AWS Fundamentals: ELB + ASG

Scalability & High Availability

Scalability – means that an application / system can handle greater loads by adapting.

Two kinds of scalability:

- Vertical Scalability
- Horizontal Scalability (= elasticity)

- Scalability is linked by different to High Availability

Vertical Scalability

- Vertically scalability means increasing the size of the instance
  - Ex: 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 database
- RDS, ElastiCache are services that can scale vertically
- There's usually a limit to how much you can vertically scale (hardwire 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 like Amazon EC2

High 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)

High Availability & Scalability for EC2

- Vertical Scaling: Increase instance size (= scale up / down)
    - From: t2.nano – 0.5G of RAM, I vCPU
    - To: u-I2tb.I.metal – 12.3 TB of RAM, 448 vCPUs

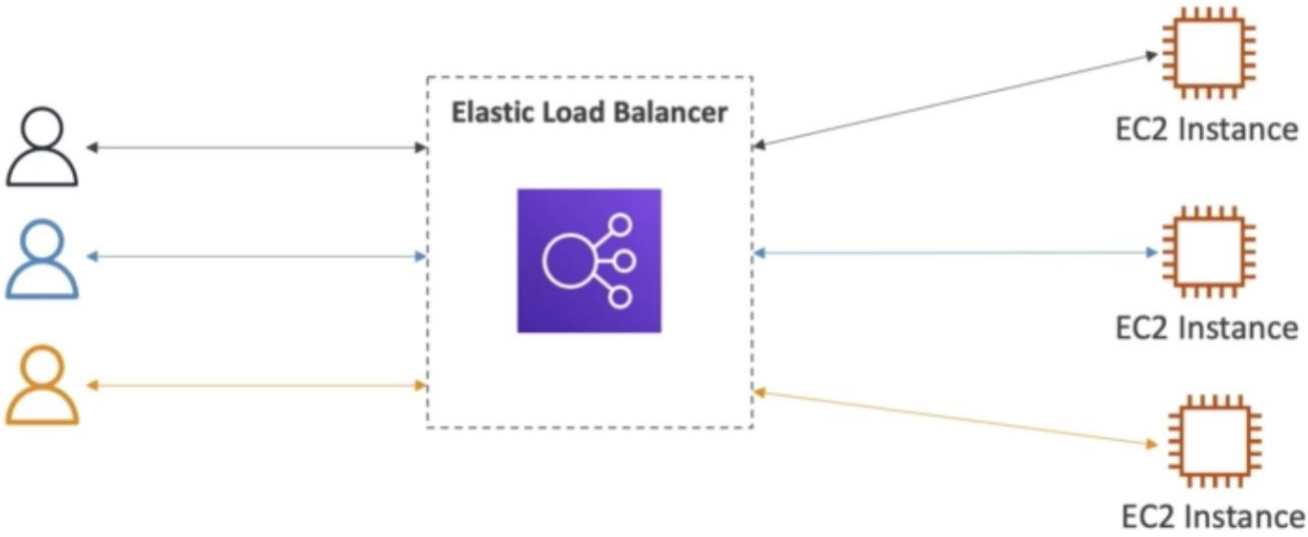
- Horizontal Scaling: Increase number of instances (= scale out / in)
    - Auto Scaling Group
    - Load Balancer

- 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 (EC2 instances) downstream.

More users there you have; the more load is going to be balanced across EC2 instances



Why use a load balancer?

1. Spread load across multiple downstream instances
2. Expose a single point of access (DNS) to your application

3. Seamlessly handle failures of downstream instances
4. Do regular health checks to your instances
5. Provide SSL termination (HTTPS) for your websites
6. Enforce stickiness with cookies
7. High availability across zones
8. 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 configurations knobs

- It costs less to setup your own load balancer, but it will be 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 Checks

- 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 route (/health is common)
- If the response is not 200 (OK), then the instance is unhealth.

Types of load balancer on AWS

AWS has 4 kinds of managed Load Balancers

- **Classic Load Balancer** (v l – old generation) – *2009* – CLB
    - HTTP, HTTPS, TCP, SSL (secure TCP)
- **Application Load Balancer** (v 2 – 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's 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

Classic Load Balancers (v 1)

- Supports TCP (Layer4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname
- XXX.region.elb.amazonaws.com

Application Load Balancer (v2)

- 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)

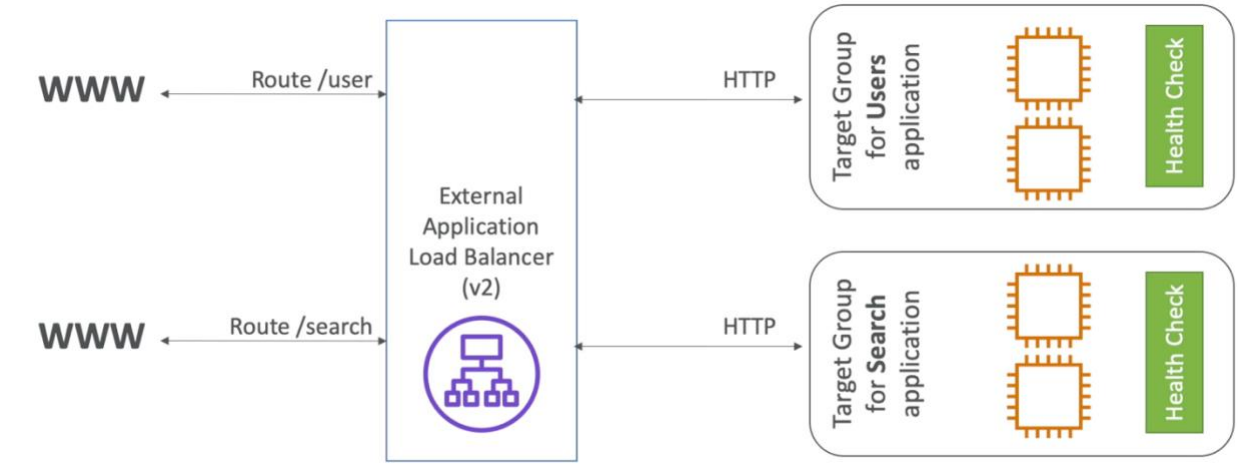
Routing tables to different target groups:

- 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

Example Graph - Application Load Balancer (v2) (ALB)

HTTP Based Traffic

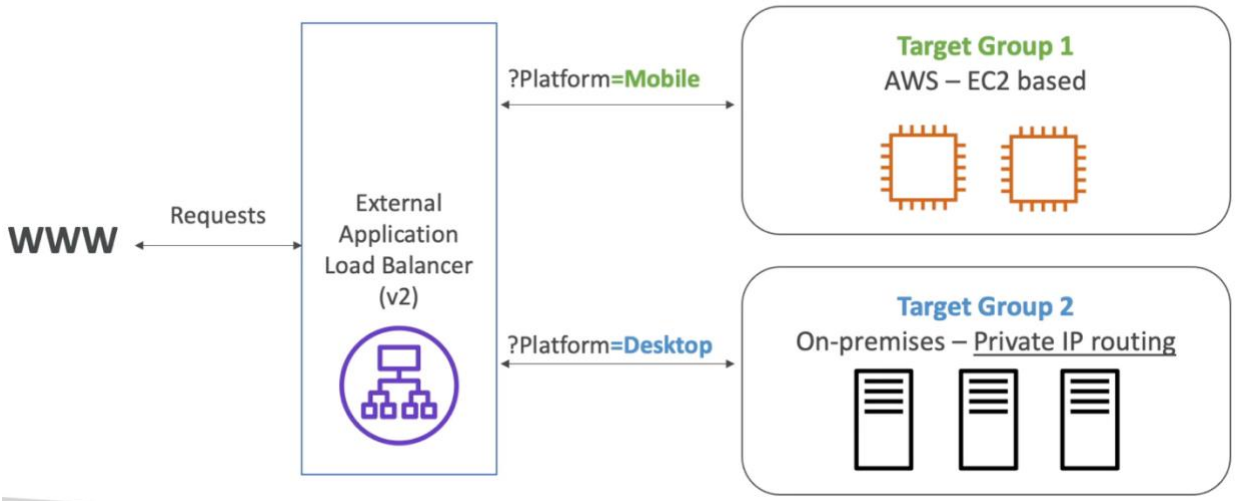


Target Groups

- 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

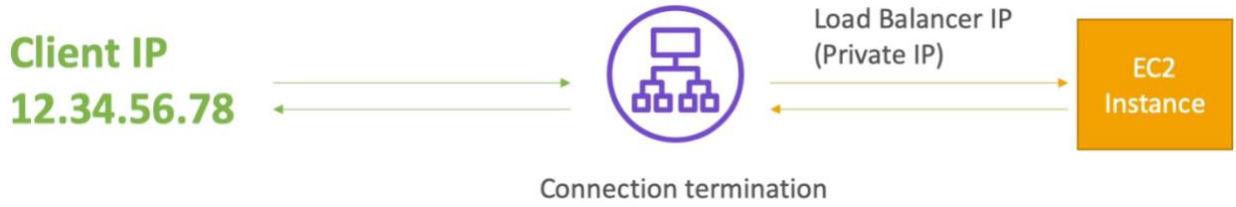
- ALB can route to multiple target groups
- Health checks are at the target group level

Query Strings & Parameters Routing



Good to know*

- 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-Forward-Port) and proto (X-Forwarded-Proto)



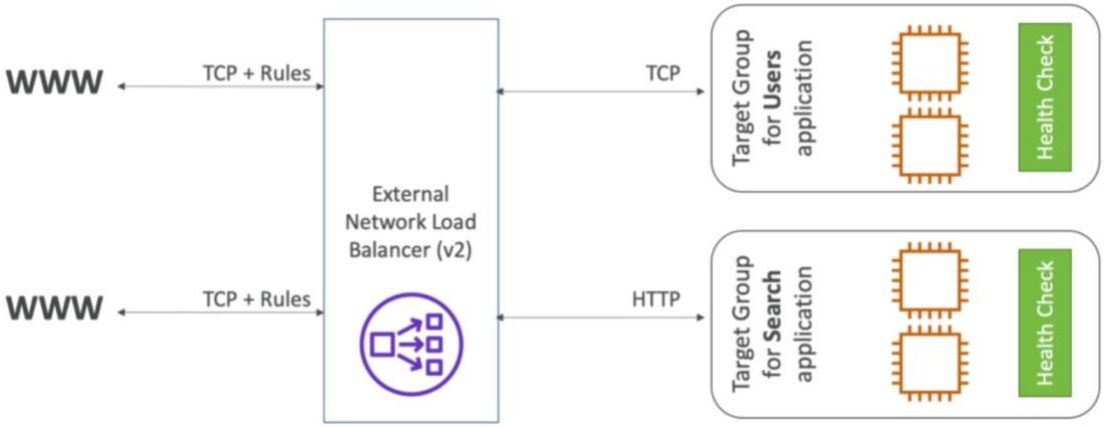
Network Load Balancer (v2)

- Network load balancer (Layer 4) allow to:
    - Forward Transmission Control Protocol (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)

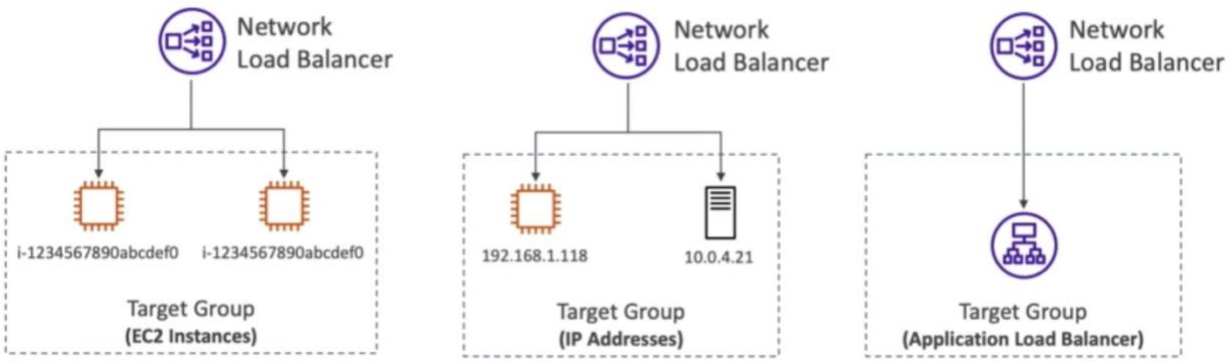
- NLB are used for extreme performance, TCP and UDP or UDP traffic
- Not included in the AWS free tier

Example of TCP (Layer 4) Based Traffic



Network Load Balancer – Target Groups (3 Options)

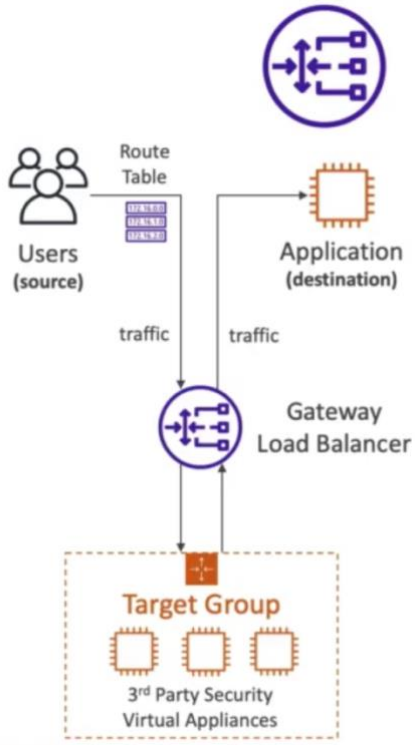
1. EC2 instances
2. IP Addresses – must be private IPs
3. Application Load Balancer



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, etc.

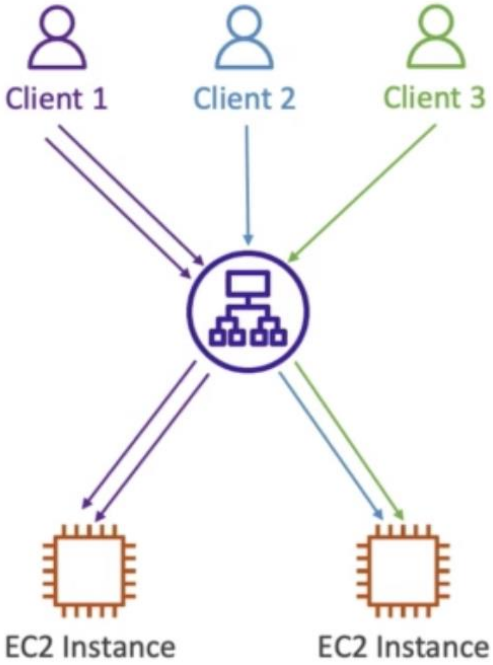
- Operates at Layer 3 (Network Layer) – IP Packets
- Combines the following functions:
    - **Transparent Network Gateway** – single entry/exit for all traffic
    - **Load Balancer** – distributes traffic to your virtual appliances
- Uses the **GENEVE** protocol on port **608 I**



Must understand this diagram – what it means and how it works

Elastic Load Balancer – 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 Balancers and Application Load Balancers
- 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 – Cookie 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

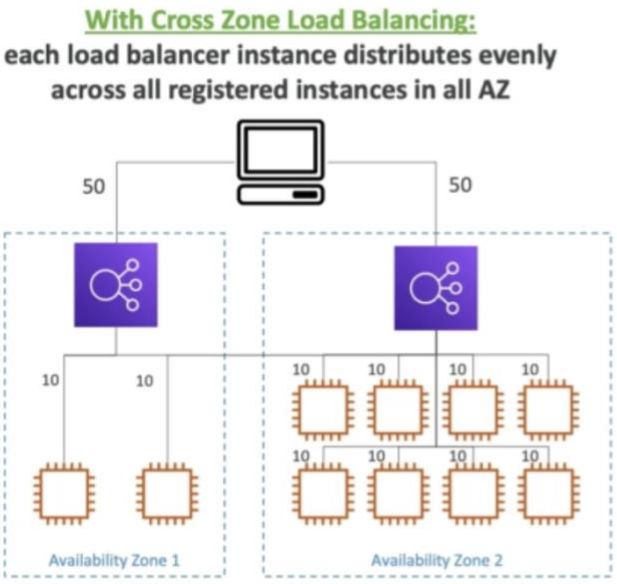
Duration-based Cookies

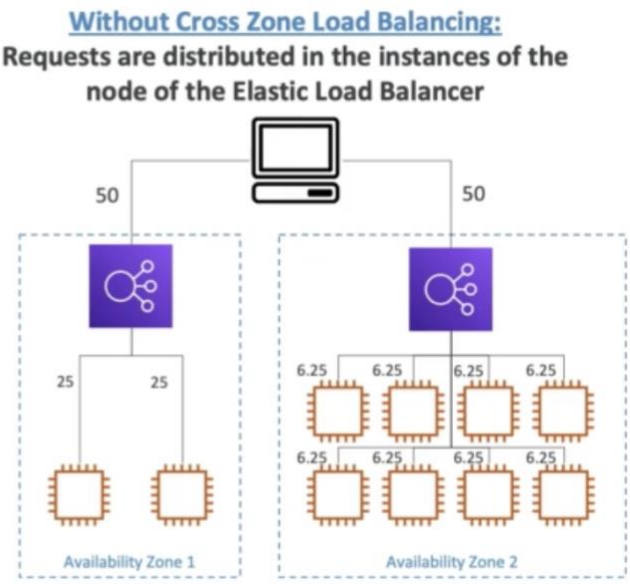
- Cookie generated by the load balancer
- Cookie name is AWSALB for ALB, AWSELB for CLB

Path to get to stickiness: Target Groups > Actions > Edit Attributes > Stickiness (check) 2 types & duration

Cross-Zone Load Balancing

Important to know –





Application Load Balancer:
Always on (can't be disable)
No charges for inter AZ data

Network 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 – Basics

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

- SSL (Secure Sockets Layer) – used to encrypted connections
- TLS (Transport Layer Security) – the newer version
- Nowadays, TLS certificates are many 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 Certificates

- 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 (IMPORTANT!)

- 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 Certificates

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

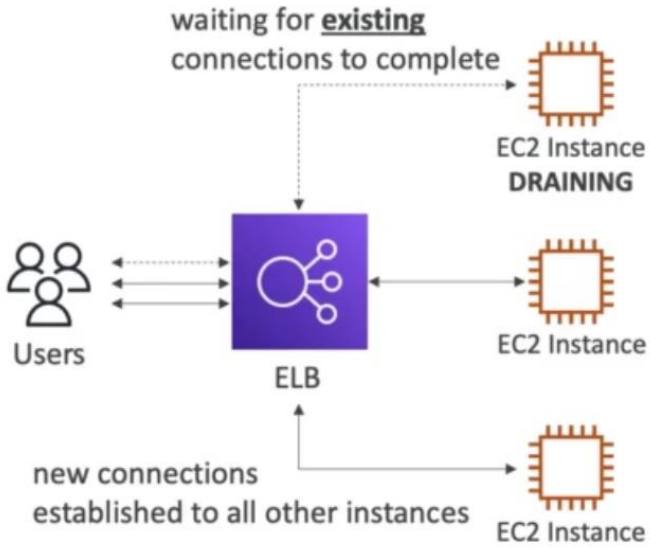
Network Load Balancer (v2)

- 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

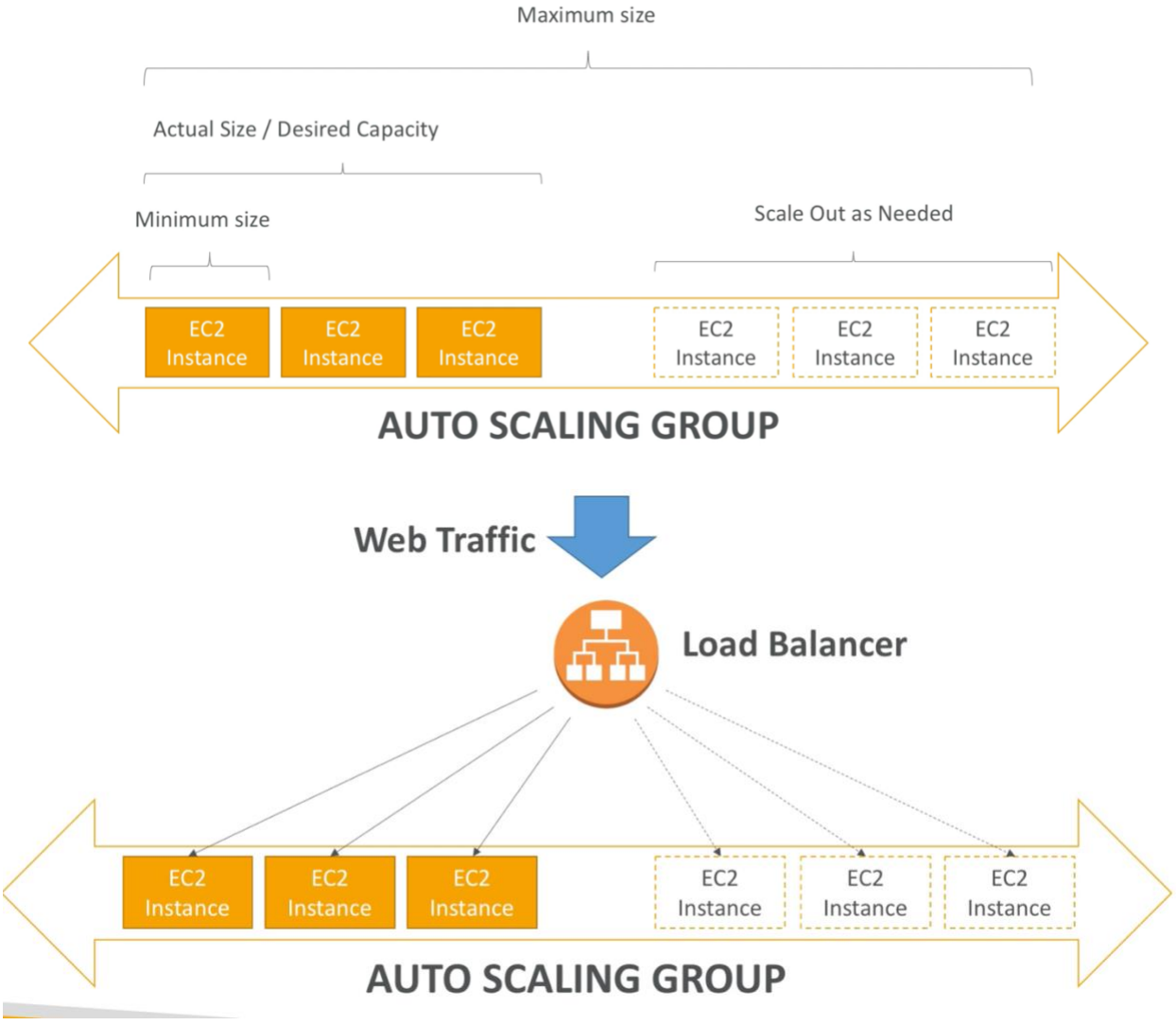


What's an Auto Scaling Group? (ASG)

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

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 machines running
- Automatically Register new instances to a load balancer



ASGs have the following attributes

- A launch configuration
  - AMI + Instance Type
  - EC2 User Data
  - EBS Volumes
  - Security Groups
  - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics 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 New Rules

- "Better" definition of auto scaling rules that are directly managed by EC2:
  - Target Average CPU Usage
  - Number of requests on the ELB per instance
  - Average Network In
  - Average Network Out
- These rules are easier to set up and can make more sense

Auto Scaling Custom Metric

We can auto scale based on a custom metric (ex: number of connected users)

1. Send custom metric from application on EC2 to CloudWatch (PutMetric API)
2. Create CloudWatch alarm to react to low/high values
3. Use the CloudWatch alarm as the scaling policy for ASG

ASG Brain Dump

- Scaling policies can be on CPU, Network, and can be on custom metrics or based on a schedule (if you know your visitor patterns)
- ASGs use Launch configurations or Launch Templates (newer)
- To update an ASG, you must provide a new launch configuration / launch template
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if you get terminated for whatever reason, the ASG will automatically create new ones as a replacement. Extra safety.
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)