AWS ELB & Auto Scaling

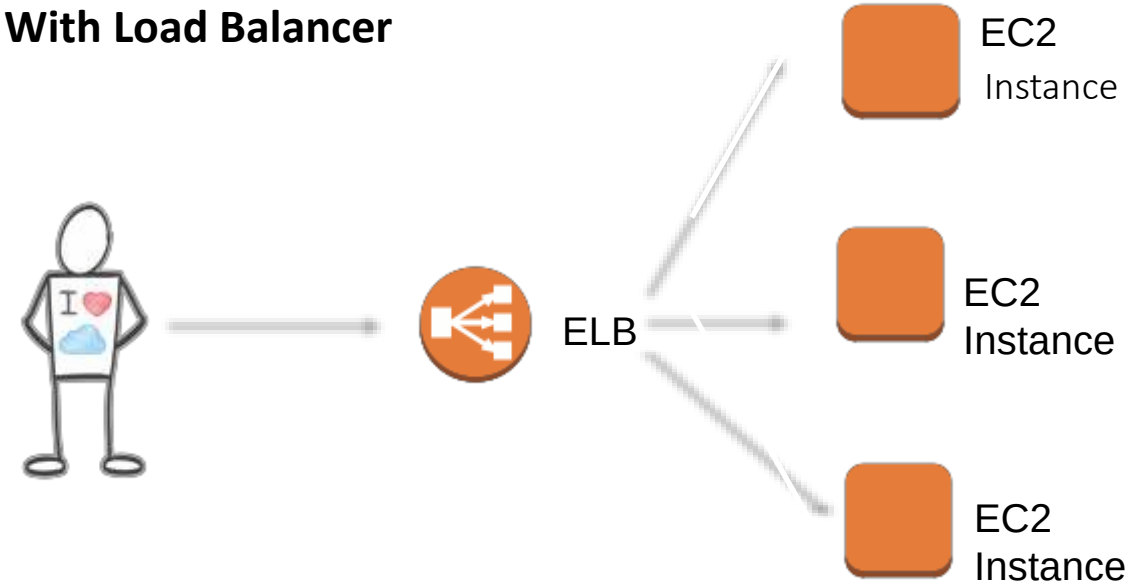
Mithun Technologies
devopstrainingblr@gmail.com
+91-9980923226

ELB (Elastic Load Balancer)

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, Containers, or IP addresses in your VPC.

With Out Load Balancer

With Load Balancer



Load balancer used to route incoming requests to multiple EC2 instances, Containers, or IP addresses in your VPC.

Elastic Load Balancer will distribute application traffic across multiple targets in different Availability Zones within the same Region.

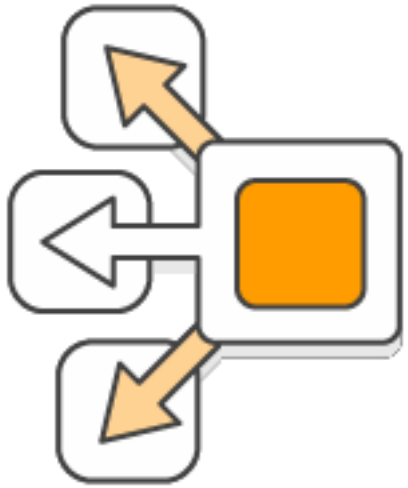
The Elastic Load Balancing Family

Application Load Balancer

HTTP & HTTPS (VPC)

Network Load Balancer

TCP Workloads (VPC)



Classic Load Balancer

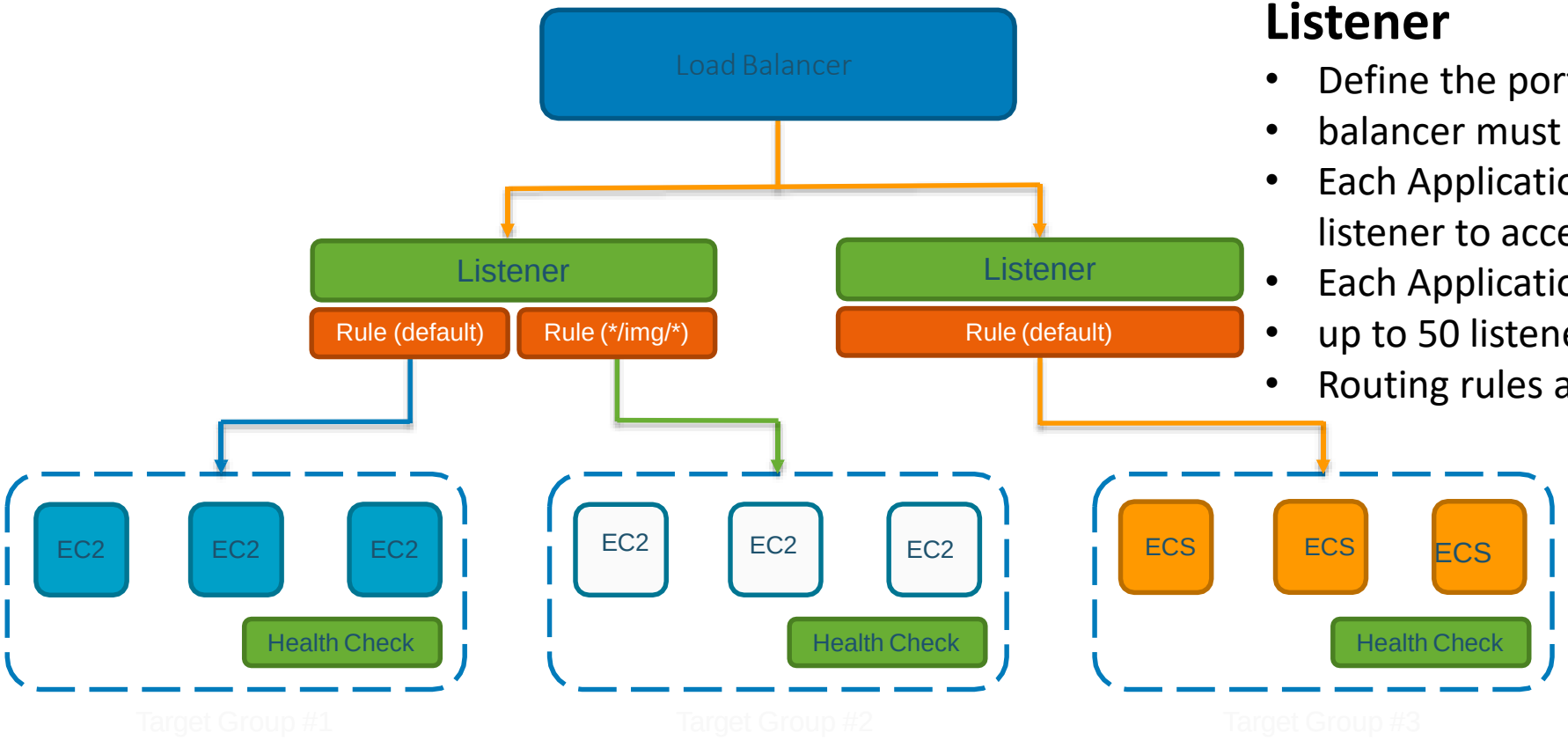
Previous Generation
for HTTP, HTTPS, TCP
(Classic Network)

- **Network Load Balancer**

  - Operates at the connection level (Layer 4), routing connections to targets – EC2 instances, microservices, and containers – within VPC based on IP protocol data.
  - Routes connections(requests) based on IP protocol(Layer 4).
  - Supports TCP/UDP Protocols.
  - Is capable of **handling millions of requests per second while maintaining ultra-low latencies**.
  - Is **optimized to handle sudden and volatile traffic patterns** .

- **Application Load Balancer**

  - Supports HTTP/HTTPS Protocols.
  - Operates at the request level (layer 7), routing traffic to targets – EC2 instances, containers, IP addresses, and Lambda functions based on the content of the request.
  - Routes based on the content of the request.
  - Supports Host-based Routing & Path-based Routing.
  - Is ideal for advanced load balancing of HTTP and HTTPS traffic, and provides advanced request routing targeted at delivery of modern application architectures, including microservices and container-based applications.



Listener

- Define the port and protocol which the load
- balancer must listen on.
- Each Application Load Balancer needs at least one listener to accept traffic.
- Each Application Load Balancer can have
- up to 50 listeners.
- Routing rules are defined on listeners.

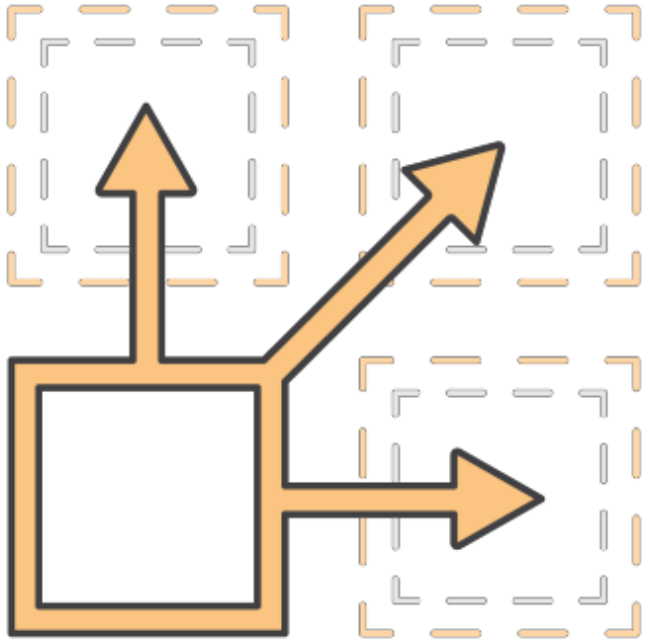
Target

- Support for EC2 instances and ECS containers, and IP Addresses.
- EC2 instances can be registered with the same target group using multiple ports.
- A single target can be registered with multiple target groups

Target Group

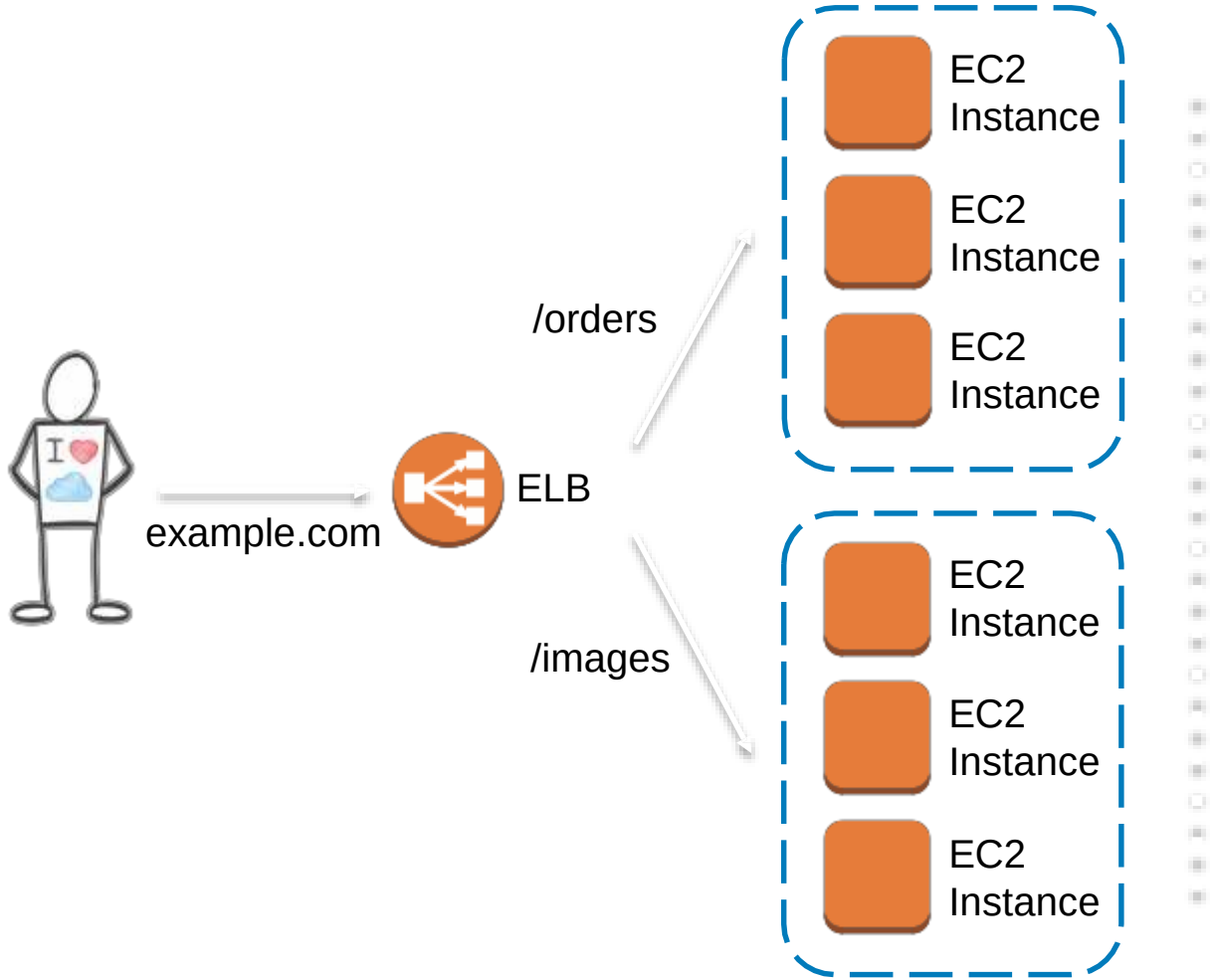
- Logical grouping of targets behind the load balancer.
- Target groups can exist independently from the load balancer.
- Regional construct that can be associated with an Auto Scaling group.
- Target groups can contain up to 1,000 targets.

Rules



- Each listener can have one or more rules for routing requests to target groups.
- Rules consist of conditions and actions.
- When a request meets the condition of the rule,
- the action is taken.
- Rules can forward requests to a specified
- target group.
- Conditions can be specified in path pattern format.
- A path pattern is case sensitive, can be up to 255 characters in length.

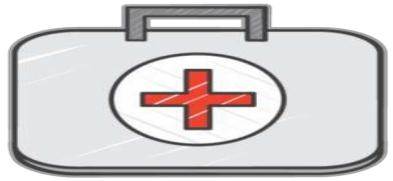
Mithun Technologies



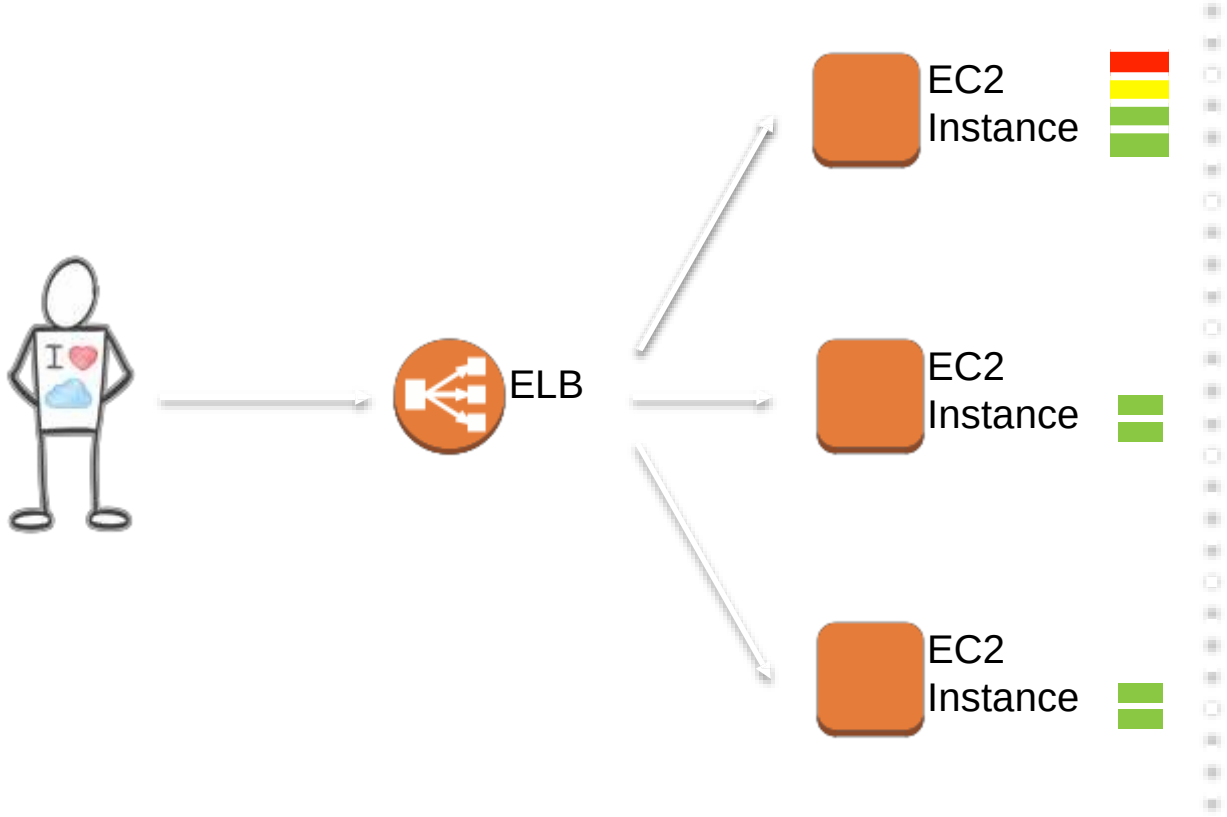
Application Load Balancer allows for multiple services to be hosted behind a single load balancer.

Mithun Technologies

Health Checks



- Health checks allow for traffic to be shifted away from failed instances.
- Support for HTTP and HTTPS health checks.
- Customize the frequency and failure thresholds.
- Consider the depth and accuracy of your health checks.
- Details of health check failures are now returned via the API and Management Console.



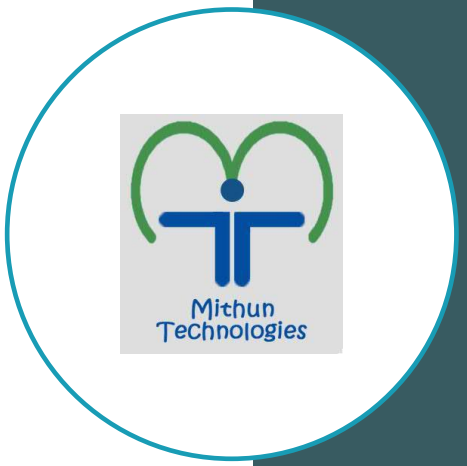
Health checks ensure that request traffic is shifted away from a failed instance.

Mithun Technologies

- **Advantages**
  - Requests distributed evenly across multiple Availability Zones.
  - Block or allow requests based on conditions such as IP addresses.
  - Preconfigured protection to block common attacks like SQL injection or cross-site scripting.
  - Monitor web requests and protect web applications from malicious requests at the load balancer.

- **Access Logs**
  - Provide detailed information on each request processed by the load balancer
  - Includes request time, client IP address, latencies, request path, and server responses.

- **Cloud Watch**
  - CloudWatch metrics provided for each load balancer.
  - Provide detailed insight into the health of the load balancer and application stack.
  - CloudWatch alarms can be configured to notify or take action should any metric go outside the acceptable range.



- **Auto Scaling Integration**
  - Auto Scaling can now scale targets within a target group.
  - Allows for applications to be scaled independently behind the Application Load Balancer.

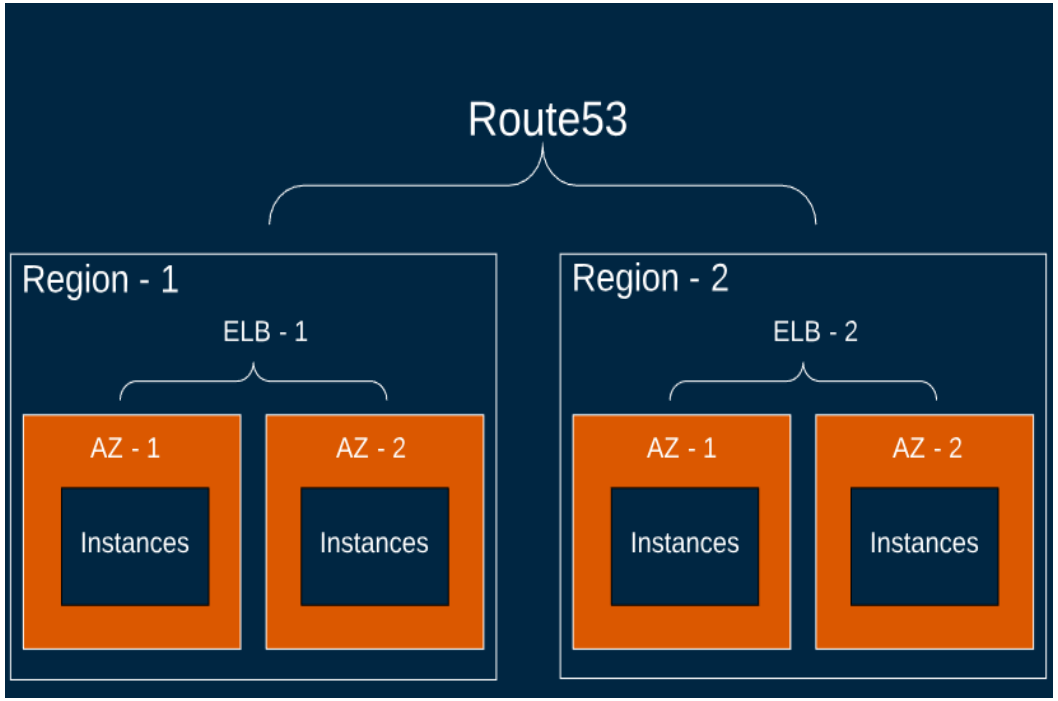
- **Pricing**
  - With the Application Load Balancer, you only pay for what you use. You are charged for each hour or partial hour your Application load balancer is running and the number of Load Balancer Capacity Units (LCU) used per hour.
  - $0.0225 per Application Load Balancer-hour (or partial hour)
  - $0.008 per LCU-hour (or partial hour)

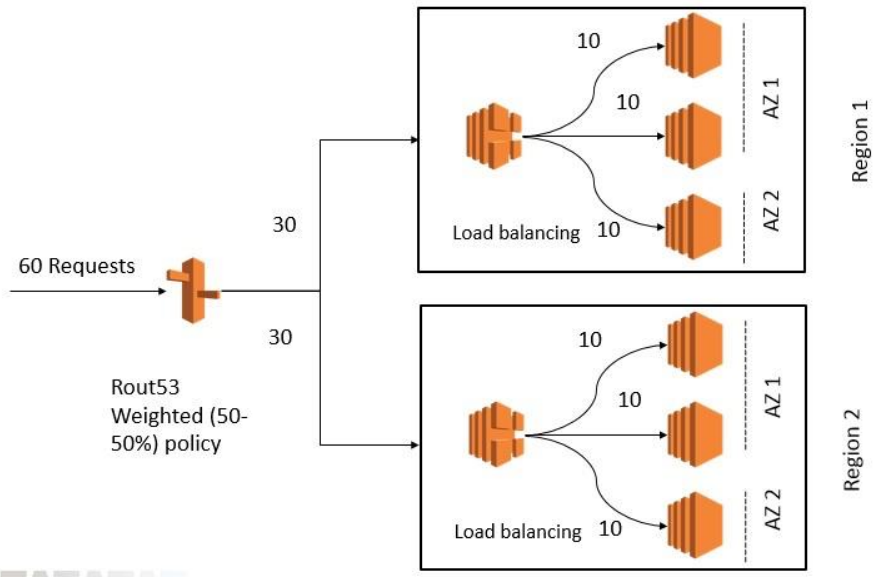
Multi Region Load Balancing

Elastic Load Balancer, will to distribute application traffic across multiple targets in different Availability Zones within the same Region. But using Route53, we can route the traffic across different Regions. So, **Multi-Region load balancing configuration possible with the combination of both Elastic Load Balancer & Route53**.

Route53 offering different routing policies as Simple, Failover, Geolocation, Latency, Weighted, Geo-proximity and Multivalue. We can select these routing policies based on application requirement during the creation of Route53 Record Set. Create Record Sets by using your Load Balancer DNS as an end-point and select the suitable route53 Routing policies.

When you provide ELB target in the Route53, It selects the Proper Region. When you create a Record Sets for ELB in the different Region make sure you are providing same name and type.





Mithun Technologies

Auto Scaling

Amazon Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.

Benefits

- Better fault tolerance. Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and launch an instance to replace it. You can also configure Amazon EC2 Auto Scaling to use multiple Availability Zones. If one Availability Zone becomes unavailable, Amazon EC2 Auto Scaling can launch instances in another one to compensate.
- Better availability. Amazon EC2 Auto Scaling can help you ensure that your application always has the right amount of capacity to handle the current traffic demand.
- Better cost management. Amazon EC2 Auto Scaling can dynamically increase and decrease capacity as needed. Because you pay for the EC2 instances you use, you save money by launching instances when they are actually needed and terminating them when they aren't needed.

Auto Scaling Concepts

Auto Scaling Groups

- EC2 instances are managed by Auto Scaling *groups*.
- Create Auto Scaling groups by defining the minimum, maximum, and, optionally, the desired number of running EC2 instances.

Launch Configuration

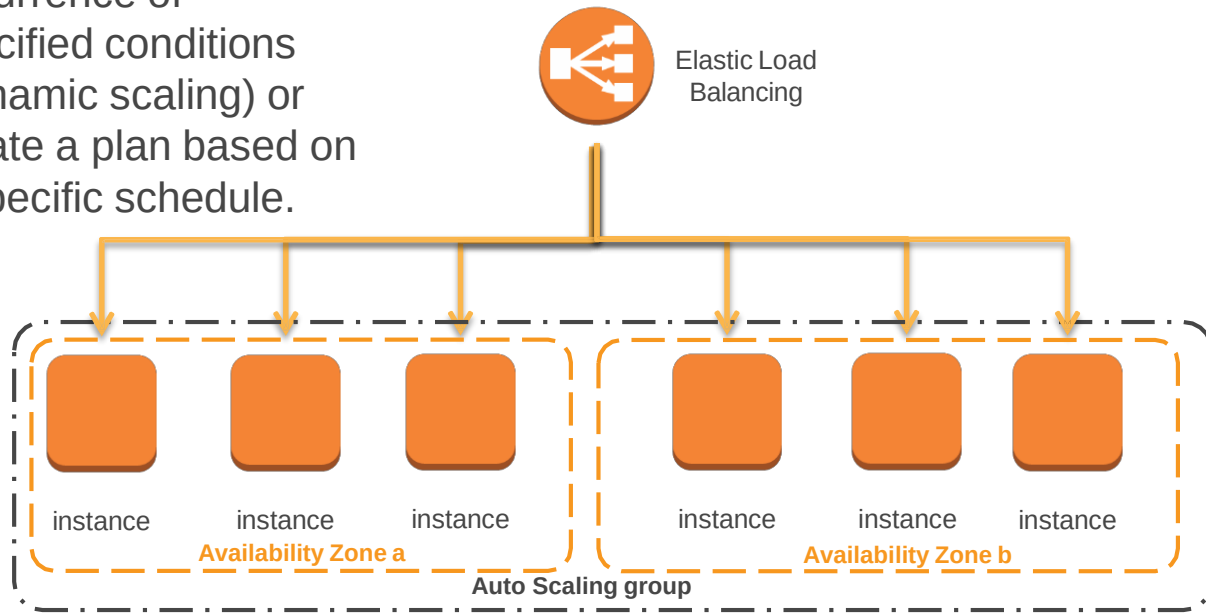
- Auto Scaling groups use a *launch configuration* to launch EC2 instances.
- Provides information about the AMI and EC2 instance types/size

Scaling Plan

- A scaling plan tells Auto Scaling when and how to scale.
- Create a scaling plan based on the occurrence of specified conditions (dynamic scaling) or create a plan based on a specific schedule.

Auto Scaling Groups

- Always keep **minimum** number of instances running
- Launch or terminate instances to meet **desired capacity**
- Never start more than **maximum** number of instances
- Keeps capacity **balanced** across AZs



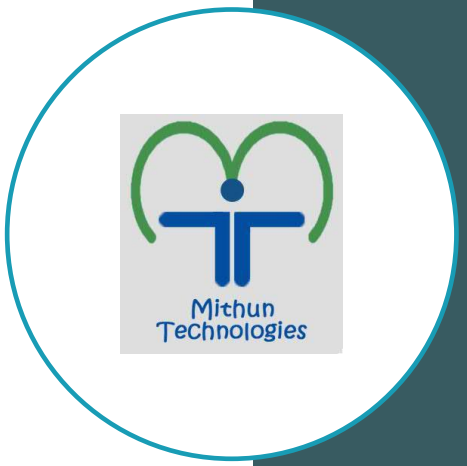
Minimum = 2 **Desired # of instances = 6** Maximum = 10

Launch Configurations

- Determine **what** is going to be launched:
  - EC2 instance type & size
  - Amazon Machine Image (AMI)
  - Security groups, SSH keys, IAM instance profile
  - User data

Bootstrapping

- Installation & setup needs to be fully automated:
  - Use Amazon Machine Image (AMI) with all required configuration & software ("golden image")
  - Base AMI + install code & configuration as needed
    - Via User data
    - Via Chef/Puppet/Ansible/…
    - Using AWS Code Deploy



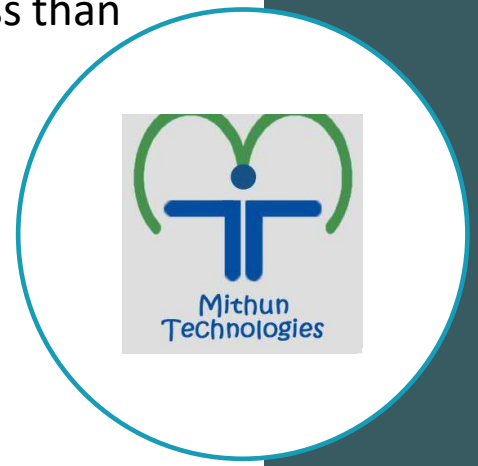
Scaling Plans

Determine when the Auto Scaling group will scale in or out:

- desired capacity > current capacity: launch instances
- desired capacity < current capacity: terminate instances

Plans

- Default: ensure current capacity of **healthy** instances remains within boundaries (never less than minimum).
- 'Manual scaling': modify desired capacity (via API, console, CLI) to trigger a scaling event
- Scheduled: scale in / out based on timed events.
- Dynamic scaling: scale on Amazon CloudWatch metrics.

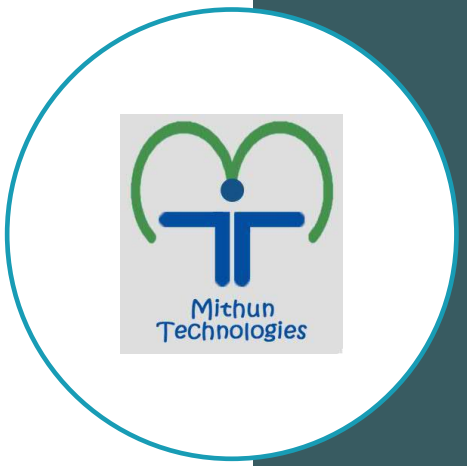


Termination Policy

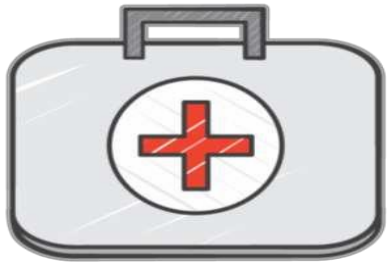
- Determine which instances are terminated first:
  - Longest running
  - Oldest launch configuration
  - Closest to full billing hour
- But: rebalancing of capacity across AZs takes precedence!

Load Balance your Auto Scaling Group

- Configure your Auto Scaling Group to work with one or more Elastic Load Balancers
- Automatically registers new instances, deregisters on termination.
- Use ELB health checks in your Auto Scaling Group.
- Use Elastic Load Balancing metrics in scaling policies.



Health Checks



- Performed periodically
- Instances are marked as unhealthy or healthy
- Unhealthy instances are terminated and replaced
  - (if new number of instances < minimum or < desired capacity)
- EC2 instance status:
  - Instance is unhealthy when instance state !='running'
  - **or** system health check =='impaired'
- ELB health checks:
  - instance is unhealthy when ELB health check results in "OutOfService" (**or** EC2 health check failed)
- Manual: mark individual instances as 'unhealthy' Instance unhealthy when marked as such **or** EC2 health check failed. Use to integrate with external monitoring systems.

Mithun Technologies

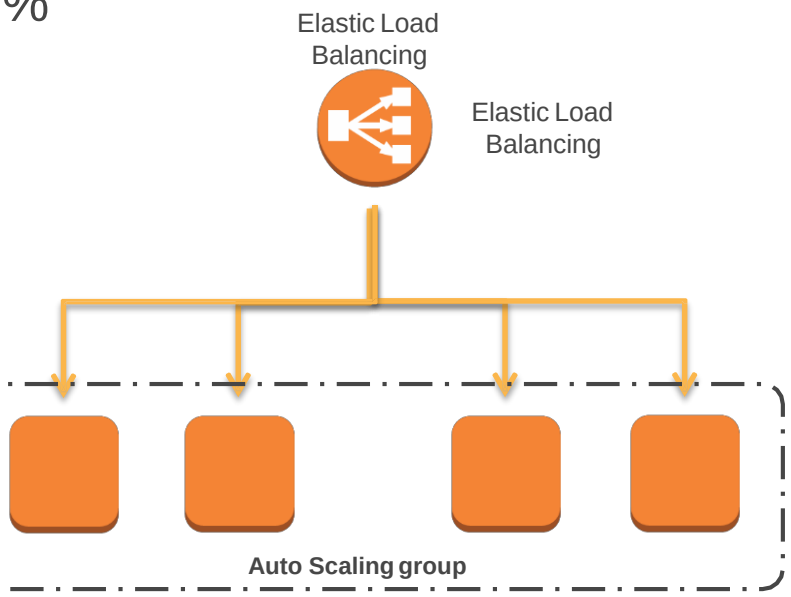
Scaling Policies

- Can change capacity of the group in 3 different ways:
- Set fixed capacity, e.g., desired capacity = 4 instances
- Add / remove fixed number of instances, e.g., + 2
- Add / remove percentage of existing capacity, e.g. +20%

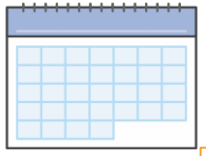
Scaling on a Schedule

cron-like syntax for recurring scaling events

Scaling event



Schedule individual events (up to 135 events per group)



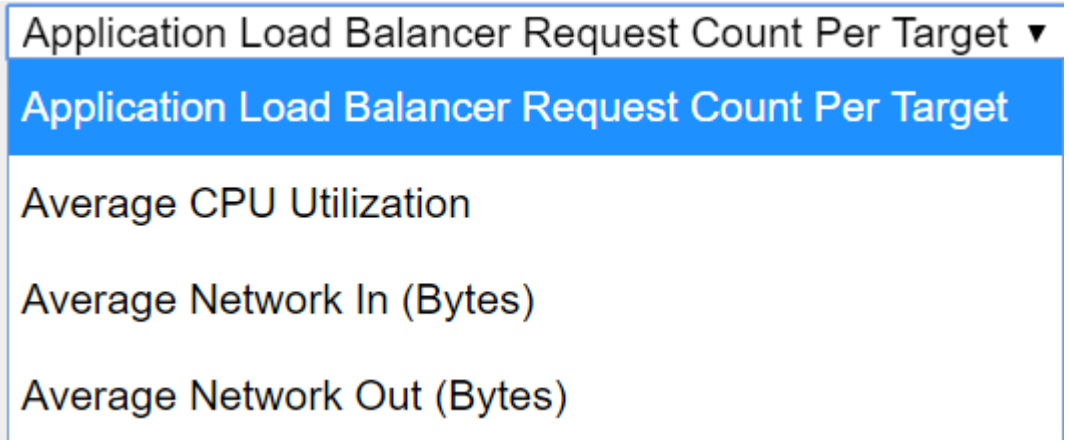
Scaling event

Auto Scaling CLI & SDK only!
Not available in AWS management console

Dynamic Scaling Policies

- Trigger scaling events based on demand:
- Demand is measured based on metrics
- Changes in metrics can be mapped to scaling policies

Metric Types



Amazon CloudWatch

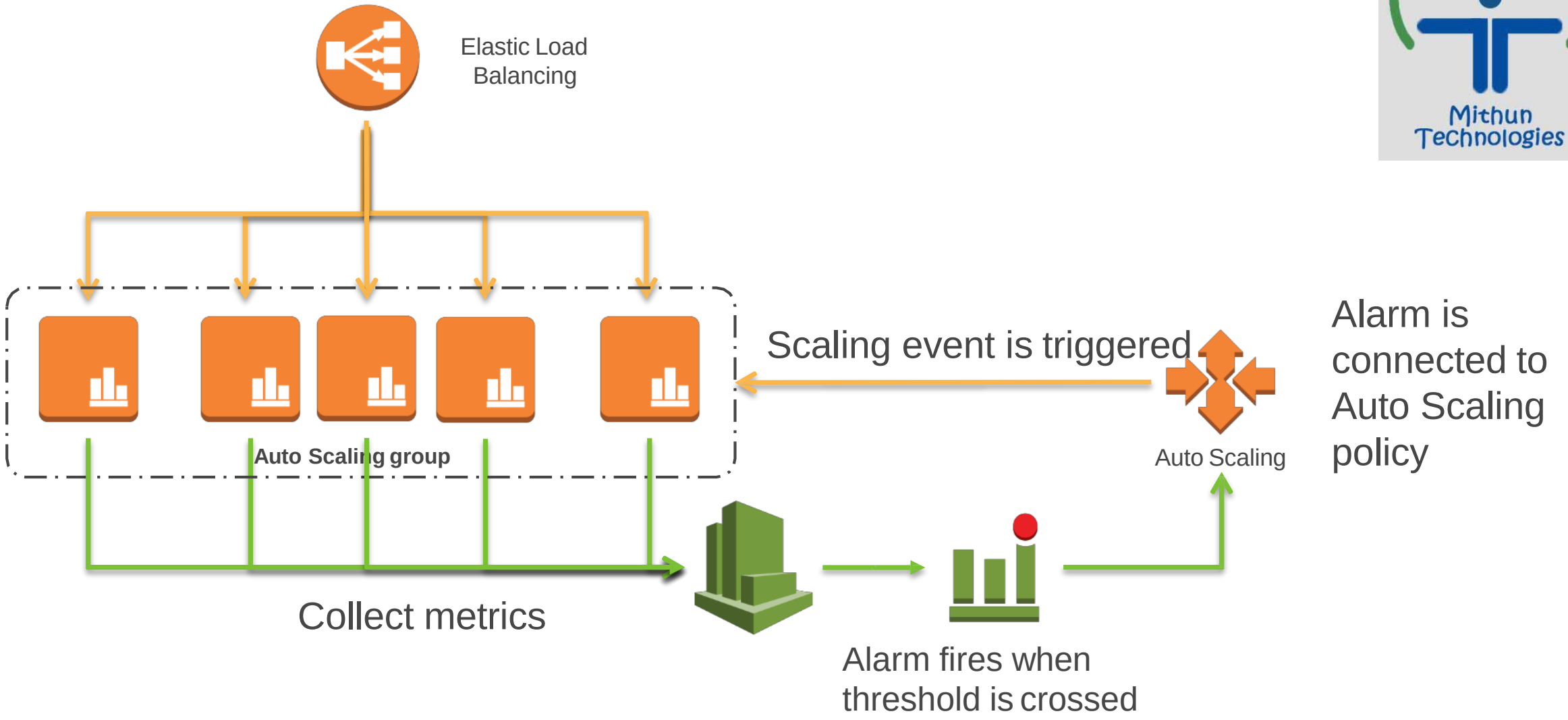
Amazon **CloudWatch**:A web service that enables you to monitor and manage various metrics, and configure alarm actions based on data from those metrics.

A **metric** is a variable that you want to monitor, e.g., CPU usage or incoming network traffic.

A **CloudWatch *alarm*** is an object that monitors a single metric over a specific period.

The alarm changes its **state** when the value of the metric breaches a defined range and maintains the change for a specified number of periods.

How Alarms Work



Amazon CloudWatch can aggregate metrics across pre-defined dimensions, e.g., aggregate average CPU utilization of all EC2 instances in an Auto Scaling group.

Questions ?

Mithun Technologies
devopstrainingblr@gmail.com
+91-9980923226