Expertise
and insight
for the future

Jukka Kumpunen

High availability and load balancing for web services

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

16 April 2019

metropolia.fi/en

Metropolia
University of Applied Sciences

Author Title Number of Pages Date	Jukka Kumpunen High availability and load balancing for web services 29 pages 16 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Mobile Solutions
Instructor	Jarkko Vuori, Principal Lecturer

The goal of this thesis project was to get familiar with high availability and load balancing for servers and websites. This was done by studying the benefits that high availability and load balancing offer for companies, and why using them is important especially for large company websites. The purpose of the project was to provide information for companies whose service unavailability can cause financial loss or affect customer satisfaction. With the help of the methods covered in this thesis, these types of situations can be avoided.

The methods companies use to implement high availability and load balancing vary from company to company. There are many viable solutions, such as software- and hardware-based load balancing and load balancing using cloud services or the domain name system (DNS). The price of high availability and technical difficulty of administration depend on the chosen method and the number of users a service has.

Companies must target a certain level of error tolerance that depends on the services they offer. Critical services whose unavailability quickly affects customers and the revenue of a company must be implemented in a way that can tolerate server issues and significant, unexpected changes in popularity. Methods suitable for this include software-based high-availability and cloud-based load-balancing solutions. The results of this project can be used by companies that are planning the server infrastructure and administration of their upcoming online services.

Keywords	high availability, load balancing, DNS

Tekijä Otsikko Sivumäärä Aika	Jukka Kumpunen Verkkopalveluiden korkea käytettävyys ja kuormantasaus 29 sivua 16.4.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Mobiilisovellukset
Ohjaaja	Yliopettaja Jarkko Vuori

Insinöörityössä perehdyttiin palvelinten ja verkkosivujen korkeaan käytettävyyteen ja kuormantasaukseen. Työssä selvitettiin hyödyt, joita korkea käytettävyys ja kuormantasaus tarjoaa yrityksille, ja se, miksi niiden käyttäminen on tärkeää erityisesti isoille yrityssivustoille ja palveluille. Työn tarkoituksena oli tuottaa tietoa yrityksille, joiden verkkopalveluiden häiriöajat aiheuttavat taloudellisia kustannuksia tai vaikuttavat asiakastyytyväisyyteen. Työssä käsiteltyjen menetelmien avulla tällaisia häiriöitä voidaan välttää.

Yritysten tavat toteuttaa korkea käytettävyys ja kuormantasaus vaihtelevat. Vaihtoehtoja on monia, kuten ohjelmisto- ja laitteistotason kuormantasaus tai kuormantasaus pilvipalveluja tai nimipalvelimia käyttäen. Korkean käytettävyyden hinta ja ylläpidon tekninen haasteellisuus riippuu valitusta menetelmästä ja palvelun käyttäjämäärästä.

Yritysten tulee tavoitella tiettyä vikasietoisuustasoa, joka riippuu sen tarjoamista palveluista. Kriittiset palvelut, joiden toimimattomuus vaikuttaa nopeasti asiakkaisiin ja yrityksen tuloihin, tulee toteuttaa tavalla, joka sietää palvelinten ongelmatilanteita ja suuria odottamattomia käyttäjämäärän vaihteluja. Tähän hyvin soveltuvia menetelmiä ovat ohjelmistotasolla toteutettu korkea käytettävyys ja pilvipalveluiden tarjoamat kuormantasaajat. Työn tuloksista on hyötyä yrityksille, jotka suunnittelevat tulevien verkkopalveluidensa ylläpitoa ja verkkoinfrastruktuuria.

Avainsanat	korkea käytettävyys, kuormantasaus, DNS

Metropolia
University of Applied Sciences

Contents

1 Introduction 1

2 Simple load balancing using the domain name system (DNS) 2

3 Software-based high availability 7

4 Implementing load balancing on a hardware level 10

5 Load balancing in the cloud 14

6 Choosing the right load balancing solution 21

7 Conclusion 25

References 27

List of Abbreviations

DNS	Domain Name System. A system that translates human-readable host-names into machine-readable IP addresses.
GSLB	Global Server Load Balancing. A method of load balancing network traffic across multiple data centers in different geographical locations.
GeoDNS	A Domain Name System configuration that changes its DNS responses based on the geographical location of the client that sends a DNS query.
FTP	File Transfer Protocol. A protocol used to transfer files between computers in a network.
SSH	Secure Shell. A cryptographic protocol used to secure network services over an unsecured network.
AWS	Amazon Web Services. A cloud hosting service offered by Amazon.
HLD	Hardware Load Balancer Device. A hardware device designed for the load balancing of network traffic between multiple servers.
SSL	Secure Sockets Layer. A cryptographic protocol used to secure communi-cations over a network. A predecessor of SSL.
TLS	Transport Layer Security. A cryptographic protocol used to secure commu-nications over a network. A successor of SSL.
ADC	Application Delivery Controller. A load balancer device that can handle ad-ditional tasks, such as data compression and caching.
LBaaS	Load Balancing as a Service. A service model in which a company offers a load balancer for use with existing servers.
ELB	Elastic Load Balancer. A cloud load balancer service part of Amazon Web Services.

EC2	Elastic Compute Cloud. A service for hosting virtual private servers in the Amazon Web Services cloud.
LCU	Load Capacity Unit. A unit to measure the monthly pricing and capacity of the Elastic Load Balancer service of Amazon Web Service.
DDoS	Distributed Denial of Service. A network attack that aims to make its target service unavailable for others by sending a large number of irrelevant requests.

1 Introduction

Popular websites can have millions of daily users, and high-traffic websites require high server capacity. The simplest way is to increase the capacity of a single server, such as its memory, disk space and processing power in the form of additional processor cores. This technique is also called vertical scaling. However, a single server cannot be scaled to serve a continuously increasing user base forever. When the user base of a website becomes large, it is time to consider other solutions. A common alternative to vertical scaling is implementing high availability and load balancing.

Load balancing is a type of scaling in which capacity is added by distributing load across multiple servers. This type of scaling is called horizontal scaling. High availability is a form of error tolerance in which the unavailability of a single server does not cause a service interruption. A server can be unavailable due to a hardware failure, technical maintenance, software issue or it can simply be overloaded. In a typical scenario, any of these would cause downtime during which the service hosted on the server would be inaccessible for users. Downtime and slowly loading websites can be avoided by setting up a high availability infrastructure.

The goal of this thesis project was to become familiar with ways to achieve high availability and increase the error tolerance of web services. This was done by studying some existing solutions of companies used in production environments. This thesis focuses on high availability from the perspective of servers and web services hosted on them. Since load balancing is an important part of high availability, was also explored.

The purpose of this thesis project was to offer multiple ways for companies and individuals to build a highly available infrastructure where the operation of a web service would not be fully dependent on a single server. Various factors, such as cost, complexity and the time required for maintenance, were considered when comparing possible choices for a load balancing solution. This comparison should assist companies in choosing a suitable load balancing strategy for their services. The purpose of a highly available infrastructure is to reduce downtime, which may negatively affect sales and customer satisfaction.

Metropolia
University of Applied Sciences

2 Simple load balancing using the domain name system (DNS)

One of the simplest and easiest ways to achieve high availability is to use the domain name system (DNS). DNS is a system that connects IP addresses to names that are easier for humans to remember. These names are most commonly used to access websites on the Internet. Because of DNS, a user can write a hostname such as www.example.com to the address bar of a web browser, and the browser knows how to load the correct website. It does this by contacting a name server that keeps track of hostnames and their corresponding IP addresses. The IP addresses belong to web servers that have been set up to host the website at that hostname.

A hostname is a name in DNS that points to an IP address, which is usually a single server. A domain name functions at an upper level in the hierarchy and can have multiple hostnames in DNS. For instance, the domain name example.com can have hostnames www.example.com and mail.example.com. The domain name system consists of a group of servers that have been configured to keep track of which hostnames point to which IP addresses. This system can be used to implement high availability and load balancing at a simple and basic level. DNS can distribute online traffic in a round-robin fashion. This means traffic will be distributed across multiple servers one server at a time as illustrated in Figure 1. One request will be sent to one server and the next one will be sent to a different server.

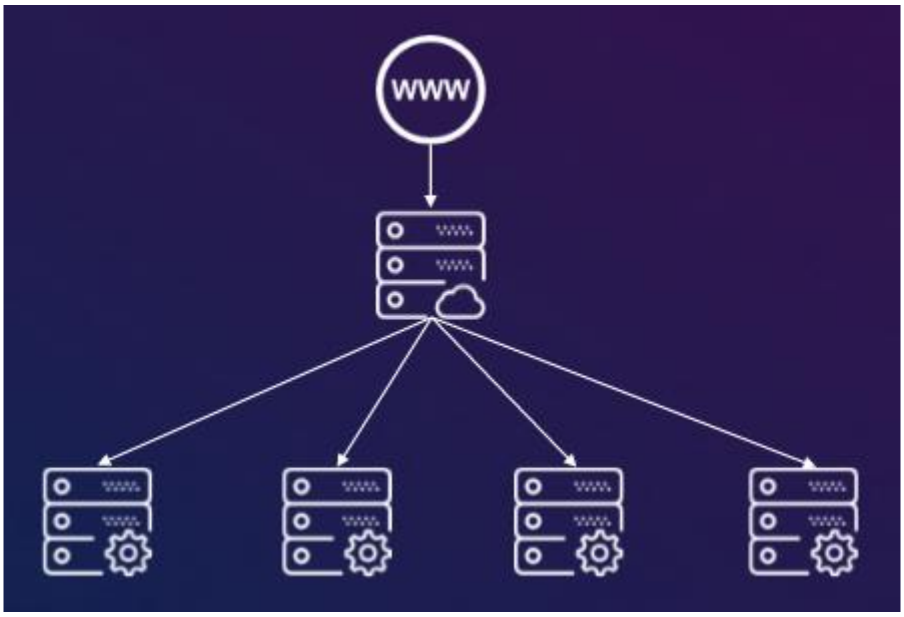


Figure 1. Load balancing between multiple servers using the domain name system (1).

The downside of the round-robin technique is that there is no way to know if a server is ready to accept more incoming traffic. A server can already be overloaded serving existing clients or it may be having network or hardware issues that prevent it from processing any more traffic. The DNS server is unaware of possible issues since its only job is to tell clients what IP address to connect to, regardless of the status of the server at that IP address. However, round-robin is easy to set up and requires close to none maintenance after it has been set up by a system administrator. A system administrator only has to make changes in the settings of a DNS server.

Each domain name has its own DNS settings on the DNS servers responsible for that domain name. These settings store information about hostnames and map them to IP addresses. These mappings are used to allow clients, such as web browsers, to connect to the correct web server. The settings are stored in a text file called a DNS zone, and each mapping is called a DNS record (2). For instance, a DNS record can be created to map the IP address of 1.2.3.4 to a hostname www.example.com. This type of record that maps a hostname to an IP address is called an A record. Usually, a DNS zone only has hostnames that map to a single IP address. When a client program asks for the IP address of that hostname, the DNS server always returns the same single IP address. However, a DNS server can be configured to respond with multiple IP addresses for the same hostname. This allows multiple web servers to host the same content, allowing for simple load balancing of the website. If a DNS zone file is created with multiple A records with different IP addresses for the same hostname, the DNS server returns a list of all the IP addresses. The client then chooses one of the IP addresses and tries to connect to it. Modern web browsers are smart enough to attempt connection to the other IP addresses if the first one does not respond (3). A browser tries to connect to each of the returned IP addresses one by one until a web page can be loaded. When one server is unavailable, the browser communicates with the rest of the servers in the load balancing group as shown in Figure 2.

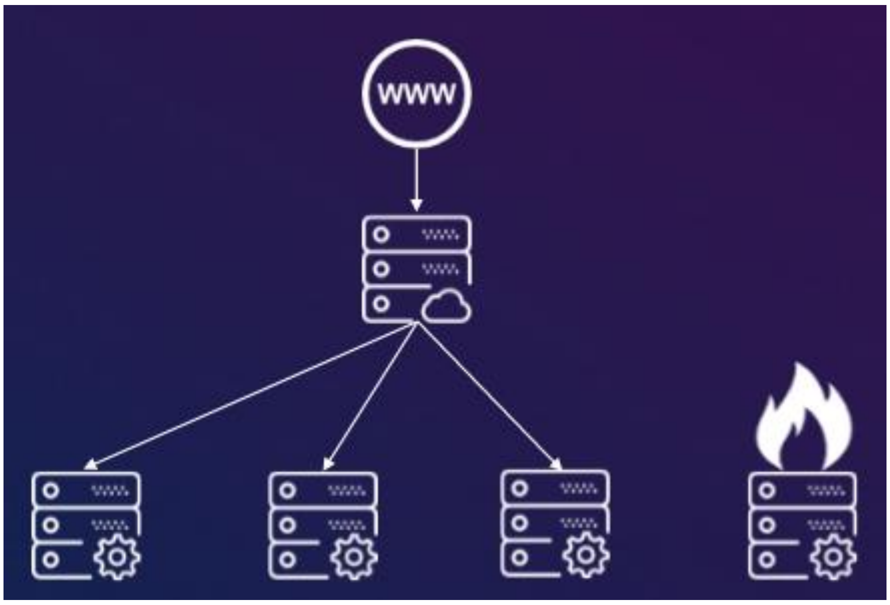


Figure 2. Load balancing in a scenario where one server is unavailable to accept traffic (1).

DNS can also be configured to distribute traffic based on the location of a client. This allows clients to connect to the geographically nearest server, which helps reduce load times since traffic is taking the shortest path to the destination. Support for geographical DNS, also called GeoDNS, depends on what DNS server software is used. For example, popular DNS server BIND supports GeoDNS. With GeoIP, BIND checks the IP address of the client asking for DNS information, and compares it to a list of IP addresses to find out which country the request came from. It then returns the IP address of the nearest web server to the client. (4.)

GeoDNS can also be implemented using Anycast. Anycast is a way of routing Internet traffic so that connections are automatically routed to the nearest IP address (5,6). With Anycast, a single IP address can map to multiple IP addresses. With Anycast, the DNS server only return one Anycast-enabled IP address to clients. Anycast can be used on the IPs returned by a DNS server and on the DNS server itself.

DNS can be used to implement simple high availability and load balancing. However, the possibilities to customize it for a specific purpose are limited. For instance, traffic cannot be directed to a server based on its current load (2). This is because DNS was not originally designed to be used as a high-availability solution but simply to map hostnames to IP addresses. Although load balancing with DNS has some limitations, it is a simple way to create a group of load balanced servers. After the initial DNS zone file configuration, this method requires no maintenance unless an IP address must be changed. It should be noted that if the DNS servers are self-hosted and not from a managed DNS service, the server software should be updated regularly to address possible security issues. With a managed DNS service this is not a concern since maintenance is done by the service provider and DNS zones can often be managed using a web-based user interface. Regardless of who manages the servers, DNS load balancing is often the cheapest solution for simple load balancing.

The DNS of a domain name can be hosted on a single server, which can be a virtual or a dedicated server. However, at least two name servers are recommended for redundancy against server and network issues. DNS servers have low memory and CPU requirements, and the price of deploying such servers remains low, generally below 10 Euros per month per server (7). Managed DNS hosting services fall into the same price category where prices start from a few Euros per month (8). If a website is hosted using a shared web hosting service, the hosting provider usually offers free DNS hosting along with website hosting. Some providers, such as Cloudflare, offer free DNS hosting for websites regardless of how a website is hosted (9). Having multiple choices of different price levels available makes it easy to get started with DNS-based load balancing with little to no starting costs.

Round robin is only one of the ways of using DNS for load balancing. A more advanced DNS-based load balancing method is called DNS delegation. In the delegation method, a primary DNS server is responsible for the DNS zone of the domain name, such as example.com. A subdomain, for example www.example.com, has its own DNS server. The DNS server of example.com is configured to respond that the DNS of of www.example.com is handled by another DNS server by using an A record (10). In addition to serving as another DNS server, the server at www.example.com is also configured to host the website at the same address. The idea is to set up multiple servers like this, all with a DNS server and a web server that serves a website. The primary DNS server at example.com returns a list of IP addresses of all the DNS servers responsible for www.example.com. Each of these servers is configured to respond to DNS requests with their own IP address. This means that DNS requests and HTTP requests are handled by the same server, which is what DNS delegation is about. The configuration in Listing 1 demonstrates the delegation setup with two web servers using BIND.

```
ns1.www.example.com.  IN A      172.16.1.20
ns2.www.example.com.  IN A      172.16.1.21

www.example.com.  IN NS    ns1.www.example.com.
www.example.com.  IN NS    ns2.www.example.com.
```

Listing 1. Example configuration of DNS delegation with two web servers (10).

Delegation-based load balancing does not differ much from the round-robin technique. In fact, the primary DNS server at example.com is configured to distribute traffic across the other DNS servers in a round-robin fashion. Delegation is used on top of it to improve load balancing. The benefit of DNS delegation is that if a web server is offline, the only DNS server that would tell clients to connect to it would also be offline. Therefore, web server and other client programs do not even attempt to connect to it and wait for a response. This makes loading times shorter and clients only receive IP addresses of servers that are online. The primary DNS server can also be configured to use GeoDNS (10). This allows it to return an IP address that is closest to the client to further reduce loading times.

Although DNS load balancing is affordable and simple to set up and there are multiple ways to implement it, there are some downsides to it as well. One such downside is that responses from DNS servers are usually cached (11). Caching means that the DNS server returns a list of IP addresses that may not be up to date and may contain IP addresses of servers that are offline. Each device, such as a computer or mobile phone, needs to know the IP address of at least one DNS server to be able to use the Domain Name System. By default, the DNS server addresses come from the Internet Service Provider (ISP) of the Internet connection that is being used). If a DNS server does not know the IP address of a hostname, it has to ask other DNS servers for that information and then return the response to the client device. Therefore, DNS servers are configured to cache common DNS requests so that they do not have to request the same information from other DNS servers again and again (2). Caching may not be an issue with modern browsers that automatically try other IP addresses returned by the DNS server, but other programs such as File Transfer Protocol (FTP) or Secure Shell (SSH) clients might not support it. If a program does not support retrying with multiple IP addresses, the program acts as if there is no DNS failover set up for the service, which may have a negative impact on the uptime of that service. Uptime refers to the amount of time an online service is online, usually measured in percentages. For instance, an uptime of 99% means that the service is available 99% of the time. Even if a client supports retrying the connection with a different IP address, attempting to connect to multiple servers one at a time will increase loading times.

3 Software-based high availability

Software-based high availability takes load balancing to the next level. Instead of using DNS to create a highly available server infrastructure, it is possible to use software specifically built for that purpose. Software load balancers offer more customization and better management compared to DNS load balancing. Some popular software load balancers are HAProxy and Nginx. HAProxy is a program dedicated to load balancing while Nginx is an HTTP server that can be configured to work as a load balancer.

HAProxy is free software that has been designed for load balancing Internet traffic and especially for websites that receive a large amount of daily traffic (12). Many popular websites, such as GitHub, Stack Overflow, AWS, Twitter and the service provider OVH use it in their infrastructure to improve the availability of their services (1). HAProxy consists of two components: front end and back end, which are illustrated in Figure 3.

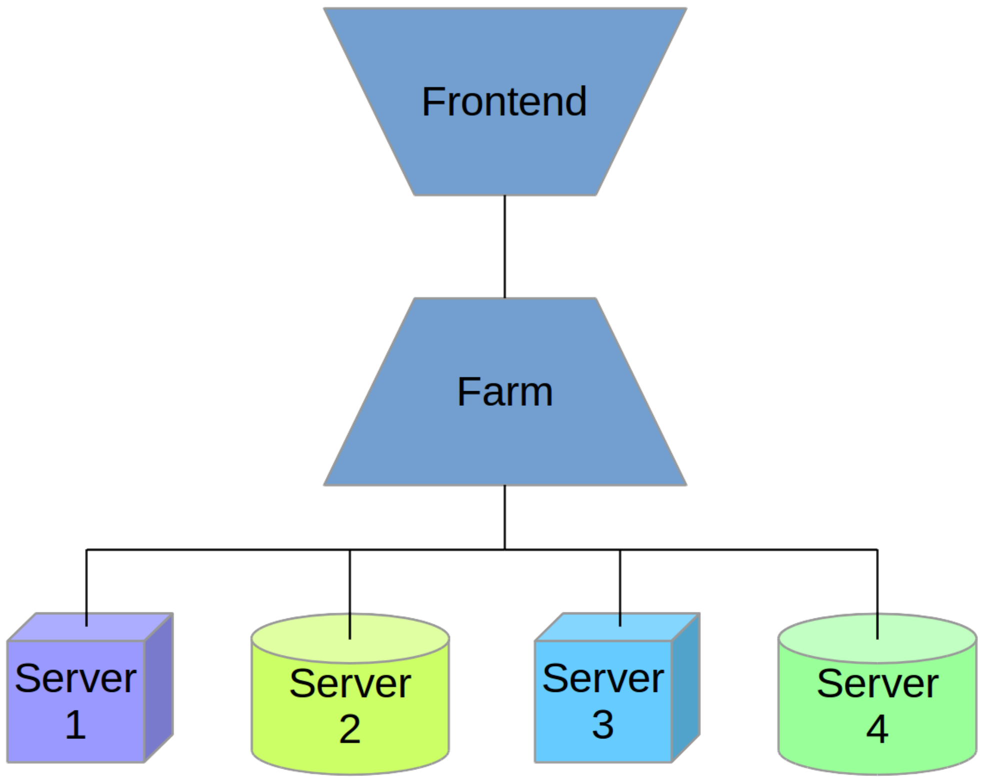


Figure 3. High availability and load balancing implemented using HAProxy by the service provider OVH (1).

The back end is a component that consists of one or multiple web servers. Traffic is split between the servers in a way that has been configured in the settings of HAProxy. One of the supported ways to split traffic is the round robin technique. It works the same way as with DNS: traffic is distributed between servers one by one regardless of whether the server is online and available to accept more traffic. (13.) HAProxy also supports more advanced setups, such as configuring a large website to be hosted by multiple groups of servers. HAProxy can be customized using text-base configuration files. Listing 1 shows how HAProxy can be configured to host different parts of a website separately using multiple back-end components.

```
backend web-backend
balance roundrobin
     server www1 www1.example.com:80 check
     server www2 www2.example.com:80 check

backend blog-backend
     balance roundrobin
     mode http
     server blog1 blog1.example.com:80 check
     server blog2 blog2.example.com:80 check
```

Listing 2. A sample configuration using multiple back-end components in HAProxy (13).

A single back end can support multiple servers, and therefore setting up more than one back end is not required to add more scalability to a small website. However, it may be beneficial with popular websites since it also helps with maintaining the website. If some areas of the websites work individually from each other, they can also be updated individually without causing downtime for other areas. For instance, if the blog of a website uses WordPress, a popular content management system (CMS) that relies of a database connection in order to work. WordPress and the database can be updated without affecting the rest of the website during the maintenance period.

Web browsers do not communicate directly with the back end. Instead, they send HTTP requests to the front end. When the front end received an HTTP request, it checks the rules defined in its configuration files. The rules contain information about when to accept or deny a request and which back end the request should be redirected to. Listing 2 is an example configuration that redirects all blog traffic to its own back end and the rest to a default back end. It is assumed that the blog can be found at www.example.com/blog and it uses the HTTP protocol at port 80.

```
frontend http
     bind *:80
     mode http

     acl url_blog path_beg /blog
     use_backend blog-backend if url_blog

     default_backend web-backend
```

Listing 3. Configuring the front-end component of the HAProxy load balancer. (13).

Implementing a software-based load balancing is technically more challenging than DNS-based load balancing. Setting up a fully functional configuration requires experience with the chosen load balancer software. In addition to that, it requires at least two servers: one front end and one back end. The servers can be either virtual or dedicated servers and do not have high requirements for memory, CPU power or disk space. However, to make setting up high availability worthwhile, there should be more than one server in the back-end group. This ensures that when one of the back-end components becomes unavailable, the rest of the back ends can continue to host the service. Since this method of load balancing is software-based the configurations stored on the servers are easier to back up and restore in case of hardware failure compared to physical appliances (14, p. 10). If a hardware failure occurs, the load balancer can be restored from a backup image or snapshot on a new server, making it fast and rather simple to set up an identical load balancer on a new server or in a new location.

Since HAProxy is free software, the only cost comes from the servers used as front ends and back ends and possible technical maintenance. Keeping the server software up to date and other possible maintenance work can increase costs of this method, although it is possible to automate typical software updates. Since all servers in a HAProxy setup can be either virtual or dedicated, the price for a small and simple high availability setup varies from approximately 10 Euros to hundreds of Euros depending on the prices and hardware of the servers (7).

4 Implementing load balancing on a hardware level

Hardware-based load balancing works significantly in a different way compared to DNS and software load balancing. Hardware-based load balancing requires that special hardware be installed in a data center as well as configuring them. Hardware-based load balancing can also be configured to use the round-robin technique although it supports other methods as well, such as distributing traffic based on the load of servers (15). Load balancing implemented using hardware works in the same way as other load balancers: the load balancer receives all traffic and distributes it forward to actual servers that have been configured to host an online service. Figure 4 illustrates how traffic is delivered from the Internet through a hardware load balancer device (HLD) to multiple servers in a data center.

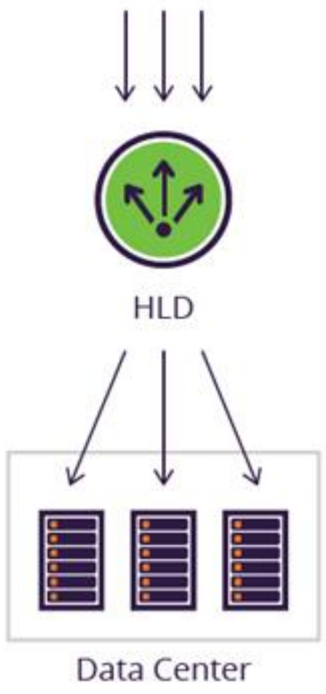


Figure 4. Load balancing between servers using hardware load balancer devices (HLDs) (16).

The advantage of load balancers that are installed in data centers is their scalability to support large amounts of traffic (15). If a website becomes too large for one load balancer to handle, more load balancers can be installed to increase capacity (14, p. 9). A single load balancer can typically handle a large amount of traffic since the device has been designed and optimized only for one task: load balancing (17). Some hardware devices can also handle tasks other than just load balancing. These kinds of devices are called application delivery controllers (ADCs). (18.) Application delivery controllers can store responses to common HTTP requests in memory so that the request does not need to reach an actual server and can be returned quickly by the load balancer itself (15). This not only makes sending a response faster but also reduces the load of servers. Hardware-based load balancers can also handle SSL and TLS transactions. Since establishing and maintaining a secure connection requires processing power, this also reduces server load. The process of handling SSL connections on a load balancer instead of application servers is referred to as SSL offloading. (19.) SSL offloading works by terminating an SSL connection at a load balancer and forwarding the unencrypted version of the request to an application server. Since the server does not have to decrypt the connection, it can process the request without spending time and processing power to decrypt the request. Load balancers can also compress network traffic into a smaller size so that the same amount of information can be transferred using less data transfer (19). ADCs can also function as firewalls to prevent malicious traffic from reaching web servers (20). Thanks to these advanced features, it is possible to save server and network resources, which helps in scaling a web service for larger audiences. ADCs are often considered a more modern version of load balancers that can only do load balancing. Some companies have switched their existing load balancers with application delivery controllers to make them more useful (18).

With hardware load balancing, it is also possible set up load balancing between data centers. This type of load balancing is often called global server load balancing (GSLB). In global server load balancing, network traffic is divided between two or more data centers as shown in Figure 5.

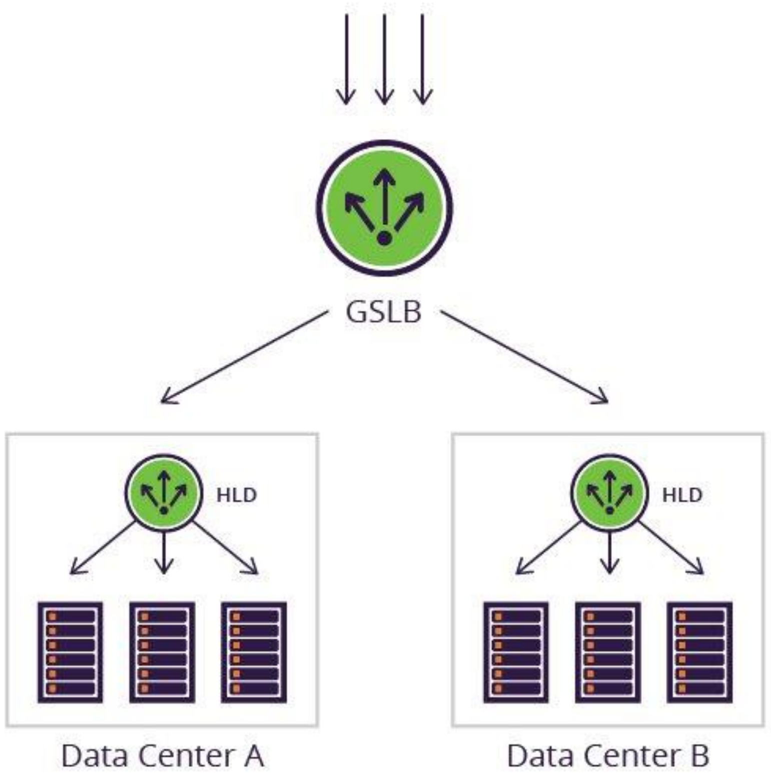


Figure 5. Load balancing between data centers using global server load balancing (15).

Global server load balancing requires one hardware load balancer (HLD in Figure 5) per data center. Like with software and DNS load balancing, the load balancer must also be installed and configured before it can be used. One additional load balancer (GSLB in Figure 5) is also needed to distribute traffic between data centers. This type of high availability helps prevent large outages that affect whole data centers at once. If one data center becomes unavailable, all traffic will be routed to the other data center. An outage that affects a whole data center can be caused by natural disasters or large-scale Internet connection interruptions.

When communicating with a web service while using global server load balancing, the first step is to resolve the IP address of the web service using DNS. This process is no different when using global server load balancing, thus a DNS query is sent to a DNS server. The difference is in the response returned by the DNS server. Instead of returning the IP address of a web server directly, it returns the IP address of the load balancer, and the client program connects to it. When the load balancer receives the request, it forwards it to one of the available web servers. When a web server has processed the request, it is returned to the client by the load balancer. This kind of lifecycle is illustrated in Figure 6.

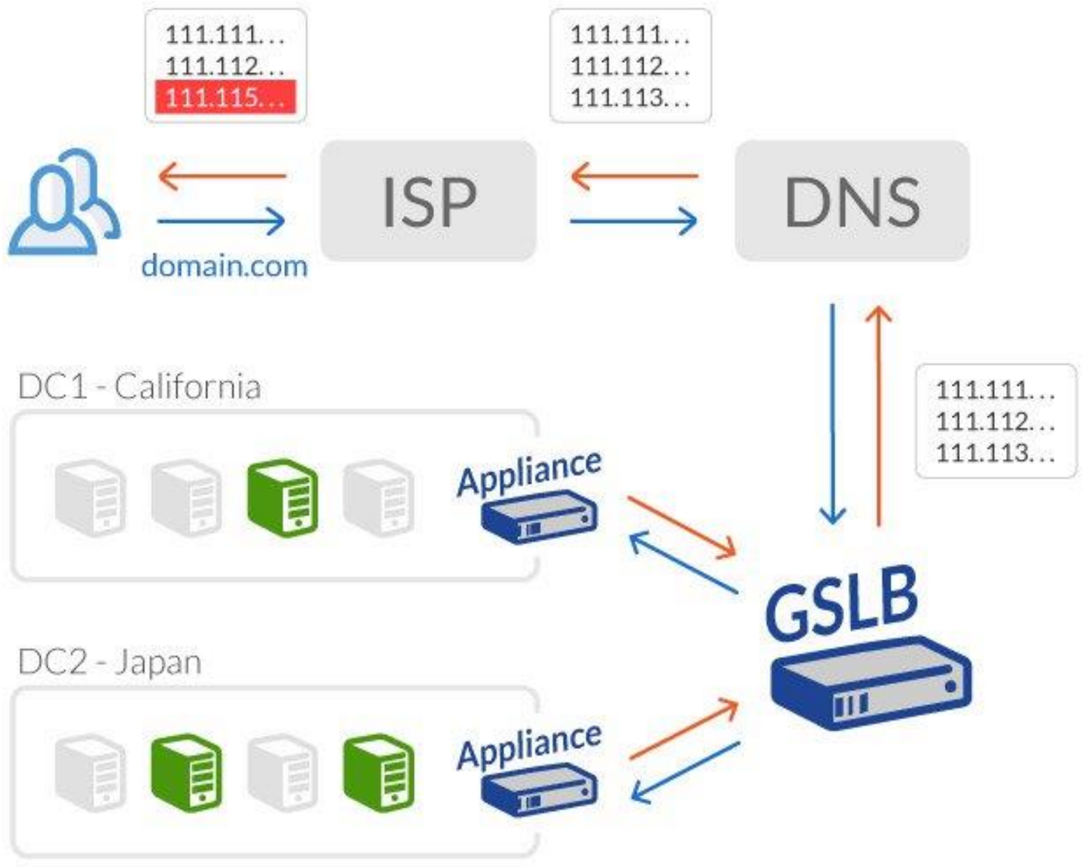


Figure 6. The lifecycle of an HTTP request when using global server load balancing (18).

Hardware load balancers allow for implementing high availability in ways that DNS or load balancer software is not capable of supporting. However, one major downside of hardware-level load balancing is the initial cost and technical complexity both for installation and technical maintenance. Setting up load balancing for a single website in a single location can be simple, but load balancing for multiple websites in many locations can be difficult to set up and manage. (14, p. 9). One additional downside to note is the possibility of hardware failure. Just like a server, hardware load balancers can suffer from hardware failures at any time and replacing a load balancer requires a new corresponding unit. Since hardware load balancers are proprietary hardware, a new unit must either be purchased beforehand or be ordered from the manufacturer, in case of which replacement may take longer than expected (14, p. 10). Hardware load balancers can still be a good choice with servers hosted on-premise because of their scalability for high-traffic websites. Hardware load balancers are often meant for data centers, although it is possible for a single website to benefit from hardware load balancing if the website receives a large amount of traffic.

5 Load balancing in the cloud

Cloud hosting has become a popular alternative to self-managed servers located on the premises of a company or in a data center. Cloud hosting offers some benefits over self-hosted environments, such as simplicity and not having to manage everything by yourself. Getting started is easier and does not require large investments in hardware since everything runs in data centers managed by cloud hosting companies (21). Cloud hosting can be used for many purposes, such as web hosting, backups and working with big data (22). The possible use cases for cloud hosting are expanding constantly. In February 2017, Digital Ocean introduced cloud load balancers for their customers and other companies have done the same since then (23).

Cloud load balancing is somewhat similar to the other load balancing methods above, although there are significant differences too. The most important difference is in where the load balancer is located. With cloud load balancing, the load balancer can be either a software or hardware load balancer, depending on how the service provider has set up their environment. In both cases, the load balancer is located in the data center of the service provider and requires no maintenance from the company that uses the load balancing service. Since cloud load balancing is a managed product offered as a single service, it is often called load balancing as a service (LBaaS). Like with other types of load balancing, all traffic is received by the load balancer and redirected to servers in a load balancing cluster. Figure 7 shows a typical load balancing setup where traffic is sent from the Internet to a load balancer (LBaaS) and then forwarded to a group of servers.

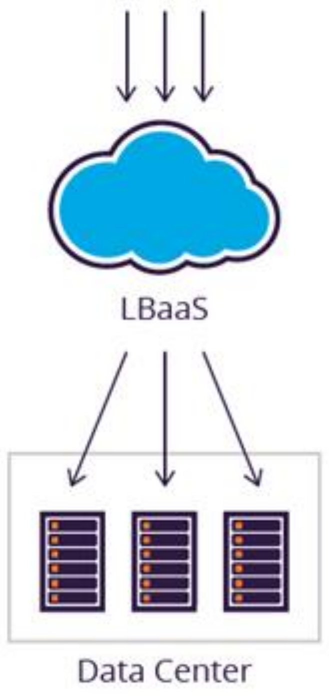
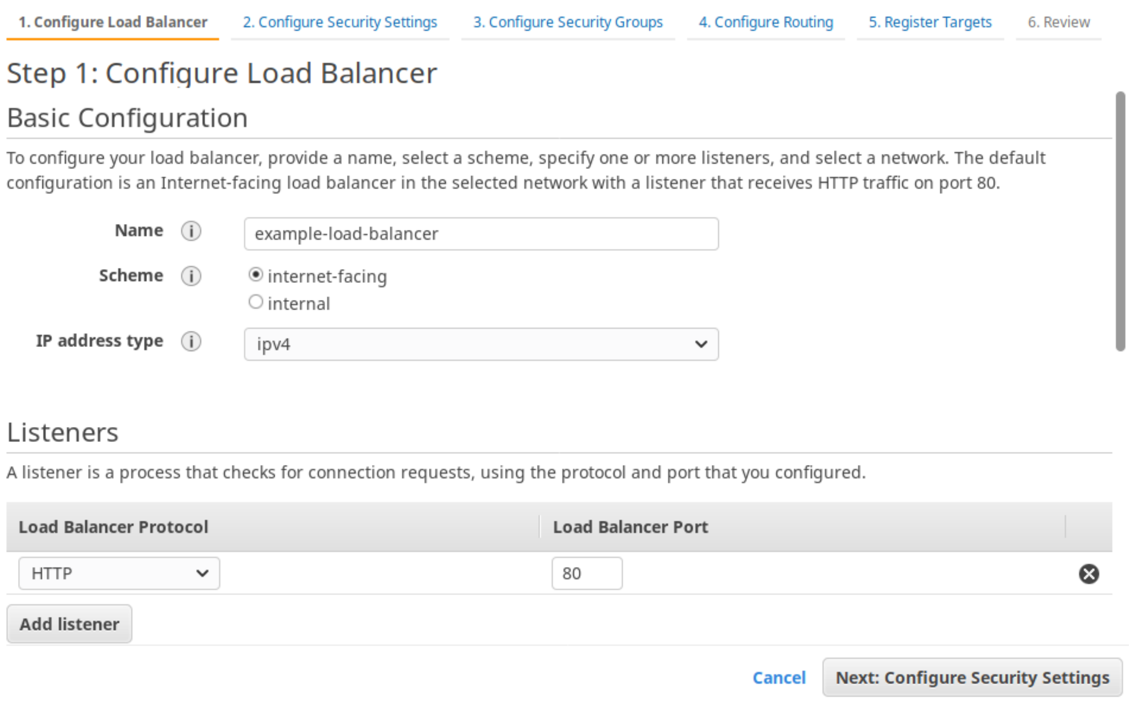


Figure 7. Load balancing between servers using load balancer as a service (LBaaS) (16).

Like the other load balancing methods, cloud load balancers must also be configured before they can be used. This can usually be done with a web browser's graphical user interface on the website of the service provider. Load balancers are often set up using setup wizards with a few steps to follow to get the load balancer configured for your needs. The following steps serve as a simplified walkthrough of creating an Elastic Load Balancer (ELB), the cloud load balancer of the popular cloud computing platform Amazon Web Services (AWS). The walkthrough gives a concrete example of setting up a load balancer using a cloud hosting service. Cloud load balancers are quite customizable and support many configurable options. In this example some of them are left to their default values for simplicity. Figure 8 shows the first step of this process where the most important decision to choose the port that the load balancer receives traffic to. In this example, the load balancer will handle HTTP traffic. Therefore, the protocol is HTTP and port is 80.


1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review

Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.

Name: example-load-balancer

Scheme: [✓ internet-facing] internal

IP address type: ipv4

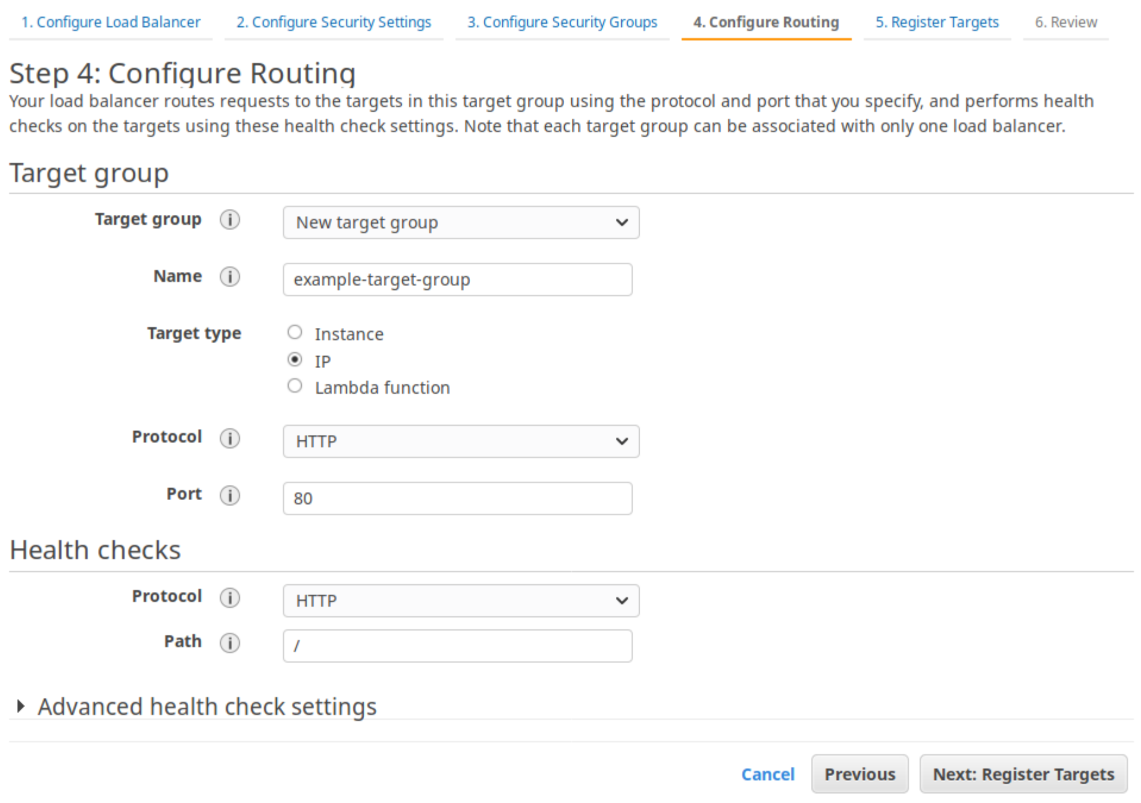
Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80

Figure 8. Configuring a port for the cloud load balancer to receive traffic to.

The next step is to configure the entry point that receives traffic called targets in Elastic Load Balancer. In Figure 9, the selected target type is an IP address and it listens for HTTP traffic on port 80. This means that all HTTP traffic that the IP address receives at port 80 will be redirected to a group of servers in a load balancing cluster.


1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

Target group

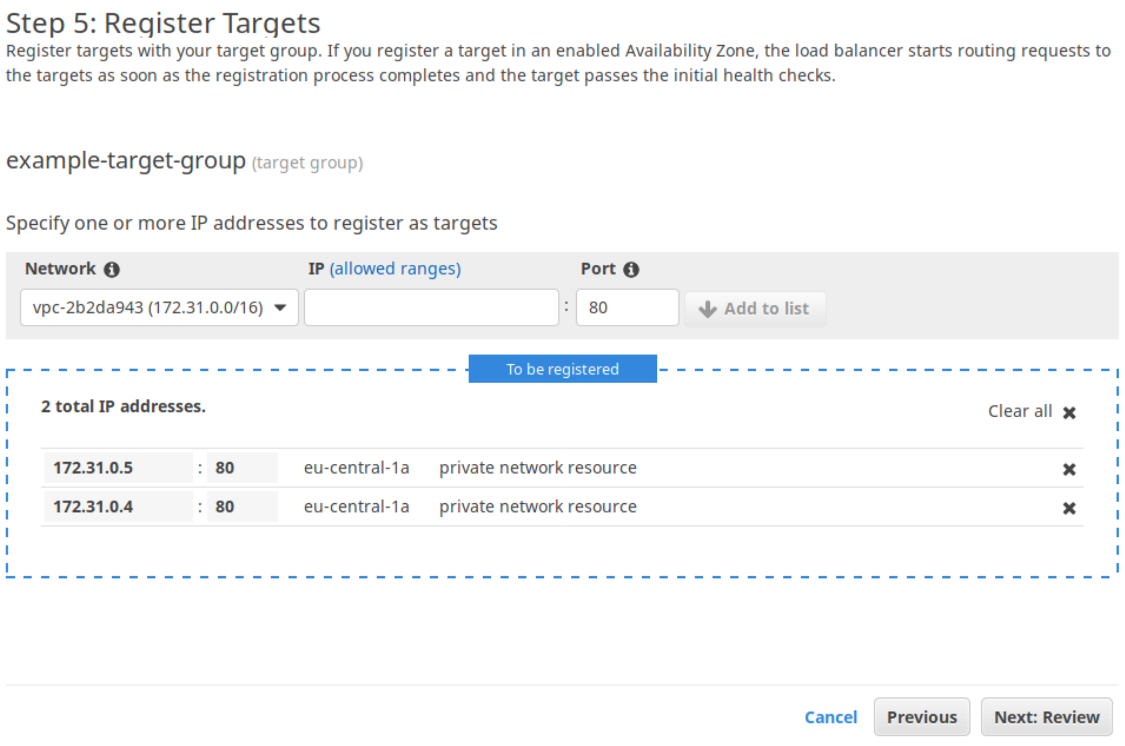
Target group	New target group
Name	example-target-group
Target type	○ Instance ● IP ○ Lambda function
Protocol	HTTP
Port	80

Health checks

Protocol	HTTP
Path	/

Figure 9. Configuring the entry point for an Elastic Load Balancer.

In the final step it is time to configure the servers that the load balancer redirects traffic to. In AWS, this can be done by specifying the IP addresses of the servers. In the example shown in Figure 10, there are two target servers with the IP addresses of 172.31.0.4 and 172.31.0.5. Both servers are virtual private servers hosted in AWS Elastic Compute Cloud (EC2), a service for hosting virtual private servers in the AWS. It is also possible to choose the port that all traffic is redirected to. In this case, 80 was chosen since the traffic is HTTP traffic.



Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

example-target-group (target group)

Specify one or more IP addresses to register as targets

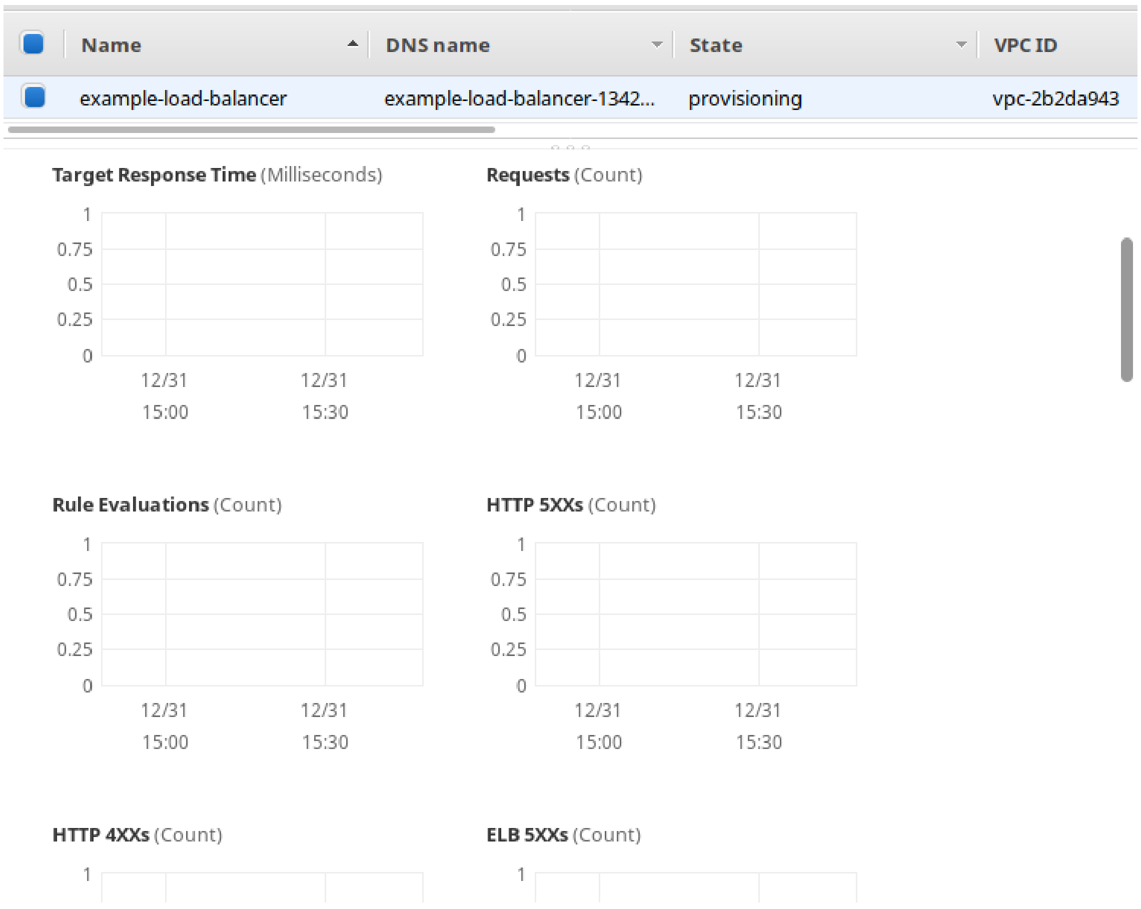
Network	IP (allowed ranges)	Port
vpc-2b2da943 (172.31.0.0/16)		80

2 total IP addresses.

172.31.0.5 : 80	eu-central-1a	private network resource	✖
172.31.0.4 : 80	eu-central-1a	private network resource	✖

Figure 10. Configuring the servers that the load balancer redirects traffic to.

Once the target servers have been configured, the Elastic Load Balancer can be created. The setup typically takes a few minutes at most. When the setup is completed, the AWS console shows a hostname that has been associated with the Elastic Load Balancer, such as example-load-balancer-0123456789.eu-central-1.elb.amazonaws.com. When HTTP traffic is sent to this address at port 80, the load balancer automatically redirects the traffic to the configured target servers. Elastic Load Balancer also performs periodic health checks for the targets. If a target goes down, traffic will no longer be redirected to that server (24). Additionally, an alert can be sent to a system administrator to notify that a one of the servers is no longer able to accept traffic. The AWS management console also offers tool to monitor the health of the load balancer cluster in real-time to detect and tackle issues proactively. Figure 11 shows some of the statistics that can be viewed using the management console. It is possible to track the average response time, the number of requests gone through the load balancer, how many requests have resulted in HTTP errors and more. These kinds of statistics can be used to detect issues in loading times and determine if website visitors are encountering issues on the website. While not directly related to server uptime, these cases affect usability and user experience on the website.



	Name	DNS name	State	VPC ID
	example-load-balancer	example-load-balancer-1342...	provisioning	vpc-2b2da943

Figure 11. Performance and error statistics of the Elastic Load Balancer service.

Cloud load balancing aims to be simple to set up and require as little management as possible from their users. The setup process typically takes between a few minutes to a few hours, as opposed to for example hardware load balancing that may take days to set up. The cost of setting up a cloud load balancer is generally very low since most cloud services operate using a pay-as-you-go payment model in which you only pay for the services when they are in use. Elastic Load Balancer requires no upfront payment or setup fees, unlike with software and hardware load balancing where servers or hardware must be purchased beforehand. Getting started requires a free AWS account and an active payment method, such as a credit card. When a load balancer is created, it will be billed based on two metrics: each hour the load balancer is running and capacity of the load balancer measured in Load Balancer Capacity Units (LCUs). An LCU defines for instance how many new connections can be established every second and how many active connections can be maintained per minute. One load balancer costs approximately 0.0225 USD per hour and each LCU approximately 0.08 USD per hour (25). Although using the load balancer is billed hourly, the price will be charged once at the end of the month. One load balancer can have as many LCUs as needed to keep up with traffic demands. On the basis of this pricing information, it can be concluded that getting started with a cloud load balancer typically costs only a few cents, and each hour costs a few cents, depending on how much traffic is handled by the load balancer. While this can be seen as a clear advantage compared to software or hardware load balancing where setup fees are high, payment models like this can be confusing and estimating the real cost of high-traffic load balancers can be difficult. To alleviate this issue, many cloud hosting providers, including AWS, have a calculator that allows customers to estimate how much they would pay for the service before signing up for it. Either way, this type of payment model prevents from paying for unused load balancing capacity. Because you only pay for what you use, the fees grow steadily as the load-balanced service becomes more popular.

Although Elastic Load Balancer runs in the AWS cloud, it is possible to use it with servers hosted outside of AWS (24). Some cloud load balancing providers do not allow this, which makes it important to check whether it is possible before choosing a service provider. This feature can be used to help migrating existing services hosted on-premise to the cloud, add more capacity to existing to an on-premise infrastructure or outsource parts of high-availability and failover capacity to the cloud. Similar to global server load balancing across multiple data centers, AWS Elastic Load Balancer supports cross-zone load balancing. With cross-zone load balancing, it is possible to distribute traffic across multiple availability zones that are located in different geographical locations.

Perhaps the most beneficial feature of cloud-based load balancing is that the load balancer is fully managed by the service provider. The provider takes care of regular updates, hardware failures, device replacements and makes sure that the only task left for users is to configure the service to their liking. Configuration can be done using a graphical user interface (GUI) instead of having to work with text-based configuration files. The user interface is often a web page with forms whose user-entered values are turned into similar configuration files that software-based load balancers are configured with. This method provides an easy way to configure a load balancer using a GUI. Depending on the service provider, the feature set of a cloud load balancer is usually similar to a software or hardware load balancer. For instance, cloud load balancers can cache common HTTP requests like hardware load balancers. Some load balancing services are also capable of preventing attacks, such as distributed denial-of-service (DDoS) attacks and SQL injections. (14, p. 11.)

6 Choosing the right load balancing solution

Choosing the right way to set up high availability and load balancing is not always straightforward. Some methods are generally considered more effective than others, but before making the final decision, the context and requirements for a specific high-availability setup should be considered. For example, the type of service and how critical its uptime is as well as how many daily users there are should all be considered. These factors help rule out methods that are too inefficient for the amount users and methods that are overly complicated and expensive to maintain if the user base is small or if uptime is not mission critical.

For small-business owners, single website owners and individuals, price can be an important deciding factor. In that case, DNS load balancing might be enough to cover the needs of a small web service. DNS load balancing can be done for free or for a small monthly fee using managed DNS hosting services. It is also possible to manage DNS servers yourself, in which case a few virtual or dedicated servers must be paid for monthly. DNS load balancing is a simple solution to a website that has outgrown the capacity of a single server or needs to be protected from a single point of failure. DNS load balancing requires little to no maintenance and is therefore suitable for web services that are not monitored by staff around the clock. The setup process is rather simple, which makes it possible for a single website owner without technical background to set it up. Since DNS load balancing may not as efficient as other solutions, it is possible to change to a different load balancing and high availability setup once DNS load balancing is not effective enough anymore. Since DNS-based load balancing does not scale well for high-traffic web services, it is usually most suitable as the first type of load balancing for a small growing website. However, service availability might be critical even for a low-traffic website, and depending on the type of service or website, every minute of downtime might negatively affect customer satisfaction, sales, company reputation or it may drive existing customers away to use competing services. Regardless of the size or popularity of a website, other choices should be considered if uptime is critical for the operation of a service.

Another low-budget high-availability and load-balancing solution is cloud-based services. Getting started with cloud load balancing typically costs only a few Euros thanks to the pay-as-you-go payment model of most cloud services. When configuring a cloud load balancer for the first time, the load balancer receives little to no traffic, and therefore the cost will be low while the high-availability infrastructure is being set up. The actual operating costs vary based on how much traffic passes the load balancer, and this is what makes cloud load balancing suitable for almost all types of services. A small online service is likely to generate only a small amount of income, but the price of load balancing will also be low. Likewise, a popular website typically generates more income and should be enough to cover the higher cost of load balancing. Since the cost of cloud load balancing is related to the number of users, it is price-wise suitable for all sizes of online services. Maintaining the load balancer does not require a dedicated team. Software update and other types of maintenance is taken care of by the service provider. Setup has to be done only once, and the process is usually easier than setting up a software or hardware load balancer. There is no need to set up a new load balancer every once in a while when the load-balancer service becomes more popular. If a company does not have many system administrators or wants to focus on tasks other than system administration, cloud load balancing may be the way to go. Since cloud load balancing is managed by the service provider, they are responsible for the uptime of the load balancer. While the uptime of such load balancing services is generally very high, in case of outages, there is not much a single user of the service can do other than wait for the service provider to fix technical issues. This kind of loss of control can be considered a downside of cloud hosting in general. In-house solutions, such as self-hosted software and hardware load balancers are better for a greater degree of control. However, unlike with self-hosted solutions, the amount of labor to maintain the load-balancing solution does not increase as the service grows in popularity. Cloud load balancing services usually come with analytics tools that can be used to view how much and what type of traffic goes throw the load balancer (26). These kinds of tools help in estimating future growth, detecting malicious traffic, the number of visits to error pages and more. It is also possible to automate cloud load balancers if the service provider offers an API (26). This allows companies to integrate the management of the load balancer with their internal or external systems. For instance, when a company creates a new website, it can automatically set up load balancing rules and other settings for that website using the API of the load balancing service provider.

Software load balancing is similar to cloud load balancing because ultimately cloud load balancing is also based on software. The most important difference is that cloud load balancing is a managed service while software load balancing is self-managed. Before choosing a self-managed high-availability solution, it should be made sure that there is staff with enough experience and time to manage the load balancers. This includes the initial installation, setting up new load balancers to increase capacity, software updates, replacement of failed load balancers and solving other possible software issues. Since software load balancing is self-managed, it gives more freedom in terms of customization than managed alternatives that offer a predetermined set of features. With software load balancing, everything, including the load balancing software, is fully configurable. It is also possible to decide the level of error tolerance choose how many servers the load balancer setup consists of. There can be only one load balancer or multiple to increase tolerance of hardware or software failure. Having more control over the infrastructure also gives the ability to choose how much to invest in a high availability setup. Both free and paid software solutions are available, and it is possible to choose how powerful and how many servers are used. However, hardware failures can cause unexpected costs. With managed services, they are covered by the monthly fees and are paid by the service provider. Software load balancing can be used even for large websites, and many popular websites, such as Stack Overflow and GitHub use it to balance the high amount of traffic their servers receive (1). It has been proven to be a good choice for large websites that cannot tolerate long or regular service interruptions.

The last type of load balancing is hardware load balancing. It is significantly different from other types of load balancing and allows for some advanced high availability setups. Global server load balancing makes it possible to load balance traffic across multiple data centers and hardware load balancers can do much more than just forwarding traffic, including handling SSL connections, compressing traffic and caching common requests. These features reduce the load of application servers by separating load balancing from server resources. Due to its high initial costs, hardware load balancing is often used by large companies and popular online services. Hardware load balancers are capable of handling large amounts of traffic, and more capacity can be added by installing additional load balancers or upgrading existing ones with more powerful ones. Having only one hardware load balancers means there is a single point of failure in case the hardware load balancer fails. To achieve true high availability, at least two load balancers should be installed. This requires some technical expertise and physical access to the servers that are being load balanced. Hardware load balancing is not only expensive but also takes time to set up and manage. This should be taken into account when considering setting up a high availability infrastructure using hardware load balancers. Since hardware load balancers are not cheap or easy to maintain, they are often only used if a website can tolerate little to no downtime. If a company has the required staff to operate hardware load balancers and low budget is not an issue, it may be a good choice.

Regardless of the type of load balancing, there are some other important aspects to consider. First, both short-term and long-term needs should be considered (27). DNS-based load balancing may be enough for a company today, but it may prove to be inefficient next year when a website is more popular. Each type of load balancing can be upscaled to a certain point, and some methods are more scalable than others. DNS load balancing does not scale very well, especially when the importance of high uptime increases. In this case, a company may be forced to change its high availability infrastructure to a different, more scalable solution. This may be a difficult and long process and might even cause more downtime when the migration is in progress. Therefore, it is important to also consider the long-term effectiveness of each high availability solution. A second aspect to consider is money: the total cost of ownership and return on investment (27). It may be difficult to calculate an exact initial and month-to-month cost for each method, but even a rough estimate can be helpful. For cloud load balancing, most providers offer a pricing calculator that greatly helps in estimating the fees for cloud-based load balancers. With hardware load balancers, the fees mostly come from initial hardware acquisitions. If no hardware failures occur, there are little to no monthly fees. DNS and software load balancing are easy to estimate since the pricing only changes if more capacity is needed. Estimating the fees of each method gives one metric that helps in making the decision of which high availability setup to choose. A third aspect to consider is how the current server infrastructure has been set up. If a company owns its hardware and has all servers hosted on-premise, adding a few hardware load balancers to the setup sounds reasonable. If most servers are hosted by the same cloud hosting provider and the provider offers cloud load balancing, choosing their service should be on top of the list of choices. If servers are hosted by multiple independent providers or by a provider that does not offer any high-availability solutions, DNS or software load balancing might be the way to go. Some companies might also want to consider privacy and confidentiality of their data when making the decision. If the services deal with confidential data, some companies might not want to route their traffic through third-party load balancing solutions. This makes cloud hosting and software load balancing with external, non-premise servers an unattractive choice. Hardware load balancers allow for more control in many ways, not only when configuring them. They also allow a company to direct traffic to their own server infrastructure in a location of their choice, instead of relying on a third party. This can reduce the risk of leaking sensitive information to third parties. A company may also have customers or policies that demand that their data be handled in a specific location or that handling of data must meet certain security requirements. In this situation, the data can be handled in a location to which the company has physical access and access of other persons is restricted. This is a good use case for hardware load balancing. Software and DNS load balancing can also be used on-premise if the servers are hosted in-house.

7 Conclusion

Each way of implementing high availability comes with its own pros and cons. Choosing the best method depends on the situation and factors such as how critical uptime is and how many users an online service has. Table 1 acts as a simple comparison between the methods covered earlier and as a simplified way to quickly choose the most suitable solution for high availability and load balancing.

Table 1. Comparison of high-availability and load-balancing solutions.

	DNS	**Software**	**Hardware**	**Cloud**
Technical difficulty of initial setup	Easy	Medium	Difficult	Easy
Cost of initial setup	Low (< €50)	Low (< €100)	High (> €1000)	Low (< €10)
Monthly price	Low (< €50/mo)	Medium (< €100/mo)	Low (< €100/mo)	Medium (< €300/mo)
Scalability	Low	High	Medium	High
Notes	Rarely recommended alone for production use	Scalable and affordable enough for many use cases	For large companies with popular web services or for which uptime is critical	Load balancing service is sometimes only compatible with servers rented from the same company

Although DNS-based load balancing is not often suitable alone for achieving high availability, it can be enough for small websites for which uptime is not critical. However, it can be used together with other types of load balancing to further improve high availability. In general, DNS load balancing is not as reliable as other solutions. Traffic can be distributed to servers that are offline and traffic distribution cannot be affected by factors such as server load. Software or hardware load balancing is suitable for medium and large websites depending on budget and technical expertise. Hardware load balancers are more expensive and more difficult to install and manage. However, they can handle more than just load balancing, including caching common requests, handling SSL connections and act as a firewall. They also give more control over the server infrastructure of a company because everything can be hosted on-premise and physical access to servers is possible. They also help in situations where privacy and data confidentiality must be considered. Software-based solutions are easier to install, are highly scalable and can be hosted on-premise or on external servers. Software load balancing is easier to maintain if server maintenance is handled by a hosting provider. Large websites can benefit from both software and hardware load balancing.

If budget is not an issue and the staff of a company has the required technical expertise and time for maintenance, hardware load balancing might be a better choice. However, it is not a requirement for popular websites, and some the most popular websites on the Internet use software load balancing. Although hardware load balancing is technically more complex than other methods, it allows for setups that are not possible or more difficult to implement with other methods, such as global server load balancing and data confidentiality. Cloud load balancing is usually the easiest solution to set up and manage. It requires little understanding of networking and can typically be configured using a web-based UI. It also upscales automatically if a website becomes more popular. Since the pricing of cloud load balancing depends on how popular a website is and the initial cost is low, pricing-wise it is suitable for all types of websites.

References

1 OVH SAS. Introduction to the OVH load balancer [Internet]. 2018 [cited 29 September 2018]. Available from: https://docs.ovh.com/gb/en/load-balancer/loadbalancer-introduction.

2 NS1 LLC. DNS Failover - Basic Concepts and Limitations [Internet]. [date unknown] [cited 6 January 2018]. Available from: https://ns1.com/resources/dns-failover-basic-concepts-and-limitations.

3 National Bureau of Economic Research. DNS round robin for web server failover [Internet]. [date unknown] [cited 16 October 2018]. Available from: https://www.nber.org/sys-admin/dns-failover.html.

4 Hedges M. HOWTO Implement GeoDNS using BIND [Internet]. [date unknown] [cited 21 November 2018]. Available from: https://geoip.site.

5 Cloud DNS Ltd. Free Anycast DNS | ClouDNS [Internet]. [date unknown] [cited 16 October 2018]. Available from: https://www.cloudns.net/anycast-dns.

6 OVH SAS. Anycast DNS – Speed up a DNS server - Domain- OVH [Internet]. [date unknown] [cited 16 October 2018]. Available from: https://www.ovh.co.uk/domains/dns-anycast.

7 OVH SAS. VPS Hosting: your Virtual Server - OVH [Internet]. [date unknown] [cited 17 October 2018]. Available from: https://www.ovh.com/world/vps.

8 Rage4 Networks Limited. Rage4 Networks Limited [Internet]. [date unknown] [cited 21 November 2018]. Available from: https://rage4.com/#pricing.

9 Cloudflare, Inc. Our Plans | Cloudflare [Internet]. [date unknown] [cited 22 January 2019]. Available from: https://www.cloudflare.com/plans.

10 Schroder C. Simple Load Balancing with DNS on Linux [Internet]. 2018 [cited 22 January 2019]. Available from: https://www.linux.com/learn/intro-to-linux/2018/3/simple-load-balancing-dns-linux.

11 Nginx, Inc. What Is DNS Load Balancing? | NGINX [Internet]. [date unknown] [cited 5 January 2019]. Available from: https://www.nginx.com/resources/glossary/dns-load-balancing.

12 HAProxy Technologies, LLC. HAProxy - The Reliable, High Performance TCP/HTTP Load Balancer [Internet]. [date unknown] [cited 22 October 2018]. Available from: https://www.haproxy.org.

13 Anicas M. 2014. An Introduction to HAProxy and Load Balancing Concepts [Internet]. [2014] [cited 22 October 2018]. Available from:

Metropolia University of Applied Sciences

https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts.

14 Imperva, Inc. How to Select a Load Balancing Solution [Internet]. 2016 [cited 12 January 2019]. Available from: https://lp.incapsula.com/rs/804-TEY-921/images/Guide%20-%20How%20To%20Select%20A%20Load%20Balancing%20Solution%20%28new%29.pdf.

15 Imperva, Inc. What is Hardware Load Balancer (HLD) [Internet]. [date unknown] [cited 22 October 2018]. Available from: https://www.incapsula.com/load-balancing/hardware-load-balancer-hld.html.

16 Imperva, Inc. How to Select a Load Balancing Solution [Internet]. [date unknown] [cited 2 February 2019]. Available from: https://www.incapsula.com/load-balancing/hardware-load-balancer-hld.html.

17 Avi Networks, Inc. Load Balancing 101 - Learn All About Load Balancers [Internet]. [date unknown] [cited 2 February 2019]. Available from: https://avinetworks.com/what-is-load-balancing.

18 Imperva, Inc. Cloud-based Load Balancing Services [Internet]. [date unknown] [cited 2 February 2019]. Available from: https://www.incapsula.com/load-balancing/load-balancing-services.html.

19 Barracuda Networks, Inc. Barracuda Load Balancer ADC - Features [Internet]. [date unknown] [cited 23 October 2018]. Available from: https://www.barracuda.com/products/loadbalancer/features.

20 Fortinet, Inc. Application Delivery Controllers (ADC) FortiADC [Internet]. [date unknown] [cited 2 February 2019]. Available from: https://www.fortinet.com/products/application-delivery-controller/fortiadc.html.

21 Ismail N. Top 10 benefits of cloud computing [Internet]. 2017 [cited 20 December 2018]. Available from: https://www.information-age.com/top-10-benefits-cloud-computing-123467995.

22 Cloud Academy, Inc. Cloud service models and most common cloud computing use cases [Internet]. 2017 [cited 20 December 2018]. Available from: https://cloudacademy.com/blog/most-common-cloud-service-models.

23 [publisher unknown]. DigitalOcean releases load balancers, the easiest and fastest way to achieve 100% uptime for production workloads. SD Times [Internet]. 2017 [cited 20 December 2018]. Available from: https://sdtimes.com/digitalocean-releases-load-balancers-easiest-fastest-way-achieve-100-uptime-production-workloads.

24 Amazon Web Services, Inc. Elastic Load Balancing features - Amazon Web Services [Internet]. [date unknown] [cited 5 January 2019]. Available from: https://aws.amazon.com/elasticloadbalancing/features.

25 Amazon Web Services, Inc. Elastic Load Balancing pricing - Amazon Web Services [Internet]. [date unknown] [cited 5 January 2019]. Available from: https://aws.amazon.com/elasticloadbalancing/pricing.

26 Anand K. How to Choose a Cloud Load Balancer [Internet]. 2018 [cited 3 February 2019]. Available from: https://www.a10networks.com/blog/how-choose-cloud-load-balancer.

27 Zeifman I. What You Should Know Before Choosing a Load Balancer [Internet]. 2015 [cited 3 February 2019]. Available from: https://www.incapsula.com/blog/choosing-load-balancer.html.