

## How websites work

There are also a few other components that help the web run more efficiently and provide extra features.

### Load Balancers

When a website's traffic starts getting quite large or is running an application that needs to have high availability, one web server might no longer do the job. Load balancers provide two main features, ensuring high traffic websites can handle the load and providing a failover if a server becomes unresponsive.

When you request a website with a load balancer, the load balancer will receive your request first and then forward it to one of the multiple servers behind it. The load balancer uses different algorithms to help it decide which server is best to deal with the request. A couple of examples of these algorithms are **round-robin**, which sends it to each server in turn, or **weighted**, which checks how many requests a server is currently dealing with and sends it to the least busy server.

Load balancers also perform periodic checks with each server to ensure they are running correctly; this is called a **health check**. If a server doesn't respond appropriately or doesn't respond, the load balancer will stop sending traffic until it responds appropriately again.

### CDN (Content Delivery Networks)

A CDN can be an excellent resource for cutting down traffic to a busy website. It allows you to host static files from your website, such as JavaScript, CSS, Images, Videos, and host them across thousands of servers all over the world. When a user requests one of the hosted files, the CDN works out where the nearest server is physically located and sends the request there instead of potentially the other side of the world.

### Databases

Often websites will need a way of storing information for their users. Webservers can communicate with databases to store and recall data from them. Databases can range from just a simple plain text file up to complex clusters of multiple servers providing speed and resilience. You'll come across some common databases: MySQL, MSSQL, MongoDB, Postgres, and more; each has its specific features.

### WAF (Web Application Firewall)

A WAF sits between your web request and the web server; its primary purpose is to protect the webserver from hacking or denial of service attacks. It analyses the web requests for common attack techniques, whether the request is from a real browser rather than a bot. It also checks if an excessive amount of web requests are being sent by utilising something called rate limiting, which will only allow a certain amount of requests from an IP per second. If a request is deemed a potential attack, it will be dropped and never sent to the webserver.

## What is a Web Server?

A web server is a software that listens for incoming connections and then utilises the HTTP protocol to deliver web content to its clients. The most common web server software you'll come across is Apache, Nginx, IIS and NodeJS. A Web server delivers files from what's called its root directory, which is defined in the software settings. For example, Nginx and Apache share the same default location of /var/www/html in Linux operating systems, and IIS uses C:\inetpub\wwwroot for the Windows operating systems. So, for example, if you requested the file `http://www.example.com/picture.jpg`, it would send the file /var/www/html/picture.jpg from its local hard drive.

## Virtual Hosts

Web servers can host multiple websites with different domain names; to achieve this, they use virtual hosts. The web server software checks the hostname being requested from the HTTP headers and matches that against its virtual hosts (virtual hosts are just text-based configuration files). If it finds a match, the correct website will be provided. If no match is found, the default website will be provided instead.

Virtual Hosts can have their root directory mapped to different locations on the hard drive. For example, one.com apped to /var/www/website\_one, and two.com apped to /var/www/website\_two

There's no limit to the number of different websites you can host on a web server.

## Static Vs Dynamic Content

Static content, as the name suggests, is content that never changes. Common examples of this are pictures, javascript, CSS, etc., but can also include HTML that never changes. Furthermore, these are files that are directly served from the webserver with no changes made to them.

Dynamic content, on the other hand, is content that could change with different requests. Take, for example, a blog. On the homepage of the blog, it will show you the latest entries. If a new entry is created, the home page is then updated with the latest entry, or a second example might be a search page on a blog. Depending on what word you search, different results will be displayed.

These changes to what you end up seeing are done in what is called the **Backend** with the use of programming and scripting languages. It's called the Backend because what is being done is all done behind the scenes. You can't view the websites' HTML source and see what's happening in the Backend, while the HTML is the result of the processing from the Backend. Everything you see in your browser is called the **Frontend**.

## Scripting and Backend Languages

There's not much of a limit to what a backend language can achieve, and these are what make a website interactive to the user. Some examples of these languages are PHP, Python, Ruby, NodeJS, Perl and many more. These languages can interact with databases, call external services, process data from the user, and so much more. A very basic PHP

---

example of this would be if you requested the website `http://example.com/index.php?name=adam`

If `index.php` was built like this:

```
<html><body>Hello <?php echo $_GET["name"]; ?></body></html>
```

It would output the following to the client:

```
<html><body>Hello adam</body></html>
```

You'll notice that the client doesn't see any PHP code because it's on the **Backend**. This interactivity opens up a lot more security issues for web applications that haven't been created securely, as you learn in further modules.

---