[Reverse Proxy V/S Load Balancing]

1. Reverse Proxy

Definition:

A **reverse proxy** is a server that sits between clients and backend servers. It forwards client requests to the appropriate backend server and returns the server's response to the client.

Key Features:

- **Client Shielding:** Hides the identity and details of backend servers.
- **Traffic Optimization:** Caches content, compresses responses, and handles SSL termination to improve performance.
- **Security:** Acts as a barrier, blocking malicious traffic before it reaches backend servers.
- **Content Routing:** Can direct requests to specific backend servers based on the URL, headers, or cookies.

Use Cases:

- Websites needing SSL offloading or caching.
- Protecting and anonymizing backend servers.
- Simplifying requests for services running on different domains or ports.

Examples of Reverse Proxy Tools:

- Nginx
- Apache HTTP Server
- HAProxy

2. Load Balancer

Definition:

A **load balancer** distributes incoming requests across multiple servers to ensure no single server is overwhelmed, improving performance, availability, and reliability.

Key Features:

- **Traffic Distribution:** Routes requests to servers based on algorithms like round robin, least connections, or weighted distribution.
- **High Availability:** Redirects traffic away from failed servers to healthy ones.
- **Scalability:** Handles increased traffic by balancing it among added servers.
- **Session Persistence:** Ensures requests from the same client go to the same server when required.

Use Cases:

- Applications needing fault tolerance and redundancy.
- High-traffic systems requiring efficient traffic management.
- Scaling applications horizontally with multiple servers.

Examples of Load Balancer Tools:

- AWS Elastic Load Balancer (ELB)
- Nginx (also acts as a reverse proxy)
- F5 BIG-IP
- HAProxy

Which to Choose?

- **Use a Reverse Proxy** if:
  - You need SSL termination, caching, or to hide your backend servers.
  - Your application runs on a single server or a small-scale deployment.
- **Use a Load Balancer** if:
  - You are managing high traffic and need fault tolerance or scalability.
  - Your application runs on multiple servers.