Elastic Load Balancing (ELB)

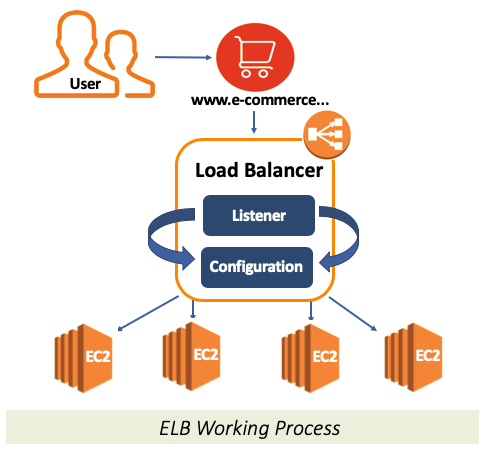
ELB (Elastic Load Balancing) is a service allowing to make the load distribution for your existing resources (instances) in AWS.

It automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers.

We have set up 1 server and started our web site. Let's assume that this is an e-commerce website and it should be running 24/7. But, if our virtual machine fails for any reason, it will probably take 30 minutes to make it work again because of installation, adjustment, etc. Therefore, you will lose customers and money.

We need to construct the infrastructure over multiple servers. And this time you need to direct the traffic between these servers. This is what we call the **Load Balancer** that tackle this job for you.

### Load Balancer Working Process



ELB (Elastic Load Balancing) is basically a network service that we can direct the incoming traffic to Load Balancer instead of direct virtual machines.

In this way, we can regulate the traffic according to the rules we determined and direct the traffic to the target instance.

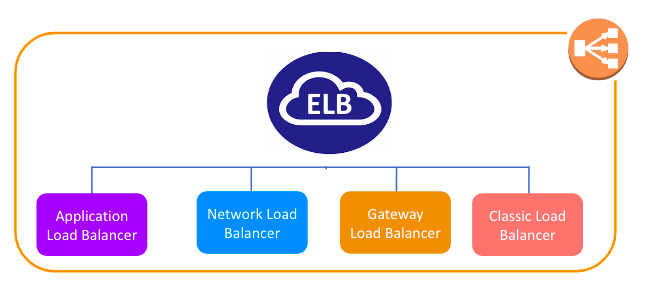
Load Balancer basically consists of 2 components.

* **Listener**
* **Configuration**

**Listener**, as the name suggests, listens to the incoming traffic through a specific port according to the component adjustments.

**Configuration** is a set of rules that takes into account the Listener's analysis and provides to direct traffic.

### Types of ELB:



### Application Load Balancer

It is operated at Layer 7 of the OSI Model

It identifies the incoming traffic and forwards it to the right resources. You can route the request according to the information here by reading the **contents of the package**.

In other words, it directs traffic according to the contents of the package. For example, if a URL has /API extension, then it is routed to the appropriate application resources.

If we have tasks like HTTP, HTTPS-based interactive web pages, mobile applications, and containers, etc., we'll use the Application Load Balancer. It is best suited for the load balancing of **HTTP and HTTPS** traffic.

Like Network Load Balancer, you can use the Application Load Balancer compatible with **Auto Scaling** function of AWS.

### Network Load Balancer

It is operated at Layer 4 of the OSI model.

It makes routing at the transport layer (TCP/SSL), and it can handle millions of requests per second. It is best suited for load balancing the **TCP traffic** when **high performance is required**.

When a load balancer receives a connection, it then selects a target from the target group by using a flow hash routing algorithm. It doesn't look at the contents of the package, it does the basic routing according to the rules on it.

If we need a **simple and fast** load balancer over a basic TCP-based port, this is the Network Load Balancer.

Gateway Load Balancer

It is operated at Layer 3 (Network Layer) of the OSI model.

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.

It listens for all IP packets across all ports and forwards traffic to the target group that's specified in the listener rule.

### Classic Load Balancer

It has been serving at Layer 4 of the **OSI** model and the oldest type of ELB.

Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level.

Classic Load Balancer is intended for applications that were built within the EC2-Classic network.

It can also be used for load balancing the HTTP or HTTPs traffic and use layer 7-specific features. AWS doesn't recommend you to use Classic Load Balancer anymore.

## Auto Scaling:

## Auto scaling automate the increasing or decreasing of my existing virtual machines when demand changes over time.

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

## We can add or remove a new virtual machine when it is necessary. So, you can expand your system to respond to that demand according to the size of the request you receive, and automatically collapse when the demand decreases.

### Benefits of Auto Scaling



Setup Scaling Quickly:

AWS Auto Scaling lets you set target utilization levels for multiple resources in a single, intuitive interface. You can quickly see the average utilization of all of your scalable resources without having to navigate to other consoles.

Pay Only for What You Need:

AWS Auto Scaling can help you optimize your utilization and cost efficiencies when consuming AWS services so you only pay for the resources you actually need. When demand drops, AWS Auto Scaling will automatically remove any excess resource capacity so you avoid overspending. **AWS Auto Scaling is free to use and allows you to optimize the costs of your AWS environment.**

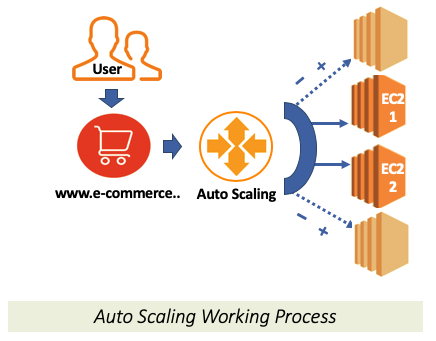
Automatically Maintain Performance:

Using AWS Auto Scaling, you maintain optimal application performance and availability, even when workloads are periodic, unpredictable, or continuously changing. AWS Auto Scaling continually monitors your applications to make sure that they are operating at your desired performance levels.

Make Smart Scaling Decisions:

If you try to make scaling instead of AWS Auto Scaling you may not always find an optimum solution. AWS Auto Scaling automatically creates all of the scaling policies and sets targets for you based on your preference. AWS Auto Scaling monitors your application and automatically adds or removes capacity from your resource groups in real-time as demands change.

### Auto Scaling Working Process



### Installing Stress Tool

* First connect to ec2 instance via Terminal
* Get root privileges

~ sudo su

* Then install and run stress application with the following commands and increase CPU usage.

~ yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

~ yum install stress

~ stress --cpu 80 --timeout 2000

(or)

~ sudo amazon-linux-extras install epel -y

~ sudo yum install -y stress

~ stress --cpu 80 --timeout 20000

* Stress tool will start to increase the CPU usage of the instance.
* Do the same process for the other running instances if needed. The CPU utilization will gradually increase.
* Wait for a while to see the results.