digitalocean.com

How To Configure the Apache Web Server on an Ubuntu or Debian VPS | DigitalOcean

By: Justin Ellingwood

What is Apache?

Apache is the most popular web server on the internet. It is used to serve more than half of all active websites.

Although there are many viable web servers that will serve your content, it is helpful to understand how Apache works because of its ubiquity.

In this article, we will examine some general configuration files and options that can be controlled within them. This article will follow the Ubuntu/Debian layout of Apache files, which is different from how other distributions build the configuration hierarchy.

How to Install Apache on Ubuntu and Debian

If you do not already have Apache installed, you can do so now by issuing the following commands:

```
sudo apt-get update
sudo apt-get install apache2
```

This is all that is necessary to have a working web server. If you visit your VPS's IP address in a web browser, you will get the default Apache index page:

```
your_domain_name_or_ip_address
```

```
It works!
This is the default web page for this server.
The web server software is running but no
content has been added, yet.
```

The Apache File Hierarchy in Ubuntu and Debian

On Ubuntu and Debian, Apache keeps its main configuration files within the "/etc/apache2" folder:

```
cd /etc/apache2
ls -F
```

```
apache2.conf  envvars     magic
mods-enabled/  sites-available/
conf.d/       httpd.conf  mods-available/
ports.conf     sites-enabled/
```

There are a number of plain text files and some sub-directories in this directory. These are some of the more useful locations to be familiar with:

- **apache2.conf**: This is the main configuration file for the server. Almost all configuration can be done from within this file, although it is recommended to use separate, designated files for simplicity. This file will configure defaults and be the central point of access for the server to read configuration details.

- **ports.conf**: This file is used to specify the ports that virtual hosts should listen on. Be sure to check that this file is correct if you are configuring SSL.

- **conf.d**/: This directory is used for controlling specific aspects of the Apache configuration. For example, it is often used to define SSL configuration and default security choices.

- **sites-available**/: This directory contains all of the virtual host files that define different web sites. These will establish which content gets served for which requests. These are available configurations, not active configurations.

- **sites-enabled**/: This directory establishes which virtual host definitions are actually being used. Usually, this directory consists of symbolic links to files defined in the "sites-available" directory.

- **mods-[enabled,available]**/: These directories are similar in function to the sites directories, but they define modules that can be optionally loaded instead.

As you can see, Apache configuration does not take place in a

single monolithic file, but instead happens through a modular design where new files can be added and modified as needed.

Looking at the Apache2.conf File

The main configuration details for your Apache server are held in the "/etc/apache2/apache2.conf" file.

This file is divided into three main sections: configuration for the global Apache server process, configuration for the default server, and configuration of Virtual Hosts.

In Ubuntu and Debian, the majority of the file is for global definitions, and the configuration of the default server and virtual hosts is handled at the end, by using the "Include ..." directive.

The "Include" directive allows Apache to read other configuration files into the current file at the location that the statement appears. The result is that Apache dynamically generates an overarching configuration file on startup.

If you scroll to the bottom of the file, there are a number of different "Include" statements. These load module definitions, the ports.conf document, the specific configuration files in the "conf.d/" directory, and finally, the Virtual Host definitions in the "sites-enabled/" directory.

We will focus on the first part of the file to learn how Apache defines its global settings.

Global Configuration Section

This section is used to configure some options that control how Apache works as a whole.

There are some interesting options you may want to modify in this section:

Timeout

By default, this parameter is set to "300", which means that the server has a maximum of 300 seconds to fulfill each request.

This is probably too high for most set ups and can safely be dropped to something between 30 and 60 seconds.

KeepAlive

This option, if set to "On", will allow each connection to remain open to handle multiple requests from the same client.

If this is set to "Off", each request will have to establish a new connection, which can result in significant overhead depending on your setup and traffic situation.

MaxKeepAliveRequests

This controls how many separate request each connection will handle before dying. Keeping this number high will allow Apache to serve content to each client more effectively.

Setting this value to 0 will allow Apache to serve an unlimited amount of request for each connection.

KeepAliveTimeout

This setting specifies how long to wait for the next request after finishing the last one. If the timeout threshold is reached, then the connection will die.

This just means that the next time content is requested, the server will establish a new connection to handle the request for the content that make up the page the client is visiting.

MPM Configuration

The next section specifies the configuration of the MPM (Multi-Processing Module) options. You can cross-reference which section your Apache installation was compiled with by exiting into the terminal and typing:

```
apache2 -l

Compiled in modules:
  core.c
  mod_log_config.c
  mod_logio.c
  prefork.c
  http_core.c
  mod_so.c
```

As you can see, in this server, "prefork.c" is a module that was compiled in and is also in the "apache2.conf" file. Your installation may have multiple to choose from, but only one can be selected.

You can adjust the configuration of the prefork MPM in the appropriate section.

Exploring the Default Virtual Host File

The default Virtual Host declaration can be found in a file called "default" in the "sites-available" directory.

We can learn about the general format of a Virtual Host file by examining this file. Open the file with the following command:

```
sudo nano /etc/apache2/sites-available/default
```

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks
MultiViews
                AllowOverride None
                Order allow,deny
```

```
                allow from all
        </Directory>
. . .
```

The default Virtual Host is configured to handle any request on port 80, the standard http port. This is defined in the declaration header where it says "*:80", meaning port 80 on any interface.

This does not mean that it will necessarily handle each request to the server on this port however. Apache uses the most specific Virtual Host definition that matches the request. This means that if there was a more specific definition, it could supersede this definition.

Virtual Host Top Level Configuration

These options are set within the Virtual Host definition outside of any other lower level sub-declaration. They apply to the whole Virtual Host.

The "ServerAdmin" option specifies a contact email that should be used when there are server problems.

This can be inserted into an error page if you have "ServerSignature" set to "Email" in the "/etc/apache2/conf.d /security" file, so make sure you are willing to receive the mail if you adjust that setting.

If we were using this as a template for other Virtual Host definitions, we would want to add a "ServerName" definition that specifies the

domain name or IP address that this request should handle. This is the option that would add specificity to the Virtual Host, allowing it to trump the default definition if it matches the ServerName value.

You can also make the Virtual Host apply to more than one name by using the "ServerAlias" definition. This provide alternate paths to get to the same content. A good use-case for this is adding the same domain, preceded by "www".

The "DocumentRoot" option specifies where the content that is requested for this Virtual Host will be located. The default Virtual Host is set up to serve content out of the "/var/www" directory on Ubuntu.

Directory Definitions

Within the Virtual Host definition, there are definitions for how the server handles different directories within the file system. Apache will apply all of these directions in order from shortest to longest, so there is again a chance to override previous options.

The first directory definition applies rules for the "/", or root, directory. This will provide the baseline configuration for your Virtual Host, as it applies to all files served on the file system.

By default, Ubuntu does not set up any access restrictions to the filesystem. Apache recommends that you add some default access restrictions. You can modify this like so:

```
<Directory />
```

```
        Options FollowSymLinks
        AllowOverride None
        Order Deny,Allow
        Deny from All
</Directory>
```

This will deny access to all content unless specified otherwise in subsequent directory definitions.

The next directory definition is for the document root, so it specifies the "allow from all" option that overrides the "/" option for this directory.

The "AllowOverride" option is used to decide whether an ".htaccess" file can override settings if it is placed in the content directory. This is not allowed by default, but can be useful to enable in a variety of circumstances.

Alias and ScriptAlias Statements

Directory definitions are sometimes preceded by "Alias" or "ScriptAlias" statements. Alias maps a url path to a directory path.

ScriptAlias operates in the same way, but is used to define directories that will have executable components in them.

For instance, this line in a Virtual Host that handles request to "example.com" would allow access to content within "/path /to/content/" by navigating to "example.com/content/":

```
Alias /content/ /path/to/content/
```

Following the alias, you should remember to define the directory with access privileges as discussed in the previous section.

Enabling Sites and Modules in Apache

Once you have a Virtual Host file that meets your requirements, you can use the tools included with Apache to transition them into live sites.

To automatically create a symbolic link in the "sites-enabled" directory to an existing file in the "sites-available" directory, issue the following command:

```
sudo a2ensite virtual_host_file_name
```

After enabling a site, issue the following command to tell Apache to re-read its configuration files, allowing the change to propagate:

```
sudo service apache2 reload
```

There is also a companion command for disabling a Virtual Host. It operates by removing the symbolic link from the "sites-enabled" directory:

```
sudo a2dissite virtual_host_file_name
```

Again, reload the configuration to make the change happen:

```
sudo service apache2 reload
```

Modules can be enabled or disabled by using the "a2enmod" and "a2dismod" commands respectively. They work in the same way as the "site" versions of these commands.

Remember to reload your configuration changes after modules have been enabled or disabled as well.

Conclusion

We have gone over some basic Apache configuration files. Apache is versatile and very modular, so configuration needs will be different depending on your setup.

You should have a good understanding of what the main configuration files are used for and how they interact with each other. If you need to know about specific configuration options, the provided files are well commented and Apache provides excellent documentation.

Hopefully, the configuration files will not be as intimidating now, and you feel more comfortable experimenting and modifying to suit your needs.

By Justin Ellingwood