Networking & Content Delivery: ELB

ELB service and its varations will be discussed in this lesson, followed by our recommendations on the use of ELB service of AWS.

We'll cover the following

- Classic
- ALB
- NLB
- Support for TLS/HTTPS
- Amazon as a man-in-the-middle
- NLB vs ALB
  - TCP passthrough
  - Single vs Multi-tenant system
  - Cost effective
- Our Recommendations

ELB is a load balancer service and comes in three variants:

- **Classic**
- **Application (ALB)**
- **Network (NLB)**

Amazon **ELB**

Classic#

Classic#

Classic is a legacy option and remains there only because it works with very old AWS accounts, where you can still run EC2 instances outside of a VPC. For any new setup, you should choose one of the other two variants.

ALB#

ALBs are proper reverse proxies that sit between the internet and your application. Every request to your application gets handled by the load balancer first. The load balancer then makes another request to your application and finally forwards the response from your application to the caller. ALBs have lots of features, and they support sophisticated routing rules, redirects, responses from Lambda functions, authentication, sticky sessions, and many other things.

NLB#

On the other hand, NLBs behave like load balancers, but they work by routing network packets rather than by proxying HTTP requests. An NLB is more like a very sophisticated network router. When a client connects to a server through an NLB, the server would see the client as if it were connected to the client directly.

Support for TLS/HTTPS#

Both ALBs and NLBs support TLS/HTTPS, and they integrate very well with the AWS Certificate Manager. This lets you set up TLS certificates and forget about them. The certificates get renewed automatically and deployed to your load balancers without any downtime. And all the certificates are free.

To have end-to-end TLS from the caller to your application, you will also have to enable TLS on your application. Otherwise, the traffic from the load balancer to your application will travel unencrypted on part of the network. Unfortunately, certificates from the Certificate Manager cannot be exported, so you can't use them for your application. Instead, common practice is to create a self-signed certificate on your host and use that for your application. The load balancers do not validate the server's certificate (neither the name, nor the expiry), so in this case, a self-signed certificate works fine.

Amazon as a man-in-the-middle#

The fact that ALBs and NLBs don't validate certificates might seem concerning. However, since these load balancers run in a VPC, Amazon authenticates each network packet and guarantees that the packets go only to the hosts you configured in your load balancer. The protection from spoofing and man-in-the-middle is provided by Amazon.

That said, keep in mind that:

> by installing TLS certificates on your load balancers, you're letting Amazon become a man-in-the-middle itself. Amazon's hardware and software will be decrypting your network traffic and re-encrypting it when forwarding it to your application (if you enable TLS on your application).

If you'd rather not trust Amazon with this responsibility, you must use:

> an NLB with TCP passthrough (without enabling TLS on the load balancer). But in that case, you must keep a valid TLS certificate on your application host and deal with certificate renewals yourself.

NLB vs ALB#

TCP passthrough#

Only NLBs support TCP passthrough, but since NLBs work on the network layer, they also lack support for many of the features found in ALBs. So, unless you need TCP passthrough, why would you ever want to use an NLB?

Well, ALBs have two main disadvantages:

- First, their proxy approach adds a few milliseconds to each request, so they're slightly slower than NLBs.
- Second, they may not scale quickly enough to handle a big burst of traffic.

Single vs Multi-tenant system#

An ALB behaves like a single-tenant system. AWS keeps track of your request rates and then automatically scales your ALB up or down based on the demand it sees. The exact logic of this behavior is opaque, so the only way to be assured that your ALB's elasticity meets your demands is:

- To test it yourself.
- Or, if you know a certain amount of traffic is on the way, you can ask AWS (through a support ticket) to preemptively provide sufficient capacity for you.

On the other hand, NLBs behave like a multi-tenant system and are scaled up and down in aggregate, rather than per individual load balancer. Therefore, in theory, *you should never have an NLB fail to scale to your needs (unless you exhaust all of Amazon's capacity).*

Cost effective#

NLBs are also slightly less expensive than ALBs. But a single ALB can be used to handle multiple domains (via host-based routing) while an NLB cannot, so in some situations, an ALB can be more cost-effective. Nevertheless, cost is unlikely to be the deciding factor when choosing between an ALB and an NLB.

Our Recommendations#

- Our recommendation is to consider using an **NLB** first since it offers the peace of mind of not having to worry about any obscure capacity limits. The fact that it's also faster and less expensive is a nice bonus.

- An **ALB** makes a great choice too, especially if you find value in any of its unique features. For the vast majority of use cases, you shouldn't run into its elasticity limits. And even if that were to happen, it should adapt on its own without your intervention (but only after some request throttling).

Q NLBs are proper reverse proxies that sit between the internet and your application.

A) True

B) False

Submit Answer

Reset Quiz

In the next lesson, we will take a look at Route 53 and it’s usage.

Back

Compute: Lambda

Next

Networking & Content Delivery: Route...

✓ Mark as Completed

Report an Issue