**What is a Web Server?**

What Is a Web Server? A web server is usually a kind of software program that is designed to handle all of your web requests. In simple terms, the webserver is a computer that delivers web pages when they are requested for accessing the content by the end-users.

File servers, database servers, mail servers, and web servers use different kinds of server software. Each of these applications can access files stored on a physical server and use them for various purposes.

The job of a web server is to serve websites on the internet. To achieve that goal, it acts as a middleman between the server and client machines. It pulls content from the server on each user request and delivers it to the web.

The biggest challenge of a web server is to serve many different web users at the same time — each of whom is requesting different pages. Web servers process files written in different programming languages such as PHP, Python, Java, and others.

They turn them to static HTML (Hypertext Markup Language) files and serve these files in the browser of web users. When you hear the word web server, think of it as the tool responsible for the proper server-client communication.

## Types of Web Server

* Apache HTTP Server. This is the most popular web server in the world developed by the Apache Software Foundation.
* Internet Information Services. The Internet Information Server (IIS) is a high-performance Web Server from Microsoft.
* Lighttpd.
* Sun Java System Web Server.

## What is Apache – Web Server?

Apache is one of the most popular web servers that allows you to run a secure website without too much of a headache.

Apache is an open-source and free web server software that [powers around 60% of websites](https://w3techs.com/technologies/details/ws-apache/all/all) around the world. The official name is [Apache HTTP Server](https://httpd.apache.org/), and it’s maintained and developed by the Apache Software Foundation.

It allows website owners to serve content on the web — hence the name “web server.” It’s one of the oldest and most reliable web servers, with the first version released more than 20 years ago, in 1995.

When someone wants to visit a website, they enter a domain name into the address bar of their browser. Then, the web server delivers the requested files by acting as a virtual delivery man.

## How Does Apache Web Server Work?

Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server. Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). Apache is a cross-platform software, therefore it works on both Unix and Windows servers.

When a visitor wants to load a page on your website, for instance, the homepage or your “About Us” page, their browser sends a request to your server, and Apache returns a response with all the requested files (text, images, etc.). The server and the client communicate through the HTTP (Hypertext Transfer Protocol) protocol, and the Apache software is responsible for the smooth and secure communication between the two machines.

## Apache vs. Other Web Servers

Besides Apache, there are many other popular web servers. Each web server application has been created for a different purpose. While Apache is the most widely used, it has quite a few alternatives and rivals.

### **Apache vs. NGINX**

[Nginx](https://www.nginx.com/), pronounced Engine-X, is a newer web server application first released in 2004. As of today, it has gained quite a popularity among website owners. Nginx was created to solve the so-called [c10k problem](https://en.wikipedia.org/wiki/C10k_problem), meaning that a web server that uses threads to handle user requests is unable to manage more than 10,000 connections at the same time.

1. Since Apache uses the thread-based structure, owners of traffic-heavy websites may encounter performance problems. Nginx is one of the web servers that address the c10k problem and probably the most successful one.
2. Nginx has an event-driven architecture that doesn’t create a new process for each request. Instead, it handles every incoming request in a single thread. This master process manages several worker processes that perform the actual processing of requests. The event-based model of Nginx distributes user requests among worker processes in an efficient way, therefore leading to much better scalability.
3. If you need to manage a high-traffic website, Nginx is an excellent choice, as it can do that by using minimal resources. It cannot be a coincidence that it’s used by many high-visibility websites such as Netflix, Hulu, Pinterest, and Airbnb.
4. However, for small and medium players, Apache comes with a handful of advantages over Nginx, such as its easy configuration, lots of modules, and a beginner-friendly environment.

### **Apache vs. Tomcat**

Tomcat is a web server also developed by the Apache Software Foundation, thus its official name is [Apache Tomcat](https://tomcat.apache.org/). It’s an HTTP server as well, however it powers Java applications instead of static websites. Tomcat can run several different Java specifications such as Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket.

1. Tomcat has been created specifically for Java apps, while Apache is a general-purpose HTTP server. You can use Apache together with different programming languages (PHP, Python, Perl, etc.) with the help of the appropriate Apache module (mod\_php, mod\_python, mod\_perl, etc.).
2. Although you can use a Tomcat server to serve static web pages as well, it’s less efficient for that purpose compared to the Apache server. For instance, Tomcat pre-loads the Java Virtual Machine and other Java-related libraries you won’t need on most websites.
3. Tomcat is also less configurable compared to other web servers. For example, to run [WordPress](https://www.hostinger.com/wordpress-hosting), the best choice is a general-purpose HTTP server such as Apache or NGINX.

## Apache Pros and Cons

An Apache web server can be an excellent choice to run your website on a stable and versatile platform. However, it also comes with some disadvantages you need to pay attention to.

**Pros**:

1. Open-source and free, even for commercial use.
2. Reliable, stable software.
3. Frequently updated, regular security patches.
4. Flexible due to its module-based structure.
5. Easy to configure, beginner-friendly.
6. Cross-platform (works on both Unix and Windows servers).
7. Works out of the box with WordPress sites.
8. Huge community and easily available support in case of any problem.

**Cons**:

1. Performance problems on extremely traffic-heavy websites.
2. Too many configuration options can lead to security vulnerabilities.

# **What is a LAMP Stack?**

The LAMP stack is the foundation for Linux hosted websites is the Linux, Apache, MySQL and PHP (LAMP) software stack.

## The Four Layers of a LAMP Stack

Linux based web servers consist of four software components. These components, arranged in layers supporting one another, make up the software stack. Websites and Web Applications run on top of this underlying stack. The common software components that make up a traditional LAMP stack are:

* **Linux**: The operating system (OS) makes up our first layer. Linux sets the foundation for the stack model. All other layers run on top of this layer.
* **Apache**: The second layer consists of web server software, typically Apache Web Server. This layer resides on top of the Linux layer. Web servers are responsible for translating from web browsers to their correct website.
* **MySQL**: Our third layer is where databases live. MySQL stores details that can be queried by scripting to construct a website. MySQL usually sits on top of the Linux layer alongside Apache/layer 2. In high end configurations, MySQL can be off loaded to a separate host server.
* **PHP**: Sitting on top of them all is our fourth and final layer. The scripting layer consists of PHP and/or other similar web programming languages. Websites and Web Applications run within this layer.

We can visualize the LAMP stack like so:

[A picture containing diagram

Description automatically generated](https://lwstatic-a.akamaihd.net/kb/wp-content/uploads/2018/02/kb-lamp-stack.jpg)

### **Some Alternatives**

The four traditional layers of a LAMP stack consist of free and open-source products. Linux, Apache, MySQL and PHP are the cornerstone of a free, non-proprietary LAMP stack. There are several variants of the four stack model as well. These variants use alternative software replacing one or more of the traditional components. Some examples of these alternatives are:

* **WAMP**: Windows, Apache, MySQL & PHP
* **WISA**: Windows, IIS, SQL & ASP.net
* **MAMP**: MacOS, Apache, MySQL & PHP

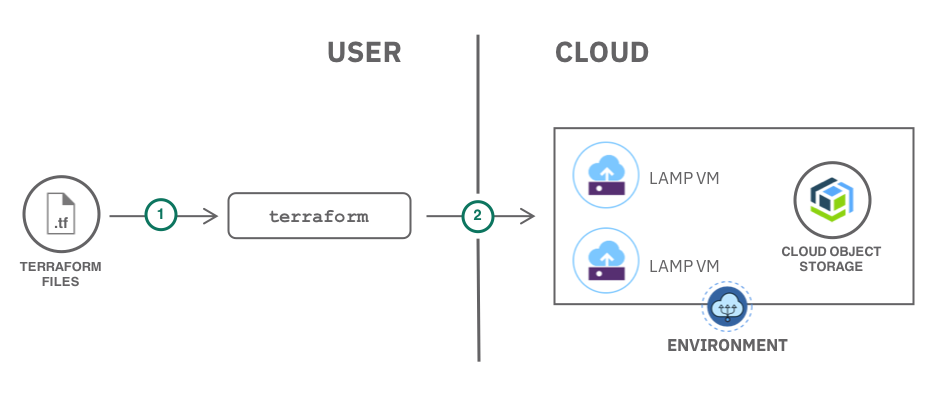
You can explore these alternative software stacks in greater depth using online resource. The LAMP stack Wiki is a great place to start:

**Components**

LAMP stands for Linux, Apache, MySQL, and PHP. Together, they provide a proven set of software for delivering high-performance web applications. Each component contributes essential capabilities to the stack:

* **Linux: The operating system.** Linux is a free and open source operating system (OS) that has been around since the mid-1990s. Today, it has an extensive worldwide user base that extends across industries. Linux is popular in part because it offers more flexibility and configuration options than some other operating systems.
* **Apache: The web server.** The Apache web server processes requests and serves up web assets via HTTP so that the application is accessible to anyone in the public domain over a simple web URL. Developed and maintained by an open community, Apache is a mature, feature-rich server that runs a large share of the websites currently on the internet.
* **MySQL: The database.** MySQL is an open source [relational database management system](https://www.ibm.com/cloud/learn/relational-databases) for storing application data. With My SQL, you can store all your information in a format that is easily queried with the SQL language. SQL is a great choice if you are dealing with a business domain that is well structured, and you want to translate that structure into the backend. MySQL is suitable for running even large and complex sites. See "[SQL vs. NoSQL Databases: What's the Difference?](https://www.ibm.com/cloud/blog/sql-vs-nosql)" for more information on SQL and NoSQL databases.
* **PHP: The programming language.** The PHP open source scripting language works with Apache to help you create dynamic web pages. You cannot use HTML to perform dynamic processes such as pulling data out of a database. To provide this type of functionality, you simply drop PHP code into the parts of a page that you want to be dynamic.

PHP is designed for efficiency. It makes programming easier—and a bit more fun—by allowing you to write new code, hit refresh, and immediately see the resulting changes without the need for compiling. If you prefer, you can swap out PHP in favor of Perl or the increasingly popular Python language.



LAMP architecture

LAMP has a classic layered architecture, with Linux at the lowest level. The next layer is Apache and MySQL, followed by PHP. Although PHP is nominally at the top or presentation layer, the PHP component sits inside Apache.

**How the elements work together**

A high-level look at the LAMP stack order of execution shows how the elements interoperate. The process starts when the Apache web server receives requests for web pages from a user’s browser. If the request is for a PHP file, Apache passes the request to PHP, which loads the file and executes the code contained in the file. PHP also communicates with MySQL to fetch any data referenced in the code.

PHP then uses the code in the file and the data from the database to create the HTML that browsers require to display web pages. The LAMP stack is efficient at handling not only static web pages, but also dynamic pages where the content may change each time it is loaded depending on the date, time, user identity and other factors.

After running the file code, PHP then passes the resulting data back to the Apache web server to send to the browser. It can also store this new data in MySQL. And of course, all of these operations are enabled by the Linux operating system running at the base of the stack.

**Flexibility**

Although LAMP uses Linux as the OS, you can use the other components with an alternative OS to meet your specific needs. For example, there is a WAMP stack, which uses Microsoft Windows; MAMP with the Mac OS; and even WIMP, using Windows and the Internet Information Services webserver from Microsoft.

Because LAMP is all open source and non-proprietary, you can avoid lock-in. You have the flexibility to select the right components for specific projects or business requirements.

LAMP offers flexibility in other ways as well. Apache is modular in design, and you will find there are existing, customizable modules available for many different extensions. These modules range from support for other languages to authentication capabilities.

Another advantage of LAMP is its secure architecture and well-established encryption practices that have been proven in the enterprise.

**Efficiency**

LAMP can help you reduce development time. Because LAMP is an open source stack that has been available for more than a decade, there is today a substantial LAMP ecosystem. You can build on what other people have done in the past and make it your own. Work within an Apache module that gets you 80% of the way there, customize the last 20%, and save considerable time as a result.