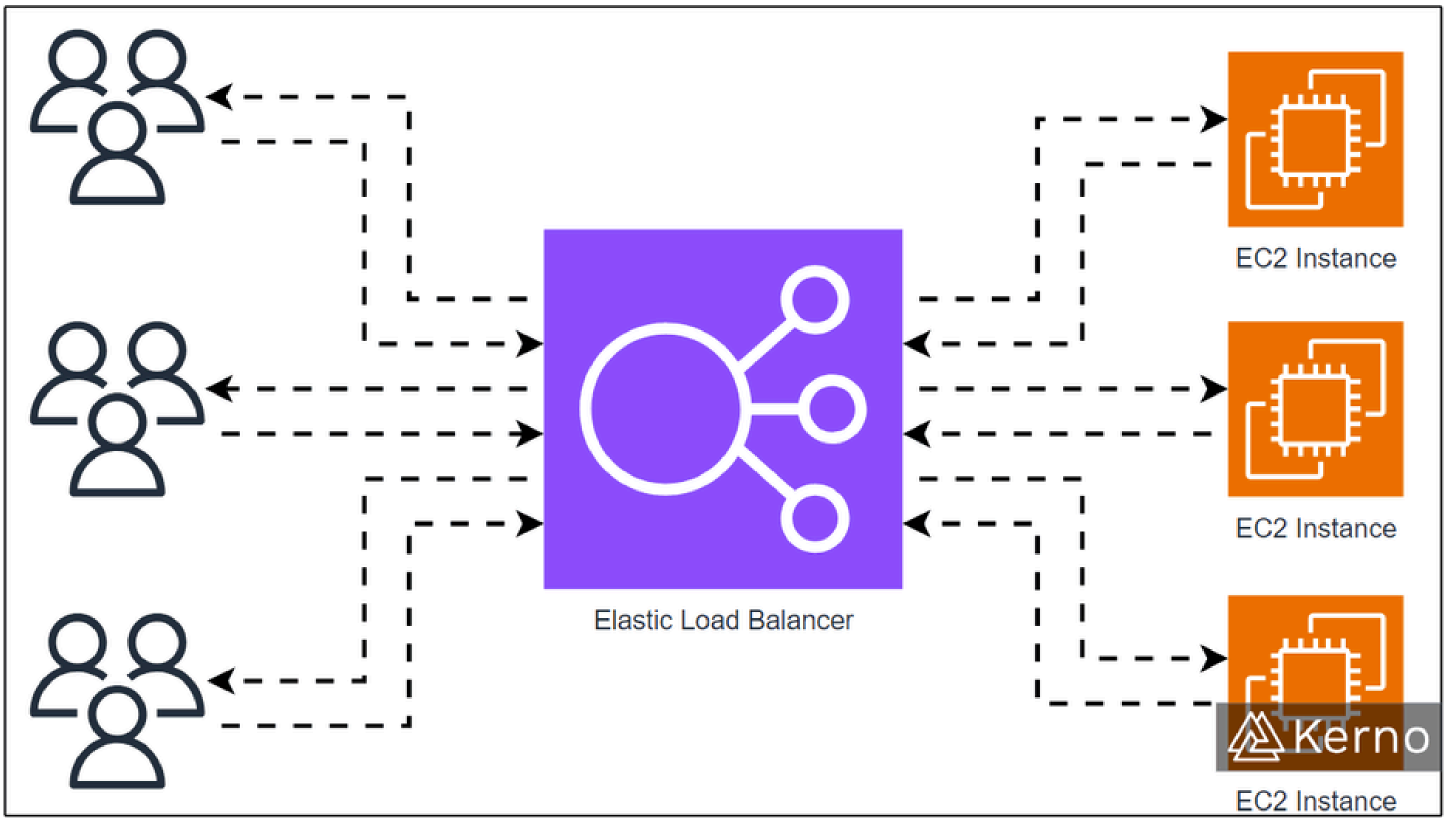
Elastic Load Balancer (ELB) in AWS

The **Elastic Load Balancer (ELB)** is a fully managed service that automatically distributes incoming application traffic across multiple targets (e.g., EC2 instances, containers, IP addresses) in one or more availability zones. It improves fault tolerance, ensures high availability, and provides scalability for your applications.



What is Elastic Load Balancer (ELB)?

- Acts as a middle layer between the clients and your backend targets (e.g., EC2 instances).
- Distributes traffic efficiently, ensuring no single resource becomes a bottleneck.
- Integrates with AWS Auto Scaling to handle varying levels of traffic.
- Supports traffic distribution across multiple availability zones, enhancing resilience.

Types of Elastic Load Balancers

AWS offers four types of Elastic Load Balancers, each optimized for specific use cases:

1. **Application Load Balancer (ALB):**

- Operates at the **application layer (Layer 7)** of the OSI model.
- Ideal for HTTP/HTTPS traffic.

- Key Features:
  - Advanced routing based on URL paths, hostnames, HTTP headers, and query strings.
  - SSL/TLS termination for secure connections.
  - WebSocket and gRPC support for real-time communication.
- **Use Cases:**
  - Hosting microservices and containerized applications.
  - Applications requiring content-based routing.

2. **Network Load Balancer (NLB):**

- Operates at the **transport layer (Layer 4)**.
- Handles TCP, UDP, and TLS traffic with high throughput and low latency.
- Key Features:
  - Supports static IP addresses and Elastic IPs.
  - Connection-based routing.
  - Scales to millions of requests per second.
- **Use Cases:**
  - Real-time applications requiring ultra-low latency (e.g., gaming).
  - Load balancing non-HTTP protocols like MQTT or DNS.

3. **Gateway Load Balancer (GWLB):**

- Operates at Layer 3 and provides load balancing for virtual appliances (e.g., firewalls, intrusion detection systems).
- Simplifies deploying and managing third-party security appliances.
- Key Features:
  - Traffic mirroring for inspection.
  - Integrates seamlessly with virtual private cloud (VPC).
- **Use Cases:**
  - Deploying security and network monitoring appliances.

4. **Classic Load Balancer (CLB):**

- Operates at both Layer 4 (TCP/SSL) and Layer 7 (HTTP/HTTPS).
- Legacy service; primarily used for applications that do not require advanced features of ALB or NLB.
- **Use Cases:** Legacy applications or simple traffic balancing setups.

Key Features of Elastic Load Balancer

1. **Automatic Scaling:**
   - Scales up or down automatically to handle varying traffic loads.
2. **Health Checks:**
   - Monitors the health of backend targets and routes traffic only to healthy ones.
3. **High Availability:**
   - Distributes traffic across multiple availability zones for fault tolerance.
4. **Security Integration:**
   - Supports SSL/TLS encryption for secure communication.

    - Integrates with AWS Web Application Firewall (WAF) for protection against web threats.
5. **Sticky Sessions:**
    - Ensures that a client session is consistently routed to the same target.
6. **Monitoring and Logging:**
    - Integrated with AWS CloudWatch for monitoring.
    - Access logs can be stored in S3 for analysis.
7. **IPv6 Support:**
    - Supports both IPv4 and IPv6 protocols.
8. **Cross-Zone Load Balancing:**
    - Ensures traffic is evenly distributed across all registered targets, regardless of their availability zone.

How Elastic Load Balancer Works

1. **Traffic Distribution:**
    - Clients send requests to the ELB.
    - ELB distributes the traffic to registered targets based on the configured load balancing algorithm.
2. **Health Monitoring:**
    - Performs periodic health checks on targets.
    - Routes traffic only to healthy targets, ensuring seamless operation.
3. **Scaling:**
    - Works with Auto Scaling to launch or terminate targets based on demand.

Load Balancing Algorithms

1. **Round Robin (default):** Routes requests sequentially to targets.
2. **Least Connections:** Routes traffic to the target with the fewest active connections (NLB).
3. **IP Hash:** Distributes traffic based on client IP addresses.

Use Cases of Elastic Load Balancer

1. **Web Applications:**
    - Distribute HTTP/HTTPS requests for websites or APIs.
2. **Microservices Architectures:**
    - Use ALB for routing requests to specific microservices based on paths or hostnames.
3. **Gaming Applications:**
    - Use NLB for ultra-low latency and high throughput.
4. **Enterprise Security:**
    - Deploy GWLB to route traffic through security appliances.

Security Features

1. **SSL Termination:**
    - Offloads SSL decryption, reducing the burden on backend targets.
2. **AWS Web Application Firewall (WAF):**
    - Protects against SQL injection, cross-site scripting, and other attacks.

3. **Integration with IAM:**
    - Manages access control using AWS Identity and Access Management (IAM).
4. **Private Load Balancing:**
    - Deploy ELBs within a private VPC for internal applications.

Best Practices for Elastic Load Balancer

1. **Enable Cross-Zone Load Balancing:**
    - Distributes traffic evenly across all targets, even in different availability zones.
2. **Use Health Checks:**
    - Ensure unhealthy instances are removed automatically.
3. **Use Secure Listeners:**
    - Configure HTTPS listeners to encrypt communication.
4. **Monitor with CloudWatch:**
    - Track metrics like request count, latency, and error rates.
5. **Use Access Logs:**
    - Enable logging to analyze traffic patterns and troubleshoot issues.

Limitations of ELB

1. **Cold Start for New Instances:**
    - Targets may take time to become healthy when added to the ELB.
2. **Costs:**
    - ELBs incur additional charges for traffic distribution and data processed.
3. **Regional Availability:**
    - ELBs operate within a region; they cannot distribute traffic globally.

Example Configuration

1. **Setup:**
    - Deploy an ALB with two availability zones (AZs).
    - Register EC2 instances as targets in an Auto Scaling group.
2. **Routing Rules:**
    - Route /api/* requests to a microservice.
    - Route /static/* requests to a content delivery backend.
3. **Monitoring:**
    - Enable CloudWatch metrics for request count, latency, and error rates.