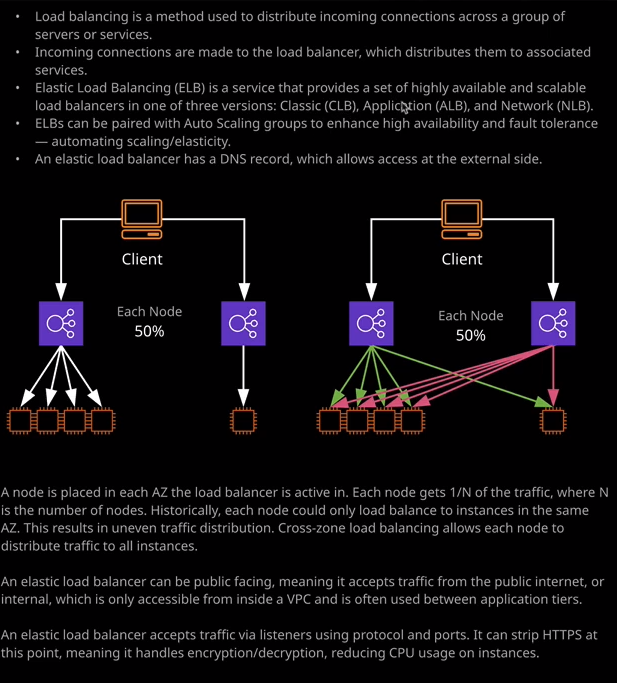
**Load balancing and Auto-scaling**

Now, in this section of the course, we're going to be talking about products and services which don't only help with pure AWS deployments, but in fact, **enhance your ability to either use hybrid environments or to implement scaling and elasticity within AWS deployments**. Now, the first topic we're going to cover **load balancing and auto scaling**.

AWS have lots of different load balancing products available and over the next few lessons, we're going to cover each of them in turn followed by talking about launch configuration and auto scaling groups. Now, before I do that, it is possible that you haven't experienced the load balancing architecture before and so in this lesson, I want to talk about **load balancing fundamentals** because understanding exactly what the load balancer is supposed to do at a high level is really important to grasping the differences between the various different load balancer products in AWS. So let's get started.



At a high level, **load balancing is a method used to distribute incoming connections across a group of servers or services.** So in this diagram, we've got our client. Our client connects to a load balancer, and the load balancer distributes those connections across backend instances. The architecture is that connections are made from the client. So from your desktop or your laptop or your mobile they're made to the load balancer. The load balancer decides, based on a number of different methods and criteria which backend instance is going to be used and then it makes a connection with that backend instance, and then load balancer brokers the connection between you and that backend instance, so you might connect to a load balancer will be directed to instance A, you might click on another link and be moved across to instance B and from your client's perspective, it's abstracted away it's completely transparent to the user of the service. Now in these matters is load balancers could be used in a lot of highly available or scalable architectures, because if you deploy an application rather than having to point an application at one or more instances specifically, you can point to the load balancer and then the load balancer might have to attached instances, or it might have 50. From a client perspective, it's completely transparent. So you're abstracted away from the underlying compute services that are providing the functionality. Now that helps both from a scaleability perspective but also from a high availability perspective because if a server fails, then the load balancer can just decide to redirect your connection to another working instance.

Now **AWS have got a number of different types of load balancers. All of these come under the family of the elastic load balancing or elastic load balancer service, and that's known as ELB. So ELB provides currently three different versions. We've got the classic load balancer, which is known as CLB. The application load balancer, which is known as ALB and the network load balancer, which is known as NLB** and I'll be talking about those is in detail over the next few lessons.

Now **elastic load balancers can also be paired and are often paired with auto scaling groups**. We haven't covered auto scaling groups yet, but essentially **they allow you to automatically scale an application for maybe using one or two instances, potentially to 10 instances, 20 instances or even more.** Now **normally you associate a load balancer with particular instances but if you integrate a load balancer and an auto scaling group, you can automatically have the load balancer associate itself with any instances inside the auto scaling group and because the auto scaling group scales in and out automatically. That also means the load balancer is associated or disassociated with those instances automatically.** So I'll talk about our integration again later in this topic.

Now, the architecture of how a load balancer actually works in AWS is a little bit deceptive. **When you create a load balancer from a user interface perspective, you're actually creating one entity. So one load balancer but what it actually does is it creates a node, so an elastic load balancer and node in any availability zone that you configure that load balancer it to be in.** **So if I configure a load balancer to be in two availability zones, then it will have a node in each of those availability zones so a total of two.** Now load balancer when you provision one and you'll see how this is done in the following lessons in this topic, **a load balancer it gets a DNS record that DNS record automatically points at each of the individual load balancer nodes. So if you have two load balancer and nodes than each node gets 50% of that incoming traffic. So based on the number of connections, a node will get half of the number of connections if there are two nodes. If there are three nodes, it'll get 33.3 recurring percent of incoming connections.** Now that's critical to understand architecturally.

The reason is that historically a load balancer node was only capable of distributing connections to any instances in that same availability zone. So let's say we have this architecture that's shown on the diagram on the right of my screen. We got five instances and we decide to deploy a load balancer into two availability zones. In one availability zone only have one instance, and in the other availability zone, we have four instances. Now it might make sense to think that the load balancer would distribute the traffic evenly across all five of these instances, with each getting 20% of the total connections but historically, that's not how it used to work. If we had two nodes and each node would receive 50% of the traffic and that traffic to that node would be distributed evenly across any working instances in that availability zone. Now, how that works with this particular example is that this single instance inside this availability zone would receive 100% of the traffic that reached this node so 100% of 50%. So it alone would get 50% of the total traffic volume. This node, which receives the other 50% well it would distribute its load across each of the four instances that it was associated with evenly. So each from these instances would receive 25% of 50%. So you'd end up with a massive disparity between the load that was placed on each of these underlying instances. Now, **historically, this is how it was and there was a piece of functionality known as cross zone load balancing. Now historically, cross zone load balancing was something that you could enable,** but by default it was disabled but now it's enabled by default and so the architecture looks more like this. So if we use an example where we still have two nodes. So the load balancer was configured for two availability zones then each node of the load balancer whilst it does still receive 50% of the overall connections based on DNS, it can distribute any of its incoming load across all of the instances equally and so, in this case, each of the load balancer nodes can distrubte the traffic to all five instances. So in this particular case, each node gets 50% and then for each node, each instance receives 20% so one in five of those connection attempts and the same happens for both of the nodes. So by default now, the distribution of traffic across all of the instances is much more even. So, essentially, that's what I wanted to cover.

Just the basic architecture, because most load balancers work in approximately the same way. **A load balancer has a DNS name that could be used to access the load balancer. This DNS name is distributed across any of the nodes for that load balancer and then the load balancer is configured to listen on a certain protocol and port combination.** So you define the load balancer for example to listen to HTTP on port 80 and then for any connections that it receives on port 80 you can have it send on those connections to the backend instances and you can also, in most cases, define the protocol and port to use for those backend instance connections. Now, why this matters is **you can configure a load balancer to accept traffic using a secure protocol such as HTTPS and then configure it to talk to the backend instances using HTTP. So you can have the load balancer handle the encryption for you, and that means you don't have to configure it on these backend instances. Now this is known as SSL offloading** and I'll be talking about this in much more detail in the next lesson but I wanted to introduce it now. So that you're aware of the concept. So with that being said, this is how load balancers work essentially it is **their job to accept connections distributed to backend instances and abstract away from these instances and that can be used to help you scale or to allow high availability**. **So tolerate any failed backend** instances. So that's what we're going to cover over the next number of lessons. We're going to start off talking about the classic load balancer then the application load balancer, then the network load balancer and then we're going to bring it all together and talk about how we can integrate it with launch configurations and launch templates and auto scaling groups. So this is going to be a really valuable topic. It's going to be something that will come up in the exam, and you will need in data production usage. So at this point, go ahead, mark this lesson as complete and when you ready, join me in the next.