**Load Balancing: Interview Q&A**

## **1. When would you choose an Application Load Balancer (ALB) over a Network Load Balancer (NLB), and vice versa?**

✅ **Answer:**

The choice between an **Application Load Balancer (ALB)** and a **Network Load Balancer (NLB)** depends on the **type of traffic, performance requirements, and routing features** needed.

### **When to Choose an ALB**

* **Protocol:** HTTP / HTTPS (Layer 7)
* **Advanced Routing:** Supports host-based, path-based, and content-based routing.
* **Use Cases:**
  + Web applications requiring routing to different services based on URL or hostname.
  + Microservices architectures where requests are routed to specific services.
* **Example:** A web app with multiple endpoints like /api and /frontend where traffic needs to go to different EC2 instances or containers.

### **When to Choose an NLB**

* **Protocol:** TCP, UDP, or gRPC (Layer 4)
* **Performance:** Ultra-low latency and high throughput.
* **Use Cases:**
  + High-performance backend systems.
  + Applications requiring static IP addresses or handling millions of requests per second.
  + Real-time services like gaming, IoT, or financial transaction systems.
* **Example:** A database or microservice cluster where raw TCP traffic needs to be balanced efficiently with minimal latency.

### **Summary Table**

| **Feature** | **ALB** | **NLB** |
| --- | --- | --- |
| Layer | 7 (HTTP/HTTPS) | 4 (TCP/UDP) |
| Routing | Host-based, path-based, content-based | IP & port only |
| Performance | Moderate latency | Ultra-low latency |
| Use Case | Web apps, microservices | High-performance backend, real-time apps |

**Interview Answer:**

"I would choose an **Application Load Balancer (ALB)** when I need advanced routing capabilities, such as host-based or path-based routing, and when dealing with HTTP or HTTPS traffic. It’s ideal for web applications and microservices where requests need to be routed based on URL or hostname.

On the other hand, I would choose a **Network Load Balancer (NLB)** when I need ultra-low latency, high throughput, or support for TCP, UDP, or gRPC traffic. NLB is suitable for backend systems, real-time applications, or services that require static IPs and high performance.

For example, I’d use an ALB for a web app with multiple endpoints like /api and /frontend, while I’d use an NLB for a high-performance database cluster or a real-time gaming service."

## **2. What is a target group in the context of ALB, and how is it used for routing traffic to instances?**

**Interview Answer:**

A **target group** in the context of an **Application Load Balancer (ALB)** defines the **destination for incoming traffic**. Targets can be **EC2 instances, IP addresses, or Lambda functions**.

### **How It Works**

* The ALB uses **listener rules** to route requests to the appropriate target group.
* Routing can be based on **host headers, URL paths, or other conditions**.
* Targets in a group receive traffic, supporting **load balancing, scaling, and fault tolerance**.

### **Example Use Case**

* One target group for /api endpoints pointing to backend microservices.
* Another target group for /frontend pointing to web servers.
* ALB routes requests based on path rules, ensuring traffic goes to the **correct set of resources**.

### **Benefits**

* Organizes traffic distribution efficiently.
* Enables **auto-scaling** for different services.
* Provides **flexibility** in routing and traffic management.

### 💡 **Tip:**

* Use separate target groups for different services or endpoints to **simplify scaling and management**.
* Combine with **health checks** to ensure only healthy targets receive traffic.

## **3. Explain the concept of listeners and rules in load balancer configuration.**

A **listener** in a load balancer is a process that waits for connection requests from clients using a specific **protocol and port**, such as HTTP on port 80 or HTTPS on port 443. It essentially “listens” for incoming traffic.

**Rules** define **how the load balancer should handle and route those requests**. They use conditions like URL paths, hostnames, or HTTP headers to decide which target group should receive the request.

For example, with an **Application Load Balancer (ALB)**:

* Requests to /api can be routed to **Target Group A**, which might contain API servers.
* Requests to /app can be routed to **Target Group B**, which might contain frontend servers.

This allows fine-grained traffic distribution and better organization of backend services.

**4. What are the health checks performed by AWS load balancers, and how do they impact instance health?**

**Answer:**  
Health checks are periodic tests performed by AWS load balancers (ALB, NLB, or CLB) to verify whether registered targets (EC2 instances, IPs, or Lambda functions) are available and able to handle traffic.

* **How it works:**
  + The load balancer sends requests to a specified endpoint, such as /health or /status.
  + You can configure the protocol (HTTP, HTTPS, TCP), port, and path.
  + You can also set thresholds for success/failure counts and check intervals.
* **Impact on instance health:**
  + If a target fails the configured number of health checks, the load balancer marks it as **unhealthy**.
  + The load balancer **stops routing traffic** to unhealthy instances, preventing failed requests from reaching users.
  + Once the target passes health checks again, it is marked **healthy**, and traffic is resumed.

**Key Benefit:**  
Health checks ensure high availability and reliability by routing traffic only to healthy instances, minimizing downtime and failed requests.

**5. How can you ensure session persistence or stickiness for clients using a load balancer in AWS?**

**Answer:**  
You can ensure session persistence, also called **sticky sessions** or **session affinity**, by configuring the load balancer to route requests from the same client to the same target instance consistently.

* **For an Application Load Balancer (ALB):**
  + Use **cookies** to maintain session stickiness:
    - **Application-Controlled Cookie:** The application generates the cookie, and the ALB uses it to route requests.
    - **Load-Balancer-Generated Cookie:** The ALB generates a cookie automatically to track the client session.

This ensures that user sessions are maintained correctly, which is important for applications that store session state locally on the instance.

**6. How does AWS ensure high availability for load balancers, and what are the best practices for achieving redundancy?**

**Answer:**  
AWS ensures **high availability** for load balancers by deploying them across **multiple Availability Zones (AZs)**. This allows the load balancer to continue serving traffic even if one AZ becomes unavailable.

**Best practices for redundancy:**

1. **Enable cross-zone load balancing** – ensures traffic is evenly distributed across all targets in all AZs.
2. **Distribute targets evenly across AZs** – prevents overloading a single AZ and improves fault tolerance.
3. **Use health checks** – detect unhealthy instances and automatically stop routing traffic to them, ensuring only healthy targets receive requests.

This combination provides **fault tolerance, high availability, and consistent performance** for applications.

**7. Explain the use of cross-zone load balancing in AWS, and when would you enable or disable it?**

**Answer:**  
**Cross-zone load balancing** allows a load balancer to distribute incoming traffic evenly across all targets in **all enabled Availability Zones (AZs)**, regardless of which AZ the request originated from.

* **When to enable:**
  + For **consistent traffic distribution** across all targets.
  + To improve **resource utilization** and reduce the chance of overloading a single AZ.
* **When to disable:**
  + If you need **AZ-specific routing**, such as minimizing cross-AZ data transfer costs.
  + In cases where traffic should stay within the same AZ for compliance or latency reasons.

**Key benefit:** Ensures high availability and optimal load distribution while giving flexibility for cost or architecture considerations.

**8. What is the importance of distributing instances across multiple Availability Zones (AZs) when using load balancers in AWS?**

**Answer:**  
Distributing instances across multiple AZs ensures **high availability and fault tolerance**.

* If one AZ becomes unavailable due to a failure or maintenance, the load balancer can automatically route traffic to **healthy instances in other AZs**, minimizing downtime.
* This setup also **balances load across zones**, improving performance and resilience for the application.

**Key takeaway:** Multi-AZ deployment is essential for designing **highly available and reliable architectures** in AWS.

**9. Explain the process of configuring SSL/TLS certificates for securing traffic between clients and the load balancer.**

**Answer:**  
To secure traffic between clients and a load balancer using SSL/TLS:

1. **Obtain a certificate:**
   * Use **AWS Certificate Manager (ACM)** to provision a managed certificate, or
   * Upload a **custom SSL/TLS certificate** if needed.
2. **Attach the certificate to the load balancer:**
   * Configure the **HTTPS listener** on the load balancer and associate it with the certificate.
3. **Result:**
   * All traffic between clients and the load balancer is **encrypted**, ensuring data security and integrity.

**Key benefit:** Simplifies secure communication without managing certificates manually when using ACM.

**10. What is AWS Web Application Firewall (WAF), and how can it be integrated with a load balancer for application security?**

**Answer:**  
**AWS WAF** is a firewall service that protects web applications from common exploits, such as **SQL injection, cross-site scripting (XSS), and other malicious requests**.

* **Integration with a load balancer:**
  + AWS WAF can be **associated with an Application Load Balancer (ALB)** by attaching WAF rules to it.
  + These rules inspect incoming traffic and **allow, block, or count requests** based on defined conditions (IP addresses, headers, query strings, etc.).

**Key benefit:** Provides an additional **layer of security** for web applications without modifying application code.

**11. What are blue-green deployments, and how can AWS load balancers be used to facilitate this deployment strategy?**

**Answer:**  
**Blue-green deployments** are a deployment strategy where you maintain two identical environments:

* **Blue:** The current live environment.
* **Green:** The new version to be deployed.

**How load balancers help:**

* The **load balancer** can shift traffic from the blue environment to the green environment gradually or instantly.
* You can start by sending a small percentage of traffic to green for testing and monitoring.
* Once the green environment is verified to be stable, the load balancer routes all traffic to it.

**Key benefit:**

* Ensures **zero-downtime deployments** and provides an easy **rollback option** if issues are detected.