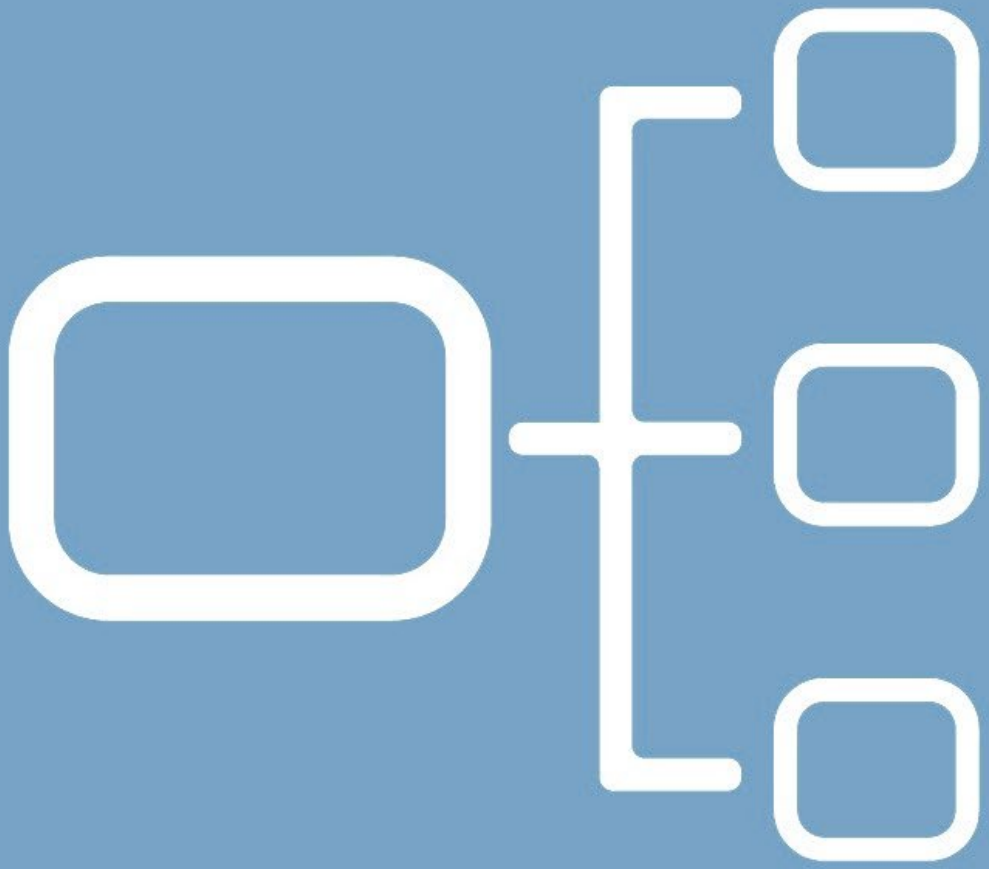


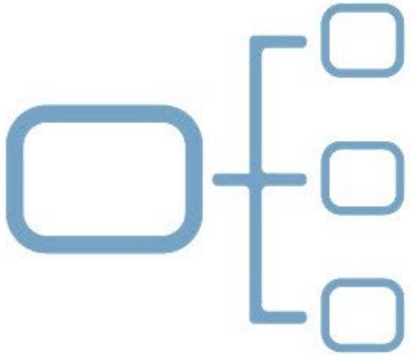
LoadBalancers

Amin Ziaei

Neshan Boot Camp

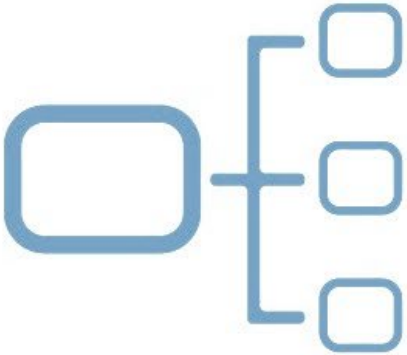
September 2023

Presentation titles

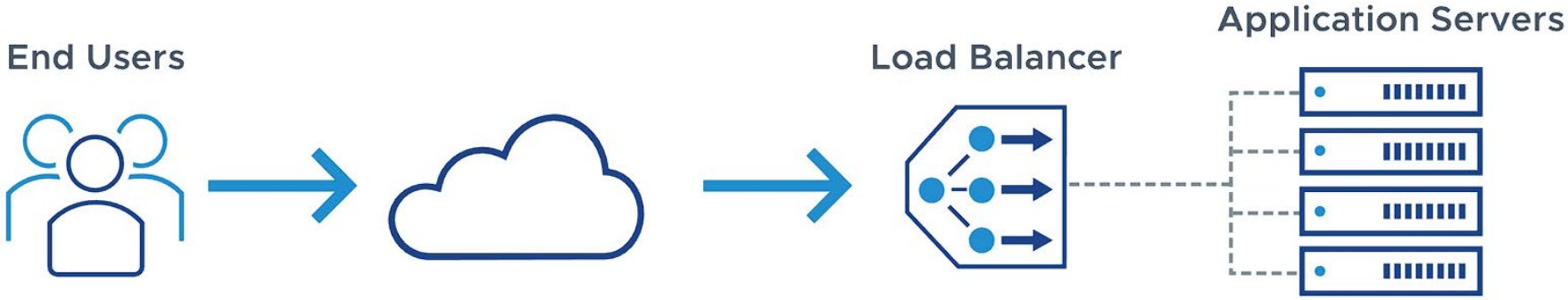


- What is a load balancer?
- Benefits of load balancing
- Types of load balancer
- General Load Balancer Types
- Load balancing algorithm
- Software Load Balancers
- Comparison and conclusion

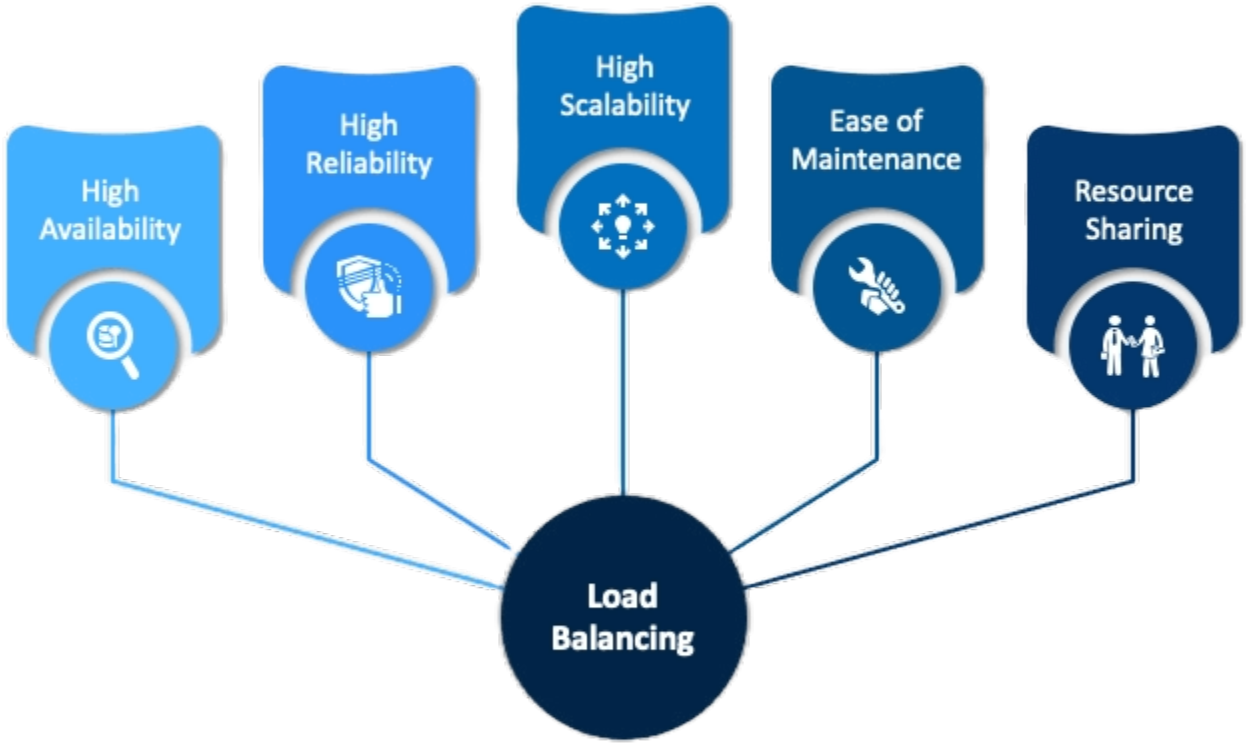
What is a load balancer?



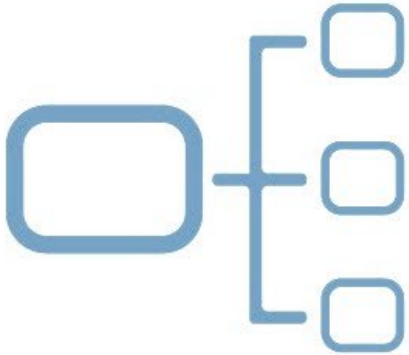
- A load balancer is a device or software that evenly distributes incoming network traffic across multiple servers or resources, such as web servers, database servers, or application servers



Benefits of load balancing



Types of load balancer



- **Hardware load balancers**

A hardware-based load balancer is a hardware appliance that can securely process and redirect gigabytes of traffic to hundreds of different servers. You can store it in your data centres and use virtualization to create multiple digital or virtual load balancers that you can centrally manage.

- **Software load balancers**

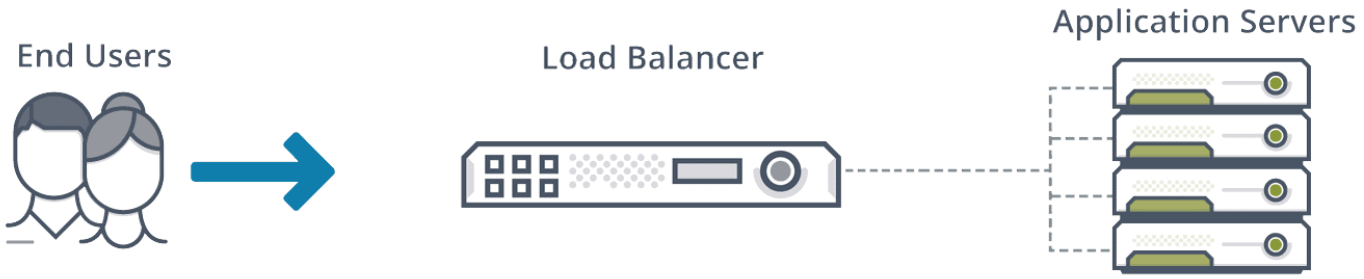
Software-based load balancers are applications that perform all load-balancing functions.You can install them on any server or access them as a fully managed third-party service.

Software-based load balancers run on standard hardware (desktop, PCs) and standard operating systems.

Types of load balancer

- **Hardware load balancers**

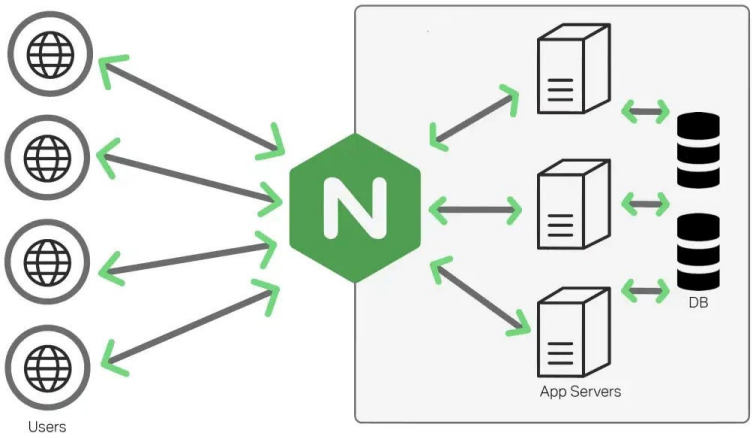
A hardware-based load balancer is a hardware appliance that can securely process and redirect gigabytes of traffic to hundreds of different servers. You can store it in your data centres and use virtualization to create multiple digital or virtual load balancers that you can centrally manage.



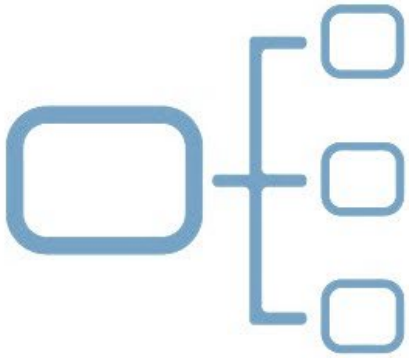
- **Software load balancers**

Software-based load balancers are applications that perform all load-balancing functions.You can install them on any server or access them as a fully managed third-party service.

Software-based load balancers run on standard hardware (desktop, PCs) and standard operating systems.



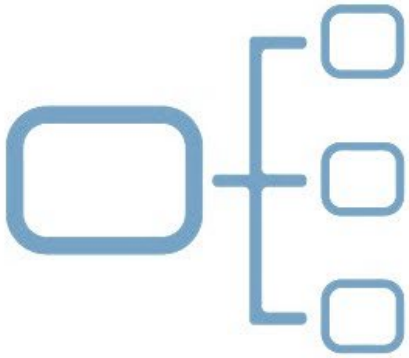
General Load Balancer Types



- **Layer 4** load balancer operates at the transport level, manages traffic based on network information such as application ports without seeing the actual content of messages.

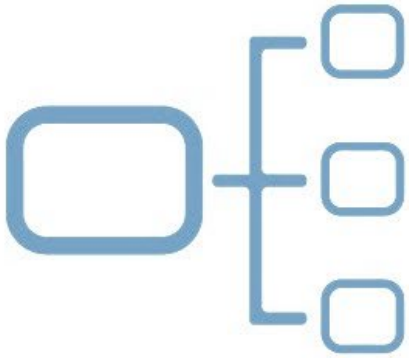
- **Layer 7** load balancer operates at the application level, using protocols such as HTTP, HTTPs and Web Socket to make decisions based on the actual content of each message. A Layer 7 load balancer terminates network traffic, performs decryption as needed, inspects messages, makes content-based routing decisions.

Comparison



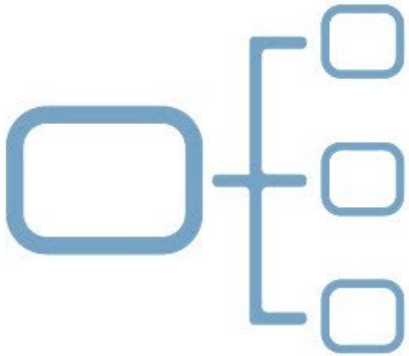
Feature	Layer 4 Load Balancer	Layer 7 Load Balancer
Layer in the OSI model	Transport layer (Layer 4)	Application layer (Layer 7)
Traffic type	TCP and UDP	HTTP, HTTPS, and other application-layer protocols
Load balancing algorithm	Round robin, weighted round robin, least connections, and others	More complex algorithms, such as session affinity, URL-based routing, and content-based routing
Application awareness	No	Yes
Capabilities	Can load balance traffic for a variety of applications	Can provide more sophisticated features, such as URL-based routing, content-based routing, and session affinity
Complexity	Simpler to configure and manage	More complex to configure and manage

Load balancing algorithm

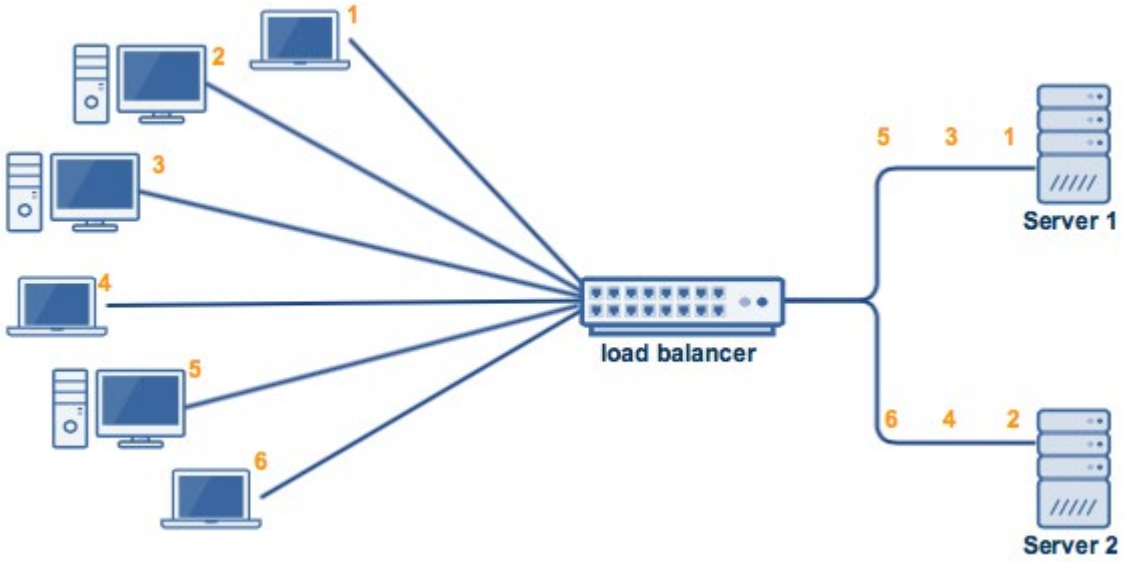


- ✓ Round Robin
- ✓ Weighted round robin
- ✓ Least Connections
- ✓ Least Time
- ✓ URL hash
- ✓ Session ID hash
- ✓ IP Hash
- ✓ Session affinity
- ✓ Content-based routing

Round Robin

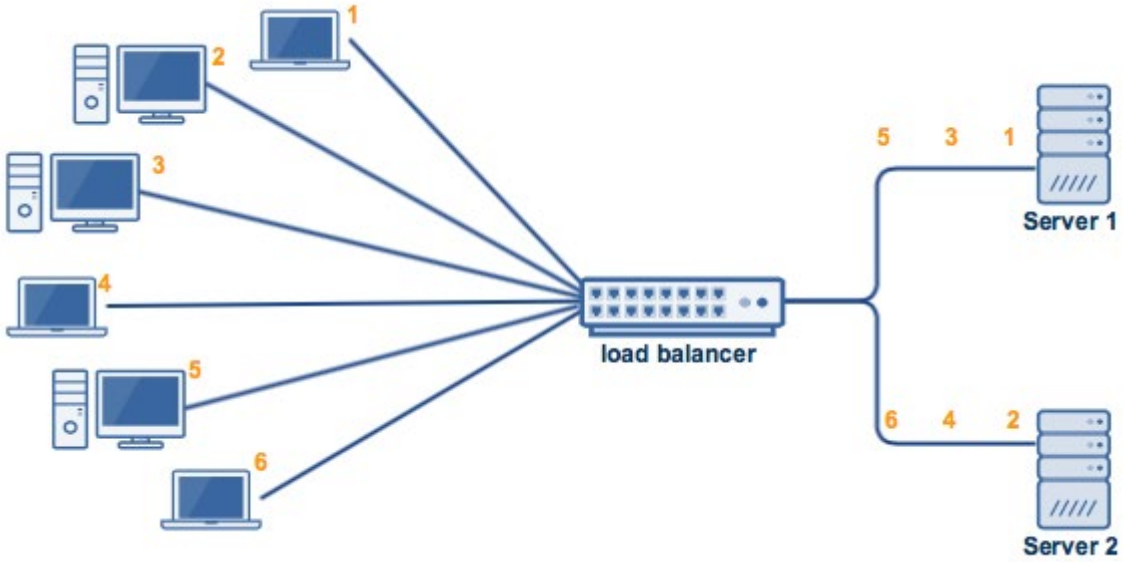


- Requests are distributed across the group of servers sequentially.



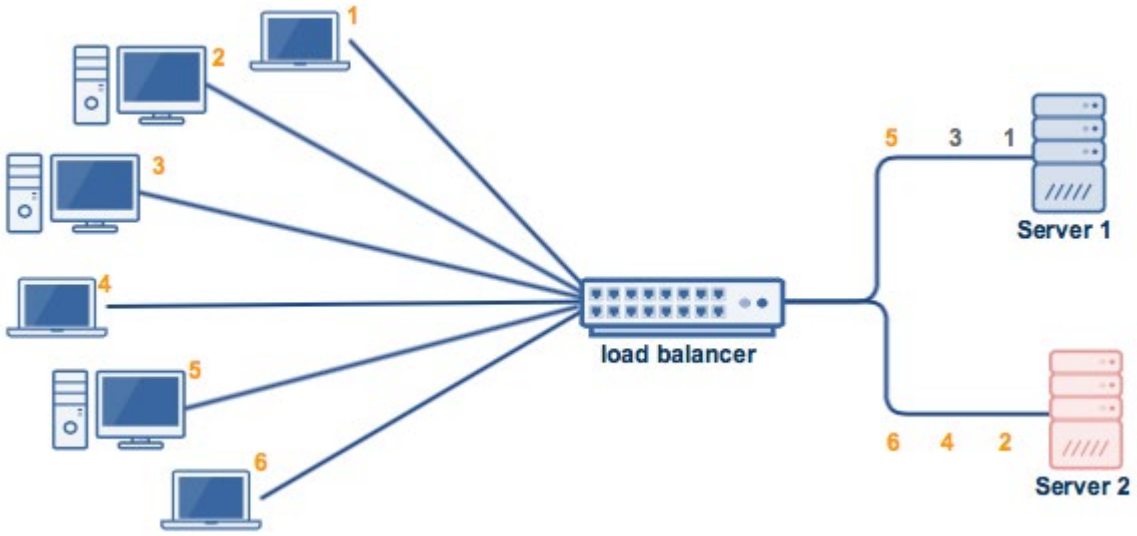
Weighted round robin

- Allows an administrator to assign different weights to each server. Servers deemed able to handle more traffic will receive slightly more



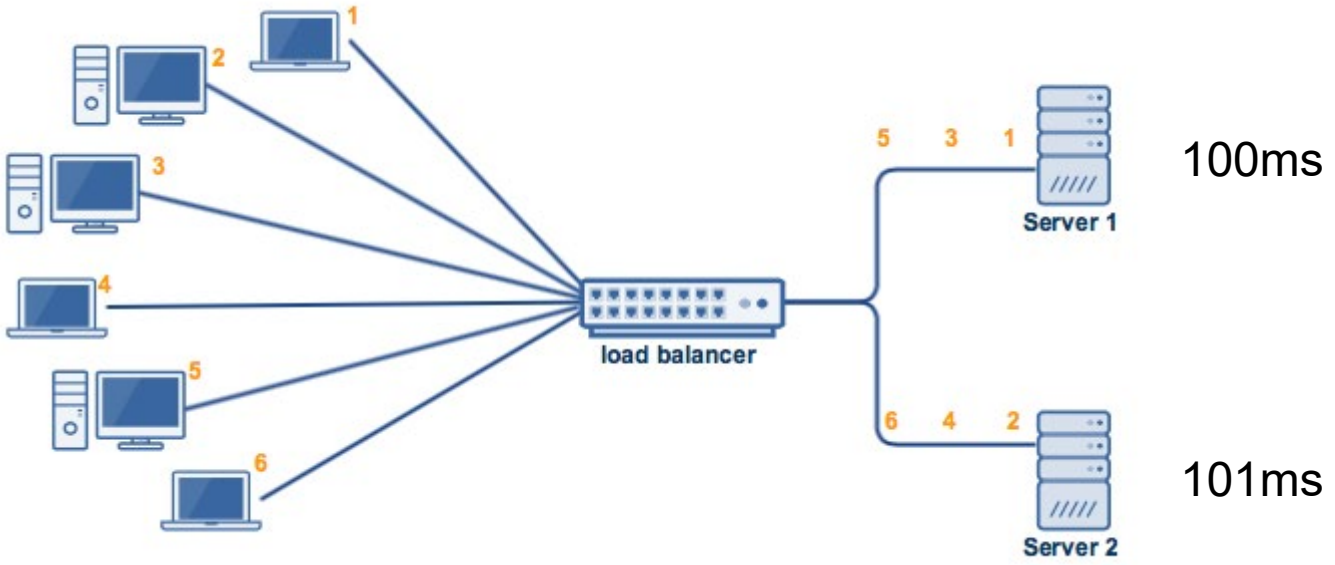
Least connection

- A new request is sent to the server with the fewest current connections to clients. The relative computing capacity of each server is factored into determining which one has the least connections.



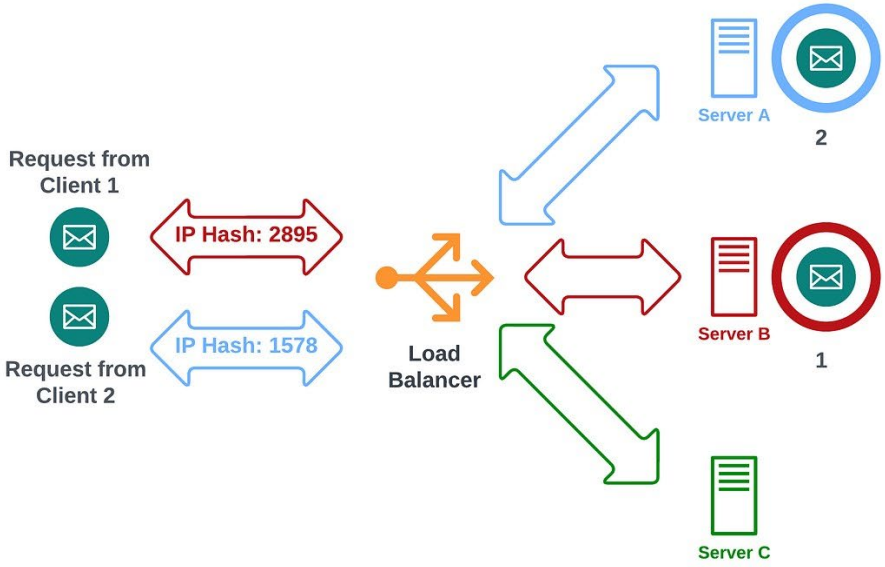
Least Time

- Sends requests to the server selected by a formula that combines the fastest response time and fewest active connections

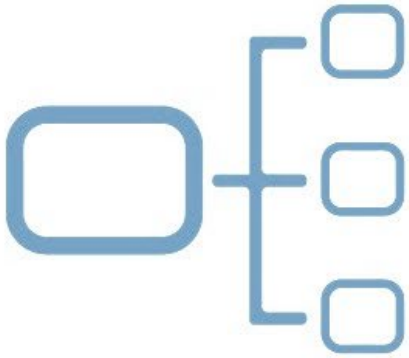


IP Hash

- This algorithm uses the source IP address of the client as the hash key. This is a good choice when you want to keep clients connected to the same server for the duration of their session.
- This is a good choice when you want to keep clients connected to the same server for the duration of their session



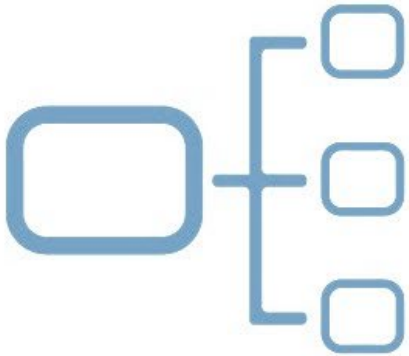
Hash



Distributes requests based on a key you define

- URL hash: This algorithm uses the URL of the requested resource as the hash key. This is a good choice when you want to distribute traffic based on the content being requested.

- Session ID hash: This algorithm uses a unique identifier for each client session as the hash key. This is a good choice when you want to keep clients connected to the same server for the duration of their session, even if they are requesting different resources.



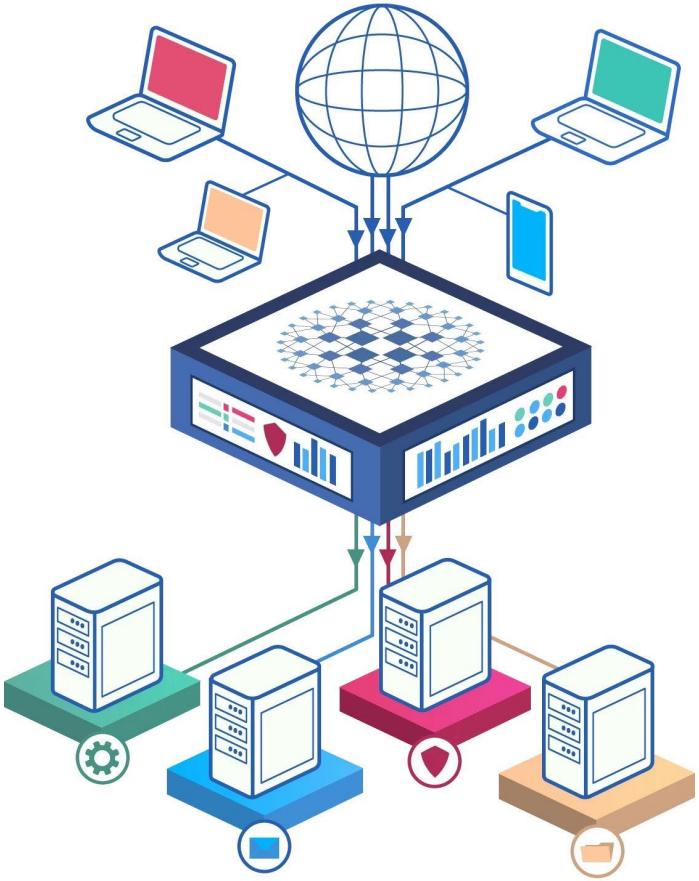
- Session affinity: This algorithm ensures that requests from the same client are always routed to the same server. This is a good choice for applications where you need to maintain state information across requests.

- Content-based routing: This algorithm routes requests based on the content of the request, such as the URL or the HTTP headers. This is a good choice for applications where you need to serve different content to different users.

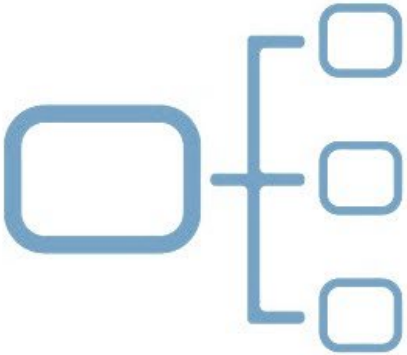
Algorithm	Description	Pros	Cons	Use Case
Round Robin	Sends requests to servers in a sequential order.	Simple to implement and understand.	Does not take into account the load or performance of the servers.	Basic load balancing, where all servers are treated equally.
Weighted Round Robin	Assigns weights to each server, so that requests are sent to the servers with the highest weights more often.	Can help to balance the load more evenly across the servers.	More complex to implement than round robin.	When there are different servers with different capacities, this can be used to ensure that the more powerful servers are handling more of the load.
Least Connections	Sends requests to the server with the fewest active connections.	Can help to improve performance by distributing the load more evenly across the servers.	Can lead to server imbalances over time, as servers with more connections will eventually become overloaded.	When the servers have different processing power, this can be used to ensure that the servers with fewer connections are not overloaded.
Least Time	Sends requests to the server that can respond the fastest.	Can help to improve performance by ensuring that requests are handled as quickly as possible.	More complex to implement than least connections.	When the servers have different response times, this can be used to ensure that requests are sent to the servers that can handle them the fastest.
URL Hash	Sends requests to the server based on the URL of the request.	Can be used to ensure that requests for the same resource are always sent to the same server.	Can lead to server imbalances over time, as some servers may receive more traffic than others.	When there are different servers that are specialized for handling different resources, this can be used to ensure that requests for those resources are always sent to the correct server.
Session ID Hash	Sends requests to the server based on the session ID of the request.	Can be used to ensure that requests for the same session are always sent to the same server.	Can lead to server imbalances over time, as some servers may receive more sessions than others.	When it is important to keep track of the state of a user's session, this can be used to ensure that all requests for that session are handled by the same server.
IP Hash	Sends requests to the server based on the IP address of the client.	Can be used to improve performance by distributing the load more evenly across the servers in a network.	Can lead to server imbalances over time, as some servers may receive more traffic from certain regions than others.	When the servers are located in different regions, this can be used to ensure that requests from users in those regions are sent to the servers that are closest to them.
Session affinity	Sends requests for the same session to the same server.	Can improve performance and user experience by keeping clients' state on the same server.	Can lead to server imbalances over time, as some servers may receive more sessions than others.	When it is important to keep track of the state of a user's session, this can be used to ensure that all requests for that session are handled by the same server.
Content-based routing	Sends requests to the server that is best able to handle the request, based on the content of the request.	Can improve performance by routing requests to the servers that are most specialized for handling them.	More complex to implement than other load balancing algorithms.	When the servers are specialized for handling different types of content, this can be used to ensure that requests for that content are always sent to the correct server.

Software Load Balancers

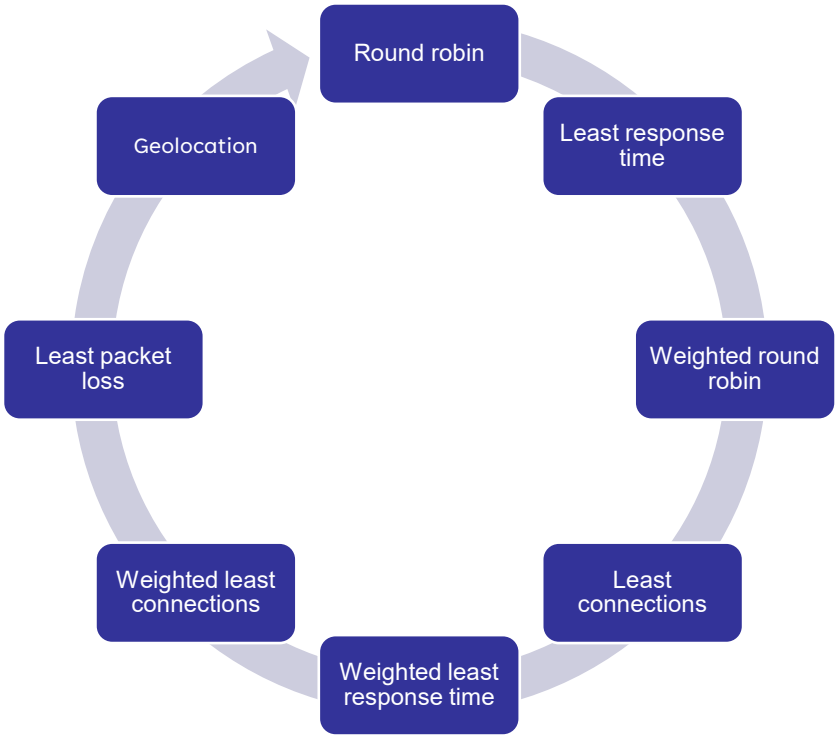
- ✓ Seesaw
- ✓ KEMP
- ✓ HAProxy
- ✓ Nginx
- ✓ Traefik



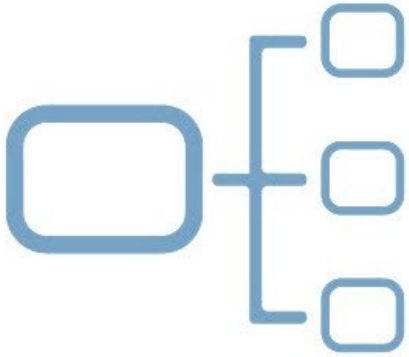
Seesaw



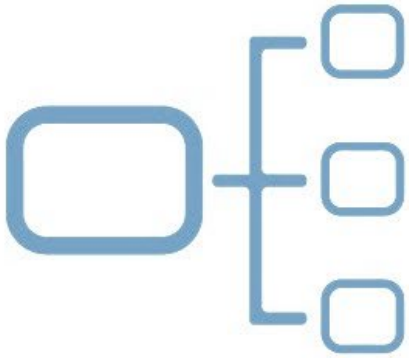
- Seesaw is an open-source load balancer developed by Google. It is based on the Linux Virtual Server (LVS) technology and can be used to distribute traffic across multiple servers.



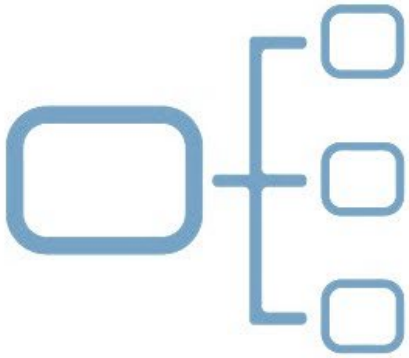
Seesaw



Feature	Pros	Cons
Open source and free to use	Can be customized to meet specific needs	Can be complex to configure and manage
Highly scalable and can handle a large number of connections	Supports a variety of load balancing algorithms	Not as widely supported as some other load balancers
Can be used to load balance both TCP and UDP traffic	Supports health checks to ensure that only healthy servers are used for load balancing	May not be as suitable for all workloads
Can be configured to work with a variety of operating systems and hardware platforms	Can be used to implement advanced features such as anycast and Direct Server Return (DSR)	



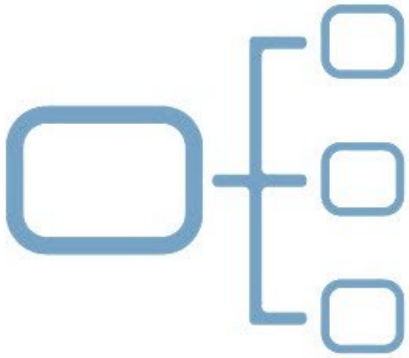
- A FREE advanced application delivery controller by KEMP is supported on all major hypervisors. You can either download and use it in your data center or deploy it in cloud DC like AWS or Azure.
- KEMP LB is used by some of the big brands like Apple, Sony, JP Morgan, Audi, Hyundai, etc. The free edition provides sufficient features; however, if you need more, you can check out their commercial license.



- ✓ Load balancing
- ✓ SSL offloading
- ✓ Content caching
- ✓ WAF
- ✓ DDoS protection

Pros:

- ✓ Kemp Load Balancer is a highly scalable and reliable load balancer.
- ✓ It offers a wide range of features, including load balancing, SSL offloading, content caching, WAF, and DDoS protection.
- ✓ It is easy to use and manage.
- ✓ It is available in a free version.

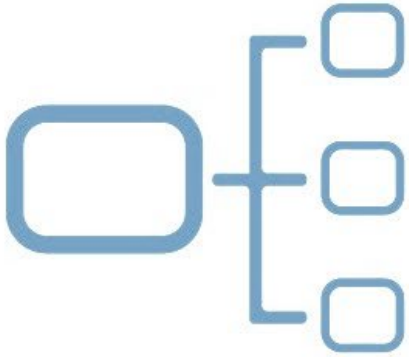


Cons:

- ✓ The free version of Kemp Load Balancer is limited in terms of the number of servers that it can load balance and the features that it offers.
- ✓ The paid versions of Kemp Load Balancer can be expensive.

Free version:

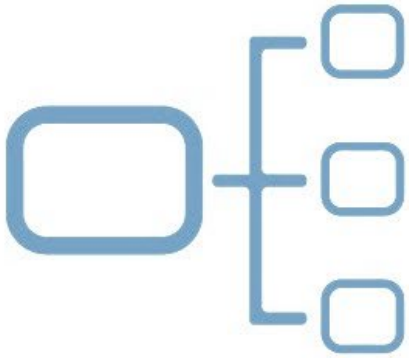
- The free version of Kemp Load Balancer is limited to load balancing up to two servers and does not offer SSL offloading, content caching, WAF, or DDoS protection. However, it is a good option for small businesses or organizations that are just getting started with load balancing.



- HAProxy (High Availability Proxy) is a free, open-source software load balancer and proxy server. It is used to improve the performance, reliability, and availability of applications by distributing traffic across multiple servers.

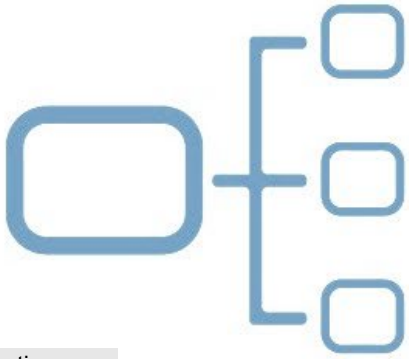
features :

- ✓ Load balancing HTTP and TCP traffic
- ✓ Reverse proxying
- ✓ Content caching
- ✓ SSL/TLS termination
- ✓ DDoS mitigation
- ✓ Application health checking

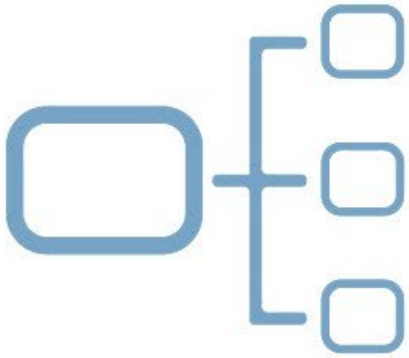


HAProxy supports a variety of load balancing algorithms:

- Round robin. This is the default load balancing algorithm in HAProxy.
- Least connections
- Weighted round robin
- Source IP hash
- Least time since last activity: This algorithm distributes traffic to the server that has been idle for the longest period of time. This can help to improve performance by avoiding servers that are overloaded.



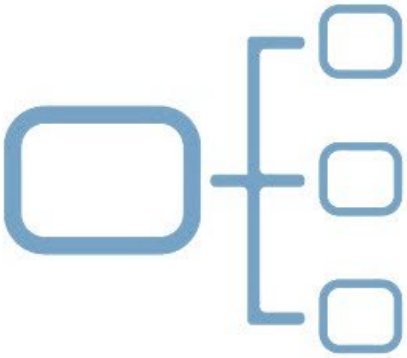
Feature	Pros	Cons	Other considerations
Performance	High performance and scalability	Can be complex to configure	Can be used to load balance HTTP and TCP traffic, reverse proxy, content caching, SSL/TLS termination, DDoS mitigation, and application health checking
Reliability	Highly reliable and stable	Can be difficult to troubleshoot	Supports a variety of load balancing algorithms, including round robin, least connections, weighted round robin, source IP hash, and least time since last activity
Flexibility	Very flexible and can be used for a variety of purposes	Documentation can be difficult to understand	Supports a wide range of features, including health checks, retries, and connection draining
Security	Includes a number of security features, such as SSL/TLS termination and DDoS mitigation	Not as secure as some other load balancers	Supports a variety of authentication methods, including basic authentication, digest authentication, and form-based authentication
Cost	Free and open-source	Requires a fair amount of expertise to configure and manage	Can be used on a wide range of platforms, including Linux, FreeBSD, macOS, and Windows



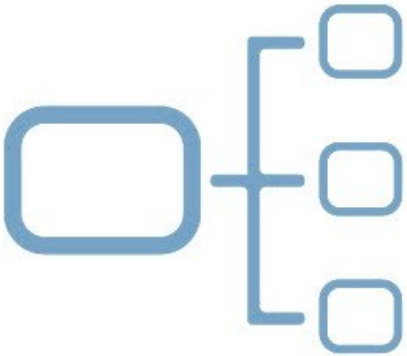
- Nginx is an open-source web server software that also functions as a reverse proxy, load balancer, mail proxy, and HTTP cache.

- Nginx's functionality can be extended with modules. There are core modules, which are compiled into the server by default, and optional modules, which can be included at compile time.

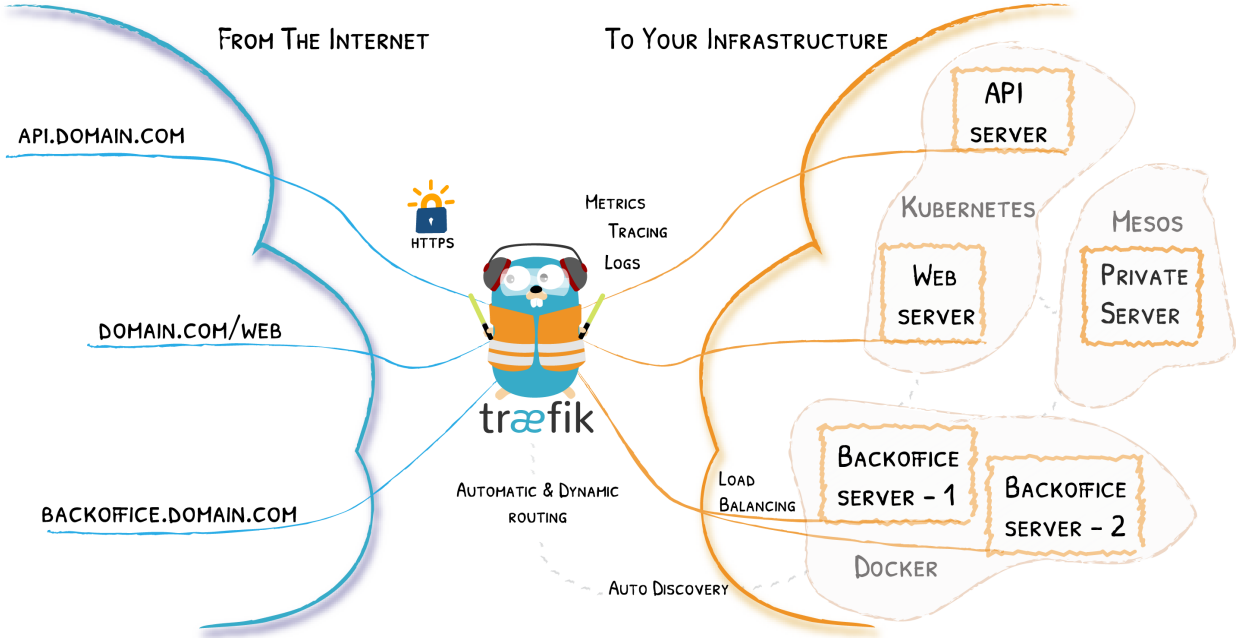
- One of the key benefits of Nginx's modular architecture is that you can choose to include only the modules you need, keeping your server lean and efficient. This is part of why Nginx is known for its high performance and low memory footprint.

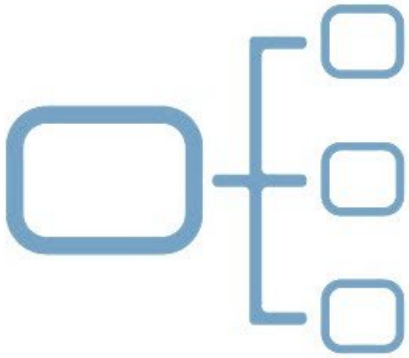


- **Reverse Proxy with Caching**: Nginx can act as a reverse proxy, where it accepts requests from clients and forwards them to other servers. It can also cache the responses from these servers to improve performance.
- **Load Balancing**: Nginx can distribute client requests across multiple backend servers using a variety of algorithms, improving the website's scalability and reliability.
- **HTTP/2(3) Support**: Nginx supports HTTP/2 and HTTP/3, a major revision of the HTTP protocol that provides improved performance.
- **SSL/TLS Support**: Nginx provides SSL/TLS support for encrypted connections, which is crucial for protecting sensitive data.
- **Modular Architecture**: Nginx has a modular architecture that allows developers to extend its functionality with modules. There are modules for things like gzip compression, SSL configuration, real-time metrics, and more.
- **HTTP Cache**: Nginx can store copies of responses to requests for a certain amount of time, improving performance by reducing the load on the server.
- **Media Streaming**: Nginx can be used for streaming media content, thanks to its ability to handle a large number of simultaneous connections.



- Traefik is an open-source edge router that makes publishing your services a fun and easy experience. It receives requests on behalf of your system and finds out which components are responsible for handling them

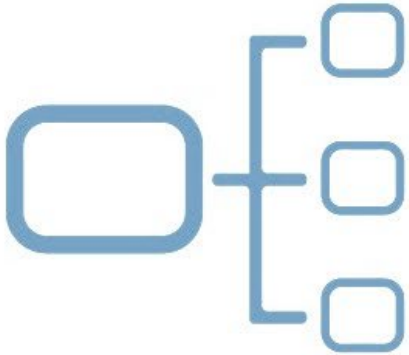




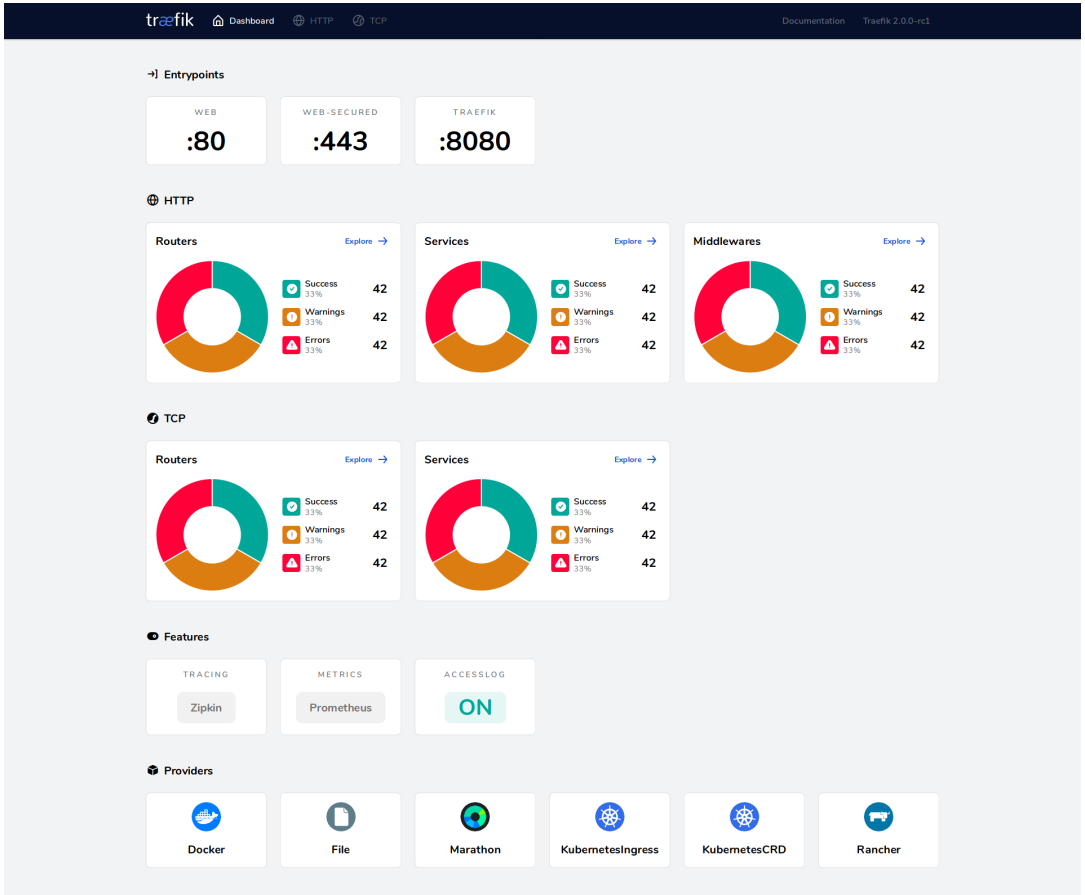
Features:

- ✓ Automatic service discovery: Traefik automatically discovers your services from your infrastructure components, such as Docker Swarm, Kubernetes, and Consul.
- ✓ Dynamic configuration: Traefik's configuration is automatically updated whenever your services change.
- ✓ Load balancing: Traefik can load balance traffic across multiple instances of your services.
- ✓ API gateway: Traefik can act as an API gateway, providing authentication, authorization, and other features.
- ✓ Middleware: Traefik comes with a powerful set of middlewares that can be used to add additional features, such as TLS encryption, rate limiting, and caching.

træfik

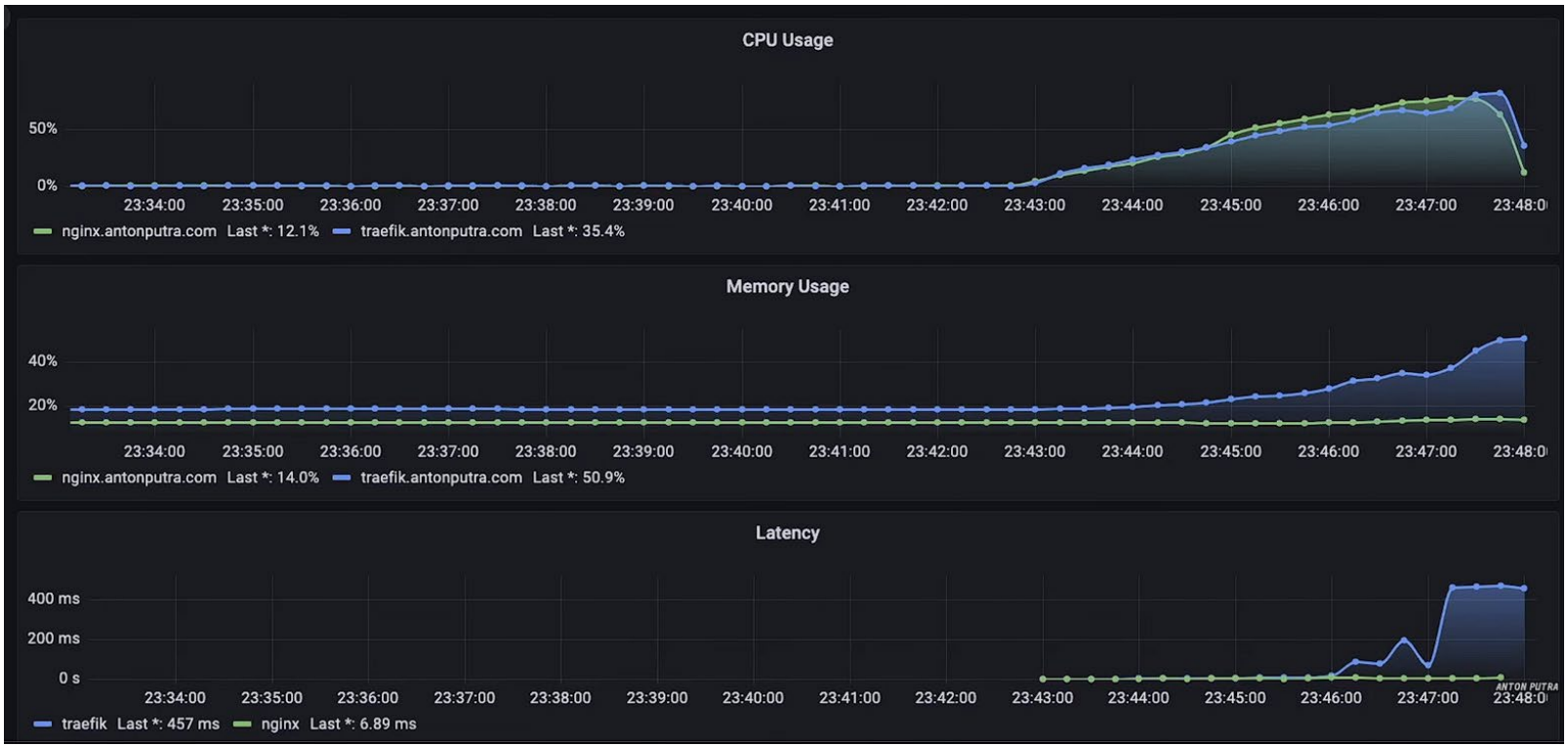


- ✓ Provides metrics (Rest, Prometheus, Datadog, Statsd, InfluxDB 2.X)
- ✓ Provides HTTPS to your microservices by leveraging Let's Encrypt (wildcard certificates support)

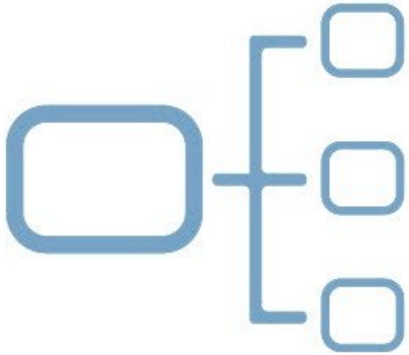


traefik

NGINX®



Comparison and conclusion



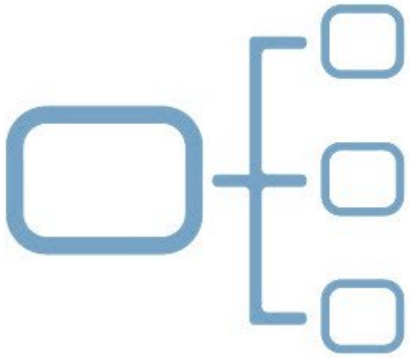
Software	Price	Pros	Cons	Key Features	Use Cases
Seesaw	Free and open-source	Easy to use, lightweight, and scalable.	Can be limited in terms of features and functionality.	Load balancing, reverse proxying, and API management.	Small and medium-sized businesses, and personal projects.
KEMP	Commercial	Feature-rich, scalable, and reliable.	Can be expensive.	Load balancing, reverse proxying, application security, and global server load balancing.	Enterprise businesses and organizations with complex load balancing and application security needs.
HAProxy	Free and open-source	High performance, scalability, and fault tolerance.	Can be difficult to configure and manage.	Load balancing, reverse proxying, and high availability.	High-traffic websites and applications, and organizations with mission-critical applications.
Nginx	Free and open-source	High performance, lightweight, and versatile.	Can be limited in terms of features and functionality.	Load balancing, reverse proxying, HTTP caching, and web application firewall.	Small and medium-sized businesses, and high-traffic websites and applications.
Traefik	Free and open-source	Easy to use, lightweight, and scalable.	Can be limited in terms of features and functionality.	Load balancing, reverse proxying, service discovery, and API management.	Microservices-based applications, and organizations with complex load balancing and service discovery needs

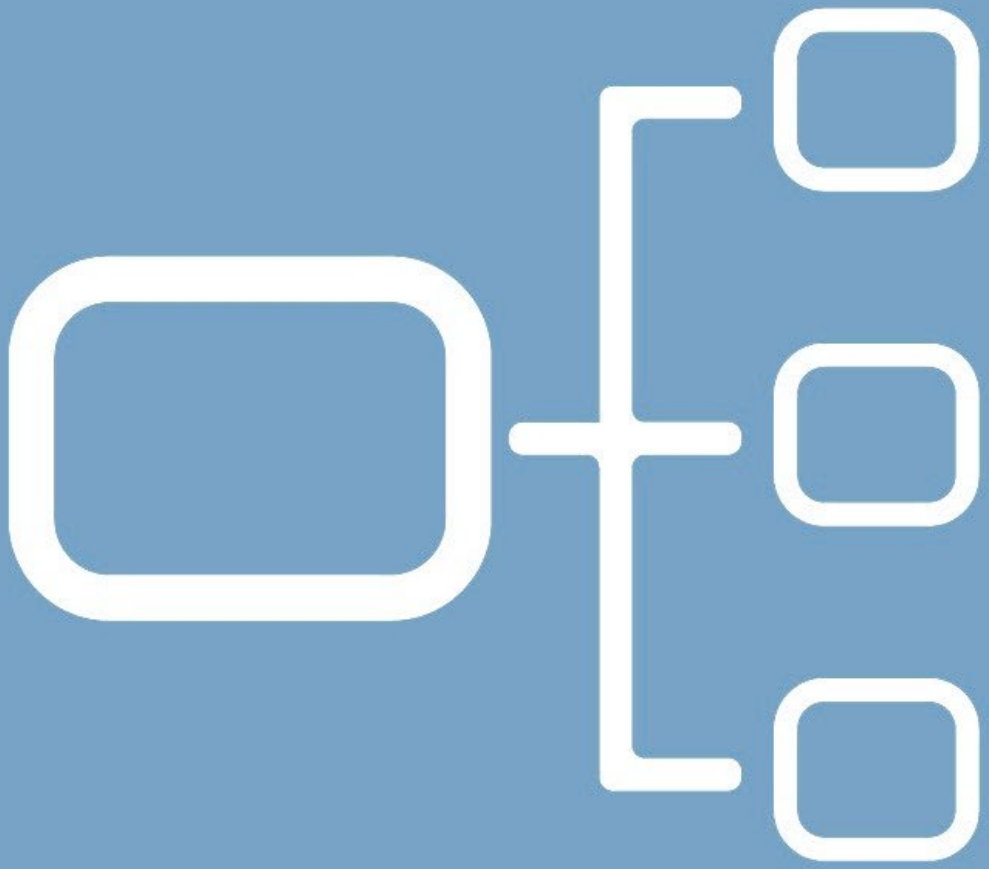
```
upstream backend {
    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com;
}

server {
    listen 443 ssl;
    server_name example.com;

    ssl_certificate /path/to/ssl_certificate.crt;
    ssl_certificate_key /path/to/ssl_certificate.key;

    location / {
        proxy_pass http://backend;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-User-Agent $http_user_agent;
}
}
```





Thanks