Reading 4.4: Route Traffic with Amazon Elastic Load Balancing

WHAT'S A LOAD BALANCER?

Load balancing refers to the process of distributing tasks across a set of resources. In the case of the corporate directory application, the resources are EC2 instances that host the application, and the tasks are the different requests being sent. It's time to distribute the requests across all the servers hosting the application using a load balancer.

To do this, you first need to enable the load balancer to take all of the traffic and redirect it to the backend servers based on an algorithm. The most popular algorithm is round-robin, which sends the traffic to each server one after the other.

A typical request for the application would start from the browser of the client. It's sent to a load balancer. Then, it's sent to one of the EC2 instances that hosts the application. The return traffic would go back through the load balancer and back to the client browser. Thus, the load balancer is directly in the path of the traffic.

Although it is possible to install your own software load balancing solution on EC2 instances, AWS provides a service for that called Elastic Load Balancing (ELB).

FEATURES OF ELB

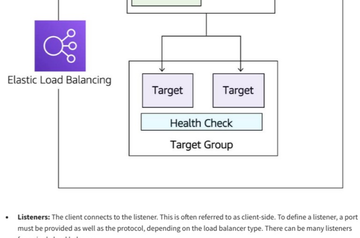
The ELB service provides a major advantage over using your own solution to do load balancing, in that you don't need to manage or operate it. It can distribute incoming application traffic across EC2 instances as well as containers, IP addresses, and AWS Lambda functions.

- The fact that ELB can load balance to IP addresses means that it can work in a hybrid mode as well, where it also load balances to on-premises servers.
- ELB is highly available. The only option you have to ensure is that the load balancer is deployed across multiple Availability Zones.
- In terms of scalability, ELB automatically scales to meet the demand of the incoming traffic. It handles the incoming traffic and sends it to your backend application.

HEALTH CHECKS

Taking the time to define an appropriate health check is critical. Only verifying that the port of an application is open doesn't mean that the application is working. It also doesn't mean that simply making a call to the home page of an application is the right way either.

For example, the employee directory application depends on a database, and S3. The health check should validate all of those elements. One way to do that would be to create a monitoring webpage like "/monitor" that will make a call to the database to ensure it can connect and get data, and make a call to S3. Then, you point the health check on the load balancer to the "/monitor" page.



After determining the availability of a new EC2 instance, the load balancer starts sending traffic to it. If ELB determines that an EC2 instance is no longer working, it stops sending traffic to it and lets EC2 Auto Scaling know. EC2 Auto Scaling's responsibility is to remove it from the group and replace it with a new EC2 instance. Traffic only sends to the new instance if it passes the health check.

In the case of a scale down action that EC2 Auto Scaling needs to take due to a scaling policy, it lets ELB know that EC2 instances will be terminated. ELB can prevent EC2 Auto Scaling from terminating the EC2 instance until all connections to that instance end, while preventing any new connections. That feature is called **connection draining**.

ELB COMPONENTS

The ELB service is made up of three main components.

- **Listeners:** The client connects to the listener. This is often referred to as client-side. To define a listener, a port must be provided as well as the protocol, depending on the load balancer type. There can be many listeners for a single load balancer.
- **Target groups:** The backend servers, or server-side, is defined in one or more target groups. This is where you define the type of backend you want to direct traffic to, such as EC2 instances, AWS Lambda functions, or IP addresses. Also, a health check needs to be defined for each target group.
- **Rules:** To associate a target group to a listener, a rule must be used. Rules are made up of a condition that can be the source IP address of the client and a condition to decide which target group to send the traffic to.

APPLICATION LOAD BALANCER

Here are some primary features of Application Load Balancer (ALB).

ALB routes traffic based on request data. It makes routing decisions based on the HTTP protocol like the URL path (/upload) and host, HTTP headers and method, as well as the source IP address of the client. This enables granular routing to the target groups.

Send responses directly to the client. ALB has the ability to reply directly to the client with a fixed response like a custom HTML page. It also has the ability to send a redirect to the client which is useful when you need to redirect to a specific website or to redirect the request from HTTP to HTTPS, removing that work from your backend servers.

ALB supports TLS offloading. Speaking of HTTPS and saving work from backend servers, ALB understands HTTPS traffic. To be able to pass HTTPS traffic through ALB, an SSL certificate is provided by either importing a certificate via Identity and Access Management (IAM) or AWS Certificate Manager (ACM) services, or by creating one for free using ACM. This ensures the traffic between the client and ALB is encrypted.

Authenticate users. On the topic of security, ALB has the ability to authenticate the users before they are allowed to pass through the load balancer. ALB uses the OpenID Connect protocol and integrates with other AWS services to support more protocols like SAML, LDAP, Microsoft AD, and more.

Secure traffic. To prevent traffic from reaching the load balancer, you configure a security group to specify the supported IP address ranges.

ALB uses the round-robin routing algorithm. ALB ensures each server receives the same number of requests in general. This type of routing works for most applications.

ALB uses the least outstanding request routing algorithm. If the requests to the backend vary in complexity where one request may need a lot more CPU time than another, then the least outstanding request algorithm is more appropriate. It's also the right routing algorithm to use if the targets vary in processing capabilities. An outstanding request is when a request is sent to the backend server and a response hasn't been received yet.

For example, if the EC2 instances in a target group aren't the same size, one server's CPU utilization will be higher than the other if the same number of requests are sent to each server using the round-robin routing algorithm. That same server will have more outstanding requests as well. Using the least outstanding request routing algorithm would ensure an equal usage across targets.

ALB has sticky sessions. In the case where requests need to be sent to the same backend server because the application is stateful, then use the sticky session feature. This feature uses an HTTP cookie to remember across connections which server to send the traffic to.Finally, ALB is specifically for HTTP and HTTPS traffic. If your application uses a different protocol, then consider the Network Load Balancer (NLB).

NETWORK LOAD BALANCER

Here are some primary features of Network Load Balancer (NLB).**Network Load Balancer supports TCP, UDP, and TLS protocols.** HTTPS uses TCP and TLS as protocol. However, NLB operates at the connection layer, so it doesn't understand what a HTTPS request is. That means all features discussed above that are required to understand the HTTP and HTTPS protocol, like routing rules based on that protocol, authentication, and least outstanding request routing algorithm, are not available with NLB.

NLB uses a flow hash routing algorithm. The algorithm is based on:

- The protocol
- The source IP address and source port
- The destination IP address and destination port
- The TCP sequence number

If all of these parameters are the same, then the packets are sent to the exact same target. If any of them are different in the next packets, then the request may be sent to a different target.

NLB has sticky sessions. Different from ALB, these sessions are based on the source IP address of the client instead of a cookie.

NLB supports TLS offloading. NLB understands the TLS protocol. It can also offload TLS from the backend servers similar to how ALB works.

NLB handles millions of requests per second. While ALB can also support this number of requests, it needs to scale to reach that number. This takes time. NLB can instantly handle this amount of requests.

NLB supports static and elastic IP addresses. There are some situations where the application client needs to send requests directly to the load balancer IP address instead of using DNS. For example, this is useful if your application can't use DNS or if the connecting clients require firewall rules based on IP addresses. In this case, NLB is the right type of load balancer to use.

NLP preserves source IP address. NLB preserves the source IP address of the client when sending the traffic to the backend. With ALB, if you look at the source IP address of the requests, you will find the IP address of the load balancer. While with NLB, you would see the real IP address of the client, which is required by the backend application in some cases.

SELECT BETWEEN ELB TYPES

Selecting between the ELB service types is done by determining which feature is required for your application. Below you can find a list of the major features that you learned in this unit and the previous.

Feature	Application Load Balancer	Network Load Balancer
Protocols	HTTP, HTTPS	TCP, UDP, TLS
Connection draining (deregistration delay)	✔	
IP addresses as targets	✔	✔
Static IP and Elastic IP address		✔
Preserve Source IP address		✔
Routing based on Source IP address, path, host, HTTP headers, HTTP method, and query string	✔	
Redirects	✔	
Fixed response	✔	
User authentication	✔	

Resources:

- *External Site:* AWS: Elastic Load Balancer product comparison
- *External Site:* AWS: AWS Certificate Manager
- *External Site:* AWS: Authenticate users using an Application Load Balancer
- *External Site:* AWS: How AWS WAF works

Mark as completed