Elastic Load Balancing

08 June 2023 11:44

Elastic Load Balancing

Distribute network traffic to improve application scalability

Classic Load Balancer :

Load balance across multiple Amazon EC2 instances at the request level and the connection level.

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 are built within the EC2-Classic network.

We recommend Application Load Balancer for Layer 7 traffic and Network Load Balancer for Layer 4 traffic when using Virtual Private Cloud (VPC).

Application Load Balancer :

Load balance HTTP and HTTPS traffic with advanced request routing targeted at the delivery of modern applications.

Application Load Balancer 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. Ideal for advanced load balancing of HTTP and HTTPS traffic, Application Load Balancer provides advanced request routing targeted at delivery of modern application architectures, including microservices and container-based applications. Application Load Balancer simplifies and improves the security of your application, by ensuring that the latest SSL/TLS ciphers and protocols are used at all times.

Network Load Balancer :

Load balance Transmission Control Protocol, User Datagram Protocol, and Transport Layer Security traffic with high performance.

Network Load Balancer operates at the connection level (Layer 4), routing connections to targets (Amazon EC2 instances, microservices, and containers) within Amazon VPC, based on IP protocol data. Ideal for load balancing of both **TCP and UDP traffic**, Network Load Balancer is capable of handling millions of requests per second while maintaining ultra-low latencies. Network Load Balancer is optimized to handle sudden and volatile traffic patterns while using a single static IP address per Availability Zone. It is integrated with other popular AWS services such as Auto Scaling, Amazon EC2 Container Service (ECS), Amazon CloudFormation, and AWS Certificate Manager (ACM).

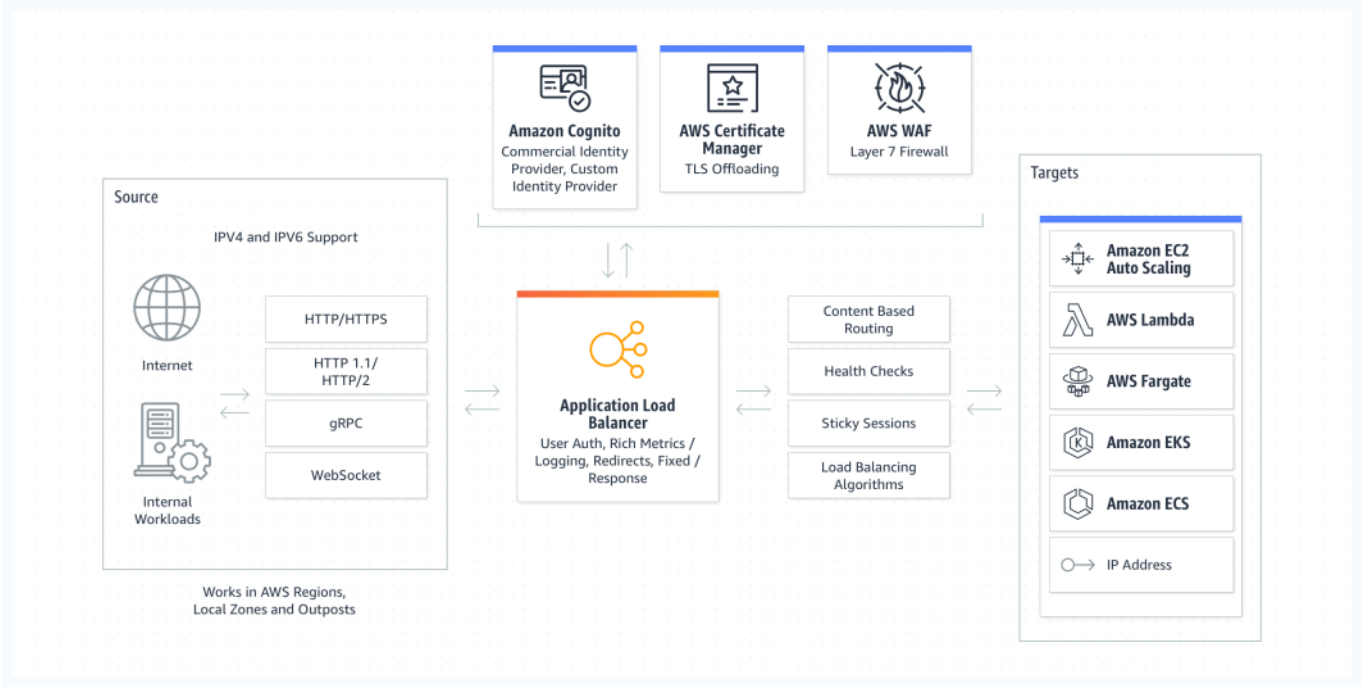
Gateway Load Balancer :

Deploy, scale, and run third-party virtual appliances

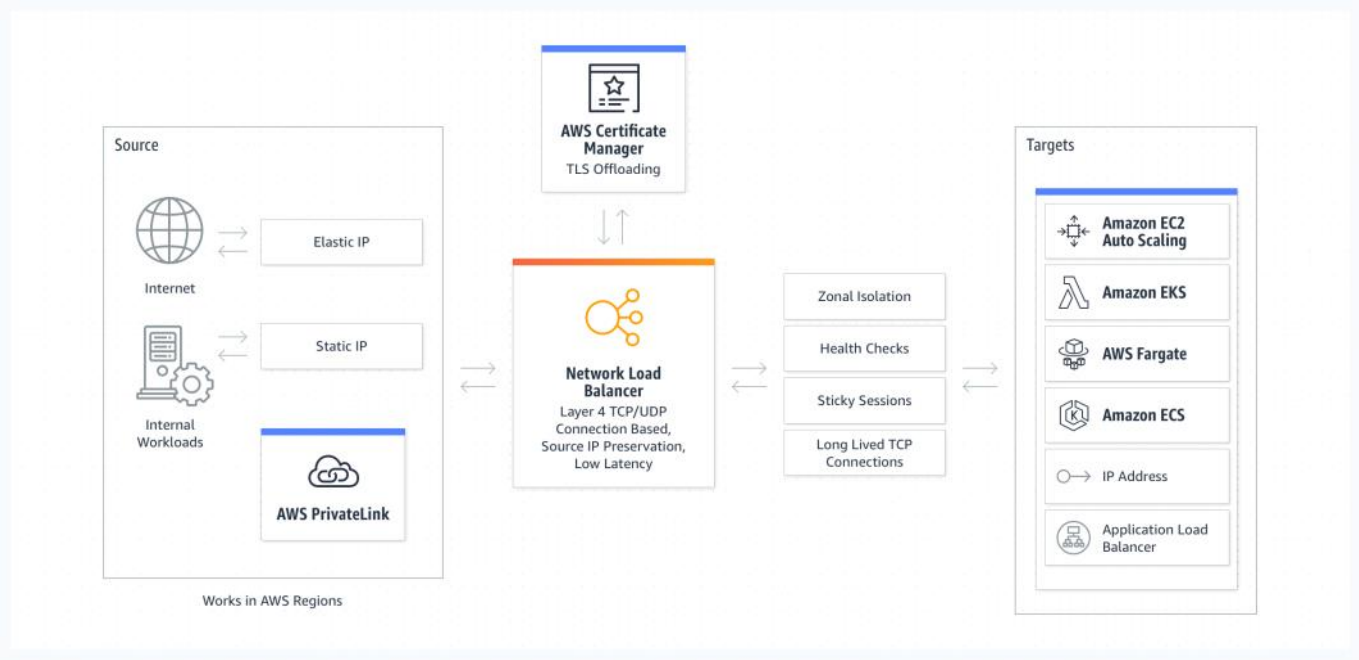
Gateway Load Balancer helps you easily deploy, scale, and manage your third-party virtual appliances. It gives you one gateway for distributing traffic across multiple virtual appliances while scaling them up or down, based on demand. This decreases potential points of failure in your network and increases availability.

You can find, test, and buy virtual appliances from third-party vendors directly in AWS Marketplace. This integrated experience streamlines the deployment process so you see value from your virtual appliances more quickly—whether you want to keep working with your current vendors or try something new.

Application Load Balancer :



Network Load Balancer



ELB Schemes

- Internet-facing: *ELB nodes have public IP addresses.*
- Internal: *ELB nodes have private IP addresses.*

Both internet-facing and internal load balancers route requests to your targets using private IP addresses. Therefore, your targets do not need public IP addresses to receive requests from an internal or an internet-facing load balancer.

ELB Target Types

ELB distribute incoming traffic to:

- Instances *(EC2, EC2 with Auto Scaling, Containers with ECS)*
- IP addresses *(VPC Subnets, RFC 1918 CIDR, On-premises with Direct Connect or Site-to-Site VPN)*
- Lambda functions *(for ALB type)*

ELB Key Points

- ELB is one of the most the ideal solution for adding elasticity to your application.
- ELB does not have predefined IPv4 addresses, it is resolved using DNS name.
- ELB can manage traffic within a region and not across multiple regions.
- You can specify only one subnet per Availability Zone (AZ).
- ELB has more than one listeners. e.g. HTTP, HTTPS.
- For internet-facing load balancers, the IPv4 addresses of the nodes are assigned by AWS. For internal load balancers, the IPv4 addresses are assigned from the subnet CIDR.
- To ensure that traffic is evenly distributed, you need to ensure the "*Cross-Zone Load balancing*" option is enabled.

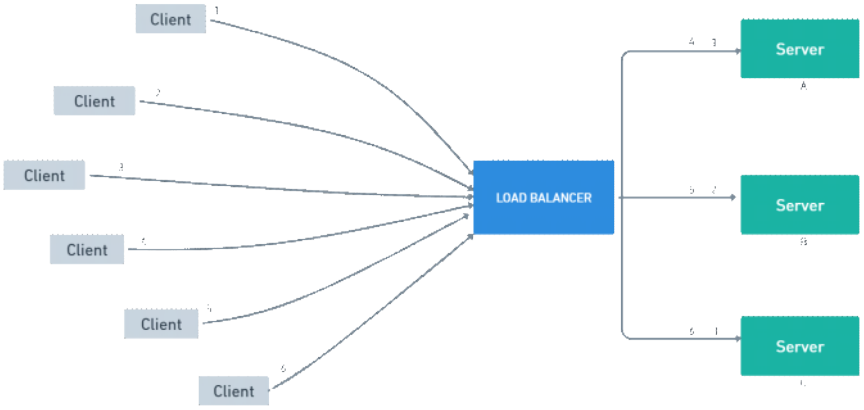
Distribute traffic across multiple instances: how do they do this? Depending on the protocol used, they do this in two ways:

1. Round Robin: Each instance has its turn in an equal manner, one after the other gets a request. It doesn't take into account how many requests the instance is already handling. TCP uses this on the Classic Load Balancer and so does HTTP/HTTPS on the Application Load Balancer.
2. Least outstanding requests (used by HTTP/HTTPS on the Classic Load Balancer)

Load balancing algorithms

1) Round-Robin

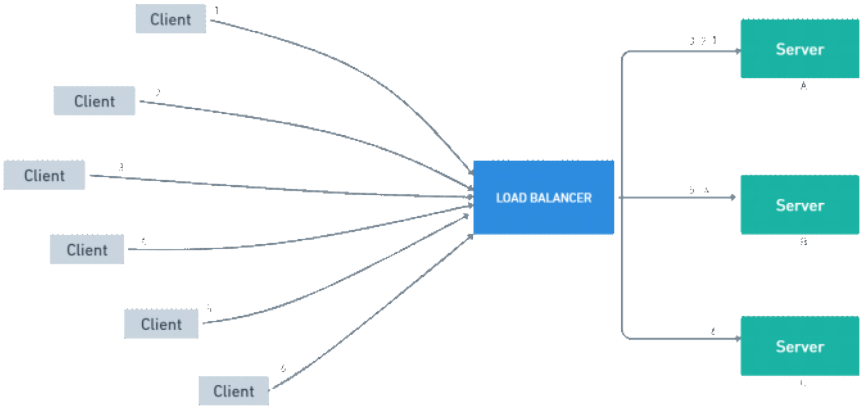
In round robin ,the requests from the client is distributed in the cyclic manner. What is that mean, we will see with the help of diagram



In the above diagram, let's say we have three servers A, B, and C and requests from the clients 1,2,3,4,5,6 come to load balancer. The load balancer will forward the request from client's 1 to server A, client's 2 requests to server B, client's 3 requests to server C. Now, for client's 4 requests, the load balancer will forward back to server A, client's 5 requests to server B and the same cycle repeats.
Loads are evenly distributed which increases the responsiveness of the servers.
But what will happen, if the server B will have higher RAM,CPU and other specs than the server A and C. In that case ,server A and C may get overloaded and fail quickly, while server B sit idle.
This method can be preferred where the servers configuration are same.

2) Weighted Round Robin

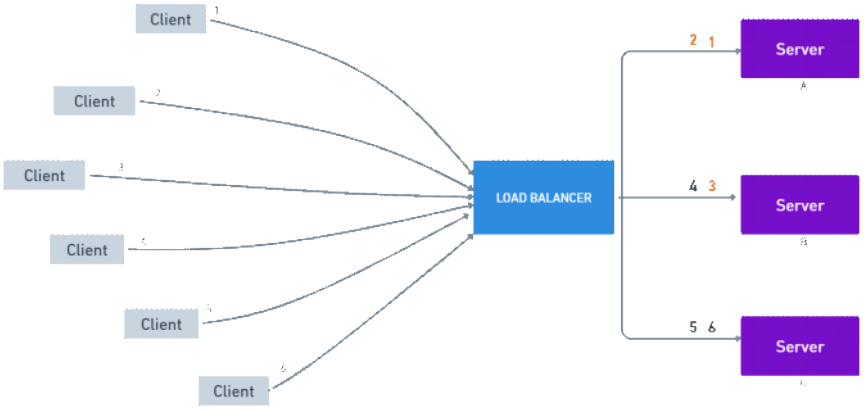
.
Dealing with different configurations of the servers, the administrator can assign the weight or ratio to the server, depending on the request it can handle. Let say, server A can take 3 requests per second, server B can take 2 requests per second on an average, and server C can take 1 requests per second.
So the load balancer will assign a weight to the server A=3,B=2,C=1.You can see the diagram below



Now, if the request comes from the clients, the load balancer will forward the first three request to server A, then client's 4 and 5 request to server B and client's 6 request to server C. After this, if there will be 7,8,9 requests would be there, the same cycle will be repeated like round-robin.

3) Least Connections

In round-robin and the weighted round-robin, we can see that the load balancer is not taking consideration of the current load connections on each server.
Let say we are taking first case where all the servers have same configuration. See the diagram,



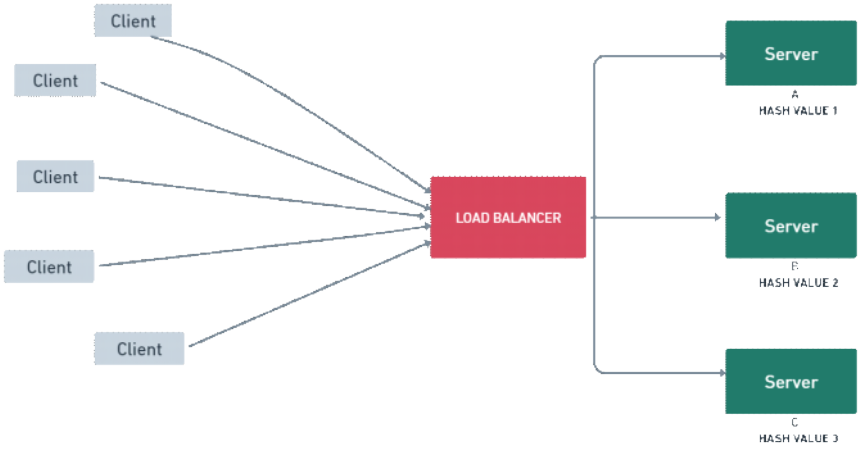
On server A the client's 1 and 2 requests are not disconnected yet
On server B client's 3 request are not disconnected. But the client's 4,5 & 6 requests are already disconnected. Now if the new request comes in so according to the round-robin algorithm, it will be forwarded to server A, then server B, and then server C. Now from here, we can see loads on server A to pile up and server A resources may be exhausted quickly.
So, here least connection method can play a major role. The least connection algorithm takes consideration of the current load on each server, so for new upcoming requests, the load balancer will forward that request to the server which has the least connections.

4) Hashing Methods

Hashing algorithms are used in the case of persistent connections(means stick a client to the specific server).This may be due to wide range content is serving to the clients like videos. Cache to be served, this reduces the response latency ,better CPU utilization.
Different hashing methods can be used like:

- **URL Hash method**
- **Souch IP Hash method**

The below diagram is shown the how requests from the client is hash based on it's URL or IP address, forwarded to the respective server.



URL Hash Method

Load balancer generates the hash value based on the HTTP URL present in requests coming from the clients. Based on hash value, requests will be forwarded to servers. So if the same request coming for the same url, it will be send to the same server.

Source IP Hash Method

In this method ,the client(or source) and server(or destination)
IP address is used to generate the hash key. Now ,according to this key, load balancer will allot the client to particular server.
This can be useful to connect the client to the same server after disconnection. Like storing the items to the cart.

5) Random Algorithm

As the name suggest, the request will be forwarded to random servers.
For forwarding the request, load balancer use random number generator.

6) Least Response Time Method

Server is selected based on the least response time, let say server A is responding in 1 second
server B is responding in 2 seconds and server C is responding in 3 seconds. Then if the more requests come to load balancer, load balancer will forward the requests to the server which has least response time ,like server A.
We have covered the Load balancing algorithms in detail. There are more load balancer algorithms like global server load balancing ,layer 7 Content switching .

The components

The components are the same for both the ALB and the NLB. The following figure illustrates the components.

- The load balancer acts as the entry point into your system.
- The listener listens for incoming connections
- The load balancer forwards requests to a target group.
- The target group consists of one or multiple targets.
- A target might be an EC2 instance, container, or internal service.
- The health check monitors the targets.

As the components are the same, it is not a big deal to switch from an ALB to an NLB -or vice versa. Of course, you can only do so if you do not use any features of the ALB that the NLB does not provide.

The default

When in doubt, choose the ALB. There are two main reasons for that:

1. The ALB comes with a lot of built-in features. See the comparison table at the end of the article.
2. The NLB comes with a few rough edges and requires more experience and care.

So when deciding between an ALB and an NLB, choose the ALB unless one of the points we discuss next applies to your scenario.

The exceptions

If the answer to one of the following questions is “yes”, consider an NLB for your workload.

Do clients connect via UDP or non-HTTP?

The ALB only supports HTTP/1.1, HTTP/2, or gRPC. So when clients use a different protocol to connect with your application, you need to use the NLB instead. For example, all scenarios that are using UDP do require an NLB. Also, when you want to use HTTP/3, the NLB is currently your only choice.

Do you need to optimize for performance?

The ALB operates on layer 7, which means the ALB inspects the details of every incoming HTTP request. In contrast, the NLB works on layer 4. All the NLB cares about is forwarding the incoming TCP or UDP connection to a target. The NLB does not inspect an incoming HTTP request, for example.

Therefore, the NLB has much less work to do than an ALB. As a result, the NLB needs significantly less time to forward an incoming request. So when performance is crucial to your workload, you should consider using an NLB to reduce latency.

Do you expect unpredictable and huge traffic spikes?

The ALB adapts to an increase of connections and requests automatically. However, it takes minutes to do so. So, when expecting substantial traffic spikes, the ALB might not be able to scale fast enough to handle all incoming requests immediately. AWS advises informing their support team when you expect a huge traffic spike in the future to allow them to pre-warm the ALB for you. However, this approach only works when you can predict traffic spikes.

In contrast, the NLB does not need to scale the number of nodes processing incoming connections. Instead, the NLB is designed to handle unpredicted and huge traffic spikes.

So, when you expect unpredicted and substantial traffic spikes, the NLB is a better fit. Unfortunately, AWS does not define what “huge traffic spike” means. Most applications will never reach the limits of an ALB. Think of traffic spikes caused by a super-bowl ad hitting your web application or the new Playstation becoming available at your online shop as scenarios where this becomes an issue.

Do you require static IP addresses for inbound traffic?

To connect to an ALB, a client needs to resolve its DNS name. For example, one of my load balancers is reachable via `jenkins-YC728XLHQAVF-1237735722.eu-west-1.elb.amazonaws.com` resolves to `52.17.44.105` and `34.248.155.206`. The IP addresses are subject to change. For example, when the ALB needs to scale, the name would point to additional IP addresses.

So when you need static IP addresses for inbound traffic, the ALB is not an option. Luckily, that’s something the NLB comes with out-of-the-box. Typical scenarios are: a third party that insists on static IP addresses to create firewall rules, or a client that does not come with the ability to resolve hostnames.

The comparison

The following table compares the ALB and NLB in detail.

	ALB	NLB
Protocols	HTTP/1, HTTP/2, gRPC	TCP, UDP
Performance	Low Latency	Very Low Latency
Traffic Spikes	⚠ Inform AWS Support about huge traffic spikes	☑ Deals with huge and unexpected traffic spikes
Static IP Addresses	✕ No. However, you could place an NLB in front of an ALB.	☑ Yes
TLS Termination	☑ Yes	☑ Yes
Targets	EC2 Instance, IP Address, Lambda	EC2 Instance, IP Address, ALB
Client IP preservation	Use HTTP header `X-Forwarded-For`	Optional, but comes with limitations
Routing Algorithm	Round Robin or Least Outstanding Requests	Random
Deregistering targets	ALB stops sending requests and waits for open requests	NLB stops opening new connections, but the application needs to terminate connections properly
Multiplexing	☑ Yes, reuses connections to targets	✕ No, does not reuse connections to targets
Maximum number of targets	1000-5000	500-1000
Security Group	Security group of ALB controls inbound traffic, targets reachable from ALB only	Security group of targets control inbound traffic, targets reachable from clients
Request based routing	☑ Yes, based on hostname, path, header, …	✕ No
WAF	☑ Yes	✕ No
Authentication	☑ Yes (OpenID Connect, SAML, …)	✕ No
Slow Start Mode	☑ Yes	✕ No
Sticky Session	☑ Yes	✕ No
IPv6	☑ Yes	☑ Yes
Costs	💰💰💰	💰💰 (But causes more connections and therefore higher load on targets.)

The mental model

When it comes to AWS, it helps to have a mental model of the provided building blocks.

Think about the ALB as a reverse proxy distributing incoming requests among a fleet of virtual machines or containers. Besides that, the ALB provides features like request-based routing, authentication, and security. The ALB is like a fully-managed, scalable, and highly available version of NGINX, HAProxy, or Caddy.

In contrast, think about the NLB as a way to route traffic to a fleet of virtual machines or containers on the network layer. You get static anycast IP addresses pointing to a dynamic pool of targets. Similar services are the AWS Global Accelerator and the Google Cloud Load Balancer.

Create 2 Ec2 Instances

Under first instance, I want to host a simple plan default website, like Microsoft default website for IIS..

http://LoadBalancer/index.html -> W**eb Server 1**

Http://LoadBalancer/Data/google.jpeg - > Web Server 2

On second instance, I want to host a web application where I want to refer one image using same Ec2 public under but difference server.

Ip Addresses :

Private IP Addresses : Not accessible from internet.

Public IP Addresses : Accessible from internet.

Static IP Address :

Web Server 1 :

43.204.219.71 ----
65.1.134.149

65.0.246.196 -- Static --

Web Server 2

65.0.19.11
43.204.233.115