**Lecture Notes for AWS Solution Architect**

**MODULE-IV**

**Elastic Load Balancing**

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, Lambda functions, and virtual appliances. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones. Elastic Load Balancing offers four types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

**Types of Load Balancer**

1. Classic Load Balance
2. Application Load Balancer
3. Network Load Balancer
4. Gateway Load Balancer

**Classic Load Balancer**

Classic Load Balancer provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and the connection level. Classic Load Balancer is intended for applications that were built within the EC2-Classic network.

**Application Load Balancer**

Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers. Application Load Balancer routes traffic to targets within Amazon VPC based on the content of the request.

**Network Load Balancer**

Network Load Balancer is best suited for load balancing of Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Transport Layer Security (TLS) traffic where extreme performance is required. Network Load Balancer routes traffic to targets within Amazon VPC and is capable of handling millions of requests per second while maintaining ultra-low latencies.

**Gateway Load Balancer**

Gateway Load Balancer makes it easy to deploy, scale, and run third-party virtual networking appliances. Providing load balancing and auto scaling for fleets of third-party appliances, Gateway Load Balancer is transparent to the source and destination of traffic. This capability makes it well suited for working with third-party appliances for security, network analytics, and other use cases.

****

**Benefits**

**Highly availability and elasticity**

Elastic Load Balancing is part of the AWS network, with native awareness of failure boundaries like AZs to keep your applications available across a region, without requiring Global Server Load Balancing (GSLB). ELB is also a fully managed service, meaning you can focus on delivering applications and not installing fleets of load balancers. Capacity is automatically added and removed based on the utilization of the underlying application servers.

**Security**

Elastic Load Balancing works with Amazon Virtual Private Cloud (VPC) to provide robust security features, including integrated certificate management, user-authentication, and SSL/TLS decryption. Together, they give you the flexibility to centrally manage TLS settings and offload CPU intensive workloads from your applications. ALB also supports integration with AWS WAF, adding a level of protection before bad actors reach the application. Further, S2N and HTTP Guardian have been developed as Open Source solutions to reduce the potential for HTTP-based attacks.

**Feature breadth**

Elastic Load Balancing offers the breadth of features needed by businesses of all sizes, while delivering them in an AWS-native experience. Elastic Load Balancing includes support for features needed in container-based workloads, including HTTP/2, gRPC, TLS offload, advanced rule-based routing, and integration with container services as an ingress controller. ALB provides customers with a native HTTP endpoint for calling Lambda functions, removing the dependency on other solutions. Further, Gateway Load Balancer creates one gateway for routing traffic through fleets of third-party appliances.

**Robust monitoring & visibility**

Elastic Load Balancing allows you to monitor the health of your applications and their performance in real time with Amazon CloudWatch metrics, logging, and request tracing. This improves visibility into the behavior of your applications, uncovering issues and identifying performance bottlenecks in your application stack. ELB helps ensure compliance with application Service Level Agreements (SLAs).

**Integration and global reach**

As a native AWS service, ELB is tightly integrated with other AWS services like EC2, ECS/EKS, Global Accelerator and operational tools such as AWS CloudFormation and AWS Billing. Across the Amazon Global Infrastructure and customer data centers with AWS Outposts and on-premises target support, ELB is available everywhere you run your AWS workloads.



**Use cases**

**Migrating to AWS**

ELB supports the load balancing capabilities critical for you to migrate to AWS. ELB is well positioned to load balance both traditional as well as cloud native applications with auto scaling capabilities that eliminate the guess work in capacity planning. ELB is easy to configure and use, which makes your migration experience simple. The managed experience of ELB means that you can focus on the most critical part of a successful migration - migrating applications - instead of configuring load balancers. ELB also integrates well with the common management tools that you are familiar with such as Terraform and Ansible.

**Modernizing applications with serverless and containers**

Organizations need to build applications faster than ever, with a combination of modular architecture patterns, serverless operational models, and agile developer processes. ELB adapts to these modern applications and their changing load without customer intervention, scaling out while still only charging for usage. Customers building serverless applications using Lambda can leverage ALB to provide a native HTTP-based endpoint, without requiring complex configurations or using an API gateway. ELB also includes support for containers and container orchestration using Kubernetes, providing load balancing between clients and applications as well as service to service communication.

**Building a hybrid cloud**

Elastic Load Balancing offers the ability to load balance across AWS and on-premises resources, using a single load balancer. You can achieve this by registering all of your resources to the same target group and associating the target group with a load balancer. Alternatively, you can use DNS-based weighted load balancing across AWS and on-premises resources across two load balancers, with one load balancer for AWS and another for on-premises resources.

**Scaling third-party virtual appliances**

When migrating to the cloud, some customers look to retain their existing appliances, and the skills and processes they have built around them. Using Gateway Load Balancer, customers deploy appliances from their preferred vendor while taking advantage of the scale and flexibility of running in the cloud.

**Internet-facing Load Balancers**

An internet-facing load balancer has a publicly resolvable DNS name, so it can route requests from clients over the internet to the EC2 instances that are registered with the load balancer. An internal load balancer is assigned to a private subnet and does not have a public IP. It cannot be accessed by a client not on the VPC (even if you create a Route53 record pointing to it). If you want clients to be able to connect to your load balancer who are not on the VPC, you need to set up an internet-facing load balancer.



The nodes of an Internet-facing load balancer have public IP addresses. The DNS name of an Internet-facing load balancer is publicly resolvable to the public IP addresses of the nodes. Therefore, Internet-facing load balancers can route requests from clients over the Internet.

**Internal Classic Load Balancers**

The nodes of an internal load balancer have only private IP addresses. The DNS name of an internal load balancer is publicly resolvable to the private IP addresses of the nodes. Therefore, internal load balancers can only route requests from clients with access to the VPC for the load balancer.



If your application has multiple tiers, for example web servers that must be connected to the internet and database servers that are only connected to the web servers, you can design an architecture that uses both internal and internet-facing load balancers. Create an internet-facing load balancer and register the web servers with it. Create an internal load balancer and register the database servers with it. The web servers receive requests from the internet-facing load balancer and send requests for the database servers to the internal load balancer. The database servers receive requests from the internal load balancer.