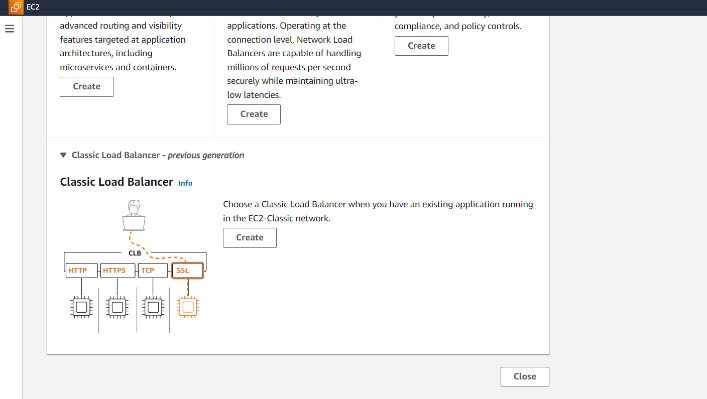
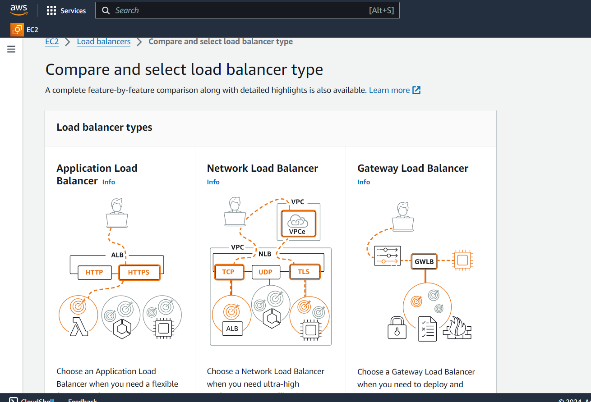
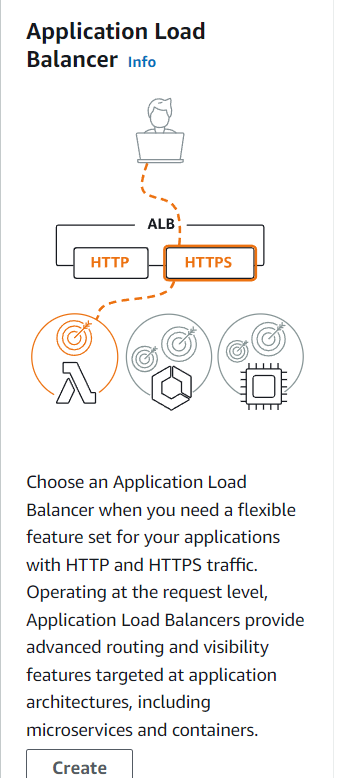
These are 4 load balancer

1. Application load balancer
2. Network load balancer
3. Gateway load balancer
4. Classic load balancer



1. **Application load balancer**



1. Clients make requests to your application.
2. The listeners in your load balancer receive requests matching the protocol and port that you configure.

3. It's a load balancer in AWS that distributes traffic among multiple targets, such as EC2 instances or containers.

4. It operates at the application layer, meaning it understands HTTP/HTTPS traffic and can route requests based on the content of the request.

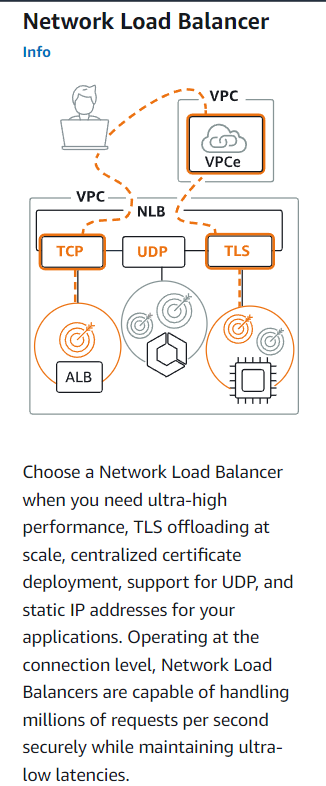
**Key Features of ALB:**

* **Sticky Sessions:** Keeps clients connected to the same target for a session, improving performance for certain applications.
* **Health Checks:** Monitors the health of targets and automatically removes unhealthy ones from the load balancing pool.
* **SSL/TLS Termination:** Can handle SSL/TLS encryption and decryption, offloading this task from your application servers.
* **Security Groups:** Provides granular control over network traffic to your load balancer and targets.
* **Target Groups:** Allows you to group targets based on their characteristics, such as instance type or port.
* **Cross-Zone Load Balancing:** Distributes traffic across multiple availability zones for high availability.

**How does it work?**

1. A client sends a request to the ALB.
2. The ALB determines the appropriate target based on the request's characteristics (e.g., path, query string, header).
3. The ALB forwards the request to the target.
4. The target processes the request and sends a response to the ALB.
5. The ALB returns the response to the client.
6. **Network load balancer**

Think of it as a traffic cop for your network applications. It distributes incoming network traffic across multiple servers (instances) in your AWS environment, ensuring that your applications can handle a high volume of traffic without crashing



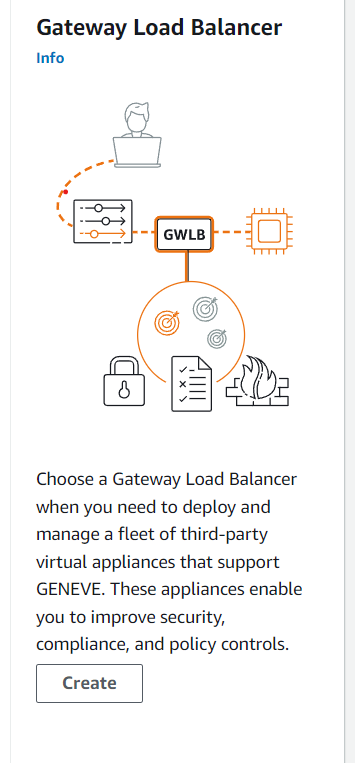
**Benefits of Using NLB:**

* Improved performance
* Increased availability
* Scalability
* Simplified management

**Key Points:**

* **Load Balancing:** NLB distributes traffic evenly among your instances, preventing any single instance from becoming overwhelmed.
* **Network Layer:** It operates at the network layer (Layer 3) of the OSI model, providing low-latency and high-throughput performance.
* **Static IP:** NLB maintains a static IP address, ensuring that your application's DNS records always point to a single, consistent IP.
* **TCP/UDP:** It can handle both TCP and UDP traffic.
* **Health Checks:** NLB continuously monitors the health of your instances using health checks. If an instance becomes unhealthy, it is removed from the load balancing pool.
* **Target Groups:** NLB uses target groups to organize your instances into logical groups. You can then route traffic to specific target groups based on your application's requirements.
* **Security Groups:** You can use security groups to control network traffic to and from your NLB.
* **Use Cases:** NLB is commonly used for applications that require high performance, low latency, and the ability to handle a large number of concurrent connections, such as:
  + Web servers
  + Game servers
  + Streaming services
  + Microservices architectures

1. **Gateway load balancer**



* A type of load balancer in AWS that distributes traffic across multiple targets based on the incoming request's destination IP address.
* Designed for applications that require high availability and scalability, especially those with a large number of connections.

**Key points**

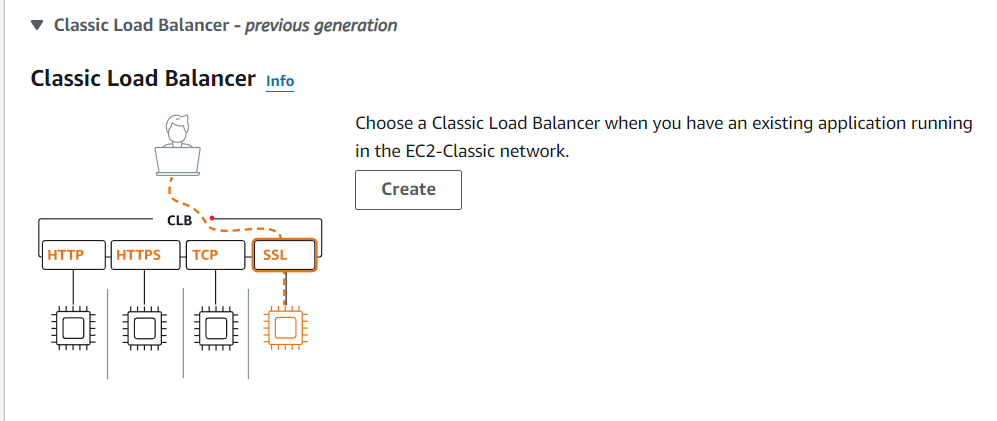
* High Availability: Ensures continuous availability of your applications by automatically routing traffic to healthy targets.
* Scalability: Handles large volumes of traffic and can easily scale up or down based on demand.
* Security: Provides built-in security features, such as SSL/TLS termination, to protect your applications and data.
* Cost-Effective: Offers a cost-effective solution for applications that require high availability and scalability.

**Use Cases:**

* **Web Applications:** Distributes traffic across multiple web servers to improve performance and reliability.
* **API Gateways:** Acts as a single entry point for your APIs, routing requests to the appropriate backend services.
* **Microservices Architectures:** Balances traffic across multiple microservices to ensure even distribution and optimal performance.

**How Does it Work?**

1. **Incoming Request:** A client sends a request to the GLB.
2. **Destination IP:** The GLB examines the destination IP address in the request.
3. **Target Selection:** Based on the destination IP, the GLB selects a target instance from the registered targets.
4. **Traffic Routing:** The GLB forwards the request to the selected target instance.
5. **Response:** The target instance processes the request and sends a response back to the GLB.
6. **Response Delivery:** The GLB forwards the response to the client.
7. **Classic load balancer**



* A type of load balancer in AWS that distributes incoming traffic across multiple EC2 instances.
* Designed for traditional application architectures.
* Provides a single point of entry for clients.

**Traffic Distribution:**

* Incoming traffic is received by the load balancer.
* The load balancer uses an algorithm (e.g., round-robin, least connections) to distribute traffic to registered EC2 instances.

**Health Checks:**

* The load balancer continuously monitors the health of registered EC2 instances.
* If an instance becomes unhealthy, it is removed from the rotation.

**Key Features and Benefits**

* **Scalability:** Easily add or remove EC2 instances to handle increasing or decreasing load.
* **High Availability:** Ensures continuous availability by distributing traffic across multiple instances.
* **Security:** Provides security features like SSL/TLS termination and access control.
* **Easy Management:** Managed through the AWS Management Console or AWS CLI.

**Use Cases**

* Traditional web applications
* Three-tier architectures
* Applications that require high availability and scalability