HOW LOAD BALANCER

Load balancer acts as a traffic coordinator, distributing incoming requests across multiple servers in a way that ensures efficient resource utilization, high availability, and improved response times.

Here's a step-by-step breakdown of how a load balancer works:

🚀 Traffic Distribution: When a user sends a request, such as accessing a website, the request arrives at the load balancer first.

🚀 Server Pool: The load balancer is connected to a group of backend servers that host the same application or service. These servers are often referred to as a server pool or server farm.

🚀 Health Monitoring: The load balancer continuously monitors the health and performance of the servers in the server pool. This can involve checking server response times, CPU usage, memory utilization, and other metrics.

🚀 Load Balancing Algorithms: The load balancer uses various algorithms to decide which server should handle the incoming request. Common algorithms include round-robin (cycling through servers in order), least connections (sending traffic to the server with the fewest active connections), and least response time (sending traffic to the server with the fastest response).

🚀 Connection Handling: Once the load balancer selects a server, it establishes a connection between the user's request and the chosen server. This connection can be based on various factors, including current server load, server health, and the selected load balancing algorithm.

🚀 Distributing Requests: As more users send requests, the load balancer evenly distributes the traffic across the available servers. This prevents any single server from becoming overwhelmed and ensures that resources are utilized optimally.

🚀 Session Persistence: In some cases, it's important to maintain a user's session on the same server throughout their interactions with the application. Load balancers can use techniques like session cookies to ensure that a user's requests are consistently directed to the same server for a specific session.

🚀 Failover and Redundancy: If a server becomes unresponsive or fails, the load balancer detects this and automatically redirects traffic to healthy servers. This process ensures that the

application remains available even if individual servers experience issues.

🚀 Scalability and Elasticity: Load balancers enable horizontal scalability, allowing new servers to be added to the server pool as traffic increases. This elasticity helps handle sudden spikes in demand without affecting the overall performance.

🚀 Monitoring and Reporting: Load balancers provide insights into server performance, traffic distribution, and other metrics. This information helps administrators make informed decisions about resource allocation and optimization.