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotels Service 1</h1></html>" > index.html
systemctl start httpd
```

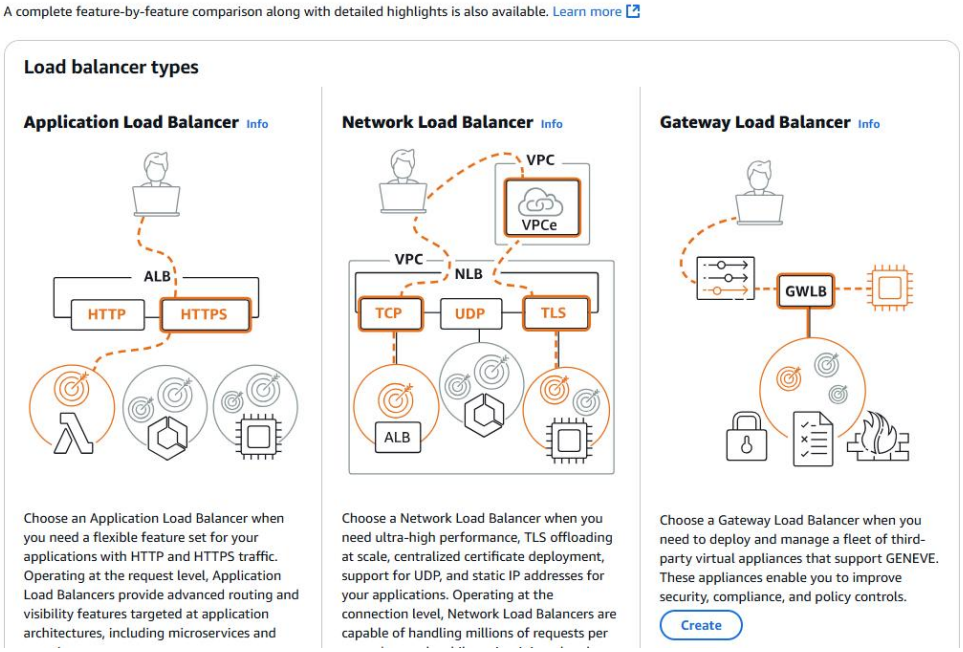
```
#! /bin/bash
sudo su
yum install httpd -y
```

cd /var/www/html
echo "<html><h1>Hotels Service 2</h1></html>" > index.html
systemctl start httpd

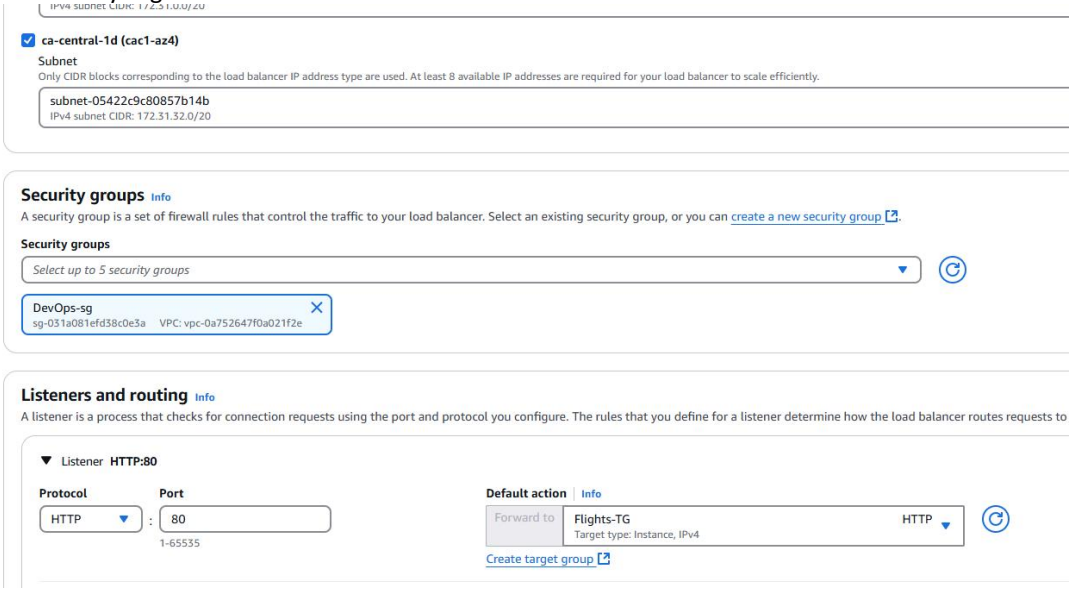
We create 2 servers for each service then we register into Target groups

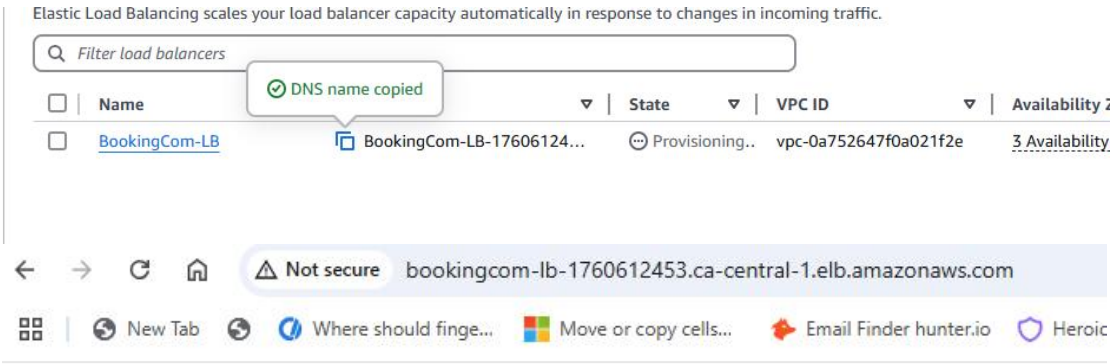


For a normal HTTP application, Application Load Balancer is recommended

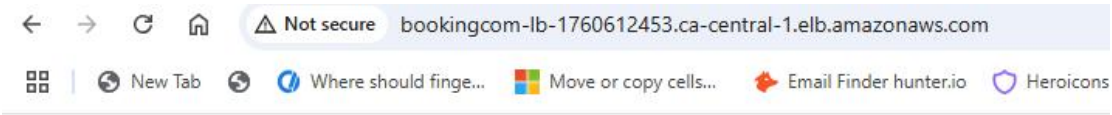


First I add only Flights-TG



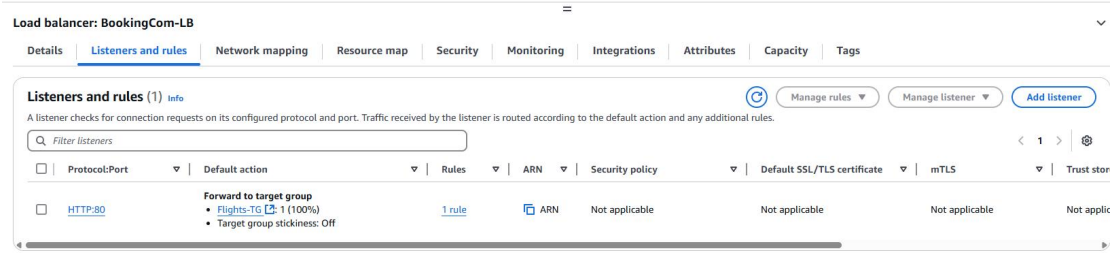


Flights Service 1

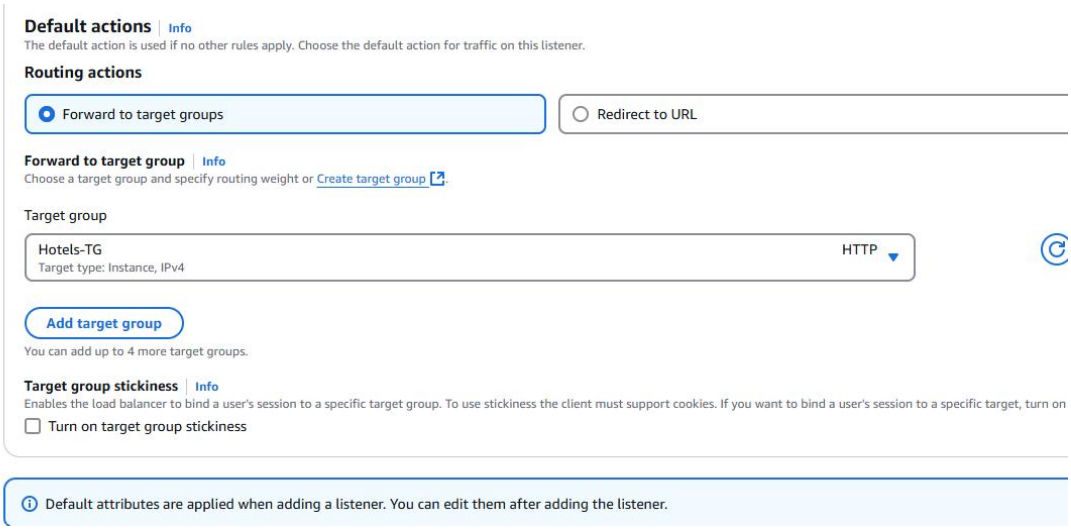


Flights Service 2

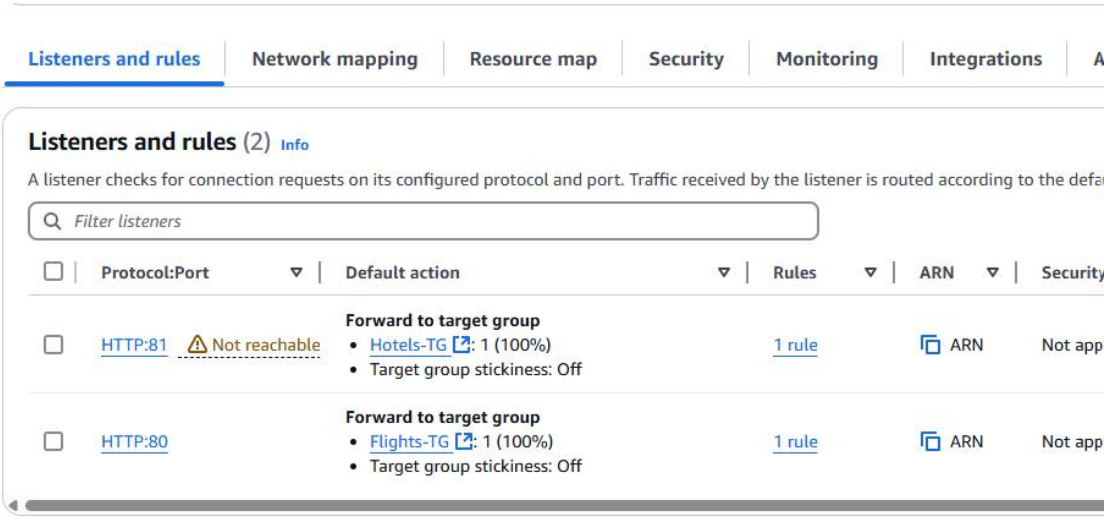
Click on Add Listener



Hotels added

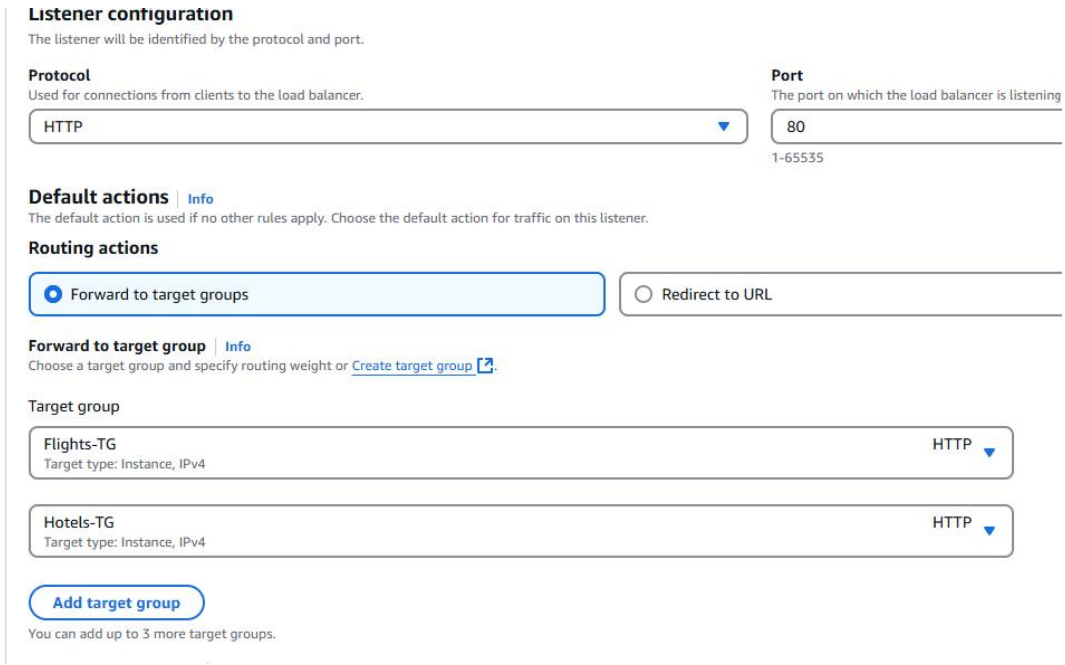


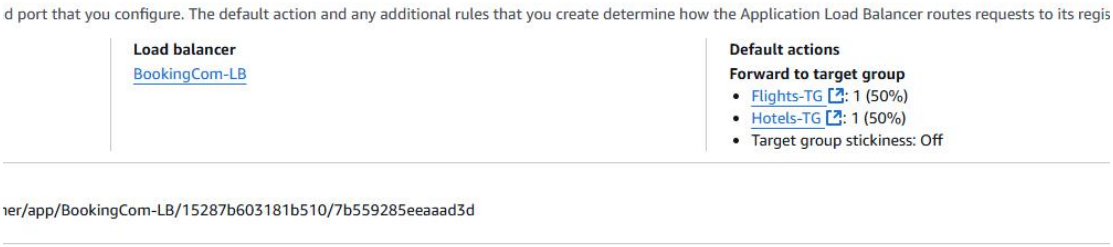
Two listeners



81 is not reachable so we are going to edit the existing listener

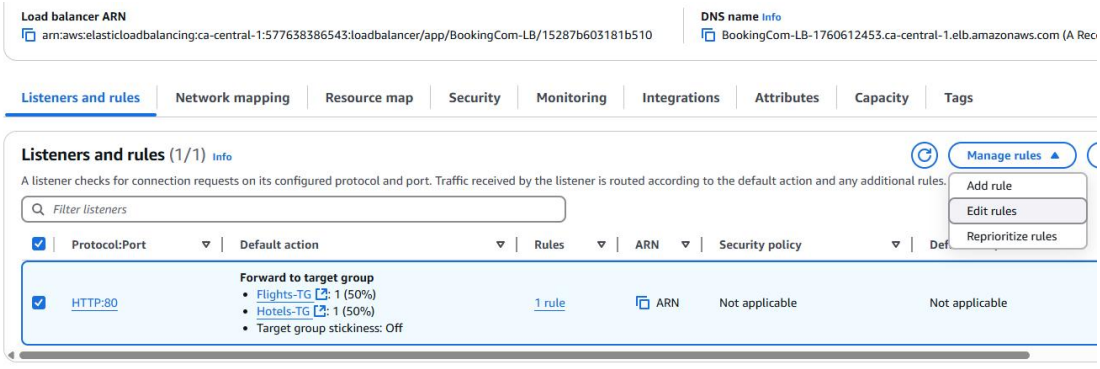
Edit existing target group and add new Target group



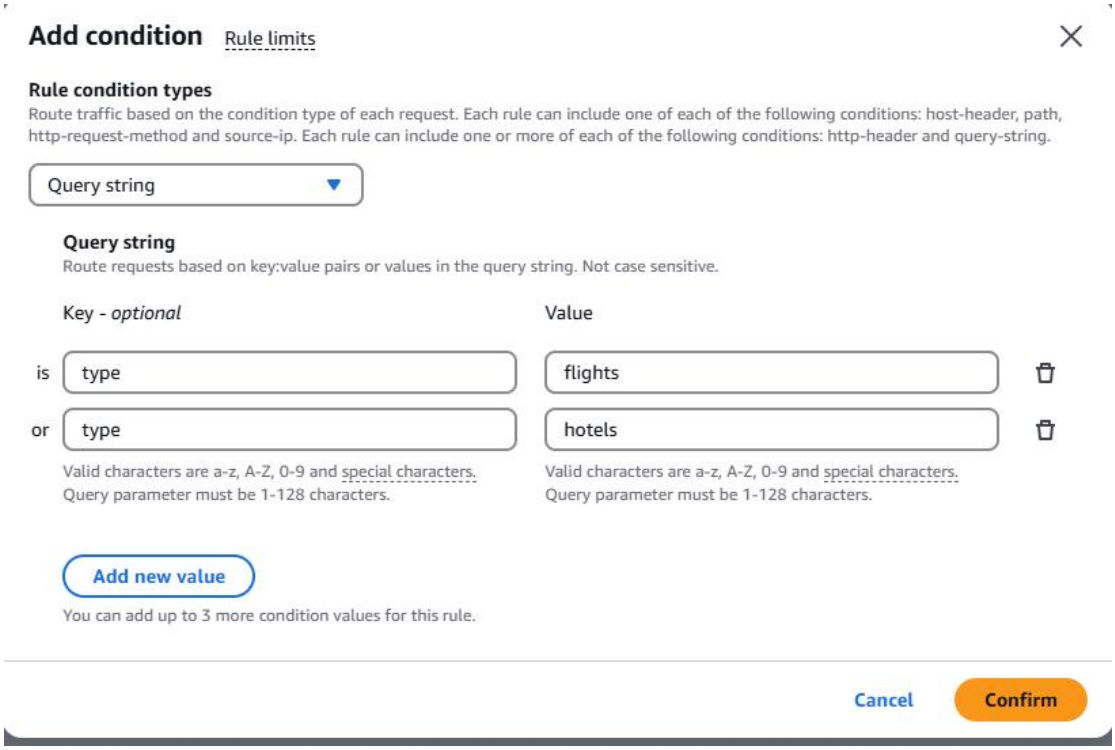


It shows both Hotel and Flight services but we don't want this random decision

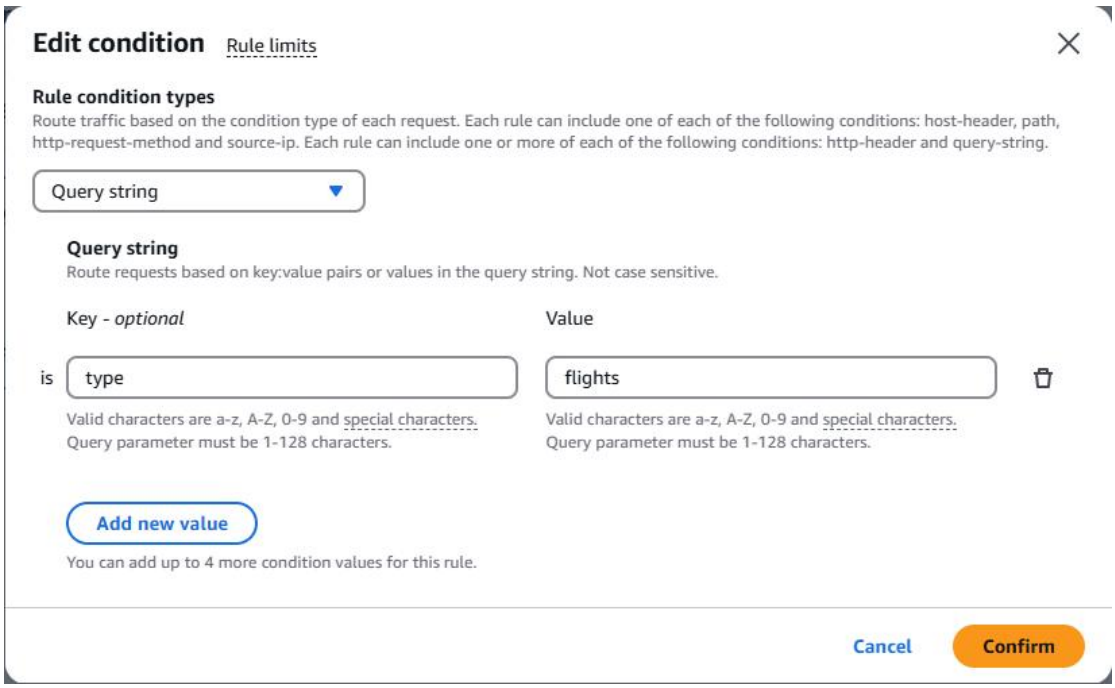
Add rule



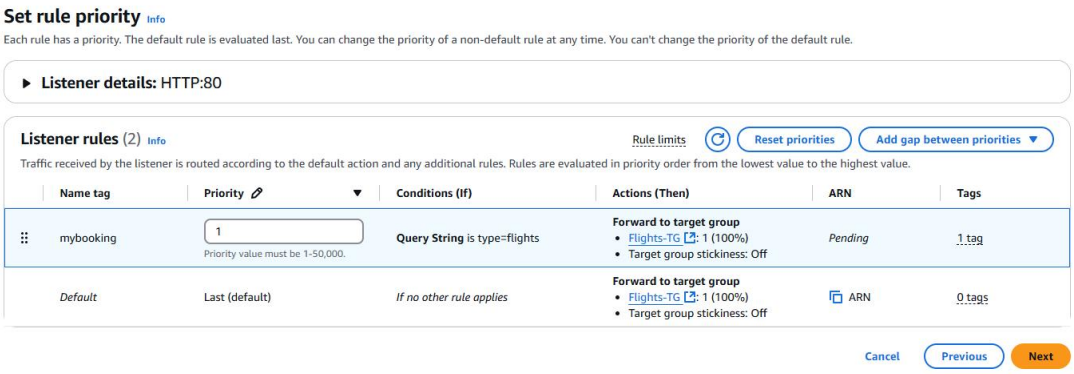
Select Query String



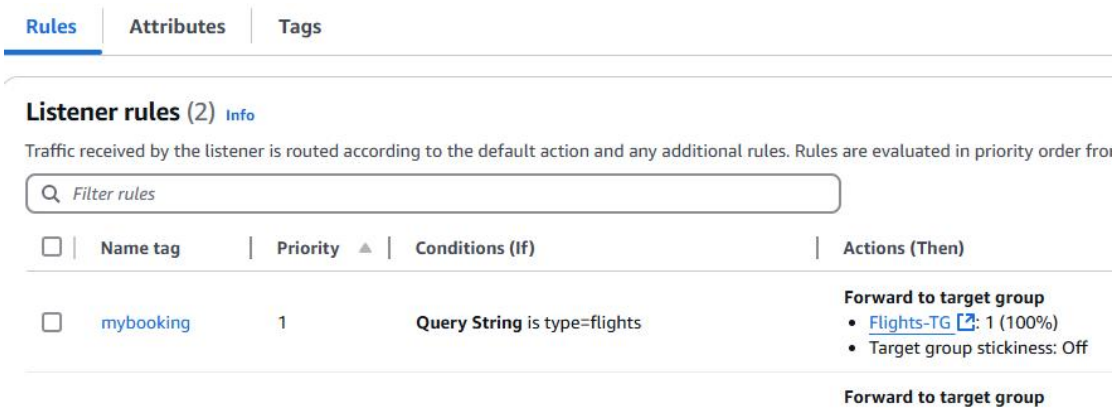
We have edited the Query String



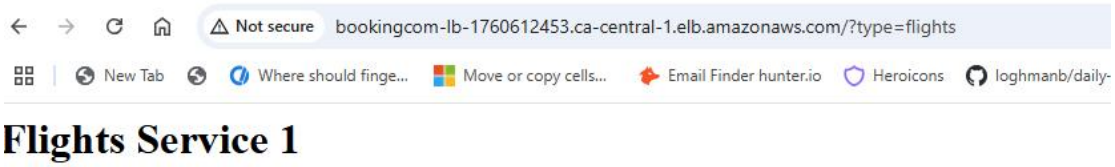
Set Priority to 1



Listener rules created

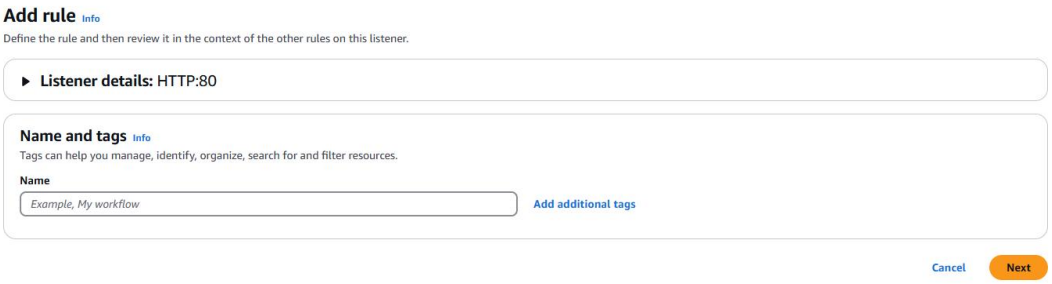


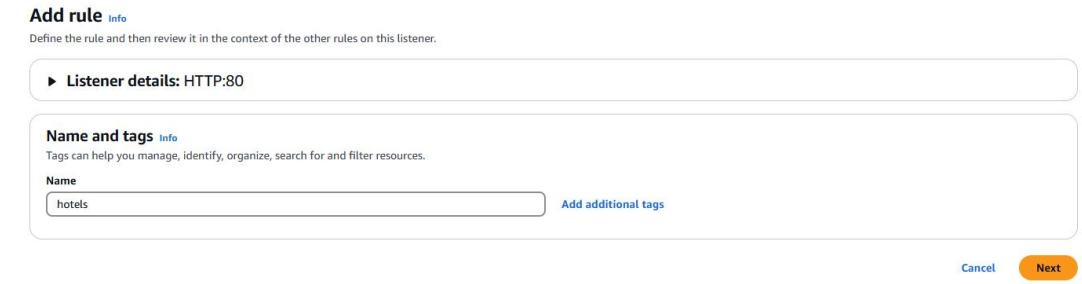
When I pass the Query parameter I get only Flights in this case
http://bookingcom-lb-1760612453.ca-central-1.elb.amazonaws.com/?type=flights

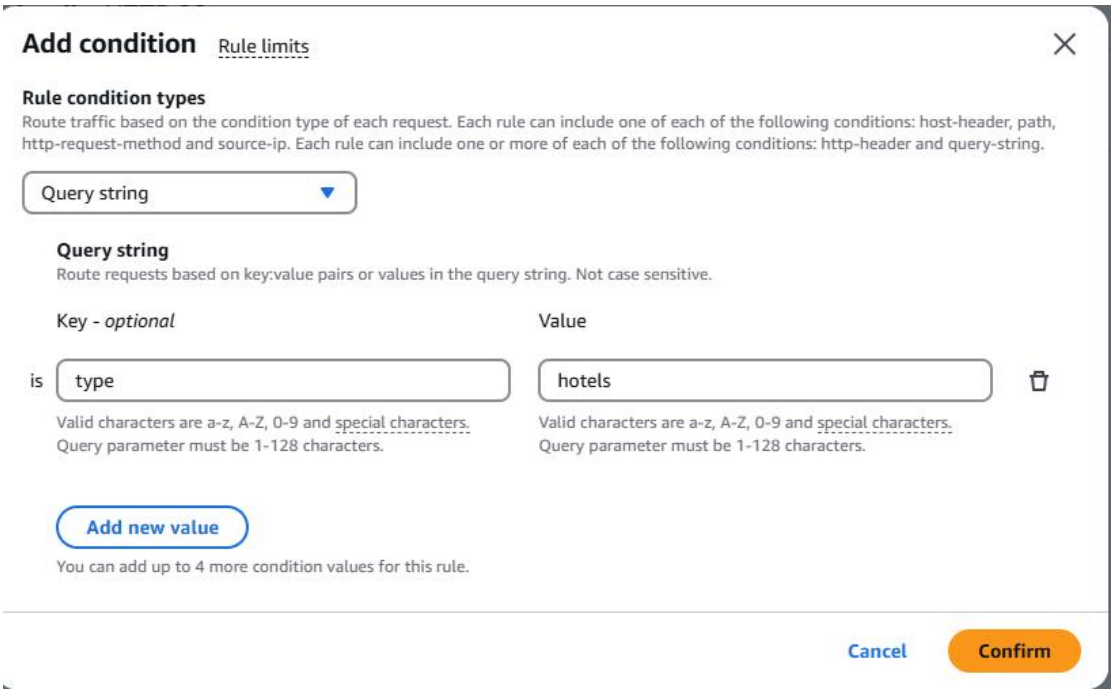


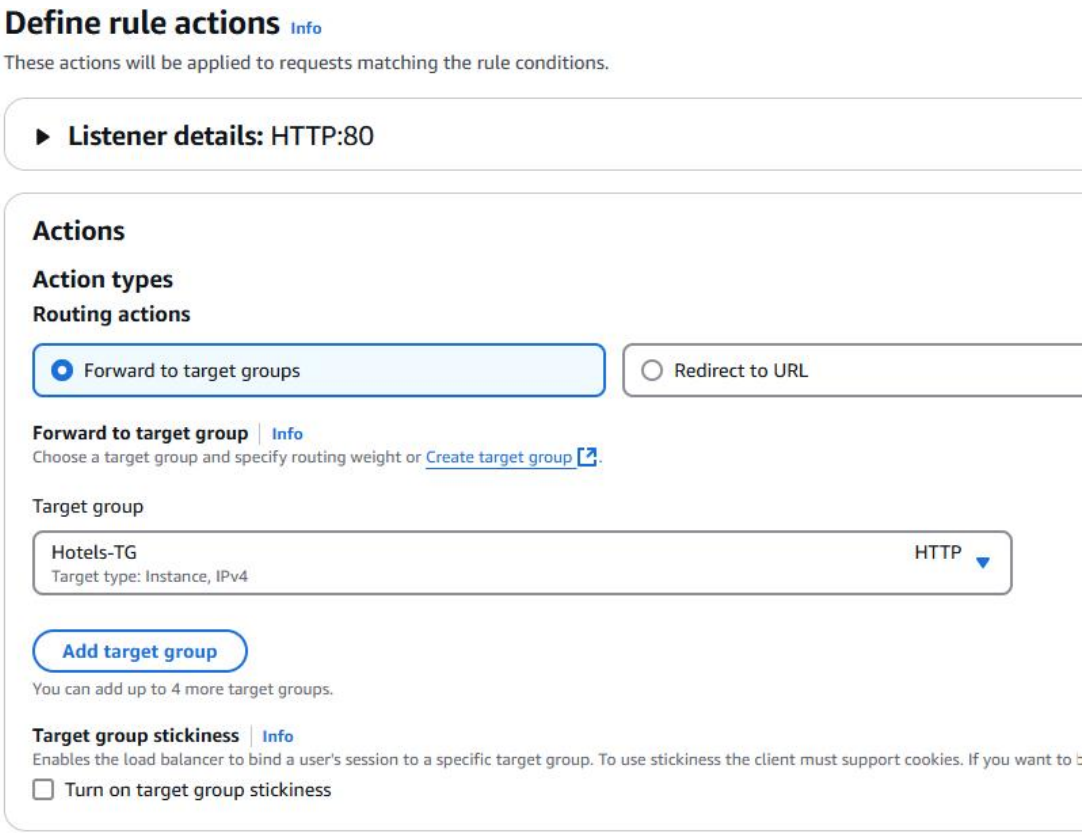
Flights Service 1

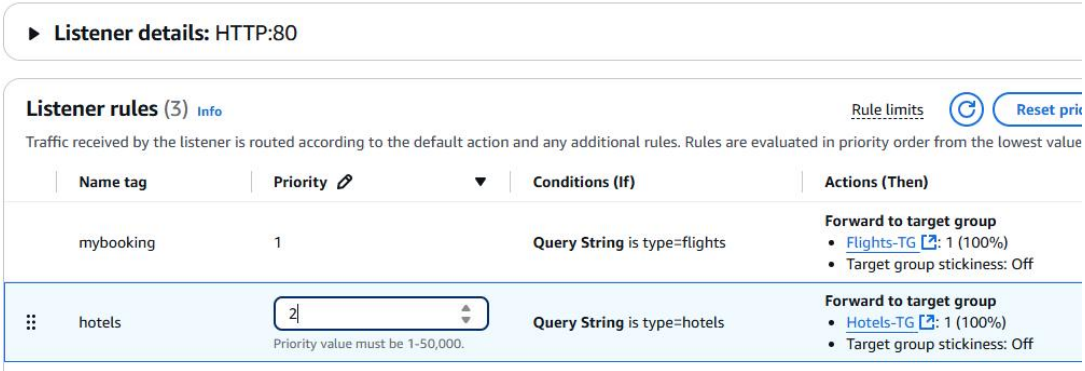
Again add the rule



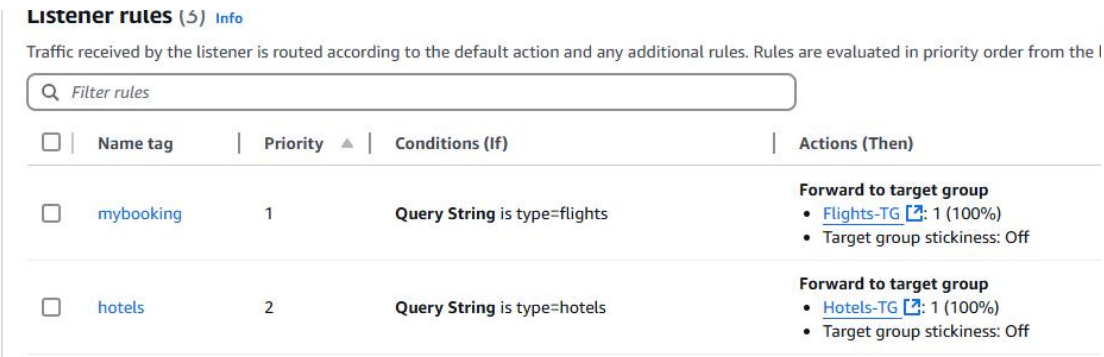

Add rule Info
Define the rule and then review it in the context of the other rules on this listener.
Listener details: HTTP:80
Name and tags Info
Tags can help you manage, identify, organize, search for and filter resources.
Name
hotels
Add additional tags
Cancel
Next


Add condition
Rule limits
Rule condition types
Route traffic based on the condition type of each request. Each rule can include one of each of the following conditions: host-header, path, http-request-method and source-ip. Each rule can include one or more of each of the following conditions: http-header and query-string.
Query string
Query string
Route requests based on key:value pairs or values in the query string. Not case sensitive.
Key - optional
Value
is
type
hotels
Valid characters are a-z, A-Z, 0-9 and special characters.
Query parameter must be 1-128 characters.
Valid characters are a-z, A-Z, 0-9 and special characters.
Query parameter must be 1-128 characters.
Add new value
You can add up to 4 more condition values for this rule.
Cancel
Confirm

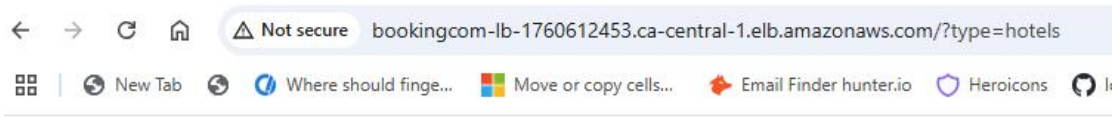




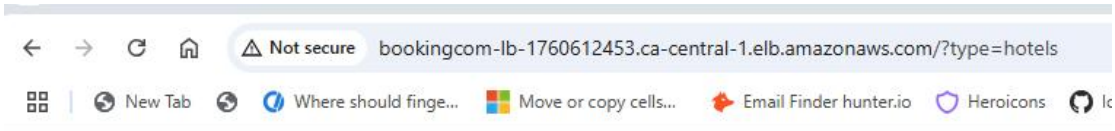
We see two rules



Now we are getting only Hotels Service
http://bookingcom-lb-1760612453.ca-central-1.elb.amazonaws.com/?type=hotels



Hotels Service 1



Hotels Service 2

In what situation, we may have to create multiple target groups?
Microservices

Types of Load Balancers:
Application Load Balancers (ALB)
Network Load Balancers (NLB)
Gateway Load Balancers (GLB)

OSI model: Open Systems Interconnections
It contains 7 layers, it will represent how request will transfer from Client to Server

Client ----------> Server
<----------
During this communication, there are 7 layers and they are called as OSI layers

Layer 7: Application layer (Application Load Balancer)
Layer 6: Presentation layer
Layer 5: Session layer
Layer 4: Transport layer (Network Load Balancer)
Layer 3: Network layer
Layer 2: Data link layer
Layer 1: Physical layer

Client'request first goes to Application layer (At this point, Application Load Balancer intercepts the request, then it determines where it has to go Hostname, entire path: like Flights or Hotels etc) -> Presentation layer -> Session layer -> Transport layer (say if it is a high performance application where we don't want to decide which Target group to go to etc, something like Gaming application where decision-making is not important then Network Load Balancer comes into picture) -> Network layer -> Data link layer -> Physical layer -> Server (App)

Decision-making -> Application Load Balancer
Non-decision-making something like Gaming then we go with Network Load Balancer

Application Load Balancer:
Operates at 7 (Application layer of OSI model)
It is desined to route HTTP and HTTPS traffic based on content (path-based routing & host-based routing with the help of HTTP headers)

Ideal for modern web applications, Microservices and (Container based application)
Request per minute -> maybe 10k / min -> lets say we have 3 servers currently
Say 100,000 RPM -> then we need 30 servers to manage 100,000 requests
If requests increase to 200,000 then we need to increase the servers accordingly

Auto-scaling concept is there to handle variations in servers based on RPM

Auto-scaling -> Fault tolerance (if some instance is unhealthy, it will automatically launch to replace it), Cost management, Availability