

AWS Load Balancers and Scalability

What is Scalability?

Scalability refers to the ability of a system to handle increased load by adding or upgrading resources. It ensures that applications remain performant as demand fluctuates.

Vertical Scalability (Scaling Up/Down)

Vertical scalability involves increasing the capacity of a single resource, such as upgrading an EC2 instance type from t2.micro to m5.large.

Horizontal Scalability (Scaling Out/In)

Horizontal scalability involves adding more instances of a resource, such as increasing the number of EC2 instances in an Auto Scaling Group to distribute the load effectively.

What is High Availability?

High Availability (HA) ensures minimal downtime by distributing workloads across multiple redundant resources.

Example: Deploying an application across multiple AWS Availability Zones (AZs) so that if one fails, traffic is redirected to another.

Goals of Scalability and High Availability

- Ensure application uptime
 - Improve performance
 - Handle varying workloads efficiently
 - Reduce single points of failure
-

What is Load Balancing?

Load balancing is the process of distributing network traffic across multiple servers to ensure no single server is overwhelmed, improving performance and reliability.

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
-

What is an Elastic Load Balancer (ELB)? ☁️

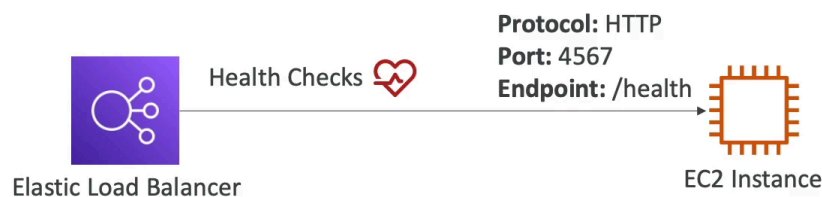
An Elastic Load Balancer (ELB) is an AWS-managed service that automatically distributes incoming traffic across multiple targets (EC2 instances, containers, or IP addresses) in one or more Availability Zones.

What are Health Checks? 💖

Health checks are automated processes that ELB performs to monitor the availability and responsiveness of registered targets. If an instance fails a health check, ELB stops sending traffic to it until it recovers.

Health Checks

- 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)



Types of Load Balancers in AWS 🚀

AWS provides different types of ELBs:

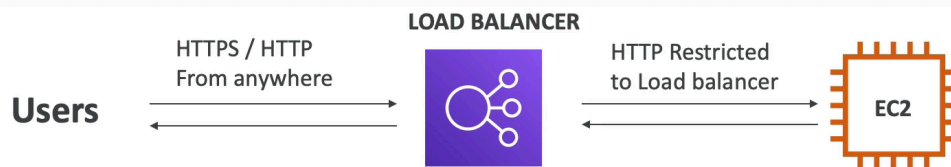
1. **Application Load Balancer (ALB)** – Operates at the application layer (Layer 7) and routes traffic based on content rules (e.g., path-based or host-based routing).
2. **Network Load Balancer (NLB)** – Operates at the transport layer (Layer 4) and provides ultra-low latency handling of TCP/UDP traffic.
3. **Gateway Load Balancer (GWLB)** – Routes traffic through third-party virtual appliances for security and monitoring.
4. **Classic Load Balancer (CLB)** – Legacy load balancer that supports basic load balancing for EC2 instances at both Layer 4 and Layer 7 (not recommended for new applications).

Types of load balancer on 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

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...

AWS Load Balancers - Key Concepts

Application Load Balancer (ALB) (V2)





An **Application Load Balancer (ALB)** is a Layer 7 load balancer that routes HTTP and HTTPS traffic based on request attributes such as host, path, headers, and query strings. It is commonly used for microservices and container-based applications.

Features:

-  Supports advanced request routing
-  Provides security through SSL/TLS termination
-  Integrates with AWS services like ECS, EC2, and Lambda
-  Scales automatically to handle traffic spikes



Target Groups

Target groups are used to route traffic to specific resources based on the type of workload. The following resources can be registered as targets:

- **EC2 Instances** –  Load balances across multiple EC2 instances.
- **ECS Tasks** –  Routes traffic to running containers in Amazon ECS.
- **Lambda Functions** –  Can trigger AWS Lambda functions to process requests.
- **IP Addresses** –  Routes requests to specific IP addresses (inside or outside AWS).

Health Checks

Health checks determine the availability of targets. ALB periodically checks the health of registered targets and routes traffic only to healthy ones.

-  Health checks run at regular intervals (default: 30 seconds).
-  A target is considered healthy after a specified number of successful responses.

- 🔍 The health check uses a specific HTTP or TCP endpoint to verify target availability.
- ❌ If a target fails multiple health checks, it is marked as unhealthy, and traffic is stopped until it recovers.

Fixed Hostname

ALB provides a **fixed DNS hostname** that remains constant for the lifecycle of the load balancer.

- Example: `my-alb-123456789.us-east-1.elb.amazonaws.com`
- 🌐 The hostname can be mapped to a custom domain using Route 53 or other DNS services.
- 📌 Unlike Classic Load Balancers, ALB does not support static IP addresses, but an **Elastic IP** can be used with a Network Load Balancer (NLB) if needed.

This document provides a basic overview of Application Load Balancers and their key components. For further details, refer to the AWS documentation.

Network Load Balancer (NLB)

Network Load Balancer (NLB)

A **Network Load Balancer (NLB)** operates at **Layer 4** (Transport Layer) and is designed to handle TCP and UDP traffic with high performance and ultra-low latency.

◆ Key Features:

- **Handles Millions of Requests** – Efficiently manages a high volume of connections per second.
 - **Low Latency** – Optimized for extreme performance with ultra-low latency.
 - **Supports TCP, UDP, and TLS Traffic** – Directly routes network packets to targets.
 - **Static IP Address** – Provides a fixed IP per Availability Zone (AZ) for easier whitelisting.
 - **Elastic IP Support** – Allows assignment of Elastic IPs to maintain a consistent endpoint.
-

Network Load Balancer (v2)



- Network load balancers (Layer 4) allow to:
 - Forward TCP & UDP traffic to your instances
 - Handle millions of request per seconds
 - Ultra-low latency
- 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

Target Groups in NLB

Target groups define where NLB forwards incoming traffic. NLB supports the following types of target groups:

✓ EC2 Instances

- Routes traffic to registered **Amazon EC2** instances.
- Automatically distributes connections based on availability and health.

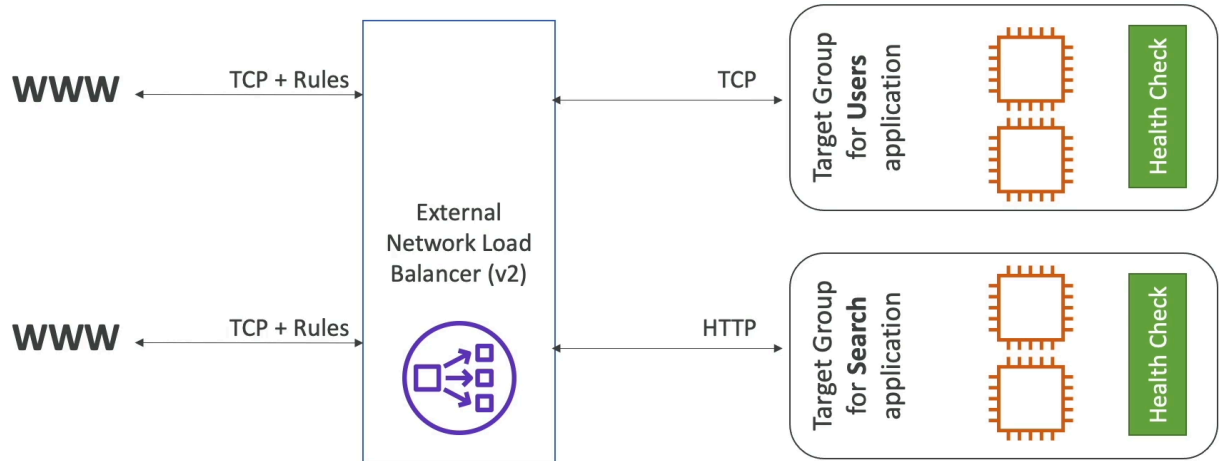
✓ IP Addresses

- Can forward requests to specific **private IP addresses**.
- Useful for balancing traffic between on-premises and AWS workloads.

✓ Application Load Balancer (ALB)

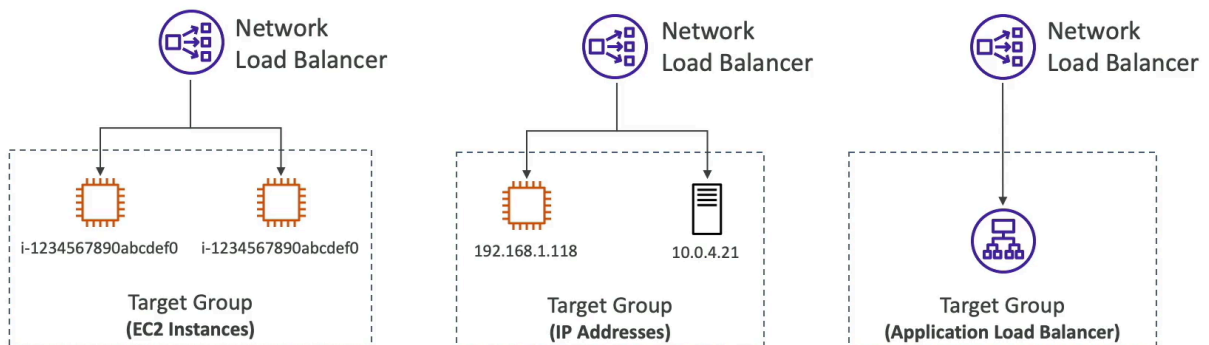
- Can forward traffic to an **ALB** as a target.
- Useful for combining Layer 4 and Layer 7 load balancing.

Network Load Balancer (v2) TCP (Layer 4) Based Traffic



Network Load Balancer – Target Groups

- EC2 instances
- IP Addresses – must be private IPs
- Application Load Balancer



Health Checks

- Monitors registered targets and only sends traffic to healthy ones.
- Uses **TCP, HTTP, or HTTPS** to verify target health.
- If a target fails multiple health checks, NLB stops sending traffic to it.

AWS Gateway Load Balancer (GLB) 🚀

What is a Gateway Load Balancer?

AWS **Gateway Load Balancer (GLB)** is a specialized load balancer that helps manage third-party network virtual appliances, such as firewalls, intrusion detection systems, and deep packet inspection systems. It simplifies deployment and scaling of these appliances.

Why Use GLB? 🤔

- **Manages Third-Party Appliances:** Helps distribute traffic across multiple virtual appliances for **firewalls, intrusion prevention, and security monitoring**.
- **Transparent Network Gateway:** Acts as a **single entry/exit point** for traffic inspection without modifying packets.
- **Scalability & High Availability:** Automatically scales and replaces failed appliances to ensure **uninterrupted traffic flow**.

Key Features 🛡️

- **Single Gateway for Traffic Inspection**
- **Ensures High Availability & Fault Tolerance**
- **Works at Layer 3 (Network Layer)**
- **Supports Auto Scaling of Appliances**
- **Integrated with AWS Transit Gateway**

Target Groups in GLB 🎯

Gateway Load Balancer forwards traffic to the following target types:

- **EC2 Instances** (running network appliances)
- **IP Addresses** (for appliances outside AWS)

GLB enables efficient traffic routing and management for security appliances without disrupting network performance. ✅

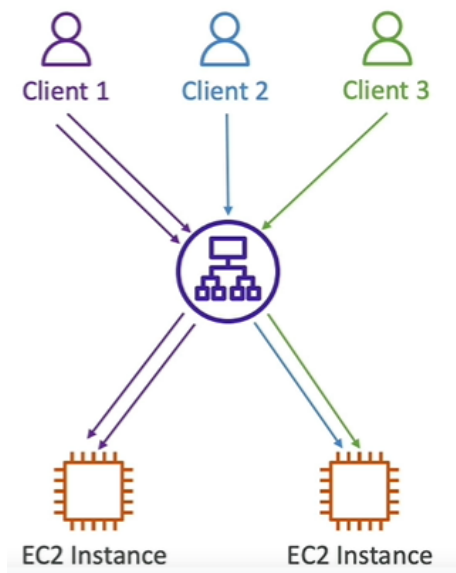
Sticky Sessions (Session Affinity) 🔄🍪

What is a Sticky Session?

- Sticky Sessions (also called **Session Affinity**) ensure that a **client is always redirected to the same backend instance**.
- Helps maintain **session persistence** for applications that store user data on the backend server.
- Supported by **Classic Load Balancer (CLB)**, **Application Load Balancer (ALB)**, and **Network Load Balancer (NLB)**.

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



What is a Cookie? 🍪

- A **small piece of data** stored on the client's browser, used to track and maintain session states.
- In AWS Load Balancers, cookies are used to **bind** a client's request to a specific backend instance.

Types of Cookies in AWS Load Balancers

Application-Based Cookies 🏗️

1. **Custom Cookie**
 - Generated by the **backend application**.
 - Can include **custom attributes** for user sessions.
 - Cookie name must be unique per **target group**.
 - **Do not use AWSALB, AWSALBAPP, or AWSALBTG** (these are reserved for AWS).
2. **Application Cookie**
 - Generated by the **load balancer**.
 - Cookie name: **AWSALBAPP**.

Duration-Based Cookies ⌚

- Cookie is generated by the **load balancer**.
- Cookie name: **AWSALB** for ALB, **AWSELB** for CLB.
- Used when the session should last for a specified **duration** rather than being application-controlled.

Cross Zone Balancing 🔄

Cross Zone Balancing allows AWS Load Balancers to distribute incoming traffic evenly across all registered targets in all Availability Zones (AZs). This ensures better resource utilization and improves application reliability.

Example:

Imagine you have an Application Load Balancer (ALB) deployed across two Availability Zones:

- AZ1 has 2 EC2 instances 🖥️ 🖥️
- AZ2 has 1 EC2 instance 🖥️

With Cross Zone Balancing **Enabled** ✅:

- The ALB distributes traffic **equally** to all three instances across both AZs.

With Cross Zone Balancing **Disabled** ❌:

- Each ALB node in an AZ only routes traffic to instances within the same AZ, causing an **uneven** distribution if instance counts are imbalanced.

Charges of Cross Zone Balancing 💰

- **Application Load Balancer (ALB)** 🏗️: **No additional charge** (Cross Zone Balancing is always enabled by default)
- **Network Load Balancer (NLB)** 🌐: **Free if only one AZ is used**; if multiple AZs are used, there is an inter-AZ data transfer cost.
- **Gateway Load Balancer (GLB)** 🏠: **Same pricing as NLB**, with inter-AZ data transfer costs when enabled.

AWS Load Balancers: SSL Certificates & Encryption 🔒

What is SSL? 🔑

SSL (Secure Sockets Layer) is a security protocol that encrypts data between a client (browser) and a server. It ensures data confidentiality, integrity, and authentication. Modern systems use TLS (Transport Layer Security) instead of SSL.

What is In-Flight Encryption? ✈️🔒

In-flight encryption refers to encrypting data while it is being transmitted between a client and a server. This prevents attackers from intercepting or tampering with sensitive data during transmission.

What is TLS? 🔒📜

TLS (Transport Layer Security) is the successor of SSL, offering stronger encryption and better security. It protects data in transit and is widely used in HTTPS connections.

What is Server Name Indication (SNI)? 🌐🔒

SNI is an extension of TLS that allows multiple SSL certificates to be hosted on a single IP address. It helps servers identify which certificate to present based on the hostname requested by the client.

By using AWS Load Balancers with SSL/TLS encryption, businesses can secure data transmissions and ensure a safe browsing experience. 🚀

AWS Load Balancers: Connection Draining & Deregistration Delay

Connection Draining (AWS CLB)

- Ensures active requests complete before deregistering an EC2 instance.
- New requests are routed to healthy instances while existing ones finish.
- Helps in seamless scaling and maintenance without disrupting users.
- Default timeout: 300 seconds (can be adjusted).

Deregistration Delay (AWS ALB & NLB)

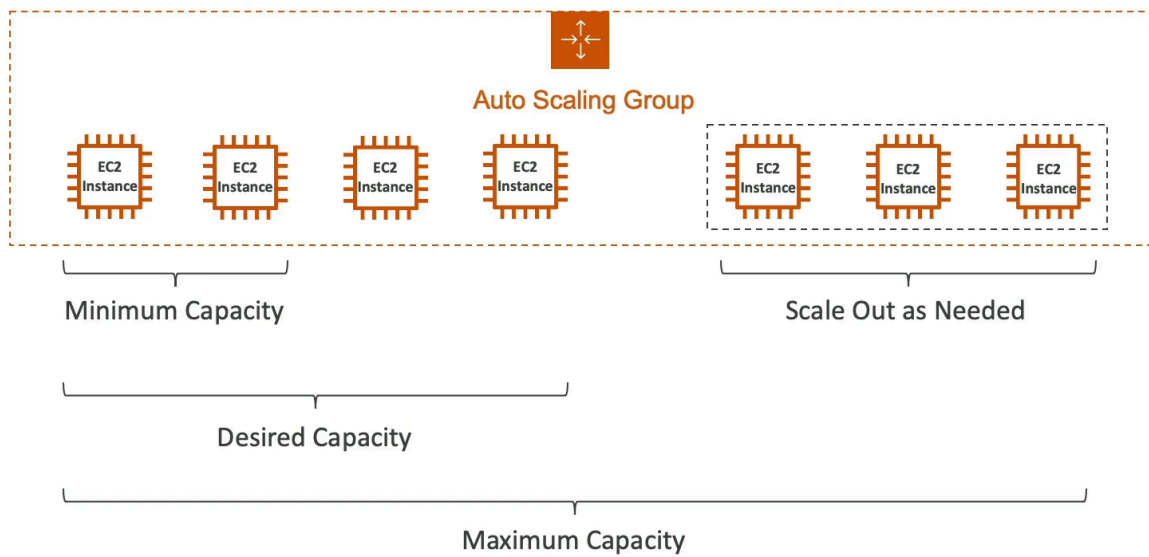
- Similar to Connection Draining but for ALB & NLB.
- Controls the time before an instance is fully removed from a target group.
- ALB Default: 300 seconds | NLB Default: 0 seconds (can be changed).
- Prevents sudden termination of connections, ensuring a smooth transition.

Key Benefits:

- Avoids request failures during instance removal.
- Ensures users do not face interruptions.
- Enhances load balancer efficiency during scaling operations.

AWS Auto Scaling Groups (ASG) 🚀

An Auto Scaling Group (ASG) is an AWS service that automatically adjusts the number of EC2 instances in response to demand. It ensures that the right number of instances are running to handle traffic while optimizing costs. ASG can scale out (add instances) when demand increases and scale in (remove instances) when demand decreases.



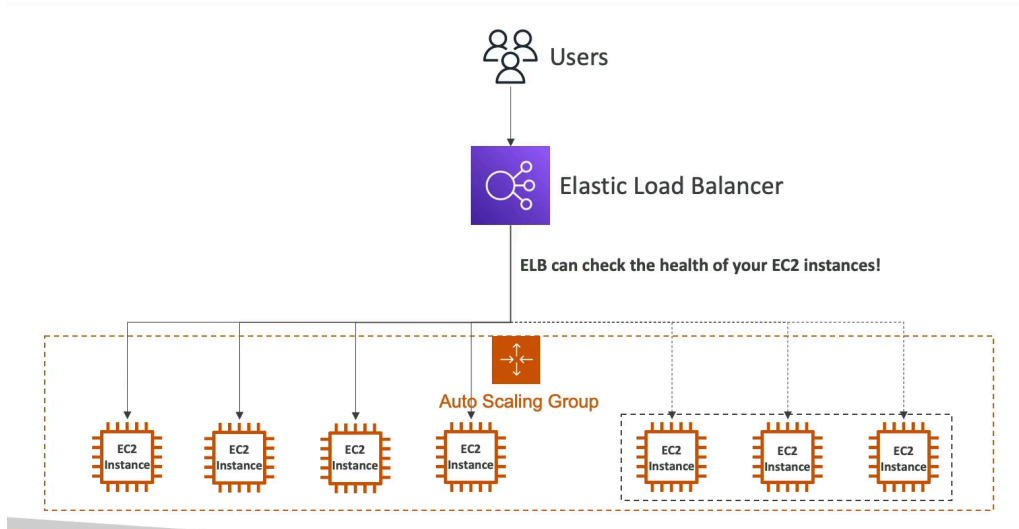
Goal of Auto Scaling Groups (ASG) 🎯

An **Auto Scaling Group (ASG)** helps maintain the right number of EC2 instances to handle varying levels of traffic. It ensures **high availability**, **cost optimization**, and **fault tolerance**. ASG can automatically **scale in** (reduce instances) or **scale out** (add instances) based on demand.

How ASG Works with AWS Load Balancer ⚖️









ASGs work seamlessly with **Elastic Load Balancers (ELB)** to distribute traffic evenly across EC2 instances. The Load Balancer:

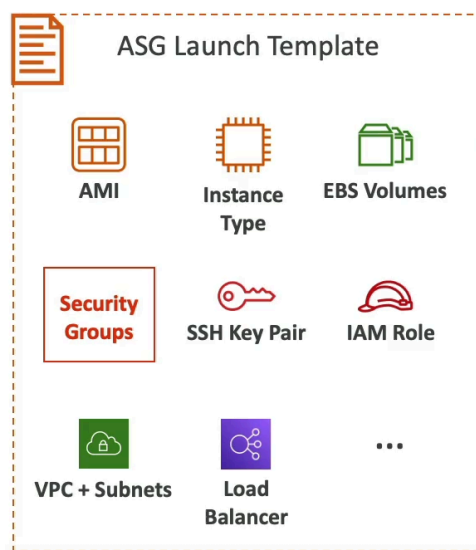
- **Routes traffic** to healthy instances ✅
- **Removes unhealthy instances** ❌
- **Works with auto-scaling policies** to ensure **dynamic traffic handling** 📊



ASG Launch Template

A **Launch Template** is used to define how new EC2 instances should be launched within an ASG. It includes:

- **Amazon Machine Image (AMI)**  - OS & pre-installed software
- **Instance Type**  - Specifies CPU, memory, and network performance
- **EBS Volumes**  - Storage configuration
- **Security Groups**  - Firewall rules for instance security
- **SSH Key Pair**  - Secure remote access
- **IAM Role**  - Permissions for instances
- **VPC & Subnets**  - Networking setup
- **Load Balancer**  - Connect ASG to ELB



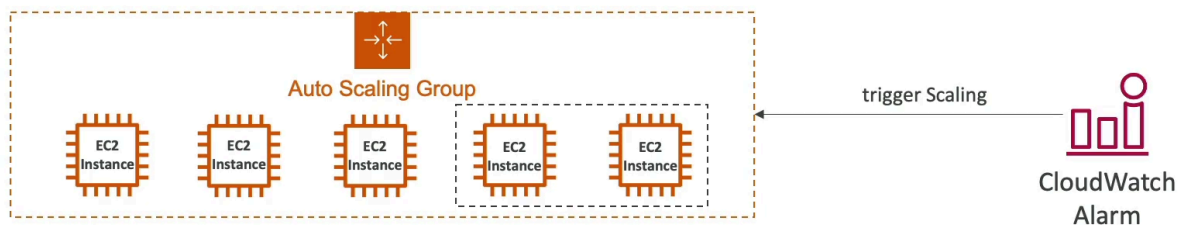
Scaling Policies with CloudWatch Alarms and Scaling

CloudWatch Alarms trigger scaling actions based on system metrics such as:





- CPU Utilization 
- Network Traffic 
- Request Count 

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)



Types of Scaling Policies:

1. **Dynamic Scaling**  - Adjusts instances in response to real-time demand.
 - **Target Tracking**  - Adjusts instances based on a target metric (e.g., keep CPU at 50%).
 - **Step Scaling**  - Adds/removes instances based on threshold breaches.
2. **Scheduled Scaling**  - Predictable scaling for expected traffic increases/decreases.

Summary




✓ ASG ensures **high availability** and **cost efficiency** ✓ Works with ELB for **traffic distribution** ✓ Uses **Launch Templates** for consistent instance configuration ✓ Implements **CloudWatch Alarms** for automatic scaling

AWS Auto Scaling Policies Explained

1. Dynamic Scaling


Dynamic Scaling automatically adjusts the number of EC2 instances based on real-time metrics. It's reactive and keeps your application responsive.

Types of Dynamic Scaling:

-  **Target Tracking Scaling**
 - Automatically adjusts capacity to maintain a target metric (like CPU utilization).
 - *Example:* Maintain CPU at 50%.
-  **Step Scaling**
 - Adds or removes instances in steps based on metric thresholds.
 - *Example:* Add 1 instance if CPU > 60%, add 2 if CPU > 80%.
-  **Simple Scaling**
 - Performs a scaling activity based on a single alarm condition.
 - *Example:* If CPU > 70% for 5 minutes → add 1 instance.

2. Scheduled Scaling

You can schedule scaling actions ahead of time.

- *Example:* Increase instances at 8 AM when traffic usually spikes.
-  Great for predictable workloads!

3. Predictive Scaling

Uses machine learning to forecast future traffic and scales accordingly.

- *Example:* It predicts higher traffic on Friday evenings and scales ahead of time.
 - 🤖 Smart and proactive!
-

Good Metrics to Scale On

- ⚙️ **CPU Utilization** – Helps scale based on how much processing power is being used.
 - 🔄 **RequestCountPerTarget** – Ideal for load balancers, tracks incoming traffic.
 - 📶 **Average Network In/Out** – Monitors data transfer in and out of instances.
 - 🛠️ **Custom Metrics** – Tailored to your application (like queue depth or latency).
-

Scaling Cooldowns

What is a cooldown period?

- 🧑 A cooldown period is a waiting time after a scaling activity.
- It ensures that the system stabilizes before another scaling action occurs.
- Prevents **over-scaling** and **resource thrashing**.

How it works:

- After a scale-out or scale-in, AWS waits for a specific time (default: 300 seconds) before triggering another action.

