

AWS ELB and AutoScaling

CS516 – Cloud Computing

Computer Science Department

Maharishi International University

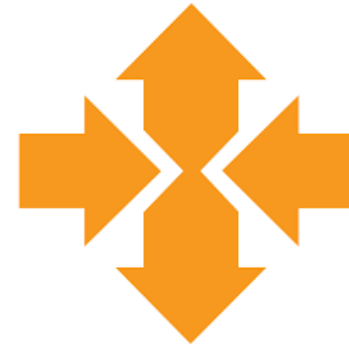
Maharishi International University - Fairfield, Iowa



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.

Content

- Elastic Load Balancer (ELB)
 - Listener
 - Listener rule
 - Target groups
- Types of ELB (OSI Model)
 - Application Load Balancer (ALB)
 - Network Load Balancer (NLB)
 - Classic Load Balancer
 - Gateway Load Balancer
- Auto Scaling (with CloudWatch)
 - Target Tracking Scaling policy



Elastic Load Balancer (ELB)

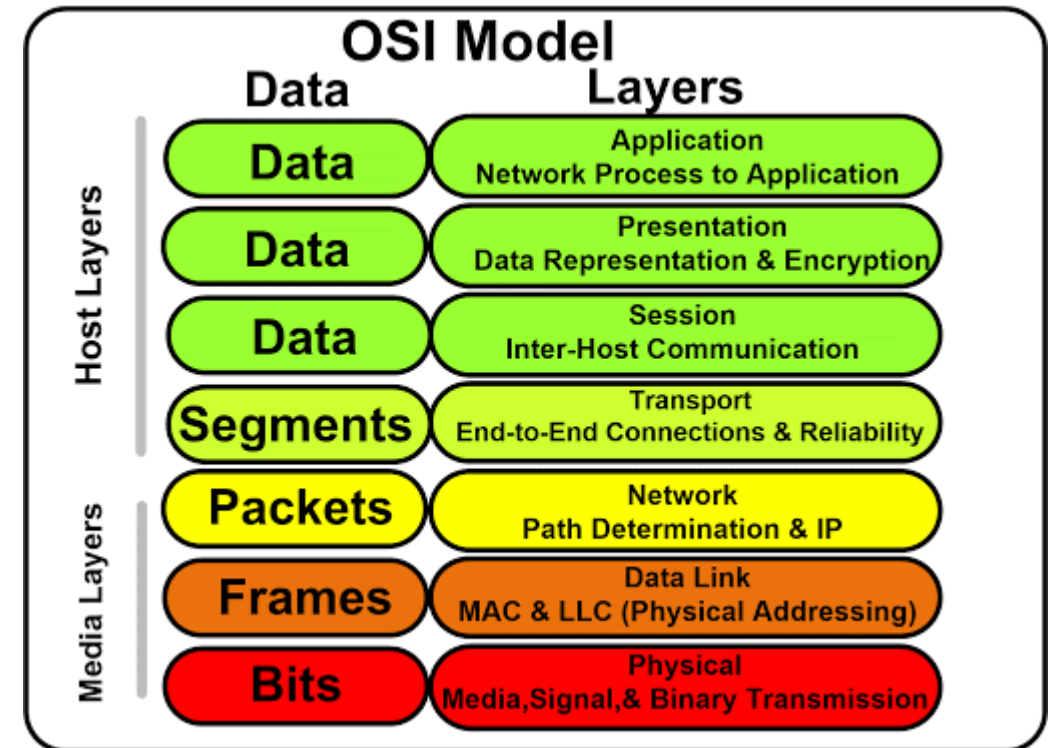
ELB automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables **fault-tolerance** in applications, seamlessly providing the required amount of load-balancing capacity needed to route application traffic.

A tool that distributes incoming web traffic (visitors to a web site) and **equally across multiple EC2** instances that are running a web site.

Helps prevent one server from being overloaded while another server can handle more visitors.

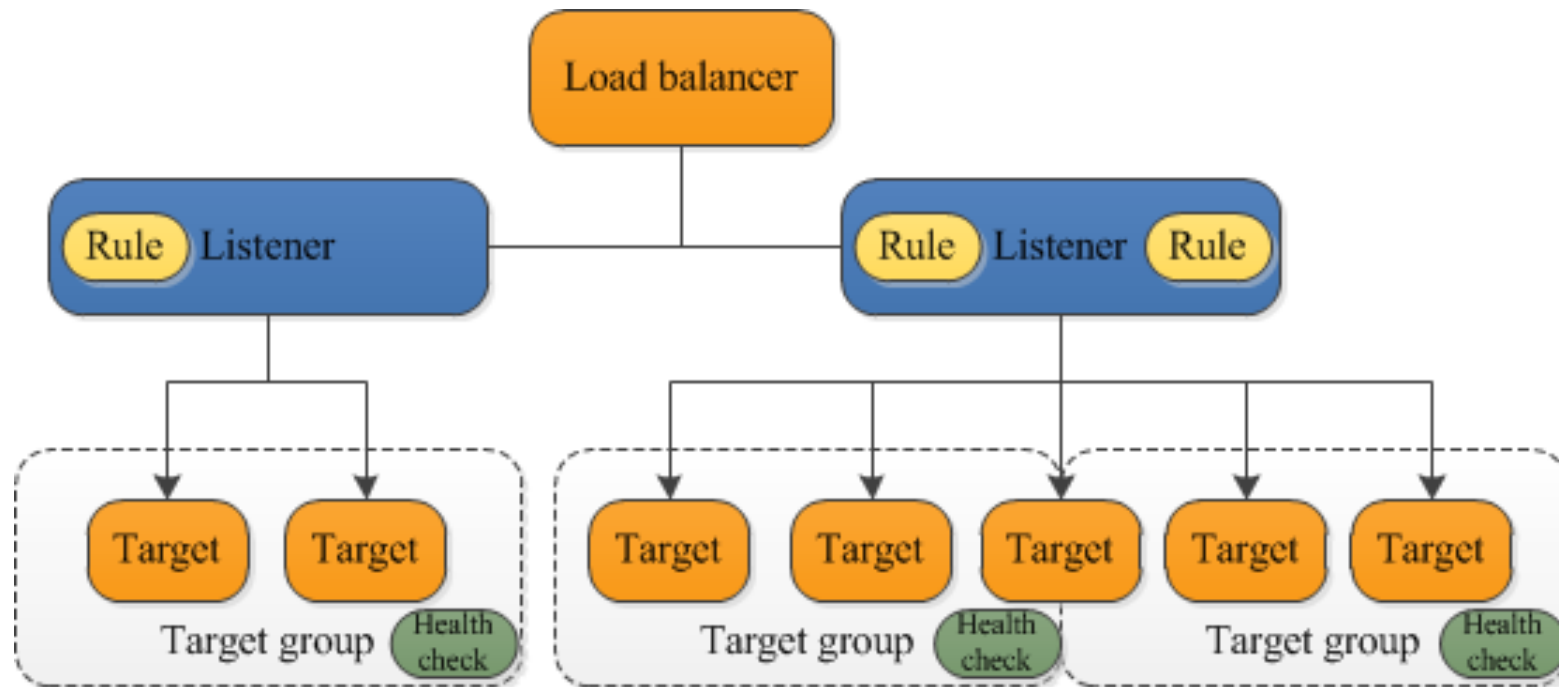
Types of ELB

1. **Classic Load Balancer (CLB)** – Old generation, not recommended for new apps. Performs routing at Layer 4 and Layer 7.
2. **Network Load Balancer (NLB)** – Routes connections based on IP protocol data (layer 4). Ultra high performance and low latency. Supports UDP and static IP addresses as targets.
3. **Application Load Balancer (ALB)** – Routes based on the content of the request (layer 7). Supports path-based, host-based, query string, parameter-based, and source IP based routing. Supports IP addresses, Lambda, containers as targets.
4. **Gateway Load Balancer** – Operates at Layer 3 (Network). Gateway Load Balancers enable you to deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems.



Application Load Balancer (ALB)

A load balancer serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances in multiple Availability Zones. This increases the availability of your application. You add one or more listeners to your load balancer.



ALB Listener

A listener checks for connection requests from clients, using the **protocol** and **port** that you configure.

A **certificate** is attached to the listener. You must define a default certificate if using https. AWS and custom certificates are stored on Amazon Certificate Manager (ACM). AWS certificates are free!

The screenshot displays the AWS Management Console interface for configuring an ALB listener. At the top, there are buttons for 'Create Load Balancer' and 'Actions'. Below this is a search bar and a table listing the load balancers. The table has columns for Name, DNS name, State, VPC ID, Availability Zones, Type, and Created At. One load balancer, 'my-alb', is listed with a state of 'provisioning'. Below the table, the 'Load balancer: my-alb' section is shown, with tabs for Description, Listeners, Monitoring, Integrated services, and Tags. The 'Listeners' tab is selected, showing a description of the listener's role. Below the description are buttons for 'Add listener', 'Edit', and 'Delete'. A table lists the existing listener with columns for Listener ID, Security policy, SSL Certificate, and Rules. The listener 'HTTP : 80' is shown with a security policy of 'N/A' and an SSL certificate of 'N/A'. The rules are 'Default: forwarding to my-tg' and 'View/edit rules'.

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
my-alb	my-alb-967328916.us-east-1...	provisioning	vpc-063dae80fe38de125	us-east-1b, us-east-1c	application	May 23, 2021 at 1:03:47 PM ...

Load balancer: my-alb

Description Listeners Monitoring Integrated services Tags

A listener checks for connection requests using its configured protocol and port, and the load balancer uses the listener rules to route requests to targets. You can add, remove, or update listeners and listener rules.

Add listener Edit Delete

Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/> HTTP : 80 arn...bfe3f9eb5b0c8ddd ▾	N/A	N/A	Default: forwarding to my-tg View/edit rules

ALB Listener Rules

The **rules** that you define for a listener determine how the load balancer routes requests to its **registered targets**.

Each rule consists of a **priority**, **actions**, **conditions**. When the conditions for a rule are met, then its actions are performed. You must define a default rule for each listener, and you can optionally define additional rules.

<

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.

↑ Insert Rule ↓

RULE ID	IF (all match)	THEN
1 A rule ID (ARN) is generated when you save your rule.	<div>+ Add condition ▾</div> <div>Host header... Path... Http header... Http request method... Query string... Source IP...</div>	<div>+ Add action ▾</div>
last HTTP 80: default action <i>This rule cannot be moved or deleted</i>		<div>THEN</div> <div>Forward to</div> <div>my-tg: 1 (100%)</div> <div>Group-level stickiness: Off</div>

ALB Target Groups

Each target group routes requests to one or more registered targets, such as EC2 instances, using the protocol and port number that you specify.

You can configure health checks on a per target group basis. Health checks are performed on all targets registered to a target group.


Note: When you associate a resource with an ALB, you associate it with its target group. When creating an ALB, target group can have no resources. But the important thing is you must specify the right target group type (ip, instance, lambda).

ALB Target Groups – Why is an instance unhealthy?



EC2 > Target groups > my-tg






my-tg

[Delete](#)

 `arn:aws:elasticloadbalancing:us-east-1:637278349805:targetgroup/my-tg/1a341e1a092d8a30`

Details

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-063dae80fe38de125 
Load balancer my-alb 			

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
1	 0	 1	 0	 0	 0

Targets | Monitoring | **Health checks** | Attributes | Tags

Health check settings

[Edit](#)

Protocol HTTP	Path /	Port Traffic port	Healthy threshold 2 consecutive health check successes
Unhealthy threshold 2 consecutive health check failures	Timeout 120 seconds	Interval 300 seconds	Success codes 200

Instance's SG only allowed the developer's IP address

EC2 > Security Groups > sg-0bde078b4803140da - allow-my-http-ssh

sg-0bde078b4803140da - allow-my-http-ssh

Actions ▼

Details

Security group name
allow-my-http-ssh

Security group ID
sg-0bde078b4803140da

Description
launch-wizard-1 created 2021-05-22T12:26:31.891-05:00

VPC ID
vpc-063dae80fe38de125

Owner
637278349805

Inbound rules count
2 Permission entries

Outbound rules count
1 Permission entry

Inbound rules

Outbound rules

Tags

Inbound rules (2)

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	69.18.21.229/32	–
SSH	TCP	22	69.18.21.229/32	–

Whitelisting the ALB SG in Instance's SG

EC2 > Security Groups > sg-0bde078b4803140da - allow-my-http-ssh

sg-0bde078b4803140da - allow-my-http-ssh

Actions ▼

Details

Security group name

allow-my-http-ssh

Security group ID

sg-0bde078b4803140da

Description

launch-wizard-1 created 2021-05-22T12:26:31.891-05:00

VPC ID

vpc-063dae80fe38de125

Owner

637278349805

Inbound rules count

3 Permission entries

Outbound rules count

1 Permission entry

Inbound rules

Outbound rules

Tags

Inbound rules (3)

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	69.18.21.229/32	–
HTTP	TCP	80	sg-0ebb0b5c99091815d / my-alb-sg	–
SSH	TCP	22	69.18.21.229/32	–

Health check is now green

EC2 > Target groups > my-tg

my-tg

arn:aws:elasticloadbalancing:us-east-1:637278349805:targetgroup/my-tg/1a341e1a092d8a30

Delete

Details

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-063dae80fe38de125
Load balancer my-alb			

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
1	✓ 1	✗ 0	⋮ 0	⌚ 0	⊖ 0

Targets | Monitoring | Health checks | Attributes | Tags

Registered targets (1/1)

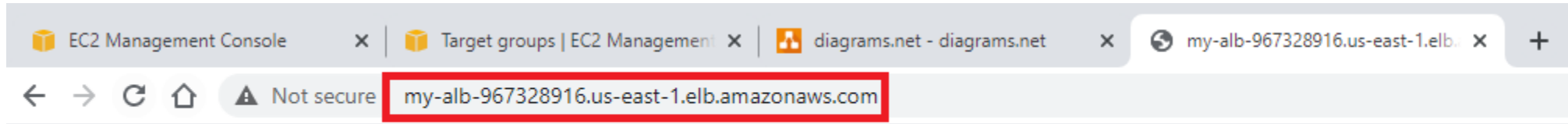
Filter resources by property or value

Refresh | Deregister | Register targets

< 1 > ⚙

<input checked="" type="checkbox"/>	Instance ID	Name	Port	Zone	Status	Status details
<input checked="" type="checkbox"/>	i-0c334940879eef22e		80	us-east-1b	✓ healthy	

ALB to EC2 result



Hello from Unubold

Routing algorithms

The routing algorithms choose the target in the target group.

Round-robin (default) load balancing is one of the simplest methods for distributing client requests across a group of servers. Going down the list of servers in the group, the round-robin load balancer forwards a client request to each server in turn. When it reaches the end of the list, the load balancer loops back and goes down the list again.

Least outstanding requests is an algorithm that chooses which instance receives the next request by selecting the instance that, at that moment, has the lowest number of outstanding (pending, unfinished) requests.

Auto Scaling benefits

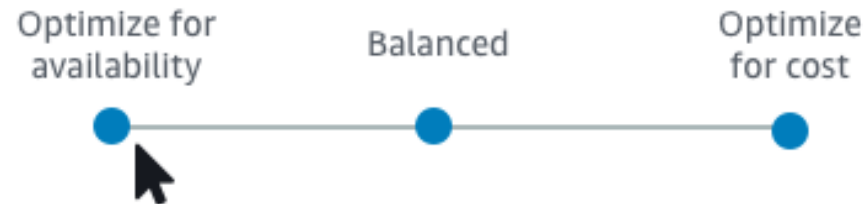
- **Better fault tolerance** - Auto Scaling can detect when a resource is unhealthy, terminate it, and launch a resource to replace it. You can also configure resources to use multiple AZs. If one AZ becomes unavailable, Auto Scaling can launch resources in another one to compensate.
- **Better availability** - Auto Scaling helps ensure that your application always has the right amount of capacity to handle the current traffic demand.
- **Better cost management** - Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the resources you use.

AWS Auto Scaling

Auto Scaling is enabled by **CloudWatch**

Scaling Strategy - Your own custom strategy. You can optimize

- for availability
- for cost
- a balance of both.

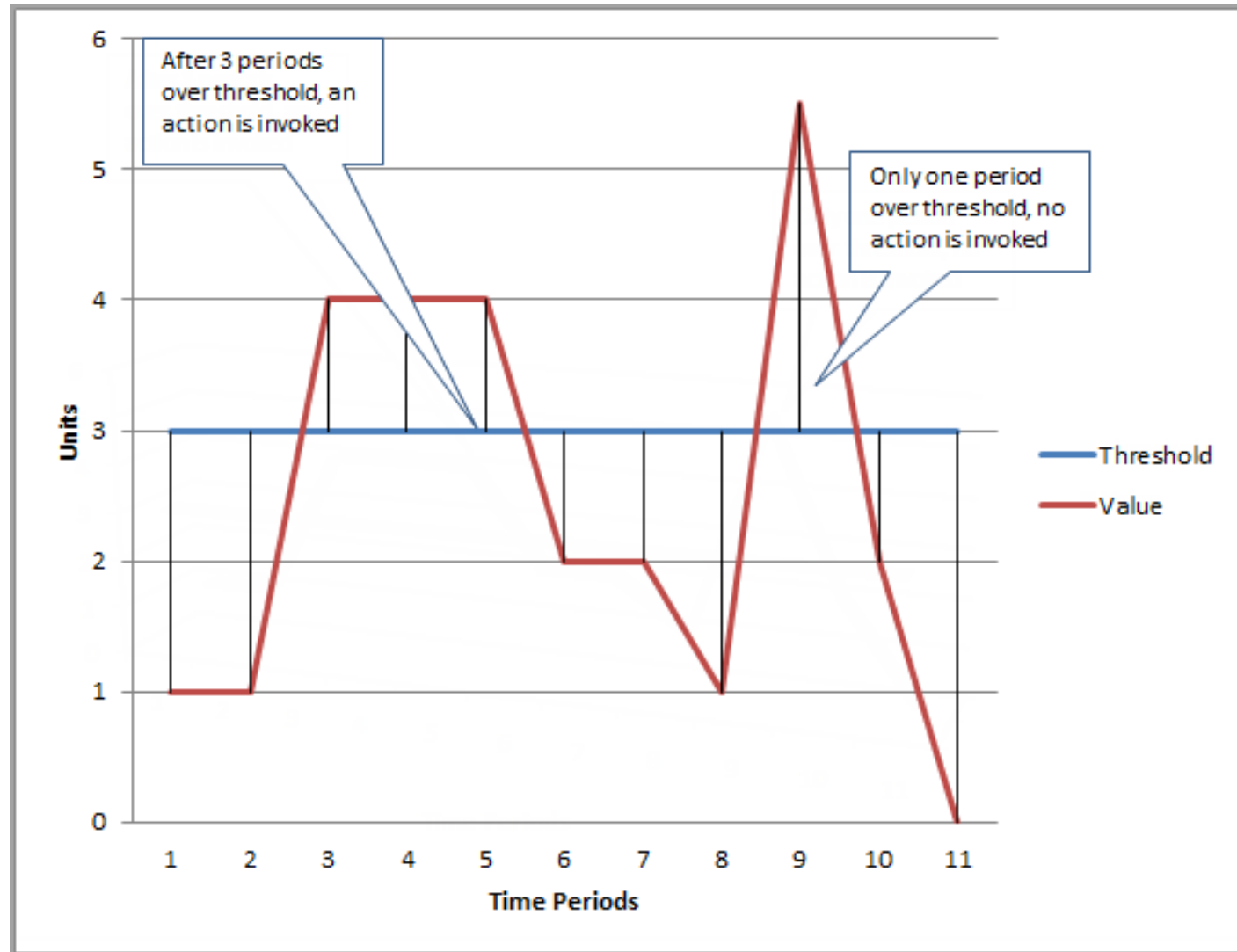


What is Auto Scaling?

Auto Scaling is nothing but it changes the desired capacity of the services such as EC2 based on the CloudWatch Alarms.

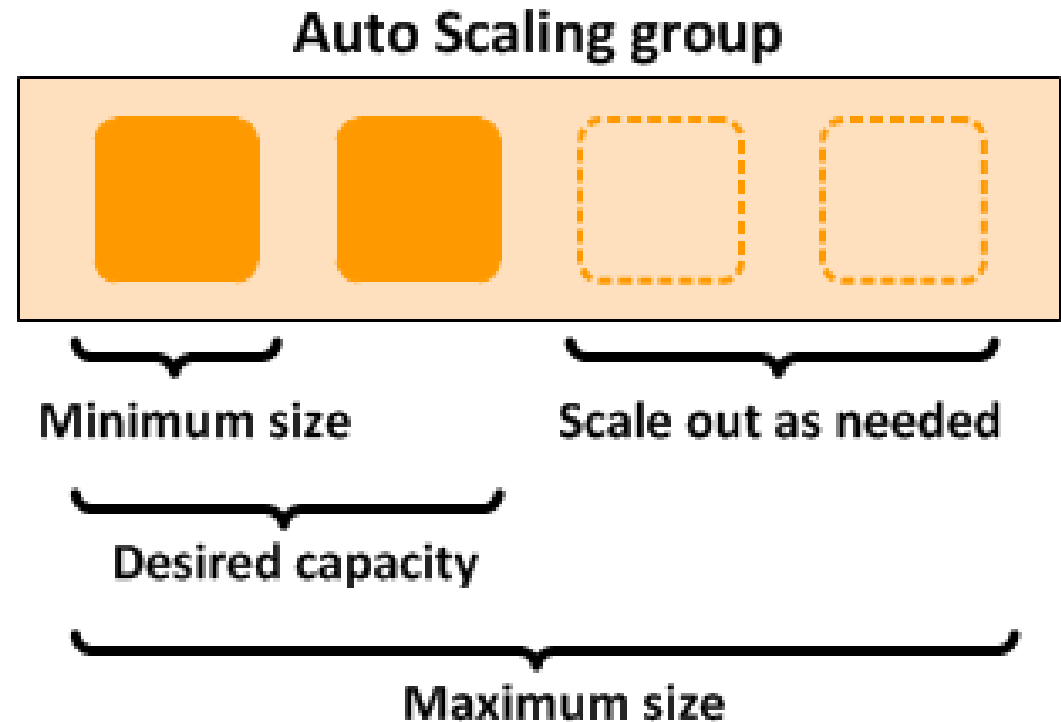
AWS Auto Scaling notifies the service that it is time to scale out or scale in. Then scaling happens on the service.

CloudWatch Alarm for AutoScaling



Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called **Auto Scaling groups**.



EC2 Auto Scaling components

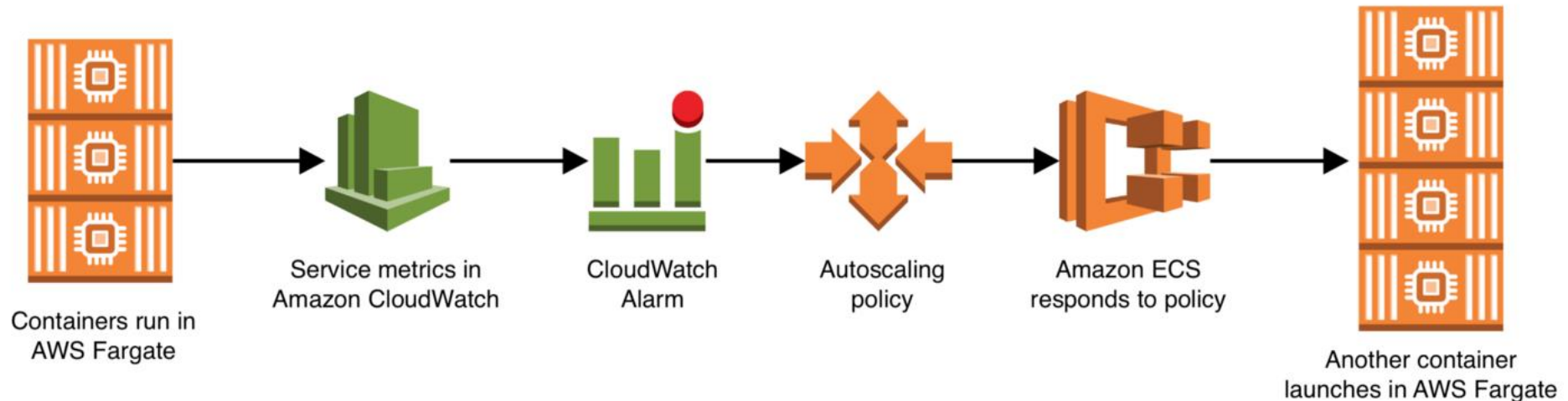
Groups - Your EC2 instances are organized into groups so that they can be treated as a logical unit for the purposes of scaling and management.

Configuration templates - Your group uses a launch (configuration) template where you can specify information such as the AMI ID, instance type, key pair, security groups, and block device mapping for your instances.

Scaling options - Amazon EC2 Auto Scaling provides several ways for you to scale your Auto Scaling groups. For example, you can configure a group to scale based on the occurrence of specified conditions (dynamic scaling) or on a schedule.

Application Auto Scaling

Application Auto Scaling is a web service for developers and system administrators who need a solution for automatically scaling their scalable resources for individual AWS services beyond Amazon EC2. You can scale Elastic Container Service (ECS), Lambda, Aurora replicas (RDS), DynamoDB, and more.



Features of Application Auto Scaling

Application Auto Scaling allows you to automatically scale your scalable resources according to conditions that you define:

1. **Target tracking scaling** - Scale a resource based on a target value for a specific CloudWatch metric. Like thermostat at your home.
2. **Step scaling** - Scale a resource based on a set of scaling adjustments that vary based on the size of the alarm breach.
3. **Scheduled scaling** - Scale a resource based on the date and time.
4. **Predictive scaling** - Uses machine learning to analyze each resource's historical workload and regularly forecasts the future load for the next two days.

Cooldown Period

The amount of time to wait for a previous scaling activity to take effect is called the cooldown period.

There are two types of cooldown periods:

1. **Scale-out** cooldown period, the intention is to continuously (but not excessively) scale out. After successfully scales out, it starts to calculate the cooldown time. The scaling policy won't increase the **desired capacity** again unless either a larger scale out is triggered or the cooldown period ends.
2. **Scale-in** cooldown period, the intention is to scale in conservatively to protect your application's availability, so scale-in activities are blocked until the cooldown period has expired. However, if another alarm triggers a scale-out activity during the scale-in cooldown period, Application Auto Scaling scales out the target immediately.