

1.3: Web Servers

While we could simply focus on how to create web pages and websites, none of this is possible without the underlying hardware and software components that support the pages we create. Examining what these components are and how they interact helps us understand what our server is capable of.

The diagram below represents the basic elements of a web server. Hardware, an operating system, and an http server comprise the bare necessities. The addition of a database and scripting language extend a server's capabilities and are utilized in most servers as well.

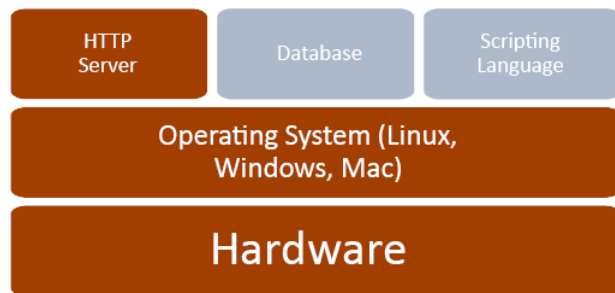
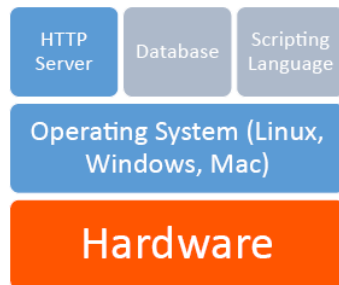


Figure 1.3.1: Web Server Software Structure

Hardware



The mention of phrases like data center, hosting provider, or even big name companies like Microsoft and Google can invoke mental images of large, sterile rooms full of tall racks of hardware with blinking lights and a maze of wires. Those more familiar with such rooms will also know the chill resulting from the heavily air conditioned atmosphere and droning whir of fans that typically accompany them. This, however, is not a requirement nor an accurate portrayal of a great deal of servers connected to the Internet. With the addition of the right software (assuming you are consuming this text digitally), the device you are using to read this with could become an internet connected server. While it would not sustain the demands made of domains like Amazon.com or MSN.com, you would be able to perform the basic actions of a server with most of today's devices.

Even though we have reached this point, it is difficult to forget the mental picture conjured by the thoughts of the data center. In the current “traditional” model, thin, physically compact servers are stacked vertically. These are referred to as rack mount hardware. Many rack mount systems today contain hardware similar to what we have in our desktops, despite the difference in appearance.

A number of companies, including Google, Yahoo, and Facebook, are looking to reinvent this concept. Google for instance has already used custom-built servers in parts of its network in an effort to improve efficiency and reduce costs. One implementation they have tried proved so efficient that they were able to eliminate large power backup units by placing a 9 volt battery in each server—giving it enough emergency power to keep running until the building's backup power source could kick in. They have also experimented with alternative cooling methods like using water from retention ponds, or placing datacenters where they can take advantage of natural resources like sea water for cooling or wind and solar for energy.

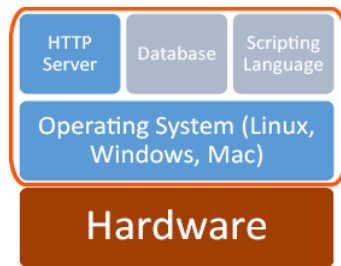
Additional Notes

Take note! While all of the programs we refer to in our LAMP stack have free, open source versions, not all uses may be covered by those licenses (using them for study and research purposes is covered).

Even small, low powered devices are finding demand as servers in part to enable the [Internet of Things](#). Devices like the [Raspberry Pi12](#) and an explosion of similar products like “android sticks” can be purchased for as little as \$25 USD. These small, “just-

enough-power” devices are used to connect data from the environment or other devices to the Internet, leaving the data center behind and living instead at the source of the data itself.

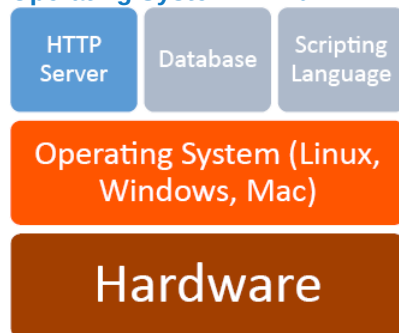
Software



A typical web server today contains four elements in addition to the physical hardware. These are the operating system, web server, a database and a scripting language. One of the most popular combinations of these systems has been abbreviated to LAMP, standing for Linux, Apache, MySQL, and PHP, named in the same order. There are many combinations of solutions that meet these features, resulting in a number of variations of the acronym, such as WAMP for Windows, Apache, MySQL, PHP or MAMP, identical with exception of Mac (or, rightfully, a Macintosh developed operating system). Among the plethora of combinations, the use of LAMP prevails as the catch all reference to a server with these types of services.

All that is ultimately required to convey static pages to an end user are the operating system and HTTP server, the first half of the WAMP acronym. The balance adds the capability for interactivity and for the information to change based on the result of user interactions.

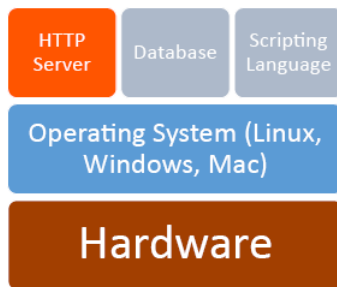
Operating System—Linux



Your operating system is what allows you to interact with the applications and hardware that make up your computer. It facilitates resource allocation to your applications, and communication between hardware and software. Typically, operating systems for servers fall under three categories: Linux based, Windows based, and Mac based. Within each of these categories are more options, such as various version of Mac and Windows operating systems, and the wide variety of Linux operating systems. We will utilize Linux, the predominant choice.

Developed by Linus Torvalds in the early 1990s while he was a student, Linux was created so Linus could access UNIX systems at his university without relying on an operating system. As his project became more robust, he decided to share it with others, seeking input but believing it would remain a more personal endeavor. What he could not have predicted was the community that would come together and participate in helping shape it into what is today. As the basis of a large number of Linux-based operating systems (or “flavors” of Linux), the Linux core can be found around the world, even in the server rooms of its competitors like Microsoft.

HTTP Server—Apache



Apache is an open source web server originally developed for UNIX systems. Now supported on most platforms including UNIX, Linux, Windows, and Mac, Apache is one of the most utilized server applications. First developed in 1995, Apache follows a similar open source approach as Linux, allowing users to expand on the software and contribute to the community of users. The user group around Apache developed The Apache Foundation, which maintains a library of solutions for web services.

In a web server, Apache serves as the HTTP component, which compiles the results from scripting languages, databases, and HTML files to generate content that is sent to the user. Apache (or any web service) will track which files on the server do and do not belong to the website, and also controls what options are available to the end user through its configuration files.

Apache and other HTTP servers allow us to share our webpages, scripts, and files with our end users. Any output from our database and scripting languages is turned into HTML output that the client's browser displays as our webpage. While we can view HTML and JavaScript files on a computer that is not a webserver, we need an http server to view them as a destination on a network.

Configuration Files

When we create a new system, settings may not be exactly as we want them, or the time may come where we want to add, remove, or change something about our server. To do this, we will need to edit the configuration files that control the different pieces of our system. Our actual web server config file is called `httpd.conf`. For PHP settings, we need to refer to `php.ini`, and for MySQL we refer to `my.cnf`. These files may be located in different places depending on the operating system, and the version in use, so it is best to use your system's file search tools to find where they are on your machine. Configuration (or setting) files are typically a plain text format file with one setting on each line with comments near each value describing the setting's use. These files will also use the same commenting delimiter for their notes to enable or disable individual settings. Typically the delimiter used is a semi colon ; or pound sign #.

If you want to change settings about your server itself such as the port it listens on, what folder it looks for files in, its name, or other related features, look to the `httpd.conf` file. From the `php.in` file you can control elements like which modules are installed and enabled for your system, how much data scripts are allowed to consume, and more. Similarly your MySQL config file determines what port it listens on, which user it runs as on your server, what your admin account's credentials are, and more.

Changes to these files typically require you to restart your web server (in our case, for apache or PHP changes), or at least the service that you are changing (in our case, MySQL changes). This can be done using the control panel if you are using a combination program like Wamp 2, or by using your operating system's service tools or by using system commands at a command prompt. Restarting Wamp 2 in a GUI operating system like Windows can be done by right clicking on Wamp's icon in the tray. In a Cent OS server, the same effect can be achieved by typing "service httpd restart." If all else fails, you can always physically restart the machine (referred to as "bouncing"), but this is something you will want to avoid on a live system as it will cause a much longer period of down time.

If you use installer packages, or a combo installer like Wamp 2, you will probably get by initially without making any changes to these files. Binary installers however will not know where or how to make changes to config files and you will need to follow the instructions to edit these files by hand to integrate all of your elements.

Why would I use a combination other than Linux, Apache, My SQL, and PHP?

Given the popularity of this particular combination of four, it is easy to wonder why it has not simply become *the* system. However, needs and preferences may change why a particular approach is selected. Perhaps you are in an all Windows environment and feel more comfortable with a Windows operating system. Maybe your data is already available in a flat file or XML format and you want a database that can use XML files, like [MongoDB](#).¹³ Or, you might prefer the approach and packages available in Python to

those found in PHP. Each system has its particular strengths and weaknesses, and should be chosen based on the needs of the project.

Open Source

At this point, you have come across many references to terms like free, free to edit, and open source throughout the text. In fact, all of the elements in our example LAMP are free, open source solutions. Open source means the provider of the software allows the end user access to the actual code of their software, allowing the end user to make changes anywhere in the program.

This differs from traditional software where you own a copy or license to use the program, but cannot extend or change elements of the program beyond what the developer allows. An executable in Windows for example is closed source. You cannot open the executable to read its code or make changes. If you wanted to change the program, the developer would have to provide you with the files used to create it (called source code) so you could make changes and compile your own, modified, executable program.

Open source is growing in popularity but the concept has existed for quite some time. Recently, larger governments have begun to embrace free, open source solutions as a means to reduce costs and achieve modifications that customize programs to fit their needs. Historically open source was viewed as a security risk as anyone could submit changes to the project, and it was feared that vulnerabilities or malicious code would be inserted. In fact, with so many users able to view and modify the files, it has actually made those with malicious intent less able to hide their modifications (sometimes called the “many eyes” approach to reliability). Development time has also been reduced as the community of developers on a popular open source project can greatly exceed that of a closed source solution with limited development staff.

A popular acronym referring to these projects is FOSS—Free, Open Source Software. As not all open source programs are free in terms of purchasing or licensing, FOSS indicates solutions that are free of costs as well as free to change. These solutions may be developed entirely by a community of volunteers, or may come from a commercial company with developers dedicated to the project. While it is odd to think of a company giving away its creation for free, these companies generate revenue by building advertising into their software or offering premium services such as product support or contracting with clients to customize the product. Many companies will also offer only some tools as open source alongside other products they sell, or offer a “freemium” model where the open sourced platform contains most of the features of their software. Here, additional features or add-ons beyond the open source package carry additional licensing and costs.

FTP

While not included in LAMP acronyms, another important element to note is the existence of a file transfer protocol (FTP) server. As you typically will want to perform development activities on private server before editing your live server, you will need a mechanism that allows you to move files between the two. FTP is designed for moving files between systems, allowing you to synchronize items when you are ready. In addition to an FTP server, you will also likely want an FTP client application for the machine(s) that contain the files you want to move. The client allows you to see files in both locations and interact with them to determine which file is moved to which machine. There are a number of free file transfer programs available, some of which can even be integrated into browsers like Chrome by using browser extensions.

1.3: Web Servers is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- **1.3: Web Servers** by [Michael Mendez](#) has no license indicated.