Generally, a configuration is the arrangement - or the process of making the arrangement - of the parts that make up a whole.

In installing hardware and software, configuration is sometimes the methodical process of defining options that are provided.

**Apache Directives:**

The main purpose of using the Apache Web server is its flexibility. One of the reasons for this flexibility is the availability of the directives and modules in the Apache configuration file. Below is the list of the directives and modules that are most frequently used. These modules and directives are the ones that all Apache administrators should know well and be able to implement to get the most flexibility and power from their Apache server.

1. **Mod\_rewrite –**

mod\_rewrite is the rule-based rewrite engine that allows Apache to rewrite requested URLs. This module basically catches incoming URL requests and rewrites them as needed. These rewrites can be according to the needs of a specific server application, the needs of a specific domain, or many other, varied requirements of a server, system, or company. This module also allows you to use more user-friendly URLs. So instead of having to type *http://this.is.my.domain/this\_is\_my\_address?=id.4567*, your users could just type *http://this.is.my.domain/address*. This module is incredibly complex and should be well researched before implementing.

1. **Alias** –

Alias is one of the must-use directives, as it allows you to point your Web server to directories outside your document root. Once set up correctly, any URL ending in the alias will automatically resolve to the path set in the alias. So you could take a folder, say, /home/sites/docs, which wouldn’t normally be accessible by Apache, and make it such that a user could just go to http://this.is.my.domain/docs and see everything within that folder.

1. **AddType –**

If you plan on serving up .php files, you’ll need to add the MIME in your configuration file; otherwise, Apache will just display the code for .php files in the browser. To get Apache to actually execute the files, the MIME type has to be set up — which is done with the help of the AddType directive.

1. **AddHandler –**

Where AddType configures MIMEs, AddHandler configures extensions. This allows you to define .php, .phtml as both handlers for the php MIME. If you find out your server isn’t executing .php files properly, make sure you have this directive set up in the form AddHandler application along with the AddType directive.

1. **VirtualHost –**

VirtualHost allows you to create multiple virtual hosts on a single Apache server. This directive accepts most all other container directives. If you’re using Apache2, these virtual hosts will be defined within the /etc/apache2/sites-available directory. You define each virtual host separately within that directory using the VirtualHost directive and then enable the site.

1. **DocumentRoot –**

DocumentRoot defines the document root of the server. Typically this will be /var/www on Apache2 and some servers can be set to /etc/httpd. You can set this to whatever you like, but make sure the directory configured has the right permissions so that the Web server user has full permission to access the directory. Change this directive only if you know exactly what you’re doing.

1. **KeepAlive –**

KeepAlive is an important directive. It defines whether a server allows more than one request per connection. If used properly, KeepAlive can prevent any one client from consuming too much of a server’s resources. By default, KeepAlive is set to off, which prevents the server from becoming too busy. If you do enable it, use the related KeepAliveTimeout directive and set it to a low number.

1. **Listen –**

Listen is the directive used to define the port Apache listens on. By default, Apache listens on port 80, but you can define this to a nonstandard port if needed. There is one warning to this. If you change the port to 1024, the Apache daemon can be started only by the root user. You can also use Listen to define specific addresses that the server will accept connections from.

1. **LoadModule –**

LoadModule is the directive used to inform the Apache server of a module to be loaded. Tons of modules are included in a default Apache installation and more can be found. But not all modules are loaded by default. If you look in the Apache configuration directory, /etc/apache2, for example, you’ll find a subfolder that contains all the available modules. It will probably require some Google time to determine exactly what each module does. Once you’re sure you want to load a module, do so with the LoadModule directive.

1. **Options –**

Options allow you to define specific options to be available for a defined directory. This is an incredibly handy directive to have at your fingertips, as it allows you to fine-tune on a per-directory basis. Some of the more useful options available are FollowSymLinks, Indexes, ExecCGI, and Includes. Now with Options you can use to tell Apache to add the option or to remove an option.

**Use caution**

Apache is an amazingly powerful tool that can be extended in many ways. With the help of the right directives and modules, you can make it do just about anything you need. But use directives and modules with caution otherwise, you run the risk of configuring a nonfunctioning server. Before you include any of these modules and/or directives in your setup, make sure you know how they are properly used so you can get the most out of them.