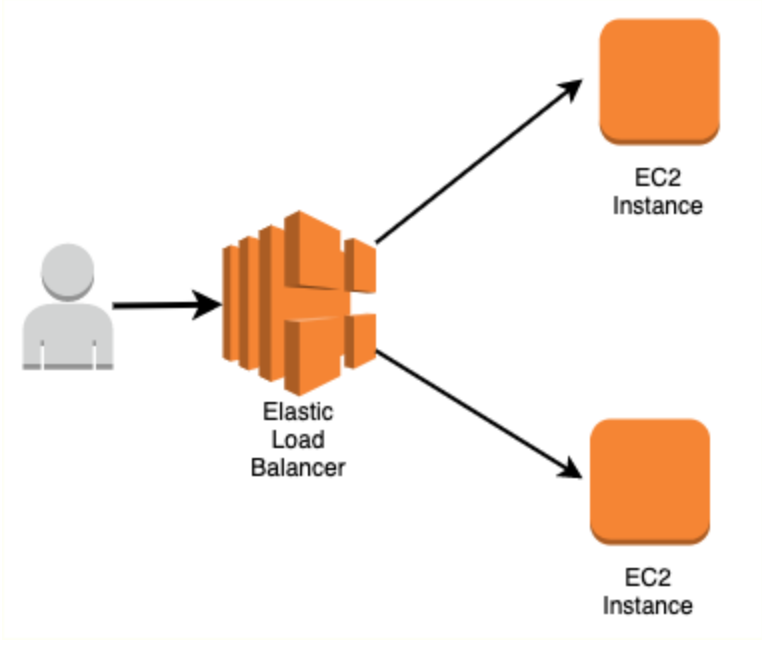
Load Balancing

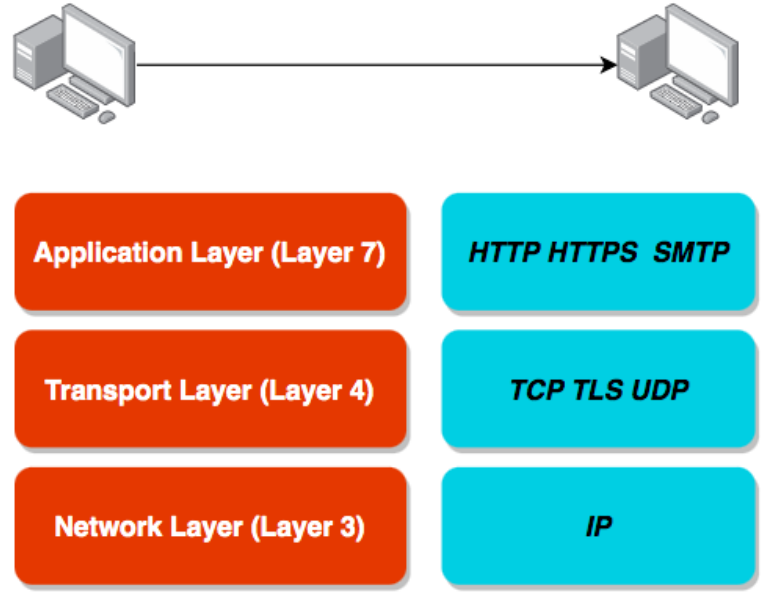
Elastic Load Balancer

- Distribute traffic across EC2 instances in one or more AZs in a single region
- **Managed service** - AWS ensures that it is highly available
- Auto scales to handle huge loads
- Load Balancers can be **public or private**
- **Health checks** - route traffic to healthy instances



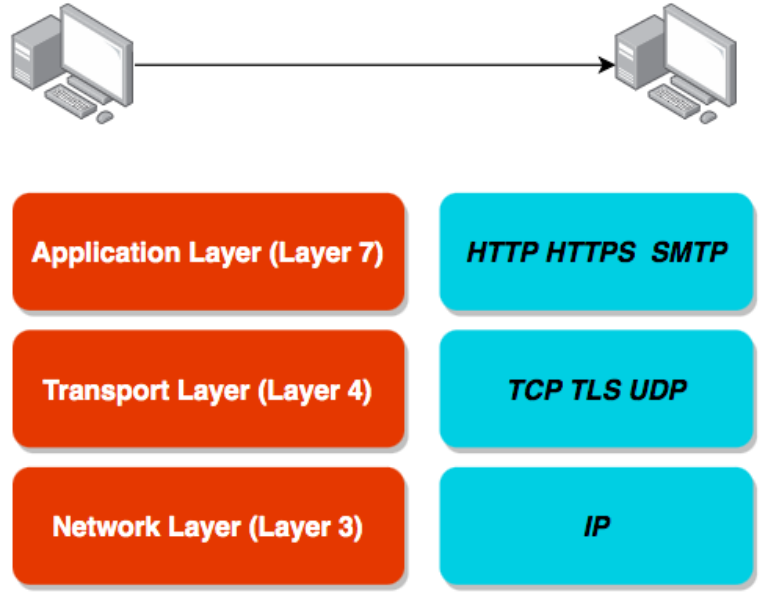
HTTP vs HTTPS vs TCP vs TLS vs UDP

- Computers use protocols to communicate
- Multiple layers and multiple protocols
- **Network Layer** - Transfer bits and bytes
- **Transport Layer** - Are the bits and bytes transferred properly?
- **Application Layer** - Make REST API calls and Send Emails
- (Remember) Each layer makes use of the layers beneath it
- (Remember) Most applications talk at application layer. BUT some applications talk at transport layer directly(high performance).



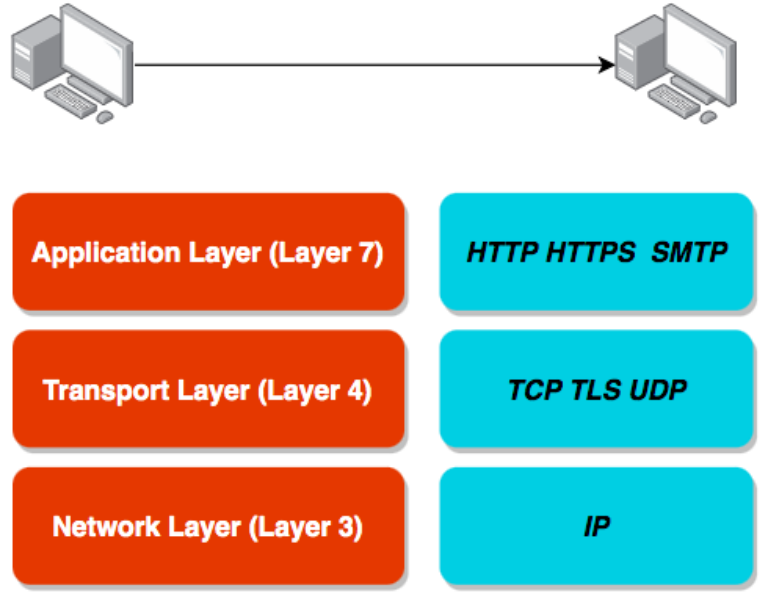
HTTP vs HTTPS vs TCP vs TLS vs UDP

- Network Layer:
  - IP (Internet Protocol): Transfer bytes. Unreliable.
- Transport Layer:
  - TCP (Transmission Control): Reliability > Performance
  - TLS (Transport Layer Security): Secure TCP
  - UDP (User Datagram Protocol): Performance > Reliability
- Application Layer:
  - HTTP(Hypertext Transfer Protocol): Stateless Request Response Cycle
  - HTTPS: Secure HTTP
  - SMTP: Email Transfer Protocol
  - and a lot of others...



HTTP vs HTTPS vs TCP vs TLS vs UDP

- **Most applications** typically communicate at application layer
  - Web apps/REST API(HTTP/HTTPS), Email Servers(SMTP), File Transfers(FTP)
  - All these applications use TCP/TLS at network layer(for reliability)
- **HOWEVER** applications needing high performance **directly** communicate at transport layer:
  - Gaming applications and live video streaming use UDP (sacrifice reliability for performance)
- **Objective**: Understand Big Picture. Its OK if you do not understand all details.



Three Types of Elastic Load Balancers

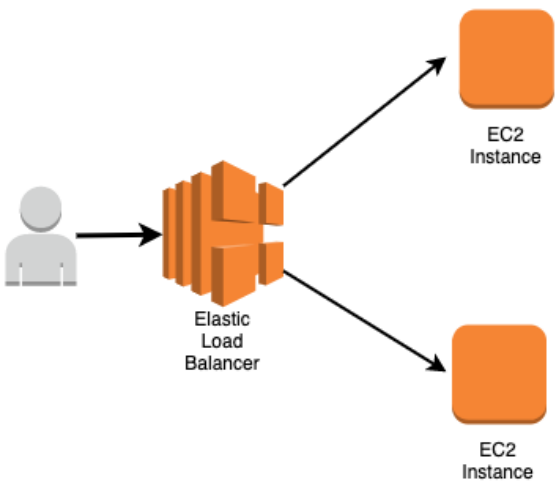
- **Classic** Load Balancer (Layer 4 and Layer 7)
  - Old generation supporting Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS) protocols
  - Not Recommended by AWS
- **Application** Load Balancer (Layer 7)
  - New generation supporting HTTP/HTTPS and advanced routing approaches
- **Network** Load Balancer (Layer 4)
  - New generation supporting TCP/TLS and UDP
  - Very high performance usecases

Classic Load Balancer

- **Older** version of ELB
- **Not recommended anymore**
- Supports TCP, SSL/TLS and HTTP(S) (Layer 4 and 7)
- **Demo**: Create a Classic Load Balancer

Application Load Balancer

- **Most popular** and frequently used ELB in AWS
- Supports WebSockets and HTTP/HTTPS (Layer 7)
- Supports all important load balancer features
- Scales **automatically** based on demand (Auto Scaling)
- Can load balance between:
  - EC2 instances (AWS)
  - Containerized applications (Amazon ECS)
  - Web applications (using IP addresses)
  - Lambdas (serverless)
- **Demo** : Create an Application Load Balancer



Load Balancers - Security Group Best Practice

Load Balancer allow traffic from everywhere!

Inbound rules

Type	Protocol	Port range	Source
HTTP	TCP	80	0.0.0.0/0

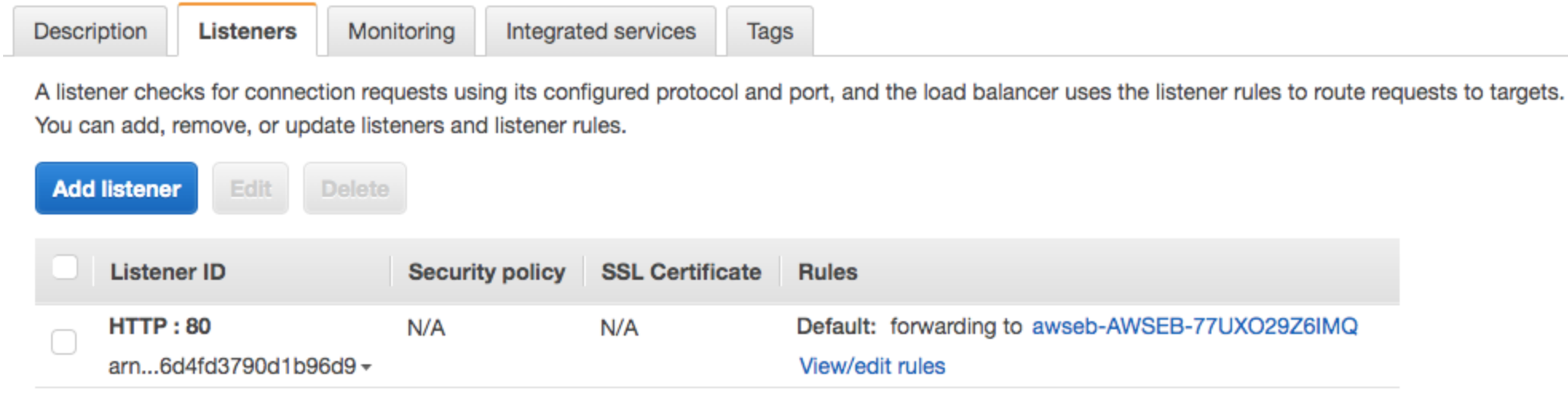
EC2 Security Group **ONLY** allows traffic from Load Balancer Security Group

Inbound rules

Type	Protocol	Port range	Source
HTTP	TCP	80	sg-03eb042440351fdad (awseb-e-grzepvhpv3-stack-AWSEBLoadBalancerSecurityGroup-1EG2SPQRTIQ02)

(Best Practice) Restrict allowed traffic using Security Groups

Listeners



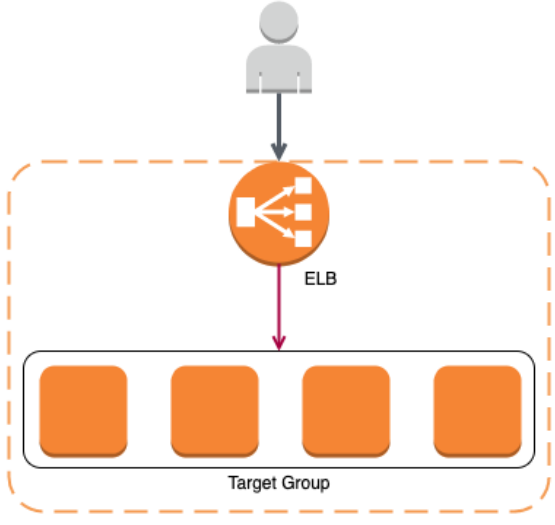
- Each Load Balancer has **one or more listeners** listening for connection requests from the client
- Each listener has:
  - a protocol
  - a port
  - a set of rules to route requests to targets

Multiple Listeners

- You can have multiple listeners listening for a different protocol or port
- In the above example:
  - HTTP requests on port 80 are routed to the EC2 instances target group
  - HTTPS requests on port 443 are routed to port 80
  - HTTP requests on port 8080 get a fixed response (customized HTML)

Target Groups

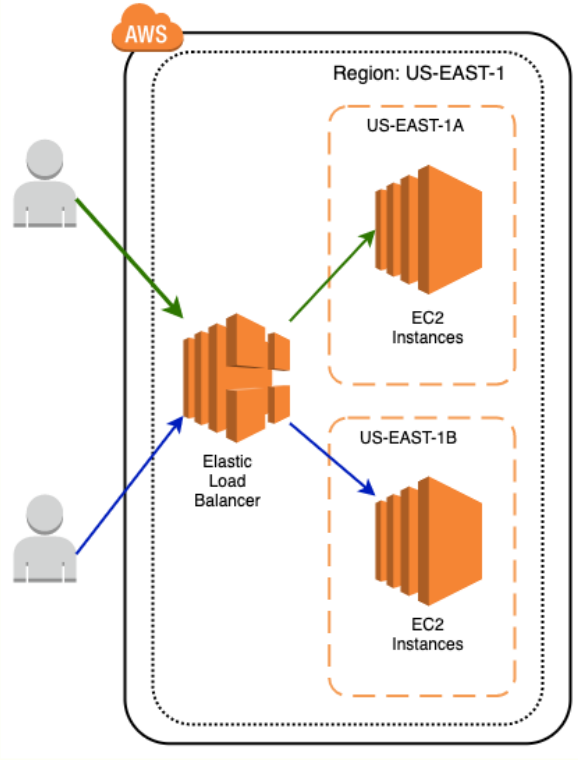
- How to group instances that ALB has to distribute the load between?
  - Create a Target Group
- A target group can be:
  - A set of EC2 instances
  - A lambda function
  - Or a set of IP addresses



Target Group Configuration - Sticky Session

Enable sticky user sessions

- Send all requests from one user to the same instance
- Implemented using a cookie
- Supported by ALB and CLB



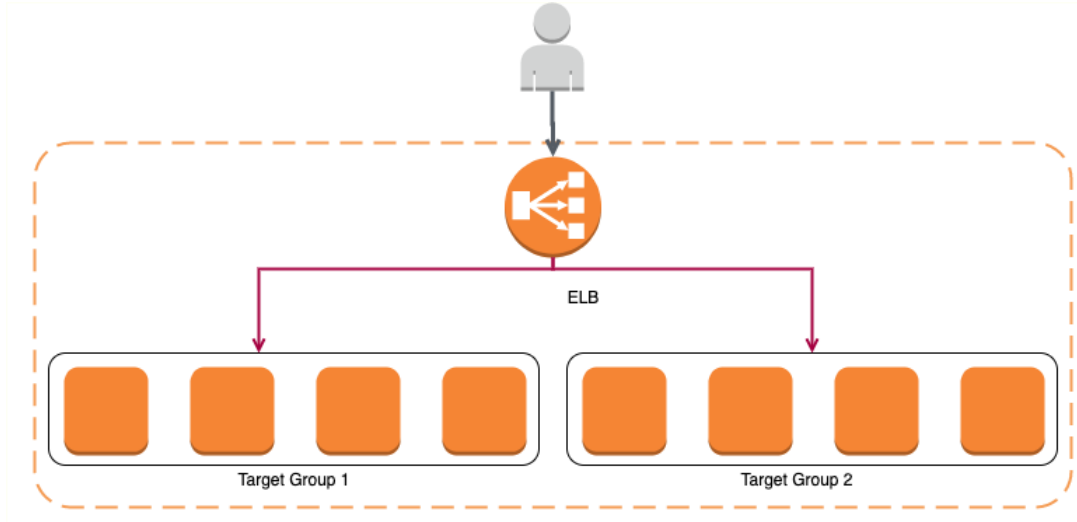
Target Group Configuration - Deregistration delay

How long should ELB wait before de-registering a target?

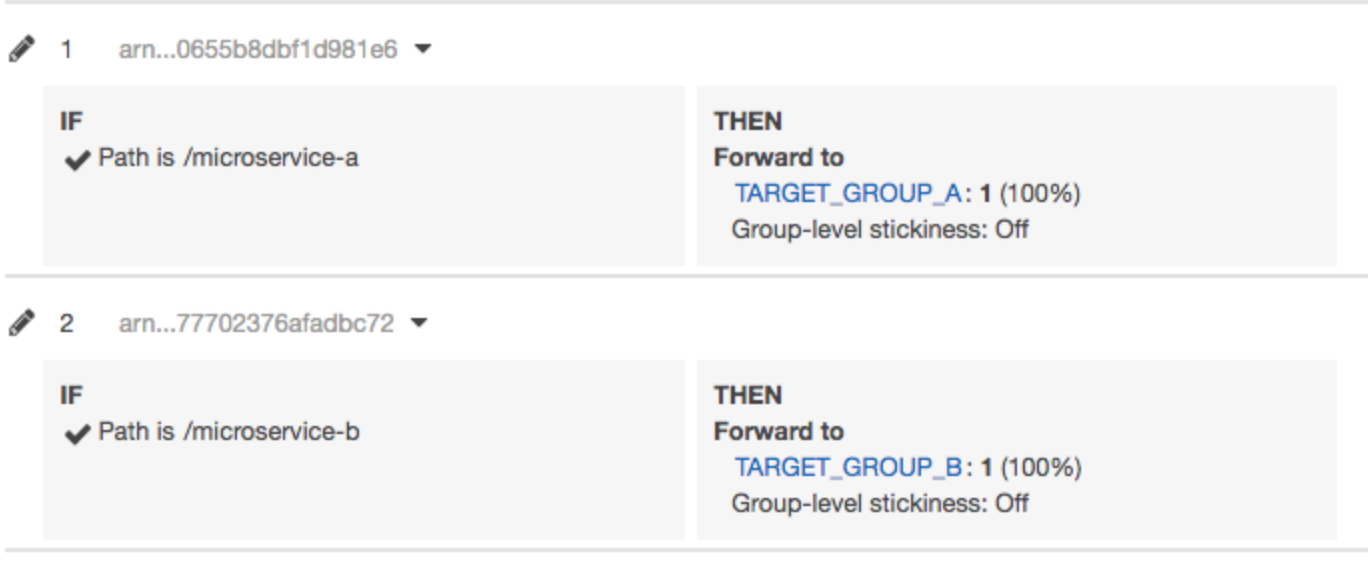
- Load balancer stops routing new requests to a target when you unregister it
- What about requests that are **already in progress** with that target?
- This setting ensures that load balancer gives **in-flight requests** a chance to complete execution
- 0 to 3600 seconds (default 300 seconds)
- Also called Connection Draining

Microservices architectures - Multiple Target Group(s)

- Microservices architectures have 1000s of microservices
  - *http://www.xyz.com/microservice-a*
  - *http://www.xyz.com/microservice-b*
- Should we create multiple ALBs?
- **Nope**. One ALB can support multiple microservices!
- Create separate target group for each microservices
- (Remember) Classic Load Balancer, **does NOT** support multiple target groups.

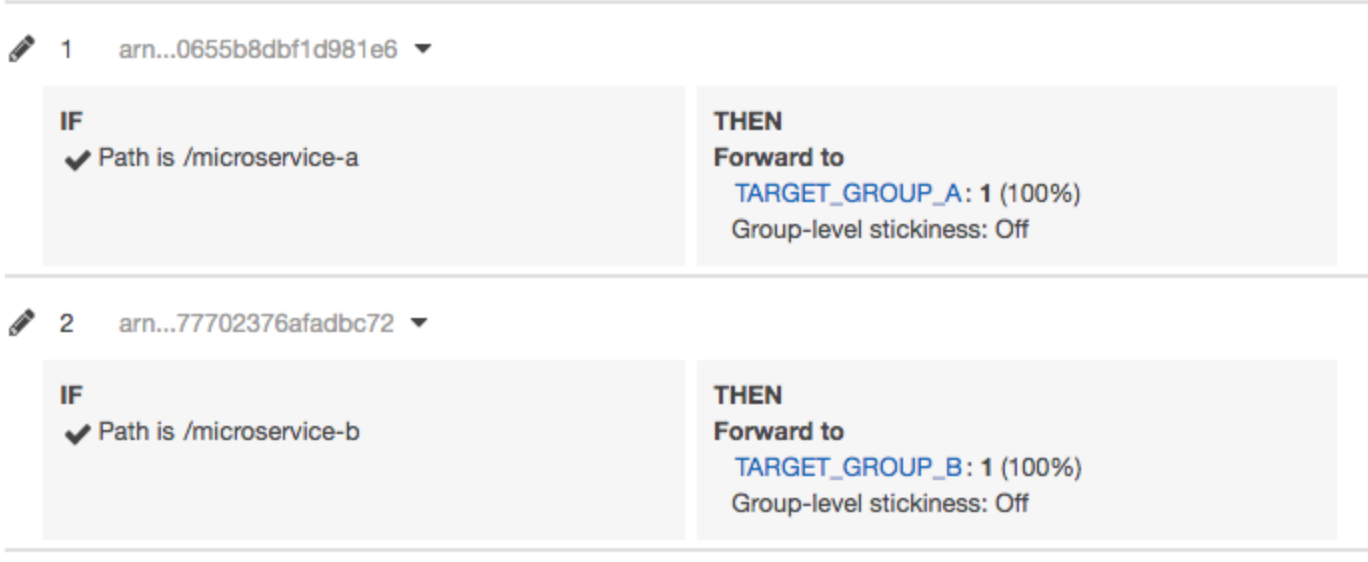


Listener Rules



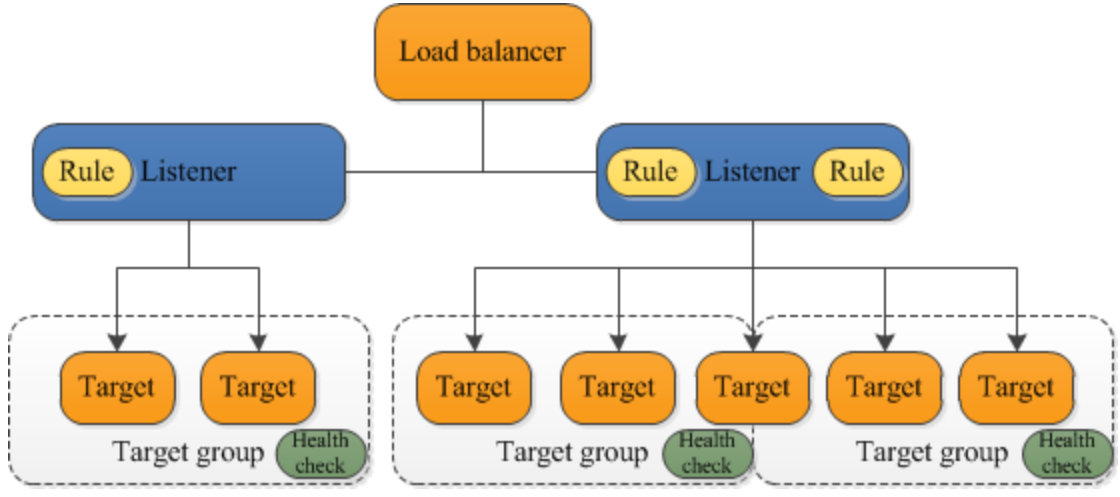
- How do I identify which request should be sent to which target group?
- Configure multiple listener rules for the same listener
- Rules are executed in the order they are configured.
- Default Rule is executed last.

Listener Rules - Possibilities



- Based on **path** - in28minutes.com/a to target group A and in28minutes.com/b to target group B
- Based on **Host** - a.in28minutes.com to target group A and b.in28minutes.com to target group B
- Based on **HTTP headers** (Authorization header) and methods (POST, GET, etc)
- Based on **Query Strings** (/microservice?target=a, /microservice?target=b)
- Based on **IP Address** - all requests from a range of IP address to target group A. Others to target group B

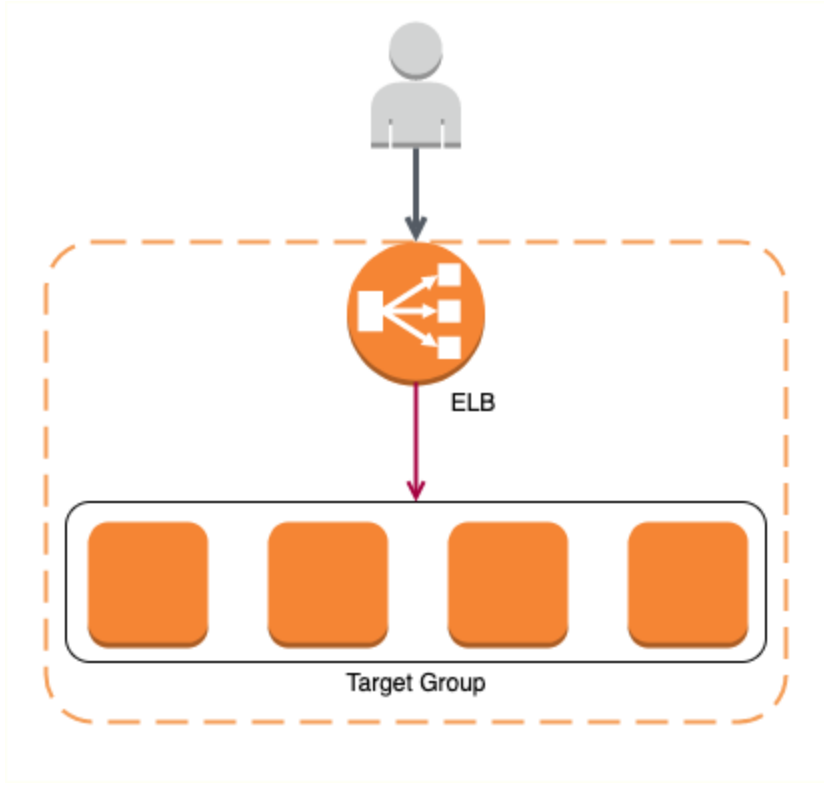
Architecture Summary



https://docs.amazonaws.cn/en_us/elasticloadbalancing/latest/application/introduction.html

- Highly decoupled architecture
- Load balancer can have multiple listeners (protocol + port combinations).
- Each listener can have multiple rules each routing to a target group based on request content.
- A target can be part of multiple target groups.

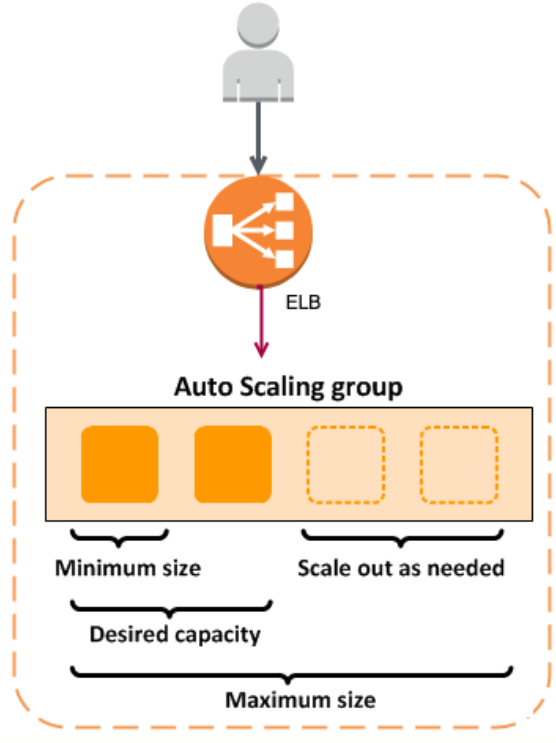
Introducing Auto Scaling Groups



- Target Groups are configured with a static set of instances. How do you scale out and scale in **automatically**?
  - Configure a Auto Scaling Group

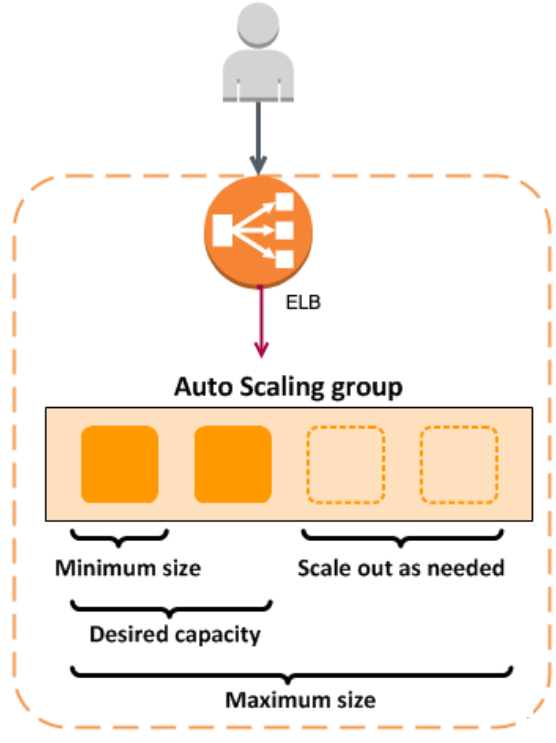
Auto Scaling Groups

- Auto Scaling Group responsibilities:
  - **Maintain** configured number of instances (using periodic health checks)
    - If an instance goes down, ASG launches replacement instance
  - **Auto scale** to adjust to load (scale-in and scale-out based on auto scaling policies)
- ASG can launch On-Demand Instances, Spot Instances, or both
  - **Best Practice**: Use Launch Template
- An ELB can distribute load to **active instances** as ASG expands and contracts based on the load
- **DEMO:** Creating Auto Scaling Groups

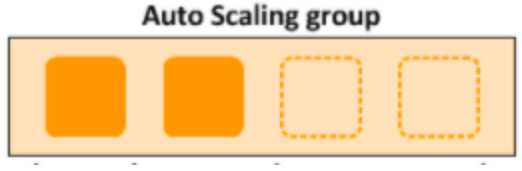


Auto Scaling Components

- **Launch Configuration/Template**
  - EC2 instance size and AMI
- **Auto Scaling Group**
  - Reference to Launch Configuration/Template
  - Min, max and desired size of ASG
  - EC2 health checks by default. Optionally enable ELB health checks.
  - **Auto Scaling Policies**
    - When and How to execute scaling?

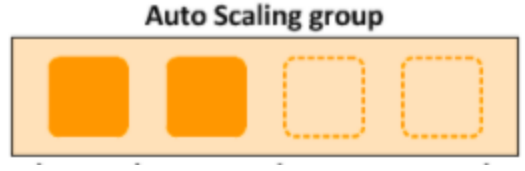


Auto Scaling Group - Use Cases



ASG Use case	Description	More details
Maintain current instance levels at all times	min = max = desired = CONSTANT When an instance becomes unhealthy, it is replaced.	Constant load
Scale manually	Change desired capacity as needed	You need complete control over scaling
Scale based on a schedule	Schedule a date and time for scaling up and down.	Batch programs with regular schedules
Scale based on demand (Dynamic/Automatic Scaling)	Create scaling policy (what to monitor?) and scaling action (what action?)	Unpredictable load

Dynamic Scaling Policy Types

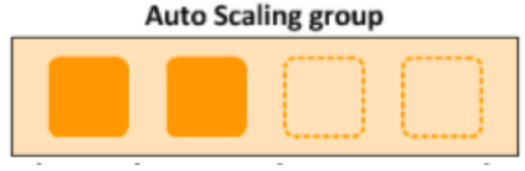


Scaling Policy	Example(s)	Description
Target tracking scaling	Maintain CPU Utilization at 70%.	Modify current capacity based on a target value for a specific metric.
Simple scaling	+5 if CPU utilization > 80% -3 if CPU utilization < 60%	Waits for cooldown period before triggering additional actions.
Step scaling	+1 if CPU utilization between 70% and 80% +3 if CPU utilization between 80% and 100% Similar settings for scale down	Warm up time can be configured for each instance

Scaling Policies - Background

- Two parts:
  - CloudWatch alarm (Is CPU utilization >80%? or < 60%).
  - Scaling action (+5 EC2 instances or -3 EC2 instances)

Auto Scaling - Scenarios



Scenario	Solution
Change instance type or size of ASG instances	Launch configuration or Launch template cannot be edited. Create a new version and ensure that the ASG is using the new version. Terminate instances in small groups.
Roll out a new security patch (new AMI) to ASG instances	Same as above.
Perform actions before an instance is added or removed	Create a Lifecycle Hook. You can configure CloudWatch to trigger actions based on it.

Auto Scaling - Scenarios

Scenario	Solution
Which instance in an ASG is terminated first when a scale-in happens?	(**Default Termination Policy**) Within constraints, goal is to distribute instances evenly across available AZs. Next priority is to terminate older instances.
Preventing frequent scale up and down	Adjust cooldown period to suit your need (default - 300 seconds). Align CloudWatch monitoring interval
I would want to protect newly launched instances from scale-in	Enable instance scale-in protection

Network Load Balancer

- Functions at the **Transport Layer** - Layer 4 (Protocols TCP, TLS and UDP)
- For **high performance** use cases (millions of requests per second)
- Can be assigned a **Static IP/Elastic IP**
- Can load balance between:
  - EC2 instances
  - Containerized applications (Amazon ECS)
  - Web applications (using IP addresses)
- Demo

Review

Elastic Load Balancer

- Distribute traffic across EC2 instances in one or more AZs in a single region
- **Managed Service** - highly available, Auto scales, public or private

Classic Load Balancer

- Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS)
- **Old**. Not Recommended by AWS

Network Load Balancer

- Layer 4(TCP/TLS and UDP)
- Very **high performance usecases**
- Can be assigned a Static IP/Elastic IP

Review

Application Load Balancer

- Layer 7(HTTP/HTTPS)
- Supports **advanced routing approaches** (path, host, http headers, query strings and origin IP addresses)
- Load balance between EC2 instances, containers, IP addresses and lambdas

Concepts

- Each Load Balancer has one or more **listeners** (different protocol or port) listening for connection requests from the client
- **Target group** is a group representing the targets (ex: EC2 instances)
- One ALB or NLB can support multiple microservices (multiple target groups)!

Review

Concepts

- **Auto Scaling Group** - Maintain configured number of instances (using periodic health checks). Auto scale to adjust to load.
- **Dynamic Scaling Policies** - Target tracking scaling, Simple scaling and Step scaling.
- **CloudWatch alarms** track the metric (Is CPU utilization >80%? or < 60%) and trigger the auto scaling action (+5 EC2 instances or -3 EC2 instances)