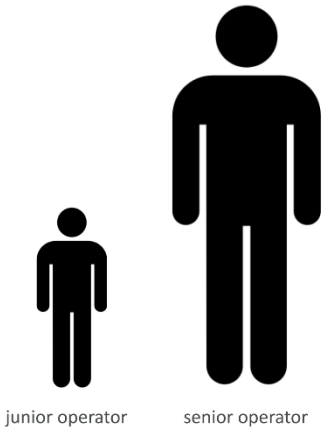
High Availability (HA)

Scalability & Hight Availability

- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)
- Scalability is linked but different to High Availability

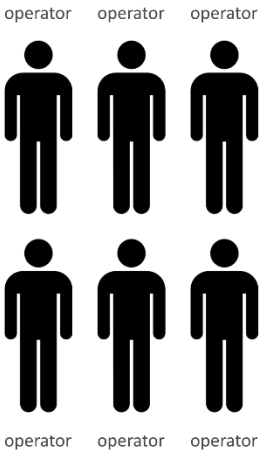
Vertical Scalability

- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- RDS, ElastiCache are services that can scale vertically.
- There's usually a limit to how much you can vertically scale (hardware limit)



Horizontal Scalability

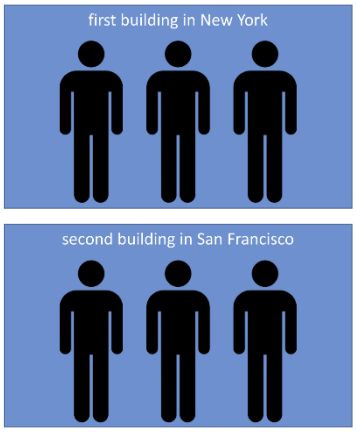
- Horizontal Scalability means increasing the number of instances / systems for your application
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2



Hight Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss

- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)



Hight Availability & Scalability For Ec2

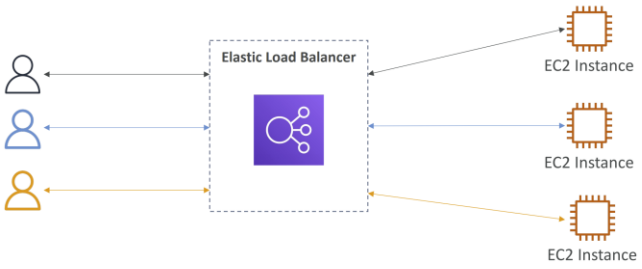
- Vertical Scaling: Increase instance size (= scale up / down)
  - From: t2.nano - 0.5G of RAM, 1 vCPU
  - To: u-12tb1.metal – 12.3 TB of RAM, 448 vCPUs

- Horizontal Scaling: Increase number of instances (= scale out / in)
  - Auto Scaling Group
  - Load Balancer

- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ

Load balancing

- Load Balances are servers that forward traffic to multiple servers (e.g., EC2 instances) downstream



Why use a Load balancer

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

Why Use an Elastic Load Balancer

- An Elastic Load Balancer is a managed load balancer
  - AWS guarantees that it will be working
  - AWS takes care of upgrades, maintenance, high availability
  - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end
- It is integrated with many AWS offerings / services
  - EC2, EC2 Auto Scaling Groups, Amazon ECS
  - AWS Certificate Manager (ACM), CloudWatch
  - Route 53, AWS WAF, AWS Global Accelerator

Health Check

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (/health is common)
- If the response is not 200 (OK), then the instance is unhealthy

Ec2 instance can be unhealthy in 2 ways:

1. ec2 instance is down means power off or any other hardware failure.
2. Ec2 instance is up but the webserver is down.

Need in HA
1. Auto scaling
2.laungh template
3.for launch template we need predefine template
4.auto scaling group
5.sns
6.cloud watch
7.load balancer
8.target group

Types of load balancer in AWS

- AWS has 4 kinds of managed Load Balancers
- Classic Load Balancer (v1 - old generation) – 2009 – CLB
  - HTTP, HTTPS, TCP, SSL (secure TCP)
- Application Load Balancer (v2 - new generation) – 2016 – ALB
  - HTTP, HTTPS, WebSocket
- Network Load Balancer (v2 - new generation) – 2017 – NLB
  - TCP, TLS (secure TCP), UDP
- Gateway Load Balancer – 2020 – GWLB
  - Operates at layer 3 (Network layer) – IP Protocol

- Overall, it is recommended to use the newer generation load balancers as they provide more features
- Some load balancers can be setup as internal (private) or external (public) ELBs

Load balancer Security Groups

Load Balancer Security Group:

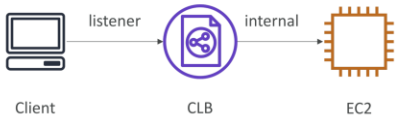
Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

Application Security Group: Allow traffic only from Load Balancer

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

Classic Load balancer

- Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname XXX.region.elb.amazonaws.com



Application Load balancer

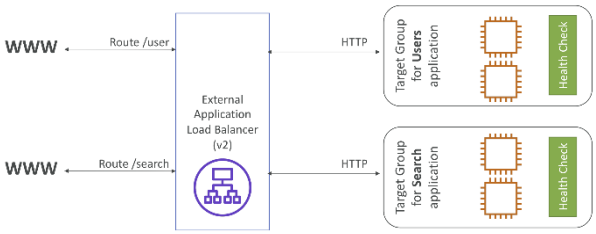
- Application load balancers is Layer 7 (HTTP)

- Load balancing to multiple HTTP applications across machines (target groups)
- Load balancing to multiple applications on the same machine (ex: containers)
- Support for HTTP/2 and WebSocket
- Support redirects (from HTTP to HTTPS for example)

- Routing tables to different target groups:
  - Routing based on path in URL (example.com/users & example.com/posts)
  - Routing based on hostname in URL (one.example.com & other.example.com)
  - Routing based on Query String, Headers (example.com/users?id=123&order=false)

- ALB are a great fit for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port in ECS
- In comparison, we'd need multiple Classic Load Balancer per application

HTTP Based Traffic

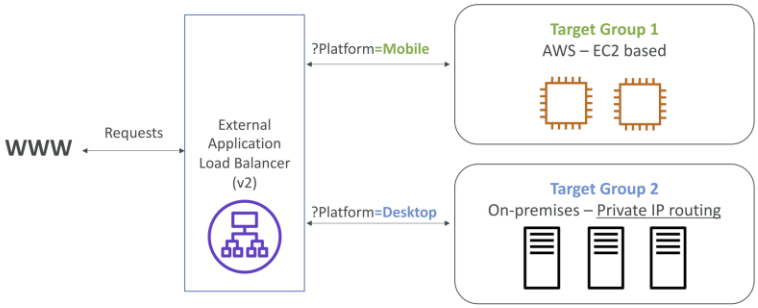


Target group

- EC2 instances (can be managed by an Auto Scaling Group) – HTTP
- ECS tasks (managed by ECS itself) – HTTP
- Lambda functions – HTTP request is translated into a JSON event
- IP Addresses – must be private IPs

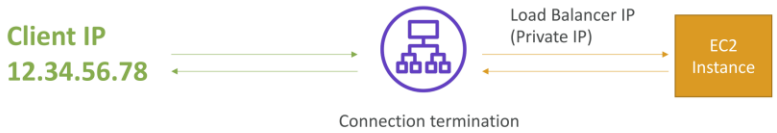
- ALB can route to multiple target groups
- Health checks are at the target group level

Query String parameters Routing



Some important things

- Fixed hostname (XXX.region.elb.amazonaws.com)
- The application servers don't see the IP of the client directly
  - The true IP of the client is inserted in the header X-Forwarded-For
  - We can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)



LAB: Application load balancer (ALB) create

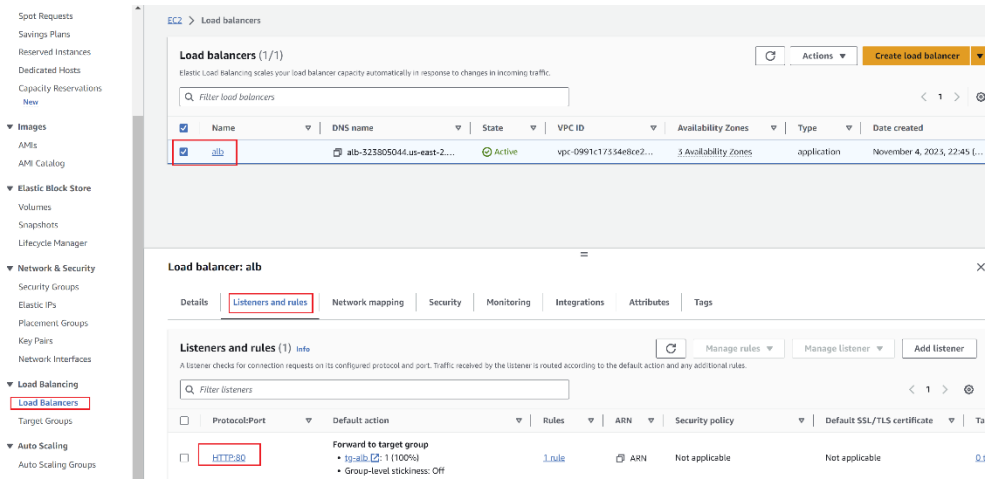
Create 2 security group one for ec2 and another one for ALB
Create 2 ec2 instances with user data
Browse the 2 ec2 IP and make sure both IP showing their web page
Create Target group and add this 2 ec2 in this target group
Create a Application Load Balancer
Now we can browse by load balancer DNS and we can see website come from both ec2
Now down any ec2 and then check load balancer DNS and we can see now page only come from one ec2

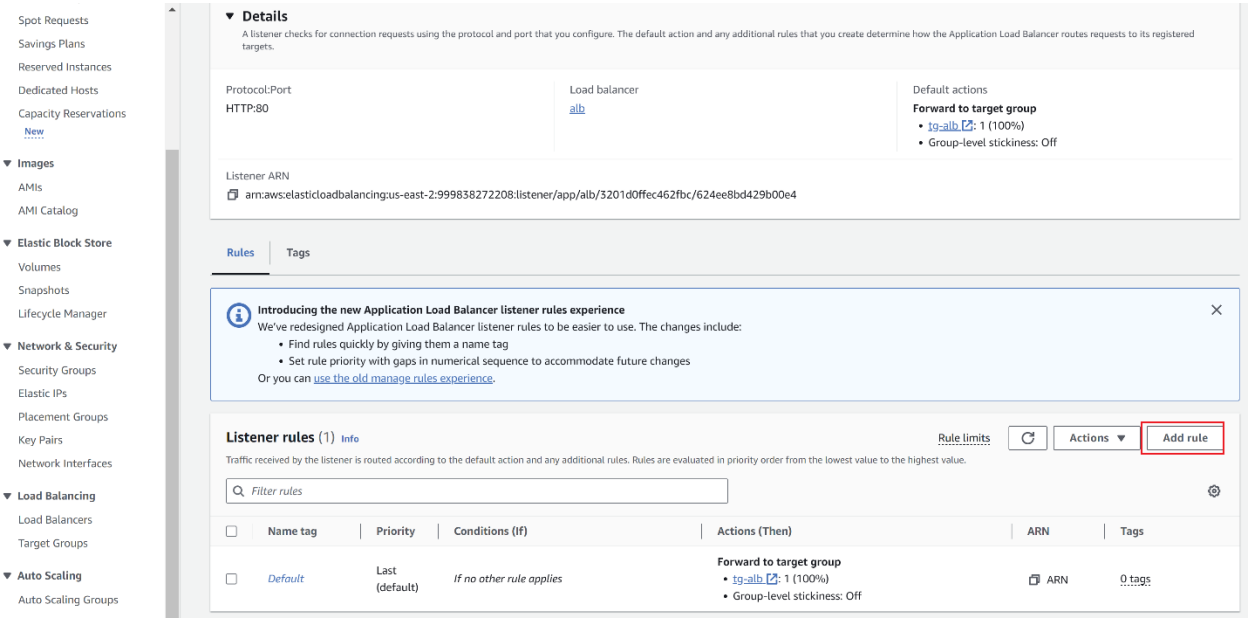
We can see website is showing from both ec2 ip and DNS now we want website only show from load balancer no webpage show from ec2
So, in ec2-security group we need to allow only load balancer http traffic for that purpose in security group we add load balancer security group id in ec2-security group

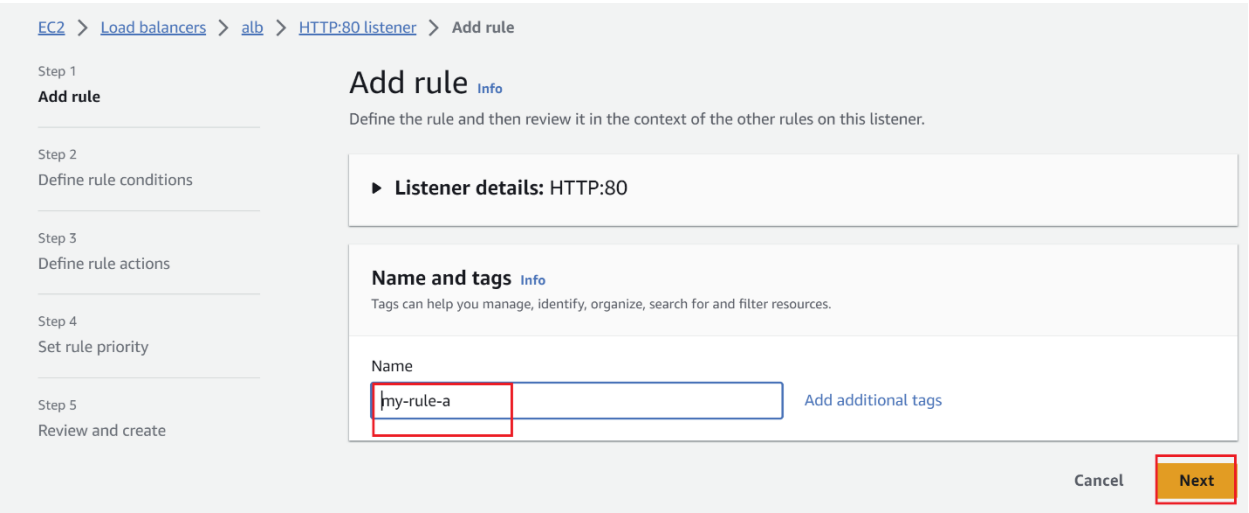
LAB: Create Application load balancer (ALB) listener rule

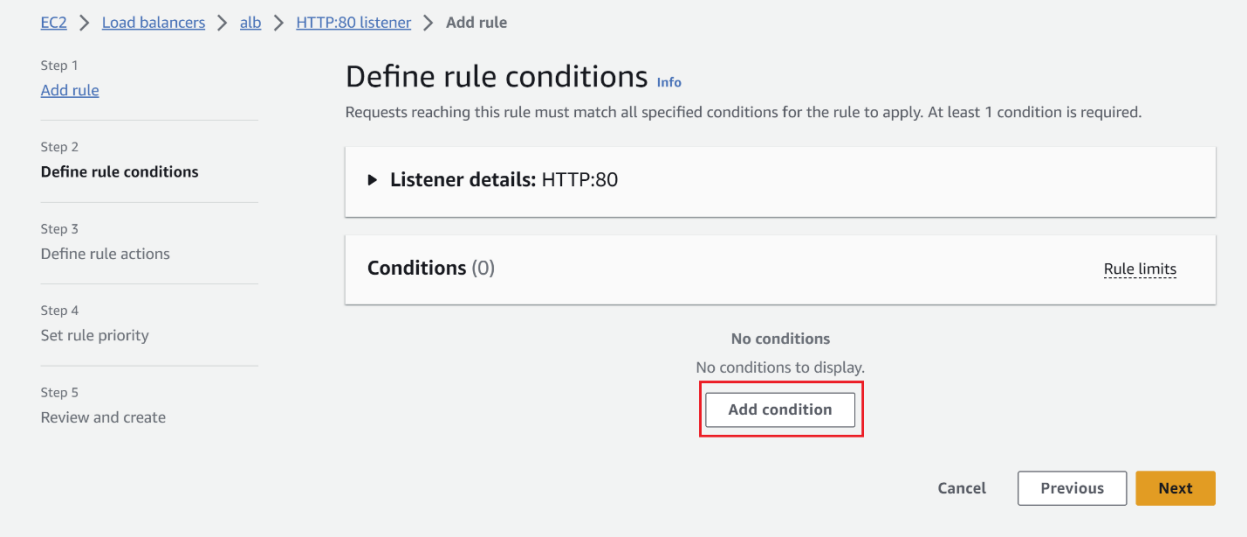
Edit ALB listener rule
Load balance>select you load balancer>go to listener and rules tab>click HTTP:80 >click Add rule >give a name>click Add condition >select path and writ a path for example /error>click confirm>click Next>select Return fixed response and select response code 404 and write a message for example not found, custom error >click next>write a priority for example 5>click next>click create

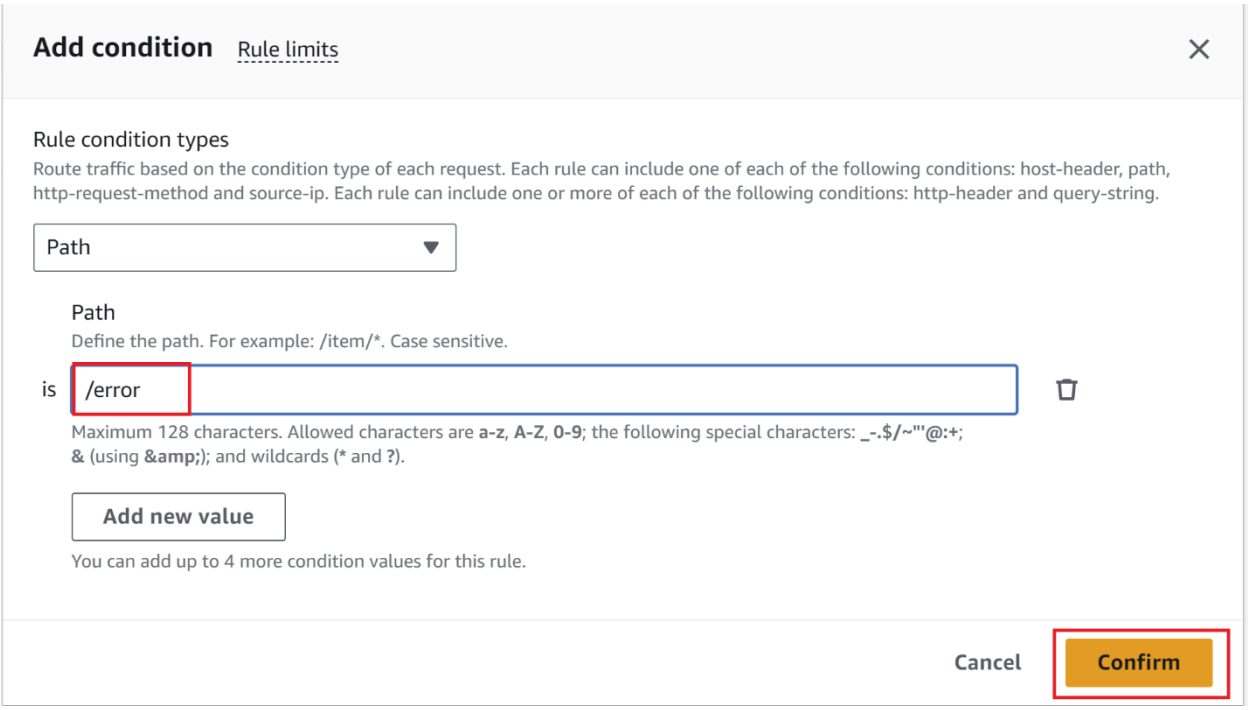
now we can browse load balancer/error and the message we set


EC2 > Load balancers
Load balancers (1/1)
Actions
Create load balancer
Name
DNS name
State
VPC ID
Availability Zones
Type
Date created
alb
Active
application
Load balancer: alb
Details
Listeners and rules
Network mapping
Security
Monitoring
Integrations
Attributes
Tags
Listeners and rules (1)
Manage rules
Manage listener
Add listener
Protocol:Port
Default action
Rules
ARN
Security policy
Default SSL/TLS certificate
HTTP:80
Forward to target group
tg-alb: 1 (100%)
Group-level stickiness: Off
1 rule
Not applicable
Not applicable


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.
Protocol:Port
HTTP:80
Load balancer
alb
Default actions
Forward to target group
tg-alb: 1 (100%)
Group-level stickiness: Off
Listener ARN
arn:aws:elasticloadbalancing:us-east-2:999838272208:listener/app/alb/3201d0ffec462fbc/624ee8bd429b00e4
Rules
Tags
Introducing the new Application Load Balancer listener rules experience
We've redesigned Application Load Balancer listener rules to be easier to use. The changes include:
Find rules quickly by giving them a name tag
Set rule priority with gaps in numerical sequence to accommodate future changes
Or you can use the old manage rules experience.
Listener rules (1)
Rule limits
Actions
Add rule
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
Tags
Default
Last (default)
If no other rule applies
Forward to target group
tg-alb: 1 (100%)
Group-level stickiness: Off
0 tags


EC2 > Load balancers > alb > 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
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
my-rule-a
Add additional tags
Cancel
Next


EC2 > Load balancers > alb > 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 conditions Info
Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.
Listener details: HTTP:80
Conditions (0)
Rule limits
No conditions
No conditions to display.
Add condition
Cancel
Previous
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.
Path
Path
Define the path. For example: /item/*. Case sensitive.
is
/error
Maximum 128 characters. Allowed characters are a-z, A-Z, 0-9; the following special characters: _-.$/~"'@:+; & (using &); and wildcards (* and ?).
Add new value
You can add up to 4 more condition values for this rule.
Cancel
Confirm

Define rule conditions Info

Requests reaching this rule must match all specified conditions for the rule to apply. At least 1 condition is required.

▶ **Listener details:** HTTP:80

Conditions (1) Rule limits Edit Delete Add condition

Path (1) Info

If
Path

is
/error

AND

Cancel Previous Next

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

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.

404

2xx, 4xx, 5xx

Content type - *optional*

The format of your message.

text/plain

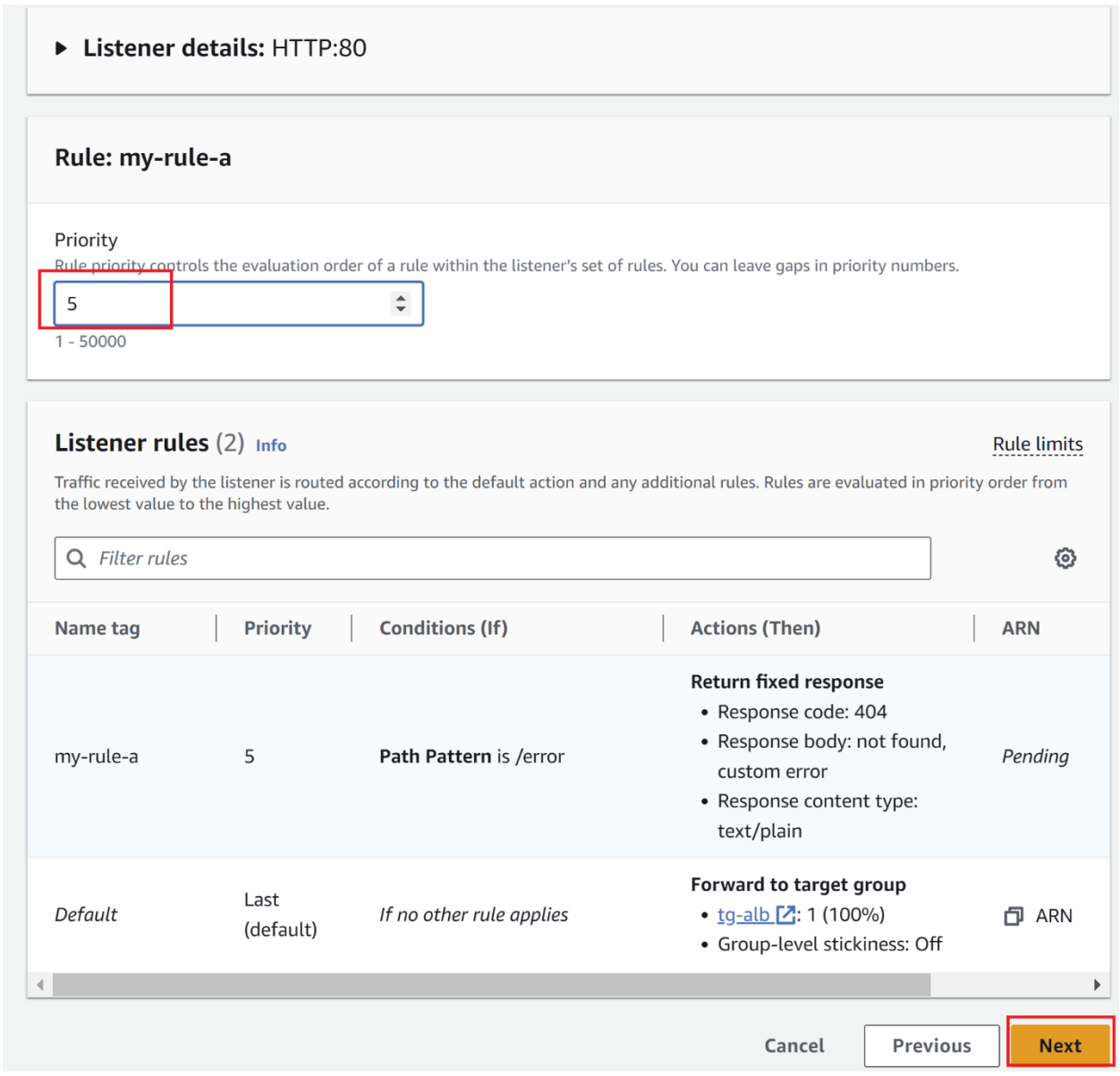
Response body - *optional*

Enter your response message.

not found, custom error

1024 character maximum

Cancel | Previous | Next


Listener details: HTTP:80
Rule: my-rule-a
Priority
Rule priority controls the evaluation order of a rule within the listener's set of rules. You can leave gaps in priority numbers.
5
1 - 50000
Listener rules (2) Info
Rule limits
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.
Filter rules
Name tag
Priority
Conditions (If)
Actions (Then)
ARN
my-rule-a
5
Path Pattern is /error
Return fixed response
Response code: 404
Response body: not found, custom error
Response content type: text/plain
Pending
Default
Last (default)
If no other rule applies
Forward to target group
tg-alb: 1 (100%)
Group-level stickiness: Off
ARN
Cancel
Previous
Next

Review and create

▶ **Listener details:** HTTP:80

Rule details: my-rule-a — Edit

Priority	Conditions (If)	Actions (Then)
5	If request matches all: **Path Pattern** is /error	**Return fixed response** • Response code: 404 • Response body: not found, custom error • Response content type: text/plain

Rule ARN

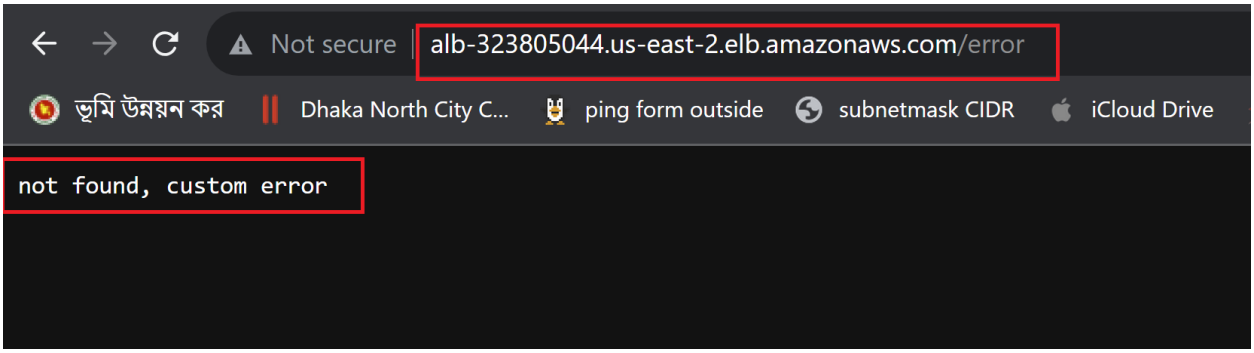
Pending

Rule tags (1) — Edit

Tags can help you manage, identify, organize, search for and filter resources.

Key	Value
Name	my-rule-a

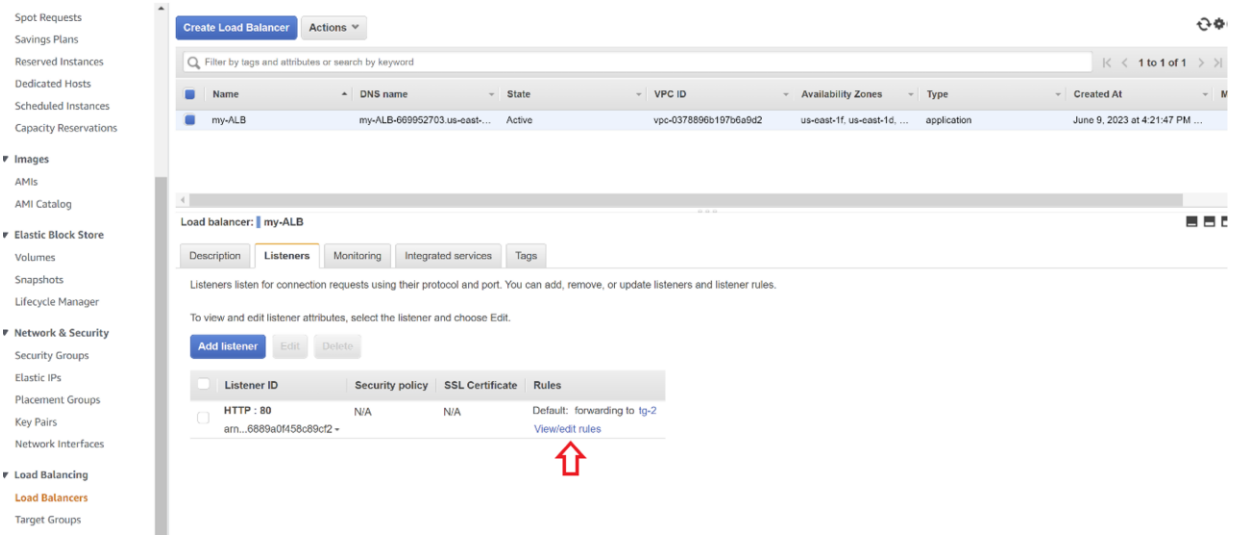
Cancel | Previous | Create

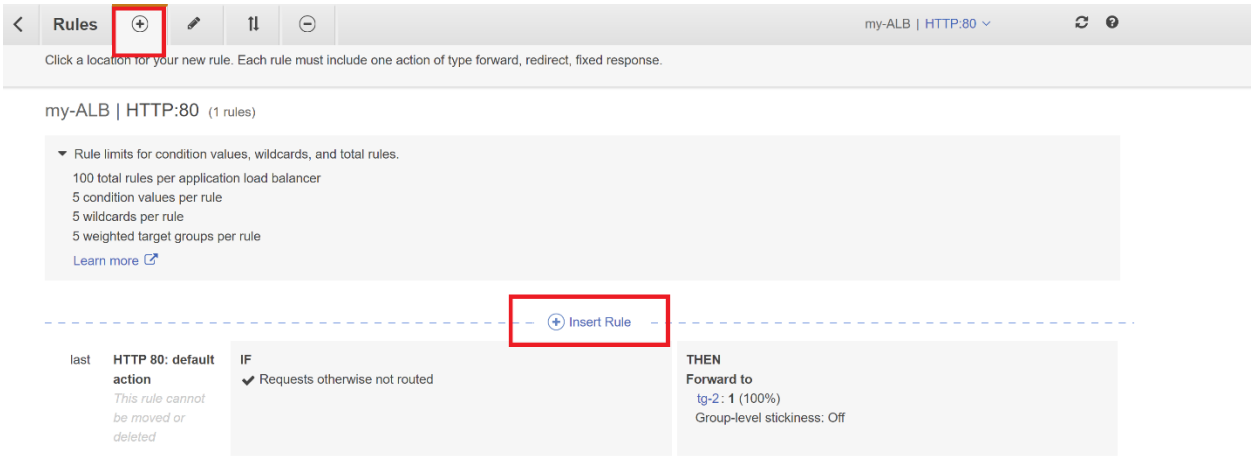


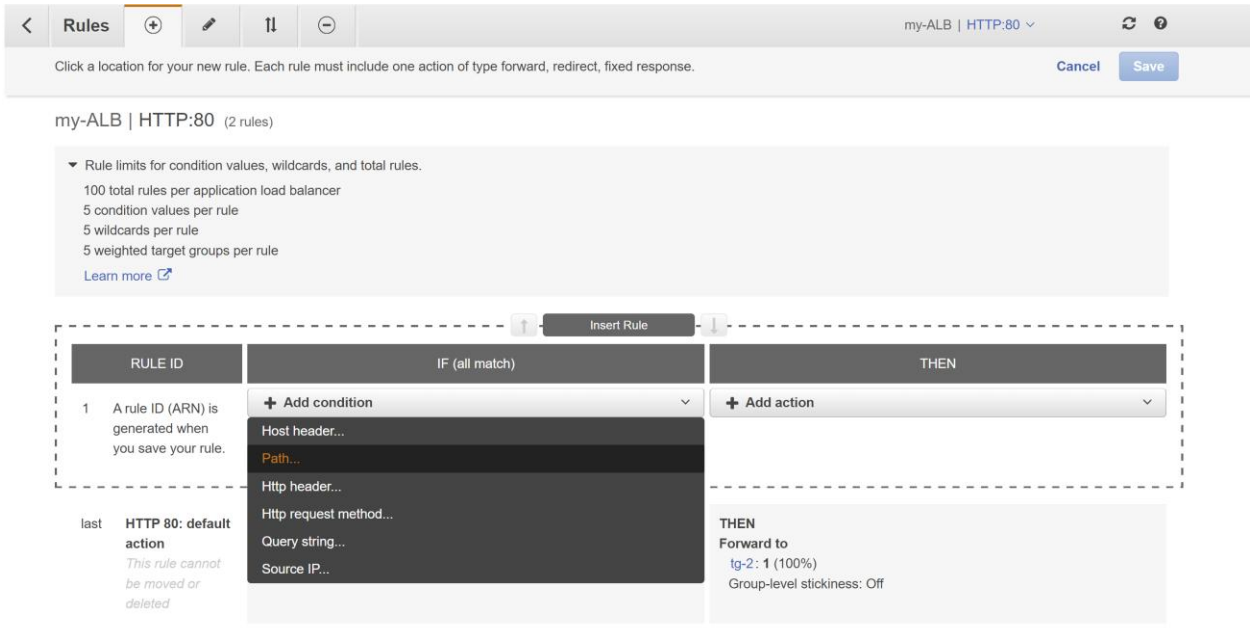
- Routing tables to different target groups:
  - Routing based on path in URL (example.com/users & example.com/posts)
  - Routing based on hostname in URL (one.example.com & other.example.com)
  - Routing based on Query String, Headers (example.com/users?id=123&order=false)

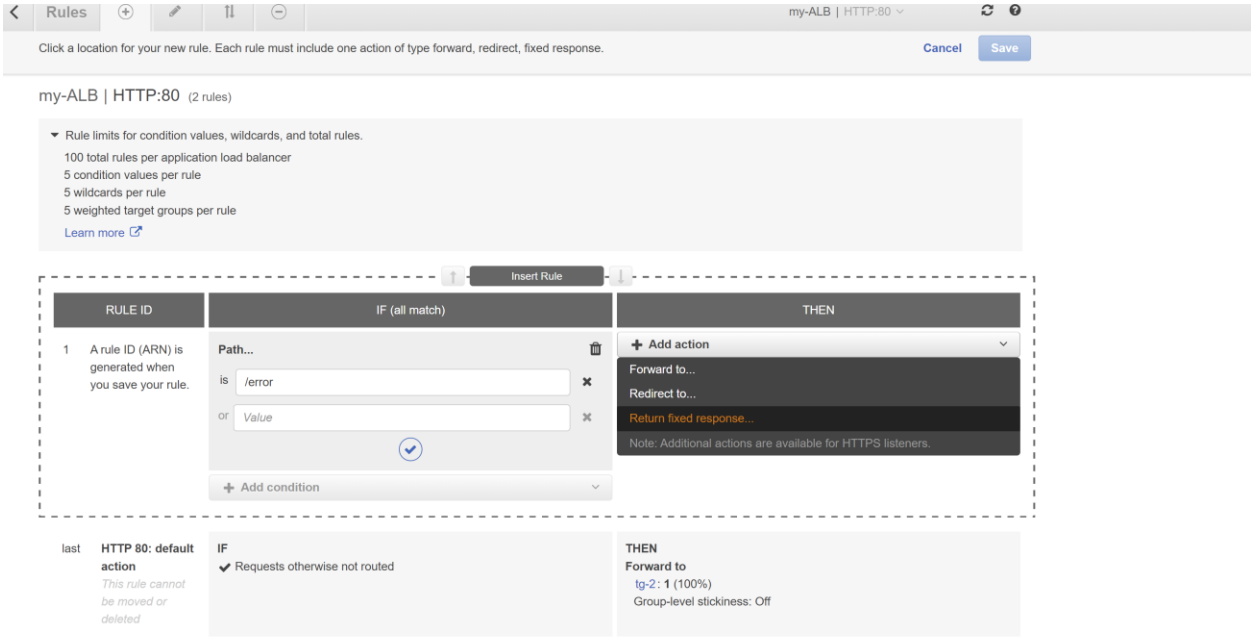
- ALB are a great fit for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port in ECS
- In comparison, we'd need multiple Classic Load Balancer per application

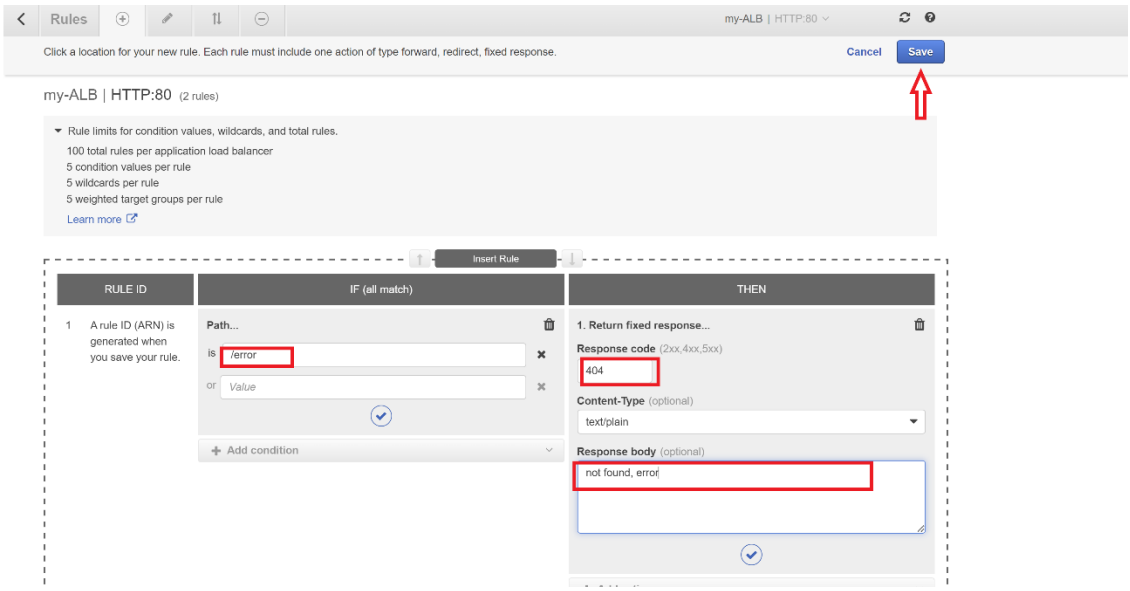
Old version rule create





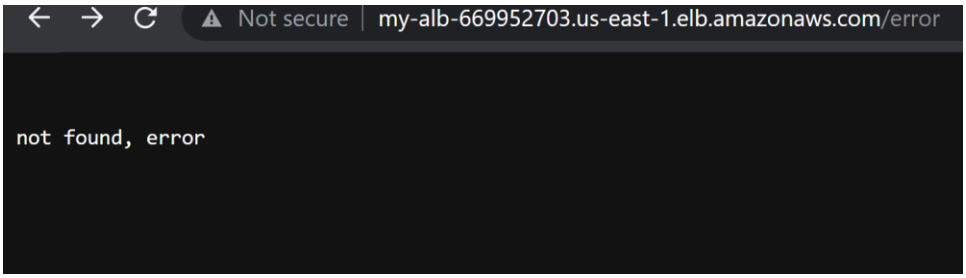

Rules
my-ALB | HTTP:80
Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response.
Cancel
Save
my-ALB | HTTP:80 (2 rules)
Rule limits for condition values, wildcards, and total rules.
100 total rules per application load balancer
5 condition values per rule
5 wildcards per rule
5 weighted target groups per rule
Learn more
Insert Rule
RULE ID
IF (all match)
THEN
1
A rule ID (ARN) is generated when you save your rule.
Add condition
Add action
Host header...
Path...
Http header...
Http request method...
Query string...
Source IP...
last
HTTP 80: default action
This rule cannot be moved or deleted
THEN
Forward to
tg-2: 1 (100%)
Group-level stickiness: Off


Rules
my-ALB | HTTP:80
Click a location for your new rule. Each rule must include one action of type forward, redirect, fixed response.
Cancel
Save
my-ALB | HTTP:80 (2 rules)
Rule limits for condition values, wildcards, and total rules.
100 total rules per application load balancer
5 condition values per rule
5 wildcards per rule
5 weighted target groups per rule
Learn more
Insert Rule
RULE ID
IF (all match)
THEN
1
A rule ID (ARN) is generated when you save your rule.
Path...
is
/error
or
Value
Add condition
Add action
Forward to...
Redirect to...
Return fixed response...
Note: Additional actions are available for HTTPS listeners.
last
HTTP 80: default action
This rule cannot be moved or deleted
IF
Requests otherwise not routed
THEN
Forward to
tg-2: 1 (100%)
Group-level stickiness: Off



Now click save button to save it and brows the dns with /error as follow

http://my-alb-669952703.us-east-1.elb.amazonaws.com/eeror

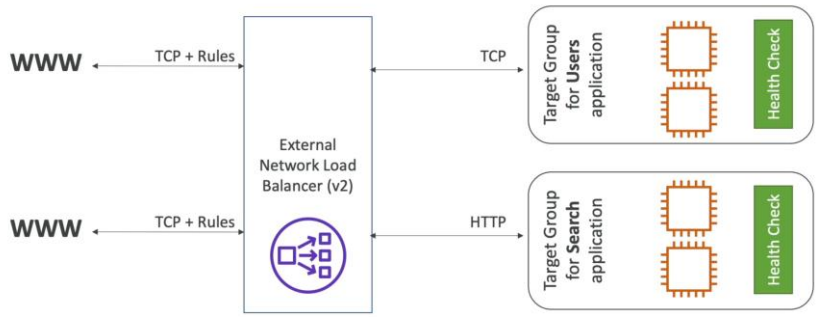


Network Load Balancer

- Network load balancers (Layer 4) allow to:
  - Forward TCP & UDP traffic to your instances
  - Handle millions of request per seconds
  - Less latency ~100 ms (vs 400 ms for ALB)

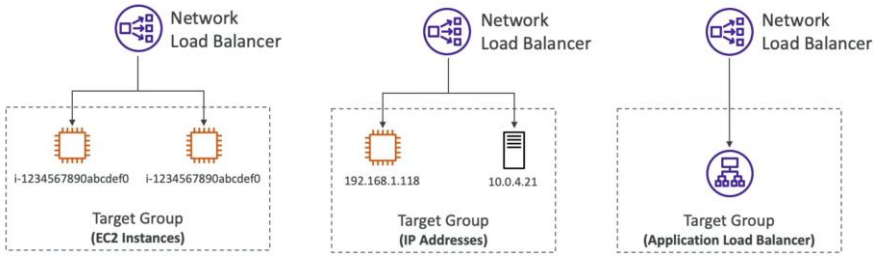
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)

- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier



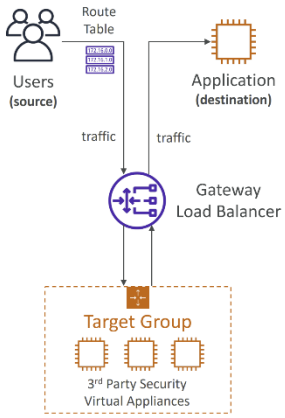
Network Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs
- Application Load Balancer
- Health Checks support the TCP, HTTP and HTTPS Protocols



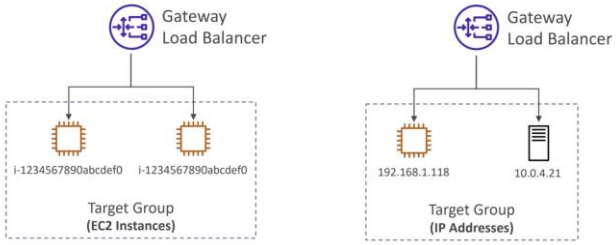
Gateway Load Balancer

- Deploy, scale, and manage a fleet of 3rd party network virtual appliances in AWS
- Example: Firewalls, Intrusion Detection and Prevention Systems, Deep Packet Inspection Systems, payload manipulation, …
- Operates at Layer 3 (Network Layer) – IP Packets
- Combines the following functions:
  - Transparent Network Gateway – single entry/exit for all traffic
  - Load Balancer – distributes traffic to your virtual appliances
- Uses the GENEVE protocol on port 6081



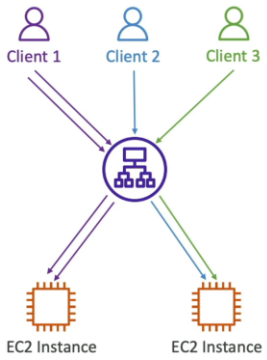
Gateway Load Balancer -Target group

- EC2 instances
- IP Addresses – must be private IPs



Sticky Sessions (Session Affinity)

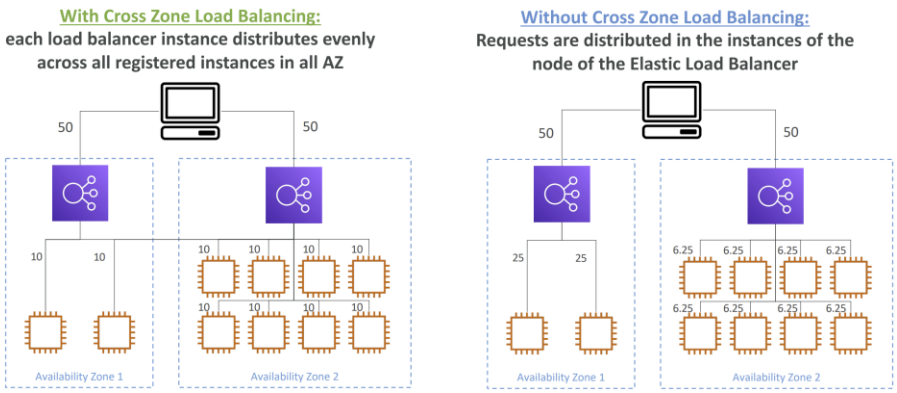
- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for Classic Load Balancer, Application Load Balancer, and Network Load Balancer
- The "cookie" used for stickiness has an expiration date you control
- Use case: make sure the user doesn't lose his session data
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances



Sticky Sessions – Cookies Names

- Application-based Cookies
  - Custom cookie
    - Generated by the target
    - Can include any custom attributes required by the application
    - Cookie name must be specified individually for each target group
    - Don't use AWSALB, AWSALBAPP, or AWSALBTG (reserved for use by the ELB)
  - Application cookie
    - Generated by the load balancer
    - Cookie name is AWSALBAPP
- Duration-based Cookies
  - Cookie generated by the load balancer
  - Cookie name is AWSALB for ALB, AWSELB for CLB

Cross Zone Load Balancer



Cross Zone Load balancing

- Application Load Balancer
  - Enabled by default (can be disabled at the Target Group level)
  - No charges for inter AZ data
- Network Load Balancer & Gateway Load Balancer
  - Disabled by default
  - You pay charges ($) for inter AZ data if enabled
- Classic Load Balancer
  - Disabled by default
  - No charges for inter AZ data if enabled

SSL/TLS

- An SSL Certificate allows traffic between your clients and your load balancer to be encrypted in transit (in-flight encryption)

- **SSL** refers to Secure Sockets Layer, used to encrypt connections
- **TLS** refers to Transport Layer Security, which is a newer version
- Nowadays, **TLS certificates are mainly used**, but people still refer as SSL

- Public SSL certificates are issued by Certificate Authorities (CA)
- Comodo, Symantec, GoDaddy, GlobalSign, Digicert, Letsencrypt, etc…

- SSL certificates have an expiration date (you set) and must be renewed

Load balancer – SSL Certificate

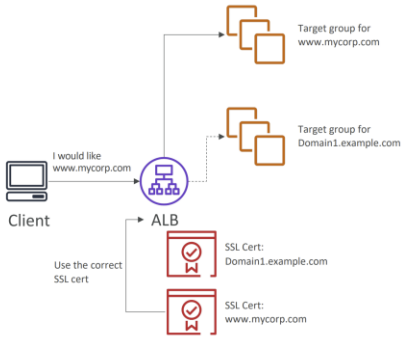
- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create upload your own certificates alternatively
- HTTPS listener:
  - You must specify a default certificate
  - You can add an optional list of certs to support multiple domains
  - **Clients can use SNI (Server Name Indication) to specify the hostname they reach**
  - Ability to specify a security policy to support older versions of SSL / TLS (legacy clients)

SSL server Name Indication (SNI)

- SNI solves the problem of loading **multiple SSL certificates onto one web server** (to serve multiple websites)
- It's a "newer" protocol, and requires the client to **indicate** the hostname of the target server in the initial SSL handshake
- The server will then find the correct certificate, or return the default one

Note:

- Only works for ALB & NLB (newer generation), CloudFront
- Does not work for CLB (older gen)



Elastic Load balancers – SSL Certificate

- Classic Load Balancer (v1)
  - Support only one SSL certificate
  - Must use multiple CLB for multiple hostname with multiple SSL certificates

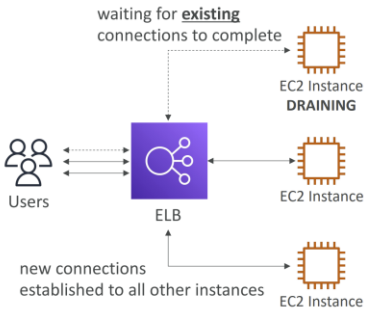
- Application Load Balancer (v2)
  - Supports multiple listeners with multiple SSL certificates
  - Uses Server Name Indication (SNI) to make it work

- Network Load Balancer (v2)
  - Supports multiple listeners with multiple SSL certificates
  - Uses Server Name Indication (SNI) to make it work

Connection draining

- Feature naming
  - Connection Draining – for CLB
  - Deregistration Delay – for ALB & NLB

- Time to complete "in-flight requests" while the instance is de-registering or unhealthy
- Stops sending new requests to the EC2 instance which is de-registering
- Between 1 to 3600 seconds (default: 300 seconds)
- Can be disabled (set value to 0)
- Set to a low value if your requests are short



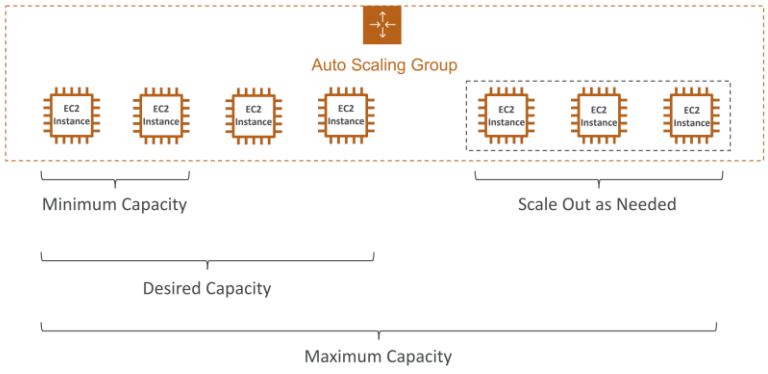
Auto scaling group

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly

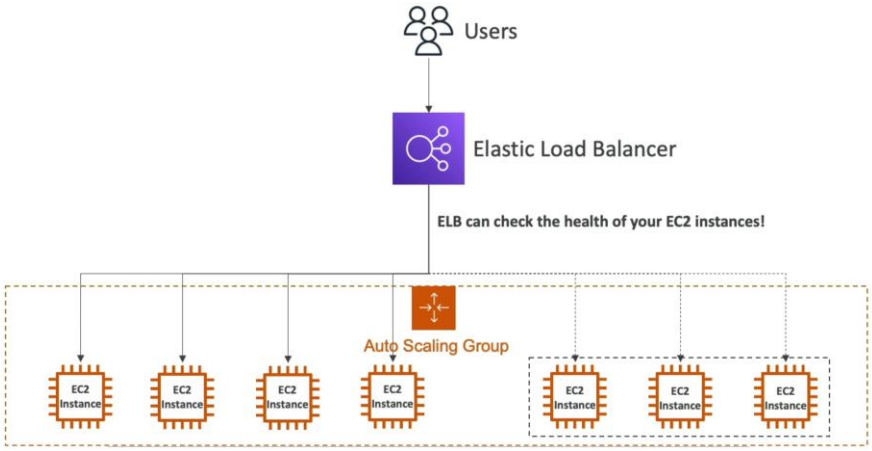
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of EC2 instances running
  - Automatically register new instances to a load balancer
  - Re-create an EC2 instance in case a previous one is terminated (ex: if unhealthy)

- ASG are free (you only pay for the underlying EC2 instances)

Auto scaling group in AWS



Auto scaling group in AWS with load balancer

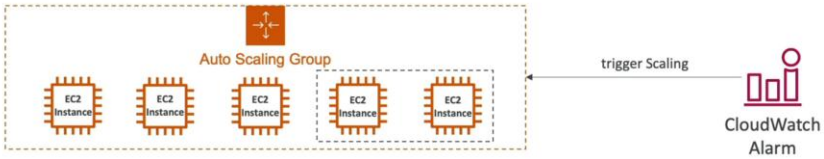


Auto scaling group attributes

- A **Launch Template** (older "Launch Configurations" are deprecated)
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
  - IAM Roles for your EC2 Instances
  - Network + Subnets Information
  - Load Balancer Information
- Min Size / Max Size / Initial Capacity
- Scaling Policies

Auto Scaling cloud watch Alarm & Scaling

- It is possible to scale an ASG based on CloudWatch alarms
- An alarm monitors a metric (such as **Average CPU**, or a **custom metric**)
- Metrics such as Average CPU are computed for the overall ASG instances
- Based on the alarm:
  - We can create scale-out policies (increase the number of instances)
  - We can create scale-in policies (decrease the number of instances)

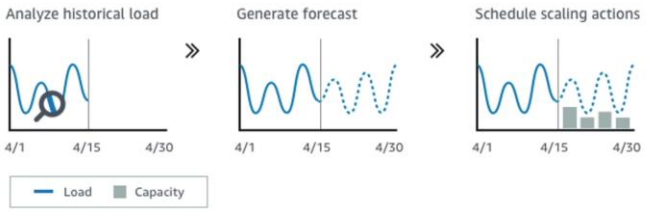


Auto scaling Groups – Dynamic Scaling Policies

- Target Tracking Scaling
  - Most simple and easy to set-up
  - Example: I want the average ASG CPU to stay at around 40%
- Simple / Step Scaling
  - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
  - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
- Scheduled Actions
  - Anticipate a scaling based on known usage patterns
  - Example: increase the min capacity to 10 at 5 pm on Fridays

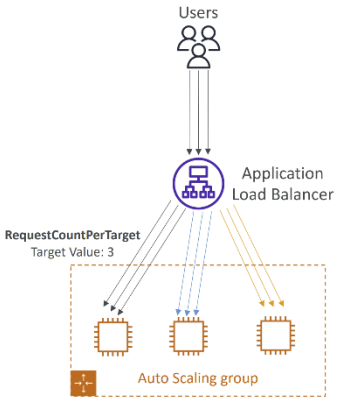
Auto Scaling Groups – Predictive Scaling

- **Predictive scaling:** continuously forecast load and schedule scaling ahead



Good metrics to scale on

- **CPUUtilization:** Average CPU utilization across your instances
- **RequestCountPerTarget:** to make sure the number of requests per EC2 instances is stable
- **Average Network In / Out** (if you're application is network bound)
- **Any custom metric** (that you push using CloudWatch)



Auto scaling groups – scaling cooldown

- After a scaling activity happens, you are in the **cooldown period (default 300 seconds)**
- During the cooldown period, the ASG will not launch or terminate additional instances (to allow for metrics to stabilize)
- Advice: Use a ready-to-use AMI to reduce configuration time in order to be serving request fasters and reduce the cooldown period