Elastic Load Balancing (ELB)

Introduction to ELB (Elastic Load Balancing)

Elastic Load Blancing

ELB (Elastic Load Balancing), as the name suggests, 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.

Let's try to understand this load distribution issue through the web server established in the previous lesson.

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.

So, there are many shortcomings in setting up a single machine so, 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 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.

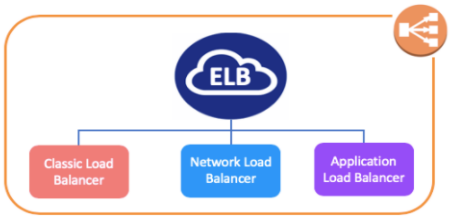
Load Balancer basically consists of 2 components.

- **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.

Type of ELB



ELB Types

There are 3 types of ELB in AWS; Classic Load Balancer, Application Load Balancer, and Network Load Balancer.

Classic ELB:

It has been serving at Layer 4 of the **OSI** model since 2009 and the oldest type of ELB.

Tips:

- OSI stands for Open System Interconnection. It is a reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer. OSI consists of seven layers, and each layer performs a particular network function.

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.

Network Load Balancer

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. So 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. So 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.

Application Load Balancer

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 extensions, 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.

Scenario for Load Balancer

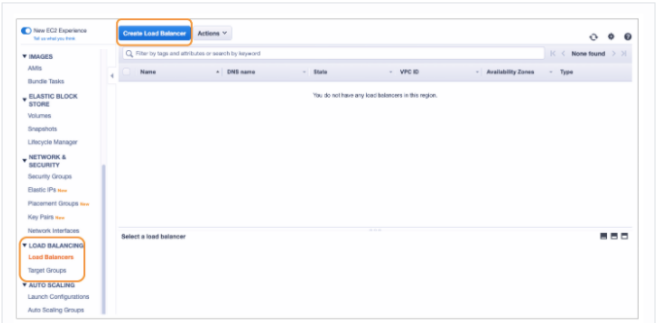
ELB Structure

Suppose that we don't want any interruption on our website. So we create a second machine with the snapshot which is the exact copy of our first server.

So, we have 2 web servers. Now, if something happens to our instance, we can use the other one. But, we need to make many settings manually and we have to be aware of any malfunction immediately to direct the traffic to the second server without delay.

Here you need Load Balancing.

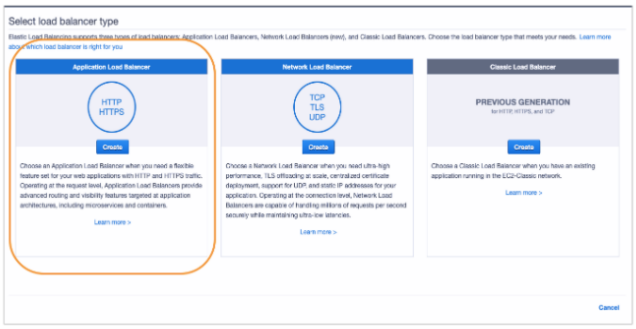
Creating Load Balancer



Creating Load Balancer

In the EC2 AWS Management console, select Load Balancers from the left-hand menu and then click **Create Load Balancer** to start creating ELB.

Select The Load Balancer Type



Load Balancer Types

First of all, you need to decide which load balancer you prefer to use. Here, we have 3 options. These are Application Load Balancer, Network Load Balancer, and Classic Load Balancer as you see in the picture above.

We already have detailed descriptions on the page and we have also covered these issues in the previous lessons. But let's take a short look.

- **Classic Load Balancer:**

One of the oldest services AWS has to offer but is gradually being deactivated.

- **Application Load Balancer:**

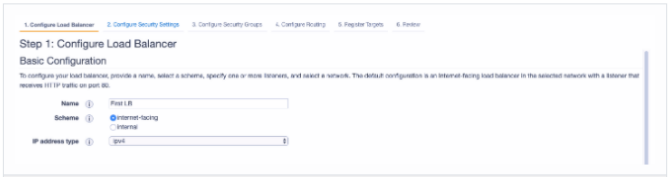
It is used for complex tasks.

- **Network Load Balancer:**

It is ideal for simple and basic TCP-based connection options.

Our job in our scenario is quite simple, but we will continue with the Application Load Balancer to see the menu options in detail.

Step 1 - Configure Load Balancer



Configure Load Balancer Page

- **Name:**

We write the name of LB as **First-LB**.

- **Scheme:**

In this part, we have **Internet-facing** or **Internal** options.

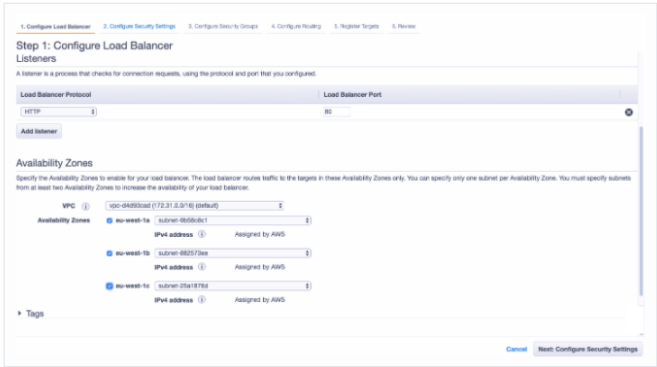
If we choose **Internet-facing**, this Load Balancer can handle the traffic coming from the internet, ie from the outside world, and distribute it to the internal resources.

But, if you select **Internal**, this time it works only in VPC and is used for our Load Balancing operations within VPC.

In our scenario, we will distribute the traffic from the outside world to 2 servers inside and reach us from the internet. So let's choose Internet-facing.

- **IP address Type:**

We have Ipv4 and dual-stack options. We will choose the Ipv4. It's enough for this process.



Configure Load Balancer Page

- **Listener:**

In the previous lessons, we said that Load Balancers have two components. One of them is the listener. The listener will roughly listen to which port on which protocol this load balancer is from the outside. Since we will receive traffic from the outside via HTTP, we select HTTP 80 port for now.

- **Availability Zone(AZ):**

In this section, we can specify the Availability Zones to enable our load balancer. AWS suggests we specify subnets from at least two Availability Zones to improve the availability of your load balancer. We select default VPC and all the AZ options.

- **Tag:**

You can enter a tag if you want.

Step 2 - Configure Security Settings

Configure Security Settings

Here, we will encounter a warning. Because we have opened the HTTP protocol, and it is not a secure protocol exactly. So AWS suggests opening an HTTPS and using a certificate.

However, we do not need to open HTTPS within the scope of training. Therefore we do not take this warning into account.

Click **Next** and go on.

Step 3 - Configure Security Groups

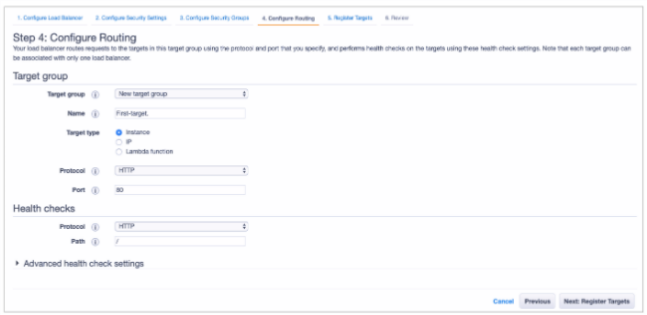
Configure Security Groups

Here, we associate the security group with the load balancer. You can use an existing security group or create a new one. But, we select the existing security group.

Click **Next** and go on.

Step 4 - Configure Routing

In the Listener settings section, we specify which port the load balancer will listen to. Now It's time to arrange inbound traffic. We use **Target Group** to distribute the traffic in AWS. Now, let's determine the rules as seen on the page below.



Configure Routing Page

- **Target group:**

You can create a new target group or use the existing one. We prefer to create a new one.

- **Name:**

We determine the name of the target group as **First-target**.

- **Target Type:**

We have 3 options. These are Instance, IP and Lambda function.

If we select the **Instance** option, the EC2 instances we created will be targeted. But, if we select **IP** option, we do it by entering specific IPs. As the last option, we can also set this load balancer to load multiple Lambda Functions.

We will select **Instance** as we will connect with EC2 instances in our scenario,

- **Protocol and Port:**

We select protocol and port that the load balancer uses when routing traffic to targets in this target group. So we select HTTP and port 80.

Step 4 - Configure Routing (What is Health Checking ?)

Health Checking

Health Checking is a function used to determine whether the instances working properly in load balancing.

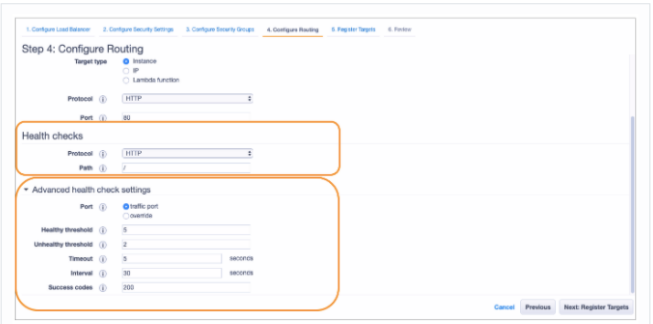
The load balancer checks if the instance working properly, and if it's healthy, the load balancer sends traffic to the instance. If it is not working properly, it will direct the traffic to the other healthy instance to prevent interruptions. This is called **Health Checking.**

Thus, clients connected from the outside will continue to operate without any interruption thanks to the load balancer backed by health checking.

The load balancers can fulfill this action over the HTTP, HTTPs and TCP protocol.

All load balancer types are compatible with health checking.

Step 4 - Configure Routing (Health Checks & Advanced Health Check Settings)



Health Checks & Advanced Health Check Settings

- **Health Checks:**
  - **Protocol:**
    Here we select HTTP again as Protocol
  - **Path:**
    It will be the destination path for health checks. The default "/" sign available for us. It means LB is going to check the homepage of the website our server is related to.
- **Advanced Health Check Settings**

This is the part where we set the rules on how to control the path "/"mentioned above.

- **Port:**

This is the port the load balancer uses when performing health checks on targets. The default option is **Traffic Port**. Keep it as is.

- **Healthy Threshold:**

Here, you decide the number of requests that will be sent to the machine in the background. The default value is 5. It means this machine will query the machine inside 5 times, if it gets positive results from five, this machine will evaluate as healthy.

- **Unhealthy Threshold:**

We determine how many errors provide LB to decide this machine is unhealthy in this part. If LB cannot get an answer 2 times in 5 queries, it will decide that this server is unhealthy.

- **Timeout:**

We write the maximum number of seconds to wait in case of connection failure. **5 seconds** is the default value.

- **Interval:**

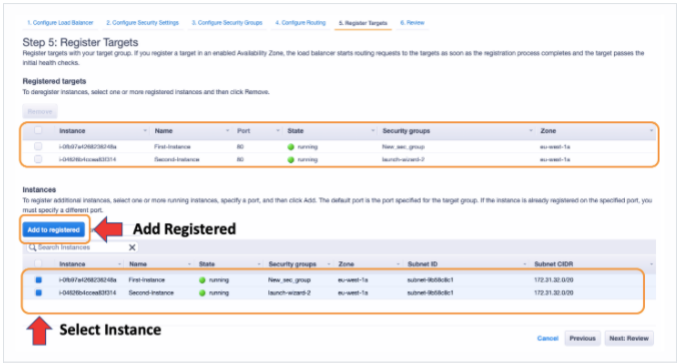
This part shows you how many seconds will pass between each experiment. **30 seconds** is the default value and available for us.

- **Success Codes:**

Success code is the code that ensures that the answer is healthy. If it receives 200 of the HTTP protocol, it will evaluate to healthy, otherwise, it will say unhealthy.

According to our settings; LB will send the request 5 times for 150 seconds and if it receives a positive response from all five, it will decide the machine is healthy. But if it gets 2 negative answers, this time it will decide that this machine is unhealthy.

Step 5 - Register Targets

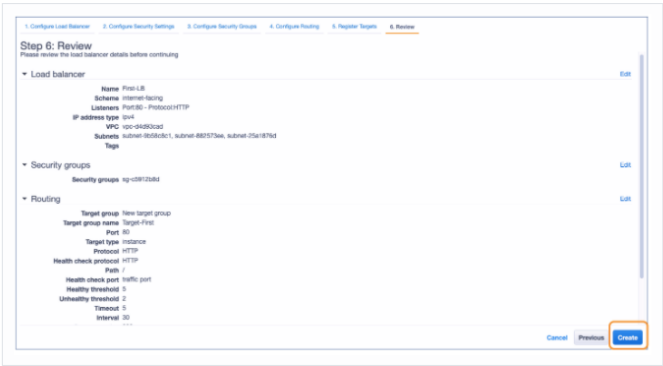


Register Targets

We have 2 servers listed in the registered targets as you see in the picture above. Now, we need to make them register.

- So, select both of them by clicking the boxes at left.
- After that, add them to the **Registered Targets** by clicking the **Add to Registered** tab seen above.
- They are now under the target group. So we're going to start to use them synchronous with the load balancer.

Step 6 - Review



Review

- Now, let's check all the settings we made on the page seen above,
- Then, when we click **Create** tab to create the load balancer.

Now our two servers working with under control of load balancer. The load balancer will analyze the health of these instances and it will distribute the traffic according to this evaluation.